Delft University of Technology

Efficient Methods for Spectral Geometry Processing

Nasikun, A.

DOI
10.4233/uuid:2bd86c48-8b81-4a66-838f-c85bdb7db334

Publication date
2022

Document Version
Final published version

Citation (APA)
Nasikun, A. (2022). *Efficient Methods for Spectral Geometry Processing*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:2bd86c48-8b81-4a66-838f-c85bdb7db334

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

EFFICIENT METHODS
FOR SPECTRAL GEOMETRY PROCESSING

EFFICIENT METHODS FOR SPECTRAL GEOMETRY PROCESSING

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
Chair of the Board for Doctorates,
to be defended publicly on
Monday, 21 March 2022 at 10.00

by

Ahmad NASIKUN

Master of Science in Electrical Engineering and Computer Science,
Seoul National University (SNU), Seoul, South Korea,
Born in Jepara, Indonesia.

This dissertation has been approved by the promotor.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr. E. Eisemann,	Delft University of Technology, promotor
Dr. K.A. Hildebrandt,	Delft University of Technology, co-promotor
Independent members:	
Prof.dr.ir. C. Vuik,	Delft University of Technology
Prof.dr. M. Botsch,	Technical University Dortmund, Germany
Prof.dr. D. Bommes,	University of Bern, Switzerland
Dr. J. Digne,	LIRIS, University of Lyon, France
Dr. A. Vaxman,	Utrecht University
Prof.dr. G. Smaragdakis,	Delft University of Technology, reserve member

Keywords: geometry processing, spectral methods, model order reduction, multigrid, Laplace–Beltrami operator, vector fields

Printed by: Gildeprint

Front & Back: by Ahmad Nasikun and Alfina Dewi.

Copyright © 2022 by Ahmad Nasikun

ISBN 000-00-0000-000-0

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

To Bubu Fina, Aziz, and Ziza,
Thank you for always being on my side
going through this wonderful journey together.

Ayah Nasikun

CONTENTS

Summary **ix**

Samenvatting **xi**

1 Introduction **1**

2 Fast Approximation of Laplace–Beltrami Eigenproblems **9**
- 2.1 Introduction . . . 10
- 2.2 Related Work . . . 11
- 2.3 Background: Laplace–Beltrami Eigenproblem . . . 13
- 2.4 Fast Approximation Algorithm . . . 15
- 2.5 Experiments . . . 21
- 2.6 Applications . . . 25
- 2.7 Conclusions. . . . 32

Appendix **41**
- 2.A Choice of basis functions . . . 41
- 2.B Eigenfunctions and edge flips. . . . 43
- 2.C Comparison to mesh coarsening . . . 44

3 Locally Supported Tangential Vector, n-Vector, and Tensor Fields **47**
- 3.1 Introduction . . . 48
- 3.2 Related work . . . 49
- 3.3 Laplace operators . . . 50
- 3.4 Spaces of locally supported fields . . . 52
- 3.5 Subspace methods . . . 56
- 3.6 Experiments . . . 58
- 3.7 Comparisons . . . 63
- 3.8 Applications . . . 68
- 3.9 Conclusion . . . 73

Appendix **81**
- 3.A Construction of the tensor field Laplacian . . . 81

4 The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems **83**
- 4.1 Introduction . . . 84
- 4.2 Related Work . . . 85
- 4.3 Background . . . 87
  - 4.3.1 Laplace–Beltrami eigenproblem . . . 87
  - 4.3.2 Subspace iteration method (SIM) . . . 88

4.4 Hierarchical Subspace Iteration Method . . . 90
    4.4.1 Hierarchy construction . . . 90
    4.4.2 Hierarchical Solver . . . 94
4.5 Experiments . . . 95
4.6 Comparisons . . . 102
4.7 Conclusion . . . 106

Appendix 115
4.A Justification of Design Choices . . . 115
    4.A.1 Distance computation . . . 115
    4.A.2 Sampling method . . . 115
    4.A.3 Convergence tolerance. . . . 119
4.B Comparisons . . . 120
    4.B.1 Lanczos and preconditioned eigensolver . . . 120
    4.B.2 Fast Approximation . . . 120
4.C Generalization . . . 122
    4.C.1 Hamiltonian operators. . . . 122
4.D Applications . . . 123
    4.D.1 Shape Signatures. . . . 123
    4.D.2 Projection . . . 124

5 Conclusion 129
5.1 Limitation and future work . . . 130

Acknowledgements 131

Curriculum Vitæ 135

List of Publications 137

SUMMARY

Research in geometry processing concerns the design of algorithms and mathematical models for the analysis and manipulation of geometric data. Examples of its applications are shape projection (*e.g.* smoothing and filtering), shape correspondence (*e.g.* functional maps), shape descriptors (*e.g.* heat and wave kernel signatures), segmentation, and surface parameterization. A set of tools that have proven to be useful for solving such tasks are spectral methods. In general, spectral methods solve geometry processing problems by taking the benefit of the spectra and the eigenfunctions of the Laplacian operator defined on a surface mesh. This allows us to extend the notion of Fourier analysis from signal and image processing to surface processing, a theoretically sound and well-researched concept. In practice, the decomposition of the Laplacian operator into a diagonal matrix of eigenvalues and a rectangular matrix of eigenvectors enables efficient treatment of a broad range of geometry processing problems.

A main adversity in spectral geometry processing is the expensive computational cost attached to the eigendecomposition of the Laplacian operator, before we can use the spectra and the eigenfunctions for the applications. Since analytical solutions are not known, one needs to opt for a numerical method to solve the eigenvalue problem. It is a numerically expensive computation, especially for a complex mesh. Another challenge comes from the storage requirement. Considering that the Laplace–Beltrami operator has global support, it takes a dense matrix to represent the eigenvectors. Therefore, the memory requirement for saving the eigenbasis can be high, particularly when a large number of eigenfunctions need to be stored. These challenges hinder the use of spectral methods for geometry processing applications.

In this thesis, we introduce new methods addressing the aforementioned challenges. In Chapter 2, we propose a fast algorithm that allows for approximating the smallest eigenvalues and the corresponding eigenvectors of the Laplace–Beltrami operator in just a fraction of the time needed to solve the original eigenvalue problem. We construct subspaces of the space of all functions that include low frequency functions and restrict the solution of the eigenproblem to the subspace. It enables the fast approximation of the eigenproblem, independent of the size of the original problem. Our novel scheme also enables significantly more efficient storage of the approximated eigenfunctions. We show that the approximated spectra are close to the reference spectra and that the fast approximation method benefits geometry processing applications, such as shape classification, geodesic distance computation, shape projection (*e.g.* filtering), and vibration modes of deformable objects.

We consider localized eigenfields of the Hodge–Laplacian, which serve as a sparse basis for the efficient design and processing of tangential fields, in Chapter 3. The basis spans subspaces of the spaces of tangential vector, n-vector, and tensor fields on a surface mesh. Restricting the design and processing of tangential fields to the subspace allows us to decouple the degrees of freedom we use for design and processing tasks from the complexity of the mesh representation. The construction is scalable, so we can efficiently compute and store subspaces for large meshes. We evaluate the performance of the novel method on various modeling and processing tasks in vector fields (fur design), n-vector fields (n-field design and hatching/line-art design), and tensor fields (curvature fields smoothing) and show that the computation time decreases up to two orders of magnitude compared to that of the original problem.

Chapter 4 introduces a novel multigrid method for numerically solving the Laplace–Beltrami eigenproblems on a surface mesh. Our new technique, the Hierarchical Subspace Iteration Method (HSIM), works on a hierarchy of nested vector spaces, in which the solution of the coarser level is used as an initial solution on the finer level. We construct the coarsest level such that the eigenproblems can be solved efficiently using a dense eigensolver. On every level, the prolongation operator maps the solution from the coarser to the finer level. The result then can be used as an initialization for subspace iterations to approximate the eigenpairs. This approach significantly reduces the number of iterations in the finest level, compared to the non-hierarchical subspace iteration method. We show that HSIM outperforms the Locally Optimal Block Preconditioned Conjugate Gradient method and the state-of-the-art Lanczos-based eigensolvers, such as Matlab's $eigs$, Manifold Harmonics, and SpectrA.

In summary, each of the chapters in this thesis proposes efficient algorithms for computing the eigendecompositions of Laplace–Beltrami and Hodge–Laplace operators, mainly using model order reduction and multigrid approaches. These methods reduce computational costs (Chapter 1-3) and storage requirements (Chapter 1-2) for the spectral processing of scalar functions and tangential fields on surface meshes.

SAMENVATTING

Onderzoek op het gebied van geometrieverwerking betreft het ontwerp van algoritmen en modellen voor de analyse en bewerking van geometrische gegevens. Voorbeelden van toepassingen hiervan zijn projectie van vormen (b.v. egaliseren en filteren), vormcorrespondentie (b.v. functionele kartering), vormdescriptoren (b.v. warmte- en golfkernsignaturen), segmentatie, en oppervlakteparametrisering. Een set hulpmiddelen die nuttig zijn gebleken voor het oplossen van dergelijke taken worden spectrale methoden genoemd. In het algemeen lossen spectrale methoden geometrieverwerkingsproblemen op door gebruik te maken van zowel spectra als eigenfuncties van de Laplace-Beltrami operator, gedefinieerd over een oppervlaktemaas. Deze methoden stellen ons in staat om Fourieranalyse uit te breiden tot oppervlaktes, een theoretisch verantwoord en goed onderzocht concept. In de praktijk maakt de ontbinding van de Laplace-Beltrami operator in een diagonaalmatrix van eigenwaarden en een orthogonale matrix van eigenvectoren een efficiënte behandeling mogelijk van een uitgebreide selectie aan problemen binnen de geometrieverwerking.

Een belangrijke uitdaging bij de verwerking van spectrale geometrie zijn de hoge computationele kosten die verbonden zijn aan de eigendecompositie van de Laplaciaanse operator, alvorens we de spectra en de eigenfunctie voor de toepassing kunnen gebruiken. Aangezien analytische oplossingen niet bekend zijn, moet men opteren voor dure numerieke oplossingen. Een ander probleem heeft te maken met de vereiste opslagruimte. Aangezien de Laplace-Beltrami operator een globale ondersteuning heeft, zijn eigenfuncties een dichtbezette vector, wat een dichtbezette matrix vereist om de eigenfuncties weer te geven. Er is dus een grote opslagcapaciteit nodig om een voldoende groot aantal eigenfuncties op te slaan. Deze uitdagingen belemmeren het gebruik van spectrale geometrie verwerking.

In dit proefschrift introduceren we nieuwe methoden om de bovengenoemde uitdagingen aan te pakken. In hoofdstuk 2 stellen we een vlot algoritme voor waarmee de kleinste eigenwaarden en de corresponderende eigenfuncties van de Laplace-Beltrami operator kunnen worden benaderd in slechts een fractie van de tijd die de huidige nieuwste methoden in het veld nodig hebben, onze methode is een orde van grootte(s) sneller. We beperken de oplossing tot de deelruimte van gladde functies, onafhankelijk van de grootte van het oorspronkelijke probleem. Dit laat het algoritme ook toe om de benaderde eigenfuncties op een veel efficiëntere manier op te slaan. We tonen experimenteel aan dat deze snelle benaderingsmethode geometrieverwerkingstoepassingen ten goede komt, zoals vormclassificatie, geodetische afstandsberekening, vormprojectie (bv. filtering), en trillingsmodi van vervormbare objecten.

In hoofdstuk 3 beschouwen we gelokaliseerde eigenvelden van de Hodge-Laplaciaan die als schaarse basis dienen voor het efficiënt ontwerpen en bewerken van tangentiële velden. De basis beslaat deelruimten van de ruimten van de gladste tangentiële vectorvelden, n-vectorvelden, en tensorvelden op oppervlakken. Door het ontwerp en de

verwerking van tangentiële velden te beperken tot de deelruimte, kunnen we de vrijheidsgraad van ontwerp- en verwerkingstaken loskoppelen van de complexiteit van de maasrepresentatie. De constructie is schaalbaar, wat een efficiënte berekening en opslag van de deelruimte voor grote mazen mogelijk maakt. We evalueren de prestaties van de nieuwe methode op verschillende modelleer- en verwerkingstaken in vectorvelden (ontwerpen van vacht), n-vectorvelden (ontwerpen van n-velden, ontwerpen van arceer-/lijnkunst), en tensorvelden (egaliseren van kromming in tensorvelden) en tonen aan dat de rekentijd aanzienlijk afneemt, ordes van grootte sneller dan het oplossen van het niet-gereduceerde systeem.

Hoofdstuk 4 introduceert een nieuwe methode om het Laplace-Beltrami eigenprobleem numeriek op te lossen. Onze nieuwe techniek, de Hiërarchische Subruimte Iteratie Methode (HSIM), werkt op een hiërarchie van geneste vectorruimten, waarbij de oplossing van het grovere niveau gebruikt wordt als initiële oplossing voor een fijner niveau, wat goede benaderingen oplevert en daardoor de iteratieve oplosser aanzienlijk sneller naar de oplossing laat convergeren. Het grofste niveau wordt zo geconstrueerd dat het efficiënt kan worden opgelost met een dichte eigenoplosser. De verlengingsoperator kan de oplossing efficiënt naar een fijner niveau leiden en deze gebruiken als initialisatie voor de deelruimte-iteratie om de eigenparen te benaderen. Deze aanpak vermindert het aantal iteraties op het fijnste niveau aanzienlijk, vergeleken met de niet-hiërarchische Subruimte Iteratie Methode (SIM).

Samenvattend stelt elk van de nieuwe benaderingen in dit proefschrift efficiënte algoritmen voor om de eigendecompositie van de Laplace–Beltrami en de Hodge–Laplacian operator te berekenen, waarbij voornamelijk gebruik gemaakt wordt van model orde reductie en multigrid benaderingen. Deze bijdragen bieden nieuwe oplossingen voor spectrale geometrie verwerking van scalaire functies en tangentiële velden op veelhoeksmazen.

1

INTRODUCTION

Geometry processing allows us to analyze and manipulate geometric data using a computer, employing concepts from applied mathematics, computer science, and engineering. The field of geometry processing encompasses a wide range of research areas that lie between 3D data acquisition and fabrication/consumption of 3D contents. It deals with the representation, reconstruction, processing, modeling, optimization, analysis, simulation, and fabrication of geometric data. Albeit being a relatively new research field, the application of geometry processing spans a broad spectrum of fields: from character animation in movies, fracture analysis in structural engineering, surface modelling in automotive design, aneurysm detection in medical study, to combustion simulation in rocket science, to name a few.

Geometry processing applications require an input of geometric data, which can be generated by capturing real-world objects using technology such as 3D laser scanning, radar, computer tomography, and magnetic resonance imaging (MRI). Another source of 3D data is models created by designers using computer-aided design (CAD) or computer animation systems. One can also synthesize geometric input via mathematical modeling. The increase in the complexity of 3D geometric data leads to significant challenges in geometry processing. It is because, in general, the complexity of the problem is positively correlated to the complexity of the model. For example, in computing geodesic distance on a surface mesh using the Heat Method [2], one needs to solve two linear systems. The number of unknowns in the linear systems corresponds to the number of vertices and faces of the surface mesh, respectively. This means that the more complex our model, the more computationally intensive the system to be solved. Such a challenge amplifies the complexity of geometry processing applications, and therefore efficient computational methods are of great necessity.

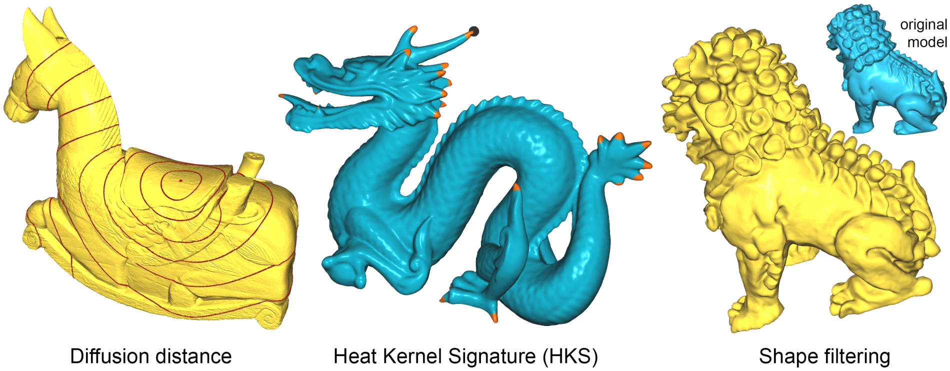


Figure 1.1: Examples of spectral methods in geometry processing: diffusion distance, heat kernel signature, and shape filtering.

An important family of methods in geometry processing is spectral methods. These methods constitute a research topic called spectral geometry processing [4, 9, 10] and are proven to successfully address a variety of problems, such as the aforementioned geodesic distance computation, point signature, mesh filtering (Fig. 1.1), compression,

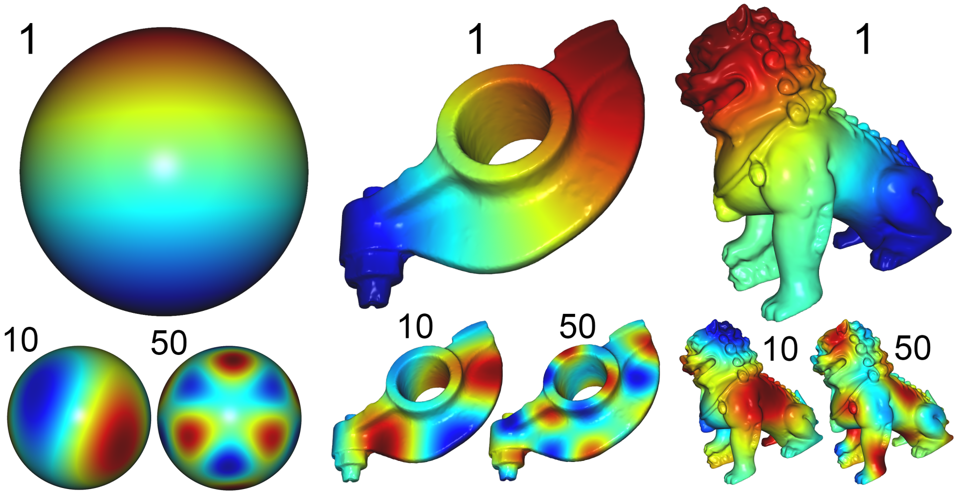


Figure 1.2: Visualization of the eigenvectors corresponding to the 1^{st}, 10^{th}, and 50^{th} smallest eigenvalues of the Laplace–Beltrami operator, computed on a sphere, the Rocker-Arm, and the Chinese Dragon models.

surface segmentation, and shape correspondence. The main ingredient of spectral methods is the use of eigenvalues and the corresponding eigenvectors of appropriately chosen linear operators. There have been different linear operators proposed in the literature. These can depend only on the intrinsic geometry of the surface or also on the embedding of the surface in space. One widely adopted operator is the Laplace–Beltrami operator, which, in the discrete setting, has cotangent values as matrix elements. Functions on a surface can be viewed as signals (from a perspective of signal processing) and the eigenbasis of the Laplace–Beltrami operator allow us to correlate a frequency spectrum to a function, extending the notion of Fourier decomposition in signal and image processing. Figure 1.2 shows examples of eigenfunctions of the Laplace–Beltrami operator corresponding to some selected eigenvalues (the 1^{st}, 10^{th}, and 50^{th} eigenvalues).

One important attribute of the Laplace–Beltrami operator is its invariance under isometric deformation of the surface mesh, meaning that the matrix representation of isometrically deformed objects is identical. This property makes the Laplace–Beltrami operator well-suited for mesh-invariant shape descriptors, both global and local descriptors. Global shape descriptors (*e.g.* the Shape-DNA [7]) allow for the clustering of similar objects and discriminating different shapes. Local shape descriptors (*e.g.* heat [8] and wave kernel signatures [1]) enable the identification of points on a surface with similar properties. In addition to its use as a shape descriptor, the isometry-invariance of the Laplace–Beltrami operator is well-suited for shape correspondence, *i.e.* finding a meaningful relation between elements of multiple objects. An example of a method for shape correspondence is functional maps [5] that uses the eigenfunctions of two objects to create a linear operator that can map functions from one surface to the other. This is useful for many applications, such as texture mapping.

A major challenge in spectral methods is the requirement to compute the lowest part of the spectrum and its frequency (generally the first 50-5000) in a precomputation. It amounts to solving an eigenvalue problem $S\Phi = M\Lambda\Phi$, where S is the matrix representing the Laplace operator and M is the mass matrix. We need to solve such a large-scale, sparse eigenproblem numerically because the closed-form solution is not known. This yields a long preprocessing time before spectral methods can be applied to solve geometry processing problems. Another challenge concerns the storage requirements. As each of the eigenfunctions of the Laplace–Beltrami operator has global support, it spans the entire surface mesh and needs to be stored as a dense vector. In consequence, it requires an extensive amount of memory to store a dense matrix that represents the eigenfunctions of the Laplace operator, particularly for a large number of eigenfunctions on a complex mesh.

In this thesis, novel methods for spectral geometry processing, in particular for the efficient computation and storage of the eigenvectors of the Laplace operator, are introduced. In Chapter 2, we present a novel scheme that enables a fast approximation of the low frequency eigenvalues and the corresponding eigenfunctions of the Laplace–Beltrami operator on a surface mesh. The underlying idea is to construct subspaces of the spaces of all functions spanning low frequency functions and to restrict the solution of the eigenproblem to the subspace. This scheme yields three benefits: (1) computing the approximated spectra is significantly more efficient than solving the original eigenvalue problems, (2) storing the approximated eigenfunctions requires considerably less memory, and (3) running the applications that utilize the reduced eigenpairs takes a noticeably shorter time. We consider this work a step forward in reducing the computational burden attached to spectral geometry processing. To show the effectiveness of our method, we demonstrate experimentally that the resulting spectra are close to the reference spectra and that spectral methods built on top of that produce similar results to those using the reference spectra.

A fast algorithm for tangential fields design and processing is introduced in Chapter 3. We propose a novel construction of locally supported tangential vector, n-vector, and tensor fields on a triangle mesh surface. Those sparse fields span subspaces of the spaces of tangential fields and serve as sparse linear bases for the design and processing of tangential data. Restricting the design and processing of tangential fields to the subspaces allows us to decouple the degrees of freedom of design and processing tasks from the complexity of the mesh representation. The construction is also scalable, which enables efficient computation and storage of the subspaces on large meshes. We evaluate the performance of the novel method for various tasks in vector field (fur design), n-vector field (n-field design and hatching/line-art design), and tensor field (curvature tensor fields smoothing) design and processing, and show that the computation can be up to two orders of magnitude faster compared to solving the unreduced systems.

We extend the fast approximation method (introduced in Chapter 2) to build the Hierarchical Subspace Iteration Method (HSIM), an efficient solver for Laplace–Beltrami eigenproblems, in Chapter 4. The main idea is to employ a multigrid approach in building the eigensolver, in which the converged solution of a lower level in the hierarchy is used as an initialization for the next level. The lowest level is constructed such that all eigenpairs can be solved using a dense eigensolver. On each level of the hierarchy, a subspace iteration method (SIM) is employed to let the initial eigenpairs converge up to a predefined accuracy. This scheme allows for a substantial reduction in the number of iterations on the finest level, compared to the standard SIM. We show that HSIM is consistently faster than state-of-the-art schemes, such as Lanczos-based eigensolvers (Matlab's $eigs$, Manifold Harmonics [9], and SpectrA [6]), the original Subspace Iteration Method, and a preconditioned eigensolver (LOBPCG [3]).

In summary, this thesis presents novel methods that enable the efficient treatment of spectral methods. When accuracy is not the main interest, our fast approximation eigensolver and sparsified eigenfields reduce the computation time by around two orders of magnitude and significantly lower the required memory. In the case that an accurate solution is needed, our hierarchical eigensolver (HSIM) outperforms existing state-of-the-art methods in solving the eigenproblem, particularly when large numbers of eigenpairs on complex meshes are sought. These are important steps towards a widespread application of spectral methods in geometry processing.

Each chapter in this thesis provides a short conclusion that highlights the contributions, mentions limitations, and suggests a number of interesting future directions to further develop the proposed methods into. The concluding Chapter 5 summarizes all chapters in this thesis in a broader context and discusses future directions of research in spectral geometry processing.

BIBLIOGRAPHY

[1] M. Aubry, U. Schlickewei, and D. Cremers. “The wave kernel signature: A quantum mechanical approach to shape analysis”. In: *ICCV*. 2011, pp. 1626–1633.

[2] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. “Geodesics in heat: A new approach to computing distance based on heat flow”. In: *ACM Transactions on Graphics (TOG)* 32.5 (2013), pp. 1–11.

[3] Andrew V Knyazev. “Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method”. In: *SIAM journal on scientific computing* 23.2 (2001), pp. 517–541.

[4] Bruno Lévy and Hao Zhang. “Spectral mesh processing”. In: *ACM SIGGRAPH ASIA Courses*. 2009, pp. 1–47.

[5] Maks Ovsjanikov et al. “Functional Maps: A Flexible Representation of Maps Between Shapes”. In: *ACM Trans. Graph.* 31.4 (2012), 30:1–30:11.

[6] Yixuan Qiu. *SpectrA: C++ Library For Large Scale Eigenvalue Problems*. https://spectralib.org/. 2015.

[7] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. “Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids”. In: *Computer-Aided Design* 38.4 (2006), pp. 342–366.

[8] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. “A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion.” In: *Computer Graphics Forum* 28.5 (2009), pp. 1383–1392.

[9] Bruno Vallet and Bruno Lévy. “Spectral Geometry Processing with Manifold Harmonics”. In: *Computer Graphics Forum* 27.2 (2008), pp. 251–260.

[10] Hao Zhang, Oliver van Kaick, and Ramsay Dyer. “Spectral Mesh Processing”. In: *Computer Graphics Forum* 29.6 (2010), pp. 1865–1894.

2

FAST APPROXIMATION OF LAPLACE–BELTRAMI EIGENPROBLEMS

There is no royal road to geometry.

Euclid

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 chapter, 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.

This chapter is based on the paper *Fast Approximation of Laplace–Beltrami Eigenproblems* published in Eurographics Computer Graphics Forum (2018) [48].

2.1. 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 chapter, 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.

2.2. RELATED WORK

Spectral mesh processing In the following, we discuss some spectral mesh processing methods. For an introduction to the topic, we refer to [73]. Vallet and Lévy [65] 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 [39] 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. [66]. Dong et al. [25] 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. [35] and Ling et al. [42]. Sharma et al. [59] and Huang et al. [36] proposed spectral methods for surface segmentation. Musialski et al. [46] 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. [61] 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 [3, 4, 11, 10].

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* [54, 53], the *diffusion distance* [47], the *global point signature* [56], the *heat kernel signature* [62], the *Auto Diffusion Function* [28] and the *wave kernel signature* [2]. 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 [13]. 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. [51] 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. [57] use functional maps between surface to analyzed difference between shapes. While functional map compute the eigenfunctions on the two shape independently, Kovnatsky et al. [40] 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 [55, 43]. For an introduction to functional maps, we refer to [50]. Spectral methods are also used in the context of *geometric deep learning* [15, 8, 24, 41, 14].

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 [31, 34], 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 [44] and the Steklov eigenvalue problem [69] have been introduced. Choukroun et al. [19, 18] 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. [45]. 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 3.6, 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* [71]. The Nyström method is used in the context of machine learning for accelerating kernel methods [26] and spectral clustering [27] as well as for approximating large scale singular value decompositions for manifold learning [63]. 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* [30]. First, a random rectangular matrix A of size $n m$, 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 [7], 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. [67] proposed a multiresolution hierarchy for the approximation of the heat kernel and used the scheme for diffusion-based feature extraction from surfaces. Patané [52] 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.

2.3. 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

$$\langle f, g\rangle_{L^2} = \int_{\Sigma} f\, g \,\mathrm{d}A \tag{2.1}$$

and

$$\langle f, g\rangle_{H_0^1} = \int_\Sigma \langle \operatorname{grad} f, \operatorname{grad} g\rangle_\Sigma \, \mathrm{d}A \tag{2.2}$$

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

$$\langle \phi, f\rangle_{H_0^1} = \lambda \langle \phi, f\rangle_{L^2} \tag{2.3}$$

holds for all $f \in H^1$. Since $\langle f, f\rangle_{H_0^1}$ 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 $\langle \phi, \phi\rangle_{H_0^1}$ 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 ith 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

$$M_{ij} = \langle \varphi_i, \varphi_j\rangle_{L^2} \qquad \text{and} \qquad S_{ij} = \langle \varphi_i, \varphi_j\rangle_{H_0^1}. \tag{2.4}$$

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 [70] and [65].

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 [1], higher-order finite elements on meshes [54, 53] or Discrete Exterior Calculus discretizations [23] 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:

$$\min_{\Phi \in \mathbb{R}^{n\times m}} \operatorname{tr}\left(\Phi^T S \Phi\right) \tag{2.5}$$

$$\text{subject to } \Phi^T M \Phi = Id.$$

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 ith column of the minimizer Φ. The eigenpairs (λ_i, Φ_i) satisfy the equation

$$S\Phi_i = \lambda_i M \Phi_i, \tag{2.6}$$

which is the discrete analog of (2.3).

2.4. 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 (3.3) 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 [38] and the linear subspace construction proposed in [68]. However, there are essential differences to these approaches. For example, while [38] solve a box constraint quadratic optimization and [68] 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 [22] 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 2.1.

In the second step, we construct a preliminary matrix $\tilde{U} \in \mathbb{R}^{n\times d}$. The ith 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 $d(v_i, v_{s_j})$

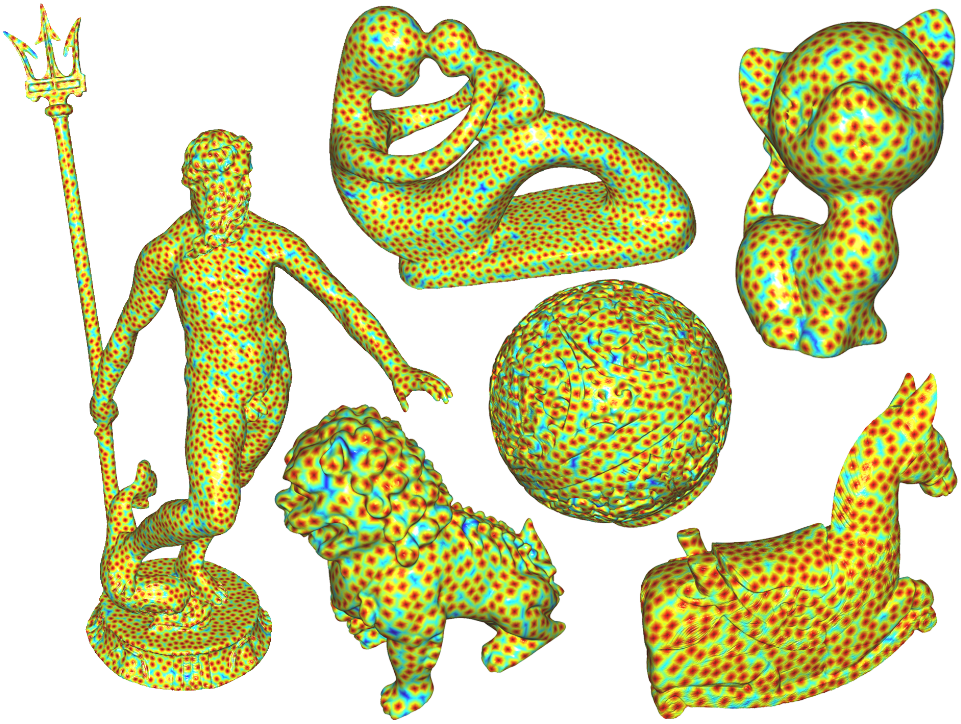


Figure 2.1: Examples of samplings used for the construction of the subspace bases showing between 1000 and 5000 samples on meshes with 100k to 2m vertices. The samples are computed with the constrained Poisson disk sampling scheme proposed in [22].

the geodesic distance between v_i and v_{s_j} 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 between the vertex and the sample increases. A visualization of a resulting function is shown in Figure 2.2.

We use a growing geodesic disk strategy for the construction of the locally supported functions. To obtain the jth column of $\tilde{U}$, we start with the sample v_{s_j} and process the

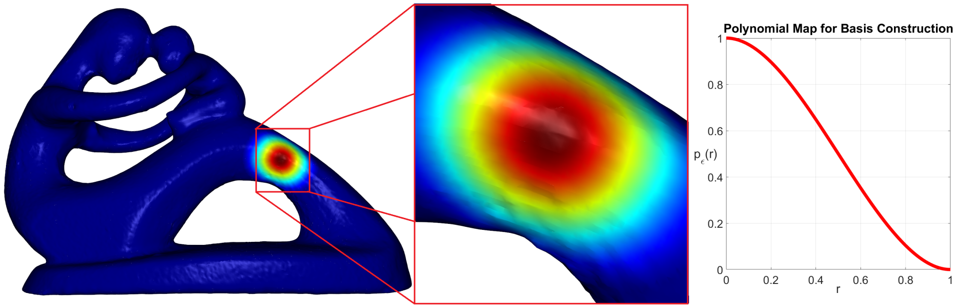


Figure 2.2: Visualization of a locally supported basis function used for the construction of the subspace bases on the Fertility model. The image on the right shows a plot of the polynomial $p_\rho(r)$, see Equation (2.A), for $\rho = 1$.

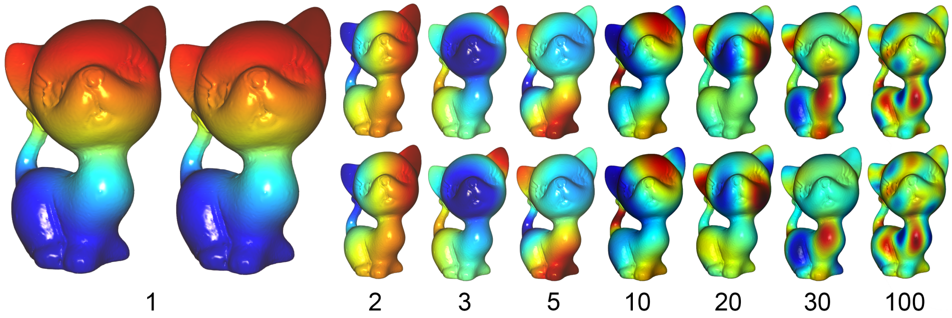


Figure 2.3: Comparison of approximate (bottom row) and reference (top row) eigenfunctions of the Laplace–Beltrami operator on the Kitten model. The reference solutions are computed with MATLAB's sparse eigensolver.

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 jth 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 2.1, 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 [21], 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 [16]. 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 [60], 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 (3.3) 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 (3.3) is

$$\min_{\bar{\phi}\in\mathbb{R}^{d\times m}} \operatorname{tr}\left(\bar{\phi}^T \bar{S} \bar{\phi}\right) \qquad (2.7)$$

$$\text{subject to } \bar{\phi}^T \bar{M} \bar{\phi} = Id.$$

The columns $\bar{\phi}_i$ of the resulting minimizer $\bar{\phi}$ are the restricted eigenvectors. The corresponding restricted eigenvalues are $\bar{\lambda}_i = \bar{\phi}_i^T \bar{S} \bar{\phi}_i$. The pairs of $(\bar{\lambda}_i, \bar{\phi}_i)$ satisfy the equation

$$\bar{S}\bar{\phi}_i = \bar{\lambda}_i \bar{M} \bar{\phi}_i. \qquad (2.8)$$

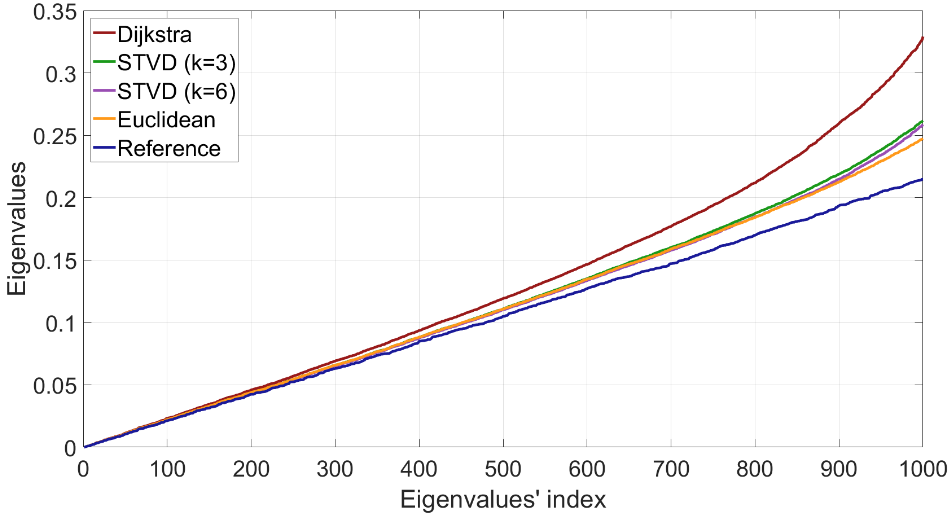


Figure 2.4: Approximations of the lowest 500 eigenvalues for different subspace constructions on the Fertility model: The subspace constructions differ in the scheme that is used for the approximation of the geodesic distances. The schemes used are: Dijkstra's algorithm, Dijkstra's algorithm with Euclidean 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.

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

$$\bar{\Phi}_i = U\bar{\phi}_i \tag{2.9}$$

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 2.1.

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 3.6. Since the eigenfunctions have dense nodal vectors, for a mesh with n vertices, storing m (unreduced) eigenvectors requires $\mathcal{O}(n\,m)$ 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 (2.9), we obtain

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

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 (3.7) of the minimization problem (3.6) 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 (2.2). Equation (2.3) 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 (2.9), we get

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

In the last step, we used the orthogonality constraint of the restricted eigenvalue problem (3.6). □

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

$$\bar{\Phi}_i = \sum_k a_{ik}\Phi_k, \tag{2.10}$$

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

$$\sum_k a_{ik}^2 = 1 \tag{2.11}$$

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*

$$\bar{\lambda}_i = \sum_k a_{ik}^2\lambda_k. \tag{2.12}$$

Proof. Using Lemmas 1 and 2, we get

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

which proves the theorem. □

The combination of equations (2.11) and (2.12) 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 $\bar{\lambda}_i$.

2.5. EXPERIMENTS

We implemented our approximation algorithm using the Eigen [29] and LibIGL [37] 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 2.5 shows plots of the first one hundred reference and approximate eigenvalues for three different models. Figure 2.4 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

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 2.1: 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.

produce good approximation results. Taking the timings, shown in Table 2.3, 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 (2.10). Figure 2.7 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 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 2.8 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 2.3 shows color plots of some of the reference and approximate eigenfunctions on the kitten mesh.

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

Table 2.2: 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.

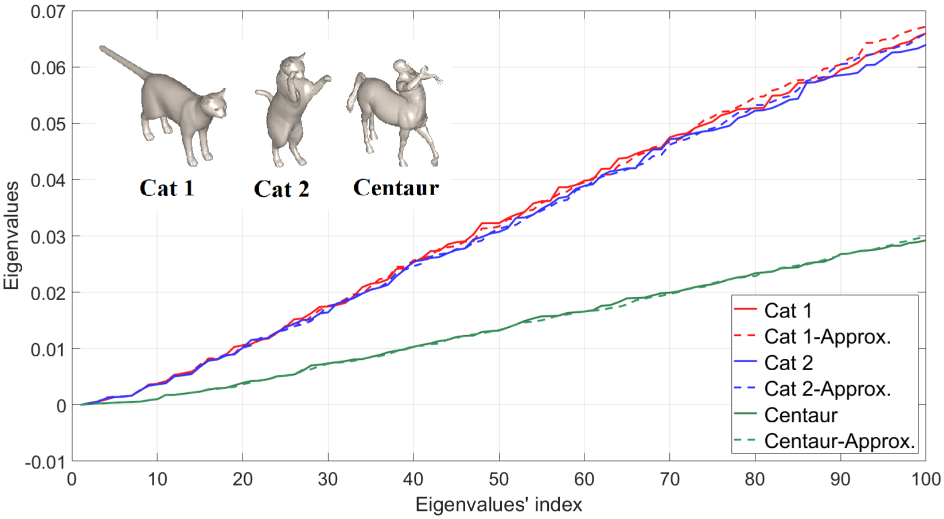


Figure 2.5: Comparison of the first 100 eigenvalues, approximation and reference solution, for three models, two Cat and one Centaur models. Reference solutions are computed using MATLAB's sparse eigensolver.

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 2.6 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 3.4 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. [20] introduced a scheme for estimating the Laplace–Beltrami operator that constructs a coarse voxel grid containing the surface mesh and creates a function

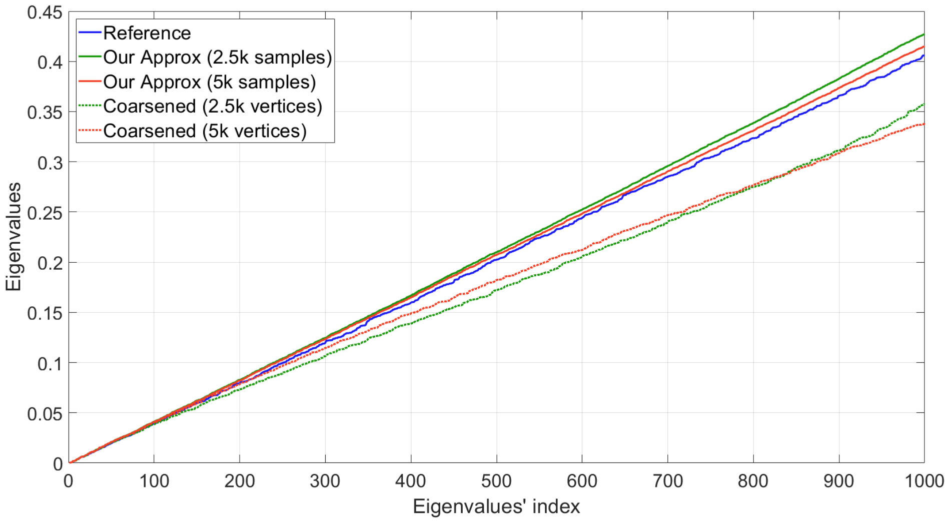


Figure 2.6: Approximations of the first 1000 eigenvalues of the Laplace–Beltrami operator on the Chinese Dragon mesh. The reference solution (blue), computed with MATLAB's sparse eigensolver, is compared with approximations computed with the proposed scheme with 2k (green) and 5k (red) dimensional subspaces and also computed from coarsened meshes with 2k (dashed green) and 5k (dashed red) vertices.

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

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

Table 2.3: 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.

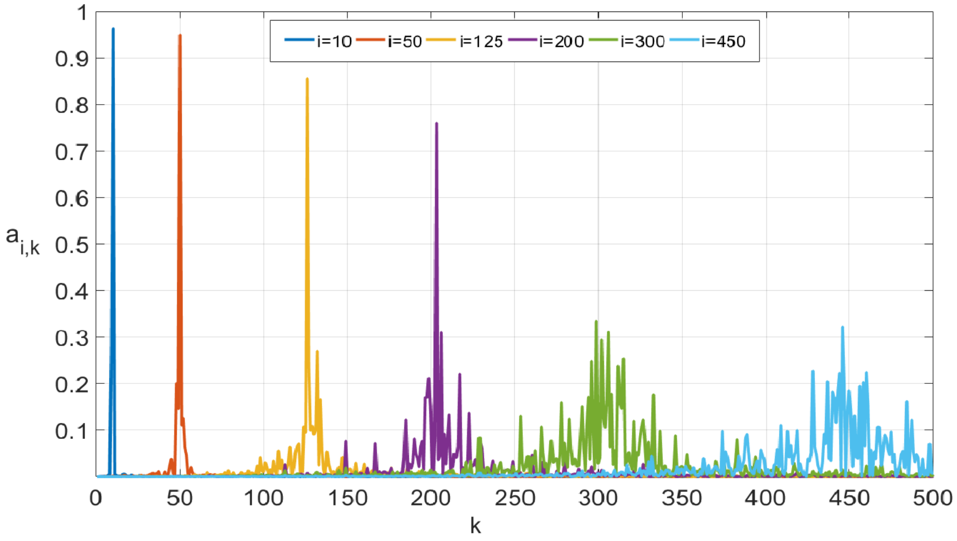


Figure 2.7: Plots of the Fourier coefficients, $a_{ik} = \langle \bar{\Phi}_i, \Phi_k \rangle_{L^2}$, of restricted eigenfunctions $\bar{\Phi}_i$ in the reference eigenbasis $\{\Phi_k\}$. The $\bar{\Phi}_i$ are computed in a 1000-dimensional space on the Kitten model.

in Table 2.1. 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 [65]. 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 2.2 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.

2.6. 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.

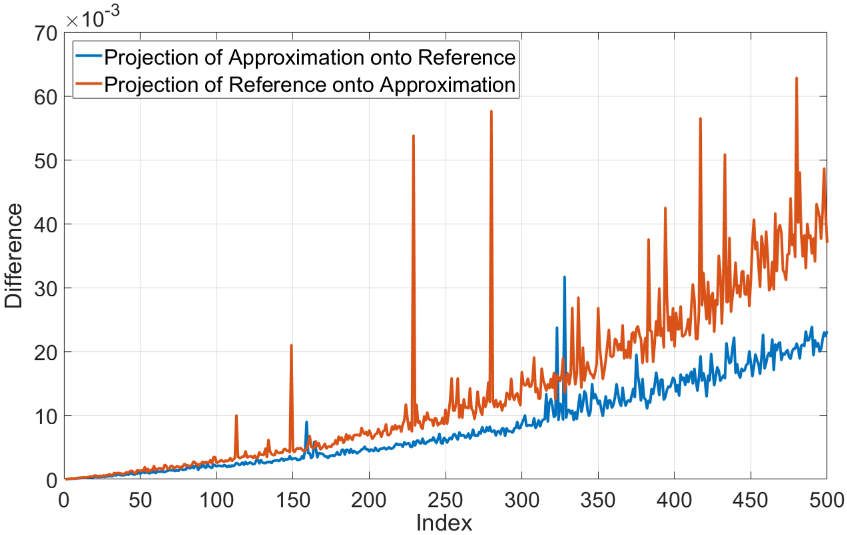


Figure 2.8: The norms of the difference between reference eigenfunction Φ_k and its projection onto the space spanned by the first 500 approximate eigenfunctions $\bar{\Phi}_i$ for $k \in \{1, 2, ..., 500\}$ are shown in red and an analogous plot with roles of Φ_k ad $\bar{\Phi}_i$ exchanged in blue.

Shape DNA Shape DNA [53] is a simple, yet effective, spectral shape descriptor. The simplicity makes it well-suited for our comparison. Figure 2.9 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 [12], 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 2.5. 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 [47]. 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 2.10 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 3.6, the precomputation time for the approximate solution is two orders of magnitude shorter.

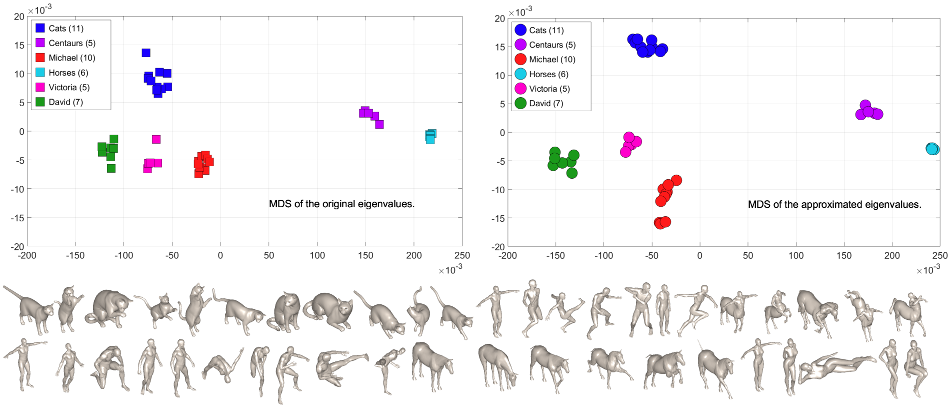


Figure 2.9: Plots visualizing the shape distances of a collections of models. The shape distances are computed from the Shape DNA, approximated with the proposed method (top-right) and reference eigenvalues (top-left) computed using MATLAB's sparse eigensolver. The plots are generated with multi-dimensional 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.

Mesh filtering Figures 2.11 and 2.12 show the results of spectral filtering [65] 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 [31, 19, 44, 69] 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 param-

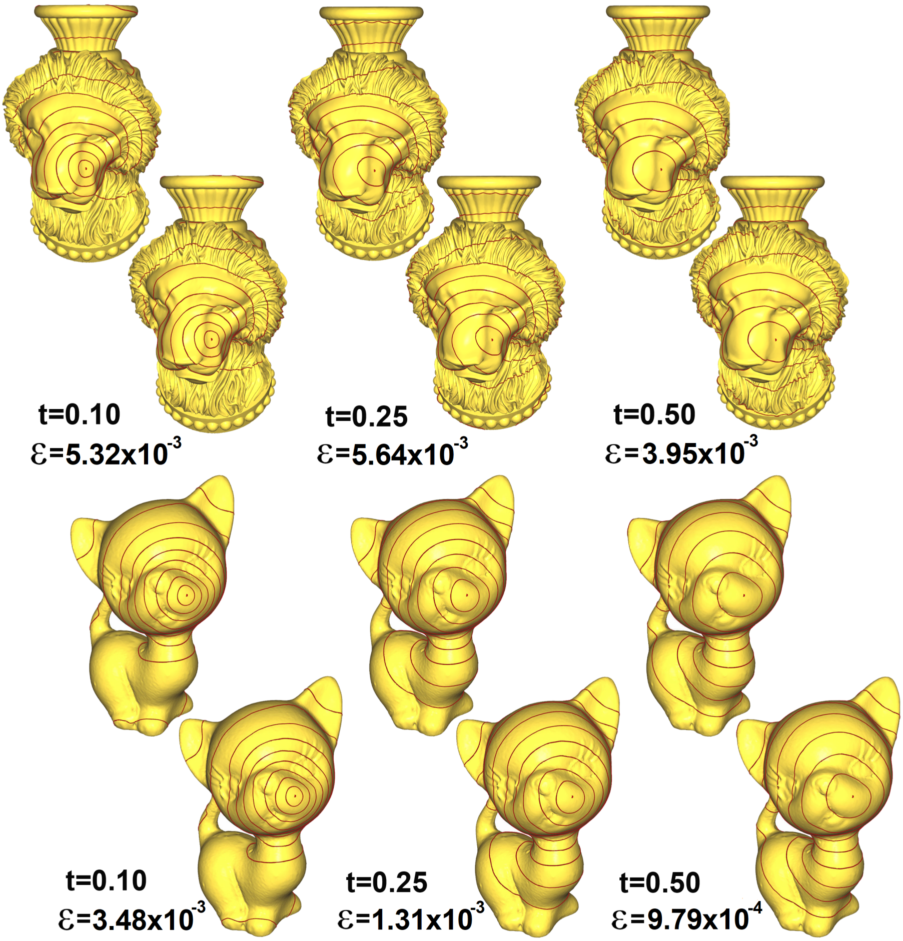


Figure 2.10: Comparison of diffusion distances computed using reference eigenpairs (first and third rows) and our approximations (second and fourth rows) on the Vase-Lion and Kitten models. Parameter of the diffusion distance, t, and relative L^2 approximation error, ϵ, are shown.

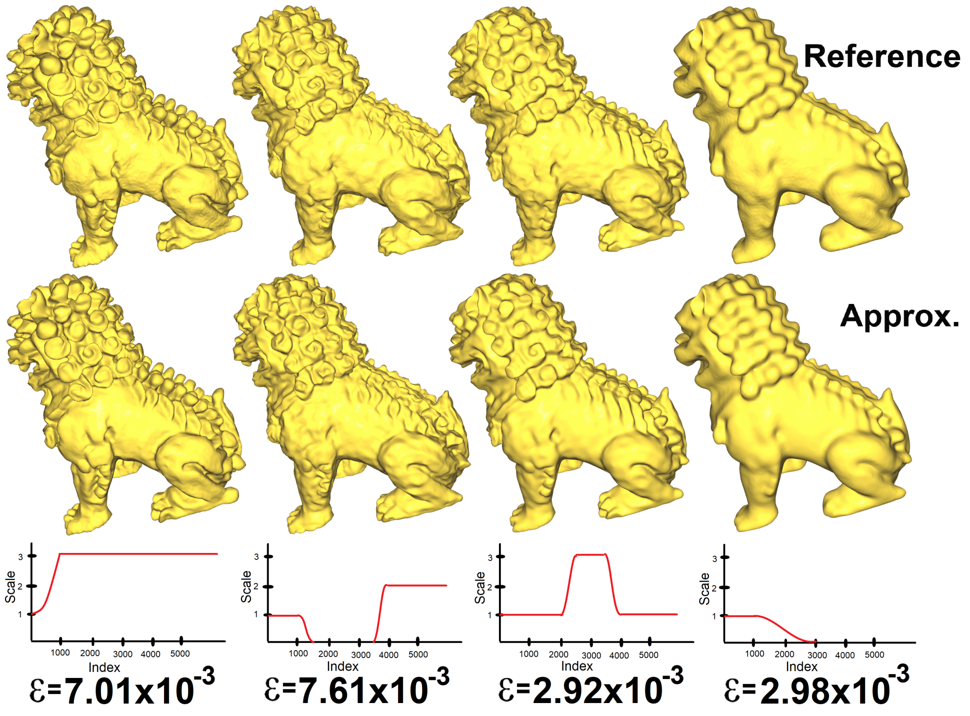


Figure 2.11: Comparison of results of spectral mesh filtering on the Chinese-Dragon model using 5000 approximated eigenvalues and eigenfunctions (bottom row) and 5000 eigenvalues and eigenfunctions computed using MATLAB (top row). Relative L^2 approximation errors, ϵ, are shown.

eter 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 2.13 shows examples of results obtained with the family of operators described in [58, pages 72–73], which extends the construction of a modified Laplacian from [31]. 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

$$(1-t)S + tA. \tag{2.13}$$

The operators are intrinsic for $t = 0$ and a potential, dependent on the extrinsic curvature, is blended in for $t > 0$. Figure 2.13 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

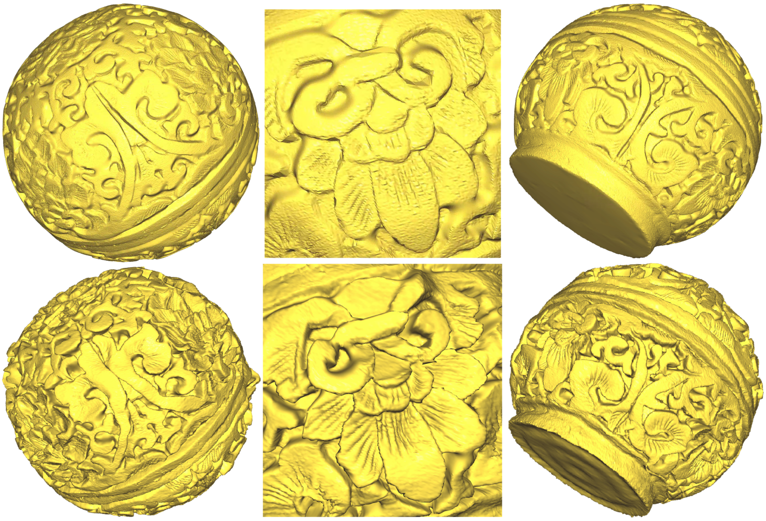


Figure 2.12: Results of a sharpening filter using 5000 approximate eigenpairs on the Red Circular Box model. Top row shows input and bottom row results.

modes are widely used in graphics for fast simulation [5, 64], simulation-based shape editing [33], sound synthesis [17], editing simulations [6], and for deformable motion design [32]. Yang et al. [72] 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 [72] 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 3.4, 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

$$H\Phi_i = \lambda M \Phi_i, \tag{2.14}$$

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

$$M\ddot{u}(t) + (\alpha M + \beta H)\dot{u}(t) + Hu(t) = F, \tag{2.15}$$

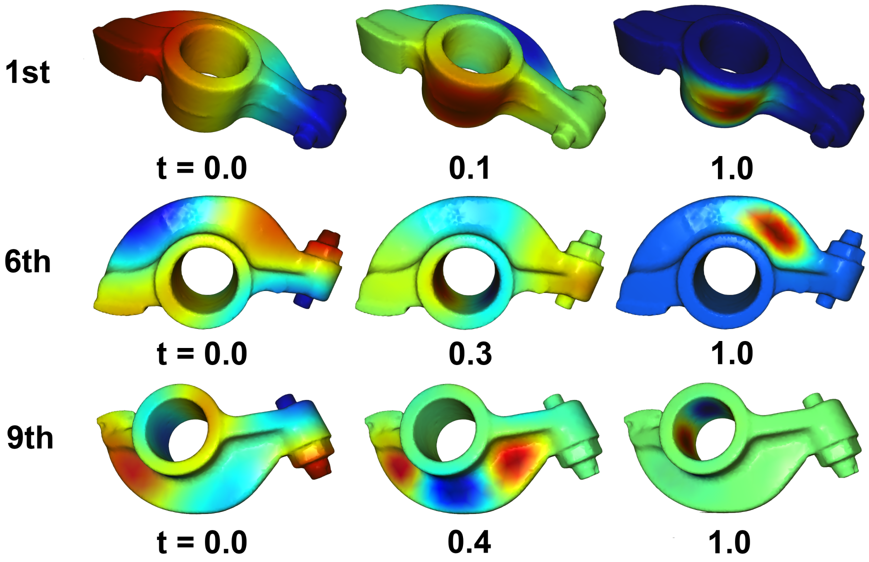


Figure 2.13: Eigenfunctions of a one-parameter family of operators are shown. For $t = 0$, the operator is the Laplace–Beltrami operator. For other values of t the operator includes extrinsic information. The higher the value of t the stronger the influence of the extrinsic information. The proposed approximation scheme allows to interactively explore the eigenfunctions the one-parameter family of operators.

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 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

$$\ddot{q}_i(t) + (\alpha + \beta\lambda_i)\dot{q}_i(t) + \lambda_i q_i(t) = \left(\Phi^T F\right)_i \tag{2.16}$$

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

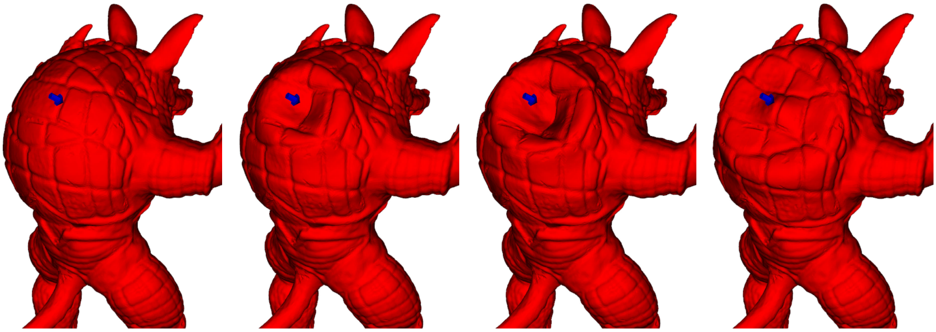


Figure 2.14: Real-time simulation of an elastic deformable (165k vertices) using 500 approximated vibration modes. A strong point force is applied to the back. High frequent details in the dynamics that spread across the mesh without (numerically) dissipating are observed.

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 2.14 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 (2.16) using fully computed vibration modes and approximated vibration modes respectively. For the first 250 frames of the simulation shown in Figure 2.14, 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 $U \cdot \big(\bar{\phi} \cdot q(t)\big)$ requires around 7 milliseconds, whereas the product $\Phi q(t)$ takes 120 milliseconds of computation time on average, which prohibits real-time applications.

2.7. 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 [49] and compressed vibration modes [9]. 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.

BIBLIOGRAPHY

[1] Marc Alexa and Max Wardetzky. “Discrete Laplacians on General Polygonal Meshes”. In: *ACM Trans. Graph.* 30.4 (2011), 102:1–102:10.

[2] M. Aubry, U. Schlickewei, and D. Cremers. “The wave kernel signature: A quantum mechanical approach to shape analysis”. In: *ICCV*. 2011, pp. 1626–1633.

[3] Omri Azencot et al. “An Operator Approach to Tangent Vector Field Processing”. In: *Computer Graphics Forum* 32.5 (2013), pp. 73–82.

[4] Omri Azencot et al. “Discrete Derivatives of Vector Fields on Surfaces – An Operator Approach”. In: *ACM Trans. Graph.* 34.3 (2015), 29:1–29:13.

[5] Jernej Barbič and Doug L. James. “Real-Time subspace integration for St. Venant-Kirchhoff deformable models”. In: *ACM Trans. Graph.* 24.3 (2005), pp. 982–990.

[6] Jernej Barbič, Funshing Sin, and Eitan Grinspun. “Interactive editing of deformable simulations”. In: *ACM Trans. Graph.* 31.4 (2012), 70:1–70:8.

[7] Klaus-Jürgen Bathe. “The Subspace Iteration Method - Revisited”. In: *Computers and Structures* 126 (2013), pp. 177–183.

[8] D. Boscaini et al. “Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks”. In: *Computer Graphics Forum* 34.5 (2015), pp. 13–23.

[9] Christopher Brandt and Klaus Hildebrandt. “Compressed Vibration Modes of Deformable Bodies”. In: *Computer Aided Geometric Design* 52–53 (2017), pp. 297–312.

[10] Christopher Brandt et al. “Modeling n-Symmetry Vector Fields using Higher-Order Energies”. In: *ACM Trans. on Graph.* 37.2 (2018), 18:1–18:18.

[11] Christopher Brandt et al. “Spectral Processing of Tangential Vector Fields”. In: *Computer Graphics Forum* 36.6 (2017), pp. 338–353.

[12] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical geometry of non-rigid shapes*. Springer, 2008.

[13] Alexander M. Bronstein et al. “Shape Google: Geometric Words and Expressions for Invariant Shape Retrieval”. In: *ACM Trans. Graph.* 30.1 (2011), 1:1–1:20.

[14] Michael M. Bronstein et al. “Geometric Deep Learning: Going beyond Euclidean data”. In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42.

[15] Joan Bruna et al. “Spectral networks and locally connected networks on graphs”. In: *International Conference on Learning Representations* (2014).

[16] Marcel Campen, Martin Heistermann, and Leif Kobbelt. “Practical Anisotropic Geodesy”. In: *Computer Graphics Forum* 32.5 (2013), pp. 63–71.

[17] Jeffrey N. Chadwick, Steven S. An, and Doug L. James. “Harmonic shells: a practical nonlinear sound model for near-rigid thin shells”. In: *ACM Trans. Graph.* 28.5 (2009), 119:1–119:10.

[18] Yoni Choukroun, Gautam Pai, and Ron Kimmel. “Schrödinger Operator for Sparse Approximation of 3D Meshes”. In: *Symposium on Geometry Processing 2017-Posters*. 2017. DOI: `10.2312/sgp.20171205`.

[19] Yoni Choukroun et al. “Elliptic operator for shape analysis”. In: *ArXiv* abs/1611.01990 (2016).

[20] Ming Chuang et al. “Estimating the Laplace–Beltrami Operator by Restricting 3D Functions”. In: *Computer Graphics Forum* 28.5 (2009), pp. 1475–1484.

[21] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[22] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. “Efficient and flexible sampling with blue noise properties of triangular meshes”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2012), pp. 914–924.

[23] Keenan Crane et al. “Digital Geometry Processing with Discrete Exterior Calculus”. In: *ACM SIGGRAPH 2013 courses*. SIGGRAPH ’13. 2013.

[24] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “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.

[25] Shen Dong et al. “Spectral surface quadrangulation”. In: *ACM Trans. Graph.* 25.3 (2006), pp. 1057–1066.

[26] Petros Drineas and Michael W. Mahoney. “On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning”. In: *J. Mach. Learn. Res.* 6 (2005), pp. 2153–2175.

[27] C. Fowlkes et al. “Spectral grouping using the Nystrom method”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (2004), pp. 214–225.

[28] Katarzyna Gebal et al. “Shape Analysis Using the Auto Diffusion Function”. In: *Computer Graphics Forum* 28.5 (2009), pp. 1405–1413.

[29] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[30] N. Halko, P. G. Martinsson, and J. A. Tropp. “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions”. In: *SIAM Review* 53.2 (2011), pp. 217–288.

[31] Klaus Hildebrandt et al. “Eigenmodes of surface energies for shape analysis”. In: *Advances in Geometric Modeling and Processing*. Vol. 6130. Lecture Notes in Computer Science. Springer, 2010, pp. 296–314.

[32] Klaus Hildebrandt et al. “Interactive spacetime control of deformable objects”. In: *ACM Trans. Graph.* 31.4 (2012), 71:1–71:8.

[33] Klaus Hildebrandt et al. “Interactive surface modeling using modal analysis”. In: *ACM Trans. Graph.* 30.5 (2011), 119:1–119:11.

[34] Klaus Hildebrandt et al. “Modal shape analysis beyond Laplacian”. In: *Computer Aided Geometric Design* 29.5 (2012), pp. 204–218.

[35] Jin Huang et al. “Spectral quadrangulation with orientation and alignment control”. In: *ACM Trans. Graph.* 27.5 (2008), pp. 1–9.

[36] Qixing Huang et al. “Shape Decomposition Using Modal Analysis”. In: *Computer Graphics Forum* 28.2 (2009), pp. 407–416.

[37] Alec Jacobson, Daniele Panozzo, et al. *libigl: A simple C++ geometry processing library.* http://libigl.github.io/libigl/. 2016.

[38] Alec Jacobson et al. “Bounded biharmonic weights for real-time deformation”. In: *ACM Trans. Graph.* 30.4 (2011), 78:1–78:8.

[39] Zachi Karni and Craig Gotsman. “Spectral Compression of Mesh Geometry”. In: *ACM SIGGRAPH.* 2000, pp. 279–286.

[40] Artiom Kovnatsky et al. “Coupled quasi-harmonic bases”. In: *Comput. Graph. Forum* 32.2 (2013), pp. 439–448.

[41] Ron Levie et al. “CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters”. In: *ArXiv* abs/1705.07664 (2017).

[42] Ruotian Ling et al. “Spectral Quadrangulation with Feature Curve Alignment and Element Size Control”. In: *ACM Trans. Graph.* 34.1 (2014), 11:1–11:11.

[43] Or Litany et al. “Fully Spectral Partial Shape Matching”. In: *Comput. Graph. Forum* 36.2 (2017), pp. 247–258.

[44] Derek Liu, Alec Jacobson, and Keenan Crane. “A Dirac Operator for Extrinsic Shape Analysis”. In: *Computer Graphics Forum* 36.5 (2017).

[45] S. Melzi et al. “Localized Manifold Harmonics for Spectral Shape Analysis”. In: *Computer Graphics Forum* 37 (2018).

[46] Przemyslaw Musialski et al. “Reduced-order Shape Optimization Using Offset Surfaces”. In: *ACM Trans. Graph.* 34.4 (2015), 102:1–102:9.

[47] Boaz Nadler et al. “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.

[48] Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. “Fast Approximation of Laplace–Beltrami Eigenproblems”. In: *Comp. Graph. Forum* 37.5 (2018).

[49] T. Neumann et al. “Compressed Manifold Modes for Mesh Processing”. In: *Comput. Graph. Forum* 33.5 (2014), pp. 35–44.

[50] Maks Ovsjanikov et al. “Computing and Processing Correspondences with Functional Maps”. In: *SIGGRAPH ASIA 2016 Courses.* ACM, 2016, 9:1–9:60.

[51] Maks Ovsjanikov et al. “Functional Maps: A Flexible Representation of Maps Between Shapes”. In: *ACM Trans. Graph.* 31.4 (2012), 30:1–30:11.

[52] Giuseppe Patanè. “Accurate and Efficient Computation of Laplacian Spectral Distances and Kernels”. In: *Comput. Graph. Forum* 36.1 (2017), pp. 184–196.

[53] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. “Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids”. In: *Computer-Aided Design* 38.4 (2006), pp. 342–366.

[54] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. “Laplace-Spectra as Fingerprints for Shape Matching”. In: *Proceedings of the ACM Symposium on Solid and Physical Modeling*. 2005, pp. 101–106.

[55] Emanuele Rodolà et al. “Partial Functional Correspondence”. In: *Comput. Graph. Forum* 36.1 (2017), pp. 222–236.

[56] Raif M. Rustamov. “Laplace–Beltrami eigenfunctions for deformation invariant shape representation”. In: *Symposium on Geometry Processing*. 2007, pp. 225–233.

[57] Raif M. Rustamov et al. “Map-based Exploration of Intrinsic Shape Differences and Variability”. In: *ACM Trans. Graph.* 32.4 (2013), 72:1–72:12.

[58] Christian Schulz. “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`.

[59] Avinash Sharma et al. “Mesh Segmentation Using Laplacian Eigenvectors and Gaussian Mixtures”. In: *Manifold Learning and Its Applications*. 2009.

[60] O. G. Smolyanov, H. von Weizäcker, and O. Wittich. “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.

[61] Ran Song et al. “Mesh Saliency via Spectral Processing”. In: *ACM Trans. Graph.* 33.1 (2014), 6:1–6:17.

[62] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. “A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion.” In: *Computer Graphics Forum* 28.5 (2009), pp. 1383–1392.

[63] Ameet Talwalkar et al. “Large-scale SVD and Manifold Learning”. In: *Journal of Machine Learning Research* 14 (2013), pp. 3129–3152.

[64] Christoph von Tycowicz et al. “An Efficient Construction of Reduced Deformable Objects”. In: *ACM Trans. Graph.* 32.6 (2013), 213:1–213:10.

[65] Bruno Vallet and Bruno Lévy. “Spectral Geometry Processing with Manifold Harmonics”. In: *Computer Graphics Forum* 27.2 (2008), pp. 251–260.

[66] Libor Váša et al. “Compressing dynamic meshes with geometric Laplacians”. In: *Computer Graphics Forum* 33.2 (2014), pp. 145–154.

[67] Amir Vaxman, Mirela Ben-Chen, and Craig Gotsman. “A Multi-resolution Approach to Heat Kernels on Discrete Surfaces”. In: *ACM Trans. Graph.* 29.4 (2010), 121:1–121:10.

[68] Yu Wang et al. “Linear Subspace Design for Real-time Shape Deformation”. In: *ACM Trans. Graph.* 34.4 (2015), 57:1–57:11.

[69] Yu Wang et al. “Steklov Geometry Processing: An Extrinsic Approach to Spectral Shape Analysis”. In: *ArXiv* abs/1707.07070 (2017).

[70] Max Wardetzky et al. “Discrete quadratic curvature energies”. In: *Computer Aided Geometric Design* 24.8-9 (2007), pp. 499–518.

[71] Christopher K. I. Williams and Matthias Seeger. “Using the Nyström Method to Speed Up Kernel Machines”. In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[72] Yin Yang et al. “Expediting Precomputation for Reduced Deformable Simulation”. In: *ACM Trans. Graph.* 34.6 (2015), 243:1–243:13.

[73] Hao Zhang, Oliver van Kaick, and Ramsay Dyer. “Spectral Mesh Processing”. In: *Computer Graphics Forum* 29.6 (2010), pp. 1865–1894.

APPENDIX

This supplementary material contains additional experimental results concerning the method presented in the submission *Fast Approximation of Laplace–Beltrami Eigenproblems.*

2.A. CHOICE OF BASIS FUNCTIONS

In Section 4 of the submission, the construction of the subspace basis is introduced. After sampling, a preliminary matrix $\tilde{U} \in \mathbb{R}^{n\times d}$ in which the ith column represents a locally supported function centered at the sample point v_{s_i} is constructed. 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 use the cubic polynomial

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

which satisfies $p_\rho(0) = 1$, $\frac{\partial}{\partial r} p_\rho(0) = 0$, $p_\rho(\rho) = 0$, and $\frac{\partial}{\partial r} p_\rho(\rho) = 0$ for our construction.

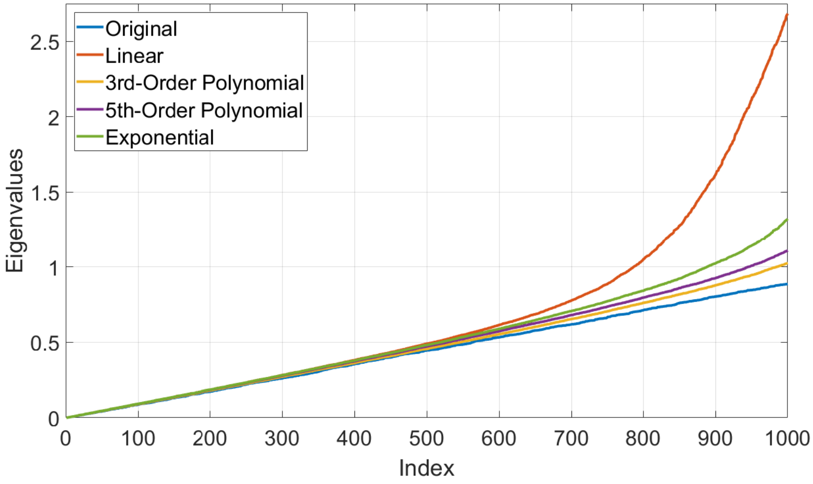


Figure 2.A.1: Experimental results that compare the approximate eigenvalues computed using different functions for basis construction to a reference solution are shown.

Of course, other choices of functions are possible. In this section, we describe three alternative choices of functions. Figure 2.A.2 shows graphs of the functions for illustration. Figure 2.A.1 shows experimental results that compare the resulting approximate eigenvalues to a reference solution.

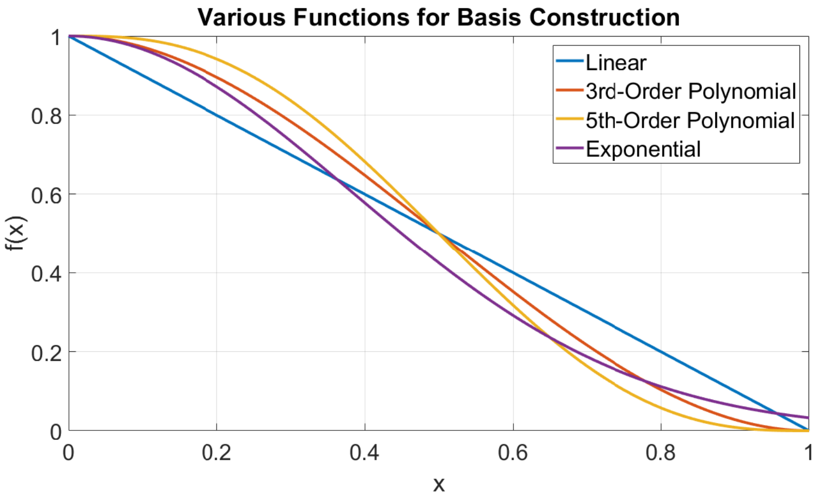


Figure 2.A.2: Graphs of the four functions for $\rho = 1$ are shown.

The first alternative function is the linear polynomial

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

that satisfies $p_\rho(0) = 1$ and $p_\rho(\rho) = 0$. Compared to the cubic polynomial, this polynomial is simpler, but only conituous and not differentiable at ρ.

The second alternative is the fifth-order polynomial

$$p_\rho^{fifth}(r) = \begin{cases} -\frac{6}{\rho^5} r^5 + \frac{15}{\rho^4} r^4 - \frac{10}{\rho^3} r^3 + 1 & \text{for } r \leq \rho \\ 0 & \text{for } r > \rho \end{cases},$$

that satisfies $p_\rho(0) = 1$, $\frac{\partial}{\partial r} p_\rho(0) = 0$, $\frac{\partial^2}{\partial r^2} p_\rho(0) = 0$, $p_\rho(\rho) = 0$, $\frac{\partial}{\partial r} p_\rho(\rho) = 0$, and $\frac{\partial^2}{\partial r^2} p_\rho(\rho) = 0$. This polynomial is not just once, but twice differentiable at ρ.

The third alternative is an exponential function

$$p_\rho^{exp}(r) = \begin{cases} e^{-\frac{\log(2) r^2}{0.45^2 \rho^2}} = & \text{for } r \leq \rho \\ 0 & \text{for } r > \rho \end{cases},$$

that we cut off at ρ.

In our experiments, we compared the approximation error for the eigenvalues we obtain with the different functions and found that the third-order and fifth-order polynomial and the exponential function, produce comparable approximation errors, where the third-order polynomial performs slightly better than the other two. The linear function produced higher errors. An example of results is shown in Figure 2.A.1. In this example 1000 approximate eigenvalues that computed in a 1000-dimensional space are shown. Note that in the submission we suggest not to use all 1000 eigenvalues but rather only the first 500.

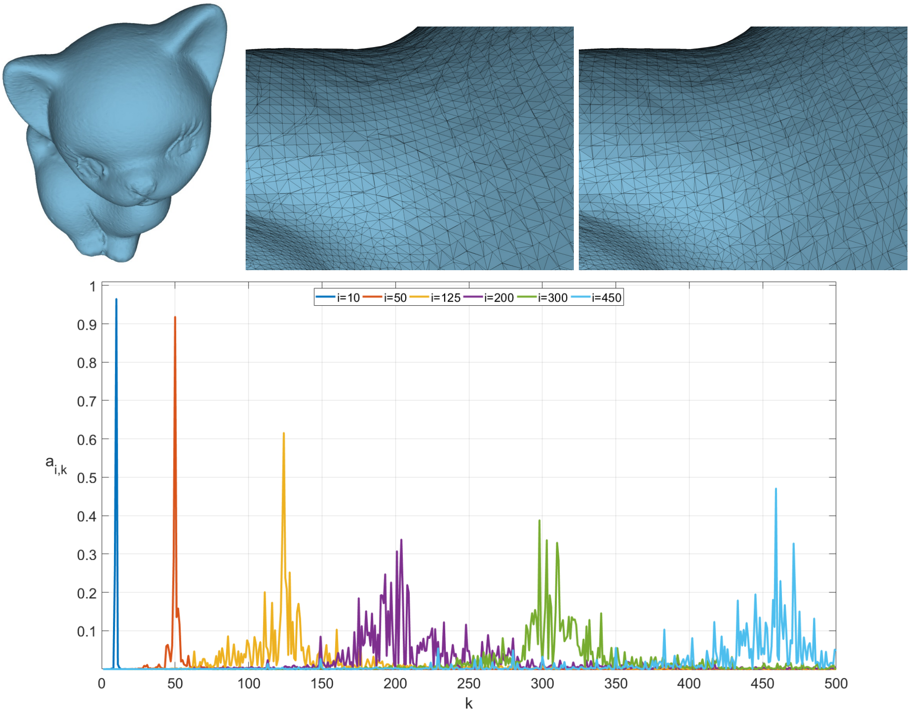


Figure 2.B.1: Eigenfunction of the kitten model before and after flipping some edges are compared. The top row show an image of the kitten model and zoom-in images of the mesh before and after some edge flips. The bottom row shows plots of the Fourier coefficients, $a_{ik} = \langle \tilde{\Phi}_i, \Phi_k \rangle_{L^2}$, of eigenfunction $\tilde{\Phi}_i$ of the kitten with flipped edges in the eigenbasis $\{\Phi_k\}$ of the kitten before edges are flipped.

2.B. EIGENFUNCTIONS AND EDGE FLIPS

To put the approximation results for the Laplace–Beltrami eigenfunctions discussed in Section 5 of the paper into a broader context, we want to add an experiments that explores how the eigenfunctions change when the metric of a surface is slightly changed. For this, we re-meshed the kitten model by applying a series of edge flips. All vertices are kept in place, but the flips change the metric and hence the discrete Laplace–Beltrami operator. Images of the two meshes are shown in Figure 2.B.1. We computed the lowest 500 eigenfunctions of both meshes. To compare them, we looked at the Fourier coefficients, $a_{ik} = \langle \tilde{\Phi}_i, \Phi_k \rangle_{L^2}$, of eigenfunction $\tilde{\Phi}_i$ of the kitten with flipped edges in the eigenbasis $\{\Phi_k\}$ of the kitten before edges are flipped. Plots of the Fourier coefficients of some of the eigenfunctions are shown in Figure 2.B.1. We observed that the difference of the eigenfunctions resulting from the edge flips is of similar magnitude as the difference to the approximate eigenfunctions computed with our approximation algorithm as shown in Figure 7 of the paper.

2.C. COMPARISON TO MESH COARSENING

In Section 5 of the submission, the proposed method is compared to a mesh coarsening approach for eigenvalue approximation and Figure 6 (of the submission) shows one example of approximate eigenvalues computed with the proposed method and mesh coarsening. Figure 2.C.1 of this supplementary material shows more examples with a comparable setting on different surfaces.

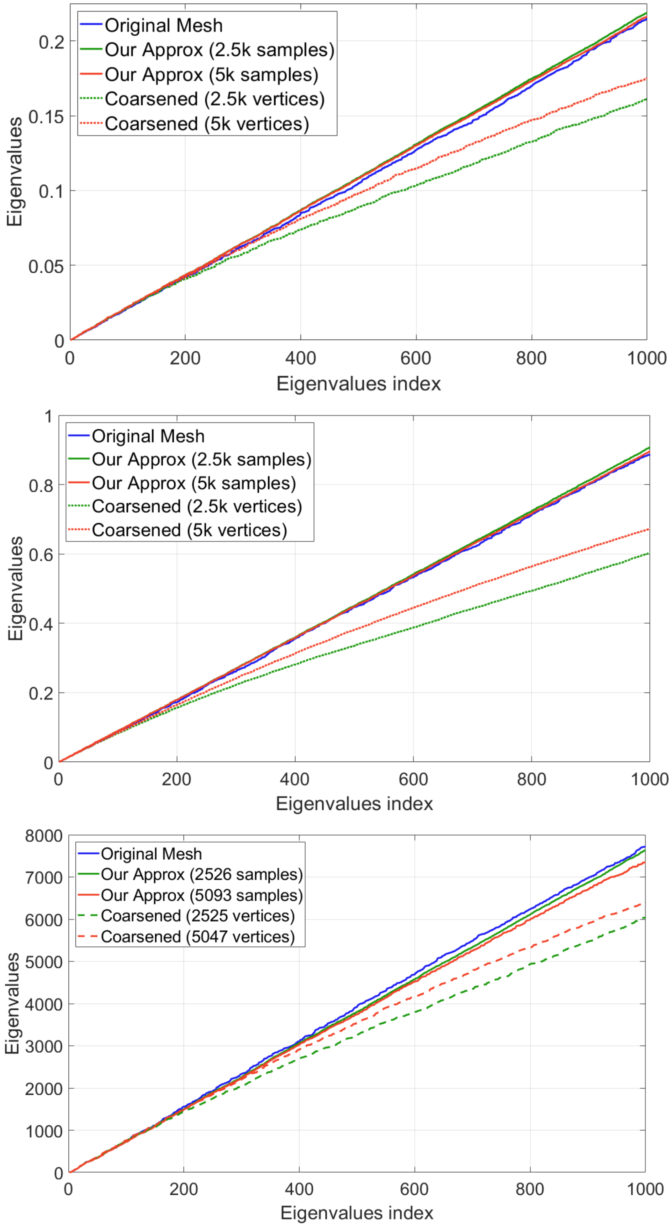


Figure 2.C.1: Approximations of the first 1000 eigenvalues of the Laplace–Beltrami operator on the Fertility, the Kitten and the Armadillo mesh are shown. The reference solutions (blue), compute with MATLAB's sparse eigensolver, is compared with approximations computed with the proposed scheme with 2k (green) and 5k (red) dimensional subspaces and computed from coarsened meshes with 2k (dashed green) and 5k (dashed red) vertices.

3

LOCALLY SUPPORTED TANGENTIAL VECTOR, n-VECTOR, AND TENSOR FIELDS

Science is a differential equation.
Religion is a boundary condition.

Alan Turing

We introduce a construction of subspaces of the spaces of tangential vector, n-Vector, and tensor fields on surfaces. The resulting subspaces can be used as the basis of fast approximation algorithms for design and processing problems that involve tangential fields. Important features of our construction are that it is based on a general principle, from which constructions for different types of tangential fields can be derived, and that it is scalable, making it possible to efficiently compute and store large subspace bases for large meshes. Moreover, the construction is adaptive, which allows for controlling the distribution of the degrees of freedom of the subspaces over the surface. We evaluate our construction in several experiments addressing approximation quality, scalability, adaptivity, computation times and memory requirements. Our design choices are justified by comparing our construction to possible alternatives. Finally, we discuss examples of how subspace methods can be used to build interactive tools for tangential field design and processing tasks.

This chapter is based on the paper *Locally Supported Tangential Vector, n-Vector, and Tensor Fields* published in Eurographics Computer Graphics Forum (2020) [49].

3.1. INTRODUCTION

Directional information along a surface is usually encoded as a tangential vector, n-vector or tensor field. Many applications in computer graphics, such as line art rendering, meshing, texturing, and BRDF design, rely on techniques for the design and processing of such tangential fields. A problem arises from the fact that the triangular meshes describing the surface are usually of high resolution and the complexity of the tangential fields is connected to that of the meshes. As a result, large-scale equations and optimization problems have to be solved for field design and processing, while the applications expect fast response times because workflows often involve user interaction. Established acceleration methods, such as radial basis functions, which proved to be effective for problems like interactive shape deformation, cannot be used for the processing of tangential fields since the fields are defined on curved surfaces and the tangential bundles of the surfaces are non-trivial.

We introduce constructions of tangential vector, n-vector and tensor fields on meshes and use them to construct subspaces for design and processing tasks. By restricting the equations and optimization problems to such a subspace, the degrees of freedom of a design or processing problem can be adjusted detached from the complexity of the triangle mesh representing the surface. This is an important step towards enabling interactive techniques for the design and processing of tangential fields on large meshes.

The idea underlying our approach is to construct tangential fields by computing the lowest eigenfields of a suitable Laplace operator restricted to small disk-shaped subsets of the surface. Boundary conditions on the eigenproblems are imposed to guaranty that each of the resulting fields vanishes at the boundaries of its subset. The construction of the individual fields is combined with a procedure that defines the subsets in the surface so that useful bases of subspaces are created. Important aspects of our construction are:

- *Generality* Our construction is derived from a general principle. By choosing an appropriate Laplace operator, specific constructions for the different types of tangential fields can be derived. We explicitly describe constructions of vector, n-vector, and tensor fields.

- *Scalability* We show that large subspaces with several thousand dimensions on meshes with more than a million triangles can be efficiently constructed and stored. A prerequisite for this is the localization of the tangential fields, which facilitates the efficient computation and storage of the individual fields. The cost for field construction depends only on the size of the disk-like subsets and storage requirements are low as the fields can be represented by sparse vectors.

- *Smoothness* The construction is chosen so that the resulting fields are smooth. The eigenproblems we solve can be written as optimization problems, and, as minimizers, the lowest eigenfields are the smoothest fields that vanish outside their assigned subsets. Here smoothness is measured by the Dirichlet energy corresponding to the Laplace operator that is used.

- *Approximation* We show that the resulting subspaces can approximate smooth fields well. To evaluate this aspect, we compute residuals when projecting fields into the subspace and when solving optimization problems in the subspaces, and compare the approximation results with results obtained using other possible constructions of tangential fields.

- *Adaptivity* We show that adaptivity can be effortlessly integrated to the construction. This allows for controlling the distribution of the degrees of freedom of a subspace over the surface. For example, fields that include details in designated areas of the surface can be better represented in the subspaces.

This is the first method for constructing tangential vector, n-vector and tensor fields with these properties. In particular, scalability and adaptivity distinguish the construction from alternative vector field constructions. These two points are crucial for efficiently generating subspaces with good approximation properties and modeling capacity on larger meshes. To justify our design choices for the proposed construction, we compare our construction with other existing and possible constructions in Section 3.7. In addition, we evaluate the approximation quality of the subspaces in different settings, Section 3.6, and show the benefits of the bases for the applications in Section 3.8.

3.2. RELATED WORK

Tangential fields The efficient design and processing of tangential vector, n-vector and tensor fields is important for a broad range of applications in computer graphics. Examples are texture generation[56, 67, 74, 16, 41], line art [30] and painterly rendering [77], anisotropic shading [47, 62], image stylization [76], surface segmentation [64, 80], surface construction [35, 51], meshing [61, 39, 8, 43, 65], and the simulation of fluid and liquids on surfaces [4, 5]. Tangential field design and processing presents many challenges, and different approaches have been proposed to address these problems. In the following, we briefly discuss approaches that are closely related to our work. For further background information and references, we refer to the surveys [24, 70].

Variational approaches for the design and processing of tangential fields minimize an objective that combines a fairness measure and functionals that penalize the deviation of the field from user input or geometric properties of the surface like curvature directions. The fairness measures quantify the variation of the field along the surface. For *vector field* design, quadratic objectives based on the divergence and curl of the fields can be used [21]. Discrete differentiable operators for *tensor fields*, based on Discrete Exterior Calculus [20], are introduced in [25]. For *n-rotational symmetric vector fields* (n-fields), the fairness measure introduced in [30] measures the deviation in angle between close-by n-vectors. To disambiguate the definition of angles between n-vectors, a periodic function is used in the fairness measure. The concept of representation vectors corresponding to n-vectors [61, 50] allows to model n-vector design using optimization of the representation vector fields. The representation vectors can be used to define a linear structure on n-vectors. This allows to model n-field design and processing problems using linear systems [40, 45, 11]. For the design of general, not necessarily rotational symmetric, n-fields, the polyvector representation [22, 23, 63] was introduced. While typically fairness measures are modeled as intrinsic objectives, an extrinsic objective was proposed in [38, 34]. The objective combines intrinsic fairness and alignment to curvature, while at the same time avoiding the need to use parallel transport on the surface for evaluation of the objective. In another line of work, explicit matchings that en-code the n pairs of corresponding vectors between neighboring n-vectors are used [39, 60, 8]. In an optimization, the matchings are treated as variables which leads to mixed-integer problems that have to be solved. In addition to variational design of vector fields, approaches that construct vector fields from user input specifying the location and degree of singularities have been proposed [78, 50, 59, 17, 42]. A subdivision scheme for discrete differential forms on meshes was introduced in [71]. The scheme combines different subdivision rules for the different k-forms such that the subdivision operations commute with the exterior derivative. This approach was extended to a subdivision exterior calculus [26] that provides a way to apply the numerical tools from Discrete Exterior Calculus to subdivision surfaces. Among other applications, the approach can be used for the design of vector fields on subdivision surfaces. In recent work [19], a structure-preserving subdivision approach for tangential direction fields was developed and used for directional field design on subdivision surfaces.

Subspace methods Subspace methods can be used for the design of fast approximation algorithms for complex systems. In the preprocessing stage, the subspace and additional structures for evaluation the objective and its derivatives are constructed. In the online stage, the precomputed structures are used to accelerate computations. The low computational cost in the online stage, makes subspace methods attractive for interactive graphics applications. Reduced systems have been proposed for the simulation of fluids [66, 44, 18], elastic solids and shells [6, 1, 75, 9], fluid-solid interaction [46, 13], example-based elastic material [79], motion planning [7, 31, 52], clothing [28], and hair [14].

In the context of mesh processing, subspace methods have been introduced for surface modeling [33, 32, 37, 72], shape interpolation [69, 58], injective mappings [29], motion processing [10], and spectral mesh processing [48]. The goal of this chapter is to explore subspace constructions for tangential vector, n-vector, and tensor fields on surfaces and the use of subspace methods for the design and processing of tangential fields.

Subspace methods for tangential fields For tangential vector fields, eigenfields of vector Laplace operators have been used for defining functional operators on spaces of vector fields [2, 3], subspace fluid simulation on surfaces [44] and spectral vector field processing [12]. Eigenfields on an n-vector field Laplacian were used as the basis of an approach for interactive n-field design in [11]. Although eigenfields are useful for the construction of subspace in certain scenarios, there are also fundamental limitations. We propose an alternative construction of subspaces that addresses these limitations. In particular, we aim at reducing the memory requirements for storing the bases and the computational cost for constructing the bases. In Section 3.7, we compare the proposed subspaces to eigenspaces.

3.3. LAPLACE OPERATORS

In this section, we briefly describe the discrete Laplace operators for tangential vector, n-vector, and tensor fields that are needed for the proposed construction of localized fields. To our knowledge, the tensor field Laplacian we describe is novel.

Discrete Fields There are various possibilities for discretizing fields on meshes. Degrees of freedom of the fields can be associated with the meshes' vertices, edges, faces or combinations of these. We refer to the survey [24] for a detailed discussion of the benefits and drawbacks of different discretizations. Our basis construction can be used with any discretization as long as a Laplace operator on the space is available. For the evaluation of our construction, we consider vector, n-vector, and tensor fields that are constant and tangential in every face. We denote by k the dimension of the space of fields we consider. For tangential vector fields, for example, k equals twice the number of triangles of the mesh.

Laplacian for vector fields A discrete Hodge–Laplace operator Δ for piecewise constant vector fields is discussed in [12]. The operator combines discrete divergence and curl operators with the gradient and a 90-degree rotation in the tangent plane, which we denote by J,

$$\Delta = -\mathrm{grad}\,\mathrm{div} - \mathrm{J}\,\mathrm{grad}\,\mathrm{curl}^{*}. \tag{3.1}$$

The divergence and curl operators map piecewise constant fields to piecewise linear polynomials and the gradient maps piecewise linear polynomials to piecewise constant vector fields. Matrix representation of all involved operators are described in [12]

For the piecewise constant fields on a mesh, a Hodge decompositions can be defined [55, 73]. This is an orthogonal decomposition of the space of piecewise constant fields in gradients and co-gradients (J grad) and harmonic fields. For the decompositions, two function spaces are needed: the space of continuous, piecewise linear polynomials (linear Lagrange finite elements) and the space of edge-midpoint continuous, piecewise linear polynomials (linear Crouzeix–Raviart elements). The Hodge–Laplacian can be constructed such that it respects the decomposition, which means that it maps gradient fields to gradient fields, co-gradient fields to co-gradient fields, and has exactly the harmonic fields in its kernel. To achieve this, one of the div and curl operators has to map to the space of continuous, piecewise linear polynomials and the other one to the edge-midpoint continuous, piecewise linear polynomials. In equation (3.1), we indicate that the curl operator maps to the space of edge-midpoint continuous functions by adding an asterisk.

Laplacians for n-fields Laplace operators for n-fields were proposed for a vertex-based representation in [40] and for face-based representation in [22, 11]. For our experiments, we use the face-based Laplacian. It computes differences of the n-vectors of each triangle to the n-vectors of the neighboring triangles. For this, the n-vectors of the neighbor triangles are parallelly transported to the corresponding triangle. In order to be able to form differences between n-vectors, a linear structure for n-vectors is required. This can be obtained using the concept of the representation vector of an n-field [61, 50]. The n-vectors are first converted to the corresponding representation vectors, then the difference is computed and the result is converted back to an n-vectors. For the choice of weights for the differences and a matrix representation of the Laplace operator, we refer to [11].

Laplacians for tensor fields We introduce a discrete Laplace operator for piecewise constant tensor fields on surface meshes. In this paragraph, we provide an overview of the construction and discuss details in the appendix. The operator is a weighted finite difference operator on the triangles of the mesh. To compute a difference between the tensor A_i of triangle T_i and the tensor A_j of a neighbor triangle T_j, we transport A_j parallelly to triangle T_i. The transport of a tensor to its neighboring triangle varies for the different tensor types depending on how the tensor transforms from one basis to another. We illustrate this at the example of $(1,1)$-tensors in the appendix. To describe the construction of the Laplace operator, we denote the transport of the tensor A_j to the triangle T_i by τ_{ji}. The Laplacian of a tensor field A is again a tensor field. In the triangle T_i the tensor field ΔA is given by

$$(\Delta A)_i = \frac{1}{m_i} \sum_{j \in N_i} w_{ij}(A_i - \tau_{ji}(A_j)), \tag{3.2}$$

where N_i is the list of the three neighbors of triangle T_i and

$$w_{ij} = \frac{3\,(\mathrm{length}(e_{ij}))^2}{\mathrm{area}(T_i \cup T_j)} \quad \text{and} \quad m_i = \mathrm{area}(T_i)$$

are weights depending on the geometry of the triangles T_i and T_j. In the appendix, we show how Voigt's notation can be used to derive a linear representation of the tensors and the transport operator τ_{ij}. This can be used to construct the stiffness and mass matrices for this Laplacian.

3.4. SPACES OF LOCALLY SUPPORTED FIELDS

In this Section, we describe our construction of subspaces of the spaces of tangential vector, n-vector and tensor fields. Our construction is based on a general principle that can be applied to the different types of tangential fields. We first introduce the construction of individual fields, then we describe how the field construction can be used to assemble bases of subspaces.

Field construction The input to the field construction are a Laplace operator, given by a stiffness matrix S and a diagonal mass matrix M, and a subset D of the set of triangles of the mesh that serves as the support of the field. Depending on whether the Laplacian operates on vector, n-vector or tensor fields, corresponding fields are constructed. Our approach is to compute the m lowest eigenfields of the Laplace operator subject to the constraint that the field is zero for all triangles that are not in the subset D. These fields can be characterized as the minimizers of the optimization problem

$$\begin{aligned} &\min_{\Phi \in \mathbb{R}^{k \times m}} \mathrm{tr}\left(\Phi^T S \Phi\right) \\ &\text{subject to } \Phi^T M \Phi = Id \quad \text{and} \\ &\Phi_{ij} = 0 \text{ if } i \text{ belongs to a triangle not in } D. \end{aligned} \tag{3.3}$$

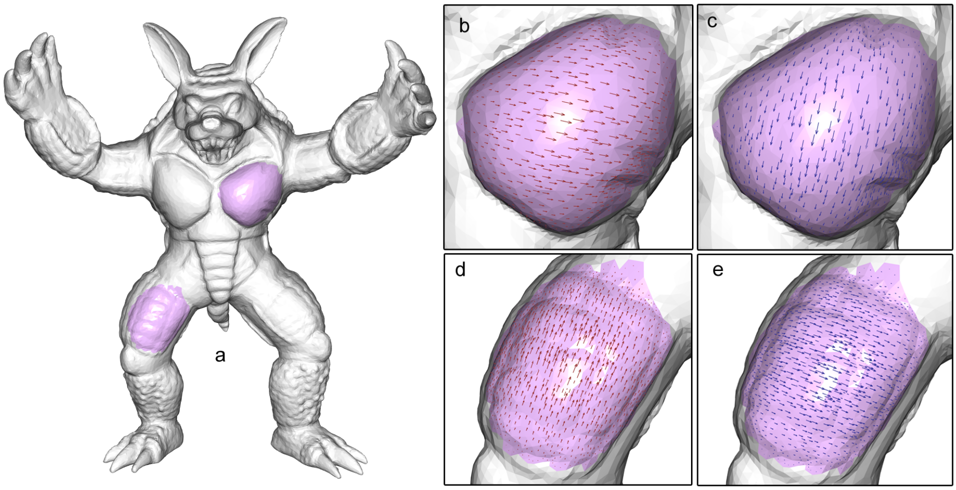


Figure 1: Examples of localized tangential vector fields computed with our approach are shown. On the left, the locations of the support areas of the fields, and, on the right, four fields are shown.

Each column of the minimizer Φ describes a localized field. The first constraint ensures that the fields are M-orthonormal and the second constraint ensures that the fields vanish outside of the specified region.

Since the second constraint is linear, we can specify a basis for the space of vector fields that satisfy the constraints. We construct a matrix $V \in \mathbb{R}^{k \times k_D}$ whose columns form a basis of the space of admissible fields, where k_D is the dimension of the space of admissible fields. This matrix has one non-zero entry per column and the entries are located at the degrees of freedom of the vector associated with the selected triangles. Each entry takes the value $1/\sqrt{M_{ii}}$, where M_{ii} is the diagonal entry of the mass matrix M and i is the row index of the entry. The matrix V allows us to parametrize the space of admissible fields, *i.e.* for any admissible field $X \in \mathbb{R}^k$ there is a corresponding $x \in \mathbb{R}^{k_D}$ such that

$$X = Vx. \tag{3.4}$$

We consider the restricted stiffness and mass matrices

$$\bar{S} = V^T S V \qquad \text{and} \qquad \bar{M} = V^T M V. \tag{3.5}$$

By our construction of V, the restricted mass matrix $\bar{M}$ is the $k_D \times k_D$ identity matrix.

Using (3.4) and (3.5), we can rephrase the optimization problem (3.3)

$$\min_{\phi \in \mathbb{R}^{k_D \times m}} \operatorname{tr}\left(\phi^T \bar{S} \phi\right) \tag{3.6}$$

$$\text{subject to } \phi^T \phi = Id.$$

Benefits of this formulation are that we reduced the problem's dimension to $k_D\, m$, simplified the second constraint, and eliminated the third constraint.

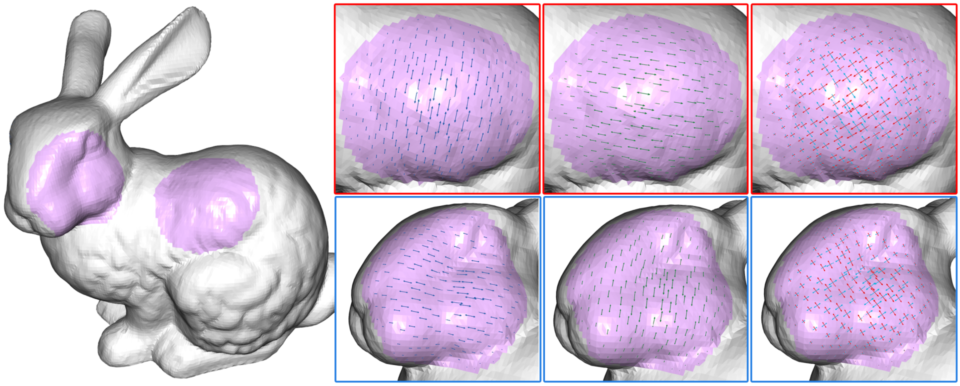


Figure 2: Examples of localized tensor fields computed with our construction are shown. For each tensor, the shown sticks point in the two eigendirections and the lengths of the sticks are proportional to the absolute values of eigenvalues of the tensor.

The problem (3.6) is a sparse eigenvalue problem and the solutions are pairs (λ_i, ϕ_i) satisfying the equation

$$\bar{S}\phi_i = \lambda_i \phi_i. \tag{3.7}$$

The solutions $\phi_i \in \mathbb{R}^{k_D}$ are mapped to the corresponding vectors $\Phi_i \in \mathbb{R}^k$ by multiplying them with the matrix V. Examples of local vector and tensor fields are shown in Figures 1 and 2.

Subspace construction To construct subspace bases using the field construction, we need to define the support regions for the individual basis fields. Our approach is to cover the surface with disk-shaped regions. In this paragraph, we discuss a uniform distribution of the regions and extend the approach to adaptive distributions of the regions in the last paragraph of this section.

Each region is an approximate geodesic disk. All disks have the same radius r and the value of r is chosen such that the disks have sufficient overlap. We will discuss the choice of r we used for our experiments in Section 3.6. To place the disks on the surface, we sample triangles of the mesh that serve as the centers of the disks. To distribute the sampling uniformly, we use a furthest point sampling scheme. The distance is measured by a weighted Dijkstra algorithm that operates on the mesh's dual graph. The nodes of this graph are the mesh's triangles and there are edges between nodes if the faces are neighbors. The weights for the edges are the geodesic distances of the barycenters of the triangles. Examples of resulting samplings are shown in Figure 4.A.1. After placing the samples, we define the regions associated with the samples using a region growing algorithm. The algorithm is using the Dijkstra distance and grows the regions until the distance r is reached.

To ensure the geodesic disks cover the whole surface, we can use the distance ρ of the last sample placed by the furthest point sampling. If r is larger than ρ, then the disks cover the whole surface. In practice, we choose r much larger than ρ as we want the disks to have sufficient overlap.

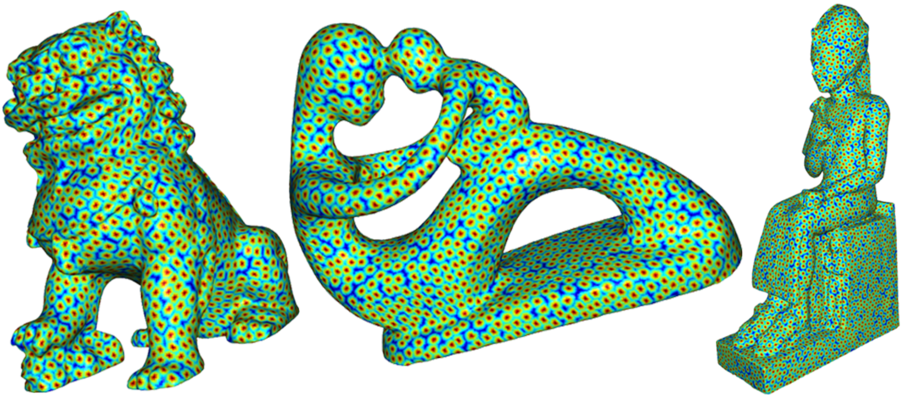


Figure 3: Examples of farthest point samplings (1k-5k samples) constructed on various mesh models (50k-2m faces).

Once the regions are defined, we compute m eigenfields for each region by solving the sparse eigenvalue problem (3.7). The choice of m depends on the type of field. For example, for vector fields and n-fields, we set $m = 2$, and, for (1,1)-tensors, we set $m = 3$. This choice is based on the multiplicity the lowest eigenvalue of the corresponding Laplace operator and our experimental results as discussed in Section 3.6. The size k_D of the sparse matrix occurring in the eigenvalue problem depends on the number of triangles belonging to the region. The resulting k_D-dimensional eigenvectors ϕ_i describe the fields in the regions. We lift the ϕ_is to vector fields Φ_i defined on the whole surface using (3.4) and stack the lifted fields Φ_i as the columns of a $k \times d$ matrix U. Here d denotes the total number of fields that are constructed, which is m times the number of regions. Since each Φ_i vanishes outside of its support region, the matrix U is sparse.

Scalability Our subspace construction is designed to be scalable, meaning that we want to be able to efficiently construct and store large subspaces on large meshes. Our motivation to aim for a scalable construction is that we want to be able to obtain subspaces that, on the one hand, include enough degrees of freedom to support general purpose design and processing tasks, and, on the other hand, allow to control the size of the optimization problem independently of the resolution of the meshes that are used.

One important feature that makes the method scalable is the localized support of the basis fields. For storing the basis, this means that the vector representing the basis fields are sparse vectors. The size of the support of the fields needs to be large enough such that there is sufficient overlap of each basis field with some other fields. On the other hand, a too large support is less efficient in terms of storage requirements. This means that if we construct two spaces with different dimension on the same surface, then the individual basis fields of the larger space will have a smaller support. Since the basis fields are stored as sparse vectors, this implies that the higher dimensional space requires more basis fields to be stored while each basis fields requires less storage. In our experiments, we found that the storage requirements for storing spaces of different dimension on the same mesh are approximately the same. This enables us to work with large subspaces with 500 to 5000 or even more dimensions. In addition, the method allows us to construct spaces on larger meshes, as the computation of the individual fields only requires solving an eigenvalue problem for a small region of the surface (the support of the field). These properties are advantages of our construction over eigenbases of Laplace operators, which have higher storage requirements, as a large dense matrix must be stored, and require higher computational costs for solving the large scale eigenvalue problems.

Adaptive subspaces Adaptivity can be integrated to the basis construction in a way that is simple to implement. The sampling method and the size of the regions depend on the weighted Dijkstra distance on the dual graph. For the uniform construction, the edge weights are chosen according to the geodesic distances of the barycenters of the triangles. To make the method adaptive, we change the edge weights in the dual graph. Changing the edge weights in some part of the surface affects the sampling and the sizes of the support regions of the basis fields. For example, when the weights are increased in some part of the surface, the sampling in the part of the surface becomes denser and the support regions of the basis fields decrease. The resulting subspace has more degrees of freedom in the part of the surface where the weight is increased, and, therefore, can better represent fields that have high frequency features and details in these areas. If the weights are reduced, fewer and larger regions are constructed in the corresponding part of the surface. For the subspaces, this means that fields with little detail in the corresponding part of the surface are represented more efficiently. The adaptive construction is simple to implement as after rescaling of the edge weights of the dual graph, the same algorithm as in the uniform case is executed. Scaling factors for the edges can be obtained for example from user input or an analysis of example fields.

3.5. SUBSPACE METHODS

The main goal of our construction is to enable subspace methods for vector, n-vector and tensor field design and processing. In this section, we will discuss a model problem that we will later use as part of the evaluation of our subspace construction.

We consider the following least-squares problem

$$\min_{X\in\mathbb{R}^n} \left(\mu_S X^T S X + \mu_B X^T B X + \mu_C \|CX - c\|^2\right), \tag{3.8}$$

where C is a matrix that defines weak constraints, c specifies the values of the constraints, the μs are a positive weights, M is the mass matrix, S is the stiffness matrix of the Laplacian and $B = SM^{-1}S$. The first two summands are the harmonic and biharmonic energies of X, which act as regularizers. This type of problem arises in field design tasks, for example, when fields are modeled with a stroke-based user interface. The resulting fields should align with the strokes but not follow them exactly. Other examples of applications that can be formulated as in (3.8) are smoothing of an input fields and extrapolating fields that is given only on parts of the surface to fields defined on the whole surface. For the smoothing application, C is the identity matrix and c the input field. For the stroke-based design and the extension of the field, C is a selector matrix that selects the vectors (n-vectors, tensors) of the parts of the surface where the input field is defined

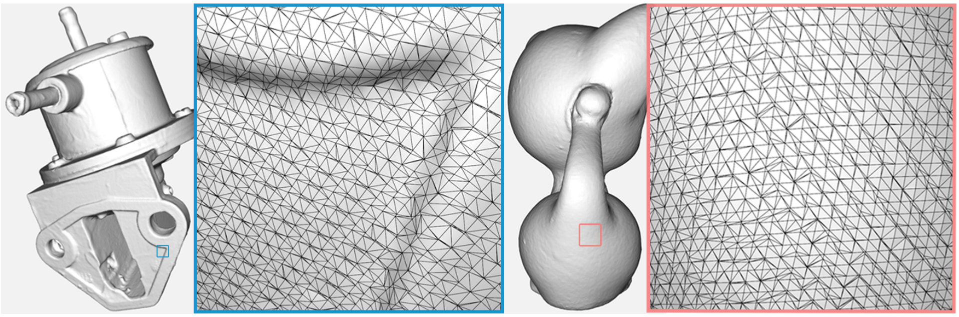


Figure 4: We tested our approach on irregular meshes having many acute angles.

and c specifies the vectors of the field in these regions. The minimizer of (3.13) satisfies

$$(\mu_S S + \mu_B B + \mu_C C^T C)X = \mu_C C^T c. \tag{3.9}$$

The reduced minimization problem, which restrict the optimization to the subspace, is

$$\min_{x\in\mathbb{R}^d} \left(\mu_S\, x^T U^T S U x + \mu_B\, x^T U^T B U x + \mu_C \,\|CUx - c\|^2\right). \tag{3.10}$$

The solution can be computed by solving the system

$$U^T(\mu_S S + \mu_B B + \mu_C C^T C)Ux = \mu_C U^T C^T c. \tag{3.11}$$

The advantage of the reduced problem is that (3.11) is a sparse low-dimensional system. In particular, the system is independent of the resolution of the mesh and only depends on the dimension of the subspace. The scalability of our basis construction allows for working with spaces of several thousand dimensions, which provide a rich space for design and processing problems. At the same time, solving the reduced problems only takes few milliseconds which enables interaction and interactive steering of parameters. For example, a user can change the parameters μ_S, μ_B, and μ_C and receive immediate feedback. In contrast, without reduction, any parameter adjustment requires solving a large-scale sparse linear system, which is prohibitive for interactive applications.

The reduction of the model problem (3.8) can be extended to include more features. For example, hard constraints can be efficiently included using a Schur complement approach. We refer to [12, 11] for details. The subspaces can also be used to reduce the complexity of general non-linear problems. Reducing the problem's dimension lowers the computational cost of minimization steps and accelerates the convergence of solvers. However, a difference to the model problem is that the evaluations of a non-linear objective and its gradient and Hessian still depend on the complexity of the mesh. Methods for fast approximation of a non-linear objective and its derivatives have been proposed, for example, in the context of real-time simulation [1, 68, 75, 9]. In this chapter, we use the model problem (3.8) for evaluating the quality of the proposed subspace basis and leave the adaption of fast approximation schemes for non-linear objectives as future work.

	Basis construction (secs.)				Sparsity	
Subspace dim.	Sampling	Patches	Eigensolve	Total	#nnz U	#nnz $\bar{S}$
500	1.6	82.9	168.8	253.4	80.4m	78.3k
1k	1.9	84.0	125.9	211.8	80.4m	163.0k
2k	2.4	80.5	126.2	209.1	80.4m	333.0k
5k	3.5	77.2	152.1	232.8	80.4m	854.5k
10k	5.3	80.1	182.6	267.9	80.5m	1.7m
20k	8.4	103.1	283.7	395.3	80.5m	3.5m

Table 1: Scalability analysis of our basis construction. Computation times and numbers of non-zero entries (#nnz) of the matrices storing the basis (U) and the restricted stiffness matrices ($\bar{S}$) for subspaces of different dimension on the Bimba model with $1m$ triangles are shown.

3.6. EXPERIMENTS

In this section, we discuss our experimental evaluation of our subspace construction focusing on scalability, approximation and adaptivity. We tested our approach on different meshes including some with low mesh quality exhibiting a large number of triangles with acute angles. Figure 4 shows examples of meshes we used.

Implementation We implemented our subspace constructions using the *Eigen* [27] and *LibIGL* [36] libraries. The basis fields of the subspaces are constructed in parallel. To solve the local eigenproblems, we use the *SpectrA* library [57] with Cholmod's supernodal sparse Cholesky decomposition [15] being applied to solve the linear systems. CUDA's GPU-based *cuSparse* is employed to lift the fields from the reduced space to the full space, which requires matrix-vector multiplication with the sparse matrix U that stores the subspace basis. To solve the linear systems in the design and processing applications, we use Pardiso's symmetric indefinite factorization [53].

Scalability In our experiments, we evaluated the memory requirements for storing the subspace basis and the computation times required for the construction of the basis for large subspaces and also for larger meshes. When reporting results, we state the subspace dimension we worked with. Since we compute a constant number of basis fields per region, 2 for vector fields and n-fields and 3 for (1,1)-tensor fields, the number of regions is half of the subspace dimension for vector and n-fields and a third for the tensor fields.

In the first experiment, we constructed subspaces of different dimension d ranging from 500 to $20k$ on a mesh with $1m$ triangles and report the number of non-zero entries of the subspace bases as well as computational times required for basis construction. The data is summarized in Table 1. For each subspace dimension, we need to choose a proper value r for the radii of the geodesics disks. It is important to choose r large enough such that the individual vector fields can interact with their neighbors and information can spread. On the other hand, a too large value of r makes the basis less

Mesh	#Faces	Dim.	Basis Construction (in seconds)			
			Sampling	Patches	Eigensolve	Total
Kitten	274k	1k	0.54	18.32	38.70	57.56
		2k	0.67	18.88	34.83	54.37
		10k	1.43	22.45	45.53	69.41
Fertility	483k	1k	1.10	35.63	64.42	101.16
		2k	1.33	32.52	64.97	98.82
		10k	2.70	41.67	77.95	122.33
Bimba	1m	1k	1.94	83.98	125.90	211.83
		2k	2.41	80.48	126.20	209.09
		10k	5.25	80.06	182.56	267.87
Ramses	1.65m	1k	4.50	178.81	261.89	445.20
		2k	5.20	168.17	222.36	395.73
		10k	10.70	159.61	347.29	517.61
Isidore horse	2.21m	1k	5.02	213.02	366.57	584.60
		2k	6.10	203.36	286.89	496.35
		10k	13.25	199.49	457.64	670.38

Table 2: Timings for the constructions of subspaces of different dimension on various meshes are shown. Computation times for farthest point sampling (Sampl.), construction of the local patches and corresponding matrices (Patches), solving the eigenproblems (Eig. solv.), and the total time for all three steps (Total) are shown.

efficient as more storage is required. Explicitly, we set

$$r = \sqrt{\frac{\sigma A}{d\pi}}, \tag{3.12}$$

where A is the area of the surface, d the dimension of the subspace and σ a parameter, which we set to 40 for this experiment. The motivation for using this formula is that we want to find the radius r such that the combined area of all disks is σ times the area of the surface. To arrive at a simple formula, we replaced the average area of the geodesic disks by $r^2\pi$, which is the the area of the Euclidean disk of radius r. In this experiment, setting $\sigma = 40$ results in matrices U whose average number of non-zero entries per row is about 40. This means that in average every triangle is in the support of 40 basis fields.

The total time required for basis construction is listed in the fifth column of the table. The higher the dimension of the subspace, the more eigenvalue problems need to be solved. On the other hand, each of the eigenvalue problems is smaller as the support of the fields decreases. The total time for constructing the bases for the different subspaces on a regular desktop computer is between 3.5 and 7 minutes. The dimensions are between 500 and $20k$ and the underlying mesh has $1m$ triangles.

In the second experiment, we constructed $1k$, $2k$, and $10k$-dimensional subspaces on meshes with a number of triangles in the range of $274k$ to $2.21m$ and measured the time needed for basis construction. Experimental results are summarized in Table 2. As in the first experiment, we observe that the computation time required for the construction of subspaces of different dimension on the same mesh changes only slightly.

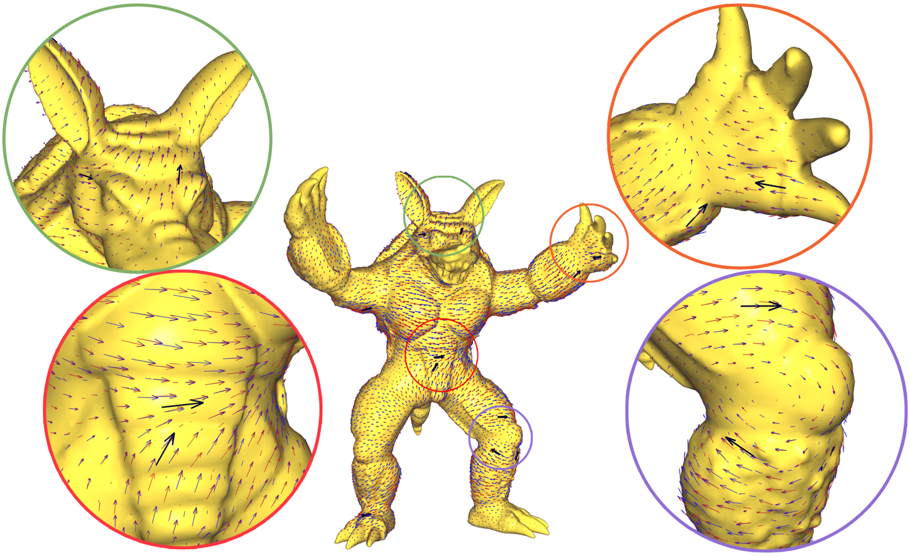


Figure 5: Visual comparison of a reference vector fields (blue) and its projection to a $2k$-dimensional subspace (red). The relative L^2 approximation error is 1.93×10^{-2}.

When comparing the construction time for the different meshes, we observe an increase in computation time that is approximately linear in the number of triangles.

Approximation In addition to the scalability of the basis construction, the approximation quality of the resulting subspace is important. In the first series of experiments, we evaluate the approximation quality by projecting a set of tangential vector fields to the subspace and computing the relative L^2-norm of the difference between the input field and projected field. For different meshes, we created sets of 50 test fields by placing 10-50 interpolation constraints at random locations on the surfaces and using the vector field construction method from [12] to generate fields interpolating the constraints. For

Mesh	Method	Ours	Biharmonic	Variant of ours	Patched Eigenfields	Gradients
Armadillo	L^2-proj.	0.02	0.05	0.06	0.14	0.16
	Minim.	0.04	0.21	0.11	0.25	0.48
Chinese Lion	L^2-proj.	0.02	0.07	0.06	0.10	0.19
	Minim.	0.03	0.18	0.09	0.18	0.46
Fertility	L^2-proj.	0.02	0.03	0.04	0.71	0.59
	Minim.	0.02	0.09	0.10	0.71	0.65

Table 3: Approximation errors of L^2-projection of vector fields to a $2k$-dimensional subspaces and solutions of the optimization problem (3.8) for subspaces resulting from the proposed subspace construction and four possible alternative constructions.

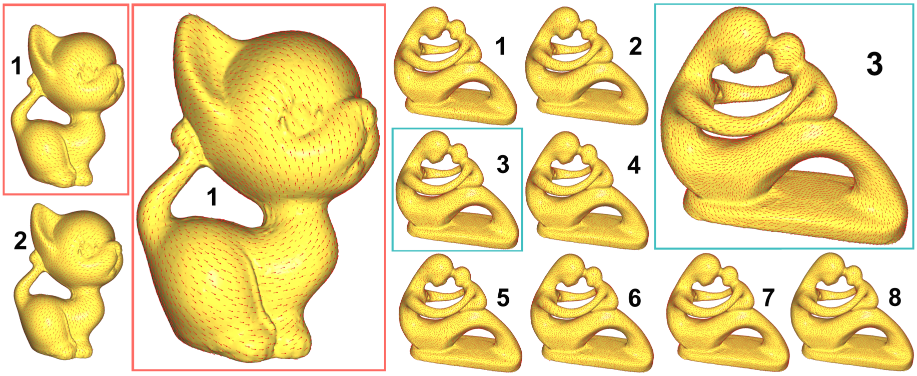


Figure 6: Approximations of harmonic fields in subspaces on models with non-trivial genus (Kitten, genus=1 and Fertility, genus=4).

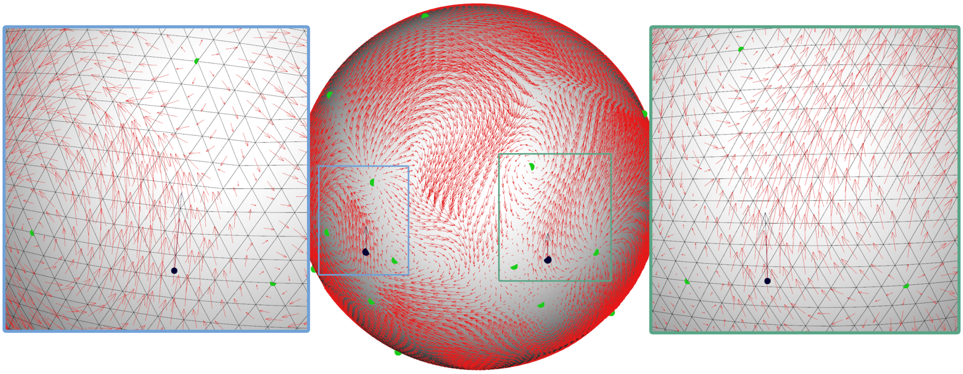


Figure 7: Example showing the placement of singularities in the subspace.

an input field X on the surface, the projection x to the subspace with basis U is defined as the minimizer of the quadratic objective

$$\|X - Ux\|_M^2. \tag{3.13}$$

The relative L^2-error is $\|X - Ux\|_M / \|X\|_M$. Table 3 shows the resulting average errors for 50 test fields on three different meshes. For all three models, the relative L^2 error using a 2k-dimensional subspace is 2 percent. Figure 4.C.3 shows an overlay of a test field and its projection on the Armadillo mesh. In addition to evaluating the projection error, we also compare the solutions of the optimization problem (3.8) and solution of the corresponding reduced problem (3.10). The relative error of the optimization problem is between 2 and 4 percent as listed in Table 3 (rows labeled 'Minim.'). Based on the comparisons to alternatives and variations of our construction, which are discussed in Section 3.7, we consider this a very good approximation quality.

In the second experiment, we measured how the relative L^2 approximation error changes with increasing size of the subspace. The test fields were generated in the same

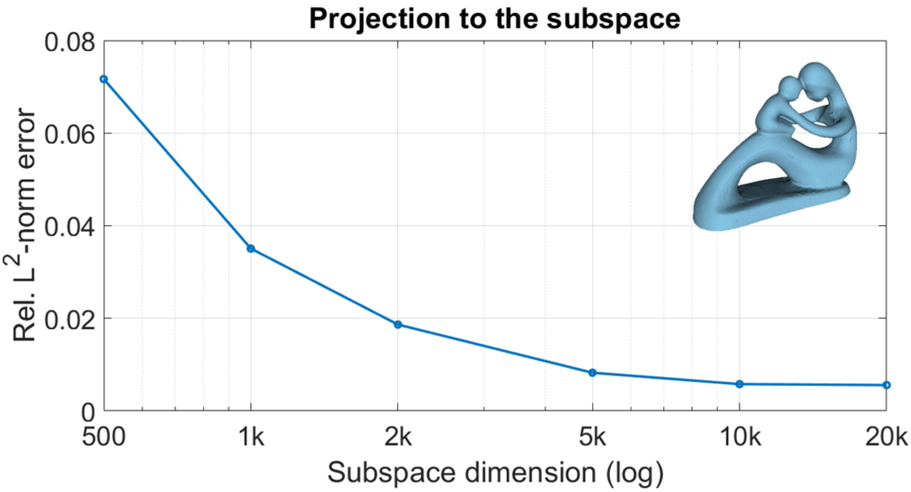


Figure 8: The relative L^2 approximation error for subspaces of different dimension is shown.

way as in the previous experiment. We constructed 500-20k dimensional subspaces on the Fertility mesh (483k triangles). To set the radii of the geodesic disk, we use equation (4.10) and set $\sigma = 80$ for all spaces. Results are shown in Figure 8. The results illustrate that the approximation error can be reduced when subspaces of higher dimension are used. The approximation error will vanish once the subspace dimension equals the dimension of the full space. As increasing the dimension of the subspace causes higher computational cost for constructing the subspace and solving the reduced problems, in practice one needs to find a compromise between expressiveness of the space on the one hand and computational costs on the other hand.

In the third experiment, we tested whether the subspaces of vector fields we construct include approximations of the harmonic fields. On smooth surfaces the harmonic fields are the eigenfields of the Hodge–Laplace operator with vanishing eigenvalue. A surface of genus g has $2g$ linearly independent harmonic fields. The discrete Hodge–Laplacian (3.1) is designed such that it preserves this structure and has a $2g$-dimensional kernel of discrete harmonic fields. We wanted to test whether our subspaces contain approximations of these fields. In our experiments, we obtained $2g$ approximate harmonic fields with very small, though not vanishing, eigenvalues. Examples of these fields are shown in Figure 6. Additionally, Figure 19 shows plots of the approximate eigenvalues for some meshes, two genus zero meshes and the genus one Kitten model. In the lower left corner, the first eigenvalues are shown. For the Kitten model, the first two values are are the eigenvalues of the approximate harmonic fields.

In the fourth experiment, we evaluated the capabilities of the subspaces to support the placement of singularities. In Figure 7, we show an example of a vector field in a 200-dimensional subspace on which singularities at certain locations on the surface are enforced using the approach discussed in [11].

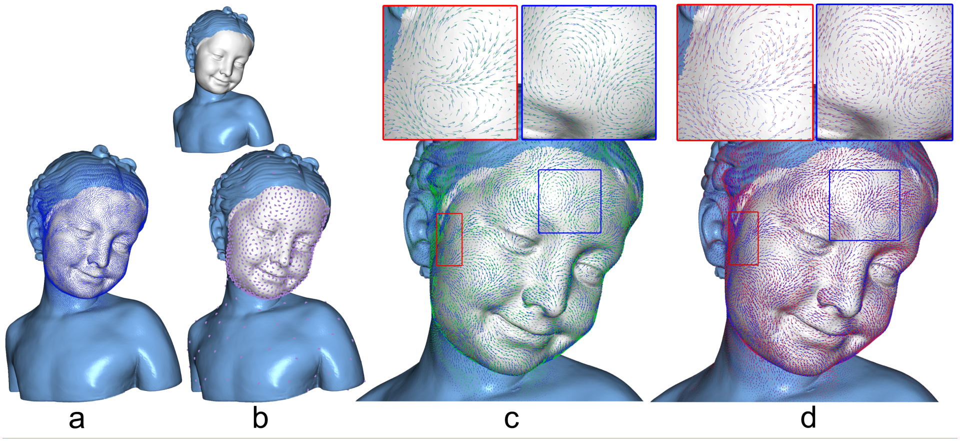


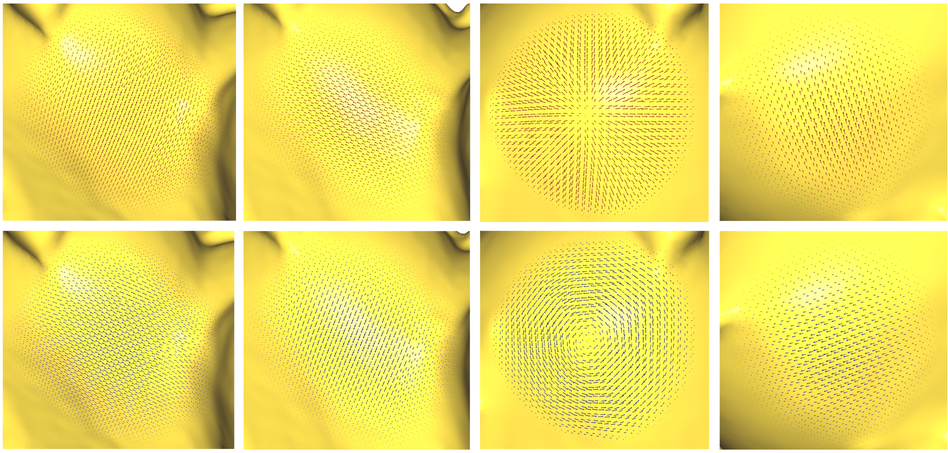
Figure 9: Results of an experiment with our adaptive subspace construction. The figure shows (on top left) the Bimba mesh with $1m$ triangles and a selected region on the surface, (a) a vector field on the surface, (b) the sampling used for the adaptive subspace construction, (c) an overlay of the vector field in blue and its L^2-projection to an adaptive subspace in green, and (d) an overlay of the vector field in blue and its L^2-projection to a uniform subspace in red.

Adaptivity To explore the benefits of adaptive subspaces, we conducted an experiment in which a vector field, which has many features concentrated in some area of a surface and almost vanishes away from that region, is projected to a uniform and an adaptive subspace. The adaptive subspace is constructed by rescaling the weights of the dual graph in a region of the surface, which we defined by hand. Results of the experiment are shown in Figure 9. The figure illustrates the benefits of adaptive subspaces for the approximation of the fields. As a quantitative evaluation, we computed the relative L^2 approximation errors. The adaptive subspace yields a relative L^2 error of 9.2×10^{-2}, which compares to 35.2×10^{-2} for the uniform subspace.

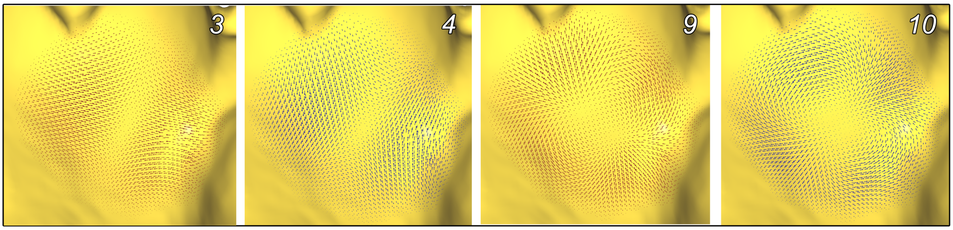
3.7. COMPARISONS

During the development of the proposed subspace construction, we implemented and tested various possible alternatives. In this section, we provide some comparisons of the proposed and possible alternative constructions. In addition, we compare our subspace construction with Laplace eigenfields.

Alternative constructions We report approximation results for four alternative constructions in Table 3. The first alternative is to use not only the lowest two eigenfields for every geodesic disk, but more. The column “Variant of ours” shows results for the case that the lowest 10 eigenfields are used for each geodesic disk. In order to get a fair comparison, fewer disks are used in total such that the dimensions of the subspaces are the same. The second alternative is to solve two biharmonic problems on each geodesic disk instead of the eigenproblems. For the biharmonic problems, we set interpolation constraints: we specify a unit vector in the center triangle of the geodesic disk and en-



Ours Biharmonic Gradients Patched Eigenf.



Variant of ours

Figure 10: Examples of basis fields resulting from the alternative constructions we compare to in Table 3. For the construction that involves 10 eigenfunctions per geodesic disk ("Variant of ours"), the 3rd, 4th, 9th and 10th eigenfields on a geodesic disk are shown.

force that the field vanishes for all triangles not in the geodesic disk. The latter constraint implements zero Dirichlet and Neumann boundary conditions for the biharmonic problem. Two fields are generated by specifying orthogonal vectors in the center triangle. For details on how to solve biharmonic problems with interpolation constraints for tangential vector fields, we refer to [12]. The results for this construction are listed in the column labeled "Biharmonic". The third construction makes use of radial basic functions on surfaces as described in [48]. The radial basis functions are defined on the surface and are localized. To obtain vector fields from the radial basis functions, we compute the gradients and their co-gradients. This construction is labeled "Gradients" in the table. The fourth alternative also uses the radial basis functions. The idea is to compute two eigenfields of the whole surface and to use the radial basis functions to scale the vectors of the fields. This results in localized fields, which we use as subspace bases. The approximation results for these bases are listed in the column labeled "Patched Eigenf." in the table. The experiments we performed are the same tests as discussed in the paragraph *Approximation* of Section 3.6. 2k-dimensional subspaces were used for all tests. In our experiments, the proposed basis construction outperformed the alternative construction by a large margin as also documented in Table 3.

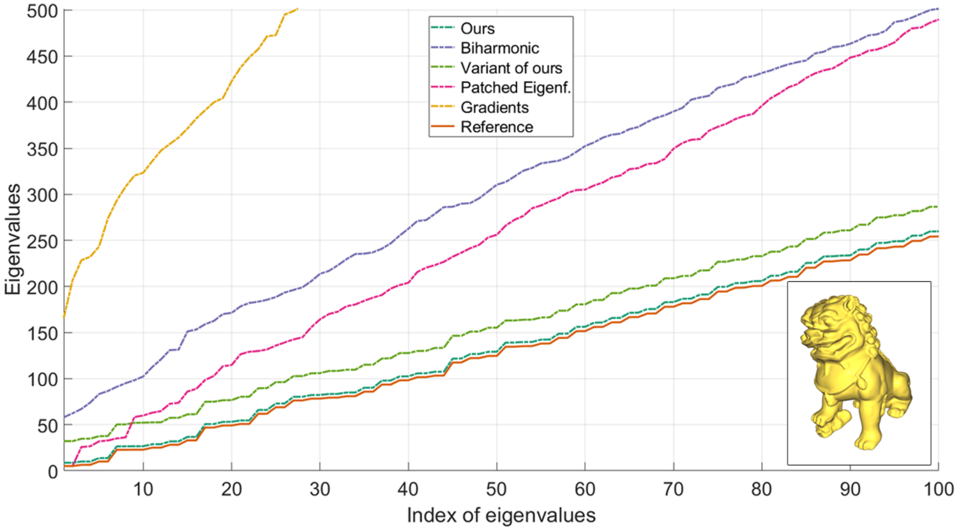


Figure 11: Approximation of eigenvalues using various alternatives of locally supported basis functions.

In the second experiment, we computed approximations of the lowest eigenvalues of the Laplace operator in subspaces constructed with the different schemes. For quantitative evaluations, we compare the results to the eigenvalues computed in the full space. Results are shown in Figure 11. Our construction very closely matches the true eigenvalues and outperforms all other constructions. Examples of basis fields resulting from the different constructions are shown in Figure 10. Though at first sight the fields obtained solving a biharmonic problem look similar to the field resulting from the proposed construction. A closer look reveals differences in the scaling of the vectors, which turns out to be important for the performance of the resulting subspaces.

Mesh coarsening In addition to the four alternatives discussed above, we can also use mesh coarsening for constructing subspaces. We compared the performance of the proposed construction to a mesh coarsening scheme that we developed. As for the approximation experiments in Section 3.6, we evaluated the relative approximation error for L^2 projection and the residual for the minimization problem (3.8). Results are shown in Table 4. In all our experiments, the results of the proposed method are significantly better than those of the coarsening approach. Figure 12 illustrates the coarsening-based subspace construction and includes a visual comparison of results obtained with our construction and the coarsening-based construction. The coarsening-based approach starts with coarsening of the mesh. To map a vector field on the coarse mesh to the fine mesh, we find for every triangle on the fine mesh the closest triangle of the coarse mesh and project the vector of the coarse mesh to the plane containing the fine triangle. The images of the canonical basis fields on the coarse mesh provide us with a subspace on the fine mesh.

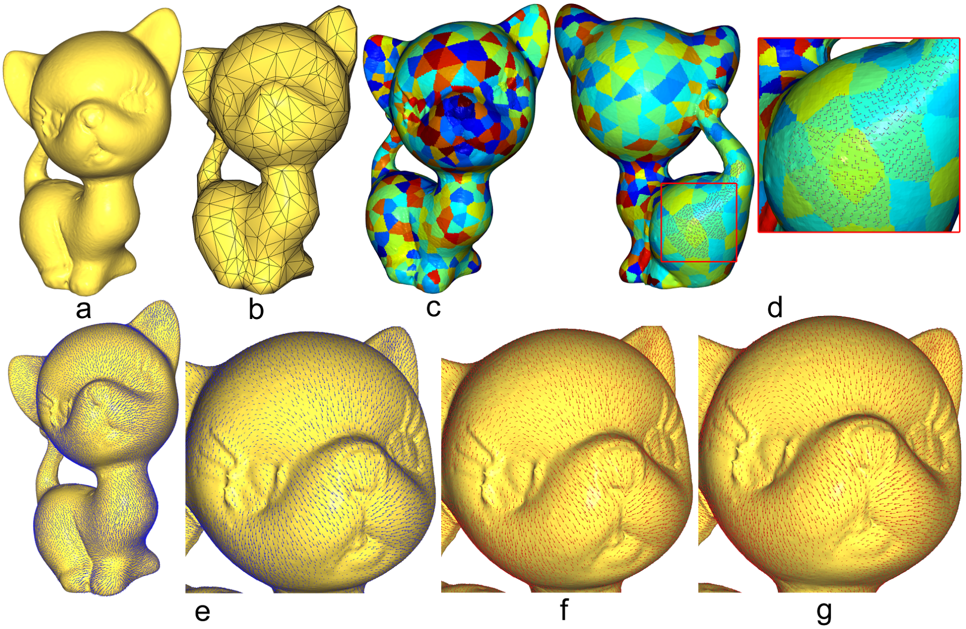


Figure 12: Comparison to a subspace construction based on mesh coarsening. The coarsening approach is illustrated in images (a)-(d). For comparison, a smooth tangential field (e) is projected onto the coarsening-based subspace (f), and onto the subspace proposed in this work (g).

Laplace eigenbasis To our knowledge, the only alternative construction of subspace of spaces of tangential vector and n-vector fields on surfaces are the eigenfields of vector field Laplacians and n-field Laplacians. In our experiments, we compared the eigenbases against our construction. The approximation quality of the eigenbases also serves as a baseline as using eigenmodes is a common subspace construction method in other contexts. While low-dimensional subspaces constructed from eigenfields can approximate smooth fields well, the eigenfields are globally supported. This means storing an eigenbasis is expensive as a large dense matrix must be stored. As a result, the eigenbasis cannot compete with the proposed construction in terms of scalability. In addition to storage requirements, the computation of the eigenfield is more costly than the local eigenproblems we need to solve for our construction. Table 5 shows results of our experiments that compare L^2 approximation error and residuals of the minimization problem (3.8) as well as the number of non-zero entries of the matrix storing the basis. For our construction the radius of the geodesic balls is set using formula (4.10) with $\sigma = 40$. The results show that an eigenbasis with just 40 eigenfields has similar storage requirement as our basis spanning a 2000-dimensional subspace. Concerning the approximation quality, however, a 2000d subspace resulting from our construction is much better than a 40d space spanned by the lowest eigenfields. To achieve a comparable approximation quality, more than 600 eigenfields are necessary. The storage requirements for such an eigenbasis is more than an order of magnitude higher than for 2000 basis fields

Model	Basis	L^2 Projection	Mimimization
Kitten	Ours	1.77×10^{-2}	2.75×10^{-2}
	Coarsening	14.8×10^{-2}	48.5×10^{-2}
CDragon	Ours	6.67×10^{-2}	3.12×10^{-2}
	Coarsening	25.6×10^{-2}	70.0×10^{-2}
Fertility	Ours	1.76×10^{-2}	2.27×10^{-2}
	Coarsening	20.0×10^{-2}	51.2×10^{-2}

Table 4: The comparison of the proposed subspace construction and a possible alternative construction based on mesh coarsening.

Model	#Faces	Basis	Dim.	#NNZ	L^2 proj	Minim.
Kitten	274k	Eigenf.	40	2.19×10^7	0.42	0.40
			250	1.37×10^8	0.06	0.06
			667	3.66×10^8	0.02	0.02
		Ours	2000	2.19×10^7	0.02	0.03
Fertility	483k	Eigenf.	40	3.87×10^7	0.37	0.36
			250	2.42×10^8	0.21	0.21
			667	6.45×10^8	Memory bound	
		Ours	2000	3.87×10^7	0.02	0.02
Bimba	1m	Eigenf.	40	8.04×10^7	0.33	0.33
			250	5.03×10^9	Memory bound	
			667	1.34×10^9	Memory bound	
		Ours	2000	8.04×10^7	0.02	0.03

Table 5: Comparison to eigenfields. Given the same storage, our locally supported bases outperform the eigenbases. Our method can achieve similar performance while requiring less storage.

computed with our construction.

Comparison to Instant Field Aligned Meshes In this paragraph, we discuss how our approach compares to the approach for real-time n-fields design that was introduced in [38]. They use an extrinsic energy, which combines fairness of the field and alignment to the surface curvature directions, to guide their n-field construction. In a precomputation, a multilevel hierarchy is constructed. Then, for field design, minimization steps with respect to the extrinsic energy are performed on the different levels of the hierarchy from coarse to fine. When used as an interactive design tool, the number of minimization steps per level are fixed in order to get fast responses. Due to the strict time constraint, the number of steps is typically not sufficient for the solver to converge to a minimum. We show a comparison of results in Figure 13. In the shown example, we compute smooth curvature aligned 4-fields on different meshes that approximate the same rockerarm surface. One can see that our approach produces more consistent fields for the different meshes compared to the approach from [38]. We refer the reader to [11] for additional evaluation of the Instant Field Aligned Meshes approach.

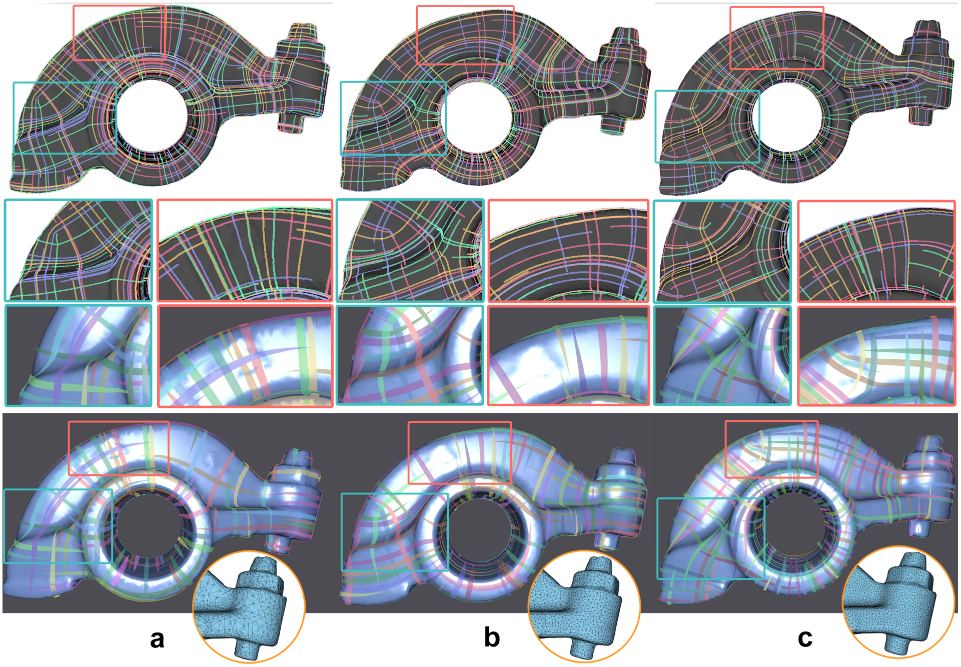


Figure 13: Comparison of smooth, curvature-aligned 4-fields computed on different meshes that approximate the same surface using the proposed techniques (top row) and the approach proposed in [38].

3.8. APPLICATIONS

In this section, we discuss some applications of the proposed subspace construction.

Vector fields design The tangential vector field design approach proposed in [21] computes fields that are on the one hand smooth and on the other hand align with input data such as strokes drawn by a user and an input field. This is modeled as a minimization problem similar to problem (3.8). Computing the minimizer requires solving a sparse linear system, whose size depends on the number of triangles of the surface. For fast solving, a sparse Cholesky factorization of the system is computed once and used for solving the systems. Cholesky updates are applied to update the system's matrix when the user changes the constraints. Though backsubstitution and the Cholesky updates are fast, they still scale with size of the mesh. As a result this approach enables interactive design only for a limited mesh resolution. Moreover, the interaction is limited to those operations that can be treated with Cholesky updates of the system's matrix. In contrast to this, the reduced system (3.10) is independent of the mesh's resolution. The proposed subspace construction allows the construction of subspaces that are large and therefore provide a rich modeling spaces to designers while keeping the computational cost for solving the reduced systems low. Figure 15 shows results of a modeling session using our approach, on the Antique Head model of 1.3m faces with a 2k-dimensional subspace. Our subspace methods provides interactive feedback. Timings are: 45ms for comput-

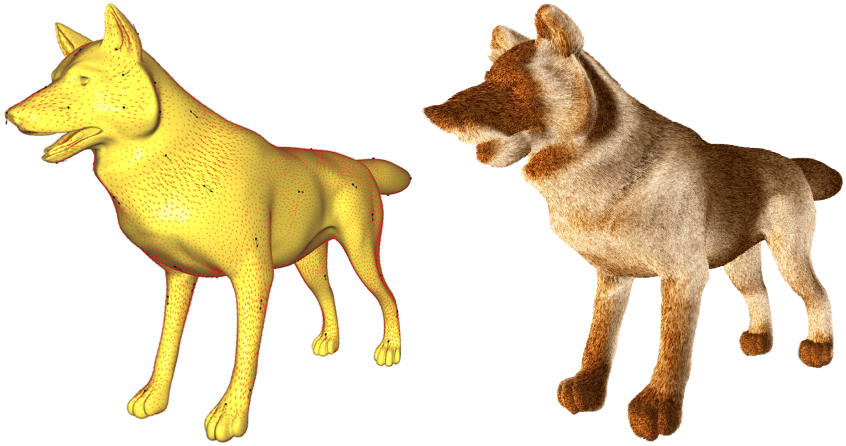


Figure 14: Interactive vector field editing on the Wolf mesh (left, 1m triangles, 2k dimensional subspace) and the resulting fur rendering (right).

ing a factorization of the reduced system, which includes the harmonic and biharmonic energies, $1.4ms$ for solving a system using the factorization and $23ms$ for mapping the reduced coordinates to a vector field on the surface. For comparison, we list corresponding timings for the unreduced system: $43s$ for computing the sparse factorization and $2.2s$ for solving a system using the factorization.

Fur design An application of the vector field design is fur editing. We used our subspaces in the fur editing approach proposed in [12]. Results for a modeling session that involves a Wolf mesh with $1m$ triangles and a 2000-dimensional subspace are shown in Figure 14. The tools allows to specify hard constraints and a Schur complement approach is used for solving the resulting reduced linear systems. The system requires less than $40ms$ for computing a solution when the interpolation constraints are modified.

n-field design The n-field design approach proposed in [11] computes fields that minimize a biharmonic energy subject to interpolation constraints imposed by a user. Additionally, a penalty for deviation from an input field can be included to the optimization. The approach requires solving a linear system and a reduction of the problem using eigenfields of the biharmonic energy is discussed in the paper. Since our subspace construction is more efficient for larger meshes and larger subspaces than the eigenbasis, see Section 3.7, our construction can be used to improve the scalability of the n-field design approach. Figure 16 shows results for n-fields design using our subspaces. In our experiments, the design tool achieved 30 fps when working with a $2k$-dimensional subspace on a model with $1m$ triangles.

Hatching n-field design can be used to control the stroke directions of line art renderings. We used our subspaces in combination with the approach for controlling hatchings of surfaces from [9]. The approach uses 2-fields to control hatching directions. The fields

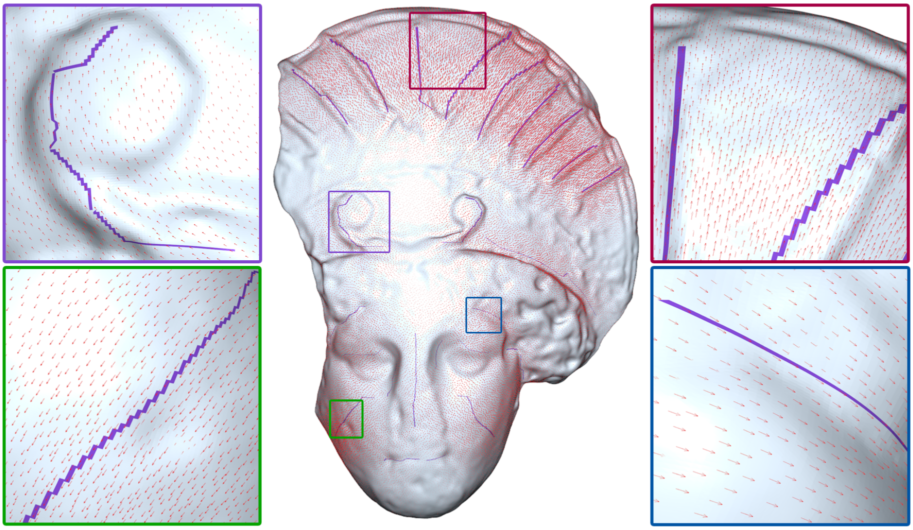


Figure 15: Interactive stroke-based vector field design on the Antique Head model (1.3m triangles, 2k subspace).

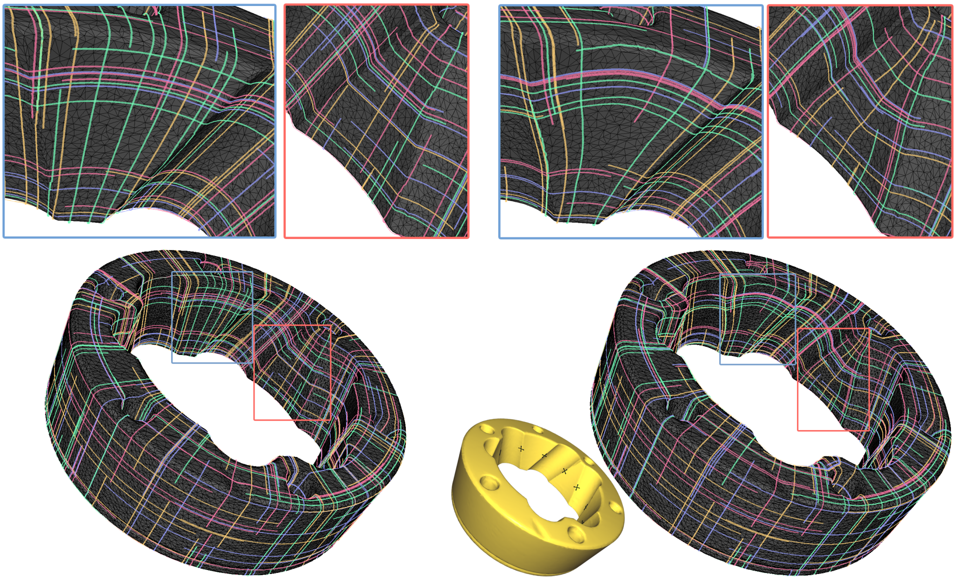


Figure 16: Example of 4-field design with interpolation constraints. Fields can be aligned to the curvature directions (left). Additional interpolation constraints allow users to modify the 4-field such that it better aligns to surface features (right). Our tool provides interactive responses when interpolation constraints are added, removed or modified.

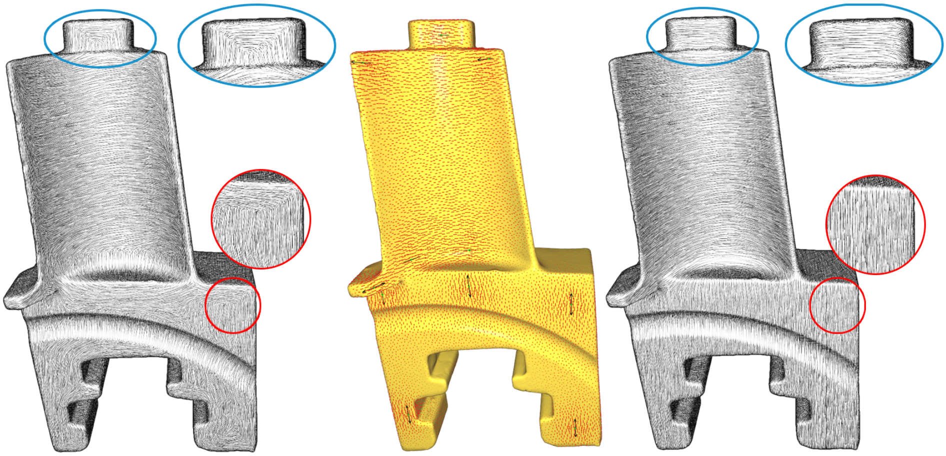


Figure 17: Hatching of the Blade model (390k faces) generated from 2-fields aligned to the maximal principal curvature (left). Modified fields, subject to interpolation constraints (right). The constraints are shown in the middle.

are first aligned to the maximum principal curvature directions of the surface. Then, the users can use interpolation constraints to modify the 2-field. Results are shown in Figure 17.

Tensor field smoothing We implemented a tensor field smoothing tool that minimizes a weighted sum of a function penalizing the deviation from the input field and the harmonic and biharmonic energies for tensor fields as fairness energies. The resulting optimization problem is analogous to problem (3.8). The tool uses our subspace construction to build a subspace and solves the reduced problem. This allows users to interactively adjust the weights for the three terms. To realize this, we precompute the reduced matrices for the three terms during the preprocessing stage, and, in the online stage, we build the weighted sum of the matrices and solve the resulting sparse linear system, whose size equals the dimension of the subspace.

We used the tool for smoothing the shape operator, whose eigenvectors are the principal curvature directions. Results are shown in Figure 18. We found this a useful tool for curvature computations, which usually need to be smoothed and require users to specify how strongly the field should be smoothed. With the tool, users receive immediate feedback when adjusting the smoothing parameters. For example, in the case of the Oil Pump model ($1.1m$ faces) shown in Figure 18, our reduced system (using a $2k$ dimensional subspace) requires $45ms$ to compute a factorization, $1.5ms$ to solve using the factorization, and $55ms$ to lift the reduced solution to a tensor field defined on the whole surface. For comparison, the timings for solving the unreduced system are: $49s$ to compute a factorization and $2.7s$ to solve using the factorization. We want to emphasize that many operations, such as changing the weights for the harmonic or biharmonic energy, require computing a new factorization as the system's matrix changes.

While we show an example, in which the user can only modify the weights, a more

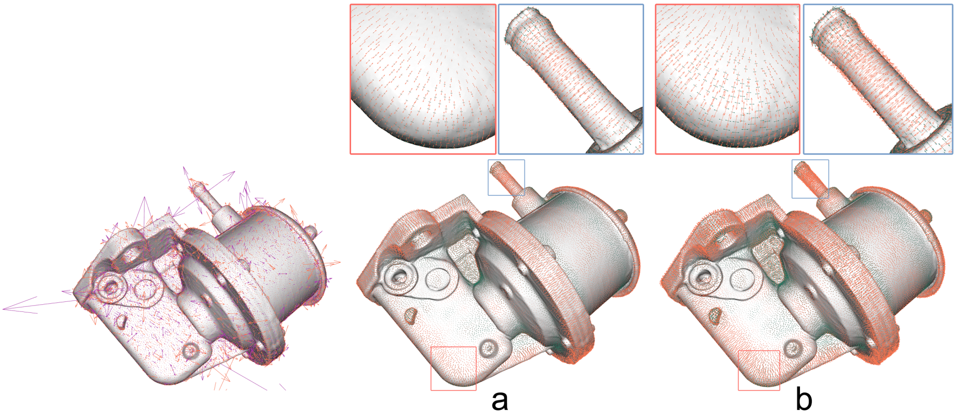


Figure 18: Results of our tensor smoothing tool. Curvatures computed on a mesh with 1m triangles (left). Smoothing results for different parameter settings (middle and right). When parameters are changed, new solutions are computed at interactive rates.

Model	Faces	Basis const.	Red. system	Eigen solve	Ref.	Speed up
Armadillo	86k	9	3.5	3.4	419	26.1
Ch.Dragon	255k	42	10.1	3.3	1376	25.0
Fertility	483k	99	17.7	3.4	Memory bound	Inf.
Ramses	1.6m	401	84.4	3.3		Inf.
Neptune	4.0m	1200	212.6	4.8		Inf.

Table 6: Timings (in seconds) for the approximation of 500 eigenfields are shown (3-5th) columns). For comparison timings for computing the unreduced reference eigenvalues (6th column) using MATLAB's sparse eigensolver are shown.

sophisticated interactive tool that allows users to specify different weights for different areas of the surface can be realized in a similar way. Another possible extension of this smoothing method would be a fast non-linear smoothing scheme that iteratively solves linear systems in the subspace.

Laplace spectrum Our subspaces can be used to efficiently compute approximations of the eigenvalues and eigenfields of Laplace operators. For computing approximations of m eigenfields, we construct a $2m$-dimensional subspace and compute the restricted eigenvalue problem in the subspace. This technique has been recently introduced for the case of eigenfunction of the Laplace–Beltrami operator in [48]. We refer to this paper for a description of the reduced eigenvalue problem, which is analogous to the reduced eigenvalue problem for eigenfields. Table 6 compares timings for solving the reduced and unreduced eigenproblems. Figure 19 compares the reference and approximated eigenvalues.

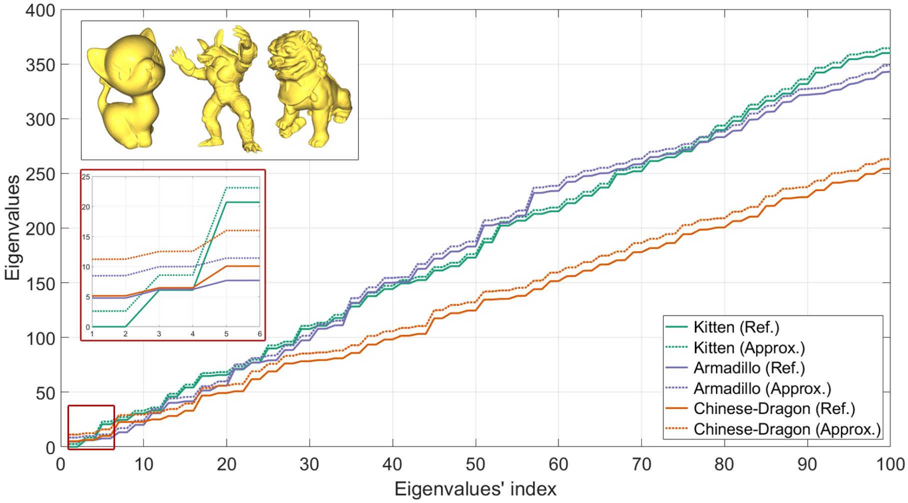


Figure 19: Approximation of eigenvalues of the Hodge–Laplace operator. For different meshes, approximated and reference eigenvalues are shown.

3.9. CONCLUSION

We introduce a construction of subspaces of tangential vector, n-vector, and tensor fields that is scalable and results in subspaces that can approximate smooth fields well. The construction can easily be extended to a construction of adaptive subspaces. We experimentally evaluate the approach and justify our construction by comparing it to possible alternative constructions. Finally, we discuss applications of our approach.

Challenges and limitations Our goal is to develop the techniques that enable interactive field design and processing tools that work on large meshes. The proposed subspace construction takes a step in this direction by enabling us to decouple the resolution of the meshes from the degrees of freedom used for the design and processing problems. In this work, we limit our focus to optimization problems with quadratic objectives. For more general problems, additional techniques need to be developed that allow us to approximate the objective and its gradients at a cost that does not depend on the mesh resolution. Developing such techniques for field design and processing problems poses interesting challenges that are beyond the scope of this chapter. Another potential use of the proposed fields would be to build subspaces for a multilevel solver for problems involving tangential fields. Due to the scalability of our approach, it could be used to build subspaces of different resolution in which systems are solved and solutions are propagated. Finally, our approach could be extended to include more sophisticated boundary conditions for the eigenvalue problems we use for field construction. For example, boundary conditions that fit to the Hodge decomposition of vector fields have been proposed in [54].

BIBLIOGRAPHY

[1] Steven S. An, Theodore Kim, and Doug L. James. "Optimizing cubature for efficient integration of subspace deformations". In: *ACM Trans. Graph.* 27.5 (2008), 165:1–165:10.

[2] Omri Azencot et al. "An Operator Approach to Tangent Vector Field Processing". In: *Computer Graphics Forum* 32.5 (2013), pp. 73–82.

[3] Omri Azencot et al. "Discrete Derivatives of Vector Fields on Surfaces – An Operator Approach". In: *ACM Trans. Graph.* 34.3 (2015), 29:1–29:13.

[4] Omri Azencot et al. "Functional Fluids on Surfaces". In: *Comp. Graph. Forum*. Vol. 33. 5. Wiley Online Library. 2014.

[5] Omri Azencot et al. "Functional Thin Films on Surfaces". In: *Symposium on Computer Animation*. 2015, pp. 137–146.

[6] Jernej Barbič and Doug L. James. "Real-Time subspace integration for St. Venant-Kirchhoff deformable models". In: *ACM Trans. Graph.* 24.3 (2005), pp. 982–990.

[7] Jernej Barbič, Marco da Silva, and Jovan Popović. "Deformable Object Animation Using Reduced Optimal Control". In: *ACM Trans. Graph.* 28.3 (2009), 53:1–53:9.

[8] David Bommes, Henrik Zimmer, and Leif Kobbelt. "Mixed-integer Quadrangulation". In: *ACM Trans. Graph.* 28.3 (2009), 77:1–77:10.

[9] Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. "Hyper-reduced projective dynamics". In: *ACM Trans. Graph.* 37.4 (2018), 80:1–80:13.

[10] Christopher Brandt, Christoph von Tycowicz, and Klaus Hildebrandt. "Geometric Flows of Curves in Shape Space for Processing Motion of Deformable Objects". In: *Comp. Graph. Forum* 35.2 (2016).

[11] Christopher Brandt et al. "Modeling n-Symmetry Vector Fields using Higher-Order Energies". In: *ACM Trans. on Graph.* 37.2 (2018), 18:1–18:18.

[12] Christopher Brandt et al. "Spectral Processing of Tangential Vector Fields". In: *Computer Graphics Forum* 36.6 (2017), pp. 338–353.

[13] Christopher Brandt et al. "The Reduced Immersed Method for Real-Time Fluid-Elastic Solid Interaction and Contact Simulation". In: *ACM Trans. Graph.* 38.6 (2019).

[14] Menglei Chai, Changxi Zheng, and Kun Zhou. "A reduced model for interactive hairs". In: *ACM Trans. Graph.* 33.4 (2014), 124:1–124:11.

[15] Yanqing Chen et al. *Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate*. New York, NY, USA, Oct. 2008. DOI: `10.1145/1391989.1391995`. URL: `http://doi.acm.org/10.1145/1391989.1391995`.

[16] Ming-Te Chi et al. “Optical illusion shape texturing using repeated asymmetric patterns”. In: *The Visual Computer* 30.6-8 (2014).

[17] Keenan Crane, Mathieu Desbrun, and Peter Schröder. “Trivial Connections on Discrete Surfaces”. In: *Comp. Graph. Forum* 29.5 (2010), pp. 1525–1533.

[18] Qiaodong Cui, Pradeep Sen, and Theodore Kim. “Scalable Laplacian eigenfluids”. In: *ACM Trans. Graph.* 37.4 (2018), 87:1–87:12.

[19] Bram Custers and Amir Vaxman. “Subdivision Directional Fields”. In: *ACM Trans. Graph.* 39.2 (2020).

[20] M. Desbrun et al. “Discrete Exterior Calculus”. preprint, arXiv:math.DG/0508341. 2005.

[21] “Design of tangent vector fields”. In: *ACM Trans. Graph.* Vol. 26. 3. ACM. 2007, p. 56.

[22] Olga Diamanti et al. “Designing N-PolyVector Fields with Complex Polynomials”. In: *Comp. Graph. Forum* 33.5 (2014), pp. 1–11.

[23] Olga Diamanti et al. “Integrable PolyVector Fields”. In: *ACM Trans. Graph.* 34.4 (2015), 38:1–38:12.

[24] Fernando de Goes, Mathieu Desbrun, and Yiying Tong. “Vector Field Processing on Triangle Meshes”. In: *SIGGRAPH Asia 2015 Courses.* 2015, 17:1–17:48.

[25] Fernando de Goes et al. “Discrete 2-Tensor Fields on Triangulations”. In: *Comput. Graph. Forum* 33.5 (2014), pp. 13–24.

[26] Fernando de Goes et al. “Subdivision exterior calculus for geometry processing”. In: *ACM Trans. Graph.* 35.4 (2016), 133:1–133:11.

[27] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3.* http://eigen.tuxfamily.org. 2010.

[28] Fabian Hahn et al. “Subspace clothing simulation using adaptive bases”. In: *ACM Trans. Graph.* 33.4 (2014), 105:1–105:9.

[29] Eden Fedida Hefetz, Edward Chien, and Ofir Weber. “A Subspace Method for Fast Locally Injective Harmonic Mapping”. In: *Comput. Graph. Forum* 38.2 (2019), pp. 105–119.

[30] Aaron Hertzmann and Denis Zorin. “Illustrating smooth surfaces”. In: *Proc. SIGGRAPH.* 2000.

[31] Klaus Hildebrandt et al. “Interactive spacetime control of deformable objects”. In: *ACM Trans. Graph.* 31.4 (2012), 71:1–71:8.

[32] Klaus Hildebrandt et al. “Interactive surface modeling using modal analysis”. In: *ACM Trans. Graph.* 30.5 (2011), 119:1–119:11.

[33] Jin Huang et al. “Subspace gradient domain mesh deformation”. In: *ACM Trans. Graph.* 25.3 (2006), pp. 1126–1134.

[34] Zhiyang Huang and Tao Ju. “Extrinsically smooth direction fields”. In: *Computers & Graphics* 58 (2016), pp. 109–117.

[35] Emmanuel Iarussi, David Bommes, and Adrien Bousseau. “Bendfields: Regularized Curvature Fields from Rough Concept Sketches”. In: *ACM Trans. Graph.* 34.3 (2015).

[36] Alec Jacobson, Daniele Panozzo, et al. *libigl: A simple C++ geometry processing library*. http://libigl.github.io/libigl/. 2016.

[37] Alec Jacobson et al. "Fast automatic skinning transformations". In: *ACM Trans. Graph.* 31.4 (2012), 77:1–77:10.

[38] Wenzel Jakob et al. "Instant Field-Aligned Meshes". In: *ACM Trans. Graph.* 34.4 (2015), 189:1–189:15.

[39] Felix Kälberer, Matthias Nieser, and Konrad Polthier. "QuadCover - Surface Parameterization using Branched Coverings". In: *Comp. Graph. Forum* 26.3 (2007).

[40] Felix Knöppel et al. "Globally optimal direction fields". In: *ACM Trans. Graph.* 32.4 (2013), 59:1–59:10.

[41] Felix Knöppel et al. "Stripe Patterns on Surfaces". In: *ACM Trans. Graph.* 34 (4 2015).

[42] Yu-Kun Lai et al. "Metric-Driven RoSy Field Design and Remeshing". In: *IEEE Trans. Vis. Comput. Graph.* 16.1 (2010).

[43] Er Li et al. "Meshless quadrangulation by global parameterization." In: *Computers & Graphics* (2011).

[44] Bei-Bei Liu et al. "Model-reduced variational fluid simulation". In: *ACM Trans. Graph.* 34.6 (2015), 244:1–244:12.

[45] Beibei Liu et al. "Discrete Connection and Covariant Derivative for Vector Field Analysis and Design". In: *ACM Trans. Graph.* 35.3 (2016), 23:1–23:17.

[46] Wenlong Lu, Ning Jin, and Ronald Fedkiw. "Two-way coupling of fluids to reduced deformable bodies". In: *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2016, pp. 67–76.

[47] Soham Uday Mehta et al. "Analytic Tangent Irradiance Environment Maps for Anisotropic Surfaces". In: *Comp. Graph. Forum* 31.4 (2012).

[48] Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. "Fast Approximation of Laplace–Beltrami Eigenproblems". In: *Comp. Graph. Forum* 37.5 (2018).

[49] Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. "Locally supported tangential vector, n-vector, and tensor fields". In: *Computer Graphics Forum*. Vol. 39. 2. Wiley Online Library. 2020, pp. 203–217.

[50] Jonathan Palacios and Eugene Zhang. "Rotational symmetry field design on surfaces". In: *ACM Trans. Graph.* Vol. 26. 3. ACM. 2007, p. 55.

[51] Hao Pan et al. "Flow Aligned Surfacing of Curve Networks". In: *ACM Trans. Graph.* 34.4 (2015).

[52] Zherong Pan and Dinesh Manocha. "Active Animations of Reduced Deformable Models with Environment Interactions". In: *ACM Trans. Graph.* 37.3 (2018), 36:1–36:17. URL: `https://dl.acm.org/citation.cfm?id=3197565`.

[53] Intel MKL PARDISO. *Parallel direct sparse solver interface*. 2013.

[54] Konstantin Poelke and Konrad Polthier. "Boundary-aware Hodge decompositions for piecewise constant vector fields". In: *Computer-Aided Design* 78 (2016), pp. 126–136.

[55] Konrad Polthier and Eike Preuß. "Variational Approach to Vector Field Decomposition". In: *Symposium on Data Visualization*. Springer, 2000, pp. 147–155.

[56] Emil Praun, Adam Finkelstein, and Hugues Hoppe. "Lapped textures". In: *Proc. SIGGRAPH*. 2000, pp. 465–470.

[57] Yixuan Qiu. *SpectrA: C++ Library For Large Scale Eigenvalue Problems*. https://spectralib.org/. 2015.

[58] Philipp von Radziewsky et al. "Optimized Subspaces for Deformation-Based Shape Editing and Interpolation". In: *Computers & Graphics* 58 (2016), pp. 128–138.

[59] Nicolas Ray et al. "Geometry-aware Direction Field Processing". In: *ACM Trans. Graph.* 29.1 (2009).

[60] Nicolas Ray et al. "N-symmetry Direction Field Design". In: *ACM Trans. Graph.* 27.2 (2008).

[61] Nicolas Ray et al. "Periodic Global Parameterization". In: *ACM Trans. Graph.* 25.4 (2006), pp. 1460–1485.

[62] Boris Raymond et al. "Optimizing BRDF Orientations for the Manipulation of Anisotropic Highlights". In: *Comput. Graph. Forum* 33.2 (2014), pp. 313–321.

[63] Andrew Sageman-Furnas et al. "Chebyshev Nets from Commuting PolyVector Fields". In: *ACM Trans. Graph.* 38.6 (2019).

[64] Justin Solomon et al. "Discovery of intrinsic primitives on triangle meshes". In: *Comp. Graph. Forum* 30.2 (2011), pp. 365–374.

[65] Marco Tarini et al. "Simple Quad Domains for Field Aligned Mesh Parametrization". In: *Proc. SIGGRAPH Asia 2011* 30.6 (2011).

[66] Adrien Treuille, Andrew Lewis, and Zoran Popovic. "Model reduction for real-time fluids". In: *ACM Trans. Graph.* 25.3 (2006), pp. 826–834.

[67] Greg Turk. "Texture synthesis on surfaces". In: *Proc. SIGGRPAH*. 2001, pp. 347–354.

[68] Christoph von Tycowicz et al. "An Efficient Construction of Reduced Deformable Objects". In: *ACM Trans. Graph.* 32.6 (2013), 213:1–213:10.

[69] Christoph von Tycowicz et al. "Real-time Nonlinear Shape Interpolation". In: *ACM Trans. Graph.* 34.3 (2015), 34:1–34:10.

[70] Amir Vaxman et al. "Directional Field Synthesis, Design, and Processing". In: *Comp. Graph. Forum* 35.2 (2016), pp. 545–572.

[71] Ke Wang et al. "Edge subdivision schemes and the construction of smooth vector fields". In: *ACM Trans. Graph.* 25.3 (2006), pp. 1041–1048.

[72] Yu Wang et al. "Linear Subspace Design for Real-time Shape Deformation". In: *ACM Trans. Graph.* 34.4 (2015), 57:1–57:11.

[73] Max Wardetzky. "Discrete Differential Operators on Polyhedral Surfaces–Convergence and Approximation". PhD thesis. Freie Universität Berlin, 2006.

[74] Li-Yi Wei and Marc Levoy. “Texture Synthesis over Arbitrary Manifold Surfaces”. In: *SIGGRAPH*. 2001.

[75] Yin Yang et al. “Expediting Precomputation for Reduced Deformable Simulation”. In: *ACM Trans. Graph.* 34.6 (2015), 243:1–243:13.

[76] Chih-Yuan Yao et al. “Region-Based Line Field Design Using Harmonic Functions”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2012).

[77] Eugene Zhang, James Hays, and Greg Turk. “Interactive Tensor Field Design and Visualization on Surfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.1 (2007), pp. 94–107.

[78] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. “Vector field design on surfaces”. In: *ACM Trans. Graph.* 25.4 (2006), pp. 1294–1326.

[79] Wenjing Zhang, Jianmin Zheng, and Nadia Magnenat-Thalmann. “Real-Time Subspace Integration for Example-Based Elastic Material”. In: *Comput. Graph. Forum* 34.2 (2015), pp. 395–404.

[80] Yixin Zhuang et al. “Anisotropic geodesics for live-wire mesh segmentation”. In: *Comput. Graph. Forum* 33.7 (2014).

APPENDIX

3.A. CONSTRUCTION OF THE TENSOR FIELD LAPLACIAN

In this section, we provide details on the construction of the tensor field Laplace operator that is introduced in Section 3.3. We describe the construction at the example of second-order symmetric (1,1)-tensors. For other types of tensors, the Laplacians are constructed analogously. The tensor fields we consider are constant in every triangle of the surface mesh.

Transport of tensor We fix the coordinate system in every triangle by taking the first oriented normalized edge vector as the x-axis and the 90-degree rotated edge vector as the y-axis. Let A be the matrix representing a tensor in a triangle T_i. We want to transport the tensor to triangle T_j. In the case that the x-axes of the local coordinate systems in both triangles T_i and T_j are aligned with common edge e_{ij}, the transport is simply the identity. In general, this is not the case. Then, to transport A from T_i to T_j, we first transform the tensor in T_i to the e_{ij}-aligned coordinate system in T_i. Let R_i be the rotation matrix that maps the coordinates of vectors in the coordinate system in T_i to the coordinates of the vectors in the e_{ij}-aligned coordinate system. Then the transformation of A is $R_i A R_i^T$. The transport of $R_i A R_i^T$ to the e_{ij}-aligned coordinate system in T_j is the identity. Finally, we transform to the non e_{ij}-aligned coordinate system in T_j. Denoting the rotation matrix that transforms the coordinates in T_j to the e_{ij}-aligned coordinate system by R_j, the transported tensor in the coordinate system of T_j is $R_j^T R_i A R_i^T R_j$.

Mandel–Voigt notation When working with a linear operators on tensor fields, it is convenient to represent the tensors as vectors instead of matrices. Any matrix representing a symmetric $(1,1)$-tensor is a linear combination of the three matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} 0 & 1/\sqrt{2} \\ 1/\sqrt{2} & 0 \end{pmatrix}, \tag{3.14}$$

which are orthonormal with respect to the Frobenius norm. The vector a that stacks the coefficients of a symmetric matrix A with respect to the three matrices (3.14),

$$A = \begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix} \mapsto a = \begin{pmatrix} a_1 \\ a_3 \\ \sqrt{2}a_2 \end{pmatrix}, \tag{3.15}$$

is called the Mandel–Voigt representation of the matrix A.

Transport in Mandel–Voigt notation To write the tensor field Laplacian in Mandel–Voigt representation, we need to describe the transport of tensors in this representation. Let $G = R_i^T R_j$, then the transport of A is given by $G^T AG$. We want to find the matrix $P \in \mathbb{R}^{3\times 3}$ such that for any tensor A with Mandel–Voigt representation a, Pa is the Mandel–Voigt representation of $G^T AG$.

Let $G = \begin{pmatrix} g_1 & g_2 \\ g_3 & g_4 \end{pmatrix}$, $A = \begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix}$, $B = G^T AG$, and b the Mandel–Voigt representation of B. Then,

$$\begin{aligned} B &= G^T AG \\ &= \begin{pmatrix} g_1 & g_3 \\ g_2 & g_4 \end{pmatrix} \begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix} \begin{pmatrix} g_1 & g_2 \\ g_3 & g_4 \end{pmatrix} \qquad (3.16) \\ &= \begin{pmatrix} {g_1}^2 a_1 + 2 g_1 g_3 a_2 + {g_3}^2 a_3 & g_1 g_2 a_1 + (g_2 g_3 + g_1 g_4) a_2 + g_3 g_4 a_3 \\ g_1 g_2 a_1 + (g_2 g_3 + g_1 g_4) a_2 + g_3 g_4 a_3 & {g_2}^2 a_1 + 2 g_2 g_4 a_2 + {g_4}^2 a_3 \end{pmatrix}. \end{aligned}$$

In Mandel–Voigt notation

$$\begin{aligned} b &= \begin{pmatrix} {g_1}^2 a_1 + 2 g_1 g_3 a_2 + {g_3}^2 a_3 \\ {g_2}^2 a_1 + 2 g_2 g_4 a_2 + {g_4}^2 a_3 \\ \sqrt{2}(g_1 g_2 a_1 + (g_2 g_3 + g_1 g_4) a_2 + g_3 g_4 a_3) \end{pmatrix} && (3.17) \\ &= \begin{pmatrix} {g_1}^2 & {g_3}^2 & \sqrt{2} g_1 g_3 \\ {g_2}^2 & {g_4}^2 & \sqrt{2} g_2 g_4 \\ \sqrt{2} g_1 g_2 & \sqrt{2} g_3 g_4 & g_1 g_4 + g_2 g_3 \end{pmatrix} \begin{pmatrix} a_1 \\ a_3 \\ \sqrt{2} a_2 \end{pmatrix} && (3.18) \\ &= Pa. && (3.19) \end{aligned}$$

Laplacian for tensor fields A benefit of using the Mandel–Voigt representation is that the transport of tensors is realized by matrix multiplication. This gives the tensor field Laplacian (3.2) a structure that is similar to the usual structure of discrete Laplace operators. Let $P_{ij} \in \mathbb{R}^{3\times 3}$ denote the matrix realizing the transport of tensors from triangle T_i to T_j. Then the tensor field Laplacian in Mandel–Voigt notation is

$$(\Delta a)_i = \frac{1}{m_i} \sum_{j \in N_i} w_{ij} (a_i - P_{ji} a_j). \tag{3.20}$$

A matrix representation of the Laplace operator can be obtained by collecting the w_{ij} and P_{ij} in a stiffness matrix S and the m_i in a diagonal mass matrix M.

4

THE HIERARCHICAL SUBSPACE ITERATION METHOD FOR LAPLACE–BELTRAMI EIGENPROBLEMS

> *Geometry enlightens the intellect and sets one's mind right. All of its proofs are very clear and orderly. It is hardly possible for errors to enter into geometrical reasoning, because it is well arranged and orderly. Thus, the mind that constantly applies itself to geometry is not likely to fall into error. In this convenient way, the person who knows geometry acquires intelligence.*
>
> Ibn Khaldun

Sparse eigenproblems are important for various applications in computer graphics. The spectrum and eigenfunctions of the Laplace–Beltrami operator, for example, are fundamental for methods in shape analysis and mesh processing. In this work, we introduce the Hierarchical Subspace Iteration Method (HSIM), a novel solver of sparse eigenproblems that operates on a hierarchy of nested vector spaces. The hierarchy is constructed such that on the coarsest space all eigenpairs can be computed with a dense eigensolver. HSIM uses these eigenpairs as initialization and iterates from coarse to fine over the hierarchy. On each level, subspace iterations, initialized with the solution from the previous level, are used to approximate the eigenpairs. This approach substantially reduces the number of iterations needed on the finest grid compared to the non-hierarchical Subspace Iteration Method. Our experiments show that HSIM outperforms state-of-the-art methods based on Lanczos and subspace iterations.

This chapter is based on the paper *The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems* in ACM Transactions on Graphics (to appear) [52].

4.1. INTRODUCTION

Large-scale sparse eigenvalue problems occur in many applications of computer graphics. An important example is the computation of the low and medium frequency spectrum and the corresponding eigenfunctions of the Laplace–Beltrami operator of a surface. These are used in a range of applications in shape analysis and mesh processing. Commonly used methods for solving Laplace–Beltrami eigenproblems are based on Lanczos iterations. These are highly efficient solvers for sparse eigenvalue problems. However, in order to be efficient, they combine various extensions of the basic Lanczos iterations, which makes the algorithms complex and introduces parameters that need to be set. One problem is that Lanczos iterations are inherently unstable, which can be counteracted by re-starting strategies. Another issue is that Lanczos iterations lead to orthogonal eigenvectors only if the arithmetic is exact. Due to rounding errors, re-orthogonalization strategies are required. An alternative to Lanczos schemes is the Subspace Iteration Method (SIM). This method does not suffer from instabilities and is therefore easier to analyze and implement. On the other hand, the SIM is often slower than Lanczos schemes.

In this chapter, we introduce the Hierarchical Subspace Iteration Method (HSIM). This method is suitable for computing the eigenpairs in the low and mid frequency part of the spectrum of an operator defined on a mesh, such as the discrete Laplace–Beltrami operator. Our goal is to maintain the benefits of the SIM while reducing the computational cost significantly. One reason why the SIM is expensive is that many iterations are needed before the method converges. Our idea is to take advantage of the fact that low and mid frequency eigenfunctions can be approximated on coarser grids. Instead of working only on the finest grid, we shift iterations to coarser grids. This enables us to perform effective subspace iterations with little computational effort on coarse grids and substantially reduce the number of iterations needed on the finest grid.

We design the hierarchical solver so that it starts on the coarsest grid. The complexity of this grid is chosen such that the relevant matrices can be represented as dense matrices and all eigenfunctions on the coarsest grid can be efficiently computed with a standard dense eigensolver. Then the hierarchy is traversed from coarse to fine, whereby the eigenproblem on each grid is solved to the desired accuracy by subspace iterations and the solution on the previous grid is used as an initialization for the subspace iterations. To make the subspace iterations more efficient, we use the eigenvalues computed on one grid to determine a value by which we shift the matrix on the next grid. To construct the hierarchy, we use vertex sampling to create a vertex hierarchy and build prolongation operators based on the geodesic vicinity of the samples. The prolongation operators are used to define a hierarchy of nested function spaces on the mesh whose degrees of freedom are associated with the vertex hierarchy. The advantage of the resulting hierarchy over alternatives, such as mesh coarsening-based hierarchies, is that we obtain a hierarchy of nested spaces. This is important for our purposes because the prolongation to the finer grids then preserves properties of a subspace basis like its orthonormality.

We evaluate our HSIM scheme on the computation of the lowest p eigenpairs of the Laplace–Beltrami operator, where p ranges from 50 to 5000. Our experiments show that the HSIM significantly reduces the number of iterations needed on the finest grid and thus accelerates the SIM method. HSIM has also outperformed three state-of-the-art Lanczos solvers and the Locally Optimal Block Preconditioned Conjugate Gradient Method in our experiments. HSIM was consistently faster than the fastest of the three Lanczos solvers over a range of computations on a variety of meshes and different numbers of eigenpairs to be computed. In particular, for challenging settings, in which more than a thousand eigenpairs needed to be computed, HSIM was up to six times faster than the fastest Lanczos solver.

We expect that applications that need to compute low and medium frequency eigenfunctions of the Laplace-Beltrami operator will benefit from the properties of HSIM, in particular methods that need to continuously solve new eigenproblems, for example in the context of isospectralization [19, 56] and geometric deep learning [14], and methods that need to compute a larger number of eigenfunctions, for example, for shape compression [36, 69], filtering [68], and shape signatures[67][1].

4.2. RELATED WORK

Spectral shape analysis and processing The eigenfunctions of the Laplace–Beltrami operator on a surface have many properties that make them useful for applications. First, the eigenfunctions form an orthonormal basis for functions on the surface, which generalizes the Fourier basis of planar domains to curved surfaces. With the help of the spectrum and the eigenfunctions, a frequency representation can be associated to functions on a surface and spectral methods from signal and image processing can be generalized to methods for the processing of surface. Examples of mesh processing applications that use the Laplace–Beltrami spectrum and eigenfunctions are surface filtering [68], mesh and animation compression [36, 69], quad meshing [22, 33, 44], surface segmentation [63, 34], vector field processing [4, 12], mesh saliency [65] and shape optimization [49]. Further properties of the Laplace–Beltrami eigenfunctions are that they are invariant under isometric surface deformation and that they reflect the symmetries of a surface. These properties make them a powerful tool for non-rigid shape analysis. For example, they are used to efficiently compute shape descriptors, such as the Diffusion Distance [50], the Shape-DNA [58, 57], the Global Point Signature [60], the Heat Kernel Signature [67], the Auto Diffusion Function [25] and the Wave Kernel Signature [3]. Moreover the eigenfunctions are the basis for Functional Maps [54, 61, 39, 53, 59, 45], isospectralization [19, 56] and spectral methods in Geometric Deep Learning [15, 9, 14, 64].

Krylov schemes Krylov methods, such as Lanczos schemes for symmetric and Arnoldi scheme for general matrices, are effective solvers for large scale eigenproblems. For a comprehensive introduction to Krylov schemes, we refer to the textbook by Saad [62]. One way to apply Lanczos schemes to generalized eigenproblems, such as the Laplace–Beltrami problem we consider, is to convert them to ordinary eigenproblems by a change of coordinates. In particular, if the scalar product is given by a diagonal mass matrix, the change of coordinates is not costly [68]. For non-diagonal matrices, the coordinate

---

[1] In the supplementary material, we demonstrate that projections into subspaces spanned by Laplace–Beltrami eigenfunctions, and, at the example of the heat kernel signature, that shape signatures can benefit from using a larger number of eigenfunctions.

transformation can be done using a Cholesky decomposition of the mass matrix [62]. ARPACK [41] provides implementations of the Implicitly Restarted Lanczos Method for symmetric eigenproblems and the Implicitly Restarted Arnoldi Method for non-symmetric eigenproblems. ARPACK is so widely used that it can be seen as the standard Lanczos and Arnoldi implementations. For example, MATLAB's sparse eigensolver `eigs` interfaces ARPACK. SPECTRA [55] is a library offering a C++ implementation of an Implicitly Restarted Lanczos Method build on top of the EIGEN matrix library [27]. An alternative to the implicitly restarted Lanczos method is the band-by-band, shift-and-invert Lanczos solver for Laplace–Beltrami eigenproblems on surfaces that was introduced in [68].

Subspace iterations An alternative to Krylov schemes is the Subspace Iteration Method (SIM). A comprehensive introduction to the SIM can be found in the textbook by Bathe [6]. Matrix shifting is important to make the subspace iterations effective. Different heuristics have been proposed ranging from conservative choices [8, 26] to more aggressive shifting strategies [79]. The SIM is well-suited for parallel computing as discussed in [7].

Preconditioned Eigensolvers The lowest eigenpairs of a matrix can be computed by minimizing the Rayleigh coefficient. The Locally Optimal Block Preconditioned Conjugate Gradient Method (LOBPCG) [37] uses a preconditioned conjugate gradient solver for this minimization. A property of the method is that it does not need to explicitly access the matrix but only needs to evaluate matrix-vector products, which can be of benefit when dealing with large matrices. In recent work [23], an improved basis selection strategy is proposed that improves the robustness of the method when larger numbers of eigenpairs are computed. LOBPCG was used for solving Steklov eigenproblems in [71]. A method that uses hierarchical preconditioning to approximate a few of the lowest eigenpairs was presented in [40]. While LOBPCG is reported to be efficient for different eigenproblems, our experiments, see Section 4.6, indicate that for the Laplace–Beltrami eigenproblems we consider, eigensolvers that use sparse direct solvers are more efficient.

Approximation schemes Schemes for the approximate solution of eigenproblems are static condensation [6] in engineering and the Nyström method [73] and random projections [29] in machine learning. Approximation schemes for the Laplace–Beltrami eigenproblem on surfaces have been introduced in [18, 51, 46, 42]. In contrast to the eigensolvers we consider in this work, these schemes do not provide any guarantee on the approximation quality of the eigenpairs.

Multigrids on surfaces The multigrid hierarchy we need is challenging since we are working with an irregular grid on a curved surface. One way to build a multigrid hierarchy for a surface mesh is to use mesh coarsening algorithms [31, 1]. This is, however, not ideal for our setting because the resulting spaces are not nested, as each space is defined on different surface. Another possibility is to build hierarchical grids on ambient space and then restrict the functions to the surface [18]. The function spaces generated by this approach, however, do not resemble the linear Lagrange finite elements on the mesh that we want to work with. Algebraic multigrids [66] are an alternative that would fit our setting. However, unlike the proposed hierarchy, algebraic multigrids only use the operator to build the hierarchy, while we also use the geometry of the surface. A multi-level approach for the computation of the heat kernels on surfaces was introduced in [70].

Multilevel eigensolvers A traditional multigrid approach to eigenproblems is to treat them as a nonlinear equation and to apply nonlinear multigrid solver to the equation [28, 11]. These methods have the advantage that they can be extended or even applied directly to nonlinear eigenproblems. For linear eigenvalue problems, however, this technique is not always efficient because the specific properties of eigenvalue problems are not used when a general nonlinear solver is used.

Another approach is to integrate a multigrid scheme for solving linear systems into an eigensolver [48, 5, 47, 2]. A solver for linear eigenproblems that needs to solve linear systems in every iteration, such as Krylov and subspace iteration methods, is used as an outer iteration. In every outer iteration, the linear systems are solved in an inner multigrid loop. For our HSIM solver, we use sparse direct solvers for the linear systems, as these are more efficient in our setting than multigrid solvers, see [10]. In a different application context, however, it could be useful to use a multigrid linear solver.

An approach in which also the outer iterations operate on two different grids was proposed in [76]. In this method, the lowest eigenpair of an elliptic operator is approximated by first computing the eigenpair on the coarse grid and then correcting it by a boundary value problem on the fine grid. This approach was accelerated in [32] and extended to include matrix shifting in [77]. A multigrid extension of this scheme was introduced in [43, 17] and later integrated with wavelet bases [75] and algebraic multigrid procedures [78]. The multigrid scheme has been used for the computation of Laplace spectra on planar domains [32] and parametrized surfaces [13]. A key difference to the HSIM is that the HSIM provides users explicit control of the residual of the resulting eigenpairs. In contrast, the multigrid approaches do not provide control over the residual. Instead, the resulting residual depends on the approximation quality of the grids in the hierarchy. We include a discussion and comparison in Section 4.6.

4.3. BACKGROUND

In this section, we first briefly review the Laplace–Beltrami eigenproblem, which we use for evaluating the proposed eigensolver. Then we describe the Subspace Iteration Method, which will be the basis of the novel Hierarchical Subspace Iteration Method.

4.3.1. LAPLACE–BELTRAMI EIGENPROBLEM

In the continuous case, we consider a compact and smooth surface Σ in $\mathbb{R}^3$. A function ϕ is an eigenfunction of the Laplace–Beltrami operator Δ on Σ with eigenvalue $\lambda \in \mathbb{R}$ if

$$-\Delta\phi = \lambda\phi \tag{4.1}$$

holds. For discretization, the weak form of (4.1) is helpful. This can be obtained by multiplying both sides of the equation with a continuously differentiable function f and integrating

$$\int_\Sigma \operatorname{grad}\phi \cdot \operatorname{grad} f \, \mathrm{d}A = \lambda \int_\Sigma \phi f \, \mathrm{d}A. \tag{4.2}$$

On the left-hand side of the equation, we applied integration by parts. A function ϕ is a solution of (4.1) with eigenvalue λ if and only if (4.2) holds for all continuously differentiable functions f. A benefit of the weak form is that evaluating both integrals in (4.2) only requires functions to be weakly differentiable (with square-integrable weak derivative) and does not involve differentials of the surface's metric tensor.

In the discrete case, Σ is a triangle mesh and we consider a finite-dimensional space of functions defined on the mesh, usually the space F of continuous functions that are linear polynomials over every triangle. Then, for functions $\phi, f \in F$, the integrals in (4.2) can be evaluated and ϕ is an eigenfunction of the discrete Laplace–Beltrami operator if there is a $\lambda \in \mathbb{R}$ such that (4.2) holds for any $f \in F$.

Any function in F is uniquely determined by its function values at the vertices of the mesh. The nodal representation of a function in F is a vector $\Phi \in \mathbb{R}^n$ that lists the function values at all vertices. If a nodal vector Φ is given, the corresponding function in F can be constructed by linear interpolation of the function values at the three vertices in every triangle. Let $\varphi_i \in F$ be the function that takes the value one at vertex i and vanishes at all other vertices. Then, the stiffness, or cotangent, matrix S and the mass matrix M are given by

$$S_{ij} = \int_\Sigma \operatorname{grad}\varphi_i \cdot \operatorname{grad}\varphi_j \mathrm{d}A \qquad \text{and} \qquad M_{ij} = \int_\Sigma \varphi_i \varphi_j \, \mathrm{d}A. \tag{4.3}$$

Explicit formulas for S_{ij} and M_{ij} can be found in [72, 68]. The eigenfunctions Φ and eigenvalues λ can be computed as the solution to the eigenvalue problem

$$S\Phi = \lambda M \Phi. \tag{4.4}$$

This is a sparse, generalized eigenvalue problem where M is symmetric and positive definite and S is symmetric. We refer to [30, 21] for more background on the discretization of the Laplace–Beltrami operator on surfaces.

4.3.2. SUBSPACE ITERATION METHOD (SIM)

The subspace iteration method (SIM) is an approach for computing eigenpairs of generalized eigenvalue problems such as (4.4). We outline the SIM in Algorithm 1. The input to the method are the stiffness and mass matrices $S, M \in \mathbb{R}^{n\times n}$, a matrix $\Phi \in \mathbb{R}^{n\times q}$ that specifies an initial subspace basis, the number of desired eigenpairs p, a tolerance ε and a shifting value μ. The dimension q of the subspace needs to be larger or equal to p. We will first discuss the subspace iterations without shifting, *i.e.* assuming $\mu = 0$, and then discuss choices of subspace dimension, initial subspace basis, tolerance and shifting value.

The SIM iteratively modifies the initial basis, which consists of q vectors Φ_i, such that it more and more becomes the desired eigenbasis. In each iteration, first an inverse iteration is applied to all q vectors (Algorithm 1, line 4), thereby increasing the low-frequency components in the vectors. For this, q linear systems of the form

$$(S - \mu M)\Psi_i = M\Phi_i \tag{4.5}$$

need to be solved. The second step in each iteration is to solve the eigenproblem restricted to the subspace spanned by the vectors Ψ_i (Algorithm 1, lines 5–7). For this, the reduced stiffness and mass matrices are computed and the q-dimensional dense eigenproblem is solved using a dense eigensolver, *e.g.* based on a QR factorization. The third step is to replace the current subspace basis with the eigenbasis (Algorithm 1, line 8). The inverse iterations amplify the low frequencies in the subspace basis. The second and third steps are needed in order to prevent the vectors from becoming linearly dependent. Without these steps, the vectors would all converge to the lowest eigenvector.

ALGORITHM 1: Subspace Iteration Method

```
    Input: Stiffness matrix $S \in \mathbb{R}^{n\times n}$, mass matrix $M \in \mathbb{R}^{n\times n}$, initial vectors $\Phi \in \mathbb{R}^{n\times q}$, number
           of eigenpairs $p$, tolerance $\varepsilon$, shifting value $\mu$
    Output: Matrix $\bar{\Lambda}$ with lowest eigenvalues of (4.4) on diagonal and $\Phi$ listing eigenvectors
            as columns. First $p$ pairs converged.
 1  Function SIM($S, M, \Phi, p, \varepsilon, \mu$):
 2      Compute sparse factorization: $LDL^T = S - \mu M$
 3      repeat
 4          Solve using factorization: $(S - \mu M)\Psi = M\Phi$
 5          Compute reduced stiffness matrix: $\bar{S} \leftarrow \Psi^T S \Psi$
 6          Compute reduced mass matrix: $\bar{M} \leftarrow \Psi^T M \Psi$
 7          Solve dense eigenproblem: $\bar{S}\bar{\Phi} = \bar{M}\bar{\Phi}\bar{\Lambda}$
 8          Update vectors: $\Phi \leftarrow \Psi\bar{\Phi}$
 9      until pairs $(\bar{\Lambda}_{ii}, \Phi_i)$ pass convergence test (4.6) for all $i \leq p$
10      return $\bar{\Lambda}$ and $\Phi$
11  End Function
```

The final step of each iteration is the convergence check, which tests whether or not the first p eigenvectors have converged. For each eigenpair Φ_i and λ_i, the relative norm of the residual of equation (4.4) is computed

$$\frac{\| S\Phi_i - \lambda_i M\Phi_i \|}{\| S\Phi_i \|} < \varepsilon \tag{4.6}$$

and the test is passed if it is below the threshold ε. The choice of the value for the convergence tolerance depends on the application context. In most of our experiments, we used $\varepsilon = 10^{-2}$, which based on our experiments, see Section 4.5, we consider appropriate for applications in shape analysis and spectral mesh processing.

Subspace dimension The choice of the dimension q affects the computational cost per iteration and the number of iterations needed for convergence. A larger subspace size increases the computational cost per iteration as more linear systems have to be solved (line 4 of Algorithm 1) and the dimension of the dense eigenproblem (line 7) increases. On the other hand, the algorithm terminates when the subspace contains (good enough approximations of) the lowest p eigenvectors. This is easier to achieve if the subspace is larger. Therefore, with a larger subspace, fewer iterations may be needed. An effective choice, which is derived and justified in [7], is $q = \max\{2p, p+8\}$.

Initialization The subspace basis Φ can be initialized with a random matrix. An alternative is to use information extracted from the matrices for initialization, which can help to reduce the required number of subspace iterations. One heuristic from [6] is to use the diagonal of the mass matrix M as the first column of the matrix representing initial vectors, random entries for the last column, and unit vectors e_i with entry $+1$ at the degree of freedom with the smallest ratio of k_{ii}/m_{ii} for the remaining $q-2$ columns.

Shifting One way to make the subspace iterations more effective is to shift the matrix S, which means to replace it with the shifted matrix $S-\mu M$. The shifted matrix keeps the same eigenvectors while the eigenvalues are shifted by $-\mu$. As a consequence, the inverse iteration, line 4 of algorithm 1, focuses on enhancing the frequencies around μ instead of around zero. This can help to reduce the number of iterations required for convergence. Different heuristics for setting the shifting value have been proposed. A conservative choice is to set μ to the average of the last two converged eigenvalues [8]. Alternative shifting strategies are to set μ to the average of the last converged and the first non-converged eigenvalue [74] or to the average of the first two non-converged eigenvalues [26]. An aggressive shifting technique that places μ further into the range of the non-converged eigenvalues is shown to accelerate the SIM in [79].

Direct solver For the inverse iterations of the subspace basis, line 4 of Algorithm 1, q linear systems with the same matrix $S-\mu M$ need to be solved. It can be effective to use a direct solver for this task since a factorization once computed can be used to solve all the systems. For the Laplace–Beltrami eigenproblems we consider, a sparse symmetric indefinite decomposition $LDL^T = S-\mu M$ is adequate.

4.4. HIERARCHICAL SUBSPACE ITERATION METHOD

In this section, we introduce the Hierarchical Subspace Iteration Method (HSIM). We first describe the construction of the hierarchy of function spaces on a mesh. Then we detail the multilevel eigensolver that operates on the hierarchy.

4.4.1. HIERARCHY CONSTRUCTION

Important goals for the construction of the hierarchy are that the construction is fast since the hierarchy must be built as part of the HSIM algorithm, that the basis functions are locally supported and the prolongation and restriction operators are sparse, and that the functions spaces are nested. Moreover, the function spaces need to be able to approximate low and mid-frequency functions well.

We describe the construction of the subspaces in three steps. First, we describe the construction of a hierarchy on the set of vertices of the mesh. Then, we define prolongation and restriction operators that act between the levels of the vertex hierarchy. Finally, we explain how the vertex hierarchy and the operators can be used to obtain the hierarchy of nested function spaces.

ALGORITHM 2: Construction of the vertex hierarchy

```
   Input: Surface mesh $\Sigma$, number of levels $T$, number of vertices per level $n^1, n^2, \dots, n^{T-1}$
   Output: Set of vertex indices $V^1, V^2, \dots, V^{T-1}$
 1 $V^T \leftarrow$ {Random number from $\{0, 1, \dots, |V_\Sigma| - 1\}$}
 2 $\tau \leftarrow T - 1$
 3 repeat
 4     $V^\tau \leftarrow V^{\tau+1}$
 5     repeat
 6         $V^\tau \leftarrow V^\tau \cup$ {Index of vertex farthest away from $V^\tau$}
 7     until $|V^\tau| = n^\tau$
 8     $\tau \leftarrow \tau - 1$
 9 until $\tau = 0$
10 return $V^1, V^2, \dots, V^{T-1}$
```

Vertex hierarchy We consider a hierarchy with T levels ranging from 0 to $T-1$, where 0 is the finest level. We denote by V^τ the set of vertices in level τ and by n^τ the number of vertices in V^τ. The sets V^τ are nested, $V^\tau \subset V^{\tau-1}$, and V^0 is the set of all vertices of the mesh. Since we will solve a dense eigenproblem to get all eigenpairs at the coarsest level, we want to control the number n^{T-1} of vertices in V^{T-1}, which we set to

$$n^{T-1} = \max\{\lceil 1.5p \rceil, 1000\}. \tag{4.7}$$

The numbers of vertices in the other levels are determined by the growth rate μ

$$n^\tau = \mu \, n^{\tau+1}, \tag{4.8}$$

where μ is given by

$$\mu = \sqrt[T]{\frac{n^0}{n^{T-1}}}. \tag{4.9}$$

The trade-off for the choice of the number of levels is that a larger number of levels helps to reduce the required number of iterations on the finest level. On the other hand, each level adds computational cost, *e.g.* for computing the reduced matrices S^τ and M^τ. In our experiments, we found HSIM to be most effective with a low number of levels. We used three levels in most cases and opted for two levels when only a small number of eigenpairs, *i.e.* $p \leq 200$, needs to be computed.

To form the sets V^τ, we use a scheme based on farthest point sampling [24]. The set V^{T-1} is initialized to contain one random vertex. Then, iteratively the vertex farthest away from all the vertices that are already in V^{T-1} is added to V^{T-1} until the desired number of vertices is reached. The sets V^{T-2} to V^1 are created in a similar manner. The scheme is summarized in Algorithm 2. The most expensive step is the computation of the farthest point in line 6. The farthest point can be computed efficiently by maintaining a distance field that stores for each vertex of the mesh the distance to the closest vertex in the current set V^τ. Since the vertices are inserted one after another, in each iteration the distance field only needs to be updated locally around the newly inserted

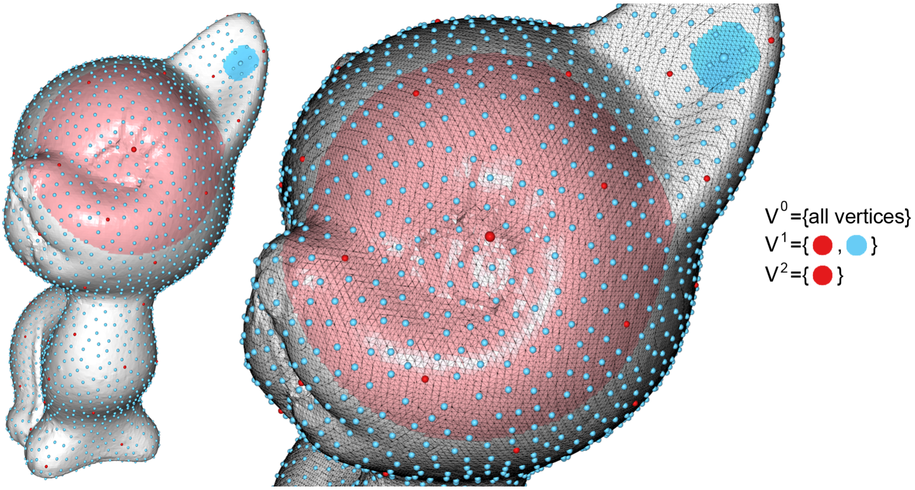


Figure 1: Illustration of a vertex hierarchy with three sets $V^2 \subset V^1 \subset V^0$. The coarsest set V^2 consists of the red vertices, V^1 of the blue and the red vertices and V^0 of all vertices of the mesh. The light red and light blue regions are geodesic disks of radii ρ^2 and ρ^1 around the highlighted red and blue vertices in the centers of the regions and illustrate the support regions of the highlighted vertices.

vertex, and the maximum of the field has to be computed. We compute the distances between vertices using Dijkstra's algorithm on the edge graph with weights corresponding to the length of the edges. We found Dijkstra's distance a sufficient approximation of the geodesic distance in our experiments. Alternatively, the Short-Term Vector Dijkstra (STVD) algorithm [16] could be used, which computes a more accurate approximation of the geodesic distance while still keeping computations localized. The supplementary material includes examples that illustrate that farthest point sampling generates hierarchies that are suitable for our purposes.

Prolongation and restriction A function on level τ is represented by a vector $f^\tau \in \mathbb{R}^{n^\tau}$. We will first describe the prolongation and restriction operators and show in the next paragraph how the prolongation operator can be used to construct the piecewise linear polynomial corresponding to a vector f^τ. The τ^{th} prolongation operator is given by a matrix $U^\tau \in \mathbb{R}^{n^\tau \times n^{\tau+1}}$ that maps vectors $f^{\tau+1} \in \mathbb{R}^{n^{\tau+1}}$ representing functions on level $\tau+1$ to vectors $f^\tau \in \mathbb{R}^{n^\tau}$ representing functions on the finer level τ. The restriction operator maps from level τ to the coarser level $\tau+1$ and is given by the transpose $U^{\tau T}$ of the prolongation matrix. This relationship of the prolongation and restriction operators ensures that the restricted matrices S^τ and M^τ, see lines 4 and 5 of Algorithm 3 for a definition of the matrices, on all levels are symmetric.

The i^{th} row of U^τ describes how the value associated with the i^{th} vertex of level $\tau+1$ is distributed among the vertices on level τ. This means the entry U^τ_{ij} is a weight of how much vertex j on level τ receives from vertex i on level $\tau+1$ during prolongation. This weight decreases with increasing geodesic distance of the vertices. To obtain sparse operators, the weight vanishes when the distance of the vertices reaches a threshold ρ^τ, which differs per level. We set ρ^τ to be

$$\rho^\tau = \sqrt{\frac{\sigma A}{d^\tau \pi}}. \tag{4.10}$$

where A is the area of the surface and σ is a control parameter. This choice of ρ^τ yields matrices U^τ that have about σ non-zero entries per row. For our experiments, we choose $\sigma = 7$. The reasoning behind (4.10) is that we want the sum of the areas of the geodesics disks of radius ρ^τ around all the vertices of level τ to be σ times the area of the surface. To make this idea easily computable, we approximate the combined areas of all the geodesic disks by n^τ times the area of the Euclidean disk of radius ρ^τ.

To construct the matrices U^τ, we first construct preliminary matrices $\tilde{U}^\tau \in \mathbb{R}^{n^\tau \times n^{\tau+1}}$ that have the entries

$$\tilde{U}^\tau_{ij} = \begin{cases} 1 - \frac{d(v_i^{\tau+1}, v_j^\tau)}{\rho^\tau} & \text{for } d(v_i^{\tau+1}, v_j^\tau) \leq \rho^\tau \\ 0 & \text{for } d(v_i^{\tau+1}, v_j^\tau) > \rho^\tau \end{cases}, \tag{4.11}$$

where $d(v_i^{\tau+1}, v_j^\tau)$ is the geodesic distance of the i^{th} vertex of $V^{\tau+1}$ to the j^{th} vertex of V^τ. The matrix U^τ is then obtained by normalizing the rows of $\tilde{U}^\tau$

$$U^\tau_{ij} = \frac{1}{\sum_{j=1}^{n^\tau} \tilde{U}^\tau_{ij}} \tilde{U}^\tau_{ij}. \tag{4.12}$$

The normalization ensures that all function spaces will include the constant functions. This is important for our purposes as the constant functions make up the kernel of the Laplace–Beltrami operator. Another property is that the set of functions on each level forms a partition of unity. As for the sampling scheme, we use Dijkstra's distance on the weighted edge graph of the mesh in our experiments to approximate the geodesic distance. A discussion of two alternatives, the Short-Term Vector Dijkstra algorithm [16] and the Heat Method [20], is included to the supplementary material.

Function spaces So far we have considered abstract vectors $f^\tau \in \mathbb{R}^{n^\tau}$. Now, we describe how the continuous piecewise linear polynomial corresponding to f^τ can be constructed. On the finest level, any $f^0 \in \mathbb{R}^{n^0}$ is the nodal vector, which lists the function values of the continuous, piecewise linear polynomial at the vertices. To get the continuous, piecewise linear polynomial that corresponds to a $f^\tau \in \mathbb{R}^{n^\tau}$ for any τ, we use the prolongation operators to lift f^τ to the finest level. The resulting vector

$$U^0 U^1 ... U^{\tau-1} f^\tau \tag{4.13}$$

is the nodal vector of the continuous, piecewise linear polynomial corresponding to f^τ. By construction, the resulting function spaces are nested and the functions are locally supported. The HSIM algorithm does not need to lift the functions using (4.13). Instead, the reduced stiffness and mass matrices S^τ and M^τ are directly computed for each level.

4.4.2. HIERARCHICAL SOLVER

The HSIM is outlined in Algorithm 3. The algorithm starts with preparing the multilevel subspace iterations. First, the number of levels, the vertex hierarchy and the prolongation matrices U^τ are computed. Then the reduced stiffness and mass matrices, S^τ and M^τ, for all levels are constructed from fine to coarse starting with level 1. In this computation, we benefit from the fact that the prolongation matrices U^τ are highly sparse. The next step is to determine the dimension q of the subspace that is used. Our experiments indicate that values between $q = 1.5p$ and $q = 2p$ are suitable. Following [7], we set $q = p + 8$ for small values of p

$$q = \max\{\lceil 1.5p \rceil, p + 8\}. \tag{4.14}$$

The last step before the multilevel iterations start is the computation of an initial subspace. This is done by solving the eigenproblem on the coarsest level of the hierarchy completely using a dense eigensolver. The dimension of the coarsest space is chosen, see (4.7), such that the dense eigenproblem can be solved efficiently. To obtain the initialization of the multilevel iterations, the first q eigenvectors are lifted to level $T-2$ of the hierarchy using the prolongation matrix U^{T-1}. This initialization of the multilevel subspace iterations can be computed quickly and results in a subspace basis that is already a rough approximation of the desired eigenbasis.

ALGORITHM 3: Hierarchical Subspace Iteration Method

```
Input: Stiffness and mass matrices of finest level $S^0, M^0 \in \mathbb{R}^{n\times n}$, number of eigenpairs $p$,
       number of levels $T$, tolerance $\varepsilon$
Output: $p$ lowest eigenpairs of the generalized eigenproblem (4.4)
Function HSIM($S^0, M^0, p, T, \varepsilon$):
    Compute vertex hierarchy (Section 4.4.1)
    Build matrices $U^\tau$ for $\tau = 0, 1, \dots, T-2$ (Section 4.4.1)
    for $\tau \leftarrow 1$ to $T-1$ do
        Build level $\tau$ stiffness matrix: $S^\tau \leftarrow (U^{\tau-1})^T S^{\tau-1} U^{\tau-1}$
        Build level $\tau$ mass matrix: $M^\tau \leftarrow (U^{\tau-1})^T M^{\tau-1} U^{\tau-1}$
    end
    Set size of subspace: $q \leftarrow \max(\lceil 1.5p \rceil, p+8)$
    Compute first $q$ eigenpairs of $S^{T-1}\Phi^{T-1} = \Lambda^{T-1} M^{T-1} \Phi^{T-1}$
    for $\tau \leftarrow (T-2)$ to $0$ do
        Prolongation of subspace basis: $\Phi^\tau \leftarrow U^\tau \Phi^{\tau+1}$
        Set shifting parameter: $\mu \leftarrow \Lambda^{\tau+1}_{jj}$ with $j = \lfloor \frac{p}{10} \rfloor$
        $(\Lambda^\tau, \Phi^\tau) \leftarrow$ SIM($S^\tau, M^\tau, \Phi^\tau, p, \varepsilon, \mu$)
    end
    return First $p$ diagonal entries of $\Lambda^0$ and first $p$ columns of $\Phi^0$
End Function
```

The multilevel iterations traverse the hierarchy from coarse to fine starting with the second coarsest level. At each level, the eigenproblem is solved up to the tolerance by subspace iterations. The subspace iterations are initialized with the eigenvectors computed at the coarser level. To make the subspace iteration more effective, we use the approximate eigenvalues computed on the previous level to specify a shifting parameter for the iterations on the current level. We employ an aggressive shifting strategy, which sets

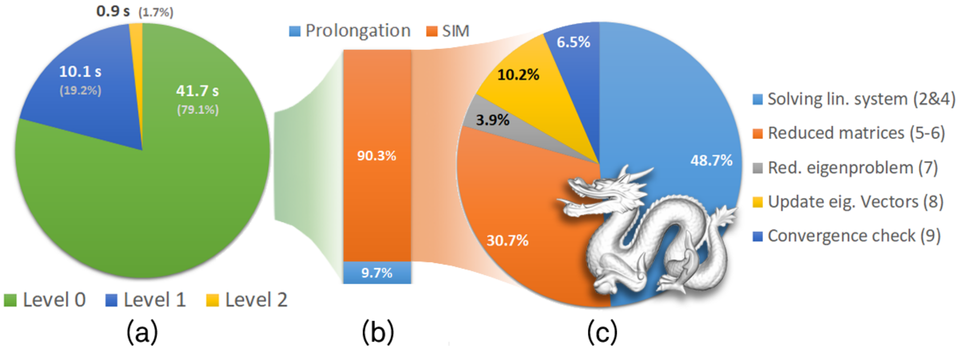


Figure 2: Analysis of timings of the HSIM for the Laplace–Beltrami eigenproblem. Distribution of the runtimes to the three levels (a), split of the time spent at the finest level between the prolongation of the solution from level 1 and the subspace iterations at the finest level (b) and distribution of the time of the subspace iteration to the individual steps in Algorithm 1. The 200 lowest eigenpairs are computed on the Dragon mesh with 150k vertices.

the shifting value to be the estimated eigenvalue with index $\lfloor p/10 \rfloor$. The shifting value is set only once for each level and used for all subspace iterations on this level. Then, only one sparse factorization of the shifted stiffness matrix $S^\tau - \mu^\tau M^\tau$ has to be computed per level. This way we achieve that, on the one hand, the shifting value is regularly updated, while, on the other hand, no additional factorizations have to be computed.

To further accelerate the subspace iterations, we do not perform additional inverse iterations, step 4 of the Algorithm 1, on the lowest r vectors that are already converged. However, to avoid error accumulation, we stop iteration of vectors only after the residual, eq. (4.6), of the first r vectors has reached one tenth of the specified tolerance ε. A further acceleration is achieved by performing two inverse iterations before orthonormalizing the vectors. Thus we execute step 4 in algorithm 1 twice before we continue with step 5.

The subspace iteration method converges quickly when the desired eigenspace is close to the initial subspace. Our hierarchical method makes use of this property by providing the subspace iterations on each level with the solution from the coarser level. As a result, only few iterations are needed on each level. In particular, the multilevel strategy substantially reduces the necessary number of iterations on the finest level compared to the SIM. The price to pay is that the hierarchy has to be built and iterations on the coarse levels are needed. Nevertheless, HSIM is 4-8 times faster than SIM in our experiments. The highest acceleration is achieved in the difficult case that a large number of eigenvectors must be computed.

4.5. EXPERIMENTS

Implementation Our implementation of HSIM uses *Eigen* [27] for linear algebra functionalities and *LibIGL* [35] for geometry processing tasks. OpenMP is used to solve the linear systems in each subspace iteration, step 4 of Algorithm 1, in parallel and to compute the prolongation and projection matrices in parallel during hierarchy construction.

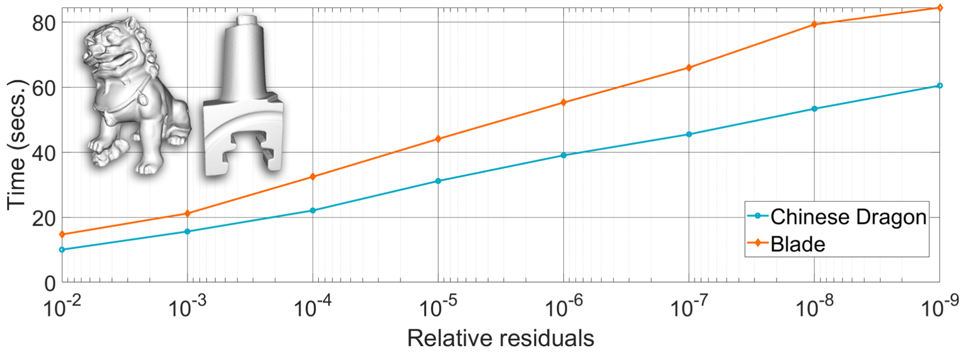


Figure 3: The plot of required computation time to achieve the desired accuracy, in computing the first 100 eigenpairs of Laplace–Beltrami operator on the Chinese Dragon (127k vertices) and on the Blade model (200k vertices).

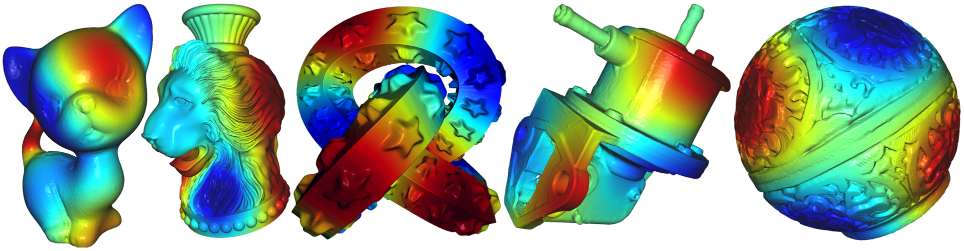


Figure 4: The 10th eigenfunction of the Laplace-Beltrami operator for the models for which the time points are listed in Table 1.

Moreover, we solve the low-dimensional eigenproblems at the coarsest level of the hierarchy, step 9 of Algorithm 3, and in each subspace iteration, step 7 of Algorithm 1, on the GPU using a direct solver for dense generalized eigenproblems from CUDA's CUSOLVER library.

Timings Table 1 lists timing for our HSIM implementation for the computation of the p lowest eigenpairs of the discrete Laplace–Beltrami operator, eq. (4.4), for meshes with different sizes and values of p. Individual timings for hierarchy construction and solving the problem using the hierarchy are listed. Moreover, iteration counts for subspace iterations on the individual levels are provided. For the coarsest level, a dense solver is used instead of the subspace iteration, therefore, the table lists F's instead of a number for the coarsest level. The convergence tolerance for the solver is set to 10^{-2} for all examples. In all cases, the required number of iteration on the finest level is reduced to one by the hierarchical approach. Figure 2 provides more details for one example, the computation of the lowest 200 eigenpairs on a dragon model with $150k$ vertices using a hierarchy with three levels. The figure shows (a) how the runtimes split over the different levels of the

<table>
<tr><th rowspan="2">Model
(#Verts)</th><th rowspan="2">#Eigs</th><th rowspan="2">#Iters</th><th colspan="3">Timings of HSIM</th></tr>
<tr><th>Hier.</th><th>Solver</th><th>Total</th></tr>
<tr><td rowspan="3">Kitten
(137k)</td><td>50</td><td>F|1</td><td>1.9</td><td>6.2</td><td>8.1</td></tr>
<tr><td>250</td><td>F|1|1</td><td>3.7</td><td>28.4</td><td>32.1</td></tr>
<tr><td>1000</td><td>F|2|1</td><td>4.3</td><td>118.9</td><td>123.2</td></tr>
<tr><td rowspan="3">Vase-Lion
(200k)</td><td>50</td><td>F|1</td><td>3.2</td><td>5.3</td><td>8.5</td></tr>
<tr><td>250</td><td>F|2|1</td><td>7.3</td><td>41.2</td><td>48.5</td></tr>
<tr><td>1000</td><td>F|3|1</td><td>9.2</td><td>188.0</td><td>197.2</td></tr>
<tr><td rowspan="3">Knot-Stars
(450k)</td><td>50</td><td>F|1</td><td>9.9</td><td>28.4</td><td>38.3</td></tr>
<tr><td>250</td><td>F|2|1</td><td>29.1</td><td>131.0</td><td>160.1</td></tr>
<tr><td>1000</td><td>F|3|1</td><td>36.3</td><td>505.7</td><td>542.0</td></tr>
<tr><td rowspan="3">Oilpump
(570k)</td><td>50</td><td>F|1</td><td>9.2</td><td>31.9</td><td>41.1</td></tr>
<tr><td>250</td><td>F|2|1</td><td>31.6</td><td>122.6</td><td>154.2</td></tr>
<tr><td>1000</td><td>F|3|1</td><td>40.3</td><td>650.6</td><td>690.9</td></tr>
<tr><td rowspan="3">Red-Circular
(700k)</td><td>50</td><td>F|1</td><td>10.2</td><td>65.5</td><td>75.7</td></tr>
<tr><td>250</td><td>F|2|1</td><td>40.6</td><td>199.3</td><td>239.9</td></tr>
<tr><td>1000</td><td>F|4|1</td><td>55.0</td><td>1061.2</td><td>1116.2</td></tr>
</table>

Table 1: Timings of HSIM for the computation of the lowest eigenpairs of the Laplace–Beltrami operator on surface meshes with different numbers of vertices. The error tolerance ε is set to 10^{-2}. Individual timings for constructing the hierarchy and for solving the eigenproblem using the hierarchy are listed (in seconds). Meshes are shown in Figure 4.

hierarchy, (b) for the finest level the division between prolongation of the solution for the second finest level and subspace iterations, and (c) the split for the individual steps of the subspace iterations (Algorithm 1) on the finest level. The figure illustrates that, when three levels are used, most of the runtime is spent on the finest level, almost 80% for the shown example, and that the restriction of the stiffness and mass matrices and solving the linear systems are the most costly steps of HSIM.

Figure 5 lists runtimes for different numbers of eigenpairs to be computed. In our experiments, we found that the runtime grows linearly even when computing several thousand eigenpairs. This is illustrated by the timings listed in the figure. We expect this linear trend to continue as long as the runtime is dominated by the time needed for the solving of the linear systems (step 4 of Algorithm 1). At some point, solving the dense eigenproblem (step 7 of Algorithm 1), which does not scale linearly with the number of eigenpairs, will be the most expensive step and the trend will no longer be linear.

For most experiments, we set the convergence tolerance, ϵ in Algorithm 3, to 10^{-2}. Figure 3 lists runtimes over the convergence tolerance for the computation of 100 eigenpairs on two different meshes, the Blade model with 200k vertices and the Chinese Dragon with 127k vertices. The figure illustrates that low tolerances such as 10^{-9} can be achieved and that the time grows proportional with the relative residual. Roughly speaking, we observe in our experiments that the number of iterations that are needed on the finest grid grows by two for a decrease of one order of magnitude in the relative residual.

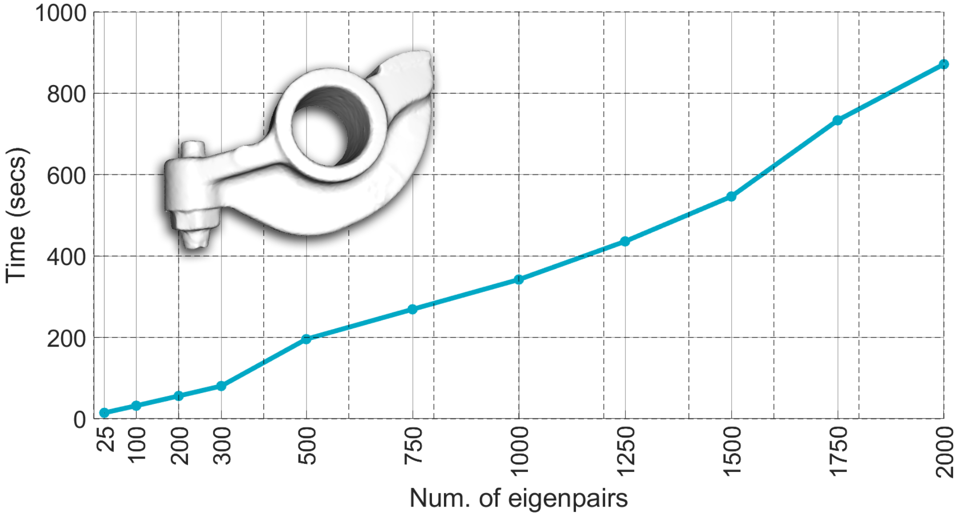
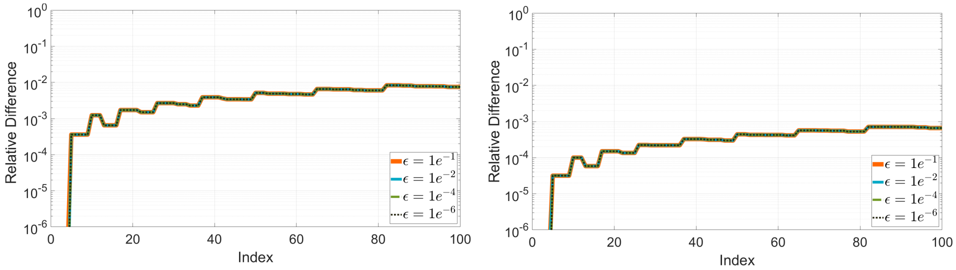


Figure 5: Plot listing the runtime of HSIM over the number of Laplace–Beltrami eigenpairs to be computed on the Rocker Arm model with 270k vertices.

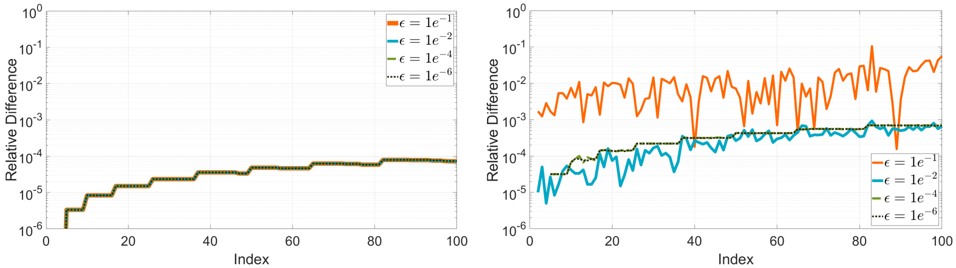
Termination criterion To test for convergence, see line 9 of Algorithm 1, we use the criterion stated in (4.6). This test concerns the convergence of the eigenvalues as well as the convergence of the eigenvectors. To determine a suitable value for the convergence tolerance ε, we performed several experiments. We discuss two experiments in this paragraph, the supplementary material includes additional experiments. Based on the results of these experiments, we used a tolerance of $\varepsilon = 10^{-2}$ for the evaluation of HSIM. In the first experiment, we consider three different discretizations of the unit sphere with regular meshes (consisting of 10k, 100k and 1m vertices) and measure the difference between the computed eigenvalues for different tolerances ($\varepsilon = 10^{-1}, 10^{-2}, 10^{-4}, 10^{-6}$) and the analytical solution. The results are shown in Figure 6. For all three discretizations, the difference between the numerical solutions for different tolerances is small compared to the approximation error, that is, the difference to the analytical solution. We would like to note that the convergence test establishes an upper bound on the convergence of the eigenpairs. In particular, for the lowest tolerance, $\varepsilon = 10^{-1}$, the solutions computed by HSIM are often already more accurate when the process terminates. One reason for this is that the method terminates only after all eigenpairs pass the convergence test. We therefore conducted an additional experiment using the inverse power method to compute the eigenpairs one by one and stop the iteration for each eigenpair when the convergence tolerance is reached. The results are shown in Figure 6 (d). In this experiment, differences in accuracy occur between the numerical solution for $\varepsilon = 10^{-1}$ and the other solutions ($\varepsilon = 10^{-2}, 10^{-4}, 10^{-6}$), which indicates that a tolerance of $\varepsilon = 10^{-1}$ is not sufficient.

In a second experiment, we compute eigenpairs for two different meshes approximating the same surface. The second mesh was created by flipping edges of the first



(a) HSIM, 10k

(b) HSIM, 100k



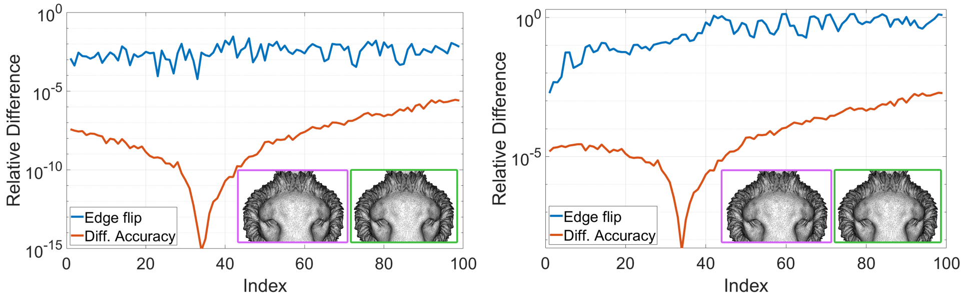
(c) HSIM, 1m

(d) IPM, 100k

Figure 6: Relative difference of numerical approximations of the eigenvalues of the unit sphere to the analytic solutions are shown.

mesh. For both meshes, we compute the lowest eigenpairs for the tolerance $\varepsilon = 10^{-2}$ and as reference for $\varepsilon = 10^{-8}$. Since the two meshes have the same vertices, we can compare both the eigenvalues and the eigenvectors. Figure 7 shows the difference between the reference solutions ($\varepsilon = 10^{-8}$) on both meshes (blue graph) and for one mesh, the difference between the solutions for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-8}$ (red graph). It can be seen that the difference between the reference solutions on the two meshes is more than three orders of magnitude larger than the difference between the solutions for different tolerances. We want to note that the convergence test (4.6) does not directly measure the deviation from the exact solution. In our experiments (for example in Figure 7), we see that the relative difference between the solution for a tolerance of $\varepsilon = 10^{-2}$ and the reference solution, which is computed with $\varepsilon = 10^{-8}$, is usually much smaller than 10^{-2}. In Figure 7, and also in Figure 11, one can observe that the errors generated by HSIM are smaller for the eigenvalue pairs whose index is about one-third of the total number of computed eigenvalues than for the others. This is due to our shifting strategy, which makes these eigenpairs converge faster.

Number of levels A parameter HSIM needs as user input is the number of levels of the hierarchy, see Algorithm 3. By increasing the number of levels, one can reduce the number of iterations required on the finest grid. On the other hand, by increasing the number of levels, the computational cost of the computations on the levels increases



(a) Eigenvalues

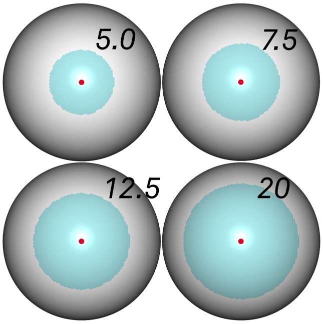
(b) Eigenvectors

Figure 7: Comparison of the relative difference of the eigenvalues and the eigenvectors between two meshes that approximate the same surface (blue graph) and solutions for different convergence tolerance on one of the meshes (red graph).

Model (#Verts)	#Eigs	Tol	Level=2		Level=3		Level=4		Level=5	
			#Iters	Time	#Iters	Time	#Iters	Time	#Iters	Time
Rocker Arm (270k)	50	1e-2	F|1	17.2	F|1|1	31.1	F|1|1|1	66.0	F|1|1|1|1	123.8
		1e-4	F|3	26.0	F|3|2	38.2	F|2|2|3	76.8	F|2|2|2|2	135.3
	2000	1e-2	F|3	1078.6	F|2|1	650.7	F|1|1|1	885.3	F|1|1|1|1	1412.0
		1e-4	F|8	2595.4	F|7|4	2137.1	F|6|4|4	2967.8	F|5|3|3|4	3586.4
Ramses (820k)	50	1e-2	F|1	45.3	F|1|1	101.0	F|1|1|1	244.0	F|1|1|1|1	495.0
	300		F|2	246.4	F|2|1	234.5	F|2|1|1	417.6	F|1|1|1|1	700.6
	750		F|2	969.6	F|2|1	657.5	F|2|1|2	1521.1	F|2|1|1|1	1607.0

Table 2: Performance of HSIM with different numbers of levels.

rapidly. Table 2 lists computation times and iteration counts for the individual levels for computations with different meshes sizes, number of eigenpairs and convergence tolerances. For most of these examples, three levels yield the shortest runtime.



Support region For the construction of the prolongation matrices U^τ the radius of its domain of influence, ρ^τ, must be defined individually for each level. We use eq. (4.10), which allows us to set the radii on all levels by means of a control parameter σ. This value is the average expected number of non-zero entries per row of the matrices U^τ. The inset figure shows the areas of influence around one point for different values σ. A smaller value for σ results in matrices U^τ with less non-zero entries and thus less computational effort per iteration. On the other hand, a too small value for σ can increase the number of iterations needed on each level.

In our experiments, we have identified a value of $\sigma = 7$ as a good trade-off. This means that in each level, each vertex of V^τ in average is coupled to six neighbor vertices, which agrees with the average valence in a triangle mesh. Table 3 shows iteration

Model (#Verts)	#Eigs	2.5		5		7		10		20	
		#Iters	Time	#Iters	Time	#Iters	Time	#Iters	Time	#Iters	Time
Vase-Lion (200k)	100	F|2	15.9	F|2	17.0	F|1	12.2	F|1	13.4	F|2	23.6
	400	F|3|2	78.3	F|3|1	60.4	F|3|1	69.6	F|2|1	74.3	F|2|1	110.4
	750	F|4|2	175.4	F|3|2	174.8	F|3|1	138.9	F|3|2	219.5	F|3|2	301.5
Eros (475k)	100	F|2	55.7	F|1	42.2	F|1	43.0	F|2	65.3	F|2	71.7
	400	F|2|3	273.6	F|2|2	231.9	F|2|1	169.8	F|4|1	207.2	F|4|1	292.7
	750	F|3|4	710.2	F|2|3	580.5	F|4|1	465.5	F|4|2	567.8	F|5|2	758.7

Table 3: Runtimes and iteration counts for different values of the parameter σ that determines the supports of the functions.

#Eigs	Residue	Shift ratio						
		No shift	0.1	0.2	0.25	1/3	0.4	0.45
50	1e-2	F|1	F|1	F|1	F|1	**F|1**	F|2	F|2
250		F|2|1	F|2|1	F|2|1	F|2|1	**F|2|1**	F|2|1	F|2|1
1000		F|3|1	F|2|1	F|2|1	F|2|1	**F|2|1**	F|3|2	F|4|2
50	1e-4	F|5	F|5	F|4	F|4	**F|4**	F|4	F|5
250		F|6|4	F|6|4	F|5|4	F|5|3	F|5|3	**F|4|3**	F|5|4
1000		F|8|4	F|8|4	F|7|4	F|7|4	**F|6|3**	F|6|4	F|7|5

Table 4: Iteration counts for different choices of shifting values are shown. Computations are done using the Gargoyle model with 85k vertices.

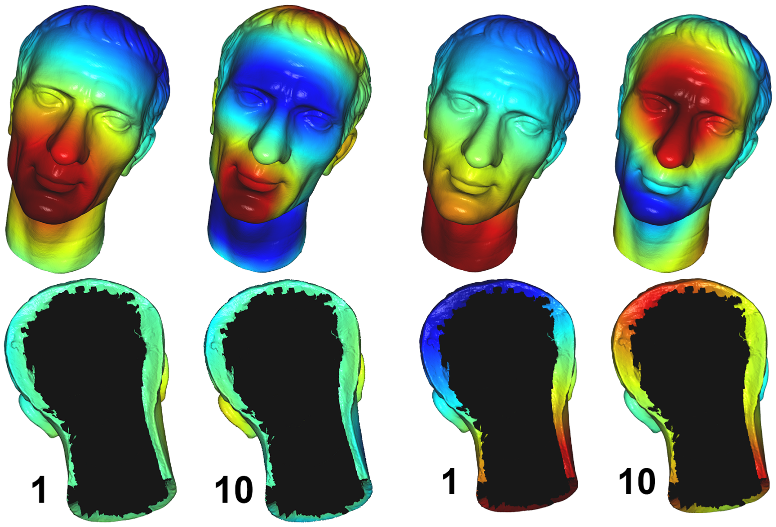
counts and runtimes for different values for σ for eigenproblems on two meshes with 200k and 475k vertices and different numbers of eigenpairs to be computed. The value $\sigma = 7$ reaches in all cases either the lowest runtime or a time close to the lowest runtime.

Shifting strategy Matrix shifting reduces the number of required subspace iterations on all levels. In our experiments, we use a heuristic, which is described in step 12 of Algorithm 3, to automatically determine μ. This heuristic is based on the aggressive shifting technique from [79]. We set μ equal to the current approximate eigenvalue Λ_{jj} with index $j = \lfloor \alpha p \rfloor$. Here α is a value between 0 and 0.5. Table 4 listed iteration counts for different values of α. Results for different numbers of eigenpairs and different error margins are shown. We used values between 0.1 and 1/3 for α in our experiments.

Surface with boundary We applied HSIM to the computation of Laplace–Beltrami eigenproblems on surfaces with boundary. We experimented with Dirichlet and Neumann boundary conditions and used the same hierarchy and basis construction as for surface without boundary. Examples of eigenfunctions on surfaces with boundary are shown in Figure 8. Table 5 shows for an example mesh the runtimes and iteration counts for Dirichlet and Neumann boundary conditions. The runtimes are comparable to the runtimes we get for meshes without boundary and a similar number of vertices.

Boundary	#Eigs	#Iters	Timing		
			Hierarchy	Solve	Total
Dirichlet	50	F|2	5.8	30.0	35.8
	250	F|2|1	19.3	93.7	113.0
Neumann	50	F|2	5.7	29.8	35.4
	250	F|2|1	18.9	94.9	113.3

Table 5: Timings and iteration counts for solving eigenproblems with boundary conditions on the Julius Caesar model with 370k vertices.



(a) Dirichlet

(b) Neumann

Figure 8: The first (left) and tenth (right) eigenvector of the Laplace–Beltrami operator on a surface with boundary using Dirichlet and Neumann boundary conditions are shown.

4.6. COMPARISONS

In this section, we discuss comparisons of HSIM to alternative methods. Laplace–Beltrami eigenproblems are commonly solved in graphics applications using Lanczos methods [68]. Therefore, we begin this section with the comparison to Lanczos solvers. An alternative to Lanczos schemes is the SIM, which is used in structural engineering applications [7]. Since HSIM is based on SIM, this comparison provides a basis to quantify the gains resulting from our hierarchy. The third solver to which we compare HSIM is the Locally Optimal Block Preconditioned Conjugate Gradient Method [37]. Lastly, we compare HSIM to the multilevel correction scheme that was introduced in [17, 43]. We use the same convergence test (4.6) for all methods and set the tolerance to $\varepsilon = 10^{-2}$, which is the value we determined in our experiments, see Section 4.5.

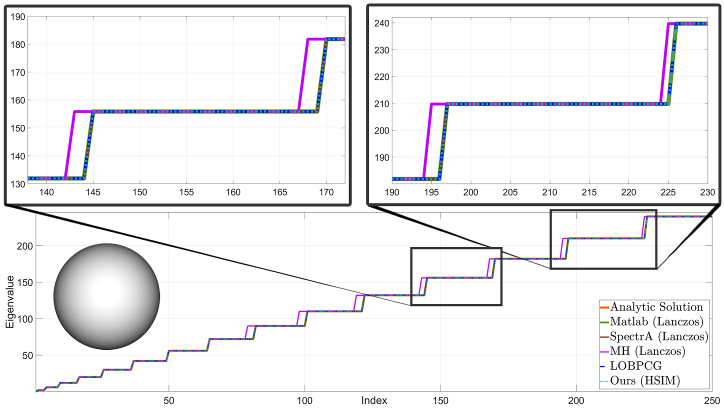
Lanczos methods Methods based on Lanczos iterations are commonly used for solving large-scale, sparse, symmetric eigenproblems. These methods have been studied and improved over decades. ARPACK's implementation of the Implicitly Restarted Lanczos Method is well-established [41]. We compare HSIM with MATLAB's `eigs` that interfaces ARPACK and with SPECTRA [55] that offers an alternative implementation of the Implicitly Restarted Lanczos Method. In addition to that, we compare to the authors' implementation of the band-by-band, shift-and-invert Lanczos solver that was introduced in [68]. We denote this solver by Manifold Harmonics (MH). If a diagonal, or lumped, mass matrix is used in (4.4), the generalized eigenproblem can easily be transformed to an 'ordinary' eigenproblem as described in [68]. We have tested all three Lanczos solvers on the 'ordinary' eigenproblem.

The runtimes for meshes of different complexity and different numbers of eigenpairs are given in Table 6. The listed runtimes for HSIM also include the construction of the hierarchy and prolongation operators. In our experiments, HSIM was consistently faster than all three Lanczos schemes. This is also reflected in the table where HSIM is the fastest method for all combinations of meshes complexity and numbers of eigenpairs. In particular in the difficult cases, where a larger number of eigenpairs have to be computed, HSIM is much faster.

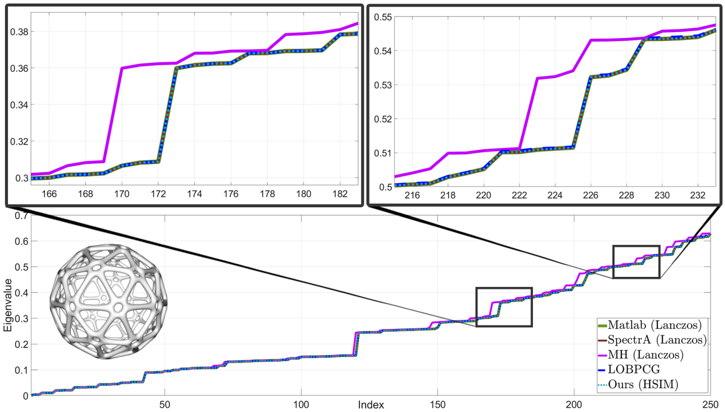
Figure 9 shows plots of the lowest eigenvalues for two surfaces computed with different solvers. On the left side of the figure, numerical approximations of the eigenvalues of the unit sphere computed with the different solvers on a mesh with 320k triangles approximating the sphere are shown. For reference, the analytical solution is included to the plot. On the right side of the figure results for a surface that exhibits different symmetries are shown. SPECTRA and MATLAB applied to the ordinary eigenproblem provided accurate results in our experiments that for the sphere example well-approximate the analytic solution. The results obtained with HSIM match the accuracy of SPECTRA and MATLAB. The band-by-band, shift-and-invert solver [68] meets the convergence tolerance for the individual eigenpairs, but some eigenpairs are skipped. This seems to happen at the transitions between the bands and we have observed it in our experiments consistently for different bandwidths.

<table>
<tr><th rowspan="2">Model (#Vert)</th><th rowspan="2">#Eigs</th><th colspan="2">SIM</th><th>par. SIM</th><th colspan="2">HSIM</th><th colspan="3">Lanczos methods</th><th>Prec. Solver</th></tr>
<tr><th>#Iters</th><th>Time</th><th>Time</th><th>#Iters</th><th>Time</th><th>Matlab</th><th>MH</th><th>SpectrA</th><th>LOBPCG</th></tr>
<tr><td rowspan="5">Sphere (160k)</td><td>50</td><td>7</td><td>49.4</td><td>23.8</td><td>F|1</td><td>11.7</td><td>16.2</td><td>27.2</td><td>24.4</td><td>48.0</td></tr>
<tr><td>250</td><td>7</td><td>274.5</td><td>155.7</td><td>F|2|1</td><td>51.3</td><td>94.3</td><td>285.3</td><td>124.8</td><td>268.3</td></tr>
<tr><td>1000</td><td>7</td><td>1088.0</td><td>642.2</td><td>F|2|1</td><td>165.6</td><td>921.8</td><td>1132.2</td><td>1235.5</td><td>2601.1</td></tr>
<tr><td>2500</td><td>7</td><td>3228.2</td><td>1930.7</td><td>F|2|1</td><td>529.8</td><td>7784.5</td><td>2987.1</td><td>7552.7</td><td>Mem. bound</td></tr>
<tr><td>4000</td><td>8</td><td>10687.8</td><td>8913.0</td><td>F|2|1</td><td>1431.2</td><td>11745.1</td><td>5836.1</td><td>13100.1</td><td>Mem. bound</td></tr>
<tr><td rowspan="4">Rocker Arm (270k)</td><td>50</td><td>8</td><td>74.9</td><td>42.2</td><td>F|1</td><td>14.6</td><td>18.6</td><td>26.4</td><td>25.8</td><td>126.5</td></tr>
<tr><td>250</td><td>8</td><td>541.7</td><td>300.1</td><td>F|2|1</td><td>79.6</td><td>130.3</td><td>178.7</td><td>185.3</td><td>711.5</td></tr>
<tr><td>1000</td><td>8</td><td>2118.9</td><td>1228.0</td><td>F|2|1</td><td>342.1</td><td>1549.0</td><td>696.4</td><td>1359.4</td><td>4014.5</td></tr>
<tr><td>2500</td><td>8</td><td>10278.5</td><td>8658.4</td><td>F|2|1</td><td>1108.1</td><td>13018.3</td><td>1798.9</td><td>9543.0</td><td>Mem. bound</td></tr>
<tr><td rowspan="3">Rolling stage (660k)</td><td>50</td><td>7</td><td>212.5</td><td>102.5</td><td>F|1</td><td>57.9</td><td>62.7</td><td>100.1</td><td>77.7</td><td>384.2</td></tr>
<tr><td>250</td><td>7</td><td>1308.8</td><td>664.4</td><td>F|2|1</td><td>206.6</td><td>362.5</td><td>773.3</td><td>675.4</td><td>1885.2</td></tr>
<tr><td>1000</td><td>7</td><td>8058.2</td><td>5358.9</td><td>F|3|1</td><td>937.8</td><td>4072.6</td><td>3034.5</td><td>8396.0</td><td>Mem. bound</td></tr>
</table>

Table 6: Comparison of HSIM to the (non-hierarchical) SIM, different Lanczos solvers, and LOBPCG. Runtimes are listed in seconds.

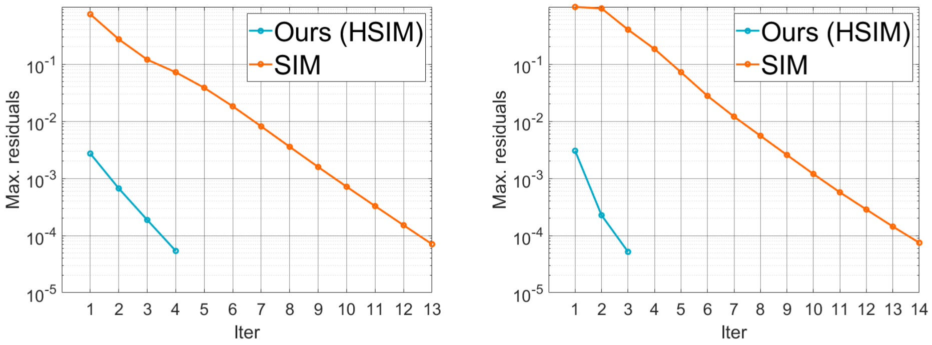


(a) Sphere, 160k



(b) Ball, 90k

Figure 9: The lowest 250 Laplace–Beltrami eigenvalues computed with HSIM and three different Lanczos solvers on a discrete sphere with 160k vertices (left) and a surface with many symmetries and 90k vertices (right). For the sphere the analytic solution is shown as a reference.



(a) 250 eigs (Vase-Lion, 200k V)

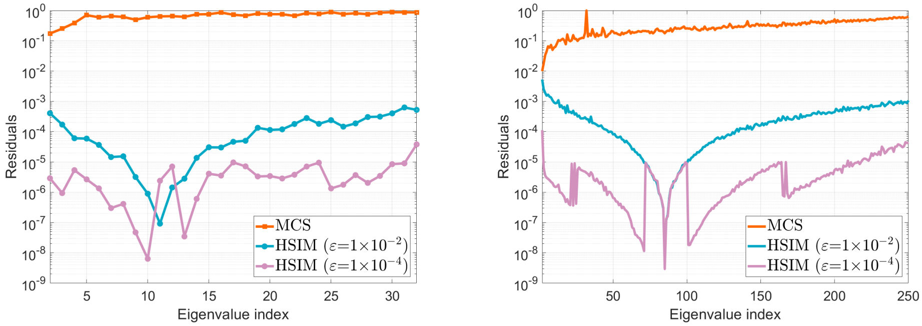
(b) 100 eigs (Eros, 475k V)

Figure 10: Plot of the maximum residual and the numbers of iterations for SIM and HSIM are shown. For HSIM the number of iterations on the finest level is used.

SIM In addition to the runtimes for Lanczos schemes, Table 6 also lists times and iteration counts for the (ordinary, non-hierarchical) SIM. If one compares the number of iterations required by HSIM on the finest level with the number of iterations required by SIM, one sees that HSIM effectively reduces the number of iterations from 7-8 to 1. Accordingly, we observe that HSIM is 4-8 times faster than SIM. The table lists additional runtimes for an optimized SIM implementation in which the linear systems in step 4 of Algorithm 1 are solved in parallel using OpenMP and the dense eigenproblems, step 7 of Algorithm 1, are solved on the GPU using CUDA's CUSOLVER library. Figure 10 shows for two examples how the number of iterations changes if a lower convergence tolerance is requested. It can be seen that the increase of iterations is less for HSIM than for SIM.

LOBPCG The last column of Table 6 lists timings for the Locally Optimal Block Preconditioned Conjugate Gradient Method (LOBPCG). To generate the timings, we used the author's implementation [38]. We experimented with different preconditions and report the best results. These we obtained with the preconditioner $S - \nu Id$, which was suggested in [37]. Here S is the stiffness matrix (of the transformed ordinary eigenvalue problem that we also used for the Lanczos solvers), $\nu \in \mathbb{R}$ is approximately in the middle of the first ten eigenvalues [37], and Id is the identity matrix. The results demonstrate that HSIM can solve the eigenproblems faster than (LOBPCG). Both LOBPCG and HSIM have to solve a number of sparse linear systems of equations to compute the eigenpairs. LOBPCG uses a preconditioned conjugate gradient method and HSIM uses a sparse direct solver for this. For the Laplace matrices we consider, the direct solvers are efficient solvers [10]. Furthermore, HSIM benefits from the fact that a factorization once computed can be reused to solve many systems.

Multilevel correction scheme We compare with the multilevel correction scheme (MCS) from [43, 17], which is an extension of the two-grid scheme from [32]. This method has in common with our HSIM method that for initialization, an eigenvalue problem on the



(a) 32 eigenpairs

(b) 250 eigenpairs

Figure 11: Plot of the residuals of MCS and our novel HSIM eigensolver. Not only HSIM is substantially more accurate, it also has explicit control on the accuracy of the eigenvalues and corresponding eigenvectors. (Dragon model, 150k vertices.)

coarsest grid is solved. However, the multilevel iterations differ substantially from HSIM. In their method, the coarse space is used in all levels and it is enriched by vectors that are computed in the multilevel iterations. An essential difference to HSIM is that HSIM reduces the error on each level to the desired tolerance margin, while in their approach there is no direct control over the accuracy of the solution. The accuracy depends on the approximation quality of the coarse grid and the growth rate between the grids. Therefore an aggressive growth rate, which is essential to the performance of our scheme, would lead to an increase in approximation error. Another substantial difference is that their scheme is focused on computing only one or a few eigenpairs, less than 30 eigenpairs in all shown experiments. This contrasts this work from our setting in which we compute more than a thousand eigenpairs. In Figure 11, we show a plot of the accuracy of the eigenpairs computed with MCS and HSIM with convergence tolerance 10^{-2} and 10^{-4}. The error produced by MCS is orders of magnitude higher than that produced by HSIM. Moreover, the plot on the right shows that the error increases with the index of the eigenvalue. This illustrates the point that MCS is focused on the computation of a few of the lowest eigenpairs. Since the MCS scheme is formulated for regular grids, we use our hierarchy with three levels in the comparisons for both schemes, MCS and HSIM.

4.7. CONCLUSION

We introduce HSIM, a hierarchical solver for sparse eigenvalue problems and evaluate HSIM on the computation of the lowest p eigenpairs of the discrete Laplace–Beltrami operator on triangle surface meshes. HSIM first constructs a hierarchy of nested subspaces of the space functions on the mesh. Then, HSIM iterates from coarse to fine over hierarchy solving the eigenproblem on all levels to the desired accuracy. HSIM is initialized with the solution of the eigenproblem on the coarsest level, which is computed by solving a low-dimensional dense eigenproblem. Our comparisons show that HSIM outperforms state-of-the-art Lanczos solvers and demonstrate the advantages of the hierarchical approach over the plain SIM.

We think that the benefits of HSIM over Lanczos and SIM solvers make HSIM attractive for methods in shape analysis and mesh processing. Therefore, we plan to release our implementation of HSIM.

Our experiments clearly show the advantages of the hierarchical approach for solving Laplace–Beltrami and related eigenproblems on surface meshes. We expect that this type of solver will be further explored. Directions of future work are to explore alternative hierarchies, e.g. wavelet bases on surfaces, and to extend the method to compute not only the lowest but arbitrary eigenpairs. Moreover, the method could be improved by further exploring the possibilities of parallelization of the method and by integrating out-of-core techniques for the computation of larger eigenbases.

A benefit of the HSIM is that it directly works for generalized eigenvalue problems, such as (4.4), and does not require to transform these to ordinary eigenvalue problems. This could be helpful when using the method for solving eigenproblems in which the mass matrix M is not a diagonal matrix, such as the discretization of the Laplace–Beltrami operator with higher-order elements [57].

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback. This project is partly supported by the Indonesia Endowment Fund for Education (LPDP) through a doctoral scholarship for Ahmad Nasikun. For our experiments, we used models from the AIM@SHAPE repository, the Stanford Computer Graphics Laboratory (Stanford 3D Scanning Repository), Turbosquid, Al-Badri and Nelles (Nefertiti), and INRIA.

BIBLIOGRAPHY

[1] Burak Aksoylu, Andrei Khodakovsky, and Peter Schröder. "Multilevel Solvers for Unstructured Surface Meshes". In: *SIAM J. Sci. Comput.* 26.4 (2005), pp. 1146–1165.

[2] Peter Arbenz et al. "A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods". In: *International Journal for Numerical Methods in Engineering* 64.2 (2005), pp. 204–236.

[3] M. Aubry, U. Schlickewei, and D. Cremers. "The wave kernel signature: A quantum mechanical approach to shape analysis". In: *ICCV*. 2011, pp. 1626–1633.

[4] Omri Azencot et al. "An Operator Approach to Tangent Vector Field Processing". In: *Computer Graphics Forum* 32.5 (2013), pp. 73–82.

[5] Randolph E. Bank. "Analysis of a Multilevel Inverse Iteration Procedure for Eigenvalue Problems". In: *SIAM Journal on Numerical Analysis* 19.5 (1982), pp. 886–898.

[6] Klaus-Jürgen Bathe. *Finite element procedures*. 2nd edition. Prentice Hall, 2014.

[7] Klaus-Jürgen Bathe. "The Subspace Iteration Method - Revisited". In: *Computers and Structures* 126 (2013), pp. 177–183.

[8] Klaus-Jürgen Bathe and Seshadri Ramaswamy. "An accelerated subspace iteration method". In: *Computer Methods in Applied Mechanics and Engineering* 23.3 (1980), pp. 313–331.

[9] D. Boscaini et al. "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks". In: *Computer Graphics Forum* 34.5 (2015), pp. 13–23.

[10] Mario Botsch, David Bommes, and Leif Kobbelt. "Efficient Linear System Solvers for Mesh Processing". In: *Mathematics of Surfaces*. Ed. by Ralph R. Martin, Helmut E. Bez, and Malcolm A. Sabin. Vol. 3604. Lecture Notes in Computer Science. Springer, 2005, pp. 62–83.

[11] A. Brandt, S. McCormick, and J. Ruge. "Multigrid Methods for Differential Eigenproblems". In: *SIAM J. Sci. Stat. Comput.* 4.2 (June 1983), pp. 244–260. ISSN: 0196-5204.

[12] Christopher Brandt et al. "Spectral Processing of Tangential Vector Fields". In: *Computer Graphics Forum* 36.6 (2017), pp. 338–353.

[13] James Brannick and Shuhao Cao. *Bootstrap Multigrid for the Shifted Laplace-Beltrami Eigenvalue Problem*. arXiv preprint arXiv:1511.07042. 2015.

[14] Michael M. Bronstein et al. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42.

[15] Joan Bruna et al. "Spectral networks and locally connected networks on graphs". In: *International Conference on Learning Representations* (2014).

[16] Marcel Campen, Martin Heistermann, and Leif Kobbelt. “Practical Anisotropic Geodesy”. In: *Computer Graphics Forum* 32.5 (2013), pp. 63–71.

[17] Hongtao Chen, Hehu Xie, and Fei Xu. “A full multigrid method for eigenvalue problems”. In: *Journal of Computational Physics* 322 (2016), pp. 747–759.

[18] Ming Chuang et al. “Estimating the Laplace–Beltrami Operator by Restricting 3D Functions”. In: *Computer Graphics Forum* 28.5 (2009), pp. 1475–1484.

[19] Luca Cosmo et al. “Isospectralization, or How to Hear Shape, Style, and Correspondence”. In: *IEEE CVPR*. 2019, pp. 7529–7538.

[20] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. “Geodesics in heat: A new approach to computing distance based on heat flow”. In: *ACM Transactions on Graphics (TOG)* 32.5 (2013), pp. 1–11.

[21] Keenan Crane et al. “Digital Geometry Processing with Discrete Exterior Calculus”. In: *ACM SIGGRAPH 2013 courses*. SIGGRAPH ’13. 2013.

[22] Shen Dong et al. “Spectral surface quadrangulation”. In: *ACM Trans. Graph.* 25.3 (2006), pp. 1057–1066.

[23] Jed A. Duersch et al. “A Robust and Efficient Implementation of LOBPCG”. In: *SIAM J. Sci. Comput.* 40.5 (2018), pp. C655–C676.

[24] Y. Eldar et al. “The Farthest Point Strategy for Progressive Image Sampling”. In: *Trans. Img. Proc.* 6.9 (1997), pp. 1305–1315.

[25] Katarzyna Gebal et al. “Shape Analysis Using the Auto Diffusion Function”. In: *Computer Graphics Forum* 28.5 (2009), pp. 1405–1413.

[26] Yu-cai Gong et al. “Comparison of subspace iteration, iterative Ritz vector method and iterative Lanczos method”. In: *Journal of Vibration Engineering* 18.02 (2005), pp. 227–232.

[27] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[28] W. Hackbusch. “On the Computation of Approximate Eigenvalues and Eigenfunctions of Elliptic Operators by Means of a Multi-Grid Method”. In: *SIAM Journal on Numerical Analysis* 16.2 (1979), pp. 201–215. ISSN: 00361429.

[29] N. Halko, P. G. Martinsson, and J. A. Tropp. “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions”. In: *SIAM Review* 53.2 (2011), pp. 217–288.

[30] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. “On the convergence of metric and geometric properties of polyhedral surfaces”. In: *Geometricae Dedicata* 123 (2006), pp. 89–112.

[31] Hugues Hoppe. “Progressive meshes”. In: *ACM SIGGRAPH*. 1996, pp. 99–108.

[32] Xiaozhe Hu and Xiaoliang Cheng. “Acceleration of a two-grid method for eigenvalue problems”. In: *Mathematics of Computation* 80.275 (2011), pp. 1287–1301. ISSN: 00255718, 10886842.

[33] Jin Huang et al. “Spectral quadrangulation with orientation and alignment control”. In: *ACM Trans. Graph.* 27.5 (2008), pp. 1–9.

[34] Qixing Huang et al. “Shape Decomposition Using Modal Analysis”. In: *Computer Graphics Forum* 28.2 (2009), pp. 407–416.

[35] Alec Jacobson, Daniele Panozzo, et al. *libigl: A simple C++ geometry processing library*. http://libigl.github.io/libigl/. 2016.

[36] Zachi Karni and Craig Gotsman. “Spectral Compression of Mesh Geometry”. In: *ACM SIGGRAPH*. 2000, pp. 279–286.

[37] Andrew V Knyazev. “Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method”. In: *SIAM journal on scientific computing* 23.2 (2001), pp. 517–541.

[38] Andrew V. Knyazev et al. “Block Locally Optimal Preconditioned Eigenvalue Xolvers (BLOPEX) in Hypre and PETSc”. In: *SIAM J. Sci. Comput.* 29.5 (2007), pp. 2224–2239.

[39] Artiom Kovnatsky et al. “Coupled quasi-harmonic bases”. In: *Comput. Graph. Forum* 32.2 (2013), pp. 439–448.

[40] Dilip Krishnan, Raanan Fattal, and Richard Szeliski. “Efficient preconditioning of laplacian matrices for computer graphics”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–15.

[41] R. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users’ Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, 1998.

[42] Thibault Lescoat et al. “Spectral Mesh Simplification”. In: *Computer Graphics Forum* 39.2 (2020), pp. 315–324.

[43] Qun Lin and Hehu Xie. “A multi-level correction scheme for eigenvalue problems”. In: *Math. Comp.* 84 (2015), pp. 71–88.

[44] Ruotian Ling et al. “Spectral Quadrangulation with Feature Curve Alignment and Element Size Control”. In: *ACM Trans. Graph.* 34.1 (2014), 11:1–11:11.

[45] Or Litany et al. “Fully Spectral Partial Shape Matching”. In: *Comput. Graph. Forum* 36.2 (2017), pp. 247–258.

[46] Hsueh-Ti Derek Liu, Alec Jacobson, and Maks Ovsjanikov. “Spectral coarsening of geometric operators”. In: *ACM Trans. Graph.* 38.4 (2019), 105:1–105:13.

[47] Janne Martikainen, Tuomo Rossi, and Jari Toivanen. “Computation of a few smallest eigenvalues of elliptic operators using fast elliptic solvers”. In: *Communications in Numerical Methods in Engineering* 17.8 (2001), pp. 521–527.

[48] Stephen F. McCormick. “A Mesh Refinement Method for $Ax = \lambda Bx$”. In: *Mathematics of Computation* 36.154 (1981), pp. 485–498.

[49] Przemyslaw Musialski et al. “Reduced-order Shape Optimization Using Offset Surfaces”. In: *ACM Trans. Graph.* 34.4 (2015), 102:1–102:9.

[50] Boaz Nadler et al. “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.

[51] Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. “Fast Approximation of Laplace–Beltrami Eigenproblems”. In: *Comp. Graph. Forum* 37.5 (2018).

[52] Ahmad Nasikun and Klaus Hildebrandt. “Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems”. In: *ACM Trans. on Graphics* ().

[53] Maks Ovsjanikov et al. “Computing and Processing Correspondences with Functional Maps”. In: *SIGGRAPH ASIA 2016 Courses*. ACM, 2016, 9:1–9:60.

[54] Maks Ovsjanikov et al. “Functional Maps: A Flexible Representation of Maps Between Shapes”. In: *ACM Trans. Graph.* 31.4 (2012), 30:1–30:11.

[55] Yixuan Qiu. *SpectrA: C++ Library For Large Scale Eigenvalue Problems*. https://spectralib.org/. 2015.

[56] Arianna Rampini et al. “Correspondence-Free Region Localization for Partial Shape Similarity via Hamiltonian Spectrum Alignment”. In: *IEEE 3D Vision*. 2019, pp. 37–46.

[57] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. “Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids”. In: *Computer-Aided Design* 38.4 (2006), pp. 342–366.

[58] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. “Laplace-Spectra as Fingerprints for Shape Matching”. In: *Proceedings of the ACM Symposium on Solid and Physical Modeling*. 2005, pp. 101–106.

[59] Emanuele Rodolà et al. “Partial Functional Correspondence”. In: *Comput. Graph. Forum* 36.1 (2017), pp. 222–236.

[60] Raif M. Rustamov. “Laplace–Beltrami eigenfunctions for deformation invariant shape representation”. In: *Symposium on Geometry Processing*. 2007, pp. 225–233.

[61] Raif M. Rustamov et al. “Map-based Exploration of Intrinsic Shape Differences and Variability”. In: *ACM Trans. Graph.* 32.4 (2013), 72:1–72:12.

[62] Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. Vol. 66. Siam, 2011.

[63] Avinash Sharma et al. “Mesh Segmentation Using Laplacian Eigenvectors and Gaussian Mixtures”. In: *Manifold Learning and Its Applications*. 2009.

[64] Nicholas Sharp et al. “Diffusion is All You Need for Learning on Surfaces”. In: *CoRR* abs/2012.00888 (2020). URL: `https://arxiv.org/abs/2012.00888`.

[65] Ran Song et al. “Mesh Saliency via Spectral Processing”. In: *ACM Trans. Graph.* 33.1 (2014), 6:1–6:17.

[66] Klaus Stüben. “A review of algebraic multigrid”. In: *Journal of Computational and Applied Mathematics* 128.1 (2001), pp. 281–309.

[67] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. “A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion.” In: *Computer Graphics Forum* 28.5 (2009), pp. 1383–1392.

[68] Bruno Vallet and Bruno Lévy. “Spectral Geometry Processing with Manifold Harmonics”. In: *Computer Graphics Forum* 27.2 (2008), pp. 251–260.

[69] Libor Váša et al. “Compressing dynamic meshes with geometric Laplacians”. In: *Computer Graphics Forum* 33.2 (2014), pp. 145–154.

[70] Amir Vaxman, Mirela Ben-Chen, and Craig Gotsman. “A Multi-resolution Approach to Heat Kernels on Discrete Surfaces”. In: *ACM Trans. Graph.* 29.4 (2010), 121:1–121:10.

[71] Yu Wang and Justin Solomon. “Intrinsic and extrinsic operators for shape analysis”. In: *Handbook of Numerical Analysis*. Vol. 20. Elsevier, 2019, pp. 41–115.

[72] Max Wardetzky et al. “Discrete quadratic curvature energies”. In: *Computer Aided Geometric Design* 24.8-9 (2007), pp. 499–518.

[73] Christopher K. I. Williams and Matthias Seeger. “Using the Nyström Method to Speed Up Kernel Machines”. In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[74] Edward L Wilson and Tetsuji Itoh. “An eigensolution strategy for large systems”. In: *Computers & Structures* 16.1-4 (1983), pp. 259–265.

[75] Hehu Xie, Lei Zhang, and Houman Owhadi. “Fast Eigenpairs Computation with Operator Adapted Wavelets and Hierarchical Subspace Correction”. In: *SIAM Journal on Numerical Analysis* 57.6 (2019), pp. 2519–2550.

[76] Jinchao Xu and Aihui Zhou. “A Two-Grid Discretization Scheme for Eigenvalue Problems”. In: *Mathematics of Computation* 70.233 (2001), pp. 17–25. ISSN: 00255718, 10886842.

[77] Yidu Yang and Hai Bi. “Two-grid finite element discretization schemes based on shifted-inverse power method for elliptic eigenvalue problems”. In: *SIAM Journal on Numerical Analysis* 49.3/4 (2011), pp. 1602–1624. ISSN: 00361429.

[78] Ning Zhang et al. *An Algebraic Multigrid Method for Eigenvalue Problems in Some Different Cases*. arXiv:1503.08462. 2015.

[79] Qian-Cheng Zhao et al. “Accelerated subspace iteration with aggressive shift”. In: *Computers & structures* 85.19-20 (2007), pp. 1562–1578.

APPENDIX

4.A. JUSTIFICATION OF DESIGN CHOICES

In this section, we present experiments that address the justification of our design and evaluation choices for the Hierarchical Subspace Iteration Method (HSIM).

4.A.1. DISTANCE COMPUTATION

The construction of the basis functions, see Equation (11) in [8], requires the computation of geodesic distances. For the evaluation of HSIM, we used Dijkstra's algorithm on the weighted edge graph of the mesh using the edge lengths as weights. Since the basis functions have local support, we stop the single source Dijkstra computation when all vertices in the support of the basis function have been processed. Alternatives to Dijkstra's algorithm are the Short Term Vector Dijkstra (STVD) algorithm [1] and the Heat Method [3]. Table 4.A.1 compares timings and iteration counts obtained by using Dijkstra's algorithm, the STVD algorithm and the heat method for basis construction. One can see that the required numbers of iterations are similar for all three methods with some slight variations. Therefore, the timings for the case that Dijkstra's algorithm is used are comparable to those when the STVD algorithm is used. There is only a small overhead resulting from the additional computational effort of the STVD algorithm compared to Dijkstra's algorithm. The heat method is much slower since for each point the distance to all other points is computed instead of only in a local neighborhood. We would like to note that there are also possibilities to localize the distance computation with the heat method [4]. This, however, would be beyond the scope of this experiment. These results illustrate our impression that the STVD algorithm or a localized version of the heat method can be used as alternatives for the basis construction. Since we did not observe any substantial advantages of STVD or the Heat method over Dijkstra's algorithm in our experiments, and to keep the method simple, we used Dijkstra's algorithm for our evaluation of HSIM.

4.A.2. SAMPLING METHOD

The nested function spaces we use for HSIM are constructed from a vertex hierarchy, which assigns a level to every vertex of the mesh. We use farthest point sampling for computing the distribution of the vertices. In this section, we show examples of vertex hierarchies on different meshes to illustrate why we think farthest point sampling is suitable for this process. In the examples, we use meshes with spatially varying resolution, in some areas of the surfaces the triangles are much smaller than in others. Figure 4.A.1 shows vertex hierarchies on four surfaces. The hierarchies shown have four levels, color-coded in red, blue, and green, with the finest levels encompassing all vertices and not shown. It can be seen that the vertices in the different levels are distributed fairly uniformly over the surface despite the irregular mesh. In Figure 4.A.2 four more examples

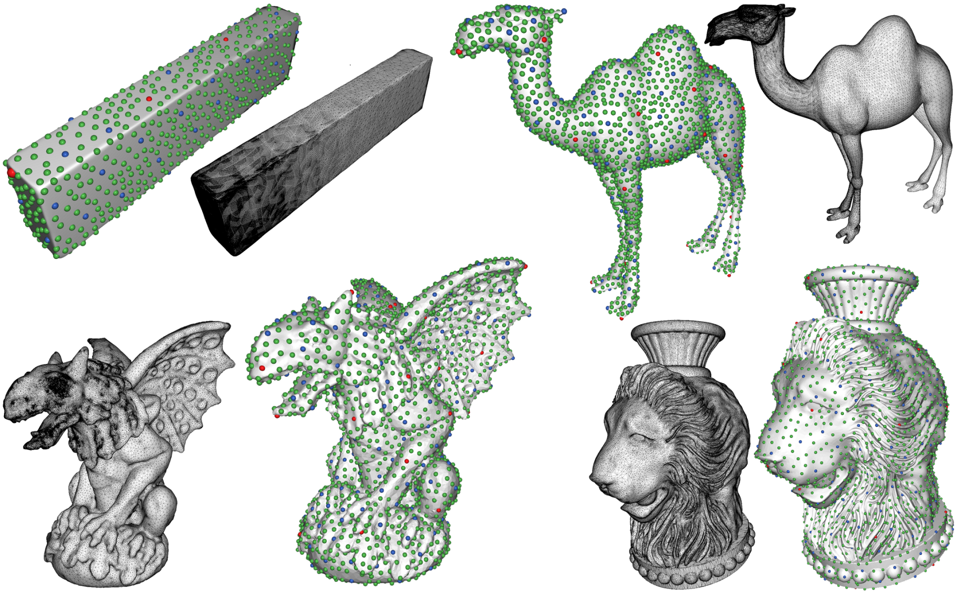


Figure 4.A.1: Results of vertex hierarchy construction using farthest point sampling on meshes with spatially varying sampling densities are shown.

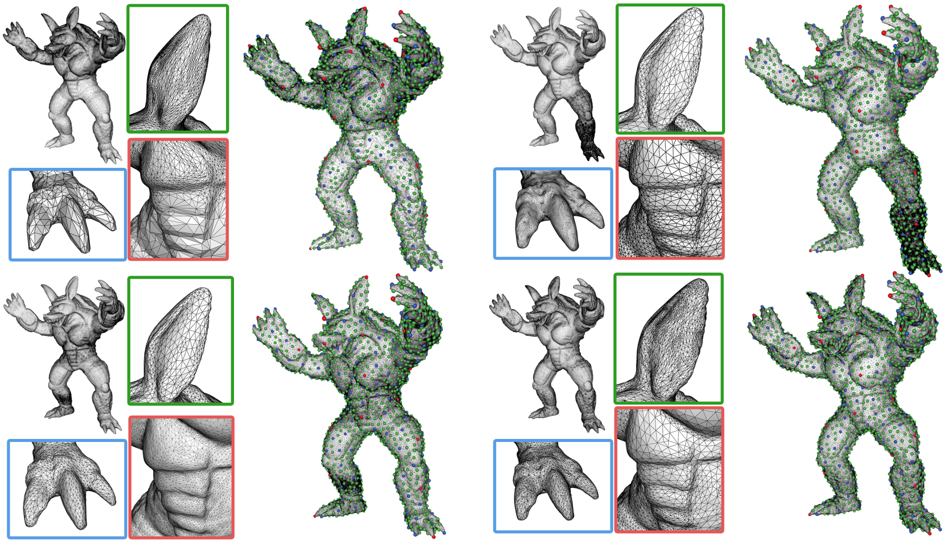
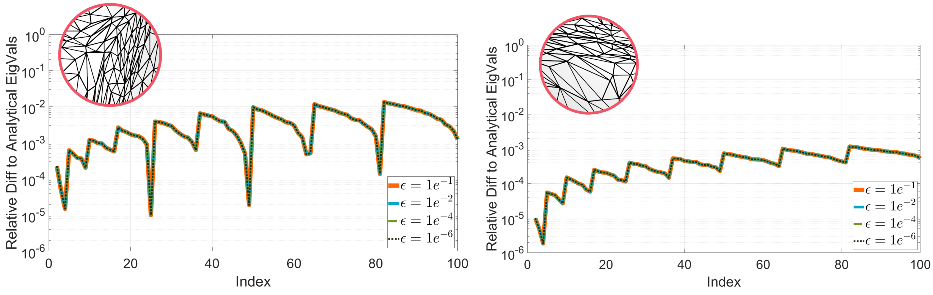
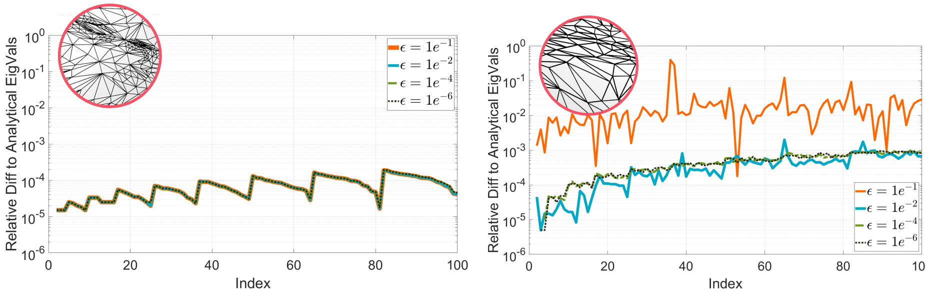


Figure 4.A.2: Results of vertex hierarchy construction using farthest point sampling are shown for four meshes that approximate the same surface but have different spatially varying sampling densities.

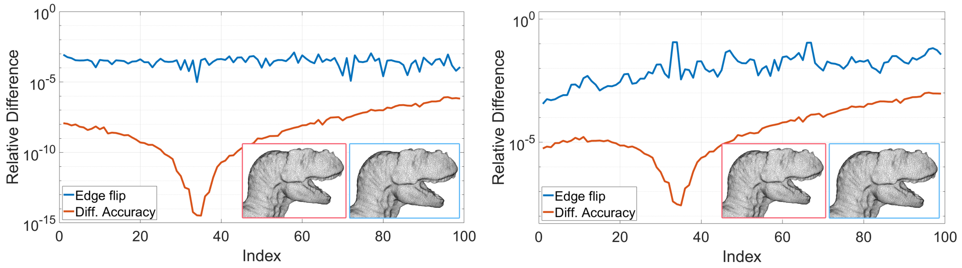


(a) HSIM, 10k vertices (b) HSIM, 100k vertices



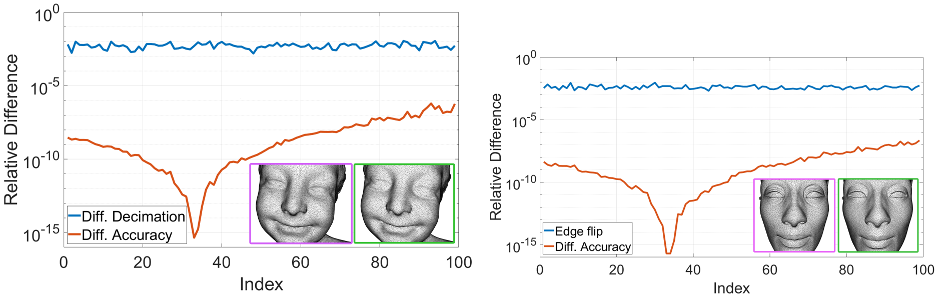
(c) HSIM, 1m vertices (d) IPM, 100k vertices

Figure 4.A.3: Relative difference of numerical approximations of the eigenvalues of the unit sphere to the analytic solutions.

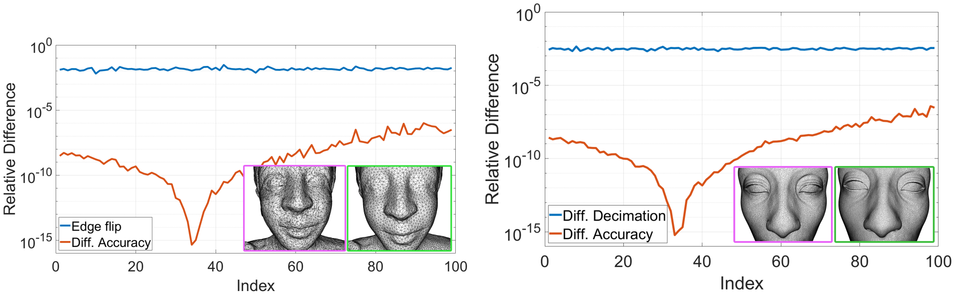


(a) Eigenvalues (b) Eigenvectors

Figure 4.A.4: Comparisons of the relative difference of the eigenvalues and the eigenvectors between two meshes that approximate the same surface (blue graph) and solutions for different tolerance on one of the meshes (red graph).



(a) Bimba, 100k (b) Nefertiti 100k



(c) Ramses 100k (d) Ramses, 500k.

Figure 4.A.5: For four pairs of meshes, each pair approximating the same surface, comparisons of the relative differences between the eigenvalues of the two meshes (blue graphs) and solutions for different convergence tolerance on one of the meshes (red graphs) are shown.

Model (#Verts.)	Type	Acc.	Hier.	Solve		Total
				Iter	Time	
Gargoyle (85k)	Dijkstra	1e-2	2.2	F|1|1	8.4	10.5
		1e-4		F|3|3	15.8	18.0
	STVD	1e-2	4.0	F|1|1	8.9	13.0
		1e-4		F|4|4	20.8	24.8
	Heat M.	1e-2	203.6	F|1|1	10.4	214.0
		1e-4		F|4|3	22.6	226.2
Fertility (270k)	Dijkstra	1e-2	9.6	F|1|1	26.2	35.8
		1e-4		F|3|4	59.3	68.9
	STVD	1e-2	30.4	F|1|1	27.3	57.7
		1e-4		F|3|2	44.8	75.2
	Heat M.	1e-2	1218.8	F|1|1	30.8	1249.6
		1e-4		F|3|3	53.7	1272.5
Oil-pump (570k)	Dijkstra	1e-2	30.3	F|1|1	63.6	93.9
		1e-4		F|4|3	112.6	142.8
	STVD	1e-2	100.1	F|1|1	65.6	166.6
		1e-4		F|4|3	113.1	214.1
	Heat M.	1e-2	4339.4	F|2|2	88.9	4628.3
		1e-4		F|4|4	136.6	4676.0

Table 4.A.1: The timings and iteration counts for computing 100 eigenpairs on different meshes with three different schemes for approximating the geodesic distances are shown. The timings for the construction of the hierarchy are additionally listed.

are shown. In this case, we show vertex hierarchies with the same numbers of vertices in each level on different meshes that approximate the same surface. As illustrated in the shown results, we consider the farthest point sampling as a suitable method to build up the vertex hierarchies for HSIM. Nevertheless, one could also use alternatives like Poisson disk sampling, which is used in [7] for the construction of function spaces.

4.A.3. CONVERGENCE TOLERANCE

This paragraph includes further experiments related to the discussion of the convergence tolerance that we used for the evaluation of HSIM, see also Section 5 of [8]. Figure 4.A.3 shows a variant of Figure 7 from [8], where we consider non-regular meshes inscribed to the sphere. Figure 4.A.4 shows a variant of Figure 8 from [8], using a different mesh.

In Figure 4.A.5, we show results of an additional experiment. As in Figure 4.A.4, we computed eigenpairs on two meshes with tolerances $\varepsilon = 10^{-2}$ and as reference with $\varepsilon = 10^{-8}$. We generated the meshes by simplifying one mesh with two different mesh coarsening algorithms. We used the Bimba mesh with 500k vertices to get two simplified meshes with 100k vertices each, the Nefertiti mesh with 1m vertices to obtain two simplified meshes with 100k vertices and the Ramses mesh with 750k vertices to get two meshes with 500k vertices and two meshes with 100k vertices. In Figure 4.A.5, we plot for all four pairs of meshes the differences between the reference results that are computed

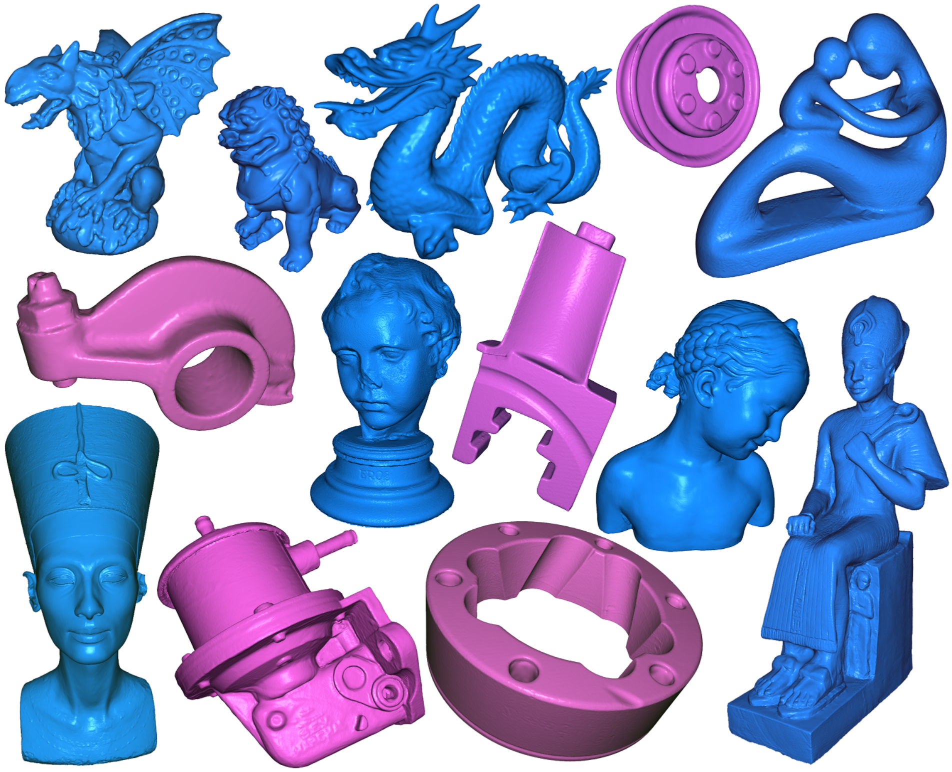


Figure 4.A.6: Renderings of the meshes used for the comparisons listed in Table 4.A.2.

with a tolerance of $\varepsilon = 10^{-8}$ on both meshes and for one mesh the difference between the results for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-8}$. For all four pairs of meshes, the difference between the reference results on the two meshes is much larger than the difference between the results for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-8}$.

4.B. COMPARISONS

In this section, we show additional comparisons to alternative approaches for solving eigenproblems.

4.B.1. LANCZOS AND PRECONDITIONED EIGENSOLVER

In Section 6 of [8], the timings of HSIM are compared with the timings of Lanczos solvers and LOBPCG. Table 4.A.2 shows additional results that complement Table 4 in [8].

4.B.2. FAST APPROXIMATION

We compare HSIM with the fast approximation method from [7]. The approximation method has the advantage that the computation times are much shorter and storing the approximate eigenfunctions requires less memory. On the other hand, the approxi-

Model	#Verts.	#Eigs.	HSIM	Matlab	LOBPCG
Gargoyle	85k	50	4.2	5.1	27.1
		250	19.9	49.3	154.7
		1000	88.6	442.3	995.0
Chinese Dragon	135k	50	7.5	7.6	57.7
		250	30.5	56.3	298.0
		1000	127.5	523.3	1678.7
Dragon	150k	50	7.2	9.6	83.0
		250	36.5	65.7	325.1
		1000	143.8	795.9	2102.5
Blade	200k	50	10.5	14.1	97.9
		250	49.2	93.9	453.6
		1000	177.8	1091.9	2591.0
Fertility	240k	50	14.6	16.6	133.3
		250	90.8	121.6	678.6
		1000	236.1	1369.9	4003.0
Rocker-Arm	270k	50	17.5	22.2	135.9
		250	73.9	175.8	744.4
		1000	252.1	1537.2	4837.9
Pulley	300k	50	19.5	21.1	228.3
		250	85.3	222.1	837.3
		1000	339.5	1795.0	5416.9
Eros	400k	50	30.7	43.6	194.5
		250	127.2	267.7	1305.5
		1000	322.6	2748.4	Mem. Bound
Bimba	500k	50	29.4	31.4	236.1
		250	132.8	254.9	1255.3
		1000	569.3	3208.0	Mem. Bound
Oilpump	570k	50	41.1	46.1	310.3
		250	154.2	315.6	1864.4
		1000	690.9	3354.7	Mem. Bound
Rolling stage	680k	50	54.6	57.8	326.3
		250	197.6	386.5	2301.2
		1000	891.6	4064.1	Mem. Bound
Ramses	825k	50	49.1	62.6	458.9
		250	221.4	413.4	2339.9
		1000	1149.1	4979.2	Mem. Bound
Nefertiti	1m	50	64.0	62.8	396.2
		250	305.4	682.7	2482.2
		500	277.9	1654.5	Mem. Bound

Table 4.A.2: Comparisons of timings of HSIM, Matlab's Lanczos solver and LOBPCG for Laplace–Beltrami eigenproblems on different meshes. Renderings of the meshes are show in Figure 4.A.6.

Model	#Eigs	Laplacian				Hamiltonian (t=0.1)				Hamiltonian (t=1.0)			
		Hier.	Solve	#Iter	Total	Hier.	Solve	#Iter	Total	Hier.	Solve	#Iter	Total
Cube (25k)	50	0.3	2.1	F|1	2.4	0.3	2.1	F|1	2.4	0.3	2.1	F|1	2.4
	250	0.6	6.7	F|1|1	7.3	0.7	6.6	F|1|1	7.3	0.7	9.2	F|3|1	9.9
	1000	0.7	67.1	F|5|2	66.8	0.8	65.9	F|5|2	66.8	0.6	67.3	F|5|2	67.9
Blade (200k)	50	2.8	7.4	F|1	10.2	2.9	7.4	F|1	10.3	2.8	10.5	F|2	13.3
	250	7.3	42.8	F|2|1	50.1	7.1	42.7	F|2|1	49.8	7.2	45.8	F|3|1	53.0
	1000	8.8	158.7	F|2|1	167.5	8.8	167.5	F|2|1	176.3	8.9	204.0	F|4|1	212.9
Bimba (500k)	50	7.9	22.6	F|1	30.5	8.0	23.8	F|1	31.7	7.7	30.9	F|2	38.7
	250	26.4	105.1	F|2|1	131.6	26.4	106.8	F|2|1	133.2	26.3	121.3	F|3|1	147.6
	1000	34.1	519.5	F|3|1	553.6	33.5	510.0	F|3|1	543.5	34.9	643.3	F|6|1	678.2

Table 4.A.3: Timings and iteration counts for Laplace–Beltrami and Hamiltonian eigenproblems are shown.

mation errors of [7] are much larger than the errors resulting from HSIM. The top row of Figure 4.B.1 shows plots of residuals of eigenpairs computed with the approximation scheme from [7] and compares them with the residuals from HSIM with tolerances $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$. The residuals obtained for the approximation scheme from [7] are 10^{0}. In contrast, HSIM allows for controlling the residuals. The bottom row of the Figure additionally shows the computed eigenvalues. While visually there is no difference between the two HSIM results, the eigenvalues computed with [7] differ significantly from the results of HSIM. We would like to note that in [7] it is advised to use only the first half of the computed eigenvalues. However, significant deviations can be observed in the first half as well.

4.C. GENERALIZATION

4.C.1. HAMILTONIAN OPERATORS

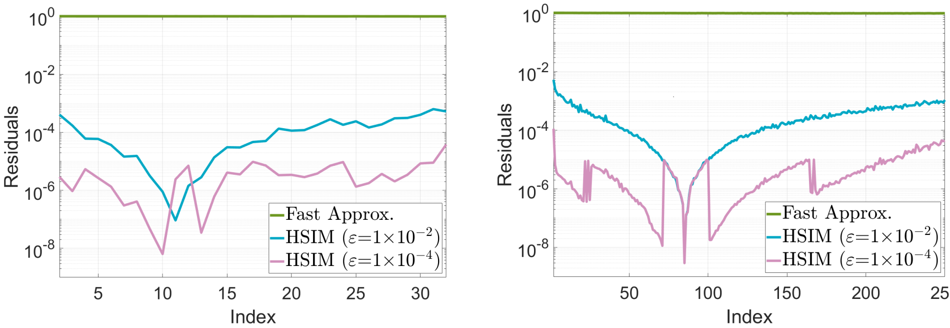
Our evaluation of HSIM is focused on Laplace–Beltrami eigenproblems. In this section, we consider a related operator, the Hamiltonian operator, and present some results for solving Hamiltonian eigenproblems using HSIM. For a background on Hamiltonian operators and their use in spectral analysis, we refer to [2]. The Hamilton operators on surfaces we consider are of the form

$$H : u \to \Delta u + V u,$$

where Δ is the Laplace–Beltrami operator and V a scalar potential function. For our experiments, we used the scalar potential

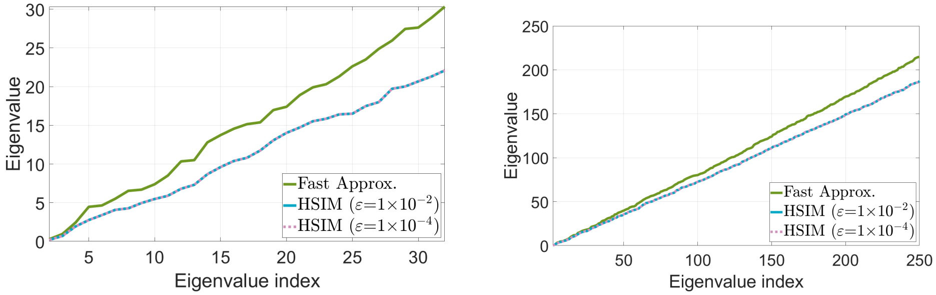
$$V = t(\kappa_1^2 + \kappa_2^2),$$

where $t \in \mathbb{R}^{\geq 0}$ and κ_1 and κ_2 are the principal curvatures. The eigenmodes of this operator have been studied in the context of shape analysis in [5]. In contrast to the Laplace–Beltrami eigenfunctions, the eigenfunctions of this operator depend not only on the intrinsic properties of the surface but also on its extrinsic curvatures. Even for $t = 0.1$, the eigenfunctions of this operator are fundamentally different from those of the Laplace–Beltrami operator as illustrated in Figure 4.B.2. Table 4.A.3 lists iteration counts and timings for solving Hamiltonian eigenproblems for $\alpha = 0.1$ and $\alpha = 1$. As a reference, we also list the timings for the corresponding Laplace–Beltrami eigenproblem. For $\alpha = 0.1$,



(a) 32 eigenpairs

(b) 250 eigenpairs



(c) 32 eigenpairs

(d) 250 eigenpairs

Figure 4.B.1: Top row: Plot of the residuals for the computation of the lowest 32 and 250 Laplace–Beltrami eigenvalues of the Dragon model with 150k vertices. Results for the fast approximation scheme from [7] and HSIM with tolerance $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-4}$ are shown. Bottom row: The computed eigenvalues are plotted.

we obtain almost the same timings as for the Laplace–Beltrami eigenproblems and for $\alpha = 1$, we noticed in some cases an increase of the required computation time of up to 30%.

4.D. APPLICATIONS

In this section, we consider methods that use the Laplace–Beltrami eigenfunctions for shape analysis and processing. We demonstrate that the methods can benefit from using a larger number of eigenfunctions. HSIM facilitates the computation of larger numbers of eigenfunctions.

4.D.1. SHAPE SIGNATURES

We first consider the Heat Kernel Signature [9] as an example of a shape signature. Figure 4.C.1 shows the Heat Kernel Signature color-coded on two meshes. For both meshes, results using 100 and 1000 eigenfunctions are shown. One can see that the surface details such as the curls of the Chinese lion model are better resolved when 1000 eigenfunctions are used. As a consequence, the Heat Kernel Distance delivers better results for finding

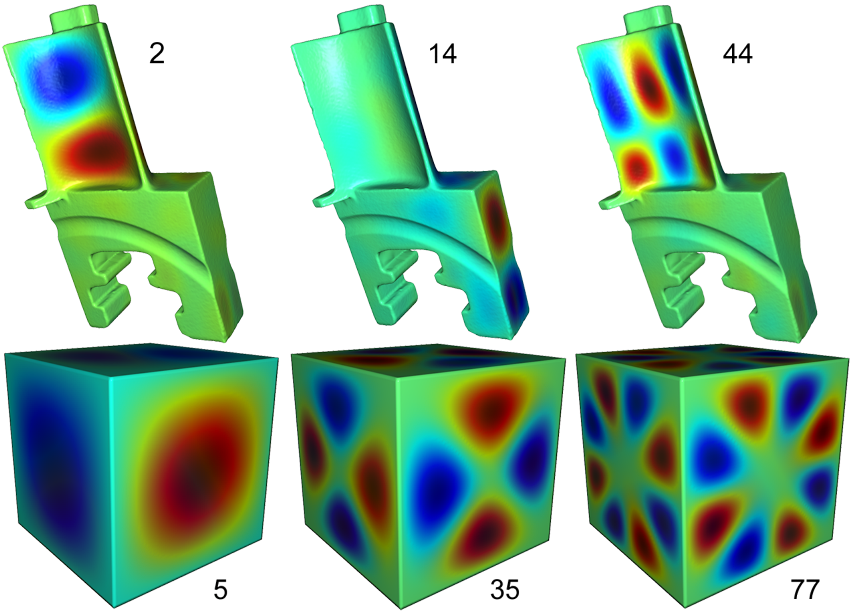
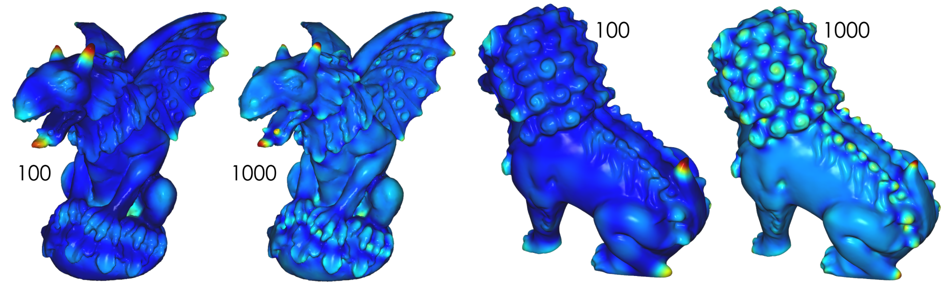


Figure 4.B.2: Eigenfunctions of the Hamilton operator are shown.

similar points on a surface when more eigenfunctions are used. Figure 4.C.2 shows results where similar points to a given point are searched. The results are shown by binary color-coding, where similar points are orange. On the Armadillo mesh, a point at the fingertip is given and on the dinosaur mesh, a point at the toe is given. It can be seen that if 1000 eigenfunctions are used, on both meshes all fingertips and toe tips are found. For 100 eigenfunctions this is not the case. Only about half of the fingertips and toes are found.

4.D.2. PROJECTION

Methods such as mesh filtering [10] and mesh compression [6] need to project the embedding of a surface to the space spanned by the lowest n eigenfunctions. Figure 4.C.3 shows the results of this projection for the centaur mesh with different values of n ranging from 10 to 4000. One can see that the higher the number of eigenfunctions is, the more surface details are preserved. Even when 2000 eigenfunctions are used the resulting projection is visually smoother than the original mesh.



(a) Gargoyle (b) Chinese Dragon

Figure 4.C.1: Heat Kernel Signatures (HKS) computed with 100 and 1000 eigenpairs are shown.

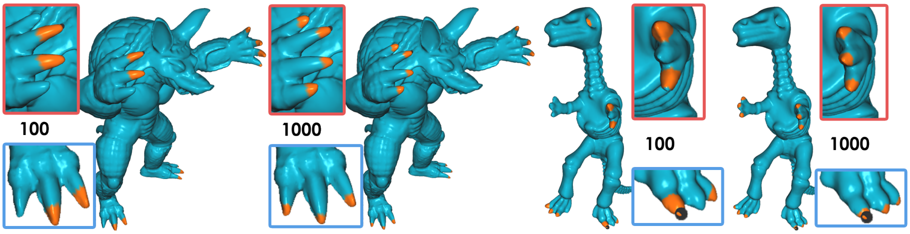


Figure 4.C.2: Points similar to the fingertip of the armadillo mesh and to the toe tip for the dinosaur mesh are indicated by binary color-coding. The similarity is computed using the heat kernel distance. Results for heat kernel distance estimation using 100 and 1000 eigenpairs are shown.

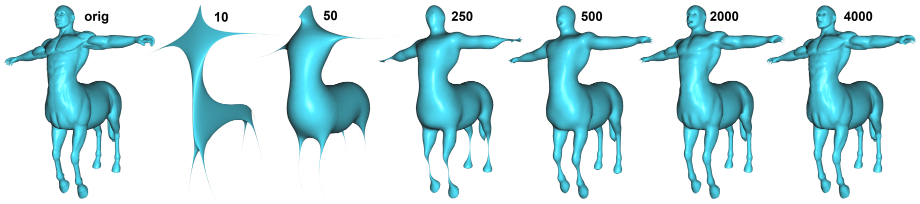


Figure 4.C.3: Geometric reconstruction of the Centaur model (left-most) using an increasing number of Laplace–Beltrami eigenfunctions. A sufficient number of eigenfunctions is required to obtain reconstruction that preserves details of the shape.

BIBLIOGRAPHY

[1] Marcel Campen, Martin Heistermann, and Leif Kobbelt. “Practical Anisotropic Geodesy”. In: *Computer Graphics Forum* 32.5 (2013), pp. 63–71.

[2] Yoni Choukroun et al. “Hamiltonian operator for spectral shape analysis”. In: *IEEE transactions on visualization and computer graphics* 26.2 (2018), pp. 1320–1331.

[3] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. “Geodesics in heat: A new approach to computing distance based on heat flow”. In: *ACM Transactions on Graphics (TOG)* 32.5 (2013), pp. 1–11.

[4] Philipp Herholz, Timothy A. Davis, and Marc Alexa. “Localized solutions of sparse linear systems for geometry processing”. In: *ACM Trans. Graph.* 36.6 (2017), 183:1–183:8.

[5] Klaus Hildebrandt et al. “Modal shape analysis beyond Laplacian”. In: *Computer Aided Geometric Design* 29.5 (2012), pp. 204–218.

[6] Zachi Karni and Craig Gotsman. “Spectral Compression of Mesh Geometry”. In: *ACM SIGGRAPH*. 2000, pp. 279–286.

[7] Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. “Fast Approximation of Laplace–Beltrami Eigenproblems”. In: *Comp. Graph. Forum* 37.5 (2018).

[8] Ahmad Nasikun and Klaus Hildebrandt. “Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems”. In: *ACM Trans. on Graphics* ().

[9] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. “A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion.” In: *Computer Graphics Forum* 28.5 (2009), pp. 1383–1392.

[10] Bruno Vallet and Bruno Lévy. “Spectral Geometry Processing with Manifold Harmonics”. In: *Computer Graphics Forum* 27.2 (2008), pp. 251–260.

5

CONCLUSION

Spectral methods have proven to be well-suited for solving various tasks in geometry processing. However, a crucial step in applying spectral methods for geometry processing is computing the smallest eigenvalues and the corresponding eigenvectors of the Laplace operator. This is a computationally expensive step, especially for large surface meshes, thus making it a challenging bottleneck in spectral geometry processing. In addition, spectral methods require an extensive amount of memory to store the eigenfunctions. All eigenfunctions have global support so they must be represented by a dense matrix, which is memory extensive for large meshes.

Our techniques enable efficient computation and storage of the spectra and eigenfunctions of Laplace operators. In Chapter 2, we introduced a fast approximation algorithm for the Laplace–Beltrami eigenproblem, which allows for a significantly faster computation time and a substantial decrease in memory required to store the approximated eigenfunctions. We presented sparsified eigenfields of the Hodge–Laplacian, in Chapter 3, which serve as the basis for fast approximation algorithms for the design and processing of tangential vector, $n-$vector, and tensor fields. Our novel, efficient hierarchical solver for Laplace–Beltrami eigenproblems, HSIM, is presented in Chapter 4. We showed that HSIM outperforms state-of-the-art eigensolvers, such as Implicitly Restarted and shift-and-invert Lanczos methods, particularly when a large number of eigenpairs of a complex mesh model are sought.

In summary, we believe that reducing the computation burden of eigenproblems is an essential step for the widespread adoption of spectral methods in geometry processing. We observed that approximate eigenfunctions are sufficient for many spectral geometry applications, and therefore advocate their use when applicable. We also learned that the sparsification of eigenvectors and approximation of eigenfunctions using the subspace method can lower the storage requirements of spectral geometry processing methods. Moreover, the approximate eigenfunctions provide a good initialization for the hierarchical iterative solver, reducing the required number of iterations significantly.

5.1. LIMITATION AND FUTURE WORK

We limited our scope to focus on triangular mesh representations of surfaces. Extending our approaches to other polygonal mesh representations is an interesting direction, provided that a Laplace operator is readily available. We also think that generalizing the methods to point clouds is an appealing direction for future research, since the use of point cloud is increasing, particularly in spectral and geometric deep learning.

Important elements of our methods are constructions of subspaces and hierarchies of nested spaces, which are ingredients for model order reduction and multigrid approaches, respectively. Unfortunately, these steps take a considerable amount of time. Further research into more efficient constructions of subspaces and hierarchies is an interesting research direction.

With the increased availability of 3D data, robustness is a challenging problem in solving the eigenproblems in a large collection of data, such as the Thingy10K dataset[1], in which no guarantee is provided on manifoldness, quality of the triangulation, and regularity. Moreover, in the context of deep learning, online computation of the eigenfunctions is preferable over storing them, due to memory restriction. Storing a considerable number of eigenfunctions for all 3D models in the dataset requires gigantic memory. Unless one is willing to store and to access them offline, on the disks, in which the retrieval is costly. As a future research direction, we consider constructing a discrete Laplace operator that can robustly handle such representations and designing a robust eigensolver for the operator. Another challenge relates to the increasing complexity of 3D geometric models. For example, (reconstructed) models in the Tanks and Temples dataset[2] have several million vertices. Devising efficient algorithms for extremely large models is an interesting avenue for research in spectral geometry processing. To do so, we envision a set of techniques using out-of-core computations or a divide-and-conquer approach where the domain is divided in patches for efficient computation.

In terms of applications where accuracy is required, one has to, unfortunately, accept the costly computation of the eigenpairs. On contrary, one may opt for fast approximation of eigenvalues and the corresponding eigenfunctions when accuracy is not the main concern. An example is spectral surface filtering, in which the filtering function is generally smooth. Hence one might not need to compute every eigenfunction accurately. This is the case in our fast approximation algorithm. The approximated eigenfunctions are linear combinations of the reference eigenfunctions with a similar eigenvalue. It is therefore interesting to identify which applications need accuracy and for which tasks approximations of the spectra and eigenfunctions are sufficient. For a large mesh, computing a considerable number of eigenfunctions is generally very costly, and therefore fast approximation could be preferable. Regarding the trade-off of accuracy for speed, our fast approximation algorithms (Chapter 2 and 3) do not provide the users with control over the accuracy of the resulting sparsified eigenbasis. This remains as another direction for future work, including a guaranteed error bound.

---

[1] https://ten-thousand-models.appspot.com/

[2] https://www.tanksandtemples.org/

ACKNOWLEDGEMENTS

A long journey, such as pursuing a doctoral study, is not possible without a great supporting system. This thesis represents not only my academic journey at Delft University of Technology (TU Delft) but also lessons that I learned along the way from colleagues and friends that I met during the course of my doctoral study. For that, I am truly indebted to them.

I would like to start by expressing my sincere gratitude to my academic supervisor, Dr. Klaus Hildebrandt, for guiding me to learn so much about various interesting topics in geometry processing during my 4.5-year stay at TU Delft. Your supervisory and guidance skills are fantastic. I thank you for making the time to meet and discuss our research progress almost every week. I consider myself super lucky to be one of your doctoral students. I also have learned a lot from your great dedication toward research and education, and balancing them with endless love to your family. For my promotor, Prof. Elmar Eisemann, I thank you for providing me such a great opportunity to be part of the Computer Graphics and Visualization (CGV) group and to strive together along with excellent researchers at the group that you lead. Your deep and broad knowledge in computer graphics, excellent leadership, great communication skills, and exceptional teaching finesse are always inspiring.

My sincere thanks to faculty members of the CGV group, Prof. Anna Vilanova, Assoc. Prof. Rafael Bidarra, Dr. Ricardo Marroquim, Dr. Thomas Höllt, Dr. Michael Weinmann, and Dr. Petr Kellnhofer. It is such a pleasure to learn and to interact with all of you during the course of my Ph.D. The discussions that we all have during the *VisuLunch* have been very crucial in the development of my knowledge in computer graphics.

I would like to thank my co-authors, Christopher Brandt and Ruben Wiersma. Christopher has been super helpful in helping me with both research and personal matters at the start of my study. He is such an inspiring researcher with a deep conceptual understanding and technical skills. I thank you for all the discussions on vector fields data processing, simulation, and research in general. I hope all the best for you, Lara, and lovely little Filine. From the start, Ruben has shown what a fantastic doctoral student he is. It is my honor to work with you on the geometric deep learning project and on being teaching assistants for the computational simulation course for the architecture students.

I am extremely indebted to Markus Billeter and Leo Scandolo. Markus has been my way out whenever I have questions about computer programming. His excellence in efficient programming is one of a kind. Leo, together with Jerry, has been showcasing an amazing job in being the motor of the group activities at CGV and is super helpful to everyone. I thank you for your assistance in providing nice rendering for my works and in helping me with a variety of technical questions that I have.

To postdoctoral researchers in CGV (Tim Balint, Pablo Bauszat, Thomas Kroes, Martin Skrodzki, Pierre Ambrosini, Mark Winter, and Amal Parakkat), please accept my deepest thanks for the interaction that we had in the CGV group. It is always nice to be surrounded by distinguished researchers in the office. For that, I thank Baran, Annemieke, Niels, (late) Leo Torok, and Mathijs. To Jerry, Faizan, Mijael, Nicolas, Yang, Xuejiao, Marcos, Peiteng, Mark, Tom, Alex, Tim (Kol), Nicola, Jingtang, Changgong, Victor, Pedro, Rafael, Qiaomu, Ali, and Chaoran, thank you very much for all the fun that we have at CGV. You all have been positively influential for the completion of my doctoral degree. Thank you to the postdocs and PhD colleagues who helped me with the proofreading and translation of my dissertation: Annemieke, Ruben, Mark, Jerry, Baran, and Fabrizio.

My special thanks go to Ruud and Bart, who consistently support our research by providing exceptional helps with all technical problems that we encounter: from fixing our NetID issue, providing an awesome microphone for Eurographics video recording, to helping out in bringing my PC and monitors to my place. You guys are the hidden super heroes of our CGV group. On this opportunity, I would like to also express my sincere gratitude to the secretariat team whose support has been immensely important to my research and study. Thank you Lauretta, Marloes, and Sandra for all your support with all administrative works without which our research and study would not be running well.

Many thanks to the Indonesian Endowment Fund for Education (LPDP) for giving me the opportunity to pursue my dream of becoming a Ph.D. in one of the finest institutions in Europe. I truly appreciate the confidence that you showed in me by providing funding for my studies and research at TU Delft. I would like to thank faculty members and academic staff at the Department of Electrical and Information Engineering, Universitas Gadjah Mada (UGM) for their continuous support and motivation, particularly Pak Sardjiya, Pak Hanung, Pak Teguh Bharata Adji, and Pak Lesnanto. I cannot wait to rejoin the department for a fruitful collaboration in research and teaching with the faculty members of the department.

I am very grateful to be surrounded by awesome friends that have kept me motivated in completing my Ph.D. journey at TU Delft. I highly appreciate the friendship from fellow Indonesian doctoral students in Delft, both at IHE Delft and TU Delft. I hope you guys can have more frequent Ph.D. gatherings when the pandemic is over. To my friends at KMD Delft and FORKOM-NL, I really appreciate the time and effort that you all spent together to help and to cherish the community. Special thanks to Pak Aries, Mas Agung, Kang Mamin, Mas Ando, and Mas Sebri for the inspiration, guidance, and teamwork at KMD. To the board member of KMD 2019/2020: Aldy, Andya, Angka, Ardiana, Arry, Asmita, Bramka, Firas, Indra, Krisna, Mikhta, Retna, Sindu, and Sulton, thank you very much for the time that you spent growing our community. My highest respect and admiration to Pak Eko, Kang Rihan, Pak Raymon, Mas Arianto, Mas Verdy, Mas Fajar, Kang Juris, Mas Ega, Mas Pandu, Mas Dicko, Mas Yasir, Mas Yasri, and Kang Yudi at FORKOM-NL that have given so much to Indonesian community in the Netherlands. It has been a lot of pleasure to be the neighbor of Mas Wicak and Mbak Desi, Mas Aga and Mbak Kitty, Mas Irfan and Mbak Liza, Mas Sebri and Mbak Siska, Mas Mikhta and Mbak Rara, and Fidllan and Devi. Thank you so much for your kindness during our stay in Delft. My thanks also go to PPI Delft for always organizing cool activities for Indonesian students in Delft and helping us when we need it. Thank you Shiddiq, Ryan, Agung, Sulton, Firdo, and the board members of PPI Delft.

I would like to dedicate the last paragraph to the most important supporting system during my doctoral study. To my wife, Alfina Dewi, I cannot possibly thank you enough for the unconditional support that you give, particularly when I am down and need an extra push at my research and study. Your *dosirak* for my lunch at the office are the best, representing how magnificent your support to me is. This Ph.D. is also yours! Aziz and Ziza, when you grow up and can read this, ayah wants to thank the two of you for always bringing up extra energy and smiles every day. You two are amazing and I am super lucky to be your father. To my father (Pak Turki), my mother (Bu Ruminah), my brother (Irul), and my sister (Nisa), thank you very much for all of the prayers and good wishes for me. Thank you to my father and mother-in-law (Pak Suprihadi dan Bu Dewi) for every prayer and your guidance. We cannot wait to see you all once we return to Indonesia.

Alhamdulillah.

Delft, 3 Feb 2022.

Ahmad Nasikun

CURRICULUM VITÆ

Ahmad NASIKUN

08-01-1988	Born in Jepara, Indonesia.

EDUCATION

2007–2012	Bachelor of Electrical Engineering **Universitas Gadjah Mada (UGM)**, Indonesia (with an exchange program at Daejeon University in 2009) *Bachelor project:* Simulation of UAVs Autonomously Approaching Certain Target Using Dubins Algorithm *Supervisor:* Assoc. Prof. Teguh Bharata Adji
2013–2015	Master of Electrical Engineering and Computer Science **Seoul National University (SNU)**, South Korea *Thesis:* 3D Printing of Deformable Objects *Promotor:* Prof. Kim Myung-Soo

PROFESSIONAL EXPERIENCE

02.2012–08.2012	Assistant to Public Relations division Department of Electrical Engineering, Universitas Gadjah Mada
10.2015–06.2016	Research assistant Department of Electrical Engineering, Universitas Gadjah Mada
07.2017–now	Faculty member Department of Electrical and Information Engineering Universitas Gadjah Mada

Awards and Scholarship

2010	First Winner of *Mahasiswa Berprestasi* (the Most Outstanding Students Award) Universitas Gadjah Mada
2011	Best Paper at WCOMLIS 2011 World Congress of Muslim Librarians and Scientists at International Islamic University Malaysia (IIUM)
2012	Best Graduate of Dept. of Electrical Engineering and Information Technology with GPA 3.92/4.00
2012–2015	Recipient of KGSP (Korean Government Scholarship Program) for Korean Language at Keimyung University and Master Degree at Seoul National University (SNU) South Korea
2017–2022	Recipient of LPDP Scholarship Indonesia Endowment Fund for Education for doctoral program at Delft University of Technology

Academic Activities

2015–2017	Organizing Comittee of CITEE and ICITEE (International) Conference on Information Technology and Electrical Engineering KMTIL Thailand and DTETI UGM
Since 2015	Reviewer of JNTETI National Journal for Electrical Engineering and Information Technology Department of Electrical and Information Engineering Universitas Gadjah Mada
Since 2015	Reviewer of ICITEE International Conference on Information Technology and Electrical Engineering KMTIL Thailand and DTETI UGM
Since 09.2020	Reviewer of **IEEE Transactions on Signal Processing**
07.2021	Reviewer for ISITIA ITS International Seminar of Intelligent Technology and Its Applications Institut Teknologi Sepuluh November (ITS), Surabaya

LIST OF PUBLICATIONS

3. **Nasikun, A.** and Hildebrandt, K. (2022, April). *The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems.* In ACM Transactions on Graphics (Vol. 41, No.2, pp. 1-14).

2. **Nasikun, A.**, Brandt, C., and Hildebrandt, K. (2020, May). *Locally supported tangential vector, n-vector, and tensor fields.* In Computer Graphics Forum (Vol. 39, No. 2, pp. 203-217).

1. **Nasikun, A.**, Brandt, C., and Hildebrandt, K., (2018, August). *Fast approximation of Laplace–Beltrami eigenproblems.* In Computer Graphics Forum (Vol. 37, No. 5, pp. 121-134).