

MODEL REDUCTION FOR INTERACTIVE 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 1 April 2019 at 10:00 o'clock

by

Christopher BRANDT

Master of Science in Mathematics, Freie Universität Berlin, Germany
born in Heidelberg, Germany

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,
Prof.dr. E. Eisemann,
Dr. K.A. Hildebrandt,

chairperson
Delft University of Technology, promotor
Delft University of Technology, daily supervisor

Independent members:

Prof.dr. C. Witteveen,
Prof.dr. L. Kobbelt,
Prof.dr. M. Ovsjanikov,
Dr. M. Ben-Chen,
Dr. J.M. Thiery,
Prof.dr.ing. C.M. Hein,

Delft University of Technology
RWTH Aachen, Germany
Ecole Polytechnique, France
Technion, Israel
Telecom ParisTech, France
Delft University of Technology, reserve member



Keywords: geometry, simulation, computer graphics, model reduction, mathematics

Printed by: Gildeprint

Cover by: Christopher Brandt

Copyright © 2019 by C. Brandt

ISBN 978.94.6323.562.4

An electronic version of this dissertation is available at

<http://repository.tudelft.nl/>.

CONTENTS

Summary	vii
Samenvatting	ix
Introduction	1
1 Spectral Processing of Tangential Vector Fields	7
1.1 Overview	7
1.2 Related Work	8
1.3 Background: Hodge Decomposition of Vector Fields	10
1.4 Discrete Hodge Decomposition	12
1.5 Discrete Hodge–Laplace Operator for Vector Fields	14
1.6 Fourier Representation of Tangential Vector Fields	17
1.7 Tangential Vector Field Splines	20
1.8 Applications and Experiments	22
1.8.1 Computation of the eigenfields	22
1.8.2 Spline editor for tangential fields.	22
1.8.3 Fur design	23
1.8.4 Speeding-up soft constrained design.	25
1.8.5 Spectral analysis and filtering	26
1.8.6 Compression.	28
1.9 Conclusion	29
References	30
2 Interactive Modeling of n-field Splines	37
2.1 Overview	37
2.2 Related Work	39
2.3 Background: n -vector fields.	41
2.4 Harmonic Energy for n -vector fields	44
2.5 n -field Splines	46
2.6 Matrix Representation	50
2.7 Computing n -field Splines	51
2.8 Real-Time Editing.	54
2.9 Experiments	55
2.10 Applications	64
2.11 Conclusion	66
References	67
Appendices	71
2.A Derivation of the discrete harmonic energy	71
2.B Linear system for n -direction field splines	73

3	Hyper-Reduced Projective Dynamics	75
3.1	Overview	75
3.2	Related Work	77
3.3	Background: Projective Dynamics	79
3.4	Vertex Positions Subspace Construction	81
3.5	Constraint Projections Fitting Method	82
3.6	Hyper-Reduced Projective Dynamics	85
3.7	Results	87
3.8	Conclusion	93
	References	94
	Appendices	99
3.A	Implementation Details.	99
3.B	Blend-Skinning Subspace Construction.	100
3.C	Subspace for Constraint Projections	100
3.D	Comparison to Simplification Schemes	101
4	Geometric Flows of Curves in Shape Space	105
4.1	Overview	105
4.2	Related Work	106
4.3	Background: Deformation Energies and Shape Averaging.	108
4.4	Discrete Curve Flow in Shape Space.	109
4.5	Analysis of the Flow and the Computation of Geodesics	110
4.6	Efficient Computation of the Flow	112
4.7	Applications and Experiments	114
4.8	Conclusion	122
	References	123
5	Compressed Vibration Modes of Elastic Bodies	127
5.1	Overview	127
5.2	Related Work	128
5.3	Background: Deformation Energies and Vibration Modes	130
5.4	Compressed Vibration Modes.	132
5.5	Computation of Compressed Vibration Modes	133
5.6	L^1 constrained vibration modes.	135
5.7	Applications	137
5.8	Experiments	140
5.9	Conclusion	147
	References	148
	Appendices	151
5.A	Analysis of the ICCM scheme for the L^1 constrained optimization	151
6	Optimal Spline Approximation via ℓ_0-Minimization	153
6.1	Overview	153
6.2	Related work	154
6.3	Optimal spline approximation via ℓ_0 -minimization.	155
6.4	Numerical optimization of ℓ_0 -regularized problems	159

6.5 Experiments and comparisons	162
6.6 Conclusion	168
References	169
Appendices	173
6.A Explicit Constraint Operators	173
7 Conclusion	175
Acknowledgements	179
Curriculum Vitæ	181
List of Publications	183

SUMMARY

The research field of geometry processing is concerned with the representation, analysis, modeling, simulation and optimization of geometric data. In this thesis, we introduce novel techniques and efficient algorithms for problems in geometry processing, such as the modeling and simulation of elastic deformable objects, the design of tangential vector fields or the automatic generation of spline curves. The complexity of the geometric data determines the computation time of algorithms within these applications. The high resolution of modern meshes, for example, poses a big challenge when geometric processing tools are expected to perform at interactive rates. To this end the goal of this thesis is to introduce fast approximation techniques for problems in geometry processing. One line of research to achieve this goal will be to introduce novel model order reduction techniques to problems in geometry processing. Model order reduction is a concept to reduce the computational complexity of models in numerical simulations, energy optimizations and modeling problems. New specialized model order reduction approaches are introduced and existing techniques are applied to enhance tools within the field of geometry processing. In addition to introducing model reduction techniques, we make several other contributions to the field. We present novel discrete differential operators and higher order smoothness energies for the modeling of tangential (n -)vector fields. These are used, to develop novel tools for the modeling of fur, stroke based renderings or anisotropic reflection properties on meshes. We propose a geometric flow for curves in shape space that allows for the processing and creation of animations of elastic deformable objects. A new optimization scheme for sparsity regularized functionals is introduced and used to compute natural, localized deformations of geometrical objects. Lastly, we reformulate the classical problem of spline optimization as a sparsity regularized optimization problem.

In Chapters 1 and 2 we introduce an approach that enables the creation and design of smooth tangential vector and n -rotational-symmetry fields on curved surfaces in real time. Such fields can represent directional information on surfaces, and are therefore useful for mesh texturing, remeshing, anisotropic shading, fluid simulations on surfaces, surface segmentation and surface reconstruction. The main contributions here are the introduction of discrete differential operators for tangential vector and n -fields that allow for specific spectral decompositions, higher-order smoothness energies and the novel concept of modeling tangential vector field splines and n -field splines. To enable real-time editing, a reduction using the spectral decompositions and a reformulation of the involved linear systems is proposed. Various novel applications are proposed that make use of these real-time modeling tools.

Chapters 3, 4 and 5 explore the field of elastic deformable objects for computer graphics. Modeling and simulating elastic objects is essential for many applications in com-

puter graphics. However, complex meshes and the non-linear behavior of elastic materials pose major barriers to achieve interactive rates for such tools. Chapter 3 describes a scheme for the real-time simulation of deformable objects. We introduce specific linear subspaces and a novel approximation scheme for local non-linearities. This hyper-reduction scheme is combined with a recent simulation framework called *Projective Dynamics*, which makes use of specific energy potentials and a local-global variational time-stepping scheme. As a result, we achieve realistic and detailed dynamics for high-resolution meshes in real-time framerates.

Chapter 4 considers the concept of shape spaces for elastic deformable objects. Shape spaces endow the space of deformations of an elastic deformable object with a Riemannian structure and thus allow us to view continuous deformations of a mesh as curves in this space. A novel geometric flow for the processing of such curves is introduced, with which we can compute shortest paths in shape space, regularize animations or create new animations from key frames. The application of existing model reduction approaches in combination with this flow enables computations in shape space in a fraction of the time required by previous approaches.

Chapter 5 introduces localized vibration modes, a novel basis designed to represent localized deformations of elastic deformable objects. Vibration modes are defined as the eigenfunctions of the Hessian of an elastic energy. Eigenfunctions corresponding to the lowest eigenvalues correspond to infinitesimal, global deformations that require the lowest amount of elastic energy. The basis we propose maintains this idea while additionally enforcing locality of the deformations through a sparsity term. A novel algorithm for the minimization of L_2 -constrained, sparsity regularized quadratic functionals is introduced. The resulting basis can be used for reduction approaches in elasticity problems, which is demonstrated for linearized, reduced simulations and elastic modeling. Additionally it allows for the analysis and segmentation of deformable objects.

In Chapter 6, we consider the classical problem of finding a spline curve that best matches an input curve or general curve-like data. The goal here is to find a spline curve defined on a low number of control points, in order to reduce the amount of data that is required to represent the input curve. By a novel reformulation of the problem we show that this amounts to solving a sparsity regularized quadratic optimization problem, similar to the problem encountered in Chapter 5. We test and compare several algorithms for the optimization of the problem and propose specific tools to approximate optimal splines. Amongst more classical applications of the resulting tools for spline optimization we show that the approach can be used to compress and regularize motion data.

In summary, each of these six chapters provide novel approaches, techniques and insights revolving around interactive geometry processing and model reduction. These contributions offer enhancements to the state-of-the-art in simulation, elastic modeling, tangential field design, and spline optimization.

SAMENVATTING

Het onderzoeksdomein van geometrieverwerking betreft de representatie, analyse, modellering, simulatie en optimalisatie van geometrische data. In dit proefschrift introduceren we nieuwe technieken en efficiënte algoritmes voor problemen in de geometrieverwerking, zoals het modelleren en simuleren van elastische vervormbare objecten, het ontwerpen van tangentiële vectorvelden of het automatisch genereren van spline krommen. De complexiteit van de geometrische gegevens bepaalt voor deze toepassingen de benodigde rekentijd voor de algoritmen. Bijvoorbeeld, de hoge resolutie van moderne meshes is een grote uitdaging wanneer van geometrische verwerkingstools verwacht wordt interactief te presteren. Het doel van dit proefschrift is het introduceren van snelle benaderingstechnieken voor problemen in de geometrische verwerking. Een van de taken in onderzoek om dit doel te bereiken is het introduceren van nieuwe zogenaamde “model-order-reduction” technieken voor problemen in de geometrieverwerking. Het concept van model-order-reduction is om de rekencomplexiteit te verminderen van modellen in numerieke simulaties, energieoptimalisaties en modelleerproblemen. Nieuwe gespecialiseerde model-order-reduction-benaderingen worden geïntroduceerd en bestaande technieken worden toegepast om de tools binnen het domein van geometrieverwerking te verbeteren. Naast de introductie van modelreductietechnieken leveren we nog verschillende andere bijdragen aan het veld. We presenteren nieuwe discrete differentiële operatoren en gladdere energieën voor het modelleren van tangentiële (n -)vectorvelden. Deze worden gebruikt om nieuwe tools te ontwikkelen voor het modelleren van vacht, stroke-based rendering of anisotrope reflectie-eigenschappen op meshes. Een nieuw optimalisatieschema voor schaarsheid geregulariseerde functies wordt geïntroduceerd en gebruikt om natuurlijke, gelokaliseerde vervormingen van geometrische objecten te berekenen. Ten slotte herformuleren we het klassieke probleem van spline-optimalisatie als een schaarsheid geregulariseerd optimalisatieprobleem.

In de hoofdstukken 1 en 2 introduceren we een aanpak die de creatie en het ontwerp van gladde tangentiële vectorvelden en n -rotatie-symmetrievelden op gebogen oppervlakken in real time mogelijk maakt. Dergelijke velden kunnen richtingsinformatie op oppervlakken weergeven en zijn daarom nuttig voor het bepalen van de textuur van een mesh, remeshing, anisotrope schakering, vloeistofsimulaties op oppervlakken, oppervlaktesegmentatie en oppervlaktereconstructie. De belangrijkste bijdragen hier zijn de introductie van discrete differentiële operatoren voor tangentiële vectorvelden en n -velden die specifieke spectrale decomposities, hogere orde gladheid energieën en het nieuwe concept van het modelleren van tangentiële vectorveld splines en n -veld splines. Om het aanpassen in real-time mogelijk te maken, wordt een reductie met behulp van de spectrale decomposities en een herformulering van de betrokken lineaire systemen voorgesteld. Er worden verschillende nieuwe toepassingen voorgesteld die gebruik maken van deze real-time modelleringstools.

In de hoofdstukken 3, 4 en 5 wordt het veld van elastische vervormbare objecten voor computergraphics verkend. Het modelleren en simuleren van elastische objecten is essentieel voor veel toepassingen in computergraphics. Complexe meshes en het niet-lineaire gedrag van elastische materialen vormen echter lastige problemen voor het maken van interactieve tools. Hoofdstuk 3 beschrijft een schema voor de real-time simulatie van vervormbare objecten. We introduceren specifieke lineaire deelruimten en een nieuw benaderingsschema voor lokale niet-lineairiteiten. Dit hyperreductieschema wordt gecombineerd met de specifieke energiepotentialen en een aanpak die gebruik maakt van variatierekening van de tijdstappen die in een recente simulatie aanpak met de naam Projective Dynamics verschijnen.

Hoofdstuk 4 behandelt het concept van vormruimten voor elastische vervormbare objecten. Vormruimten geven de ruimte van vervormingen van een elastisch vervormbaar object met een Riemannische structuur en stellen ons in staat om continue vervormingen van een mesh als krommen in deze ruimte te bekijken. Er wordt een nieuwe geometrische stroming voor de verwerking van dergelijke krommen geïntroduceerd, waarmee we de kortste paden in de vormruimte kunnen berekenen, animaties kunnen regulariseren of nieuwe animaties kunnen maken gebaseerd op gegeven animatieframes. De toepassing van bestaande modelreductiemethoden in combinatie met deze stroming maakt berekeningen in de vormruimte mogelijk in een fractie van de tijd die eerdere benaderingen nodig hadden.

Hoofdstuk 5 introduceert gelokaliseerde trilling modi, een nieuwe basis die ontworpen is om gelokaliseerde vervormingen van elastische vervormbare objecten te representeren. Trilling modi worden gedefinieerd als de eigenfuncties van de Hessiaan van een elastische energie. Eigenfuncties die overeenkomen met de laagste eigenwaarden komen overeen met oneindig kleine, globale vervormingen die de laagste hoeveelheid elastische energie vereisen. De basis die wij voorstellen handhaaft dit idee en dwingt tegelijkertijd de lokaalheid van de vervormingen af door middel van een schaarsheid term. Een nieuw algoritme voor het minimaliseren van L_2 -beheerste, schaarsheid geregeliseerde kwadratische functies wordt geïntroduceerd. De resulterende basis kan gebruikt worden voor reductie aanpakken van elasticiteitsproblemen, wat gedemonstreerd wordt voor gelineariseerde, gereduceerde simulaties en elastische modellering. Daarnaast maakt het de analyse en segmentatie van vervormbare objecten mogelijk.

In hoofdstuk 6 behandelen we het klassieke probleem van het vinden van een spline curve die het best overeenkomt met een gegeven curve of algemene curve-achtige data. Het doel hier is om een spline curve te vinden die gedefinieerd is op een laag aantal controlepunten, om zo de hoeveelheid gegevens die nodig is om de gegeven curve weer te geven te verminderen. Door een nieuwe herformulering van het probleem laten we zien dat dit neerkomt op het oplossen van een schaarsheid geregeliseerde kwadratische optimalisatie probleem, net zoals het probleem dat we in hoofdstuk 5 tegenkwamen. We testen en vergelijken verschillende algoritmes voor de optimalisatie van het probleem en stellen specifieke tools voor om de optimale splines te benaderen. Onder de meer

klassieke toepassingen van de resulterende tools voor spline optimalisatie tonen we aan dat de aanpak gebruikt kan worden om bewegingsgegevens te comprimeren en te regulariseren.

INTRODUCTION

Geometry processing is a field of research dedicated to the design of algorithms for the acquisition, (re-)construction, design, analysis, manipulation and simulation of geometrical data. It provides indispensable tools that allow us to model real-life phenomena on a computer. The field combines aspects from mathematics, computer science, engineering and physics. In particular, differential geometry and numerical analysis provide the theoretical basis, which is required to talk about shape, its representation, analysis and manipulation, in a rigorous way. Computer science guides the design and efficient implementation of the algorithms to carry out the desired tasks. Engineering provides means to generate input for and output from such algorithms, e.g. in the form of scanning or printing geometrical objects. Where motion and deformation are concerned, physics provide theoretically and experimentally founded concepts and formulas that yield plausible results for simulations or modeling tools.

To properly represent a complex geometrical object in a computer, a huge amount of data is required. The processing of such data is computationally intensive and consumes a huge amount of resources, but users expect tools to process geometrical data to work at interactive rates. Interactivity is essential since creative processes thrive from instant feedback through directly visible results. To address this contradiction, a lot of research in geometry processing is concerned with devising more efficient algorithms to process data, and more efficient representations of the data itself.

In an interactive setting, limitations on how to deal with the data are present. When a simulation involves highly resolved meshes, it is impossible to perform complex operations on each vertex of the mesh. Computing the smoothest function approximating certain data on a grid involves solving a linear system. For large systems, this process can no longer be performed at interactive rates. Modern GPUs are able to carry out thousands of similar operations in parallel, but this only reduces computation time in cases where the algorithms can be split into small, local and independent tasks. For many tasks in geometry processing, parallelization is not possible or would produce an unacceptable overhead. Even for parallelizable tasks there is a certain threshold on the size of the input data after which interactivity is lost.

Model order reduction (or simply *model reduction*, in the following) is a concept that can provide solutions to this dilemma. It aims to reduce the overall complexity of numerical algorithms for mathematical models, such as simulation, control, and design problems. One aspect of model reduction is to provide representations that can approximate the system's state with less data, e.g. replacing the list of the positions of all vertices of a mesh by a few coordinates that describe the shape in some abstract way. Another aspect is to reduce the number of operations required to process the state of the system,

such as the evaluation of some geometric quantity associated to each vertex. When both of these aspects are combined, in a fashion such that the complexity of the final algorithm does not depend on the original size of the system anymore, it guarantees that the algorithm can perform in a complexity that is independent of the size of the input data. The field of model reduction emerged from the study of finite element discretizations for systems of ordinary equations, but we use the term in its generality, *i.e.* wherever dimensionality and complexity reductions are used to accelerate numerical algorithms for mathematical models.

This thesis is dedicated to the study of model reduction approaches in the field of geometry processing. Various reduced representations and further approximations are presented, which then yield accelerated numerical algorithms for novel and classical applications in geometry processing. The applications considered here range from the modeling of tangential vector- and n -fields over elastic simulation and modeling to compressing curve-like data. While these applications are vastly different in terms of underlying theory and use cases, the approaches to model reduction are surprisingly similar.

Most reduction approaches in this thesis make use of linear subspaces. This implies that the approximated system's state is a linear combination of a finite number of basis vectors or functions. The coefficients of this linear combination yield a reduced representation, whose size can be chosen independently of the number of variables in the full system. How to construct these subspaces such that a large number of typical states of the system can be well approximated is an ongoing area of research, and will be investigated in this thesis for several specific problems. In some cases, the restriction of the problem to such a linear subspace yields a natural, or canonical, reduction of the algorithm, that results in a fully reduced system. In other cases, this restriction does not yield a reduction of the algorithms complexity. For example, if a geometric quantity needs to be evaluated at every vertex in a non-linear manner, the mesh would first need to be reconstructed from its reduced representation, in order to use the resulting vertex positions to evaluate the quantity. In such cases, another layer of approximation is required, a *hyper-reduction*. Whereas linear subspaces are a fairly universal approach to reducing the size of the problem, approximation schemes of the latter kind vary greatly, depending on the specific problem and approximation requirements. In this thesis we will study examples where a linear subspace is enough to yield a fully reduced system, as well as examples where a hyper reduction is required.

Chapters 1 to 6 are based on the publications that are listed at the end of this thesis. As such, they contain novel contributions to the field of geometry processing, which are listed at the beginning of each chapter. Each chapter either focuses on a specific task that benefits from model reduction approaches (*e.g.* the processing curves in the space of elastic shapes or the design of tangential vector fields), or on the introduction of advances in model reduction approaches for specific tasks (*e.g.* a novel hyper-reduction scheme for simulation or the approximation of curve-like data by sparse splines).

Chapters 1 and 2 are dedicated to the study of tangential vector- and n -fields on tri-

angle meshes. Specifically we consider piecewise constant fields, where one tangential (n -)vector is defined on each triangle. Such fields can represent geometric data defined on the surface of the mesh and as such are used in various design applications that are presented in the respective chapters. We develop interactive tools for the design and manipulation of both types of fields. The computationally intensive task is to find the smoothest field that satisfies a set of user specified constraints. To this end, specific subspaces of tangential fields are designed, based on differential operators that are carefully discretized for our specific representation. Since the smoothness energies considered in both chapters are quadratic in the fields, a model reduction approach that enables fast computation of the fields can be limited to constructing linear subspaces, as the complexity for the energy evaluation can then be canonically reduced. Even so, our specific choice of a basis decouples the energy term in a way that reduces the complexity of the evaluation even further.

In Chapter 1 we establish a specific discrete Hodge–Laplace operator for tangential vector fields, based on the Hodge decomposition, which enables a Fourier representation of tangential vector fields. We construct a linear subspace from the lowest part of the spectrum of the Hodge–Laplace operator. Each basis vector is either divergence-free, curl-free or harmonic, therefore the basis refines the Hodge decomposition of vector fields. This enables directly working in curl-free or divergence-free spaces, easily removing curl or divergence from existing fields or specialized filters on the respective parts of the spectrum. We show applications of such filters for the analysis and compression of fields. Tangential vector field splines are then defined as minimizers of a higher order smoothness energy subject to interpolation constraints. A tool for the interactive modeling of such splines is devised and applied to the editing of fur on complex meshes in real time.

In Chapter 2, we extend the above concepts to tangential n -rotational-symmetry fields (n -fields), that is, fields which possess n vectors at each point of the surface, exhibiting an n -fold rotational symmetry. We introduce a specific discretization of such fields and extend the higher-order smoothness energy from Chapter 1 to n -fields. From this energy, we obtain a Laplace operator for n -fields, which can be used to construct linear subspaces by considering the lowest part of its spectrum. n -field splines are then defined analogously to vector field splines and it is shown how the modeling of n -fields using hard constraints on directions or singularities greatly benefits from the higher order energies. Combining the novel subspaces with a reformulation of the resulting linear system enables interactive editing of n -fields on meshes of arbitrary size for the first time.

Chapters 3, 4 and 5 make use of the concepts of deformation and physical elasticity. The deformations of shapes are quantified by elastic energies, that are discretized on triangle or tetrahedral meshes. To properly retain certain physical phenomena, *non-linear* elastic energies are required. This poses a difficulty to applying model reduction approaches that was not present in Chapters 1 and 2. A reduced representation alone will no longer be enough to achieve interactive rates, since another layer of approxima-

tion for the non-linearities will be required. Aside from the common theme of elasticity and deformation, each of these three chapters has a vastly different focus.

In Chapter 3 we devise a novel hyper-reduction scheme for a specific simulation framework called *Projective Dynamics*. An approximation scheme for specific energy potentials that appear in Projective Dynamics is introduced. Together with a novel, fast, scalable and general subspace construction for the approximation of the vertex positions, real-time framerates for the simulation of highly complex meshes (> 1.6 million tetrahedrons) can be achieved on a consumer desktop computer. The simulation framework supports multiple types of objects (shells, solids, springs), is unconditionally stable, allows for changing the material parameters in real time and requires low precomputation times.

In Chapter 4 elastic energies are used to devise a physically-motivated metric on Riemannian *shape spaces*, *i.e.* spaces containing (a subset of the) deformations of a shape. This allows notions such as shortest paths between two deformations of the same shape (*i.e.* a geodesic in shape space), spline curves within the space of deformations of a shape or smoothly varying deformations. Computations in such shape spaces are expensive. For example, the degrees of freedom to compute a geodesic in shape space are those of multiple (tens to hundreds) of copies of a complex shape, as the whole (discretized) curve is the unknown objective of a non-linear energy functional. We accelerate computations in shape space by making use of existing model reduction techniques and introducing an optimization scheme that employs a novel geometric flow. The flow iteratively smooths curves in shape space by local averaging operations, akin to smoothing approaches for polygonal curves. As linear subspaces we either use a principal component analysis of existing collections of deformations of a shape or an existing subspace construction from prior work that uses derivatives of shapes with respect to an interpolation parameter. For the fast evaluation of (the gradients of) the elastic energy functions, we make use of two previously introduced approximation schemes: *ghost meshes* or the *optimized cubature*. The model reduction approaches, together with the novel flow, enable fast computations of curves and lines in shape space. Applications include shape interpolation, creating animations from key frames, regularizing motion or removing linear and temporal artifacts from animations.

Chapter 5 is dedicated to the construction of a specific linear subspace to represent localized deformations of a shape. We extend the notion of natural vibration modes of an elastic energy to a one parameter family of bases, which correspond to low frequency deformations that are localized to certain areas or volumes of a shape. The modes are defined as the solutions to an optimization problem that combines a quadratic elastic energy term, which penalizes high frequency deformations, with a weighted L_1 -norm term, which enforces sparsity. A novel algorithm to compute such *compressed eigenmodes* of a shape operator is devised, which outperforms other state of the art algorithms for similar problems. The basis is shown to enhance applications such as linearized simulations, elastic modeling, deformation compression and shape segmentation.

Chapter 6 also makes use of the concept of sparsity. There, we try to approximate curves in d -dimensional Euclidean space by spline curves that are defined by the lowest number of control points possible. We consider a general formulation of this problem, allowing for control points at arbitrary parameters of the curve and for various types of splines. We reformulate the problem in a way that finding an optimal spline curve amounts to solving a sparsity-regularized quadratic minimization problem. We apply, adapt and compare several algorithms for the approximation of the problem. The approach can be used to heavily reduce the amount of data required to represent curves, but also other data that smoothly depends on one parameter. As an example we compress animation data acquired from motion capturing. In addition to reducing the data, which enables faster post-processing, the approach also regularizes the data by replacing noisy parts with smooth spline curves.

Each chapter closes with a short conclusion that summarizes the contributions, shows limitations of the proposed methods or offers specific suggestions of future work in the respective area of focus. The concluding Chapter 7 offers a more general outlook on the implications of this work and proposes future directions for model reduction approaches in computer graphics.

1

SPECTRAL PROCESSING OF TANGENTIAL VECTOR FIELDS

This chapter is based on the publication *Spectral Processing of Tangential Vector Fields* by Christopher Brandt, Leonardo Scandolo, Elmar Eisemann and Klaus Hildebrandt, published in Computer Graphics Forum in 2017.

1.1. OVERVIEW

Tangential vector fields are a fundamental representation of directional information on a surface. For example, gradients of functions are tangential vector fields and flows, stresses, strains, or curvature directions are tangential to a surface. Therefore, the processing of tangential vector fields is important for algorithms for many applications in graphics.

Spectral methods are well established for the processing of functions and signals over planar domains. Within the last two decades, spectral methods for the processing of curved surfaces and functions on them have been developed and the field of spectral mesh processing has been established. Central is the eigendecomposition of the Laplace–Beltrami operator, which generalizes the Fourier basis from planar domains to curved surfaces. Whereas on planar domains, spectral methods can be directly generalized to the processing of vector fields by simply applying the method to each of the component functions of the vector field, this is not possible for tangential vector fields on curved surfaces because there is no rigid Cartesian coordinate system.

In this chapter, we introduce a framework for spectral processing of tangential vector fields on curved surfaces, akin to spectral mesh processing. The foundation is a Fourier-type representation that associates frequencies with tangential vector fields. To construct the Fourier-type basis on the space of tangential vector fields, we combine the eigendecomposition of the Hodge–Laplace operator and the Hodge decomposition to obtain basis fields that are either integrable, co-integrable or harmonic.

We formulate the spectral processing framework for piecewise constant tangential

vector fields on triangle meshes. These vector fields are widely used in graphics applications and a discrete Hodge decomposition has been established. To define the spectral basis, we introduce a discrete Hodge–Laplace operator for piecewise constant vector fields on surface meshes. The operator shares important properties with its continuous counterpart and fits conceptually to the prominent *cotan* discretization of the Laplace–Beltrami operator. We show that a sparse matrix representation of the operator can be obtained by combining a set of simple matrices. Additionally, we derive a discrete Dirichlet energy that can be used as a regularizer or fairness energy for piecewise constant tangential vector fields on surfaces. For solving the eigenproblem of the discrete Hodge–Laplace operator, we introduce a scheme that boils the computation of the eigenfields down to the computation of the eigenfunctions of two discrete Laplace–Beltrami operators.

To illustrate the potential of our framework for the applications, we introduce tools for spectral filtering and analysis, compression and real-time tangential vector field design. The filtering tools allow users to design filters in the spectral domain. Individual filters can be specified for the integrable and co-integrable parts of a field. The compression scheme allows for lossy compression of tangential vector fields at high compression rates. For vector field design, we follow the approach introduced by Fisher et al. [1]. A tangential vector field is constructed by minimizing the Dirichlet energy subject to soft constraints that implement the user input.

To ensure interactive editing of tangential vector fields on arbitrarily complex surfaces, we employ a model reduction approach by restricting the design space to the space spanned by 1-2k low-frequency eigenfields of the introduced Hodge–Laplace operator. This significantly accelerates the computation times of smoothest fields, as compared to the approach presented in [1] (up to a factor of 200 in our experiments).

Our second main contribution is a “spline-like”-editor for tangential vector fields. It allows for modeling tangential vector fields using hard constraints on the field and its divergence and curl. Tangential vector field splines (TVFS) can be defined (analogous to cubic splines) as the minimizers of a biharmonic energy under constraints. This idea was already discussed in [Fisher2007] as a possible extension of the design method introduced in the paper. However their approach has two limitations: only soft constraints can be imposed and the resulting scheme is not fast enough to allow for interactive editing. In this work, we overcome both limitations and transfer the approach to piecewise constant vector fields. To be able to formulate TVFS for piecewise constant vector fields, we introduce a biharmonic energy for these fields. Furthermore, we introduce a numerical scheme for computing the TVFS in real-time. The scheme combines the spectral basis and an efficient solver of the quadratic problem.

1.2. RELATED WORK

Spectral mesh processing Within the last two decades a large number of methods for tasks in shape analysis and processing that use the spectrum and eigenfunctions of the Laplace–Beltrami operator have been proposed. In the following, we briefly discuss some of the applications. For an introduction to spectral mesh processing and a broader overview of applications, we refer to the surveys by Lévy and Zhang [2] and Zhang, van Kaick and Dyer [3].

The eigendecomposition of the Laplace–Beltrami operator yields a basis of the space of functions on a curved surface analogous to the Fourier basis for signals over a one-dimensional domain. By transforming a function into this basis, it is decomposed into oscillations that are ordered by their frequencies. Even the surface itself (*i.e.* the embedding of the surface in \mathbb{R}^3) can be treated in this way. Vallet and Lévy [4] proposed a framework for the design of mesh filters. The framework allows for amplifying and suppressing the different frequencies in the embedding of the mesh. For example, high and low pass filters for sharpening and smoothing the mesh can be constructed. Karni and Gotsman [5] proposed a scheme for the compression of triangle meshes based on the eigendecomposition of a combinatorial Laplacian. Recently, a scheme for the compression of dynamic mesh sequences was introduced by Váša et al. [6]. The regular pattern of minima and maxima of the eigenfunctions are used for generating a quadrangular mesh on a surface. Dong et al. [7] introduce a technique that uses the Morse–Smale complex of a carefully chosen eigenfunction to generate a coarse quadrangulation. Huang et al. [8] and Ling et al. [9] extended this approach such that it can provide a user with control of the shape, size, orientation, and feature alignment of the faces of the resulting quadrangulation. Spectral methods for shape segmentation have been introduced by Sharma et al. [10] and Huang et al. [11].

The eigenfunctions enjoy many desirable properties. They are intrinsic, which means they do not change if the mesh is isometrically deformed. Since they are derived as the discretization of a continuous concept, they are mesh independent. They are variational, which makes them robust to remeshing. These properties make them well-suited for the design of pose-independent and mesh-independent shape descriptors. Various descriptors have been designed including the *Shape-DNA* introduced by Reuter et al. [12, 13], Rustamov’s *Global Point Signature (GPS)* [14], the *Heat Kernel Signature (HKS)* proposed by Sun et al. [15], the *Auto Diffusion Function* introduced by Gebal et al. [16] and the *Wave Kernel Signature* by Aubry et al. [17]. A signature involving not only intrinsic, but also extrinsic information about the surface was introduced in [18, 19]. Based on the GPS and the HKS, Ovsjanikov et al. [20, 21] introduced schemes for the detection of shape symmetries. The pose and mesh independence has also been the basis of schemes for shape correspondences and matching. Dey et al. [22] propose a robust pose-oblivious shape matching algorithm based on persistent extrema of the HKS. The functional maps framework, introduced by Ovsjanikov et al. [23], uses low-frequency eigenfunctions of the Laplace–Beltrami operator on two shapes for constructing a linear map between the function spaces of the shapes. Rustamov et al. [24] used this functional correspondence to compare shapes.

Discrete Laplace–Beltrami operators are the backbone of spectral mesh processing. Commonly, piecewise linear and continuous functions (linear Lagrange finite elements) over triangle meshes are considered. The discretization of the Laplace–Beltrami operator in this setting leads to the prominent *cotan* weights [25, 26]. Discretizations with higher-order elements were considered by Reuter et al. [12]. Properties of the discrete Laplace operators have been studied by Wardetzky et al. [27, 28]. Convergence of the discrete operators and their spectrum have been studied by Hildebrandt et al. [29, 30] and Dey et al. [31].

In this chapter, we are proposing a framework that allows for applying spectral meth-

ods for the processing of tangential vector fields on surface meshes. The spectral analysis of vector fields on planar domains has been treated by Wagner, Garth and Hagen [32] and a construction of reduced bases for fluid simulation on planar domains using Laplace eigenfunctions was introduced by de Witt, Lessig and Fiume [33]. These techniques however do not carry over to curved surfaces since they require a fixed Cartesian coordinate system, which is not available for tangential fields on curved surfaces.

Tangential vector field processing Tangential vector fields appear in many applications in graphics. They are used for controlling anisotropic shading of surfaces [34, 35], non-photorealistic rendering [36–38], texture generation [39, 40], simulation of fluid and liquids on surfaces [41, 42], surface segmentation [43, 44] and surface construction [45, 46]. For a recent surveys on direction field synthesis, design, and processing, we refer to [47, 48]. Methods that use tangential vector fields and more general direction fields for surface meshing have received much attention in recent years. Some examples are [49–56]. For a recent survey on the topic, we refer to [57].

The Hodge decomposition of vector fields is an important tool for the processing of tangential vector fields. It allows for decomposing the fields into an integrable, a co-integrable and a harmonic part. A discrete Hodge decomposition of the space of piecewise constant vector fields on a surface mesh has been introduced by Polthier and Preuss [58, 59]. Tong et al. [60] generalized this decomposition to 3-dimensional domains and introduced a multi-resolution representation of vector fields using the potential and co-potential of a vector field. Wardetzky [61] extended the approach from simply-connected domains to surfaces of arbitrary genus and proved convergence of the decomposition. The spectral decomposition we are proposing is compatible with this discrete Hodge decomposition. For a recent survey on discretizations and applications of the Hodge decomposition, we refer to [62].

Fairness energies are used for the reconstruction, design and synthesis of tangential vector and direction fields. Fairness energies for different representations of vector and direction fields have been proposed [51, 63–67]. Here, we introduce a discretization of the Dirichlet energy for piecewise constant tangential vector fields using a combination of conforming and non-conforming discrete divergence and curl operators.

Alternatively to working with vector fields, one can dualize and consider 1-forms. *Discrete Exterior Calculus* [68] provides notions of discrete k -forms and discrete operators on them including a discrete Hodge Laplacian for k -forms. A discrete Hodge decomposition for 1-forms was introduced by Fisher et al. [1] and a spectral decomposition of a discrete Hodge Laplacian on spaces of discrete k -forms has been derived by Arnold et al. [69].

1.3. BACKGROUND: HODGE DECOMPOSITION OF VECTOR FIELDS

In this section, we review the Hodge decomposition of vector fields on a smooth surface \mathcal{M} embedded in \mathbb{R}^3 . We denote the surface normal field by N , and, for any point $p \in \mathcal{M}$ the tangent plane of \mathcal{M} at p by $T_p\mathcal{M}$. Before stating the Hodge decomposition, we introduce basic differential operators on the space of functions and vector fields on surfaces.

Gradient, divergence, curl and Laplace–Beltrami The *gradient* is a linear operator mapping differentiable functions to tangential vector fields. For any point $p \in \mathcal{M}$ the gradient of f at p is defined as the tangential vector $\text{grad } f(p)$ that satisfies

$$\langle \text{grad } f(p), \mathbf{v} \rangle_{\mathbb{R}^3} = d_{\mathbf{v}}f(p)$$

for all $\mathbf{v} \in T_p\mathcal{M}$. Here, $d_{\mathbf{v}}f(p)$ is the derivative of f at p in direction \mathbf{v} . At any point, $\text{grad } f$ points in direction of the steepest ascent of f . We denote by J the operator that rotates any vector of a vector field in its tangent plane by $\frac{\pi}{2}$ (following the orientation of the surface). Besides the gradient operator, we consider the operator $J\text{grad}$, which concatenates the gradient and the rotation J . We call this operator the *co-gradient*.

The *divergence* and *curl* are linear operators that map vector fields to functions. The divergence of a vector field \mathbf{v} at a point $p \in \mathcal{M}$ is defined by

$$\text{div } \mathbf{v}(p) = \sum_{i=1}^2 \langle \nabla_{\mathbf{e}_i} \mathbf{v}(p), \mathbf{e}_i \rangle,$$

where ∇ is the covariant derivative and $\{\mathbf{e}_1, \mathbf{e}_2\}$ forms an orthogonal basis of the tangent plane at p . We define the *curl* as

$$\text{curl } \mathbf{v} = -\text{div } J\mathbf{v}. \quad (1.1)$$

The divergence and curl are related to the gradient and co-gradient. To describe this relation, we use the L^2 -scalar products on the spaces of square-integrable functions and vector fields. These are defined as

$$\langle f, g \rangle_{L^2} = \int_{\mathcal{M}} f(p)g(p)dA \quad (1.2)$$

$$\langle \mathbf{v}, \mathbf{w} \rangle_{L^2} = \int_{\mathcal{M}} \langle \mathbf{v}(p), \mathbf{w}(p) \rangle_{\mathbb{R}^3} dA \quad (1.3)$$

The divergence and gradient as well as the curl and the co-gradient satisfy

$$\langle \text{div } \mathbf{v}, f \rangle_{L^2} = -\langle \mathbf{v}, \text{grad } f \rangle_{L^2} \quad (1.4)$$

$$\langle \text{curl } \mathbf{v}, f \rangle_{L^2} = -\langle \mathbf{v}, J\text{grad } f \rangle_{L^2}, \quad (1.5)$$

for all pairs of a continuously differentiable function f and tangential vector field \mathbf{v} , which follows from an integration by parts. By combining the operators introduced above, we can define the *Laplace–Beltrami* operator

$$\Delta = -\text{div grad} \quad (1.6)$$

on the space of twice differentiable functions. Note that the operator could alternatively be defined as the negative of the curl of the co-gradient: $\Delta = -\text{curl } J\text{grad}$.

Hodge decomposition and harmonic fields The space \mathcal{X} of smooth tangential vector fields on a surface with vanishing boundary can be decomposed into three L^2 -orthogonal subspaces

$$\mathcal{X} = \text{Image}(\text{grad}) \oplus \text{Image}(J\text{grad}) \oplus \mathcal{H},$$

The first subspace is formed by the gradients of smooth functions. Fields in this space are curl-free and represent the integrable part of a vector field. Similarly, the second subspace is the space of co-gradients of smooth functions, which are divergence-free. This space represents the co-integrable part of a vector field. The third space consists of the harmonic tangential vector fields \mathcal{H} . These fields are neither gradients nor co-gradients of functions. The space \mathcal{H} can be defined as the intersection of the kernel of the divergence and the curl. In other words, these are exactly the fields that have vanishing divergence and curl. The space contains information about the topology of the surface. \mathcal{H} equals the first singular cohomology of the surface. This is an important relation between vector calculus and algebraic topology. We refer to the literature, e.g. [70], for more about this connection. One consequence is that the dimension of \mathcal{H} on a surface of genus g is $2g$.

1.4. DISCRETE HODGE DECOMPOSITION

In graphics applications, we are often dealing with piecewise constant vector fields on triangle meshes. A discrete counterpart of the Hodge decomposition for piecewise constant vector fields has been introduced in [58–61]. In this section, we review this construction.

Function spaces We denote the space of vector fields that are constant in every triangle of a mesh \mathcal{M}_h by \mathcal{X}_h . In addition, we consider two function spaces. Both consist of functions on \mathcal{M}_h that are linear polynomials in every triangle. The two spaces are constructed by imposing continuity constraints on the linear polynomials of neighboring triangles: the space S_h of piecewise linear polynomials that are globally continuous and the space S_h^* of piecewise linear polynomials that are continuous at the midpoints of all interior edges. The combination of S_h and S_h^* is needed for the discrete Hodge decomposition.

Discrete operators The gradients of functions in S_h and S_h^* are defined for all points in the interior of a triangle and they are constant within each triangle. Hence the gradient is a linear map from S_h and S_h^* into \mathcal{X}_h . While the gradient can be directly transferred to the discrete setting, we define the discrete divergence and curl operators indirectly using the gradient and equations (1.4) and (1.5). As in the continuous case, the discrete divergence and curl map vector fields to functions. Since we are working with two function spaces, we get two discrete divergence and curl operators. The conforming discrete divergence is the linear operator

$$\operatorname{div}_h : \mathcal{X}_h \mapsto S_h$$

that satisfies

$$\langle \operatorname{div}_h \mathbf{v}, f \rangle_{L^2} = - \langle \mathbf{v}, \operatorname{grad} f \rangle_{L^2}$$

for all $\mathbf{v} \in \mathcal{X}_h$ and $f \in S_h$ and the nonconforming discrete divergence is the linear operator

$$\operatorname{div}_h^* : \mathcal{X}_h \mapsto S_h^*$$

that satisfies

$$\langle \operatorname{div}_h^* \mathbf{v}, g \rangle_{L^2} = -\langle \mathbf{v}, \operatorname{grad} g \rangle_{L^2}$$

for all $\mathbf{v} \in \mathcal{X}_h$ and $g \in S_h^*$. Following the definition (1.1) of the curl in the continuous case, the conforming and nonconforming discrete curl operators are defined as

$$\operatorname{curl}_h \mathbf{v} = -\operatorname{div}_h \mathbf{J} \mathbf{v} \quad \text{and} \quad \operatorname{curl}_h^* \mathbf{v} = -\operatorname{div}_h^* \mathbf{J} \mathbf{v}.$$

We discuss explicit matrix representations of the operators in Section 1.5.

Finally, we want to remark that there is a difference between the discrete operators we define here and the definitions in [59, 61]. They define the divergences and curls of vector fields as integrated quantities, while in the presented definitions, the divergences and curls of vector fields are piecewise linear functions (hence pointwise quantities). The notions are related, one can use the mass matrix (see Section 1.5) to convert one into the other. We refer to [71] for a discussion of integrated vs. pointwise quantities.

Discrete Hodge decomposition As in the continuous case, the discrete Hodge decomposition divides the space of vector fields into three orthogonal subspaces: the image of the gradient, the image of the co-gradient and a space of harmonic vector fields. To obtain spaces of harmonic fields that are $2g$ -dimensional as in the continuous case, we need to combine the spaces S_h and S_h^* . The discrete Hodge decomposition of the space of piecewise constant vector fields is

$$\mathcal{X}_h = \operatorname{Image}(\operatorname{grad}|_{S_h}) \oplus \operatorname{Image}(\mathbf{J} \operatorname{grad}|_{S_h^*}) \oplus \mathcal{H}_h. \quad (1.7)$$

The first component, $\operatorname{Image}(\operatorname{grad}|_{S_h})$, consists of the vector fields that are gradients of functions in S_h . This part describes the integrable part of a vector field, which has vanishing *nonconforming* discrete curl. This means the first component is part of the kernel of curl_h^* . The second component, $\operatorname{Image}(\mathbf{J} \operatorname{grad}|_{S_h^*})$, consists of the vector fields that are co-gradients of functions in S_h^* , the co-integrable part. These vector fields have vanishing *conforming* discrete divergence. Hence, the second component is part of the kernel of div_h . For surfaces with genus zero (homeomorphic to a sphere), $\operatorname{Image}(\operatorname{grad}|_{S_h})$ is exactly the kernel of curl_h^* and $\operatorname{Image}(\mathbf{J} \operatorname{grad}|_{S_h^*})$ is exactly the kernel of div_h . However, for surfaces with non-vanishing genus g there is a $2g$ -dimensional space \mathcal{H}_h of piecewise constant vector fields for which curl_h^* and div_h vanish. These vector fields are neither gradients of functions in S_h nor co-gradients of functions in S_h^* . We call

$$\mathcal{H}_h = \operatorname{Kernel}(\operatorname{div}_h) \cap \operatorname{Kernel}(\operatorname{curl}_h^*)$$

the space of *discrete harmonic* vector fields.

We want to emphasize that this theory requires the interplay of the conforming and nonconforming spaces and operators. For example, a vector field is the gradient of a function in S_h (on a simply-connected domain) if curl_h^* vanishes. Moreover, to get spaces of discrete harmonic vector fields of dimension $2g$, the spaces S_h and S_h^* have to be combined. If only gradients and co-gradients of functions in S_h are used, the dimension of the resulting space of harmonic fields depend on the mesh (and not only on the genus of the surface) and are usually large (*e.g.* the dimension grows under mesh refinement).

As an alternative to (1.7), we could exchange the roles of S_h and S_h^* and obtain the decomposition $\mathcal{X}_h = \text{Image}(\text{grad}|_{S_h^*}) \oplus \text{Image}(J \text{grad}|_{S_h}) \oplus \mathcal{H}_h^*$. Since in this case the integrable part of a vector field is the gradient of a function in S_h^* , we call this the *nonconforming* discrete Hodge decomposition. The resulting space of *nonconforming* discrete harmonic fields is $\mathcal{H}_h^* = \text{Kernel}(\text{div}_h^*) \cap \text{Kernel}(\text{curl}_h)$. The fact that there are two different discrete Hodge decompositions is specific to the discrete setting and is not present in the continuous case. In [61], it was shown that both decompositions converge to their smooth counterpart under suitable refinement of the surface meshes. In this sense, the two decompositions are similar.

In the following, we will consider only the decomposition (1.7). However, for any notion we introduce, there is a corresponding notion where the roles of the spaces S_h and S_h^* are exchanged.

1.5. DISCRETE HODGE–LAPLACE OPERATOR FOR VECTOR FIELDS

In this section, we introduce a discrete Hodge–Laplace operator for piecewise constant vector fields on surface meshes and the corresponding discrete Dirichlet and biharmonic energies. Before we consider the discrete setting, we first review the smooth setting.

The smooth setting By combining the operators discussed in Section 1.3, one can construct the *Hodge–Laplace* operator

$$\Delta = -(\text{grad div} + J \text{grad curl}) \quad (1.8)$$

on the space of smooth tangential vector fields on a surface. We use the minus sign in (1.8) to get positive eigenvalues for the Hodge–Laplace operator.

Since we could not find a reference where (1.8) is stated, we want to put this formula in context with the literature. The Hodge Laplacian is usually defined as an operator on smooth k -forms on a Riemannian manifold [72]. For surfaces, 0- and 2-forms can be identified with functions and the Hodge Laplacian for these forms is the Laplace–Beltrami operator (1.6) for functions. 1-forms on a surface can be identified with vector fields via $\mathbf{v} \leftrightarrow \langle \mathbf{v}, \cdot \rangle$. Using this isomorphism, we can carry over the Hodge Laplacian from 1-forms to vector fields and express this operator in terms of the operators J , grad , div and curl . Formula (1.8) is the resulting operator. For vector fields on planar domains, this operator agrees with the vector Laplacian. In this sense, the Hodge Laplacian generalizes the vector Laplacian from vector fields on planar domain to tangential vector fields on curved surfaces.

Discrete Hodge–Laplace operators Using the definition (1.8) of the smooth Hodge–Laplace operator for tangential vector fields on a surface and the discrete operators introduced in Section 1.4, we can construct the discrete Hodge–Laplace operator for piecewise constant vector fields on a surface mesh. As for the discrete Hodge decomposition, the discretization mixes the conforming discrete divergence and the non-conforming discrete curl operators

$$\Delta_h = -(\text{grad div}_h + J \text{grad curl}_h^*).$$

The discrete Hodge–Laplace operator Δ_h shares many properties with its continuous counterpart Δ .

1. Symmetry The operator Δ_h is self-adjoint with respect to the L^2 -scalar product, *i.e.*

$$\langle \Delta_h \mathbf{v}, \mathbf{w} \rangle_{L^2} = \langle \mathbf{v}, \Delta_h \mathbf{w} \rangle_{L^2}$$

for any pair $\mathbf{v}, \mathbf{w} \in \mathcal{X}_h$.

2. Harmonic fields The discrete harmonic vector fields (which we defined in Section 1.4) are exactly the piecewise constant fields with vanishing discrete Hodge–Laplace operator, *i.e.*

$$\mathcal{H}_h = \text{Kernel}(\Delta_h).$$

3. Positive semi-definite The operator Δ_h is positive semi-definite. The symmetry guarantees that all eigenvalues are real. The harmonic fields have eigenvalue zero, all other eigenvalues are positive.

4. Locality The continuous Hodge–Laplace operator is local, *i.e.* evaluating the smooth Hodge–Laplace of a vector field at a point p does not depend on the surface or the vector field outside of an arbitrarily small neighborhood of p . We achieve this property by using a diagonal mass matrix for the L^2 -scalar product on S_h . Then, for any $\mathbf{v} \in \mathcal{X}_h$ and triangle $t \in \mathcal{M}_h$, the vector that $\Delta_h(\mathbf{v})$ assumes in t depends only on the vectors of \mathbf{v} in the triangles that share a common vertex with t and the geometry of these triangles.

5. Intrinsic The discrete Hodge–Laplace operator is an intrinsic operator, *i.e.*, it can be constructed using only the length of all edges of the mesh. As a consequence it does not change if the surface is isometrically deformed.

6. Hodge decomposition The discrete Hodge–Laplace operator respects the corresponding discrete Hodge decomposition. The image of an integrable vector field is an integrable vector field and the image of a co-integrable vector field is a co-integrable vector field.

Discrete energies Based on the discrete Hodge–Laplace operator, we introduce two quadratic functionals (or energies) on the space of piecewise constant tangential vector fields: the Dirichlet energy

$$E_{\mathbf{D}}(\mathbf{v}) = \langle \Delta_h \mathbf{v}, \mathbf{v} \rangle_{L^2} = \int_{\mathcal{M}_h} ((\text{div}_h \mathbf{v})^2 + (\text{curl}_h^* \mathbf{v})^2) dA \quad (1.9)$$

and the biharmonic energy

$$E_{\mathbf{B}}(\mathbf{v}) = \langle \Delta_h \mathbf{v}, \Delta_h \mathbf{v} \rangle_{L^2} = \int_{\mathcal{M}_h} \|\Delta_h \mathbf{v}\|^2 dA. \quad (1.10)$$

For simply-connected surfaces (topological spheres) the energies are positive definite, and, for surfaces of genus $g > 0$, the energies are semi-positive definite. In the latter case, the harmonic fields are in the kernel of the energies. We will use these energies as regularizers for the construction of smooth vector fields.

Matrix representations Matrix representations of all the discrete operators and the energies can be obtained as products and sums of a set of six simple matrices. This illustrates the structure underlying the operators and simplifies the implementation. To get a matrix representation of an operator, we first have to fix bases in the relevant spaces. We use the nodal bases, *i.e.* a function in S_h is represented by a vector listing a function value for every vertex and a function in S_h^* is represented by a vector listing a function value for every edge. The linear polynomials over each triangle corresponding to a nodal vector are uniquely determined via interpolation. For S_h^* , the nodes are located at the midpoints of the edges. For a vector field in \mathcal{X}_h , we are listing one tangential vector for every triangle. To describe the vectors, we fix a (positively oriented) orthonormal basis of the tangent plane in every triangle.

Once the bases are fixed, we can construct the matrices. The first two matrices are the gradients G and G^* on S_h and S_h^* . Both matrices are sparse (three entries per row). Explicit formulas for the computation of the gradient of a linear polynomial over a triangle can be found in [25] and [73, pp. 40–41]. Furthermore, we need three diagonal mass matrices M , M^* , and \mathbf{M} representing the L^2 -scalar products in S_h , S_h^* and \mathcal{X}_h . The i^{th} diagonal entry of M is a third of the combined area of the triangles adjacent to the i^{th} vertex and the j^{th} diagonal entry of M^* is a third of the combined area of the two triangles sharing the j^{th} edge. Alternatively, the Voronoi areas of the vertices and edge midpoints can be used. The diagonal entries of the matrix \mathbf{M} are simply the areas of the corresponding triangles. We refer to [71] for a discussion of mass matrices. The last matrix \mathbf{J} is the matrix representation of the operator \mathbf{J} that rotates every vector of a piecewise constant vector field by $\pi/2$ in the tangent plane. The matrix is block diagonal, each block consists of a 2×2 matrix that represents the $\pi/2$ -rotation in one triangle with respect to the chosen orthonormal basis. All the 2×2 matrices are the same since for any positively oriented orthogonal basis, a $\pi/2$ -rotation maps the first entry of the vector to the second and the second to the negative of the first entry.

Matrix representations of all the discussed discrete operators can be obtained as products and sums of these six matrices. For the discrete divergence and curl operators, we have the following matrix representations. We denote the matrices representing the

div_h	curl_h	div_h^*	curl_h^*
$-M^{-1}G^T\mathbf{M}$	$M^{-1}G^T\mathbf{J}\mathbf{M}$	$-M^{*-1}G^{*T}\mathbf{M}$	$M^{*-1}G^{*T}\mathbf{J}\mathbf{M}$

discrete Dirichlet energy, Hodge–Laplace operator and biharmonic energy by \mathbf{S} , \mathbf{L} and \mathbf{B} . The matrices satisfy

$$\mathbf{S} = \mathbf{M}(GM^{-1}G^T - \mathbf{J}G^*M^{*-1}G^{*T}\mathbf{J})\mathbf{M}$$

$$\mathbf{L} = \mathbf{M}^{-1}\mathbf{S}$$

$$\mathbf{B} = \mathbf{L}^T\mathbf{M}\mathbf{L} = \mathbf{S}\mathbf{M}^{-1}\mathbf{S}$$

To derive the first row, we combine the matrix representations of the operators div_h and curl_h^* and form the energy (1.9). The second row follows by construction because the discrete Hodge–Laplace operator is the self-adjoint operator corresponding to the discrete

Dirichlet energy. For the first step in the third row, we use (1.10), and the second step follows from the formula for \mathbf{L} .

We want to remark that the discrete Dirichlet energy for piecewise linear functions and the discrete Laplace–Beltrami operator can also be constructed from these matrices. The matrix S representing the Dirichlet energy of functions is given by $S = G^T \mathbf{M} G$. This is exactly the *cotan* matrix [25, 26]. The matrix L representing the discrete Laplace–Beltrami operator is given (analogous to the second row above) by $L = M^{-1} S$.

1.6. FOURIER REPRESENTATION OF TANGENTIAL VECTOR FIELDS

In this section, we describe a Fourier type representation of tangential vector fields that associates frequencies with the fields in the space \mathcal{X}_h . The representation is based on the eigenfields of the Hodge–Laplace operator, which have to be computed numerically. We introduce a scheme for computing the eigenfields of the Hodge Laplacian that reduces the problem to the computation of the eigenfunctions of the conforming and the nonconforming discrete Laplace–Beltrami operators.

Eigenfields of the discrete Hodge–Laplace operator The eigenfields of Δ_h are the solutions of the equation

$$\Delta_h \Phi = \lambda \Phi.$$

Instead of solving this eigenproblem directly, we construct the eigenfields using the eigenfunctions of the conforming and nonconforming discrete Laplace–Beltrami operators

$$\Delta_h = -\operatorname{div}_h \operatorname{grad} \quad \text{and} \quad \Delta_h^* = -\operatorname{div}_h^* \operatorname{grad}.$$

These operators are the *cotan*-Laplacians on S_h and S_h^* . Eigenfunctions of these operators are solutions of the problems

$$\Delta_h \phi = \lambda \phi \quad \text{and} \quad \Delta_h^* \psi = \mu \psi,$$

where $\phi \in S_h$ and $\psi \in S_h^*$. The numerical treatment of the eigenproblem for the conforming operator is the basis for spectral geometry processing and is treated in detail in [4]. The nonconforming case can be treated in the same way, only the matrices M^* and $S^* = G^{*T} \mathbf{M} G^*$ are used instead of M and $S = G^T \mathbf{M} G$.

The following Lemma summarizes the relation of the eigenfunctions of Δ_h and Δ_h^* and the eigenfields of Δ_h and is the basis of our scheme for computing the eigenfields.

Lemma 1 *The gradient of any eigenfunction of Δ_h is an eigenfield of Δ_h and the co-gradient of any eigenfunction of Δ_h^* is an eigenfield of Δ_h . The eigenvalues of an eigenfunction and the corresponding eigenfield are the same.*

Proof. Let ϕ be an eigenfunction of Δ_h with eigenvalue λ . Then,

$$\begin{aligned} \Delta_h \operatorname{grad} \phi &= -\operatorname{grad} \operatorname{div}_h \operatorname{grad} \phi - \mathbf{J} \operatorname{grad} \operatorname{curl}_h^* \operatorname{grad} \phi \\ &= \operatorname{grad} \Delta_h \phi = \lambda \operatorname{grad} \phi. \end{aligned}$$

This proves the lemma for eigenfunctions of Δ_h . The statement about the co-gradients of eigenfunctions of Δ_h^* is proved in a similar manner. \square



Figure 1.1: The first, 21st, 51st and 71st pairs of eigenfields on the centaur model. Green fields are curl-free, red fields are divergence-free.

Fourier representation We denote the number of vertices, edges and the genus of our mesh by n_v , n_e , and g . The following lemma shows that we can use Lemma 1 to construct an orthonormal basis of \mathcal{X}_h .

Lemma 2 *Let $\{\phi_0, \phi_1, \dots, \phi_{n_v-1}\}$ and $\{\psi_0, \psi_1, \dots, \psi_{n_e-1}\}$ be eigenbases of Δ_h and Δ_h^* (where ϕ_0 and ψ_0 are the constant functions), and let $\{\Gamma_1, \Gamma_2, \dots, \Gamma_{2g}\}$ be an orthonormal basis of the subspace of harmonic fields \mathcal{H}_h . Then, the set $B = \{\Phi_1, \Phi_2, \dots, \Phi_{n_v-1}, \Psi_1, \Psi_2, \dots, \Psi_{n_e-1}, \Gamma_1, \Gamma_2, \dots, \Gamma_{2g}\}$, where $\Phi_i = \frac{1}{\|\text{grad}\phi_i\|_{L^2}} \text{grad}\phi_i$ and $\Psi_i = \frac{1}{\|\text{grad}\psi_i\|_{L^2}} J \text{grad}\psi_i$, is an orthonormal basis of the space \mathcal{X}_h of piecewise constant tangential vector fields.*

Proof. We first show that any pair of vector fields from B is orthogonal. For $i \neq j$, we have

$$\langle \text{grad}\phi_i, \text{grad}\phi_j \rangle_{L^2} = \langle \Delta_h \phi_i, \phi_j \rangle_{L^2} = \lambda_i \langle \phi_i, \phi_j \rangle_{L^2} = 0,$$

which implies

$$\langle \Phi_i, \Phi_j \rangle_{L^2} = 0.$$

In a similar manner, we can show that any pair Ψ_i, Ψ_j is orthonormal. The discrete Hodge decomposition (1.7) guarantees that any pair of vector fields with different letters is orthogonal, because such fields are in different components of the Hodge decomposition. It remains to show that the number of vector fields in the set equals the dimension of the space \mathcal{X}_h . The set B consists of $|B| = n_v - 1 + n_e - 1 + g$ vector fields. Using the Euler formula $n_v - n_e + n_f = 2 - 2g$, we get $|B| = 2n_e - n_f$. Since our mesh is a closed manifold, every edge is in two triangles, which means $3n_f = 2n_e$. Using this equation, we get $|B| = 2n_f$, which is exactly the dimension of \mathcal{X}_h . We showed that B is an orthonormal set in \mathcal{X}_h with $2n_f$ elements, which proves that B is an orthonormal basis of \mathcal{X}_h . \square

As consequence of the lemma, we can represent any field $\mathbf{v} \in \mathcal{X}_h$ in the basis B

$$\mathbf{v} = \sum_{i=1}^{n_v-1} \alpha_i \Phi_i + \sum_{i=1}^{n_e-1} \beta_i \Psi_i + \sum_{i=1}^{2g} \gamma_i \Gamma_i, \quad (1.11)$$

where $\alpha_i = \langle \mathbf{v}, \Phi_i \rangle_{L^2}$, $\beta_i = \langle \mathbf{v}, \Psi_i \rangle_{L^2}$ and $\gamma_i = \langle \mathbf{v}, \Gamma_i \rangle_{L^2}$. Since any basis field is an eigenfield of the discrete Hodge–Laplace operator, we can associate a frequency, the square root of the eigenvalue, to every basis field. Hence, this representation associates frequencies with tangential vector fields. In this sense, (1.11) is a Fourier representation. In the following sections, we will show benefits of this representation for the applications.

In the continuous case, the eigenfunctions come in pairs. For every eigenfield Φ , the rotated field, $J\Phi$, is also an eigenfield with the same eigenvalue. Since we construct the basis as gradients and co-gradients of eigenfunction of the Laplace–Beltrami operator, we choose in every eigenspace the basis such that every basis vector is in one component of the Hodge decomposition. As a result, the Fourier representation refines the Hodge decomposition. Since in the discrete case two different function space, S_h and S_h^* , have to be combined, the symmetry is broken and “pairs” of eigenfields have only approximately the same eigenvalues. Figure 1.1 shows “pairs” of eigenfields of the discrete operator.

Computation of the eigenfields Our scheme for computing the low-frequency eigenfields proceeds in three steps. The input is a maximum eigenvalue λ_{\max} (or alternatively the number of eigenfields to be computed). The first step is to compute a basis of the space of discrete harmonic fields (which along the way provides us with a basis of the cohomology of the surface). The space is $2g$ -dimensional, where g is the genus of the surface. The idea is to project $2g$ random vector fields to the space of harmonic fields. These will span the space of harmonic fields (the probability that the random vectors or their projections are linearly dependent vanishes). Finally, we orthonormalize these vectors. To project a vector field \mathbf{v} into the space of harmonic fields, we remove the integrable and the co-integrable parts. The potentials of the integrable and the co-integrable part can be determined by solving the least squares problem

$$\operatorname{argmin}_{f \in S_h, g \in S_h^*} \|\mathbf{v} - \operatorname{grad} f - J \operatorname{grad} g\|^2.$$

Since the integrable and co-integrable parts are orthogonal, solving this problem can be carried out in two steps: first compute the integrable part, then the co-integrable part. Both steps require solving a linear system where the matrices are the cotan matrices $S = G^T \mathbf{M} G$ and $S^* = G^{*T} \mathbf{M} G^*$. The second step is to compute linearly independent eigenfunctions ϕ of Δ_h with eigenvalue smaller than λ_{\max} . Since all eigenvalues are positive, we compute bands of eigenfunctions with increasing eigenvalue starting with zero until we reach λ_{\max} . Then we compute the corresponding eigenfields $\Phi = \operatorname{grad} \phi$ and orthonormalize them. The third step is to compute the eigenfunctions ψ of Δ_h^* with eigenvalue smaller than λ_{\max} . As before, we compute the corresponding eigenfields $\Psi = J \operatorname{grad} \psi$ and orthonormalize them. For the computation of the eigenfunctions of Δ_h and Δ_h^* , we follow [4].

The resulting eigenbasis refines the discrete Hodge decomposition. By construction each type of eigenfield belongs to one subspace of \mathcal{X}_h . The fields Φ are in the integrable component, the fields Ψ in the co-integrable component and the rest forms a basis of the space of harmonic fields.

1.7. TANGENTIAL VECTOR FIELD SPLINES

Many classical splines can be characterized by variational principles. Splines in tension are minimizers of a weighted sum of the biharmonic and the Dirichlet energy subject to constraints (cubic splines are the special case when the weight of the Dirichlet energy vanishes). Following the classical example, we define the tangential vector field splines (TVFS) as minimizers of the weighted sum of the biharmonic and the Dirichlet energy

$$E_{\mathbf{B}}(\mathbf{v}) + \omega E_{\mathbf{D}}(\mathbf{v}) \quad (1.12)$$

subject to linear equality constraints on the vectors of \mathbf{v} and its divergence and curl. Vortices can be constructed by prescribing non-zero curl, sinks and sources are created via positive or negative divergence respectively. Alternatively, singularities can be constructed by specifying a few vector constraints around their desired locations. We refer to the several images to see several examples of this type of topology control in action. The biharmonic energy is needed to obtain smooth enough vector fields that satisfy the hard constraints. The effect is shown in Figure 1.2.

We want to remark that the idea of defining tangential vector field splines as minimizers of (1.12) was introduced in [1] as an extension of their vector field design approach. However, their approach has two limitations: only soft constraints can be imposed and the resulting scheme is not fast enough to allow for interactive editing.

Model reduction The computation of a TVFS amounts to solving a sparse quadratic problem with linear equality constraints. The challenge is to solve the problems at real-time rates to enable an interactive TVFS editor.

Directly solving the resulting linear systems to compute a TVFS is not an option since this can take several minutes. Directly re-using a sparse factorization is not possible, because the size of the matrix changes, whenever constraints are added or removed.

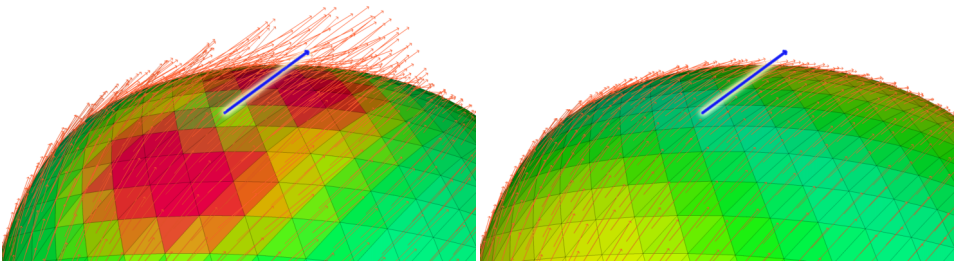


Figure 1.2: The per-face Dirichlet energy of a tangential vector field spline from a single constraint on an irregular sphere is shown (red: high Dirichlet energy, green: low Dirichlet energy). On the left, we show the minimizer of the Dirichlet energy only, without higher order regularizer, on the right, we use the biharmonic energy (1.12) with low ω .

Therefore, we employ a model reduction approach, by restricting the search space to a subset of the basis of eigenfields. This reduces the size of the system matrix but additionally enhances the speed of the linear solve required to find the smoothest field. We are using the property of the eigenbasis that in the basis of eigenfields, the Dirichlet and the biharmonic energy are represented by diagonal matrices.

In a preprocess, we compute the d eigenfields with the smallest eigenvalue as described in Section 1.6. We assemble the vectors to form the columns of a matrix $\mathbf{U} \in \mathbb{R}^{2n_f \times d}$. A tangent field in the d -dimensional subspace can be described by reduced coordinates, *i.e.* a vector $v \in \mathbb{R}^d$. The matrix \mathbf{U} transforms from reduced coordinates v to the full coordinates $x = \mathbf{U}v \in \mathbb{R}^{2n_f}$. Since we are using a basis of eigenfields, in the reduced coordinates the energies are represented by diagonal matrices. The matrix $\mathbf{\Lambda}$ representing the Dirichlet energy has the eigenvalues on the diagonal. Then, $\mathbf{\Lambda}^2$ represents the biharmonic energy and the resulting energy matrix \mathbf{D} is

$$\mathbf{D} = \mathbf{\Lambda}^2 + \omega \mathbf{\Lambda}.$$

We consider n_c linear constraints, which in the unreduced coordinates have the form $\tilde{\mathbf{C}}x = c$, where $\tilde{\mathbf{C}} \in \mathbb{R}^{n_c \times 2n_f}$, $c \in \mathbb{R}^{n_c}$ and $x \in \mathbb{R}^{2n_f}$. To constrain the divergence or curl of the field at a vertex or an edge, we copy the corresponding row from the divergence and curl matrices (see Section 1.5) into $\tilde{\mathbf{C}}$. The entry in the vector c specifies the value the divergence or curl assumes. Values at arbitrary locations in a triangle can be specified using barycentric coordinates. In a similar manner, vectors in triangles can be prescribed. Once the matrix $\tilde{\mathbf{C}}$ is constructed, we can obtain its reduced counterpart by matrix multiplication: $\mathbf{C} = \tilde{\mathbf{C}}\mathbf{U}$. The matrix \mathbf{C} ensures that the reduced solution exactly satisfies the constraints. The matrix is of small size, $\mathbf{C} \in \mathbb{R}^{n_c \times d}$.

Now we describe how to efficiently solve the constrained linear system. We first consider the case of a surface of genus 0. In this case, the matrix \mathbf{D} has full rank and since it is diagonal, it can be easily inverted. Using Lagrange multipliers, represented by a vector $\mu \in \mathbb{R}^{n_c}$, the solution v of the constraint optimization problem is computed by solving the linear system

$$\begin{aligned} \mathbf{D}v - \mathbf{C}^T \mu &= 0 \\ \mathbf{C}v &= c \end{aligned}$$

Instead of solving this system directly, we first transform it. To eliminate v from the second equation, we multiply the first equation by $\mathbf{C}\mathbf{D}^{-1}$ and subtract it from the second equation

$$\mathbf{D}v - \mathbf{C}^T \mu = 0 \tag{1.13}$$

$$\mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T \mu = c \tag{1.14}$$

To compute the solution v , we first solve (1.14) for μ and then (1.13) for v . To compute μ , we factor the matrix $\mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T$, which is a very small matrix of size $n_c \times n_c$. A new factorization is computed whenever the set of constraints changes, changing the value of the constraints affects only the right-hand side of the equation. Solving for v is very fast since \mathbf{D} is diagonal.

In the case, of a surface of genus $g > 0$, the matrix \mathbf{D} has the harmonic fields in its kernel. The resulting system can be solved by treating the harmonic part separately. The system can be re-arranged such that first the harmonic part and the Lagrange multipliers are determined and then v is computed. An alternative is to slightly modify the system by setting the eigenvalues of the harmonic part to a small positive constant (e.g. a tenth of the lowest non-zero eigenvalue).

Since we are restricting the computation to a subspace spanned by low-frequency fields, the algorithm computes a low-pass filtered TVFS. From a signal theoretic point of view, the TVFS are low-frequency fields by construction. Our computational scheme cuts-off the remaining high-frequencies of the field. In this sense, the reduced solution could even be the preferred solution. Using bases of 1-2k eigenfields, the reduced results are typically very close to the TVFS.

1.8. APPLICATIONS AND EXPERIMENTS

1.8.1. COMPUTATION OF THE EIGENFIELDS

Table 1.1 lists timings for the computation of the eigenfields of the Hodge Laplacian. The column *Bases setup* contains the total time to setup the conforming and nonconforming Laplace–Beltrami operator, computing their eigenfunctions and then computing their gradients and co-gradients. To compute the low-frequency eigenfunctions, we use the shift-and-invert Lanczos method. The implementation was done in Java with native calls to the MUMPS library [74] for solving the sparse linear systems. The computation was performed on a Dell Precision M3800.

Figure 1.1 shows examples of low-frequency eigenfields. For each integrable field (green), the corresponding co-integrable field (red) is shown.

1.8.2. SPLINE EDITOR FOR TANGENTIAL FIELDS

In Section 1.7, we described a system for modeling tangential vector field splines in real-time, making use of our reduced basis. We implemented this system, using a dense Cholesky factorization and utilizing the GPU to quickly map from the reduced space to a full field representation. In total, we get real-time responses in an interactive editing environment, where vector constraints can be specified via click-and-drag and a globally optimal tangential field is instantly updated. The method scales well with the sizes of

Model Name	#faces	Bases setup	Spline Editing Setup \Solve	Reduced Soft Design Factor \Solve	Full Soft Design Factor \Solve
Hand	12184	23s	12ms \ 1ms	148ms \ 2.5ms	1.4s \ 16ms
Rocker Arm	20088	43s	15ms \ 1ms	148ms \ 2.6ms	3.3s \ 28ms
Bunny	69666	184s	18ms \ 2ms	155ms \ 2.4ms	22.7s \ 141ms
Bumpy Torus	140240	395s	17ms \ 2ms	169ms \ 3.1ms	86.8s \ 379ms
Armadillo	331904	1246s	49ms \ 4ms	150ms \ 3.5ms	187.7s \ 818ms

Table 1.1: Timings for tangential vector field bases computation, solving the reduced tangential vector field spline system and solving the reduced and full design system. In all three systems we used a second order energy term as an additional regularizer. In all examples a basis size of 1000 was used (500 divergence free and 500 curl free eigenfields of the Hodge–Laplace operator) and 30 constraints were specified.

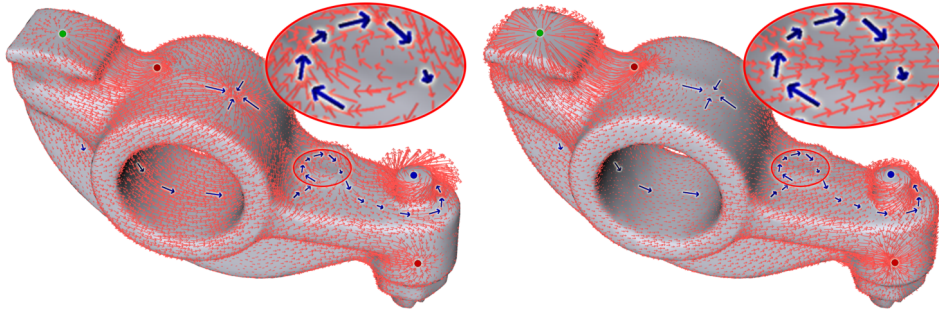


Figure 1.3: Comparison between editing using our tangential vector field spline editor as described in Section 1.7 (left) and using soft constrained vector field design (cf. [1]) (right). As can be seen, not all constraints on the right are satisfactorily obeyed.

the meshes and allows for real-time tangential vector field modeling on larger meshes: aside from mapping the reduced coordinates to the full representation (which can be done efficiently on the GPU), the size of the system to be solved is independent of the resolution of the mesh.

In Table 1.1, 4th column, we list the resulting total time for generating tangential vector field splines on various meshes from 30 user defined constraints using a basis of 1000 eigenfields. We separately list the timings for setting-up and factorizing the system and the timing for a solve (including the time for sending the reduced coordinates to the GPU and there mapping them to the full representation). When a new constraint is added, both those steps have to be executed, but if a constraint is just modified (*e.g.* changing the direction of a vector constraint), only the solve time has to be taken into account, which is below 5ms even for our largest test mesh (the Armadillo mesh).

In Figure 1.3, left, we show a result of designing a vector field using our spline editor. As can be seen, all constraints are exactly obeyed but the overall field is still very smooth. Another result of TVFS editing on a high-resolution meshes can be seen in Figures 1.4 and 1.5, where we effortlessly designed tangent vector fields on meshes with 331k and 70k faces respectively.

1.8.3. FUR DESIGN

As an application of tangential vector field spline editing, we introduce a tool for fur design on surface meshes. In Figures 1.4 and 1.5 snapshots of an interactive editing session using this tool can be seen. The efficient and intuitive way to design smooth tangent fields via few hard constraints allows a designer to edit fur on surface meshes in real-time by specifying the length and direction of the hair at certain spots, while aiming for an overall smoothly varying hair direction.

For this type of design task real-time visual responses are crucial, which is made possible with our reduction via the spectral basis. More generally does the reduced basis allow for efficient updates to the GPU when the tangential field changes, which can speed-up the visualization of CPU-run simulations of tangential vector fields.

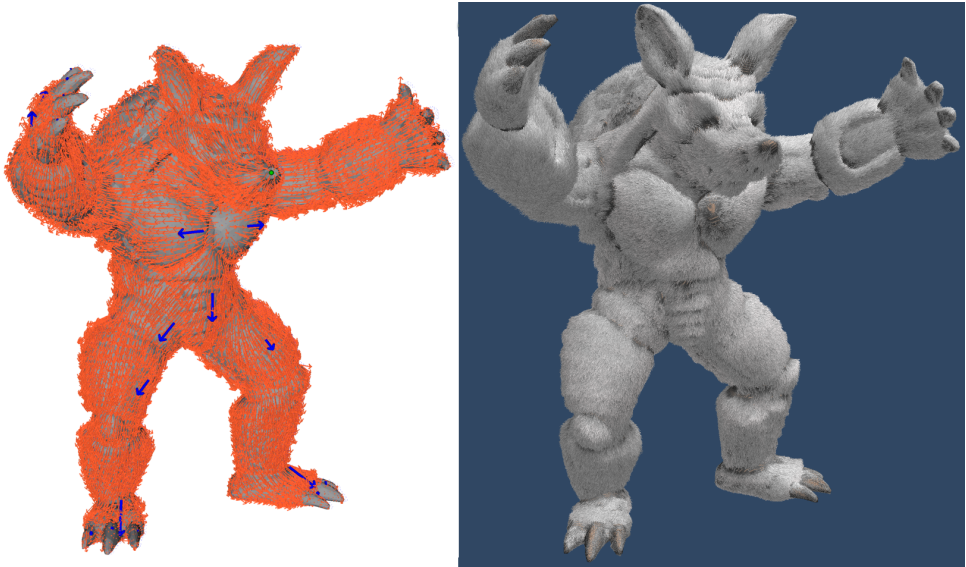


Figure 1.4: Real-time tangential vector field spline editing on the armadillo model (left, 331904 faces) and the resulting fur (right).

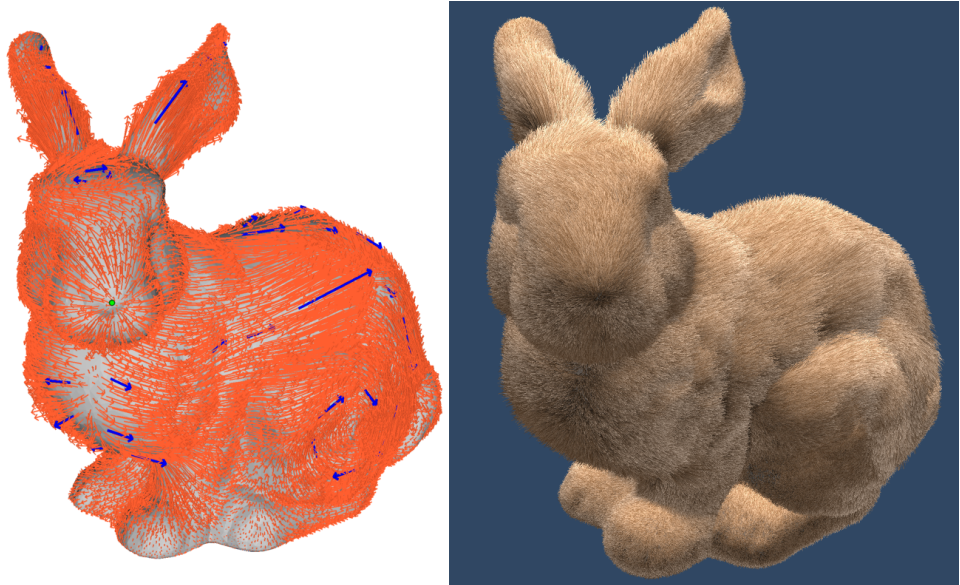


Figure 1.5: Fur design on the bunny mesh. Left: constraints and resulting tangential vector field spline, right: output field visualized as fur on the bunny.

1.8.4. SPEEDING-UP SOFT CONSTRAINED DESIGN

In Fisher et al. [1], a method for designing vector fields using weak constraints has been proposed. Sinks, sources and vortices can be constructed via prescribing non-zero curl and divergence at specific vertices/edges, but vanishing curl and divergence on the rest of the mesh. Additionally, interpolation constraints for the vector field can be imposed. Since this, in general, over-constrains the system, they are treated as weak constraints and a least-squares problem is solved. As a regularizer, either the Dirichlet energy or a weighted sum of the biharmonic and the Dirichlet energy are used. Denoting the linear constraints by $\tilde{C}x = c$, as in Section 1.7, and the weight of the biharmonic energy by ω , the linear system

$$(S + \omega B + \tilde{C}^T \tilde{C})x = \tilde{C}^T c$$

has to be solved to construct a field. To solve the system, a sparse Cholesky factorization is computed once and used to solve the systems. When the set of constraints changes, a sparse Cholesky update is computed.

By restricting the construction to a subspace of $d = 1 - 2k$ low-frequency modes, we get a fast approximation algorithm for this system. In particular, the computational cost for the reduced computation depends on the dimension of the subspace and is independent of the resolution of the mesh. The only operations that depend on the mesh size are the mapping of the reduced coordinates to the unreduced coordinates and the construction of the subspace, which is done in a preprocess. Using the notation of Section 1.7, the reduced system is

$$(\Lambda + \omega \Lambda^2 + C^T C)v = C^T c,$$

which is a d -dimensional system. To solve the system, we compute a dense Cholesky decomposition and use dense Cholesky updates, when the set of constraints changes.

After employing the spectral basis to the resulting system, we are able to speed-up the computation times of the designed tangential fields by a factor of up to 200 (for higher resolution meshes this factor will become larger). Figure 1.6 shows an example of a reduced and an unreduced solution to the system. One can see that the reduced solution is a smoother field (since additional high frequencies are cut-off). Note that Fisher et al. propose their method in the setting of discrete 1-forms. However, to make a better comparison, we re-implemented their system to work with piecewise constant tangential fields. We solve the resulting sparse linear systems using the MUMPS library, which provides sparse Cholesky factorizations. The speed-up can be observed from the timings listed in Table 1.1, where one can see, that already for small meshes we gain significantly shorter computation times, and that for large meshes, interactive design is made possible at all only by using our reduced basis. We list timings for factorizations and solving the systems separately. We create tangential fields from 30 user defined constraints and, in the reduced case, we use a basis of 1000 eigenfields. The biharmonic energy was added as a regularizer. Note that adding constraints does not require a re-factorization in either the full or the reduced case, since the factorization can be updated using sparse or dense Cholesky updates, which take less than a millisecond.

To highlight the difference of our tangential vector field spline editing and the vector field design system discussed in this section, we point to Figure 1.3. Here one can see how not all constraints in the least squares system are satisfactorily obeyed. The reason for this is that the user needs to specify the weights for the prescribed vectors and

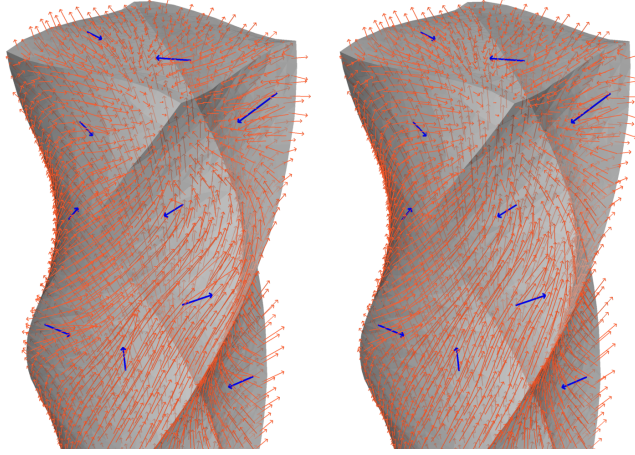


Figure 1.6: Comparison of unreduced (left) and reduced (right) vector field design.

the magnitudes of the prescribed curl and divergence, which results in a trade-off between satisfying the locations of prescribed sinks, sources and vortices versus obeying the prescribed vectors. In the shown example, no weights were found where both types of constraints were obeyed satisfactorily at the same time.

1.8.5. SPECTRAL ANALYSIS AND FILTERING

The Fourier representation (1.11) discussed in Section 1.6 allows for spectral analysis and filtering of tangential vector fields. In practice, we only compute the harmonic and first k integrable and co-integrable eigenfields, where k is between 500 and 5000. Then any piecewise constant vector field \mathbf{v} can be written as

$$\mathbf{v} = \sum_{i=1}^k (\alpha_i \Phi_i + \beta_i \Psi_i) + \sum_{i=1}^{2g} \gamma_i \Gamma_i + \mathbf{v}^r \quad (1.15)$$

where \mathbf{v}^r is the “rest-field” of \mathbf{v} , *i.e.* $\mathbf{v}^r = \sum_{i=k+1}^n \alpha_i \Phi_i + \sum_{i=k+1}^m \beta_i \Psi_i^c$. The coefficients α_i, β_i describe the contribution of \mathbf{v} to the corresponding eigenfields. They are ordered by ascending frequency and can be analyzed and manipulated for spectral processing of the field. Note that the harmonic eigenfields are in the kernel of Δ_h and thus contribute to the lowest frequency part of the field.

To enable the spectral processing of tangential vector fields via the creation of spectral filters, we use the established method (cf. [4]) of enabling the user to “draw” functions $F_\alpha, F_\beta : \mathbb{R}^+ \rightarrow [0, \tau]$, such that a new vector field \mathbf{v}^* is acquired by replacing the spectral coefficients of \mathbf{v} by new coefficients $\alpha_i^* = F_\alpha(\sqrt{\lambda_i}) \cdot \alpha_i$ and $\beta_i^* = F_\beta(\sqrt{\lambda_i}) \cdot \beta_i$. Here, λ_i is the i^{th} distinct eigenvalue of the Hodge Laplacian and τ signifies the largest possible magnification of a coefficient. The reason for scaling the coefficients by $F_{\alpha/\beta}(\sqrt{\lambda_i})$ is that the frequency of the i^{th} eigenfunction is related to the square root of λ_i .

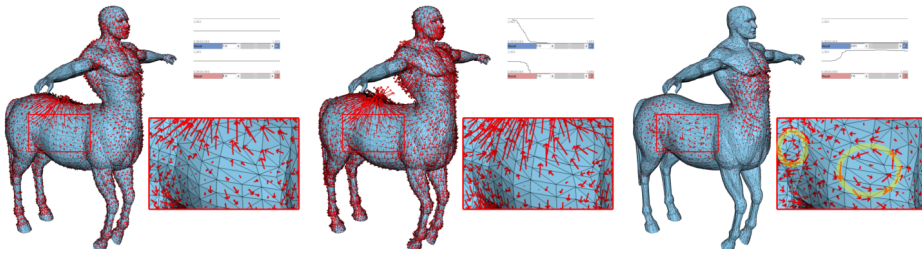


Figure 1.7: Applying spectral filters to a custom vector field (left). The integrable and co-integrable parts of the field are filtered individually. Two results (middle and right) are shown.

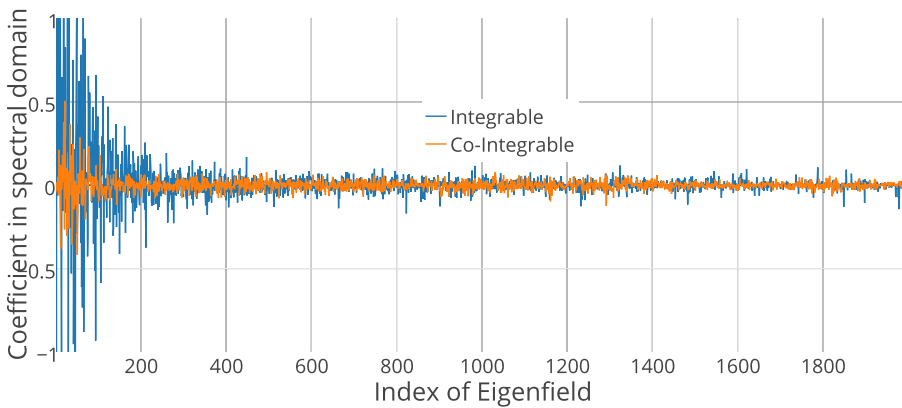


Figure 1.8: Spectral analysis of the field shown in Figure 1.7.

In addition, constants γ_h and γ_r can be defined, which specify the scaling of the harmonic part of the field and the rest field \mathbf{v}^r . A typical “low-pass” filter is $F_{\alpha/\beta}(\lambda) = e^{-\lambda}$ which keeps low frequencies intact while exponentially suppressing high frequencies, which can be used to simplify fields or remove unwanted noise. In the same vein, a “high-pass” filter is $F_{\alpha/\beta}(\lambda) = 2 - e^{-\lambda}$, which can be used to “sharpen” the field. The integrable and co-integrable parts of the field can be filtered separately, which allows for a spectral analysis of the divergence-free and curl-free part of the vector field.

Spectral filtering can be seen in action in Figure 1.7, where we show how it can be used to interactively edit and analyze tangential fields. As a first step, we perform a spectral analysis. The plot of the coefficients of the field in the spectral domain for both the integrable and co-integrable eigenfields is shown in Figure 1.8. The unaltered field is shown on the left of Figure 1.7. In the magnified area the vectors seem to exhibit noise, and indeed, when applying a low-pass filter on both the field and additionally enhancing the integrable low-frequency part (middle), we get a smoothed and more structured field. Since we magnified only the integrable part, the vortex on the stomach of the centaur disappears. To analyze the noisy features, we enhance the high-frequency part of

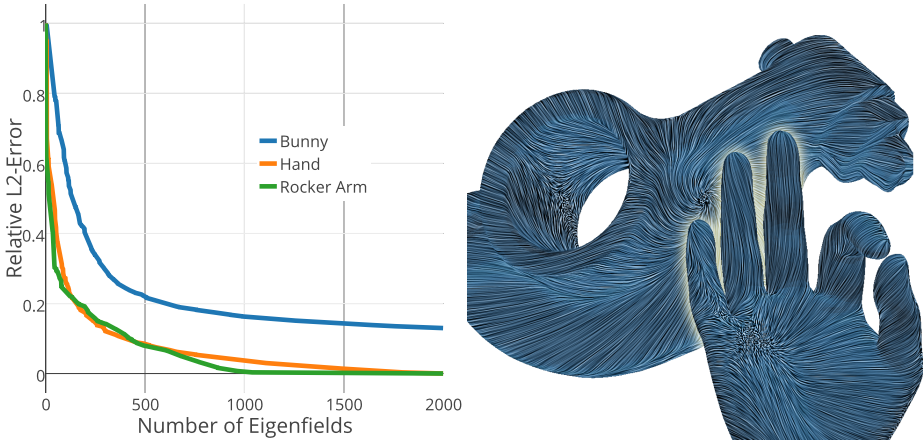


Figure 1.9: Rate distortion curves when compressing various custom tangential vector fields on three different meshes.

the field and remove the low frequency integrable part (right). We can see that the high-frequency part mainly consists of two vortices on the side of the centaur (highlighted).

1.8.6. COMPRESSION

The spectral decomposition and the transformation into the frequency domain described above can immediately be used to reduce data rates required to represent tangential vector fields on meshes: instead of storing a vector field on a triangular mesh by specifying two values per triangle (coordinates with respect to an edge or similar common representations), a field can be represented by the $n_v + n_e + 2g$ coefficients $\alpha_i, \beta_i, \gamma_i$. Compression can be achieved by cutting off the higher coefficients, since the high frequency part of the fields is typically small. This approach extends existing approaches from mesh compression [5] or dynamic mesh compression [6] to the compression of tangential vector fields on meshes. Rate-distortion curves can be found in Figure 1.9.

In Figure 1.10, we compressed a field exhibiting over 100 small curls on a bunny mesh using the lowest 1k integrable and co-integrable eigenfields. All features are kept faithfully intact.

In Figure 1.9, left, we plot the relative L^2 -error when using a varying number of eigenfields to reproduce the fields on the hand model and on the rocker arm, both shown in Figure 1.9, right, as well as the field on the bunny shown in Figure 1.10. The plot is essentially a rate-distortion curve, as the number k of pairs of eigenfields used to compress the field directly relates to the data rate, namely the number of bits per vertex is $2 \cdot 64 \cdot k / n$ (where n is the number of vertices) when using double precision for the coefficients. In case of the tangential field on the bunny mesh, the error only slowly converges to 0, since the field contains a lot of high frequency elements (of course, the error still reaches 0 as the number of eigenfields reaches the number of vertices plus the number of edges).

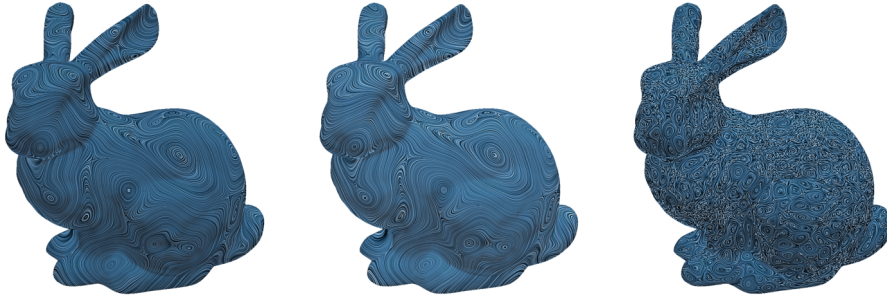


Figure 1.10: A field exposing a lot of small features (left) and the compressed version made from the 1000 lowest pairs (middle). In the last picture (right) we show the part of the original field that cannot be constructed from the 1000 lowest pairs of eigenfields.

This is visualized in Figure 1.10, right, where the part of the field is shown which *cannot* be reproduced by the 1k lowest pairs of eigenfields (note that the LIC visualization shows the structure of the field, but does not reveal its magnitude). When creating the same plot for a more regular field, namely the ones shown in Figure 1.9, we can see that we are able to reach an almost lossless compression by using the 2k lowest pairs of eigenfields. It is worth noting that quite a large number of eigenfields is required to get a low L^2 -error, even for this fairly simple field, while a very small number of eigenfields is required to get visually indistinguishable results that preserve all the large features. The compression of the field on the rocker arm yields the best results, where we get an essentially lossless compression for $k = 1100$, which corresponds to a compression rate of 36.52 when comparing to the usual representation of two doubles per triangle.

In Figure 1.11 we show a snapshot of a time-dependent tangent field on the bumpy torus (140240 faces), consisting of 500 fields, which amounts to a file size of 560,96 Megabyte when representing the vectors in each face by two single precision floating point numbers (4 bytes). We compress the sequence using a basis of 500 eigenfields which amounts to a file-size of exactly 1 Megabyte. On average we get a relative L^2 -error of 10 percent between compressed and uncompressed frames, however, we found the two time series to be visually indistinguishable.

The timings for the compression are very low, once the eigenfields have been computed, which only needs to be done once per mesh (for timings see Table 1.1). Whenever a new field on the mesh is to be compressed, all that needs to be done is to compute the L^2 -scalar products of the field with the $2 \cdot k$ eigenfields (0.67 seconds for the hand-mesh and $k = 1000$). Recovering the usual representation of the vector simply requires a dense matrix vector product of the matrix containing the eigenfields as columns with the coefficient vector (0.06 seconds for the hand-mesh and $k = 1000$, below 1ms when done using the GPU).

1.9. CONCLUSION

We introduce a framework for spectral processing of tangential vector fields using a Fourier-type representation of tangential vector fields that associates frequencies with

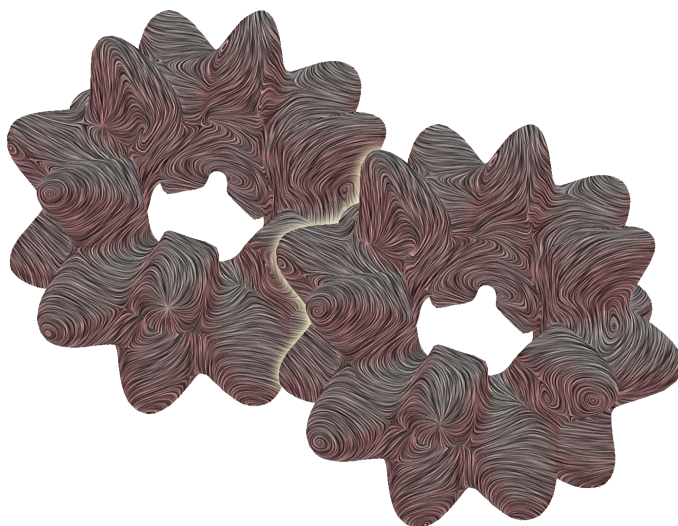


Figure 1.11: Snapshot from the uncompressed (left) and compressed (right) time-dependent tangential vector field on the bumpy torus.

tangential vector fields. To formulate the framework for piecewise constant vector fields on surface meshes, we introduce a discretization of the Hodge–Laplace operator. We demonstrate how techniques from spectral mesh processing can be transferred to tangential vector field processing using this framework. We show results for spectral filtering, analysis and compression. Moreover, we introduce a spline-like editor for modeling tangential vector fields using interpolation constraints. Based on the spectral representation, we propose a computational scheme that enables modeling of tangential vector field splines in real-time.

REFERENCES

- [1] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe, *Design of tangent vector fields*, ACM Trans. Graph. **26**, 56:1 (2007).
- [2] B. Lévy and H. Zhang, *Spectral mesh processing*, in *ACM SIGGRAPH ASIA Courses* (2009) pp. 1–47.
- [3] H. Zhang, O. van Kaick, and R. Dyer, *Spectral mesh processing*, Comput. Graph. Forum **29**, 1865 (2010).
- [4] B. Vallet and B. Lévy, *Spectral geometry processing with manifold harmonics*, Computer Graphics Forum (2008).
- [5] Z. Karni and C. Gotsman, *Spectral compression of mesh geometry*, in *ACM SIGGRAPH* (2000) pp. 279–286.

- [6] L. Váša, S. Marras, K. Hormann, and G. Brunnett, *Compressing dynamic meshes with geometric Laplacians*, *Computer Graphics Forum* **33**, 145 (2014).
- [7] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart, *Spectral surface quadrangulation*, *ACM Trans. Graph.* **25**, 1057 (2006).
- [8] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao, *Spectral quadrangulation with orientation and alignment control*, *ACM Trans. Graph.* **27**, 1 (2008).
- [9] R. Ling, J. Huang, B. Jüttler, F. Sun, H. Bao, and W. Wang, *Spectral quadrangulation with feature curve alignment and element size control*, *ACM Trans. Graph.* **34**, 11:1 (2014).
- [10] A. Sharma, R. P. Horaud, D. Knossow, and E. von Lavante, *Mesh segmentation using Laplacian eigenvectors and Gaussian mixtures*, in *Manifold Learning and Its Applications* (2009).
- [11] Q. Huang, M. Wicke, B. Adams, and L. Guibas, *Shape decomposition using modal analysis*, *Computer Graphics Forum* **28**, 407 (2009).
- [12] M. Reuter, F.-E. Wolter, and N. Peinecke, *Laplace-spectra as fingerprints for shape matching*, in *Symposium on Solid and Physical Modeling* (2005) pp. 101–106.
- [13] M. Reuter, F.-E. Wolter, and N. Peinecke, *Laplace–Beltrami spectra as "Shape-DNA" of surfaces and solids*, *Computer-Aided Design* **38**, 342 (2006).
- [14] R. M. Rustamov, *Laplace–Beltrami eigenfunctions for deformation invariant shape representation*, in *Symposium on Geometry Processing* (2007) pp. 225–233.
- [15] J. Sun, M. Ovsjanikov, and L. J. Guibas, *A concise and provably informative multi-scale signature based on heat diffusion*. *Computer Graphics Forum* **28**, 1383 (2009).
- [16] K. Gebal, J. A. Bærentzen, H. Aanæs, and R. Larsen, *Shape analysis using the auto diffusion function*, *Computer Graphics Forum* **28**, 1405 (2009).
- [17] M. Aubry, U. Schlickewei, and D. Cremers, *The wave kernel signature: A quantum mechanical approach to shape analysis*, in *ICCV* (2011) pp. 1626–1633.
- [18] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Eigenmodes of surface energies for shape analysis*, in *Proceedings of Geometric Modeling and Processing* (2010) pp. 296–314.
- [19] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Modal shape analysis beyond Laplacian*, *Computer Aided Geometric Design* **29**, 204 (2012).
- [20] M. Ovsjanikov, J. Sun, and L. Guibas, *Global intrinsic symmetries of shapes*, *Computer Graphics Forum* **27**, 1341 (2008).
- [21] M. Ovsjanikov, Q. Mérigot, F. Méholi, and L. Guibas, *One point isometric matching with the heat kernel*, *Computer Graphics Forum* **29**, 1555 (2010).

- [22] T. K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang, *Persistent heat signature for pose-oblivious matching of incomplete models*, *Computer Graphics Forum* **29**, 1545 (2010).
- [23] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, *Functional maps: A flexible representation of maps between shapes*, *ACM Trans. Graph.* **31**, 30:1 (2012).
- [24] R. M. Rustamov, M. Ovsjanikov, O. Azencot, M. Ben-Chen, F. Chazal, and L. Guibas, *Map-based exploration of intrinsic shape differences and variability*, *ACM Trans. Graph.* **32**, 72:1 (2013).
- [25] U. Pinkall and K. Polthier, *Computing discrete minimal surfaces and their conjugates*, *Experimental Mathematics* **2**, 15 (1993).
- [26] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete differential-geometry operators for triangulated 2-manifolds*, in *Visualization and Mathematics III* (Springer, 2003) pp. 35–57.
- [27] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, *Discrete Laplace operators: No free lunch*, in *Symposium on Geometry Processing* (2007) pp. 33–37.
- [28] M. Alexa and M. Wardetzky, *Discrete Laplacians on general polygonal meshes*, *ACM Trans. Graph.* **30**, 102:1 (2011).
- [29] K. Hildebrandt, K. Polthier, and M. Wardetzky, *On the convergence of metric and geometric properties of polyhedral surfaces*, *Geometricae Dedicata* **123**, 89 (2006).
- [30] K. Hildebrandt and K. Polthier, *On approximation of the Laplace–Beltrami operator and the Willmore energy of surfaces*, *Computer Graphics Forum* **30**, 1513 (2011).
- [31] T. K. Dey, P. Ranjan, and Y. Wang, *Convergence, stability, and discrete approximation of Laplace spectra*, in *Symp. Discrete Algorithms* (2010) pp. 650–663.
- [32] C. Wagner, C. Garth, and H. Hagen, *Harmonic field analysis*, in *New Developments in the Visualization and Processing of Tensor Fields* (Springer, 2012) pp. 363–379.
- [33] T. De Witt, C. Lessig, and E. Fiume, *Fluid simulation using laplacian eigenfunctions*, *ACM Trans. Graph.* **31**, 10:1 (2012).
- [34] S. U. Mehta, R. Ramamoorthi, M. Meyer, and C. Hery, *Analytic tangent irradiance environment maps for anisotropic surfaces*, *Comput. Graph. Forum* **31** (2012).
- [35] B. Raymond, G. Guennebaud, P. Barla, R. Pacanowski, and X. Granier, *Optimizing brdf orientations for the manipulation of anisotropic highlights*, *Comput. Graph. Forum* **33**, 313 (2014).
- [36] A. Hertzmann and D. Zorin, *Illustrating smooth surfaces*, in *Proc. ACM SIGGRAPH* (2000).

- [37] C.-Y. Yao, M.-T. Chi, T.-Y. Lee, and T. Ju, *Region-based line field design using harmonic functions*, IEEE Trans. Vis. Comp. Graph. **18** (2012).
- [38] M. Chi, C. Yao, E. Zhang, and T. Lee, *Optical illusion shape texturing using repeated asymmetric patterns*, The Visual Computer **30** (2014).
- [39] L.-Y. Wei and M. Levoy, *Texture synthesis over arbitrary manifold surfaces*, in Proc ACM SIGGRAPH (2001).
- [40] F. Knöppel, K. Crane, U. Pinkall, and P. Schröder, *Stripe patterns on surfaces*, ACM Trans. Graph. **34** (2015).
- [41] O. Azencot, S. Weißmann, M. Ovsjanikov, M. Wardetzky, and M. Ben-Chen, *Functional fluids on surfaces*, Computer Graphics Forum **33** (2014).
- [42] O. Azencot, O. Vantzou, M. Wardetzky, M. Rumpf, and M. Ben-Chen, *Functional thin films on surfaces*, in Symposium on Computer Animation (2015) pp. 137–146.
- [43] J. Solomon, M. Ben-Chen, A. Butscher, and L. Guibas, *Discovery of intrinsic primitives on triangle meshes*, (2011) pp. 365–374.
- [44] Y. Zhuang, M. Zou, N. Carr, and T. Ju, *Anisotropic geodesics for live-wire mesh segmentation*, Comput. Graph. Forum **33** (2014).
- [45] E. Iarussi, D. Bommes, and A. Bousseau, *Bendfields: Regularized curvature fields from rough concept sketches*, ACM Trans. Graph. **34** (2015).
- [46] H. Pan, Y. Liu, A. Sheffer, N. Vining, C.-J. Li, and W. Wang, *Flow aligned surfacing of curve networks*, ACM Trans. Graph. **34** (2015).
- [47] F. do Goes, M. Desbrun, and Y. Tong, *Vector field processing on triangle meshes*, in SIGGRAPH Asia 2015 Courses (2015) pp. 17:1–17:48.
- [48] A. Vaxman, M. Campen, O. Diamanti, D. Panozzo, D. Bommes, K. Hildebrandt, and M. Ben-Chen, *Directional field synthesis, design, and processing*, Comput. Graph. Forum **35**, 545 (2016).
- [49] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez, *Periodic global parameterization*, ACM Trans. Graph. **25**, 1460 (2006).
- [50] F. Kälberer, M. Nieser, and K. Polthier, *Quadcover - surface parameterization using branched coverings*, Comput. Graph. Forum **26** (2007).
- [51] D. Bommes, H. Zimmer, and L. Kobbelt, *Mixed-integer quadrangulation*, ACM Trans. Graph. **28**, 77:1 (2009).
- [52] E. Li, B. Lévy, X. Zhang, W. Che, W. Dong, and J.-C. Paul, *Meshless quadrangulation by global parameterization*. Computers & Graphics (2011).
- [53] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, and P. Cignoni, *Simple quad domains for field aligned mesh parametrization*, Proc. SIGGRAPH Asia 2011 **30** (2011).

- [54] H.-C. Ebke, M. Campen, D. Bommès, and L. Kobbelt, *Level-of-detail quad meshing*, ACM Trans. Graph. **33**, 184:1 (2014).
- [55] M. Campen and L. Kobbelt, *Quad layout embedding via aligned parameterization*, Computer Graphics Forum **33**, 69 (2014).
- [56] Y. Li, Y. Liu, and W. Wang, *Planar hexagonal meshing for architecture*, IEEE Trans. Vis. Comput. Graph. **21** (2015).
- [57] D. Bommès, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin, *Quad-mesh generation and processing: A survey*, Comput. Graph. Forum **32**, 51 (2013).
- [58] K. Polthier and E. Preuß, *Variational approach to vector field decomposition*, in *Symposium on Data Visualization* (Springer, 2000) pp. 147–155.
- [59] K. Polthier and E. Preuß, *Identifying vector field singularities using a discrete hodge decomposition*, in *Visualization and Mathematics III* (Springer, 2003) pp. 113–134.
- [60] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun, *Discrete multiscale vector field decomposition*, ACM Trans. Graph. **22**, 445 (2003).
- [61] M. Wardetzky, *Discrete Differential Operators on Polyhedral Surfaces—Convergence and Approximation*, Ph.D. thesis, Freie Universität Berlin (2006).
- [62] H. Bhatia, G. Norgard, V. Pascucci, and P.-T. Bremer, *The Helmholtz–Hodge decomposition—a survey*, IEEE Transactions on Visualization and Computer Graphics **19**, 1386 (2013).
- [63] E. Zhang, K. Mischaikow, and G. Turk, *Vector field design on surfaces*, ACM Trans. Graph. **25**, 1294 (2006).
- [64] E. Zhang, J. Hays, and G. Turk, *Interactive tensor field design and visualization on surfaces*, IEEE Trans. Vis. Comp. Graph. **13**, 94 (2007).
- [65] N. Ray, B. Vallet, W. C. Li, and B. Lévy, *N-symmetry direction field design*, ACM Trans. Graph. **27** (2008).
- [66] F. Knöppel, K. Crane, U. Pinkall, and P. Schröder, *Globally optimal direction fields*, ACM Trans. Graph. **32**, 59:1 (2013).
- [67] O. Diamanti, A. Vaxman, D. Panozzo, and O. Sorkine-Hornung, *Designing n-PolyVector fields with complex polynomials*, Comput. Graph. Forum **33**, 1 (2014).
- [68] M. Desbrun, A. Hirani, M. Leok, and J. Marsden, *Discrete exterior calculus*, (2005), preprint, arXiv:math.DG/0508341.
- [69] D. N. Arnold, R. S. Falk, and R. Winther, *Finite element exterior calculus, homological techniques, and applications*, Acta numerica **15** (2006).
- [70] F. W. Warner, *Foundations of differentiable manifolds and Lie groups*, Vol. 94 (Springer, 2013).

- [71] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, and E. Grinspun, *Discrete quadratic curvature energies*, *Comput. Aided Geom. Des.* **24** (2007).
- [72] I. Agricola and T. Friedrich, *Global Analysis: Differential Forms in Analysis, Geometry, and Physics* (AMS, 2002).
- [73] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing* (AK Peters, 2010).
- [74] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, *A fully asynchronous multi-frontal solver using distributed dynamic scheduling*, *SIAM Journal on Matrix Analysis and Applications* **23**, 15 (2001).

2

INTERACTIVE MODELING OF n -FIELD SPLINES

This chapter is based on the publication *Modeling n -Symmetry Vector Fields using Higher-Order Energies* by Christopher Brandt, Leonardo Scandolo, Elmar Eisemann and Klaus Hildebrandt, published in ACM Transactions on Graphics in 2018.

2.1. OVERVIEW

In the last chapter, we showed that the design, synthesis and processing of tangential vector fields on surfaces is essential for various applications in computer graphics. Often, however, we are not dealing with classical vector fields, but with n -fold rotational symmetry vector fields (or n -vector fields), like line fields ($n=2$) or cross fields ($n=4$). The structure of n -vector fields differs significantly from that of classical vector fields. For example, n -vector fields allow for more general singularities of fractional degree, and, compared to vector fields, the calculus of n -vector fields is scarcely developed. Therefore, the processing of n -vector fields poses challenging problems and promises rewarding benefits for the applications.

The goal of this chapter is to develop the techniques needed for a *modeling tool* for n -vector fields on surfaces that includes the following features:

1. *Hard interpolation constraints.* This enables users to create smooth n -vector fields by specifying a sparse sets of interpolation constraints.
2. *Smooth dependence on the constraints.* The constructed n -vector fields should depend smoothly on the constraints. This allows users to edit n -vector fields by modifying the constraints as changing the constraints smoothly changes the n -vector field that is modeled.
3. *Local editing.* To fine-tune results, the user should be able to mark a region and only model the field inside the region without affecting the n -vector field outside the marked region.

4. *Real-time responses.* To enable modeling and fine-tuning of n -vector fields, immediate responses are needed.
5. *n -direction fields.* The tool should allow modeling of n -direction fields, which are n -vector fields consisting only of unit vectors.

2

For surface modeling and other modeling tasks, tools offering analogous to the first four features proved to be effective. Therefore we are convinced that a modeling tool for n -vector fields that combines these features is helpful for a variety of graphics applications. The fifth feature is important for applications in which directions rather than vectors are needed and the magnitude of vectors is not relevant.

To realize the first three goals, we introduce *n -field splines*, a variational analogue to the tangential vector field splines introduced in the last chapter. They offer a novel approach for modeling n -vector fields and n -direction fields on surfaces. The basis of this approach are novel higher-order fairness energies for n -vector fields: a biharmonic energy and more generally m -harmonic energies. The n -vector field splines are defined, analogously to the variational characterization of classical spline functions, as the minimizers of a higher-order fairness energy subject to constraints. The use of higher-order energies enables us to integrate constraints that realize the desired modeling features to the variational problem. Local editing and interpolation constraints at single points can be enforced in the optimization and the higher-order energies ensure that the resulting minimizers smoothly transition from the edited to the constrained region and behave smoothly around the interpolation constraints.

To implement the concept of n -field splines, we developed several new techniques. We introduce a biharmonic energy and more generally m -harmonic energies for piecewise constant (face-based) n -vector fields on triangle surface meshes. By applying a principle for the design of quadratic fairness energies for direction fields proposed in [1] to our setting, we extend the m -harmonic energies for n -vector fields to m -harmonic energies for n -direction fields. Secondly, we integrate hard interpolation and alignment constraints, as well as constraints for placing singularities to the minimization of the higher-order energies for n -vector and n -direction fields. These approaches extend the weak alignment constraints for the globally optimal n -direction field approaches introduced in [1] and [2]. Thirdly, we propose an efficient approximation algorithm for n -field splines that allows for real-time modeling. The n -field splines are solutions of sparse linear systems, and, therefore, they can be robustly computed.

However, for many applications, in particular for the modeling of n -fields, interactive responses are desired or even necessary. To ensure that real-time editing is possible, independently of the resolution of the underlying surface, a model reduction approach for the computation of smoothest n -fields is required. Therefore, akin to the previous chapter, we will make use of specific subspaces, acquired from a Fourier-like spectral decomposition of the Laplace operator for n -fields. We restrict the modeled n -fields to be linear combinations of the lowest frequency eigenfields of this operator, which additionally regularizes the fields. Since the quadratic fairness energies in the subspace are represented by diagonal matrices, the computation of the reduced solutions is very fast. After a preprocessing stage in which the eigenfields of the n -field Laplacian are computed, we obtain computation times of few milliseconds for all meshes we tested and

a speed-up of a factor up to 100 over solving the full linear system using sparse direct solvers.

To emphasize the applicability of the n -field splines and the real-time solver, we apply the resulting modeling tool to two graphics problems: real-time editing of hatchings of surfaces and interactive design of anisotropic BRDFs on surfaces. In addition to these applications, we think that the proposed techniques hold potential for quad meshing applications. For example, in [3] the computational cost of vector field design is named as one of the factors that hinder interactive quad meshing via integer grid maps. Our scheme could potentially remove this barrier.

2.2. RELATED WORK

Tools for designing n -fields are important for various applications in computer graphics. They are used for texture generation [4–8], non-photorealistic line art [9] and painterly rendering [10], image stylization [11], anisotropic shading [12, 13], quad-remeshing [14–18] or hexagonal parameterization [19], and surface segmentation [20, 21] to name just a few examples. The processing of n -fields poses challenging problems and much work has been dedicated to establishing techniques that tackle these problems. In the following, we outline approaches closest related to the proposed work. For more background and further references, we refer to the recent surveys [22, 23].

The method presented in this chapter relates to spectral methods in computer graphics and generalizes the design of tangential vector fields. We refer to Section 1.2 of the previous chapter for references to related literature in these areas. In the following we focus on literature specifically concerned with n -fields and higher order smoothness energies.

Variational n -field design In variational n -field design, n -fields are constructed as solutions to optimization problems, which aim for the smoothest n -fields that satisfy design goals specified by the users. The smoothness of a n -field is measured by a fairness energy, an objective that quantifies the variation of the field along the surface. To compare the n -vectors of a n -vector fields at nearby points (*e.g.* neighboring triangles or vertices), the n vectors at one point are parallel transported along the shortest geodesic to the tangent space of the other point. One way to quantify the difference between two n -vectors (once they are in the same tangent space) is to select an arbitrary vector from both n -vectors and to measure the oriented angle between the selected vectors. Multiplying this angle by n , yields a quantity that, up to a multiple of 2π , is independent of the choice of vectors. The cosine of this quantity agrees with the cosine of the smallest angle between pairs of vector from the two n -vectors and one minus this cosine can be used as a measure of the deviation of the n -vectors. Based on this idea, a fairness energy for n -direction fields was introduced in [9]. This approach was extended by the concept of the representation vector [14, 24], which is an alternative representation of n -vector fields on meshes. Modulo π/n , the n vectors of a n -vector make the same angle to a fixed coordinate direction in the tangent plane, hence, multiplying the angles by n yields a unique representation vector. This concept allows to formulate design of n -vector or

n -direction fields as an optimization of the representation vector fields [14]. In recent work [1, 2], a n -vector field representation using complex numbers and fairness energies that are quadratic with respect to the complex representation are introduced. The benefit of this approach is that globally optimal solutions can be computed by solving sparse linear systems. The approach can be extended to the computation of optimal n -direction fields. This is achieved by imposing a constraint on the L^2 -norm of the n -vector field during optimization and pointwise normalization afterwards. For controlling the design, [1] use a weak alignment constraint to an input field, e.g. the principal curvatures directions of the surfaces. In [2] this approach is extended by a stroke-based design metaphor, in which fields weakly align with strokes placed by users. Globally optimal fields that weakly align with the strokes and an alignment field are computed and runtimes of 1s for a mesh with 50k triangles are reported. We extend this approach. It allows for spline-like modeling of n -vector fields including features like modeling with interpolation constraints, local editing and real-time responses. Our biharmonic energy for n -vector fields is also a quadratic energy, but compared to the Dirichlet energy of [1], the corresponding Euler-Lagrange equation is of higher order, which is a prerequisite for modeling with interpolation constraints and local editing.

Mixed-integer problems An alternative approach to using the representation vector is to introduce explicit assignments, so-called matchings, between the n vectors at neighboring triangles (or vertices depending on the discretization used). Once a matching is fixed, differences between n -vectors can be measured using common measures for comparing vectors. For n -vector and n -direction field design, all possible matchings are introduced as variables to the optimization with the goal to find the best possible n -field and matching [15, 16, 25]. This means that mixed-integer problems need to be solved for n -field construction. Hard interpolation constraints have been used for mixed-integer based n -field design. However, since mixed-integer problems need to be solved for field construction, these methods do not provide real-time responses.

Real-time design Recently, Jakob et al. [26] introduced a method for real-time quadrilateral and hexagonal mesh generation. The scheme proceeds in two stages: n -field design and mesh generation based on the n -field. To obtain a real-time system, the n -field design is not done by solving a global optimization problem, instead a multiresolution hierarchy is set up and local optimization steps are performed on the different levels of the hierarchy from coarse to fine. For efficiency, the objective for the local optimization steps is an extrinsic fairness energy for n -fields that does not need parallel transport of vectors. As a consequence, the objective depends not just on intrinsic properties of the surface but also on its extrinsic curvatures. In [27] the extrinsic fairness energy is further explored and the relations between extrinsic and intrinsic fairness energies are analyzed. Since [26] is the only scheme that can construct n -fields at rates comparable to our scheme, we include a comparison to their method to Section 2.9.

Polyvectors Polyvectors [28] extend the idea of a representation vector for rotational symmetric n -vectors to arbitrary (non-symmetric) n -vectors. The idea is to assign to every n -vector the complex polynomial that has the n vectors as its roots, where vectors in

\mathbb{R}^2 are identified with complex numbers. The space of polynomials of degree n is a vector space and n -vector field design problems can be formulated using this representation. A harmonic energy for polyvectors was introduced in [28] and additional objectives for quad-remeshing in [29]. For a discussion of benefits and drawbacks of the different representations of n -vector fields, we refer to the survey [23].

Controlling topology Singularities are salient features of n -fields. Methods for controlling and editing singularities of vector, n -vector and n -direction fields and for generating fields from given sets of singularities have been proposed [2, 24, 30–34]. Our approach combines variational field design with the enforcement of singularities. Singularities can be placed on the surface and the higher-order fairness energies ensure smooth transition of the field around the singularities.

Higher-order energies Optimization problems involving higher-order energies, like the biharmonic, thin plate or Willmore energy, have been used for example for fairing [35, 36], variational surface modeling [37, 38], deformation-based mesh editing [39, 40] and the construction of skinning weights [41, 42]. One example of a benefit provided by higher-order energies is more control over the boundary behavior. For example, biharmonic problems allow to prescribe positions and derivatives (in normal direction) at the boundary, which allow to create G^1 -continuous transitions at boundaries of surface patches [35]. In contrast, using harmonic problems only G^0 -transitions can be obtained. A second example is that interpolation constraints can be imposed at single points. For example, the minimizers of the thin plate energy over a two-dimensional domain subject to interpolation constraints at single points exist and are uniquely defined [43]. In contrast, minimizers of the harmonic energy over a two-dimensional domain subject to point constraints in general are discontinuous, see [44, pp 50–51] for an example. For applications like surface modeling and deformation it is desirable to be able to impose constraints on single points. The biharmonic and m -harmonic energies we propose provide the benefits of higher-order energies for the modeling of n -vector and n -direction fields.

2.3. BACKGROUND: n -VECTOR FIELDS

A n -symmetry vector (short: n -vector) in \mathbb{R}^2 is a set $\{v_1, v_2, \dots, v_n\}$ of n vectors with a $\frac{2\pi}{n}$ -fold rotational symmetry, *i.e.* rotations by $\frac{2\pi}{n}$ map the set to itself. For example, a 1-vector is an ordinary vector and a 2-vector is a pair $\{v, -v\}$, where $v \in \mathbb{R}^2$ is arbitrary.

Representation vector Consider the map that rotates any vector such that its argument, *i.e.* the oriented angle with the x -axis, scales by a factor of n and the length is preserved. If this map is applied to a n -vector $\{v_1, v_2, \dots, v_n\}$, all elements v_i have the same image u , which is called the representation vector of $\{v_1, v_2, \dots, v_n\}$. Moreover, the map from n -vectors to representation vectors is a bijection, which means that every n -vector has a unique representation vector and every vector \mathbb{R}^2 is the representation vector of a n -vector. The representation vector provides additional structure that we will use to work with n -vectors. For example, we can add n -vectors by converting them to

representation vectors, adding the representation vectors and converting them back to n -vectors. In combination with the natural scaling of n -vectors, we obtain a vector space structure on the set of n -vectors.

For later use, we want to remark that the representation vector changes when moving from one coordinate system to another. We consider a rotation of the coordinate system by an angle of φ . Then the coordinates of the n -vector in the rotated system are given by $\{R^{-\varphi} v_1, R^{-\varphi} v_2, \dots, R^{-\varphi} v_n\}$, where $R^{-\varphi}$ denotes the rotation by $-\varphi$. The representation vector in the rotated coordinate system is $R^{-n\varphi} u$. This means, the representation vector u is rotated by $-n\varphi$ when the coordinate system is rotated by φ .

n -vector fields We consider tangential n -vector fields on triangle meshes that are constant in every triangle. To work with such n -vector fields, we fix a coordinate system in the tangent plane of every triangle and consider the vector $\mathbf{u} = (u_1, u_2, \dots, u_{|\mathcal{F}|}) \in \mathbb{R}^{2|\mathcal{F}|}$, where $u_i \in \mathbb{R}^2$ is the representation vector of the n -vector of triangle T_i (with respect to the coordinate system fixed in T_i) and $|\mathcal{F}|$ is the number of triangles of the mesh. Explicitly, we choose one oriented edge in every triangle and use this as the x -axis of the coordinate system. The vector \mathbf{u} completely describes the n -field, e.g. we can reconstruct the n -vectors of every triangle T_i from its representation vector u_i . We will perform all computations using the representation vectors. Only once we convert initial input, like an alignment field, to representation vectors, and, after the computation is done, we convert the result into a n -vector field for visualization or other applications like constructing a hatching or BRDF on the surface. To simplify the presentation, we will sometimes refer to \mathbf{u} as the n -vector field and u_i as the n -vector and rely on the context to make the distinction between representation vector and n -vector.

The representation vectors, hence also the piecewise constant n -vector fields on a mesh, form a vector space. On this space, we consider the L^2 -scalar product

$$\langle \mathbf{u}, \mathbf{u}' \rangle_{L^2} = \sum_i \text{area}(T_i) \langle u_i, u'_i \rangle$$

and the corresponding L^2 -norm

$$\|\mathbf{u}\|_{L^2}^2 = \langle \mathbf{u}, \mathbf{u} \rangle_{L^2},$$

where $\mathbf{u}, \mathbf{u}' \in \mathbb{R}^{2|\mathcal{F}|}$.

n -direction fields n -direction fields are n -vector fields consisting only of unit-length n -vectors. These fields are of particular interest as for many applications the magnitudes of the vectors are irrelevant.

Transport of n -vectors The fairness measures, which will be introduced in the following sections, compare the n -vectors of a n -vector fields in adjacent triangles. To do so, one n -vector is parallel transported from its tangent plane to the tangent plane of the other n -vector. In this paragraph, we discuss the transport of a n -vector from a triangle T_i to an adjacent triangle T_j . We fix coordinate systems in both triangles and denote by $u = (u_x, u_y)$ the coordinates of the corresponding representation vector in T_i .

If the x -axes of the coordinate systems in both triangles are aligned with the oriented edge e_{ij} that is common to both triangles, the transport is simply given by the identity matrix, *i.e.* the coordinates of the transported vector in the triangle T_j agree with the coordinates u of the vector in T_i . In the general case, where the coordinate systems in the triangles T_i and T_j may not align to the common edge, we first transform to the e_{ij} -aligned coordinate system in T_i , then use the trivial transport to the e_{ij} -aligned coordinate system in T_j , and, finally, transform to the (non e_{ij} -aligned) coordinate system in T_j . Let φ_{ij} and φ_{ji} be the oriented angles between the edges chosen as x -axis in T_i and T_j and the common edge e_{ij} . Then, the representation vector transforms by multiplication with the rotation matrices $R^{-n\varphi_{ij}}$ and $R^{-n\varphi_{ji}}$. Altogether, the transport of the representation vector from T_i to T_j is given by a rotation

$$R_{ij} = R^{n(\varphi_{ji} - \varphi_{ij})}. \quad (2.1)$$

In practice, it is convenient to precompute the rotations R_{ij} for every pair of adjacent triangles.

We want to remark that the R_{ij} s form a discrete connection on the mesh, see [1, 33] for more background on discrete connections. Since the R_{ij} s depend on n , the connections differ for the different types of n -fields. For $n=1$, the R_{ij} s agree with the usual parallel transport of vectors (discrete Levi-Civita connection) induced by the metric on ambient 3-space.

Singularities The singularities of a n -vector field are of fractional degree $\frac{\iota}{n}$, where $\iota \in \mathbb{N}$. Since we are working with vector fields that are discontinuous, the classical notion of singularities cannot be applied. Therefore, the concept of discrete singularities has been developed, see [32, 33] for more background. In our setting, the representation vectors and the discrete connection can be used to extract information about the n -field singularities. To every vertex, we associate an index which equals the number of full rotations of the vectors in the one ring around the vertex divided by n . Explicitly, let T_0, \dots, T_k be the oriented triangle 1-ring (ordered clockwise) around some vertex and u_0, \dots, u_k the corresponding n -vector representatives in each of those triangles. We then transport the vector u_0 into the next triangle T_1 and compute the signed angular difference $\psi_0 \in (-\pi, \pi]$ between the transported vector and u_1 . ψ_1, \dots, ψ_k are being computed in the same fashion, where for ψ_k , we transport u_k into the triangle T_0 . The sum of the angular differences will be an integer multiple of 2π and usually 0. If it is different from 0, we say the n -vector field has a singularity of index $\sum_j \psi_j / (2\pi n)$ at that vertex. The number singularities of a n -vector field is not arbitrary. The discrete Poincaré–Hopf theorem for n -vector fields, [25], states that the sum of the indices over all vertices equals $2 - 2g$, where g is the genus of the surface.

Polyvector fields Instead of restricting to rotational symmetric n -vectors, one can consider other constraints or even n independent vectors. Such n -vectors can be described as the roots of a complex polynomial, cf. [28]. The coefficients of the polynomial then make up a representation analogous to the representation vector of rotational symmetric n -vectors. We want to remark that the constructions proposed in this chapter can be carried over to this setting. Only the addition of representation vectors is replaced by

addition of complex polynomials and the absolute value of the representation vector is replaced by a norm for complex polynomials of degree n .

2.4. HARMONIC ENERGY FOR n -VECTOR FIELDS

The basis of the *globally optimal n -direction fields* approach introduced in [1] are quadratic fairness energies for n -vector fields. The benefit of using quadratic fairness energies for n -vector field design over previous highly-nonlinear approaches is that the globally optimal solutions can be computed by solving linear systems. Whereas in [1] a space of vertex-based vector fields is used for discretization, we are considering piecewise constant face-based vector fields, a commonly used alternative setting, here. In [1], a one-parameter family of quadratic energies is studied. The harmonic energy we consider corresponds to the anti-holomorphic energy in their notation. In this section, we first introduce a novel harmonic energy for piecewise constant n -vector fields on meshes. Then we summarize how the harmonic energy can be used for n -vector and n -direction field design with weak alignment constraints following the approach of Knöppel et al. [1].

Harmonic energy for piecewise constant n -vector fields Since we are able to transport n -vectors from a triangle to its neighbors via (2.1), we can quantify the difference of vectors in neighboring triangles via $\|R_{ij}u_i - u_j\|^2$. These differences can be used to construct a harmonic energy for piecewise constant n -vector fields $\mathbf{u} = (u_1, \dots, u_{|\mathcal{F}|})$ on triangle meshes $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$:

$$E_H(\mathbf{u}) = \sum_{(i,j) \in \mathcal{E}} w_{ij} \|R_{ij}u_i - u_j\|^2 \quad (2.2)$$

$$\text{where } w_{ij} = \frac{3l_{e_{ij}}^2}{\text{area}(T_i \cup T_j)},$$

with $l_{e_{ij}}$ being the length of the common edge between triangles T_i and T_j . This harmonic energy is a natural extension to n -vector fields of the harmonic energy for face-based, piecewise constant, tangential vector fields, see [45]. A full derivation of the energy, and the weights w_{ij} in particular, can be found in Section 2.A. The energy (2.2) is quadratic in \mathbf{u} . Hence, there is a corresponding n -vector field Laplacian Δ for piecewise constant n -fields, which is the self-adjoint operator¹ corresponding to the discrete harmonic energy, *i.e.* it is defined via

$$\forall \mathbf{u} : \langle \Delta \mathbf{u}, \mathbf{u} \rangle_{\mathbf{L}^2} = E_H(\mathbf{u}). \quad (2.3)$$

In [28], a harmonic energy (called Dirichlet energy in their paper) for piecewise constant n -vector fields was already introduced. In the following, we discuss the relation of the two energies. In our notation, the energy introduced in [28] is given by

$$\sum_{(i,j) \in \mathcal{E}} \|R_{ij}u_i - u_j\|^2.$$

¹ Δ is self-adjoint means: $\langle \Delta \mathbf{u}, \mathbf{v} \rangle_{\mathbf{L}^2} = \langle \mathbf{u}, \Delta \mathbf{v} \rangle_{\mathbf{L}^2} \forall \mathbf{u}, \mathbf{v}$. This property is needed to uniquely determine the operator Δ in eq. (2.3).

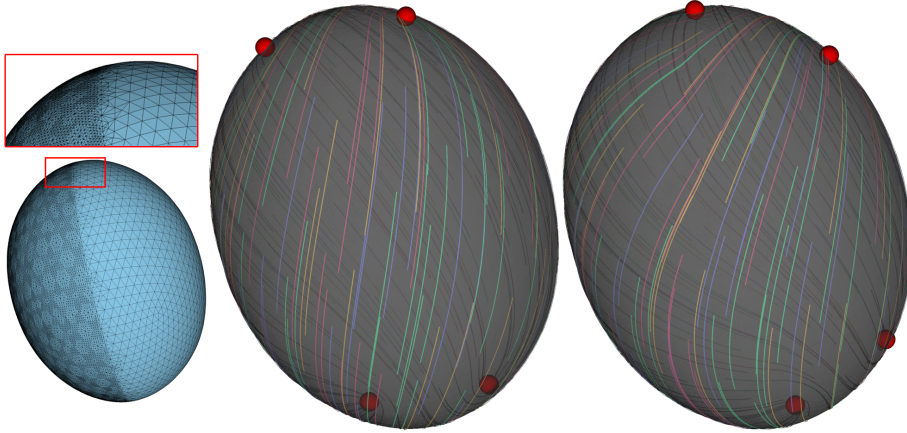


Figure 2.1: Comparing the smoothest 2-fields on an ellipse mesh with irregular triangulation (the mesh is shown on the left) using our harmonic energy (middle) and the harmonic energy proposed in [28].

The difference to the proposed harmonic energy is that no weights are used. In this sense, this is a *combinatorial* harmonic energy that does not account for the geometry of the triangulation. This leads to undesired results for meshes with irregular triangulations. An example is shown in Figure 2.1, where we compare the smoothest, 2-direction fields on a symmetric mesh using our harmonic energy and the combinatorial harmonic energy from [28]. For the combinatorial harmonic energy, the singularities appear in an unsymmetric pattern, *i.e.* the resulting fields depend on the triangulation. In contrast, when using the proposed geometric harmonic energy instead, the four singularities appear in a symmetric pattern. This is the same pattern one obtains when using a regular triangulation of the same shape.

n -direction fields The harmonic energy for n -vector fields cannot directly be used for n -direction field design. As shown in [1], minimizing the harmonic energy over n -vector fields with a unit-length constraints per vector is ill-posed. To get a well-posed problem, they propose optimizing over all rescalings of the field and adding a single L^2 -constraint on the field (*i.e.* $\|\mathbf{u}\|_{L^2}^2 = 1$) to prevent the solution $\mathbf{u} \equiv 0$. The vectors of the resulting vector field are then normalized to satisfy the pointwise unit norm constraint. Even if pointwise unit-length is not desired, but no other constraints or alignment is prescribed, the same L^2 -constraint would give a meaningful (non zero) solution to the problem of finding a smoothest n -vector field. The solution to the minimization problem

$$\min_{\|\mathbf{u}\|_{L^2}^2=1} E_H(\mathbf{u})$$

can be readily acquired by finding the smallest eigenvalue to the eigenvalue problem

$$\Delta \mathbf{u} = \lambda \mathbf{u}, \tag{2.4}$$

which can be shown using the method of Lagrange multipliers.

Field alignment Often, a balance between smoothness and the alignment to a given n -vector or direction field \mathbf{u}' is desired. For example, smooth 4-direction fields that align with estimated principal curvature directions, [46, 47], are used in applications. For n -vector fields, the following energy can be used to obtain smoothest n -vector fields aligned with some input field \mathbf{u}' :

$$E_H(\mathbf{u}) + \mu \|\mathbf{u} - \mathbf{u}'\|_{\mathbf{L}^2}^2. \quad (2.5)$$

In the case of n -direction fields, the norm of the difference of \mathbf{u} and \mathbf{u}' is no longer meaningful since during optimization, the vectors of \mathbf{u} are being freely rescaled as described above. Thus, we want to compare the angular difference between the n -vectors of \mathbf{u} and \mathbf{u}' , which can be achieved via the \mathbf{L}^2 -product for n -vector field. In particular, the following energy will be minimized:

$$E_H(\mathbf{u}) - 2\mu \langle \mathbf{u}, \mathbf{u}' \rangle_{\mathbf{L}^2}, \quad (2.6)$$

again subject to $\|\mathbf{u}\|_{\mathbf{L}^2}^2 = 1$. Due to the constraint on the \mathbf{L}^2 -norm, the scalar product of \mathbf{u} and \mathbf{u}' is bounded and minimizing the negative scalar product favors fields \mathbf{u} that align with \mathbf{u}' . The norm of the vectors of the alignment field, *i.e.* $\|u'_i\|$, can be used to control the local weighting of alignment. In particular, in regions with $u'_i = 0$, only smoothness will be taken into account.

So far we have introduced harmonic energies for face-based, piecewise constant n -vector and n -direction fields, and described how smooth fields that align to user specified n -vector or n -direction fields can be acquired by solving sparse linear systems. However, field alignment only gives an approximate tool for designing n -fields as it is oftentimes hard to specify the right weight μ and alignment vectors u'_i such that the field follows the user input satisfactorily. In the following, we will introduce higher-order smoothness energies and describe how interpolation constraints on vectors and directions can be enforced.

2.5. n -FIELD SPLINES

Similarly to the concept of tangential vector field splines, introduced in Section 1.7, n -field splines allow us to build spline-like modeling systems for n -fields. Again, the splines are characterized as the minimizers of a fairness energy subject to constraints. To ensure fairness of the solution under constraints, the order of the fairness energy has to be high enough. For n -fields on surfaces, the order of the harmonic energy derived in the previous section is not high enough to support interpolation constraints. For functions on two-dimensional domains it can be shown that minimizing the harmonic energy subject to interpolation constraints at one (or more) points, yields non-continuous solutions, see [44]. This effect can also be observed in mesh deformation and parametrization, when harmonic energies (or other energies with a second-order Euler-Lagrange equation) and hard interpolation constraints at single points are combined; we refer to [48] for examples. The same problem shows up in our experiments, an example of this kind is shown in Figure 2.2. Hence, to enable modeling of n -fields with interpolation constraints, a higher-order fairness energy is needed.

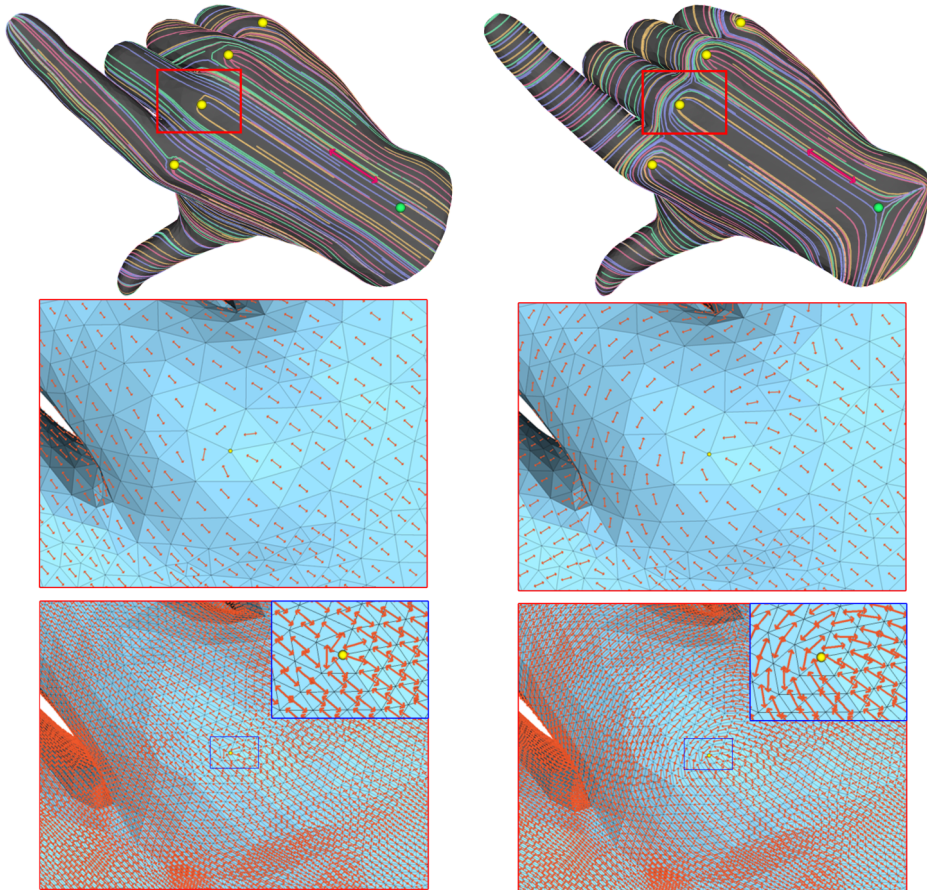


Figure 2.2: Placing singularities in a 2-direction field spline by adding hard constraints as described in Section 2.5. On the left the harmonic energy is used and the hard-constraints lead to a discontinuous solution, which does not converge under refinement (see insets). Using our higher order energy (right) allows the smooth interpolation of such constraints, which remains consistent under refinement.

m -harmonic energies The higher-order energies for n -vector fields, we introduce, are constructed using the Laplacian and the \mathbf{L}^2 -product for n -vector fields. We define the m -harmonic energies as

$$E_m(\mathbf{u}) = \langle \Delta^m \mathbf{u}, \mathbf{u} \rangle_{\mathbf{L}^2}. \quad (2.7)$$

Of particular interest is the biharmonic energy, $m = 2$, which we denote by E_B . The case $m = 1$ yields the harmonic energy. The energies E_m are quadratic and positive definite by construction. The biharmonic energy of a field \mathbf{u} equals the squared \mathbf{L}^2 -norm of the n -vector field Laplacian Δ of \mathbf{u}

$$E_B(\mathbf{u}) = \langle \Delta^2 \mathbf{u}, \mathbf{u} \rangle_{\mathbf{L}^2} = \langle \Delta \mathbf{u}, \Delta \mathbf{u} \rangle_{\mathbf{L}^2} = \|\Delta \mathbf{u}\|_{\mathbf{L}^2}^2,$$

which follows from the self-adjointness of the n -vector field Laplacian. We want to remark that starting from a discrete Laplace operator, m -harmonic energies for functions on meshes have been constructed analogously to our construction of m -harmonic energies for n -vector fields. These higher-order energies are used for example for surface modeling, fairing and deformation. For more background, we refer to the textbook by Botsch et al. [49], in particular, to Appendix A1.

In the classical setting of spline functions over an interval, minimizers of the harmonic energy under hard constraints yield piecewise linear functions and minimizers of the biharmonic energy are cubic splines. One can also build fairness energies by combining m -harmonic energies for different m . For example, in the classical case, minimizers of the weighted sum of the biharmonic and the harmonic energy are called *splines in tension*. In the following, we will focus on splines defined as minimizers of the biharmonic energy, for simplicity of presentation. Other types of n -field splines can be constructed in the same manner, just the biharmonic energy needs to be replaced by some other m -harmonic energy or a weighted sum of m -harmonic energies.

For interactive modeling, our tool will mainly use interpolation constraints. However, it is often effective to additionally use weak alignment to an existing field as a starting point. Weak alignment constraints to some input field for the splines can be imposed in the same way as describe in the previous section, only E_H is replaced by E_B in equations (2.5) and (2.6).

n -vector field splines For n -vector fields \mathbf{u} , an interpolation constraint in some triangle T_i can be enforced by a constraint of the form $u_i = d_i$, where d_i is a prescribed representation vector in T_i . To add or modify interpolation constraints in our interactive modeling system, the users do not work with representation vectors because this would be unintuitive. Instead they select a triangle and specify one vector in the triangle. The system automatically adds the missing $n - 1$ vectors. Internally, the system converts the input vector into the corresponding representation vector d_i . However, the representation vectors are only used internally and not shown to the users.

The *n -vector field splines* are defined by the variational problem

$$\begin{aligned} \operatorname{argmin}_{\mathbf{u}} E_B(\mathbf{u}) + \frac{\beta}{(\operatorname{area}(\mathcal{M}))^2} \|\mathbf{u} - \mathbf{u}'\|_{\mathbf{L}^2}^2 \\ \text{subject to } u_i = d_i \text{ for all } i \in I, \end{aligned} \quad (2.8)$$

where \mathbf{u}' is an alignment field and I is the set of triangle indices for which an interpolation constraint has been specified. The squared area of \mathcal{M} is used as a factor to make the energy invariant to rescaling of the surface. When no alignment field is given, the term $\beta/(\text{area}(\mathcal{M}))^2 \|\mathbf{u} - \mathbf{u}'\|_{\mathbf{L}^2}^2$ is removed from the problem. In addition to the interpolation constraints, other types of constraints can be imposed. We discuss singularity constraints below.

n -direction field splines In the following, we extend the approach to the modeling of n -direction fields. As for the design of n -direction fields with weak constraints, see Section 2.4, the principle is to minimize over the set n -vector fields with a constraint on the \mathbf{L}^2 -norm and obtain a n -direction field by pointwise normalization of the vectors of the minimizer. To impose interpolation constraints in this setting, we need to formulate the constraint in a way that only the direction, not the length of the vector, is prescribed. Let d_i be the representation vector of a unit n -vector prescribed as an interpolation constraint in triangle T_i . Then, we constrain u_i to be orthogonal to $J_i d_i$ (where J_i is the clockwise rotation by $\pi/2$ in the tangent plane of triangle T_i), i.e. $\langle u_i, J_i d_i \rangle = 0$. The constraint ensures that u_i is collinear with d_i . Since, we are working with representation vectors, we need to avoid the case $u_i = -t d_i$ for some $t > 0$. We prevent this case by using an alignment field \mathbf{u}' with $u'_i = s_i d_i$ for some sufficiently large $s_i > 0$. One wants to choose the s_i as small as possible while still assuring that u_i points to the right direction. Since solving the system is inexpensive, it is valid to perform a binary search for best the parameters s_i . In practice, a large enough constant for all s_i was sufficient in our tests to achieve the correct alignment direction.

The n -direction field splines are defined by

$$\begin{aligned} & \underset{\mathbf{u}}{\operatorname{argmin}} E_B(\mathbf{u}) - 2\mu \langle \mathbf{u}', \mathbf{u} \rangle_{\mathbf{L}^2} & (2.9) \\ & \text{subject to } \|\mathbf{u}\|_{\mathbf{L}^2} = 1 \\ & \text{and } \langle u_i, J_i d_i \rangle = 0 \text{ for all } i \in I, \end{aligned}$$

where \mathbf{u}' is an alignment field, with modified entries at the hard constraints as described above.

Constraints on singularities In addition to the interpolation and the weak alignment constraints, further types of constraints can be imposed on the n -field splines. In the following, we describe how we enforce singularities at vertices. Let T_0, \dots, T_k be the oriented 1-ring of triangles around some vertex v_i . Then, in order to enforce that at v_i we get a singularity with index $m \in \{q \cdot \frac{1}{n} \mid q \in \mathbb{Z}\}$, we add $k-1$ hard constraints, that enforce the n -vector in T_{j+1} to be the n -vector in T_j , rotated by $\frac{2\pi m}{k}$, for $j = 0, \dots, k-1$. When we express these constraints in terms of our n -vector field representation \mathbf{u} , they have the following form:

$$R_{j+1,j} u_{j+1} - R_{\frac{2\pi m}{k}} u_j = 0. \quad (2.10)$$

The constraints prescribe a precise rotation of the field around the singularity, and the only degree of freedom left is the magnitude and orientation of one of the n -vectors (which define the rest). This construction is a simple approach ensuring that singular

vertices with the desired index are generated. However, it has some limitations. The rotation of the vectors around the singularity is evenly distributed and all u_i on the 1-ring around the singularity are constrained to have the same magnitude. This matches our desire to compute smooth fields but does not have to be optimal in general. On the other hand, imposing the singularity constraint in a general form² would be highly non-linear and therefore increase computation times. We want to highlight that imposing hard constraints on the singularities requires higher-order energies as introduced in this chapter. Using the common harmonic energy in combination with the constraints (2.10) leads to degenerated solutions that do not converge to a smooth optimum under refinement. This will be made precise and shown by examples in Section 2.9.

With this type of singularity control we are able to enforce singularities at any vertex V_i of absolute degree smaller than $k/(2n)$, where k is the number of triangles around the vertex V_i . This limitation is due to the way we define singularities in piecewise constant n -vector fields, see Section 2.3. As described there, the sum of degrees of all singularities in a field is prescribed by the genus of the underlying mesh. Thus, inserting singularities will have global effects on the field. Alternatively singularity placement can be efficiently controlled by using hard constraints, see Figure 2.3 for an example. Completely defining the topology of a field is not a desired capability of our system since this is orthogonal to the goal of providing intuitive design tools through hard constraints and higher order smoothness.

Finally note, that this type of constraints to control the appearance of singularities is limited to n -fold rotational symmetric fields and has to be adapted individually for any other type of PolyVector field.

2.6. MATRIX REPRESENTATION

Computing n -field splines amounts to solving linear systems. Before we derive the linear systems, we introduce the matrices representing the m -harmonic energies, the L^2 -scalar product and the n -vector field Laplacian. First step is to fix a coordinate system in the plane of every triangle. The matrices act on the vectors $\mathbf{u} \in \mathbb{R}^{2|\mathcal{F}|}$ listing the coordinates of the representation vectors of all triangles. We denote the matrix representing the harmonic energy by \mathbf{S} . The matrix is defined as the symmetric matrix that satisfies

$$E_H(\mathbf{u}) = \mathbf{u}^T \mathbf{S} \mathbf{u}$$

for all $\mathbf{u} \in \mathbb{R}^{2|\mathcal{F}|}$. Explicitly, \mathbf{S} is the $2|\mathcal{F}| \times 2|\mathcal{F}|$ matrix which consists of the following 2×2 blocks:

$$\begin{pmatrix} \mathbf{S}_{2i2j} & \mathbf{S}_{2i2j+1} \\ \mathbf{S}_{2i+12j} & \mathbf{S}_{2i+12j+1} \end{pmatrix} = -w_{ij} R_{ij} \text{ for } i \neq j$$

$$\begin{pmatrix} \mathbf{S}_{2i2i} & \mathbf{S}_{2i2i+1} \\ \mathbf{S}_{2i+12i} & \mathbf{S}_{2i+12i+1} \end{pmatrix} = \sum_{k \in \mathcal{N}(i)} w_{ik} \mathbf{I}.$$

The matrix \mathbf{M} representing the L^2 -scalar product is defined as the matrix that satisfies $\langle \mathbf{u}, \mathbf{u}' \rangle_{L^2} = \mathbf{u}^T \mathbf{M} \mathbf{u}'$ for all pairs $\mathbf{u}, \mathbf{u}' \in \mathbb{R}^{2|\mathcal{F}|}$. Explicitly, it is the diagonal $2|\mathcal{F}| \times 2|\mathcal{F}|$ matrix

²The most general constraint would force the sum of angular differences between the transported n -vectors to be equal to a value that depends on the chosen singularity index.

that repeats the area of each triangle twice. This matrix is often called the mass matrix. The matrices representing the Laplacian and the m -harmonic energies can be constructed as products of \mathbf{S} and \mathbf{M} . The matrix \mathbf{L} for the n -vector field Laplacian Δ is given by $\mathbf{L} = \mathbf{M}^{-1}\mathbf{S}$. The matrix \mathbf{B} representing the biharmonic energy is

$$\mathbf{B} = \mathbf{S}\mathbf{M}^{-1}\mathbf{S},$$

and, more generally, the matrices for the m -harmonic energies are: $\mathbf{S}(\mathbf{M}^{-1}\mathbf{S})^{m-1}$. The matrices representing the scalar product and the m -harmonic energies are positive definite.

For later usage, we also introduce the 2×2 matrix that performs a rotation by $\pi/2$ in the plane of a triangle. Since we use positively oriented orthonormal bases in the tangent planes, the rotations have the same matrix representation in all tangent planes:

$$J_i = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

2.7. COMPUTING n -FIELD SPLINES

We now state the linear systems whose solutions are field-aligned, interpolating, higher-order n -vector or n -direction field splines. In the following, let $I = i_1, \dots, i_m$ be the set of hard constrained vectors, $u_i = d_i$, let \mathbf{u}' be the specified alignment field and μ the alignment weight. \mathbf{d} is the vector that stacks real and imaginary parts of the constrained directions d_{i_1}, \dots, d_{i_m} .

n -vector fields Let \mathbf{F} be the $2m \times 2|\mathcal{F}|$ matrix for which $\mathbf{F}\mathbf{u} = \mathbf{d}$ is equivalent to $u_i = d_i$ for $i \in I$. Then the optimization problem (2.8) is equivalent to solving

$$\begin{pmatrix} \mathbf{B} + \mathbf{M} & \mathbf{F}^T \\ \mathbf{F} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\mathbf{u}' \\ \mathbf{d} \end{pmatrix} \quad (2.11)$$

where the vector $\boldsymbol{\lambda}$ stacks the $2m$ (rescaled) Lagrange multipliers.

n -direction fields We first modify the alignment field in order to enforce the correct directions in the hard constrained faces, as described before, *i.e.* we set $u'_i = s_i d_i$ for $i \in I$, while leaving the rest of the alignment field as is. Let \mathbf{D} be the $m \times 2|\mathcal{F}|$ matrix for which $\mathbf{D}\mathbf{u} = \mathbf{0}$ is equivalent to $\langle u_i, J_i d_i \rangle = 0$ for all $i \in I$. Then a minimizer of (2.9) can be found by solving the linear system

$$\begin{pmatrix} \mathbf{B} - \lambda \mathbf{M} & \mathbf{D}^T \\ \mathbf{D} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\mathbf{u}' \\ \mathbf{0} \end{pmatrix}, \quad (2.12)$$

where λ takes the role of the parameter μ and has to be chosen in the range $(-\infty, \hat{\lambda}_1^2)$, where smaller λ means higher alignment. $\hat{\lambda}_1$ is the smallest eigenvalue of the n -vector field bi-Laplacian Δ^2 restricted to fields which satisfy the hard constraints. This reformulation is akin to [1]. A proof and a more precise statement regarding $\hat{\lambda}_1$ is postponed to Section 2.B. The auxiliary variable $\boldsymbol{\gamma}$ stacks the m (rescaled) Lagrange multipliers. Once

Algorithm 1 Computation of n -direction field splines

Input: The mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$, the rotational symmetry index n , an alignment field \mathbf{u}' , the alignment parameter λ , the list of hard constraints d_i along with the indices of constrained triangles I

Output: n -direction field spline interpolating the hard constraints and aligning to \mathbf{u}'

First solve:

1. Modify the alignment field at the hard constraints: $\forall i \in I: u'_i \leftarrow s_i d_i$, for sufficiently large s_i .
2. Choose a basis for the tangent space in each triangle and compute the rotation matrices R_{ij} as described in Section 2.3.
3. Set up the matrices $\mathbf{S}, \mathbf{M}, \mathbf{L}$ and \mathbf{B} as described in Section 2.6 and the matrix \mathbf{D} which stacks the constraints $\langle u_i, J_i d_i \rangle = 0$.
4. Factorize the matrix $\mathbf{V}_\lambda = \mathbf{B} - \lambda \mathbf{M}$.
5. Solve $\mathbf{D} \mathbf{V}_\lambda^{-1} \mathbf{D}^T \boldsymbol{\gamma} = \mathbf{D} \mathbf{V}_\lambda^{-1} \mathbf{M} \mathbf{u}'$ by using the factorization above to compute the matrix $\mathbf{V}_\lambda^{-1} \mathbf{D}^T$ and the vector $\mathbf{V}_\lambda^{-1} \mathbf{M} \mathbf{u}'$.
6. Use the same factorization and $\boldsymbol{\gamma}$ from above to compute $\mathbf{w} = \mathbf{V}_\lambda^{-1} (\mathbf{M} \mathbf{u}' - \mathbf{D}^T \boldsymbol{\gamma})$.
7. The n -direction field spline \mathbf{u} is now given by $u_i = \frac{w_i}{\|w_i\|}$.

Updated constraints:

1. Only compute the columns $\mathbf{V}_\lambda^{-1} \mathbf{D}^T$ for the rows of \mathbf{D} that are new or updated.
 2. Recompute the multiplications $\mathbf{D} \mathbf{V}_\lambda^{-1} \mathbf{D}^T$ and $\mathbf{D} \mathbf{V}_\lambda^{-1} \mathbf{u}'$ and solve the two systems from steps (5) and (6) above.
 3. Output the n -direction field spline u as given in (7) above.
-

\mathbf{w} is computed, the desired n -direction field spline \mathbf{u} , which interpolates the directions d_i in triangles T_i and aligns to the field \mathbf{u}' , is obtained by normalizing all vectors of \mathbf{w} : $u_i = \frac{w_i}{\|w_i\|}$. The validity of this system and the relationship between λ and μ is discussed in Section 2.B.

For both, n -vector and n -direction fields, the constraints (2.10) to enforce singularities can be readily appended to the matrices \mathbf{D} and \mathbf{F} to solve for smoothest constrained n -vector or n -direction field splines respectively.

Implementation In our implementation the user can add and modify hard constraints by selecting a face and dragging an arrow to specify a desired direction. The optimal field is updated each frame depending on the current configuration of the constraints. This allows for instant feedback when trying to adjust hard constraints in the design of n -field splines. Note that when an existing constraint is modified, in the case of n -vector fields, only the right hand side changes, so the full system (2.11) can be factorized once and solved with the modified constraints. However, once new constraints are introduced, the matrix \mathbf{F} changes (new rows have to be added) and the matrix has to be refactorized. In the case of n -direction fields, the hard constraints are encoded in the matrix \mathbf{D} , so changing and adding hard constraints both would lead to refactorization. To avoid factorizing the large matrices in both systems every time constraints are added or modified, we reorganize the equations. We will demonstrate the reformulation on system (2.12), the steps to reformulate system (2.11) are identical.

Let $\mathbf{V}_\lambda = \mathbf{B} - \lambda\mathbf{M}$ and multiply the first $2|\mathcal{F}|$ rows of the system by $\mathbf{D}\mathbf{V}_\lambda^{-1}$ from the left, which amounts to the system

$$\begin{aligned} \mathbf{D}\mathbf{w} + \mathbf{D}\mathbf{V}_\lambda^{-1}\mathbf{D}^T\boldsymbol{\gamma} &= \mathbf{D}\mathbf{V}_\lambda^{-1}\mathbf{M}\mathbf{u}' \\ \mathbf{D}\mathbf{w} &= 0 \end{aligned}$$

and by subtracting the second set of equations from the first we get

$$\mathbf{D}\mathbf{V}_\lambda^{-1}\mathbf{D}^T\boldsymbol{\gamma} = \mathbf{D}\mathbf{V}_\lambda^{-1}\mathbf{M}\mathbf{u}' \quad (2.13)$$

where the left-hand side is still symmetric. We solve (2.13) for $\boldsymbol{\gamma}$ and then recover \mathbf{w} via

$$\mathbf{w} = \mathbf{V}_\lambda^{-1}(\mathbf{M}\mathbf{u}' - \mathbf{D}^T\boldsymbol{\gamma}). \quad (2.14)$$

The advantage of this is that we only have to factorize \mathbf{V}_λ once, and whenever new constraints are added or modified, we only have to solve the dense but very small ($2m \times 2m$) system (2.13). To set up the left hand side of (2.13) we also need to solve a linear system to obtain updates to those rows of $\mathbf{V}_\lambda^{-1}\mathbf{D}$, which have to be changed due to updated hard constraints. This can be done using the factorization of \mathbf{V}_λ . Note, that in practice hard constraints will not be edited simultaneously, such that only one column of \mathbf{D} is being modified. Then, since we need to update the alignment field when we edit hard constraints, we need to compute the right hand side $\mathbf{D}\mathbf{V}_\lambda^{-1}\mathbf{u}'$. After solving (2.13), we solve for \mathbf{w} in (2.14) using the same factorization. Timings when solving for a 4-direction field with 30 hard constraints can be taken from Table 2.1 (right-most column). Note that

those timings were computed as only one hard constraint has been modified. They include setting up and solving the dense system (2.13) and then recovering \mathbf{w} via (2.14). The time needed to factorize the matrix \mathbf{V}_λ is listed separately since this is done at the preprocessing stage and only needs to be repeated when the parameter λ is changed. As can be seen, for small meshes this reformulation may be used in a real-time editing setting, however, for larger meshes (or in case the parameter λ needs to be changed interactively, in order to control the strength of alignment), we propose a change of basis, under which the matrix \mathbf{V}_λ diagonalizes. This will be described in the following section.

Algorithm 1 summarizes the computation of n -direction field splines.

2.8. REAL-TIME EDITING

In this section we describe our model reduction approach that allows for computing n -field splines in real-time, independently of the mesh resolution. The benefit of using the fast computation is that it ensures a fluid interaction and allows users to immediately get feedback on how the constraints they have placed affect their design. As a trade-off to a faster computation during a modeling session, a pre-computation has to be performed, in which a basis for n -vector fields is computed. Timings for both pre-computation and the reduced/unreduced solves for the n -field spline system are listed in Table 2.1 and discussed in Section 2.9.

Using the eigenbasis of the n -vector field Laplacian Δ , we obtain a Fourier-type decomposition of face-based, piecewise constant n -vector fields that associates a frequency spectrum to a n -field. Unfortunately, as opposed to the basis from the previous chapter, we do not get a natural division into some analogue of harmonic, divergence- or curl-free n -fields. There, we were able to compute the eigenfields as gradients and co-gradients of two different discretizations of the surface function Laplace–Beltrami operator. It is unclear how to extend such a methodology to n -fields.

Here, the eigenfields and spectrum of Δ are the solutions of the generalized eigenproblem

$$\mathbf{S}\mathbf{u} = \lambda\mathbf{M}\mathbf{u}. \quad (2.15)$$

Since Δ is self-adjoint, all eigenvalues are real and there exists an L^2 -orthonormal eigenbasis.

The eigendecomposition enables the design of spectral processing tools, such as compression and spectral filtering, for n -fields. For example, projecting a n -field to the subspace spanned by the k eigenfields with the lowest eigenvalues, is a low-pass filter for the n -field. Here, we will not explore this direction—but use the low-frequency eigenfields to derive a reduced-order scheme for the fast approximation of n -field splines.

The computation is split in an offline and an online stage. In the offline stage, the first k eigenfields are computed and the relevant matrices are constructed. In the online stage, the computations are restricted to the subspace spanned by the eigenfields. This results in a reduced computational burden in the online phase and enables real-time computation and interactive modeling of n -field splines for larger meshes.

In the following, we discuss the reduction of equations (2.11) and (2.12), which yields fast approximation algorithms for n -vector and n -direction field splines. Let \mathbf{U} be the $2|\mathcal{F}| \times k$ matrix which stacks the first k eigenvectors as its columns and let λ_i denote the

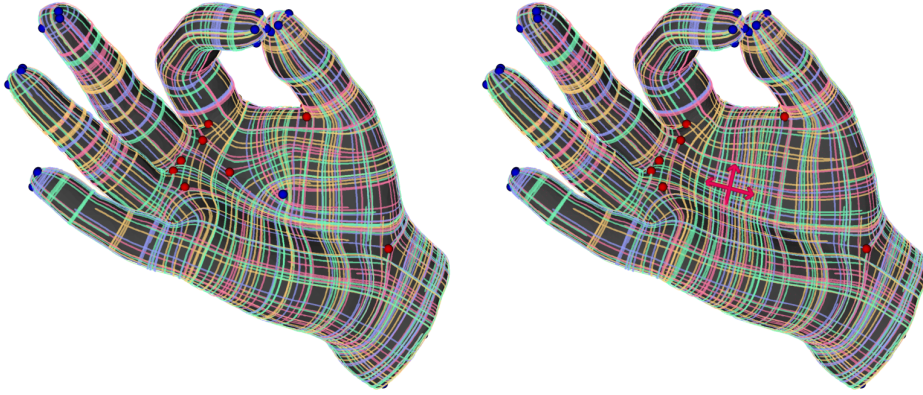


Figure 2.3: The smoothest, curvature aligned ($\lambda = -0.2$), 4-direction field on the hand (left) is intuitively modified by merging two singularities using a hard constraint (right).

first k eigenvalues. Any n -field $\mathbf{w}_{\mathbf{U}}$ that is in the subspace spanned by \mathbf{U} can be describe by reduced coordinates \mathbf{w} , which are given by $\mathbf{w}_{\mathbf{U}} = \mathbf{U}^T \mathbf{w}$. Since the eigenfields are \mathbf{L}^2 -orthonormal, $\mathbf{U}^T \mathbf{M} \mathbf{U} = \mathbf{I}$, we have

$$\mathbf{U}^T \mathbf{V}_{\lambda} \mathbf{U} = \mathbf{U}^T (\mathbf{B} - \lambda \mathbf{M}) \mathbf{U} = \text{diag}(\lambda_i^2 - \lambda) =: \Lambda_{\lambda}. \quad (2.16)$$

This means that solving the sparse $2|\mathcal{F}| \times 2|\mathcal{F}|$ system (2.14) to obtain a smooth n -vector field in the unreduced case (which has to be done at least three times when constraints are updated) is being replaced by the diagonal system $\Lambda_{\lambda} \mathbf{x}_{\mathbf{U}} = \mathbf{w}_{\mathbf{U}}$, which can be solved by k multiplications with precomputed numbers. The reduced version of (2.13) is

$$\mathbf{D}_{\mathbf{U}} \Lambda_{\lambda}^{-1} \mathbf{D}_{\mathbf{U}}^T \boldsymbol{\gamma} = \mathbf{D}_{\mathbf{U}} \Lambda_{\lambda}^{-1} \mathbf{M}_{\mathbf{U}} \mathbf{u}'_{\mathbf{U}}, \quad (2.17)$$

where $\mathbf{u}'_{\mathbf{U}} = \mathbf{U}^T \mathbf{u}'$, $\mathbf{D}_{\mathbf{U}} = \mathbf{U}^T \mathbf{D} \mathbf{U}$ and $\mathbf{M}_{\mathbf{U}} = \mathbf{U}^T \mathbf{M} \mathbf{U}$, which can be updated efficiently when \mathbf{D} changes since it is extremely sparse. Finally, $\mathbf{w}_{\mathbf{U}}$ can be recovered via

$$\mathbf{w}_{\mathbf{U}} = \Lambda_{\lambda}^{-1} (\mathbf{M}_{\mathbf{U}} \mathbf{u}'_{\mathbf{U}} + \mathbf{D}_{\mathbf{U}}^T \boldsymbol{\gamma}). \quad (2.18)$$

Using this reduction, the cost for solving for n -field splines is independent of the resolution of the meshes, aside from multiplications with \mathbf{U} . By storing \mathbf{U} on a GPU and computing multiplications there, we computed n -field splines in real time for meshes with 300k triangles in our experiments (see Table 2.1).

2.9. EXPERIMENTS

Real-time n -field spline editor Using the higher-order energies, together with the reformulations of the involved linear system and spectral reduction described in the previous sections, we implemented a tool for real-time editing of n -field splines. It allows users to click and drag on a mesh to insert hard constraints and specify their direction, while the field and its visualization are updated in real-time. As a visualization we chose to draw lines along the surface that follow one of the n directions along the n -field. To

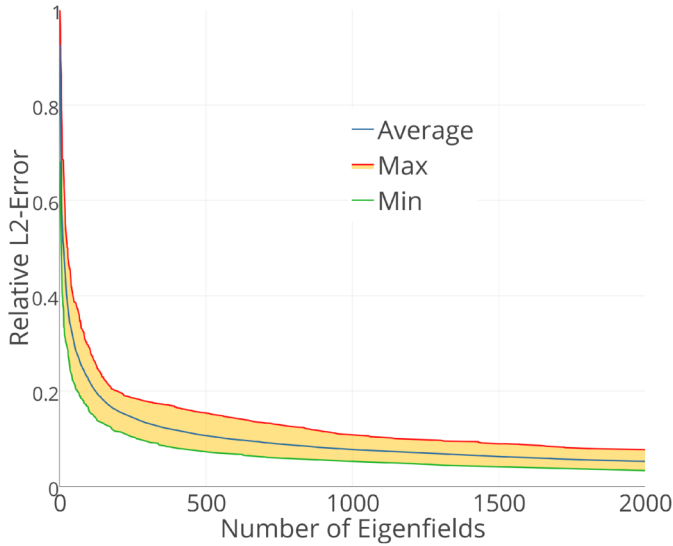


Figure 2.4: Plots of the average, minimal and maximal relative L^2 -error when comparing the reduced solution (2.12) to the full solution (2.17) of a 4-direction field on the bunny mesh from ten randomized constraints (we generated twenty tests) using a rising number of eigenfields.

keep the visualization consistent, we initialize seed points for those lines once and re-integrate them whenever the n -field spline has changed. All examples that are shown in this chapter have been created using this tool. In our experience, the precise control offered via hard constraints, combined with the instant feedback due to the reductions offer an intuitive, effective and persistent way of n -field editing that has many advantages over previous methods. A feature of the propose technique is that the output depends continuously on the constraints, which allows users to slightly perturb constraints in order to optimize features of the field on a small scale in an intuitive way. Below we discuss various aspects of the editing framework described in this chapter along with comparisons to previous work.

Basic examples In Figures 2.3, left, 2.5 and 2.12, top row, we show curvature aligned, smoothest 4-direction fields. Singularities, which are automatically optimized and not user input or achieved via integer variables, appear in desirable places, such as cube-like corners (right, upper part of the rocker arm in Figure 2.12) or at the end of extremities, in groups of four (see Figure 2.3 at the end of the fingers), which allows for the flow lines to follow the features until forming a plateau at the end. The results strongly resemble those obtained from the formulation of Knöppel et al. [1], which is formulated for vertex-based fields. Different from their approach, we allow for higher order energies and hard constraints, *i.e.* n -field splines.

In Figure 2.6, right, we show how this can be used to edit small scale details on a high resolution mesh with exact control over the alignment. When using soft constraints (*i.e.* only an alignment field) to control the layout of the field, we were unable to en-

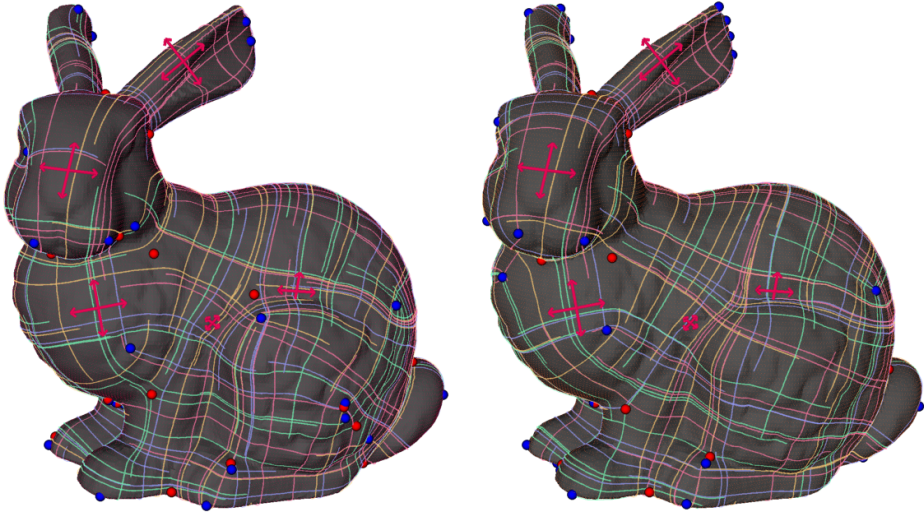


Figure 2.5: Visual comparison of the smoothest, curvature aligned ($\lambda = -0.05$), 4-direction field on the bunny, subject to five hard constraints (pink crosses), when computing it using the full system (2.12) (left) and the reduced system (2.17) (right), using a basis of 500 eigenfields. The fields possess 72 and 68 singularities respectively.

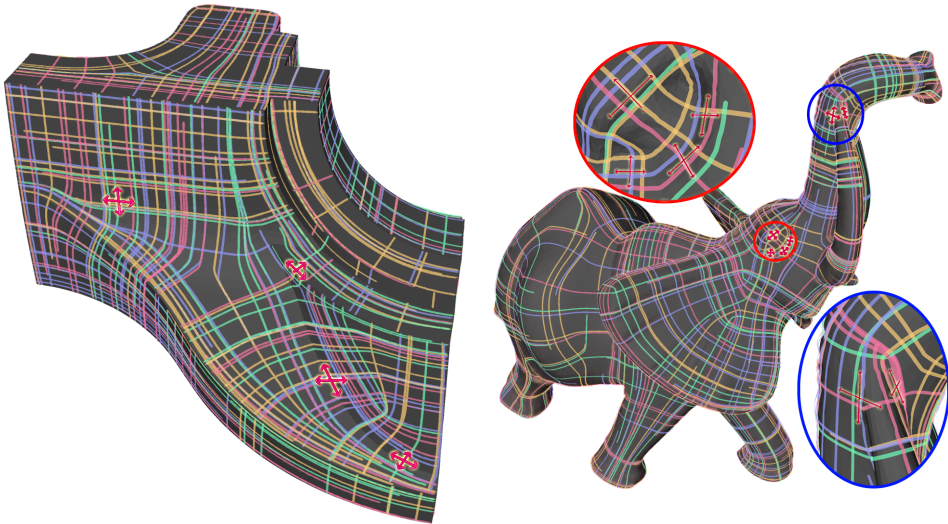


Figure 2.6: Left: By adding four hard constraints, we are able to force the smoothest, curvature aligned ($\lambda = -0.5$) 4-field to align to features that are not following the principle curvature directions. Right: Hard constraints enable precise editing at small scale: the insets show how hard constraints can be used to align to small features on a high resolution mesh with fine details. Such precise control can not be achieved via least-squares alignment terms but requires hard constraints.

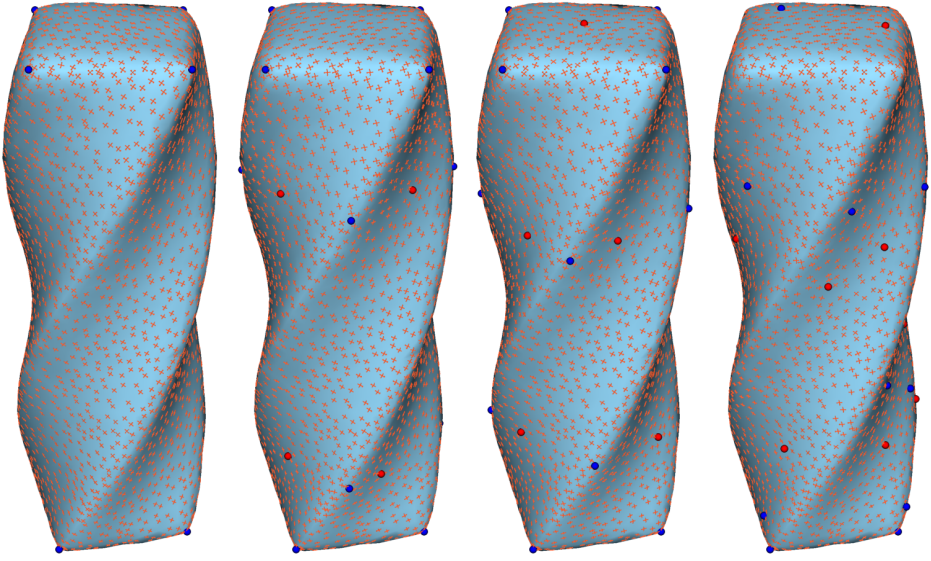


Figure 2.7: Eigenvectors of the 4-field Laplacian. From left to right: 1st, 6th, 14th and 110th eigenfield. Singularities with indices $-\frac{1}{4}$ and $\frac{1}{4}$ are marked in red and blue respectively.

force alignment satisfactorily, especially in areas where conflicting constraints are close to each other: either one of the conflicting constraints dominates the shape of the field in that region, or the field averages the influence of the nearby constraints. When using hard constraints, we can easily force the field to point in various directions (even for directly adjacent triangles) and still obtain a smooth solution for the rest of the field. In Figure 2.6, left we edit the smoothest, curvature aligned 4-field by adding three hard constraints which force the flow lines to follow two features that do **not** follow the principal curvature directions. It is intuitive to move, merge or produce new singularities by placing appropriate hard constraints, a simple example for this is shown in Figure 2.3. Direct control over the placement of singularities at desired vertices is also possible and is treated in Section 2.5 and demonstrated in an experiment further below.

In Figure 2.7 we show some of the eigenfields of the 4-field Laplacian, Δ , which are used as a reduced basis for interactive editing. It is remarkable how the singularities are laid out symmetrically and appear consistently (for the first 108 eigenfields) at the corners of the twisted, rounded bar. This indicates that a process that forces the field not to place singularities at the corners will result in less smooth fields. Therefore, we think that the pure number of singularities of field is not a reliable indicator for the smoothness of a field.

The effects of using a reduced basis are shown in Figure 2.5 where the smoothest, curvature aligned 4-field, altered by adding five hard constraints, is shown as an optimal solution of the unreduced (left) and reduced (right) system. In the reduced case a basis of the 500 lowest frequency eigenfields was constructed. While the general layout of the field remains the same, some singularities moved and merged, which is due to the fact

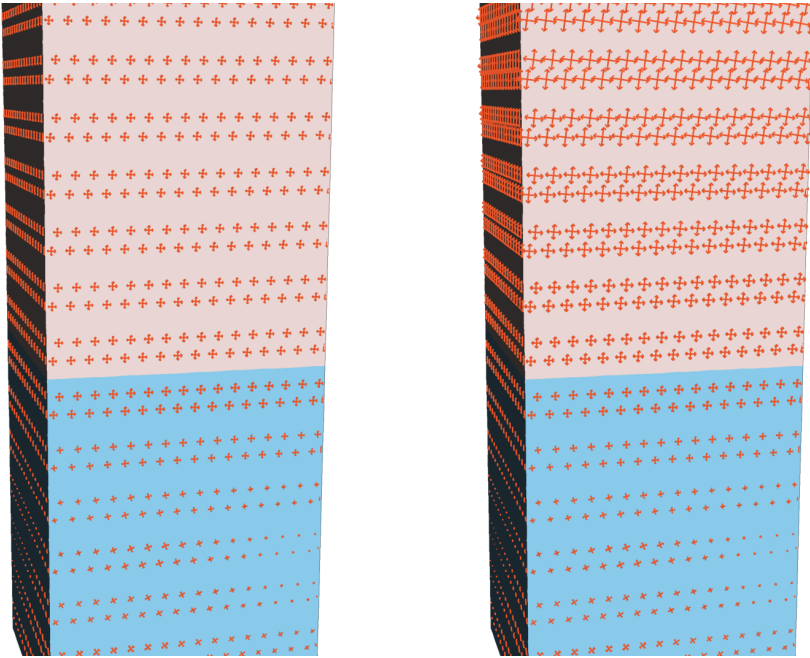


Figure 2.8: Comparing the harmonic (left) and biharmonic (right) energies in presence of boundary constraints: the blue part of the field was fixed and the rest of the field computed as smooth as possible, as defined through the different energies. As can be seen, the higher order energy continues the clockwise rotation and increase in size (*i.e.* the first derivative remains continuous), while the harmonic energy only ensures continuity of the field and thus remains constant.

that the hard constraints and curvature alignment introduced high-frequency features, which are not contained in the reduced basis.

As a numerical comparison, we generated twenty different 4-direction fields from a random set of twenty hard constraints on the bunny mesh using the unreduced and reduced systems and kept track of the relative L^2 distance between both solutions when using between one and 2000 eigenfields in the reduced case. In Figure 2.4 we plot the minimal, average and maximal L^2 distances between each reduced and unreduced field generated this way. The number of eigenfields that should be used depends on the application. In order to keep the relative distance to a full solution consistently below 10%, at least 1000 eigenfields should be used. We found, however, that about 500 eigenfields are enough to achieve fields that are visually hard to distinguish from the unreduced solution. Thus the proposed model reduction is well-suited as a technique that enables interactive modeling of n -fields on larger meshes.

n -field splines By introducing higher order smoothness energies for n -fields, we get the guarantee of continuous differentiability at constraints. To highlight this, we show an artificial example in Figure 2.8, where we constrain a complete region of the 4-vector field (lower part, in blue), while solving for the smoothest field under these constraints

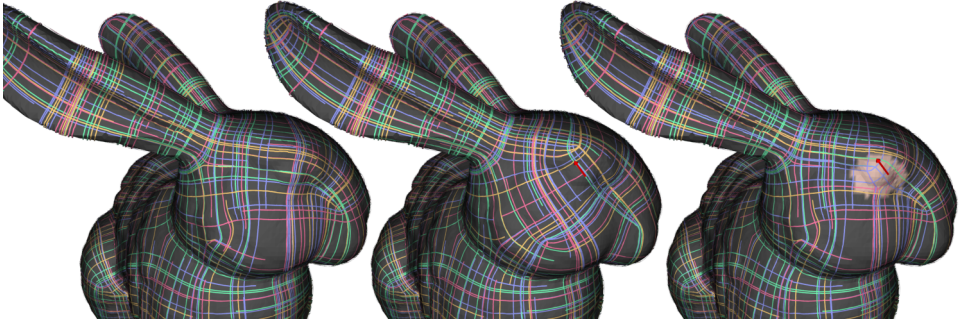


Figure 2.9: Local editing on the bunny mesh: to the smoothest (biharmonic energy), curvature aligned, 4-direction field (left) we want to add a hard constraint to achieve diagonal quads around the eye. However, the constraint has an undesired non-local effect (middle). This can be remedied by using our local editing scheme, which allows for local changes to the field, while still yielding differentiable results (right).

using the harmonic energy E_H , left, and the biharmonic energy E_B , right. As can be seen, using the harmonic energy results in a constant field on the upper half. The tendency of the field in the lower half to grow in magnitude and the clockwise rotation are not continued, which corresponds to a discontinuity in the derivative in the continuous setting. In fact, changing the blue part of the field does not have any effect on the variable, upper part, as long as the vectors in the boundary triangles remain the same. Using the biharmonic energy smoothly continues the tendencies of the lower field.

This enables us to perform tasks such as local editing, see Figure 2.9: since constraints are interpolated in a differentiable manner, it is valid to constrain the field on the whole mesh minus a small selected region. This is relevant in practice since hard and soft constraints have a global influence on the layout of the field, which is undesirable when editing small features in isolated regions. For computational efficiency, one can reduce the size of the system to be solved by discarding vectors that are constrained and do not affect the unconstrained vectors. Explicitly, the vectors of triangles that are not in the two-ring of any unconstrained triangle need not be included. Here two-ring refers to the dual graph in which the triangles are the nodes and nodes are connected if triangles share an edge. In this sense, the two-ring around the unconstrained triangles specifies boundary conditions which determine the solution in the unconstrained area.

In Figure 2.2 we show n -direction field splines where we prescribed the placement of four index $\frac{1}{2}$ (yellow) and one index $-\frac{1}{2}$ singularities in the field as described in Section 2.5. The main observation is that these types of hard constraints are not well posed under the harmonic energy. The constraints only affect the vectors in the one-rings around the placed singularities. Since a one-ring becomes smaller and smaller under refinement, the solutions converges to a discontinuous field. This is illustrated in the insets in the lower row, where the same hard constraints were used in a low and high resolution version of the hand mesh. Using the biharmonic energy leads to a smooth solution, where the singularity constraints have a global effect and the solution remains consistent under refinement. This is another example for the importance of higher order smoothness energies when posing hard constraints.

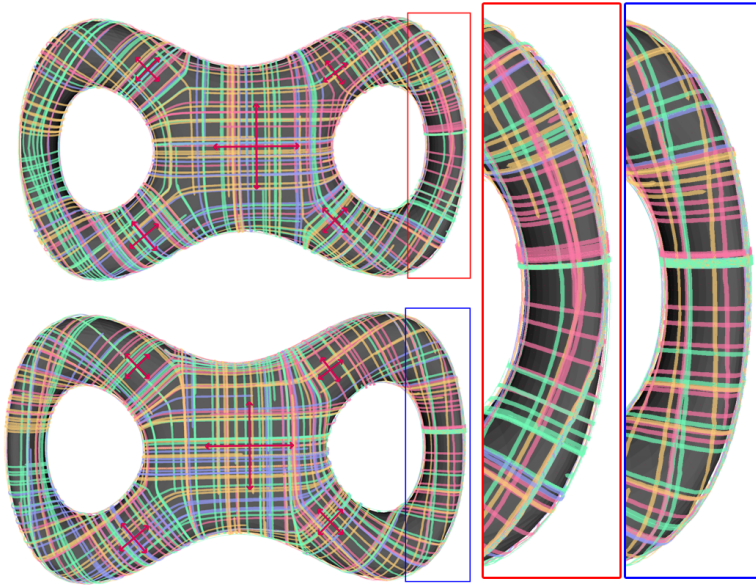


Figure 2.10: Comparing the smoothest 4-direction fields on the double torus, in the presence of five hard constraints, when using 4-field splines (top) or when using the harmonic energy (bottom). As can be seen in the highlighted areas, the constraints are interpolated in a differentiable manner when using the biharmonic energy.

The tendency to interpolate constraints in a differentiable manner when using n -field splines can also be seen in Figure 2.10, where the curvature of the flow lines near the constraints is continued in the regions on the side, away from any constraints, which is not the case when using the common harmonic energy. Note that in this example we did not use curvature alignment.

Timings In Table 2.1 we list the timings taken to solve the reduced and unreduced systems to compute smoothest n -fields subject to 30 hard constraints, using the harmonic energy and biharmonic energy. We constructed 4-direction fields, and used the 500 lowest frequency eigenfields of the corresponding Laplacian as a reduced basis. The timings to construct those bases is listed as well. When measuring the computation time, only one of the 30 hard constraints was assumed to have changed in the last frame, such that only specific parts of the system have to be set up and updated. The reduction leads to computation timings that are neither influenced by the resolution of the mesh, nor by the order of the energy. The field can be updated about 200 times per second, such that the visualization of the field and display of the mesh become the bottle-neck for real-time n -vector field editing. In the unreduced case, we obtain computation times of more than 100 ms for meshes with 70k triangles in our experiments. Times for adding or removing new constraints are even longer. The trade-off for the fast computation times in the reduced scheme is that the eigenfields need to be precomputed, which takes about 3 minutes for meshes with 70k triangles in our experiments.

Table 2.1: Timings for the computation of smoothest, 4-direction fields without and with using the proposed spectral reduction. In all examples we added 30 random hard constraints. We state timings for both the harmonic and biharmonic smoothness energy. In the reduced case, we construct a basis from the 500 lowest frequency eigenfields.

Model Name	#faces	Bases setup	Reduced Setup & Solve harmonic \ biharmonic	Factorization of $\Delta \setminus \Delta^2$	Unreduced Solve harmonic \ biharmonic
Twisted Bar	3276	5s	5ms \ 5ms	40ms \ 83ms	6ms \ 6ms
Hand	12184	24s	5ms \ 5ms	161ms \ 393ms	16ms \ 23ms
Rocker Arm	20088	44s	5ms \ 5ms	267ms \ 654ms	28ms \ 37ms
Bunny	69666	170s	6ms \ 6ms	1318ms \ 4413ms	138ms \ 194ms
Elephant	79946	181s	6ms \ 6ms	1249ms \ 3112ms	155ms \ 182ms
Armadillo	331904	15m30s	8ms \ 8ms	8155ms \ 28s	598ms \ 801ms

Comparison to Instant Field Aligned Meshes Jakob et al. [26] introduced a scheme for computing smooth n -fields via a local averaging scheme on a multiresolution hierarchy and parallelization in very low computation times. Since this is the only technique that results in comparable timings, we want to put our framework into contrast.

The techniques used in their approach are fundamentally different from the methods we propose. Our fields are globally optimal and are solutions of linear systems, while [26] use parallelization, a multiresolution hierarchy and local operations to iteratively optimize the fields. Their approach does not iterate until a minimum of an objective is found, but terminates after a number of local smoothing steps on each level of the hierarchy have been performed. In our experiments with their approach, we detected multiple drawbacks resulting from this strategy when compared to our approach. Their results are heavily affected by a changing the triangulation of a surface: Figure 2.12 shows the smoothest, curvature aligned 4-fields produced with our technique (top row) and the smoothest fields produced by Jakob et al. on three different triangulations of the rocker arm mesh. While our singularity layout remains consistent (only some pairs of very close singularities merge and cancel each other), the fields produced by Jakob et al. possess inconsistent and globally different singularity layouts.

In [26], alignment to a globally defined field, such as principal curvature directions, is handled via linearly interpolating the current n -vectors with the vectors of the alignment field in every optimization step. Since this has to be done on every level of the multiresolution hierarchy, the constraint vectors have to be propagated along the hierarchy as well, by iteratively merging the constraint vectors. It is not clear whether such an optimization scheme minimizes a smoothness/alignment energy such as (2.9). As an alternative to achieve curvature alignment, Jakob et al. propose to minimize an extrinsic smoothness energy by averaging the n -vectors in 3D world coordinates. However, this approach does not allow for a weight between smoothness and alignment. In Figure 2.11, we show a comparison between curvature aligned fields using our framework with the biharmonic energy and different weights for λ and fields using the framework of [26], where we show both the extrinsically smooth field and fields produced by using the alignment method described above, using different interpolation weights. While the extrinsically smooth field has comparable quality to our field for $\lambda = 0.0001$, it is not aligned to many features of the elephant, such as the eye, or the right part of the visible ear. There is no way to enhance the alignment when using the extrinsic energy. Us-

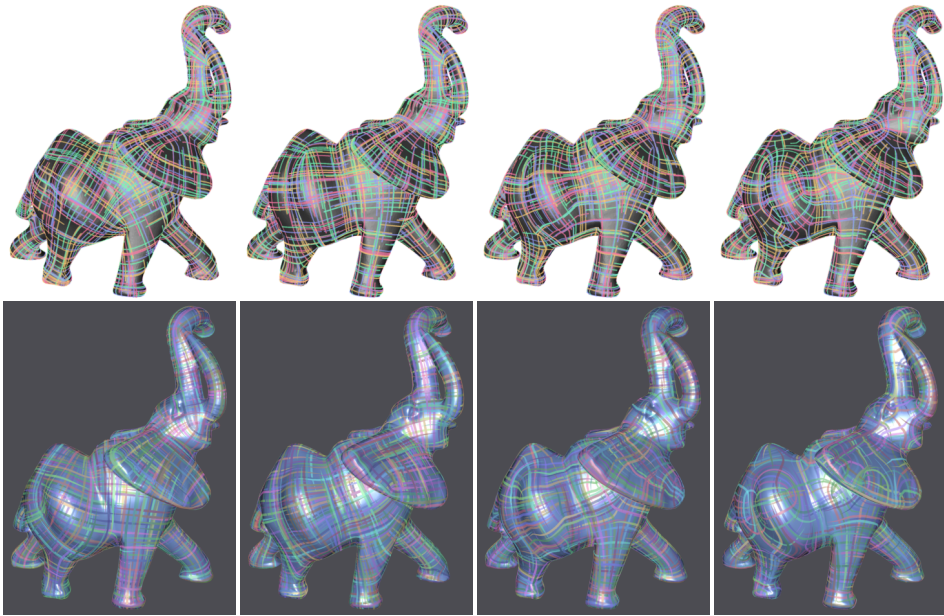


Figure 2.11: Curvature alignment using our framework (top row) and [26] (bottom row). We used the bihamonic energy and, from left to right, $\lambda = 0.0001, 0, -0.005, -0.1$. Bottom row, left, shows the result when using the extrinsic energy, and then, from left to right, alignment via local averaging using interpolation weights of 0.01, 0.1 and 0.5.

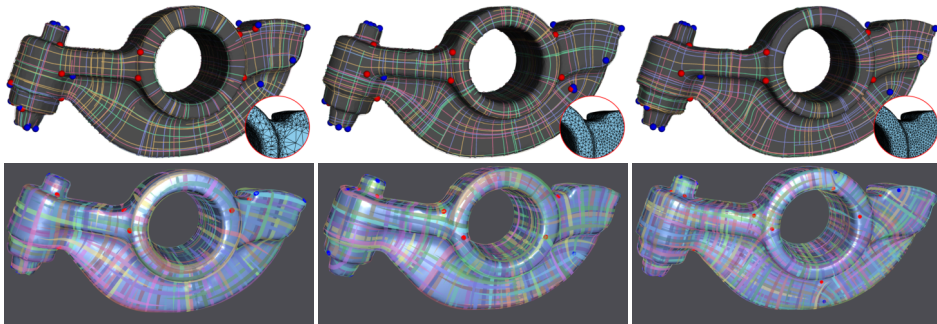


Figure 2.12: Top row: The smoothest curvature aligned 4-fields on three different triangulations, computed using our system. The fields possess 51, 40 and 42 singularities, respectively, different numbers mostly resulting from merged groups of nearby singularities. Bottom row: The smoothest 4-fields, using the extrinsic energy and the system proposed by Jakob et al. [26]. The fields possess 22, 30 and 42 singularities, respectively, being arranged in inconsistent structures.

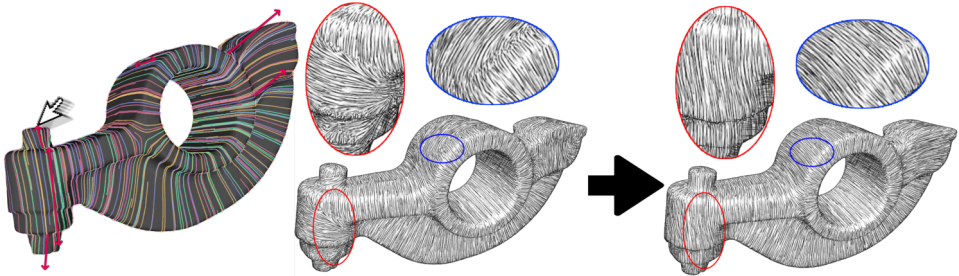


Figure 2.13: **Left:** Screenshot of a modeling session of a 2-field spline using hard constraints. The constraints are edited via dragging the arrows and the field is updated and visualized at 60fps. **Middle:** Non-photorealistic rendering of the rocker arm mesh when using the *unedited* smoothest 2-field aligned to the maximal principal curvature. **Right:** Hatching rendering of the *edited* 2-field spline subject to six hard constraints.

ing the local averaging scheme to align to the principal curvature directions produces visibly unsmooth fields, when the interpolation weights are high enough to guarantee alignment to the aforementioned features. Our technique maintains global smoothness while aligning to more features when using higher values of λ .

Finally, we want to remark that since there are no guarantees on the optimality of solutions and because of the necessity to merge n -vectors when navigating the multi-resolution hierarchy in [26], one cannot expect a continuous dependence of the solutions on the constraints. This is a key feature of a modeling tool. In our framework, slightly changing the constraints leads to small changes in the field.

2.10. APPLICATIONS

In the following we will describe two applications that strongly benefit from our ability to compute n -field splines in real-time in order to aid with artistic tasks, which require immediate feedback and smooth alignment to user defined hard constraints.

Hatching Rendering a surface mesh to resemble a line drawing, often called hatching, is a popular technique for stylizing 3D scenes, cf. [50–54]. In order to perform such a rendering, directions on the surface have to be chosen, along which the lines can be drawn. Often, as in [50, 51], these directions are somehow extracted from the principal curvature directions. Principal curvature directions are not unique in flat or umbilic regions, so directions in such regions need to be post-processed. Treating principal curvature directions as direction fields, allows to smooth the directions and to impaint directions in umbilic regions. For such smoothing processes, the principal curvature directions can be treated as 4-direction fields, see [24]. However, for guiding stroke directions, we opt for 2-direction fields, since it is the natural choice for the regions in which strokes are being drawn in only one direction and not orthogonally.

Using the principal curvature directions as the stroke directions is just one of many choices and often it is desirable to design stroke directions from scratch or modify existing directions, for example to hide singularities in highly lit or occluded places. This leads to a direct application of our real-time framework for the design maximal curvature

aligned 2-direction field splines, which can be enhanced via hard constraints. To illustrate this we implemented a real-time hatching system based on [51], which uses tonal art maps to texture surfaces according to light intensity and direction information. We enable the user to design a 2-direction field using our standard set of tools and directly update the directional information for the hatching application in real-time, which provides the user with an instant result. In Figure 2.13 we show a comparison between the 2-direction field aligned to the maximal principal curvature directions and an edited 2-direction field spline subject to six hard constraints. It can be seen how alignment to curvature directions is not suitable in regions where many singularities are present or the maximal/minimal directions suddenly exchange their roles. We provide the ability to intuitively edit the field in such regions in order to enhance the quality of the rendering.

Anisotropic BRDF In physically-based rendering systems one describes the reflection behavior of an object by using a *bidirectional reflectance distribution function* (BRDF) [55]. The BRDF at a given point on a surface depends on the direction towards the light and the direction towards the viewer and returns the color and intensity of the perceived light. Many such functions can be designed and they result in different perceptions of an object's material. Often, materials can be considered isotropic, meaning that their value does only depend on the angle between viewing/light direction and the surface normal. However, for anisotropic materials, like brushed metals, parameters have to be defined across the surface which define the orientation dependent response of the material to light. In Ashikhmin et al. [56] a BRDF model is proposed that takes (amongst others) two parameters n_u and n_v , which describe the anisotropy of the light in the local coordinate frame (u, v) . By designing a 2-direction field, one can specify the two directions of anisotropy by setting them to the direction of the field and its orthogonal direction respectively, thus describing the local coordinate frames up to the sign (to which the reflectance is indifferent).

Thus, using our framework for designing 2-direction field splines, provides control over the reflectance properties of such materials. Again, the real time response to newly imposed or modified hard constraints enables a direct feedback for the user, which is important for performing artistic tasks. A screenshot from an anisotropic BRDF design session using our framework is shown in Figure 2.14. There, we show a simple example to visualize what kind of effect we are aiming at.

Alternatively one can directly control the shape of the highlights under a fixed viewing and light direction, as proposed in Raymond et al. [13]. There, a relation between the BRDFs and the shape of the highlights is established, such that desired tangent directions of potential highlights can be specified by the user and then the BRDFs are optimized to fit these highlights. In [13] several tools to design a field with these prescribed tangent directions are offered. They are based on direct manipulation of a potential highlight, thus do not take into account the geometrical structure of such 2-direction fields and neglect the global structure of the field. Our tools offer an intuitive alternative that allows for the design of globally smooth tangent directions of potential highlights.

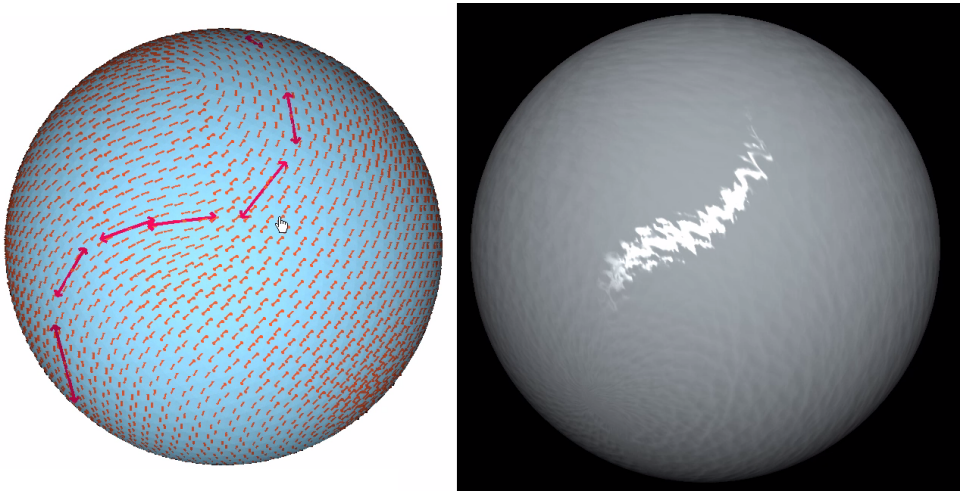


Figure 2.14: Editing BRDFs to manipulate highlights on a golf-ball. The BRDF parameters are consistently and smoothly defined over the whole mesh as they are extracted from the 2-field spline, and so, highlights will be consistent and smooth even when view and light directions change.

2.11. CONCLUSION

In this chapter, we introduced n -field splines: an approach for modeling tangential n -vector and n -direction fields on surfaces. The approach enables modeling of n -fields with hard interpolation and soft alignment constraints, placing singularities and local editing. New fairness energies for n -fields, a biharmonic energy, forms the basis of our approach. The energies are convex and quadratic so that n -vector as well as n -direction field splines can be computed by solving sparse linear systems. Based on a spectral decomposition of n -vector fields, we derive a reduced optimization scheme for computing n -field splines in real-time. We apply our approach to controlling and editing of stroke directions in line-art renderings and the modeling of anisotropic BRDFs on surfaces.

Limitations and challenges The discrete harmonic energy we introduce has one conceptual limitation, which it shares with the discrete harmonic energies for n -vector fields introduced in previous work. Only the trivial field is in its kernel. For surfaces of non-trivial genus g , there should be a $2g$ -dimensional kernel of “discrete harmonic” n -vector fields. For piecewise constant vector fields (1-fields) such structure-preserving discretizations are known, however, it remains a challenge to find such a structure-preserving discretizations for n -vector fields.

A limitation of our current implementation of the n -vector field modeling tool is that the local editing is not integrated with the global editing. We use the local editing as a last step in the modeling pipeline and cannot switch back to global editing after local edits have been performed (without constraining the whole local region). Various ways to address this problem are possible. However, it is not clear which one is the simplest and most effective.

Another interesting direction of future work is to use the n -vector field splines for

quadrilateral or hexagonal meshing.

REFERENCES

- [1] F. Knöppel, K. Crane, U. Pinkall, and P. Schröder, *Globally optimal direction fields*, ACM Trans. Graph. **32**, 59:1 (2013).
- [2] B. Liu, Y. Tong, F. D. Goes, and M. Desbrun, *Discrete connection and covariant derivative for vector field analysis and design*, ACM Trans. Graph. **35**, 23:1 (2016).
- [3] H.-C. Ebke, P. Schmidt, M. Campen, and L. Kobbelt, *Interactively controlled quad remeshing of high resolution 3D models*, ACM Trans. Graph. **35** (2016).
- [4] E. Praun, A. Finkelstein, and H. Hoppe, *Lapped textures*, in *Proc. ACM SIGGRAPH* (2000) pp. 465–470.
- [5] G. Turk, *Texture synthesis on surfaces*, in *Proc. ACM SIGGRAPH* (2001) pp. 347–354.
- [6] L.-Y. Wei and M. Levoy, *Texture synthesis over arbitrary manifold surfaces*, in *Proc. ACM SIGGRAPH* (2001).
- [7] M. Chi, C. Yao, E. Zhang, and T. Lee, *Optical illusion shape texturing using repeated asymmetric patterns*, The Visual Computer **30** (2014).
- [8] F. Knöppel, K. Crane, U. Pinkall, and P. Schröder, *Stripe patterns on surfaces*, ACM Trans. Graph. **34** (2015).
- [9] A. Hertzmann and D. Zorin, *Illustrating smooth surfaces*, in *Proc. ACM SIGGRAPH* (2000).
- [10] E. Zhang, J. Hays, and G. Turk, *Interactive tensor field design and visualization on surfaces*, IEEE Trans. Vis. Comp. Graph. **13**, 94 (2007).
- [11] C.-Y. Yao, M.-T. Chi, T.-Y. Lee, and T. Ju, *Region-based line field design using harmonic functions*, IEEE Trans. Vis. Comp. Graph. **18** (2012).
- [12] S. U. Mehta, R. Ramamoorthi, M. Meyer, and C. Hery, *Analytic tangent irradiance environment maps for anisotropic surfaces*, Comput. Graph. Forum **31** (2012).
- [13] B. Raymond, G. Guennebaud, P. Barla, R. Pacanowski, and X. Granier, *Optimizing brdf orientations for the manipulation of anisotropic highlights*, Comput. Graph. Forum **33**, 313 (2014).
- [14] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez, *Periodic global parameterization*, ACM Trans. Graph. **25**, 1460 (2006).
- [15] F. Kälberer, M. Nieser, and K. Polthier, *Quadcover - surface parameterization using branched coverings*, Comput. Graph. Forum **26** (2007).
- [16] D. Bommes, H. Zimmer, and L. Kobbelt, *Mixed-integer quadrangulation*, ACM Trans. Graph. **28**, 77:1 (2009).

- [17] E. Li, B. Lévy, X. Zhang, W. Che, W. Dong, and J.-C. Paul, *Meshless quadrangulation by global parameterization*. *Computers & Graphics* (2011).
- [18] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, and P. Cignoni, *Simple quad domains for field aligned mesh parametrization*, *Proc. SIGGRAPH Asia 2011* **30** (2011).
- [19] M. Nieser, J. Palacios, K. Polthier, and E. Zhang, *Hexagonal global parameterization of arbitrary surfaces*, *IEEE Trans. Vis. and Comp. Graph.* **18**, 865 (2012).
- [20] J. Solomon, M. Ben-Chen, A. Butscher, and L. Guibas, *Discovery of intrinsic primitives on triangle meshes*, (2011) pp. 365–374.
- [21] Y. Zhuang, M. Zou, N. Carr, and T. Ju, *Anisotropic geodesics for live-wire mesh segmentation*, *Comput. Graph. Forum* **33** (2014).
- [22] F. do Goes, M. Desbrun, and Y. Tong, *Vector field processing on triangle meshes*, in *SIGGRAPH Asia 2015 Courses* (2015) pp. 17:1–17:48.
- [23] A. Vaxman, M. Campen, O. Diamanti, D. Panozzo, D. Bommers, K. Hildebrandt, and M. Ben-Chen, *Directional field synthesis, design, and processing*, *Comput. Graph. Forum* **35**, 545 (2016).
- [24] J. Palacios and E. Zhang, *Rotational symmetry field design on surfaces*, *ACM Trans. Graph.* **26** (2007).
- [25] N. Ray, B. Vallet, W. C. Li, and B. Lévy, *N-symmetry direction field design*, *ACM Trans. Graph.* **27** (2008).
- [26] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung, *Instant field-aligned meshes*, *ACM Trans. Graph.* **34**, 189:1 (2015).
- [27] Z. Huang and T. Ju, *Extrinsically smooth direction fields*, *Computers & Graphics* **58**, 109 (2016).
- [28] O. Diamanti, A. Vaxman, D. Panozzo, and O. Sorkine-Hornung, *Designing n-PolyVector fields with complex polynomials*, *Comput. Graph. Forum* **33**, 1 (2014).
- [29] O. Diamanti, A. Vaxman, D. Panozzo, and O. Sorkine-Hornung, *Integrable PolyVector fields*, *ACM Trans. Graph.* **34**, 38:1 (2015).
- [30] E. Zhang, K. Mischaikow, and G. Turk, *Vector field design on surfaces*, *ACM Trans. Graph.* **25**, 1294 (2006).
- [31] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe, *Design of tangent vector fields*, *ACM Trans. Graph.* **26**, 56:1 (2007).
- [32] N. Ray, B. Vallet, L. Alonso, and B. Lévy, *Geometry-aware direction field processing*, *ACM Trans. Graph.* **29** (2009).
- [33] K. Crane, M. Desbrun, and P. Schröder, *Trivial connections on discrete surfaces*, *Comput. Graph. Forum* **29**, 1525 (2010).

- [34] Y.-K. Lai, M. Jin, X. Xie, Y. He, J. Palacios, E. Zhang, S.-M. Hu, and X. Gu, *Metric-driven rosy field design and remeshing*, IEEE Trans. Vis. Comput. Graph. **16** (2010).
- [35] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, *Interactive multi-resolution modeling on arbitrary meshes*, in *Proc. of ACM SIGGRAPH* (1998) pp. 105–114.
- [36] K. Hildebrandt and K. Polthier, *Constraint-based fairing of surface meshes*, in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2007) pp. 203–212.
- [37] O. Sorkine and D. Cohen-Or, *Least-squares meshes*, in *Proceedings of Shape Modeling International* (IEEE Computer Society Press, 2004) pp. 191–199.
- [38] A. Jacobson, E. Tosun, O. Sorkine, and D. Zorin, *Mixed finite elements for variational surface modeling*, Comput. Graph. Forum **29**, 1565 (2010).
- [39] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, *Laplacian surface editing*, in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004) pp. 179–188.
- [40] M. Botsch and L. Kobbelt, *An intuitive framework for real-time freeform modeling*, ACM Trans. Graph. **23**, 630 (2004).
- [41] A. Jacobson, I. Baran, J. Popović, and O. Sorkine, *Bounded biharmonic weights for real-time deformation*, ACM Trans. Graph. **30**, 78:1 (2011).
- [42] Y. Wang, A. Jacobson, J. Barbič, and L. Kavan, *Linear subspace design for real-time shape deformation*, ACM Trans. Graph. **34**, 57:1 (2015).
- [43] M. Löhndorf and J. M. Melenk, *On thin plate spline interpolation*, in *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016*, edited by M. L. Bittencourt, N. A. Dumont, and J. S. Hesthaven (Springer, 2017) pp. 451–466.
- [44] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics* (Springer, 2007).
- [45] C. Brandt, L. Scandolo, E. Eisemann, and K. Hildebrandt, *Spectral processing of tangential vector fields*, Comput. Graph. Forum **36**, 338 (2017).
- [46] D. Cohen-Steiner and J.-M. Morvan, *Restricted Delaunay triangulations and normal cycles*, ACM Sympos. Comput. Geom. , 237 (2003).
- [47] K. Hildebrandt and K. Polthier, *Generalized shape operators on polyhedral surfaces*, Computer Aided Geometric Design **28**, 321 (2011).
- [48] J. Martinez Esturo, C. Rössl, and H. Theisel, *Smoothed quadratic energies on meshes*, ACM Trans. Graph. **34**, 2 (2014).
- [49] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing* (AK Peters, 2010).

- [50] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, *Real-time hatching*, in *Proc. ACM SIGGRAPH* (2001).
- [51] Y. Kim, J. Yu, X. Yu, and S. Lee, *Line-art illustration of dynamic and specular surfaces*, *ACM Trans. Graph.* **27**, 156:1 (2008).
- [52] T. Umenhoffer, L. Szécsi, and L. Szirmay-Kalos, *Hatching for motion picture production*, *Comput. Graph. Forum* **30**, 533 (2011).
- [53] E. Kalogerakis, D. Nowrouzezahrai, S. Breslav, and A. Hertzmann, *Learning hatching for pen-and-ink illustration of surfaces*, *ACM Trans. Graph.* **31**, 1 (2012).
- [54] J. Suarez, F. Belhadj, and V. Boyer, *Real-time 3d rendering with hatching*, *The Visual Computer*, 1 (2016).
- [55] F. E. Nicodemus, *Geometrical considerations and nomenclature for reflectance*, Vol. 160 (US Department of Commerce, National Bureau of Standards, USA, 1977).
- [56] M. Ashikhmin and P. Shirley, *An anisotropic phong BRDF model*, *Journal of Graphics Tools* **5**, 25 (2000).
- [57] M. Desbrun, E. Kanso, and Y. Tong, *Discrete differential forms for computational modeling*, (Birkhäuser, 2008) pp. 287–324.
- [58] K. Polthier and E. Preuß, *Variational approach to vector field decomposition*, in *Symposium on Data Visualization* (Springer, 2000) pp. 147–155.
- [59] R. Eymard, T. Gallouët, and R. Herbin, *Finite volume methods*, in *Techniques of Scientific Computing, Part III*, *Handbook of Numerical Analysis*, edited by P. G. Ciarlet and J.-L. Lions (North-Holland, 2000) pp. 713–1020.

APPENDIX

2.A. DERIVATION OF THE DISCRETE HARMONIC ENERGY

The discrete harmonic energy for n -vector fields is defined as

$$E_H(\mathbf{u}) = \sum_{(i,j) \in \mathcal{E}} w_{ij} \|R_{ij} u_i - u_j\|^2 \quad (2.19)$$

$$\text{where } w_{ij} = \frac{3l_{e_{ij}}^2}{\text{area}(T_i \cup T_j)}.$$

In the following, we will justify the weights w_{ij} , as they emerge naturally in a face-based discretization of the harmonic energy for vector fields. The harmonic energy of a smooth vector field \mathbf{v} is given by $E_H^{\text{smooth}}(\mathbf{v}) = \langle \Delta \mathbf{v}, \mathbf{v} \rangle_{\mathbf{L}^2}$, where Δ is the Hodge–Laplace operator

$$\Delta = -(\text{grad div} + \mathbf{J} \text{grad curl}).$$

Rearranging terms, we get

$$\begin{aligned} E_H^{\text{smooth}}(\mathbf{v}) &= \langle \Delta \mathbf{v}, \mathbf{v} \rangle_{\mathbf{L}^2} \\ &= \langle -(\text{grad div} + \mathbf{J} \text{grad curl}) \mathbf{v}, \mathbf{v} \rangle_{\mathbf{L}^2} \\ &= -\langle \text{grad div } \mathbf{v}, \mathbf{v} \rangle_{\mathbf{L}^2} - \langle \mathbf{J} \text{grad curl } \mathbf{v}, \mathbf{v} \rangle_{\mathbf{L}^2} \\ &= \langle \text{div } \mathbf{v}, \text{div } \mathbf{v} \rangle_{\mathbf{L}^2} + \langle \text{curl } \mathbf{v}, \text{curl } \mathbf{v} \rangle_{\mathbf{L}^2} \end{aligned}$$

where we used the equations $\langle \text{div } \mathbf{v}, f \rangle_{\mathbf{L}^2} = -\langle \mathbf{v}, \text{grad } f \rangle_{\mathbf{L}^2}$, $\text{curl } \mathbf{v} = -\text{div } \mathbf{Jv}$ and $\langle \mathbf{Jv}, \mathbf{v} \rangle_{\mathbf{L}^2} = -\langle \mathbf{v}, \mathbf{Jv} \rangle_{\mathbf{L}^2}$, which hold for all smooth vector fields \mathbf{v} and square integrable functions f . For more background, we refer to [45]. In [31] and [57], a DEC-based discretization of this energy is used for vector field design. We are dealing with piecewise constant vector fields that jump at the edges. In this setting, discrete divergence and curl operators can be defined by testing a weak form of the divergence and curl with test functions. As test functions, we are using Crouzeix–Raviart finite elements, which are functions on the mesh that are linear polynomials in every triangle and edge-midpoint continuous. The nodes of the Crouzeix–Raviart elements are located at the midpoints of the edges of the mesh. The (non-conforming discrete) divergence and curl map piecewise constant vector fields to such edge-based functions. For more background on the discrete divergence and curl, we refer to [45, 58]. The values at the edge-midpoints are given by

$$\text{div}^* \mathbf{v}(m_{i,j}) = \frac{3}{A_{T_i \cup T_j}} (\langle v_i, J_i e_{ij} \rangle - \langle v_j, J_j e_{ij} \rangle), \quad (2.20)$$

$$\text{curl}^* \mathbf{v}(m_{i,j}) = \frac{3}{A_{T_i \cup T_j}} (\langle v_j, e_{ij} \rangle - \langle v_i, e_{ij} \rangle), \quad (2.21)$$

where $(v_1, \dots, v_{|\mathcal{F}|})$ are the piecewise constant vectors of the field per face, $m_{i,j}$ is the midpoint of the common edge between two adjacent triangles T_i and T_j , J_i is the operator that rotates a vector in the tangent plane of T_i by $\pi/2$ (following the orientation of the surface), $A_{T_i \cup T_j}$ is the combined area of T_i and T_j , and e_{ij} is the **non**-normalized directed (following the orientation of T_i) common edge between triangles T_i and T_j .

With these operators, one can discretize the harmonic energy:

$$\begin{aligned} E_H(\mathbf{v}) &= \langle \operatorname{div}^* \mathbf{v}, \operatorname{div}^* \mathbf{v} \rangle_{\mathbf{L}^2} + \langle \operatorname{curl}^* \mathbf{v}, \operatorname{curl}^* \mathbf{v} \rangle_{\mathbf{L}^2} \\ &= \sum_{(i,j) \in \mathcal{E}} \frac{3}{A_{T_i \cup T_j}} \left((\langle v_i, J_i e_{ij} \rangle - \langle v_j, J_j e_{ij} \rangle)^2 \right. \\ &\quad \left. + (\langle v_j, e_{ij} \rangle - \langle v_i, e_{ij} \rangle)^2 \right) \end{aligned} \quad (2.22)$$

$$\begin{aligned} &= \sum_{(i,j) \in \mathcal{E}} \frac{3l_{e_{ij}}^2}{A_{T_i \cup T_j}} \left(\left\langle R_{ij} v_i - v_j, J_j \frac{e_{ij}}{l_{e_{ij}}} \right\rangle^2 \right. \\ &\quad \left. + \left\langle v_j - R_{ij} v_i, \frac{e_{ij}}{l_{e_{ij}}} \right\rangle^2 \right) \end{aligned} \quad (2.23)$$

$$= \sum_{(i,j) \in \mathcal{E}} \frac{3l_{e_{ij}}^2}{A_{T_i \cup T_j}} \|R_{ij} v_i - v_j\|^2 \quad (2.24)$$

In (2.22), we simply inserted the definition of the operators and canceled the squared area factors in div^* and curl^* with the area factors coming from the sum that computes the L^2 scalar product for the Crouzeix–Raviart elements. In (2.23), we used that $\langle v_i, e_{ij} \rangle = \langle R_{ij}(v_i), e_{ij} \rangle$. (R_{ij} is the connection for 1-fields) and the linearity of the scalar product. Finally, in (2.24), we used that $\langle v, w \rangle^2 + \langle v, \mathbf{J}w \rangle^2 = \|v\|^2$ for any pair of an arbitrary vector v and a unit vector w . Then, since for $n = 1$, when using the same tangent space bases for the vectors and their n -vector representations, the vectors v_i coincide with the n -vectors u_i , so that (2.24) coincides with (2.2).

Since we end up with a weighted sum of finite differences, we can extend this energy to arbitrary n by taking the differences of the u_i representing the n -vectors by transporting them into a common tangent space. This leads to the generalized harmonic energy (2.19).

As discussed in Section 2.11, the discrete harmonic energy does not have a kernel of “discrete harmonic” n -vector fields for surfaces of non-trivial genus. For vector fields such a discretization can be obtained by combining conforming and non-conforming finite elements, see [45]. However, it remains an open question how this construction can be carried over from 1-vector fields to n -vector fields.

Finally, we want to remark that in addition to the harmonic energy and the Hodge–Laplace operator, one can consider the Bochner–Laplace operator and the corresponding quadratic energy. For more background, we refer to [1, 2]. In the notation used in these papers, the harmonic energy is called the anti-holomorphic energy. The construction of a discrete Bochner Laplace operator for piecewise constant vector fields on triangle meshes remains an interesting open problem.

Relation to Finite Volumes We want to remark that in addition to the derivation described above, the discrete harmonic energy (2.19) can be interpreted as a finite volume discretization. The control volumes are the triangles and each summand $w_{ij} \|R_{ij}u_i - u_j\|^2$ of the discrete harmonic energy measures the diffusive flux through the edge (i, j) between adjacent triangles. The weights w_{ij} are the transmissibilities.

For more background on finite volume methods, we refer to [59]. The discrete harmonic energy (2.19) is analogous to the discrete harmonic energy (or squared discrete Sobolev H_0^1 semi norm) for piecewise constant functions on triangulations of compact domains in \mathbb{R}^2 , compare [59, Chapter 9, eq. (9.12)]. In contrast to their setting, we are working with vector fields and curved surface meshes, this is why the transport of vector fields R_{ij} is needed for evaluating the fluxes. Based on this analogy, we could use the weights that are commonly used for finite volume discretization of the harmonic energy with piecewise constant functions for our purposes. These are

$$\frac{2}{\cot \alpha_{ij} + \cot b_{ij}},$$

where α_{ij} and b_{ij} are the angles opposite of the edge (i, j) in the two adjacent triangles. The problem with these weights is that they may be negative, which can be avoided by requiring the triangulation to be Delaunay. The weights we propose are positive for any triangulation by construction. To the best of our knowledge, the weights we are proposing have not been used in the context of finite volume methods. It is an interesting task to further explore the properties of these weights and their use for finite volume methods.

2.B. LINEAR SYSTEM FOR n -DIRECTION FIELD SPLINES

In this section, we derive the linear system we solve to compute n -vector field splines and discuss the relationship between μ and λ . Our proof re-uses arguments from [1], however, since we use hard constraints and different objectives and discretizations, some modifications need to be made, and we will verify correctness for our specific system.

We will show that $\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|_{\mathbf{L}^2}}$, for \mathbf{w} being a solution of the linear system from Section 2.7

$$\begin{pmatrix} \mathbf{B} - \lambda \mathbf{M} & \mathbf{D}^T \\ \mathbf{D} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\mathbf{u}' \\ 0 \end{pmatrix}, \quad (2.25)$$

is a minimizer of the previously stated n -vector field spline problem

$$\begin{aligned} \min_{\mathbf{u}} E_B(\mathbf{u}) - 2\mu \langle \mathbf{u}, \mathbf{u}' \rangle_{\mathbf{L}^2} \\ \text{subject to } \|\mathbf{u}\|_{\mathbf{L}^2} = 1 \\ \text{and } \langle u_i, J_i d_i \rangle = 0 \forall i \in I \end{aligned} \quad (2.26)$$

To this end note that (2.25) can be rewritten as

$$\begin{aligned} \mathbf{B}\mathbf{w} - \mu \|\mathbf{w}\|_{\mathbf{L}^2} \mathbf{M}\mathbf{u}' &= \lambda \mathbf{M}\mathbf{w} - \mathbf{D}^T \boldsymbol{\gamma} \\ \text{and } \mathbf{D}\mathbf{w} &= 0, \end{aligned} \quad (2.27)$$

where $\mu = \frac{1}{\|\mathbf{w}\|_{\mathbf{L}^2}}$. Multiplying (2.27) by $\frac{1}{\|\mathbf{w}\|_{\mathbf{L}^2}}$, letting $\tilde{\boldsymbol{\gamma}} = \frac{\boldsymbol{\gamma}}{\|\mathbf{w}\|_{\mathbf{L}^2}}$ and plugging in $\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|_{\mathbf{L}^2}}$ we get

$$\mathbf{B}\mathbf{u} - \mu\mathbf{M}\mathbf{u}' = \lambda\mathbf{M}\mathbf{u} - \mathbf{D}^T\tilde{\boldsymbol{\gamma}} \quad (2.28)$$

$$\text{and } \|\mathbf{u}\|_{\mathbf{L}^2} = 1$$

$$\text{and } \langle u_i, J_i d_i \rangle = 0 \quad \forall i \in I$$

since $\langle w_i, J_i d_i \rangle = 0 \Leftrightarrow \langle u_i, J_i d_i \rangle = 0$. These are the necessary conditions for the constrained optimization problem (2.26), where the Lagrange multipliers are 2λ and $2\tilde{\boldsymbol{\gamma}}$. Since the minimized functional is quadratic in \mathbf{u} , this is sufficient to show that \mathbf{u} is a feasible minimizer.

To determine the relationship between λ and μ , we first make a change of basis $\mathbf{w} = \mathbf{C}\hat{\mathbf{w}}$, where the image of $\mathbf{C} \in M(2|\mathcal{F}| \times 2|\mathcal{F}| - m)$ is the vector space of all fields that satisfy the hard constraints, *i.e.* $\mathbf{D}\mathbf{w} = 0 \Leftrightarrow \mathbf{w} \in \text{Image } \mathbf{C}$ and $\mathbf{C}^T\mathbf{C} = \mathbf{I}$. Then (2.25) is equivalent to solving

$$\mathbf{C}^T(\mathbf{B} - \lambda\mathbf{M})\mathbf{C}\hat{\mathbf{w}} = \mathbf{C}^T\mathbf{M}\mathbf{u}', \quad (2.29)$$

since \mathbf{u}' satisfies all hard constraints by construction and so $\mathbf{u}' \in \text{Image } \mathbf{C}$. Let $\hat{\mathbf{U}}$ be a full eigenbasis of the constrained bi-Laplacian $\hat{\mathbf{V}} := \mathbf{C}^T(\mathbf{L}^T\mathbf{M}\mathbf{L})\mathbf{C}$ which is mass-orthonormal, *i.e.* $\hat{\mathbf{U}}^T\mathbf{C}^T\mathbf{M}\mathbf{C}\hat{\mathbf{U}} = \mathbf{I}$. Then (2.29) is equivalent to

$$(\boldsymbol{\Lambda} - \lambda\mathbf{I})\hat{\mathbf{w}}_{\hat{\mathbf{U}}} = \hat{\mathbf{U}}^T\mathbf{C}^T\mathbf{M}\mathbf{u}', \quad (2.30)$$

where $\hat{\mathbf{w}}_{\hat{\mathbf{U}}} = \hat{\mathbf{U}}^T\hat{\mathbf{w}}$ and $\boldsymbol{\Lambda}$ stacks the eigenvalues $\hat{\lambda}_i$ of $\hat{\mathbf{V}}$ ordered by magnitude (note that the eigenvalues are all positive as the frequency of hard constrained fields can only be higher than those of an unconstrained field). Now note that $(\hat{\mathbf{w}}_{\hat{\mathbf{U}}})_i = (\hat{\mathbf{U}}^T\mathbf{C}^T\mathbf{M}\mathbf{u}')_i (\hat{\lambda}_i - \lambda)^{-1}$. So, when $\lambda \rightarrow \hat{\lambda}_1$, $(\hat{\mathbf{w}}_{\hat{\mathbf{U}}})_1$ goes to (plus or minus) infinity. This implies $\|\hat{\mathbf{w}}\| \rightarrow \infty$, and so $\mu = \frac{1}{\|\mathbf{w}\|} \rightarrow 0$. Conversely, when $\lambda \rightarrow -\infty$ we have that $(\hat{\mathbf{w}}_{\hat{\mathbf{U}}})_i \rightarrow 0$ for all i and so $\mu \rightarrow \infty$.

3

HYPER-REDUCED PROJECTIVE DYNAMICS

This chapter is based on the publication *Hyper-Reduced Projective Dynamics* by Christopher Brandt, Elmar Eisemann and Klaus Hildebrandt, published in ACM Transactions on Graphics in 2018.

3.1. OVERVIEW

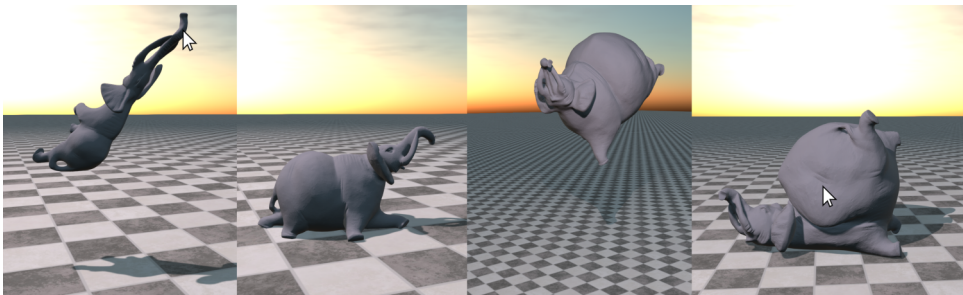


Figure 3.1: Our Hyper-Reduced Projective Dynamics method is able to handle different types of constraints simultaneously (volume preservation for tetrahedrons and strain resistance for boundary triangles) for complex geometries (52k vertices) in real-time. During the simulation we change the target volume of the tetrahedrons, which causes a balloon-like blowup effect. We resolve ground collisions and allow user interaction by click and drag. The simulation runs at 62 fps including rendering. The total precomputation time of our method is 21 seconds.

The simulation of deformable objects is an important factor in various entertainment and training applications of computer graphics. Of particular interest are real-time simulations since these allow the user to interact with the simulation and thereby greatly enrich the experience in games, virtual reality, artistic applications and medical training.

The combination of the nonlinear nature of deformable objects and the geometric complexity of objects in these scenarios make real-time simulation a challenging problem. In addition to efficiency, robustness is important for real-time simulation since interactivity leads to unforeseeable states of the system.

Projective Dynamics, introduced in [1, 2], is a simulation framework that is *general*, since it allows for the simulation of different types of deformable objects (solids, shells and rods) and materials. It is *robust*, as it capable of adequately handling large time steps, large deformations and degenerate geometries, and it is *fast*. On the technical level, a variational implicit time integration scheme is used and the resulting optimization problem is solved by an alternating local/global approach. The local steps in the optimization can be executed in parallel, and for *constraint-based* potentials, the system matrix of the linear system to be solved in the global steps is constant, allowing for the reuse of a sparse factorization of the matrix. While this makes Projective Dynamics highly efficient for real-time simulation, it is still limited to objects that can be represented by coarse geometric models.

We propose a specific model reduction approach for Projective Dynamics, *hyper-reduced Projective Dynamics*, that enables the real-time simulation of deformable objects with complex geometry. The scheme is designed such that the computational cost for time integration is independent of the complexity of the mesh representing the object. It proceeds in two stages. First, a subspace to which the simulation is restricted is constructed. In contrast to Chapter 1 and 2, this step reduces the degrees of freedom of the simulation, but does not necessarily lower the cost for solving the optimization problem required for integrating the equations of motion. This is because the complexity for evaluating the constraint projections, which describe the acting forces, is not reduced. Therefore, a second level of approximation, a *hyper-reduction*, is needed. We introduce a novel approach, the *constraint projections fitting method*, for this second stage. The idea is to construct a second subspace during the preprocessing stage—not for vertex positions—but for the constraint projections. Then, in the online stage, only a limited number of constraints are evaluated and a fitting problem in the second subspace is solved to obtain approximations of the constraint projections. This scheme is inspired by the empirical interpolation method from applied mathematics and continuum mechanics [3, 4] (though our scheme is neither empirical nor interpolating) and is the first application of this type of method for reduced simulation in graphics.

To implement the hyper-reduced Projective Dynamics framework, constructions for the two subspaces are needed. We propose two subspace constructions that refrain from using prior knowledge about the nature of the simulation (types of user interaction, external forces, collision constraints, etc.). Only the mesh and the types of materials to be simulated are known. While this setting does not allow our method to profit from specialization to a specific scenario, the resulting benefits are an automatic and accelerated preprocessing stage, which avoids costly probing of the simulation or modal analysis.

Our position subspace construction follows constructions of skinning spaces [5, 6]. However, instead of solving a systems for each basis function, we use compactly supported radial basis functions for the weights. As a result, subspaces of large scale (up to the original size of the mesh) can be constructed efficiently and the resulting basis vectors are sparse. We generalize the construction to obtain subspaces of constraint pro-

jections. Since the subspace constructions are based on sampling, our subspace constructions make the assumption that the deformations vary smoothly along the object. Note, however, that this is not a general limitation of the proposed hyper-reduction. For example, subspaces constructed from snapshots (taken from a full simulation) could add high-frequency deformations, like sharp bends, to the subspaces. The general assumption for the hyper-reduced simulation is that the object remains close to a low-dimensional submanifold in configuration space.

The resulting hyper-reduced system is highly efficient: In our experiments, we achieve real-time rates of 60 fps for simulations in 4k-dimensional subspaces, which is an order of magnitude higher than what is reported for recent hyper-reduced schemes like [7, 8]. Precomputation time for a mesh with 19k vertices is 6.8 seconds and one minute for a complex geometry with 200k vertices. Our hyper-reduced system enables rich dynamics and can plausibly resolve unexpected events, such as collisions, drastic user interactions and online changes of the material stiffness and the geometry (*e.g.* the volume).

3.2. RELATED WORK

Our method combines the benefits of Position-Based and Projective Dynamics methods with the efficiency offered by model reduction techniques. In the following, we provide an brief overview of literature in these fields and embed our method into prior work.

Position-Based Dynamics Position-Based Dynamics [9, 10] were introduced as general and efficient methods to enable the simulation of deformable objects in real-time. They can be seen as extensions of Shape Matching Methods [11, 12]. Position-Based Dynamics omits the velocity layer and instead works directly on the positions. Inner forces are expressed as equality or inequality constraints, which are being enforced in a Gauss-Seidel-type fashion. This *constraint projection* step can be carried out in parallel and heavily sped up via GPU implementations. Since then the method has been extended in terms of robustness, convergence and generality [13–15]. Most recently the problem of controlling stiffness parameters independently of the iteration count has been addressed [16]. The method has been extended to fluids [17] and continuum based materials [18, 19]. For a recent survey on Position-Based simulation techniques we refer to [20].

Projective Dynamics Projective Dynamics [1, 2] is a method for implicit time integration of physical systems, which combines a variational integration scheme [21] with a specific class of potentials modeling the inner forces, resulting in a highly flexible and fast technique that is capable of simulating strings, cloth, elastic deformable objects and recently fluids [22]. Narain et al. [23] and Overby et al. [24] apply the Alternating Direction Method of Multipliers (*ADMM*) optimization algorithm to the variational integration scheme and show that the resulting method closely relates to Projective Dynamics. From this point of view, it can be extended to nonlinear constitutive materials and hard constraints. Liu et al. [25] show that Projective Dynamics can be seen as a quasi-Newton method for implicit variational time integration for certain types of potentials. This also allows for a generalization of the method to handle a large class of materials, such as

Neo-Hookean or spline-based materials. While both generalizations give rise to similar reduced methods for nonlinear materials, our method is a hyper-reduction for the original Projective Dynamics method, and we make use of specific properties of the system to gain the efficiency shown in our results. Wang [26] presents a GPU implementation of the Projective Dynamics method, where Chebyshev iterations are combined with Jacobi or Gauss-Seidel iterations to approximate the system in the global step. This enhances convergence and reduces computation times for the Projective Dynamics time steps. It allows, for example, the simulation of a complex 20k vertex mesh at 38 fps. However, being an unreduced method, the computation time for each iteration still depends on the mesh resolution, such that real-time frame rates are only possible up to a certain number of vertices. In contrast, our Hyper-Reduced Projective Dynamics method achieves computation times that are independent of the mesh resolution. For example, we are able to simulate a mesh with 200k vertices in 60 fps, including collision handling and rendering, using a CPU implementation on a consumer desktop computer.

Model reduction for deformables An early application of model reduction to the simulation of deformable objects in graphics is [27], in which a linearized simulation is restricted to a subspace spanned by the low-frequency linear vibration modes. Since the linear modes do not capture non-linearities well, techniques for augmenting linear modal bases for nonlinear deformable object simulation have been developed: Choi and Ko [28] proposed *modal warping*, Barbič and James [29] basis augmentation with *modal derivatives*, Huang et al. [30] and Pan et al. [31] *rotation strain coordinates*, Tycowicz et al. [7] basis enrichment via linear space transformations, and Yang et al. [32, 33] a *linear inertia mode* technique. Since vibration modes are globally supported, modal bases consist of dense vectors. Brandt and Hildebrandt [34] introduced a scheme for the compression of vibration modes resulting in sparse basis vectors. An alternative to using modal analysis are *empirical eigenmodes* [29, 35], which collect snapshots of the systems to be reduced and use principal component analysis for the construction of a reduced basis. Neumann et al. [36] proposed a sparse PCA approach to get sparse empirical deformation bases. A third type of subspaces are skinning spaces. Skinning spaces constructed by sampling the geometry were used for fast simulation, for example, by Gilles et al. [37] and Jacobson et al. [38].

Hyper-reduced systems combine dimensional reduction with a scheme for fast reduced force approximation. For St. Venant–Kirchhoff materials, the elastic potential is a quartic polynomial in the vertex displacements. Barbič and James [29] suggest the pre-computation of the coefficients of this polynomial in the reduced space, which enables exact evaluation of potential, forces, and the Hessian at a cost depending only on the dimension of the subspace. For general materials, one needs to resort to approximation. The optimized cubature, introduced by An et al. [39], approximates the subspace forces by a linear combination of projections of a selection of local force vectors. The weights and sample locations of local force vectors are determined by cubature training, in which the approximation error against a set of snapshots of force vectors is minimized. Tycowicz et al. [7] introduced a *non-negative hard thresholding pursuit* solver that significantly speeds-up cubature training. Yang et al. [33] further reduce the training time by an effective strategy for training data generation, which generates only the data required to reach

a given error margin. Kim and Delaney [40] and Harmon and Zorin [41] use importance-based sampling to determine cubature point locations and optimize only the weights. Wu et al. [8] propose a domain decomposition approach to improve performance of cubature training and evaluation. Since for our hyper-reduction of Projective Dynamics, approximations of constraint projections as opposed to force vectors are needed, polynomial reduction and cubature cannot be applied. Conversely, the proposed fitting method holds potential for other model reduction problems in graphics.

In addition to simulation, model reduction has been used for controlling and editing the motion of deformable objects [42–45], interactive material design [46], sound synthesis [47], clothing simulation [48], deformation-based shape modeling [6, 38, 49, 50], elasticity-based shape interpolation [51, 52], curve processing in shape space [53], and shape optimization [54, 55].

Coarse simulation An alternative to the methods considered in this work is to simulate a coarse mesh and couple the coarse mesh to the high-resolution geometry such that deformations of the coarse mesh can be propagated to deformations of the fine mesh [56–61]. To put these approaches in context with our model reduction scheme, one can observe that the coupling of the coarse and fine mesh results in a specific subspace for the fine mesh, where the reduced coordinates are the vertex positions of the coarse mesh. The simulations in the subspaces, however, differ significantly. Instead of computing a coarse simulation, our hyper-reduced scheme aims at approximating the dynamics of the high-resolution mesh. In Appendix 3.D we present a comparison of our method and a coarse simulation.

3.3. BACKGROUND: PROJECTIVE DYNAMICS

For a mesh with time-dependent vertex positions $\mathbf{q} \in \mathbb{R}^{n \times 3}$, the goal is to find trajectories of the laws of motion

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f}_{\text{int}}(\mathbf{q}) + \mathbf{f}_{\text{ext}} \quad (3.1)$$

with initial conditions $\mathbf{q}(0) = \mathbf{q}_0$ and $\dot{\mathbf{q}}(0) = \mathbf{v}_0$. Here \mathbf{f}_{ext} are (possibly time-dependent) external forces, such as gravity, and $\mathbf{f}_{\text{int}}(\mathbf{q}) = -\sum_i \nabla W_i(\mathbf{q})$ are internal forces acting on the mesh, such as elastic forces acting on deformable objects. Equation (3.1) can be solved via implicit Euler integration in a variational manner by solving the following optimization problem

$$\mathbf{q}(t+h) = \arg \min_{\mathbf{q}} \frac{1}{2h^2} \left\| \mathbf{M}^{\frac{1}{2}} (\mathbf{q} - \mathbf{s}) \right\|_F^2 + \sum_i W_i(\mathbf{q}) \quad (3.2)$$

$$\mathbf{v}(t+h) = \frac{1}{h} (\mathbf{q}(t+h) - \mathbf{q}(t)) \quad (3.3)$$

where $\mathbf{s} = \mathbf{q}(t) + h\mathbf{v}(t) + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$. The functional that is optimized in Equation (3.2) can be read as a trade-off between the momentum potential, which expresses that the object should follow its current trajectory (possibly altered by external forces), and the elastic potential, which maintains the shape of the object.

Projective Dynamics is particularly efficient for potentials W_i that have the following form:

$$W_i(\mathbf{q}) = \sum_j \frac{\lambda_j}{2} \|\mathbf{S}_j \mathbf{q} - \mathbf{p}_j(\mathbf{q})\|^2 \quad (3.4)$$

$$\text{where } \mathbf{p}_j(\mathbf{q}) = \arg \min_{\mathbf{p} \in \Omega} \|\mathbf{S}_j \mathbf{q} - \mathbf{p}\|^2 \quad (3.5)$$

where the triplets $\{\lambda_j, \mathbf{S}_j, \mathbf{p}_j\}$ are called *constraints* and

- λ_j is a scalar weight denoting the stiffness of the constraint (and contains the area or volume associated to the constraint).
- \mathbf{S}_j is a $p \times n$ matrix, which usually is a linear discrete differential operator computing deformation gradients, Laplacians, or other quantities for a certain element of the mesh.
- \mathbf{p}_j , the *constraint projection*, is a, typically nonlinear, function from $\mathbb{R}^{n \times 3}$ to $\mathbb{R}^{p \times 3}$ that projects the differential property $\mathbf{S}_j \mathbf{q}$ onto a constraint manifold Ω .

With these types of potentials, one can model a variety of materials, *e.g.* they can be used as bending and strain energies for thin shells, elastic energies for tetrahedral meshes, spring energies, and unconventional energies such as example-based materials [21]. The derivation and implementation of specific constraints can be found in the original Projective Dynamics paper [2]. Most importantly, these constraints admit a robust minimization scheme for solving the optimization problem stated in Equation (3.2), which is a local/global approach. In the local step, the constraint projections can be evaluated independently in parallel, while in the global step, a system is solved that minimizes the objective for fixed constraints. This system decouples into the spatial dimensions x , y , and z , such that it can be solved by three linear solves of dimension n in parallel. The resulting method is listed in Algorithm 2.

Algorithm 2 Unreduced Projective Dynamics

Input: Current vertex positions $\mathbf{q}(t)$ and current velocities $\mathbf{v}(t)$

Let $\mathbf{s} = \mathbf{q}(t) + h\mathbf{v}(t) + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$

Let $\hat{\mathbf{q}} = \mathbf{s}$

for $i = 1$ **to** numIterations **do**

Local step: Evaluate all constraint projections $\mathbf{p}_i = \mathbf{p}_i(\hat{\mathbf{q}})$

Global step: Solve

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{S}_i \right) \mathbf{q} = \frac{\mathbf{M}}{h^2} \mathbf{s} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$$

 Let $\hat{\mathbf{q}} = \mathbf{q}$

end for

Output: Updated vertex positions $\mathbf{q}(t+h) = \hat{\mathbf{q}}$ and velocities $\mathbf{v}(t+h) = (\mathbf{q}(t+h) - \mathbf{q}(t))/h$

3.4. VERTEX POSITIONS SUBSPACE CONSTRUCTION

In order to reduce the degrees of freedom in the optimization problem stated in Equation (3.2), we opt for a linear subspace $\mathbf{U} \in \mathbb{R}^{n \times 4k}$, such that vertex positions $\mathbf{q} \in \mathbb{R}^{n \times 3}$ can be approximated as $\mathbf{q} \approx \mathbf{U}\tilde{\mathbf{q}}$, where $\tilde{\mathbf{q}} \in \mathbb{R}^{4k \times 3}$ and $4k \ll n$. Such linear subspaces have been employed for many problems in computer graphics and specifically for the reduction of simulation and modeling applications, see Section 3.2.

Subspace construction We opt for creating a subspace from skinning weights (see Appendix 3.B), similar to the subspaces used in [5, 6]. However, we define the required weights in a way that produces compactly supported basis functions, is well suited for scaling the trade-off between accuracy and performance, requires no user input, and can be computed in a few seconds at most. On a conceptual level we want to introduce degrees of freedom from affine transformations acting on equidistantly distributed handles (or areas) on the mesh, with smoothly varying, localized influence on the nearby vertices.

To this end, we first choose k sample vertices \mathbf{s}_j which are distributed approximately equidistant over the mesh, using furthest point sampling. That is, the first vertex sample is chosen randomly and then we iteratively add the vertex which has furthest distance from all previously chosen samples on the mesh as the next sample. To ensure that the sampling is fair, even in presence of complex details, it is important that distances are measured as the lengths of geodesics on (or, in case of volumetric meshes, through) the mesh. To quickly evaluate geodesic distances to all of the sample points, we use the Short Term Vector Dijkstra (STVD) method proposed by Campen et al. [62], which is a modification of the original Dijkstra algorithm that replaces the distance update with a method that uses a stack of predecessor edges to compute good approximations of the actual geodesic distances.

Preliminary weights for each vertex i and each of the k samples are then acquired from radial basis functions around the handles. Specifically they are defined as $\tilde{\mathbf{w}}_i = (B_{\mathbf{s}_1, r}(\mathbf{q}_i), \dots, B_{\mathbf{s}_k, r}(\mathbf{q}_i))$, where r is a radius whose choice we detail below and $B_{\mathbf{y}, r}(\mathbf{x}) = b_r(d(\mathbf{x}, \mathbf{y}))$ is a scalar function on the mesh vertices. On $[0, r]$, we choose b_r as the unique cubic polynomial with $b_r(r) = b'_r(0) = b'_r(r) = 0$ and $b_r(0) = 1$ and define $b_r(t)$ to be 0 for $t > r$. Other choices are possible, as long as they are smooth, monotone and interpolate between 1 and 0 on $[0, r]$. The final weights \mathbf{w}_i are then normalized as $\mathbf{w}_i = \tilde{\mathbf{w}}_i \cdot (1. / \sum_j \tilde{w}_i^j)$ in order to ensure reproducibility of the rest shape. To be able to evaluate the functions $B_{\mathbf{s}_j, r}$ we need to evaluate the geodesic distances $d(\mathbf{s}_j, \mathbf{q}_i)$ for all vertex positions \mathbf{q}_i . For this, we also employ the STVD method, where the search can be terminated early, since vertices with distance greater than r do not need to enter the queue (at the end unvisited vertices receive the value $B_{\mathbf{s}_j, r}(\mathbf{q}_i) = 0$). This setup enables the computation of smooth and locally supported weight functions, while not relying on solving box-constraint quadratic problems [5] or linear systems [6] on the mesh for every basis function, which becomes costly for high resolution meshes.

The choice of the parameter r is crucial for two reasons: on the one hand, the sparsity of the subspace matrix, which depends on this parameter, influences the performance of the Hyper-Reduced Projective Dynamics algorithm, since the cost of updating sampled vertex positions is tied to the sparsity pattern. On the other hand, r should be cho-

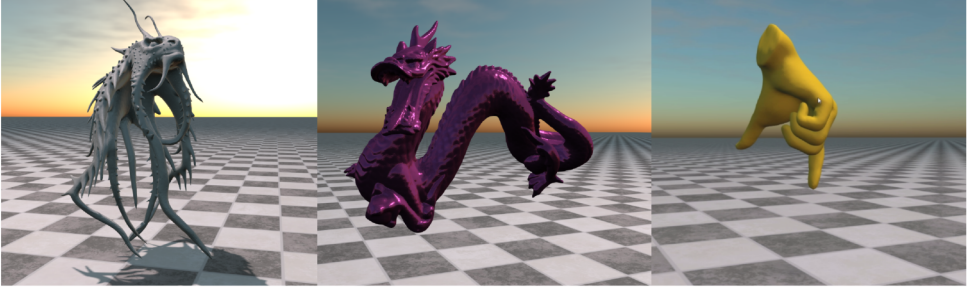


Figure 3.2: Compilation of frames from three simulations using Hyper-Reduced Projective Dynamics. The squid simulation (440k vertices, 1.6 million tetrahedrons) runs at 37 fps (including rendering and ground collision handling), the rest of the examples run at least at 60 fps.

sen large enough to ensure that each vertex is within the support of at least one weight function. In practice we found that $r = 2 \cdot \max_i \min_j d(\mathbf{s}_j, \mathbf{q}_i)$, offers a good compromise between sparsity of the base matrix and coverage of the vertices by a sufficient number of non-zero weights. Note that the quantities $\min_j d(\mathbf{s}_j, \mathbf{q}_i)$ can be evaluated cheaply by performing a last update to the available distances, adding the last source s_k .

After the weights have been computed, we construct the subspace basis matrix \mathbf{U} such that it contains the degrees of freedom from the skinning transformations according to these weights. This construction is detailed in Appendix 3.B. The result is a sparse matrix of size $n \times (4 \cdot k)$ which offers $12 \cdot k$ degrees of freedom for the simulated mesh, (k is the number of sample points chosen in the construction of the weight functions above). Restricting the system to the degrees of freedom offered by the subspace matrix \mathbf{U} , the global step of the Projective Dynamics method is replaced by

$$\mathbf{U}^T \left(\frac{\mathbf{M}}{h^2} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{S}_i \right) \mathbf{U} \tilde{\mathbf{q}} = \mathbf{U}^T \frac{\mathbf{M}}{h^2} \mathbf{U} \tilde{\mathbf{s}} + \mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i, \quad (3.6)$$

where $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{s}}$ are the current subspace coordinates for positions and the momentum term \mathbf{s} .

In Section 3.7, we investigate how well we can approximate states \mathbf{q} of a full simulation by their projections $\tilde{\mathbf{q}}$ into subspaces obtained from our construction, using different values of k , see Table 3.2.

3.5. CONSTRAINT PROJECTIONS FITTING METHOD

In the previous section we constructed a subspace for the vertex positions in order to reduce the dimensionality of the problem. This reduces the cost for the linear solve in the global step and leads to faster convergence to a subspace optimum. However, the cost of evaluating the constraint projections in the local step remains unchanged. In particular, it still depends on the resolution of the mesh instead of the desired detail of the dynamics.

Overview We propose a novel way of directly approximating the term $\mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$ on the right hand side of the reduced system (3.6) for the global step. The approach consists

of a precomputation step and several steps in the online phase.

- **Precomputation:** We construct a subspace for constraint projections \mathbf{V} and select s constraint samples to evaluate.
- **Online Phase:**
 1. Evaluate the positions of all vertices that appear in any of the sampled constraints.
 2. Evaluate the sampled constraint projections.
 3. Solve a fitting problem to find a best approximating vector $\tilde{\mathbf{p}}$ in the space spanned by \mathbf{V} .
 4. Evaluate $\mathbf{r} := \mathbf{U}^T \mathbf{S}^T \mathbf{V} \tilde{\mathbf{p}}$. (\mathbf{S} will be defined in (3.7) below.)

Also note that both the precomputation and the online steps are separated for each *type* of constraint. For example, if both bending and strain constraints for a triangular mesh are present, different subspaces \mathbf{V} are constructed, separate fitting problems are solved and their contributions to the right hand side are evaluated individually and then summed.

Subspace for constraint projections When approximating the term $\mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$, one would like to skip the costly evaluation of *all* $\mathbf{p}_i(\mathbf{q})$, as well as prevent the large vector matrix multiplication with \mathbf{U}^T . To address the first goal, we design a basis \mathbf{V} that spans a subspace of constraint projections, such that $(\mathbf{p}_0(\mathbf{q}), \mathbf{p}_1(\mathbf{q}), \dots) \approx \mathbf{V} \tilde{\mathbf{p}}$. The space needs to be general enough that it contains good approximations of any such vectors that are encountered during a simulation, but concise enough that solving a least-squares fitting problem using a few constraint projections into this subspace yields these approximations. \mathbf{V} can be constructed by using snapshots of forces (or, in our case, constraint projections) from full simulations (see *e.g.* the Discrete Empirical Interpolation Method [4]). Specifically for high-resolution meshes, conducting full simulations in which enough interesting states can be observed to construct a sufficiently rich space is a complicated and time-consuming endeavor. Additionally, it requires knowledge about the type of interaction, collision constraints and external forces that will be encountered during the reduced simulation, which we do not want to assume in order to stay as general as possible. Note that in the reduced dynamics, the contribution of the constraint projections to the right hand side will always be filtered by \mathbf{U}^T . This suggests that one might want to use the columns of \mathbf{U} directly to obtain vectors of constraint projections from these. However, this can only yield a meaningful subspace if the columns of \mathbf{U} directly correspond to meaningful shapes or deformations, which is not the case for our construction, and is an assumption we want to avoid. Thus, we propose a construction that is similar to that of the positions subspace described in Section 3.4.

Again, we choose k' approximately equidistant sample vertices on the mesh and construct weight functions \mathbf{w}_j with a limited support around each of the samples. For efficiency reasons, we use the first k' vertices of the k vertex samples from the construction of \mathbf{U} (in all our examples we have $k' < k$). Thus, the corresponding weight functions

can be also be reused, given that the radius was chosen large enough such that this subset of weight functions still covers all vertices with sufficiently many non-zero weights. The weight functions are then interpolated at the relevant elements for the current constraint type (*i.e.* at the triangles for surface strain constraints, tetrahedrons for volume preservation constraints, etc.) and then normalized to sum to one at each element.

We then evaluate all constraint projections $\mathbf{p}_i = \mathbf{p}_i(\mathbf{q}_0)$ of the rest shape \mathbf{q}_0 of the mesh. Note that usually we have $\mathbf{p}_i(\mathbf{q}_0) = \mathbf{S}_i \mathbf{q}_0$ for the rest-shape (such that no inner forces act on it). Note that for all types of constraints that have been introduced so far, these projections are geometric quantities such as mean curvature vectors (for bending energies), transformation matrices (for strain or volume preservation energies) or simply edges of the mesh (for spring energies). We can now think of these quantities as being attached to the corresponding elements of the mesh. By replacing the role of the vertex positions \mathbf{q}_i with the constraint projections \mathbf{p}_i , we can construct a subspace \mathbf{V} from the degrees of freedom offered by the weighted affine transformations of the quantities \mathbf{p}_i according to transformations chosen at each of the k' handles and weights defined at the elements. The implicit assumption in this construction is that the *transformations of* the rest-shape's constraint projections vary smoothly across the mesh when considering deformations of the shape (note that we do *not* assume that the constraint projections themselves vary smoothly, which is not the case). The explicit construction of this space is explained in Appendix 3.C. As a linear subspace, we are unable to guarantee that all constraint projections of the approximated vectors will be on the constraint manifold, but by solving a fitting problem into this space (see next paragraph) we invoke this property in a least squares sense.

To avoid redundancy in the fitting problem and take into account the geometry of the mesh, we perform a weighted PCA on the space constructed above and only use the most significant half of the principal component vectors as the final basis \mathbf{V} . As weights we use the length, area or volume associated to the elements of this constraint type.

Constraint projections fitting Since \mathbf{V} is not built from snapshots of constraint projections vectors, but as a general purpose space, it is unsuited to compute a direct interpolation from a very small number of carefully selected elements as suggested by the DEIM method [4]. Instead, we solve a least-squares fitting problem from considerably more entries in the vector of constraint projections than there are basis vectors in \mathbf{V} . Therefore we choose $s \gg k'$ approximately equidistant constraints via the furthest point sampling method, starting with k' constraints next to the k' vertex samples used in the construction of \mathbf{V} . In our examples we choose between 800 to 1000 constraint samples, contributing 9 entries in the constraint projections vector each, whereas the dimension of the constructed constraint projections subspace is between 300 and 420, see Table 3.1.

Let $\mathbf{J} \in \mathbb{R}^{p \times s \times p \times e}$ be the matrix that maps a vector containing all constraint projections to a vector containing only the s sampled constraints' projections (e is the total number of constraints, and p denotes the number of rows of each constraint projection \mathbf{p}_i), and let

$$\mathbf{S}^T = (\lambda_1 \mathbf{S}_1^T \lambda_2 \mathbf{S}_2^T \dots) \quad (3.7)$$

be the summation matrix, which maps vectors $\mathbf{p} \in \mathbb{R}^{pe \times 3}$ of stacked constraint projections \mathbf{p}_i to the term $\sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$.

During precomputation a list of all vertices that appear in the evaluation of the constraint projections of the s sampled constraints is assembled. In the local phase, the first step is to evaluate these vertices via $\mathbf{q}_i = \mathbf{U}_i \tilde{\mathbf{q}}$ for the current subspace coordinates $\tilde{\mathbf{q}}$ of the system. This can be done in parallel and the vector-vector product above is sparse (due to our specific subspace construction). Then, for each sampled constraint i , we evaluate the corresponding constraint projection $\mathbf{p}_i(\mathbf{q})$ for the current deformation \mathbf{q} . We stack these in vectors $\mathbf{p}_{\text{partial}} \in \mathbb{R}^{p \times s \times 3}$. Thereafter, the least-squares fitting problem is solved by minimizing the residual

$$\|\mathbf{J}\mathbf{V}\tilde{\mathbf{p}} - \mathbf{p}_{\text{partial}}\|_F^2 \quad (3.8)$$

which amounts to solving the three uncoupled linear systems

$$\mathbf{V}^T \mathbf{J}^T \mathbf{J} \mathbf{V} \tilde{\mathbf{p}} = \mathbf{V}^T \mathbf{J}^T \mathbf{p}_{\text{partial}} \quad (3.9)$$

for each coordinate in parallel. Finally, $\tilde{\mathbf{p}}$ is directly mapped to the approximation of the term $\mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i(\mathbf{q})$ via $\mathbf{r} = \mathbf{U}^T \mathbf{S}^T \mathbf{V} \tilde{\mathbf{p}}$. The matrix $\mathbf{U}^T \mathbf{S}^T \mathbf{V}$ can be precomputed and is small (compared to the original dimension of the system).

3.6. HYPER-REDUCED PROJECTIVE DYNAMICS

After the subspaces \mathbf{U} and \mathbf{V} for vertex positions and constraint projections have been constructed, the sampled constraints have been selected and the fitting problem has been set up, we are able to use the hyper-reduced variant of Projective Dynamics listed in Algorithm 3. There we use the notation $\tilde{\cdot}$ for quantities projected into the subspace, such as \tilde{f}_{ext} being the external forces projected into the space spanned by \mathbf{U} .

The important differences to the original Projective Dynamics method are the modifications to the local and global steps: instead of evaluating all constraint projections, we evaluate a small subset and solve a least-squares fitting problem. Also, instead of solving a global system in the size of the number of vertices, we solve the reduced system to obtain subspace coordinates.

For brevity, Algorithm 3 does not include the possibility of adding additional, fully evaluated constraint projections to the system. This is entirely possible and was used in our experiments, for example when ‘hanging’ objects using a few position constraints. These are all evaluated in the local step and added to the right hand side of the global step individually, as in the unreduced method.

If the mesh needs to be displayed at the end of the time step, the updated subspace coordinates need to be mapped to full coordinates. For complex geometries (>15k vertices), we perform this step on the GPU. Additionally, partial position updates are performed using the GPU, *i.e.* mapping the subspace coordinates to the positions of only those vertices that appear in sampled constraint projections. Other than that we implemented this method using only CPU operations. More details on our implementation can be found in Appendix 3.A. There, we also specify how we perform the collision handling and user interaction mentioned in Algorithm 3.

Relation to reduced finite element methods For the presented hyper-reduction of Projective Dynamics, we focused on a specific class of potentials, for which the method is

Algorithm 3 Hyper-Reduced Projective Dynamics

Input: Current subspace coordinates $\tilde{\mathbf{q}}(t)$ and subspace velocities $\tilde{\mathbf{v}}(t)$

Let $\tilde{\mathbf{s}} = \tilde{\mathbf{q}}(t) + h\tilde{\mathbf{v}}(t) + h^2\mathbf{U}^T\mathbf{M}^{-1}\mathbf{U}\tilde{\mathbf{f}}_{\text{ext}}$

Modify $\tilde{\mathbf{s}}$ according to user interaction and collision constraints.

Let $\hat{\mathbf{q}} = \tilde{\mathbf{s}}$

for $i = 1$ **to** numIterations **do**

Local step:

Evaluate required vertex positions $\hat{\mathbf{q}}_{\text{partial}} = \mathbf{U}_{\text{partial}}\hat{\mathbf{q}}$

Evaluate sampled constraint projections $\mathbf{p}_{\text{partial}}$

Solve the fitting problem

$$\mathbf{V}^T\mathbf{J}^T\mathbf{J}\mathbf{V}\tilde{\mathbf{p}} = \mathbf{V}^T\mathbf{J}^T\mathbf{p}_{\text{partial}}$$

Build the right hand side term $\mathbf{r} := \mathbf{U}^T\mathbf{S}^T\mathbf{V}\tilde{\mathbf{p}}$

Global step: Solve

$$\mathbf{U}^T\left(\frac{\mathbf{M}}{h^2} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{S}_i\right)\mathbf{U}\hat{\mathbf{q}} = \mathbf{U}^T\frac{\mathbf{M}}{h^2}\mathbf{U}\tilde{\mathbf{s}} + \mathbf{r}$$

Let $\hat{\hat{\mathbf{q}}} = \hat{\mathbf{q}}$

end for

Output: Updated subspace coordinates $\tilde{\mathbf{q}}(t+h) = \hat{\hat{\mathbf{q}}}$ and velocities $\tilde{\mathbf{v}}(t+h) = (\tilde{\mathbf{q}}(t+h) - \tilde{\mathbf{q}}(t))/h$

particularly efficient. This limits the type of material response our method can achieve. For example, we can simulate elastic solids with material parameters controlled by Lamé parameters, such as the materials presented in [63], but we cannot handle general non-linear materials. Reduced finite element schemes like [7, 8, 39] can deal with various types of materials. Nevertheless, the benefits that we gain from our hyper-reduction scheme for Projective Dynamics are:

- The matrix for the global step is sparse and constant, hence a sparse factorization can be computed once and re-used throughout the whole simulation.
- The iterations for time-stepping do not involve a line search.
- For both linear solves appearing in our algorithm (constraint projections fitting and global step), the x , y , and z -coordinates are decoupled; hence systems of smaller size are solved in parallel.

As a result, we observe that our approach can achieve real-time rates of 60 fps in subspaces that are roughly one order of magnitude larger than what is reported for recent reduced finite element schemes [7, 8]. The larger subspaces allow us to gracefully handle features like user interaction, ground collisions, and drastic changes of material stiffness and volume to the real-time simulation. Another feature of the proposed scheme is the fast precomputation stage, see Table 3.1. In addition, the method profits from the robustness and generality of Projective Dynamics.

3.7. RESULTS

We implemented our Hyper-Reduced Projective Dynamics method for the simulation of triangle and tetrahedral meshes, supporting spring, strain (triangle and tetrahedron), bending, and position constraints, as well as collision handling and user interaction. Details on the implementation can be found in Appendix 3.A. The resulting method is capable of simulating highly detailed meshes with more than 400k vertices in real-time, while handling collisions and user interactions.

In Figure 3.3 three frames of the same simulation using full Projective Dynamics and using Hyper-Reduced Projective Dynamics are shown. The reduction parameters for this visual comparison are listed in Table 3.1. When using small time steps and watching the simulation in slow motion, differences in both simulations can be observed: in the full simulation (left), individual fingers and ears show richer dynamics, but we found that these differences are hard to spot once we choose a more realistic time step. While the full method offers higher detail in the finer dynamics, it results in 3.7 fps and is unsuited for real-time applications.

Additionally we simulated the dropping armadillo restricted to the degrees of freedom offered by our subspace, but without approximating the forces. The resulting dynamics are coarser than in the full simulation, but finer than in the hyper-reduced simulation. This shows that our subspace offers enough degrees of freedom to enable rich dynamics. However, not employing a force approximation scheme results in the same FPS as the full simulation: while the computation time of the global step is significantly reduced (in this example from 3100 microseconds to 70 microseconds), the cost of the evaluation of the right hand side term dominates the iteration times for both approaches. Moreover, evaluating the term in the subspace-only-reduction requires evaluating the current positions $\mathbf{q} = \mathbf{U}\tilde{\mathbf{q}}$ in *every iteration* as well as multiplying the vector $\sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$ by \mathbf{U}^T . This means that the cost for the local step is larger in the subspace-only-reduction, than in the full simulation. This shows the importance of employing a force approximation along with a dimension reduction.

Table 3.1 contains data for the experiments shown in Figures 3.1, 3.2 and 3.6. A full explanation for each line of the table is given in the next paragraph. Thereafter, we discuss precomputation and online timings and the tradeoff between computation timings and approximation errors when comparing simulations using our reduced method to full Projective Dynamics simulations.

Table details In Table 3.1 we show – for the experiments illustrated in the figures of this chapter – the type(s) of constraints used, the number of vertices of the mesh, the position subspace dimension (*i.e.* the degrees of freedom for the vertex positions), the original number of constraints, the number s of constraints that we evaluate for the constraint projections fitting method and the number of vertices whose position needs to be available when computing these constraint projections. We then list the resulting pre-computation times in seconds, both for our method as well as for the unreduced method (factorization of the l.h.s. matrix), followed by the computation times (in *microseconds*) of the local and global step of one iteration of the hyper-reduced method, as well as the time it takes to evaluate the vertex positions of the vertices required to evaluate the sampled constraints. The resulting fps for our hyper-reduced method (using ten iterations

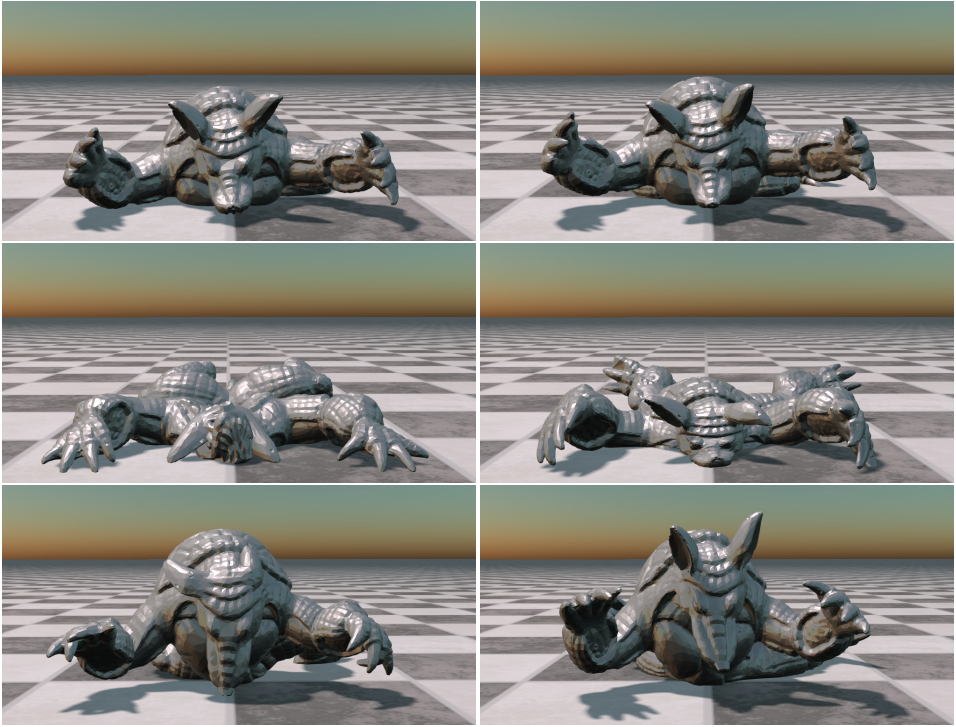


Figure 3.3: Visual comparison of simulation results from full Projective Dynamics (left column) and our method using 1440 degrees of freedom (right column) both starting with the same initialization.

per frame) are shown with and without taking into account that all vertex positions have to be evaluated once per frame if the mesh is rendered each frame. Finally, we show the fps of the unreduced method. In the following two paragraphs, we discuss the precomputation and online timings in more detail.

Precomputation times The setup of the Hyper-Reduced Projective Dynamics method encompasses

- the construction of the positions subspace, which includes
 - sampling k approximately equidistant vertex samples
 - constructing the k weight functions centered at those vertices
- the preparation of the constraint projections fitting method, which includes
 - constructing the subspace of unassembled constraint projections from k' approximately equidistant vertex samples
 - a PCA of that space
 - choosing s sampled constraints

Table 3.1: Data for the experiments shown in the figures of this chapter. See Section 3.7 for further details.

Name	Glove (Fig. 3.2)	Armadillo (Fig. 3.3 & 3.6)	Armadillo (large subspace)	Elephant (Fig. 3.1)	Dragon (Fig. 3.2)	Squid (Fig. 3.2)
Constraints type	Bend. & Tri. strain	Tet. strain	Tet. strain	Tri. & Tet. strain	Tet. strain	Tet. strain
# vertices n	24370	19064	19064	52812	196577	439061
# iterations per frame	10	10	10	10	10	10
Position subspace dimension	540	1440	3960	1200	1200	2160
Constraint projections sub.dim.	300	360	360	360	420	360
# constraints	73106	71289	71289	277363	733649	1664908
$s = \#$ evaluated constr. proj. (# of evaluated vertices)	2 · 1000 5771	970 3688	970 3688	2 · 800 3789	1000 2741	1000 3931
Precomputation (in seconds)	8.23	6.20	16.23	21.19	63.10	189.54
Local step (in μ s)	546	337	660	612	643	483
Global step (in μ s)	48	40	256	148	35	74
Partial update v. pos. (in μ s)	435	412	678	468	394	524
fps (fps incl. display)	94 (89)	120 (112)	63 (60)	81 (62)	89 (60)	116 (37)
Precomputation unreduced (in s)	0.56	0.29	0.29	1.368	11.83	12.57
fps unreduced	2.39	3.7	3.7	0.68	0.20	0.08

- the evaluation of all constraint projections of the rest shape
- the construction and factorization of the left hand side matrix of the constraint projections fitting problem
- the construction and factorization of the left hand side matrix for the global step.

The precomputation timings listed in Table 3.1 include all of the above steps. The timing depends on the number of vertices n of the original mesh, and the reduction parameters k , k' and s . As the vertex count increases, performing the weighted PCA in the construction of the constraint projections subspace becomes the most time consuming task. The precomputation times measured for the meshes in our experiments (ranging from 19k to 440k vertices) admit most types of applications and would rarely pose severe limitations. Note that all precomputed quantities are suited for arbitrary user interaction, collision constraints and external forces and the precomputation (for a specific mesh) only has to be performed again, when the material type changes, or stiffness weights change in a non-uniform way (*i.e.* by more than a constant factor).

Simulation times The online phase of our Hyper-Reduced Projective Dynamics method is made up of three steps:

- The first step (not present in the unreduced method), in which we evaluate the positions of the vertices that are required to evaluate the sampled constraint projections in the next iteration, since only subspace coordinates are available at that point.
- The local step, which encompasses evaluating the sampled constraint projections, solving a fitting problem and mapping it to the approximation of the right hand side term required for the global step.
- The global step, in which we solve the reduced linear system.

When the mesh is being displayed after each time integration step, the full vertex positions need to be evaluated once per frame via $\mathbf{q} = \mathbf{U}\bar{\mathbf{q}}$, which we perform on the GPU for

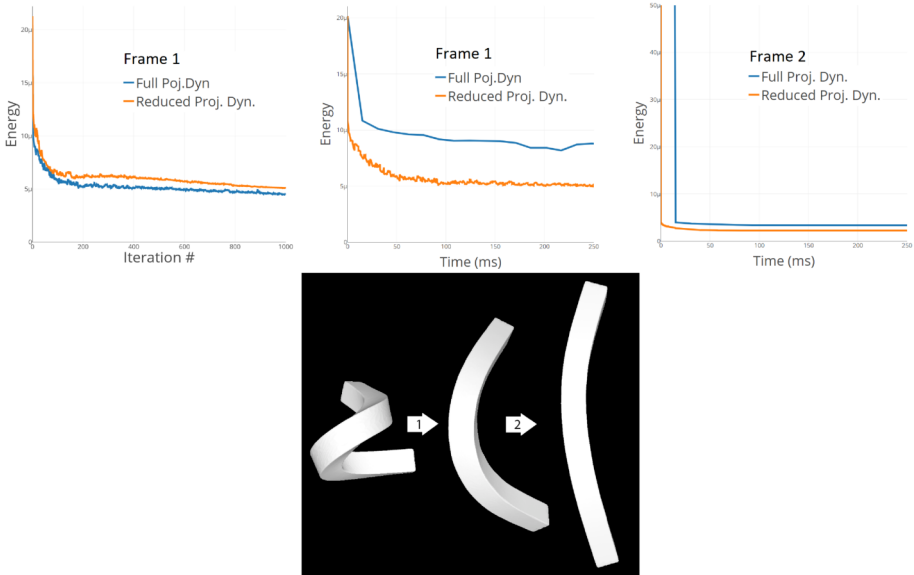


Figure 3.4: Progress plots displaying the evolution of the objective (3.2) during minimization using the local/global solver for the full and the reduced problem. Large time steps are taken and the resulting frames are shown on the left. For the first time step, the objective value versus iterations *and* time are shown. For the second timestep, objective value versus time are shown.

meshes with more than 15k vertices, and on the CPU, in parallel, otherwise. This step becomes the most time-consuming part of the simulation when the mesh resolution is high: for the Squid mesh (Figure 3.2, left) with 440k vertices, we can compute 116 time steps of the simulation per second (when using ten iterations per time step), but additionally evaluating the full positions \mathbf{q} to render the current state results in 37 fps. For all other examples, including the Dragon mesh with almost 200k vertices, we achieve 60 fps or more, including rendering, collision detection and user interaction. Our hyper-reduced method is able to handle large subspaces while still maintaining high frame rates: for the armadillo mesh we can use a subspace that offers 3960 degrees of freedom and still achieve 60 fps including rendering and collision detection, see Table 3.1.

Convergence In order to compare the convergence of both the full and our Hyper-Reduced Projective Dynamics methods, we simulate a volumetric elastic bar whose rest shape is an undeformed block, but is initially set to a heavily bended position, as seen in Figure 3.4, left. The internal forces lead to an unfolding of the bar and we choose a large time step such that the optimization problem (3.2) is sufficiently hard. Throughout the minimization we record the energy levels over time of both methods, see Figure 3.4, right. Both methods show similar behavior, in that they heavily reduce the energy in the first few iterations and then slowly converge to a lower optimum (for the second frame, the energy before the first iteration is out of range of the plot). While our method usually requires more iterations to converge to a lower energy level, it takes less time to do so

since the iterations are about 30 times faster. With higher resolutions of the mesh, this speed up factor becomes larger, *e.g.* for the Squid mesh we get a factor of 1440, using the reduction parameters listed in Table 3.1.

In all examples, we use 10 local/global iterations per time integration step. We find that for our hyper-reduced method the differences between a simulation running with 10 iterations and a fully converged solution are very small since the coupling of material stiffness and iteration count can only be observed at very low iteration counts.

Table 3.2: Normalized L_2 errors when projecting the frames of the full simulation shown in Figure 3.3, left, to subspaces of three different sizes constructed with our method.

Subspace dimension	Mean Error	Max. Error
360	0.0269	0.0513
1440	0.0106	0.0228
3960	0.0057	0.0112

Approximation errors When comparing the results of our hyper-reduced method to those of the full Projective Dynamics method, there are two main sources of approximation errors:

1. Errors due to the reduced degrees of freedom offered by our subspace for vertex positions.
2. Errors when approximating the constraint projections via our fitting method.

To measure the first type of approximation errors mentioned above, that is, to evaluate the degrees of freedom offered by our subspace, we want to quantify how well deformations of a mesh that is simulated in full detail can be approximated by our vertex position subspace. Therefore, we project each frame of the *full* simulation of a dropping armadillo (Figure 3.3, left) into subspaces constructed via the method described in Section 3.4. In Table 3.2 we list the mean and maximal L_2 differences between the full and the projected shapes of all frames of the simulation for three different subspace sizes (the original shape was normalized to have L_2 norm 1). The mean error does not include the first few frames of the simulation where the armadillo is still in its rest shape, which results in an approximation error of zero. Note, that even when running the simulation using our hyper-reduced method with the largest of the three subspaces, we still achieve 60 fps including rendering and collision detection (see Table 3.1).

The second type of approximation errors (approximated constraint projections) is measured as the relative errors between the right hand side term $\sum_i \lambda_i \mathbf{S}_i \mathbf{p}_i(\mathbf{q})$ to its approximation $\mathbf{S} \mathbf{V} \bar{\mathbf{p}}$, acquired from our constraint projections fitting method (see Section 3.5). We plot these errors for the dropping dragon simulation in Figure 3.5, where we evaluate only 1000 of the 733649 constraints to approximate the right hand side term.

Generality One of the reasons that we chose to reduce the Projective Dynamics method is its generality. It is able to handle thin shells, volumetric deformables, spring-based

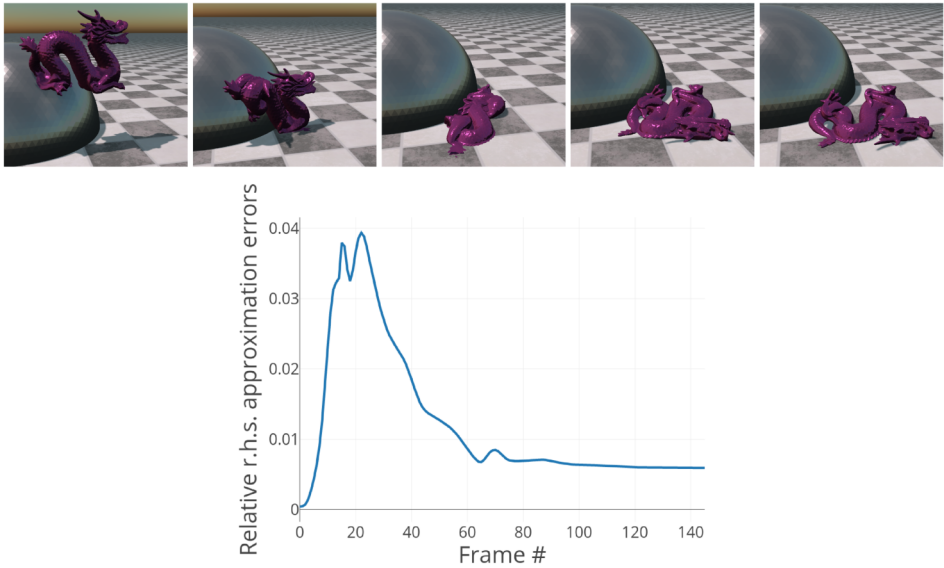


Figure 3.5: The dragon mesh is dropped, slides along a sphere and lands on the floor in a heavily deformed state from which it slowly recovers. For each frame of the simulation we plot the relative error of our approximation of the right hand side term from 1000 sampled constraint projections to the fully evaluated term from all 766k constraint projections.

materials, example-based materials, and more. In the simulations shown in the figures of this chapter, we show examples for various volumetric deformables (the armadillo, the dragon and the squid), a thin shell mesh (the glove) and a combination of surface strain and volumetric strain (the elephant). Frames of these simulations are shown in Figures 3.1 and 3.2. We show that we can handle different types of constraints simultaneously and change the target volume of the tetrahedrons, while limiting the strain on the outer triangles of the mesh during simulation, as seen in Figure 3.1. This demonstrates the range of flexibility that our hyper-reduced method is able to carry over from the Projective Dynamics method.

Moreover, we obtain additional flexibility via our subspace reduction: the system matrix for the global step is small (in comparison to the size of the full system) but still sparse (because of our specific subspace construction). This enables us to change constraint weights or add and remove constraints in the online phase of the simulation without dropping below 60 frames per second. This is shown in Figure 3.6, where we change the stiffness parameter in running simulations. For the unreduced method on the armadillo mesh, refactorizing the system matrix takes 108 milliseconds, which prohibits changes to the constraints amid simulation.

Robustness Our hyper-reduced method inherits the robustness properties of Projective Dynamics, which is, as an implicit variational integration scheme, unconditionally stable, and has been demonstrated to gracefully handle highly deformed states. In Figure 3.6 we show that our hyper-reduced method is able to restore a tetrahedral mesh

from a completely flattened state. Throughout our experiments we subjected the meshes to large external forces, rapid user interaction and collision constraints. The simulations showed stable behavior throughout.

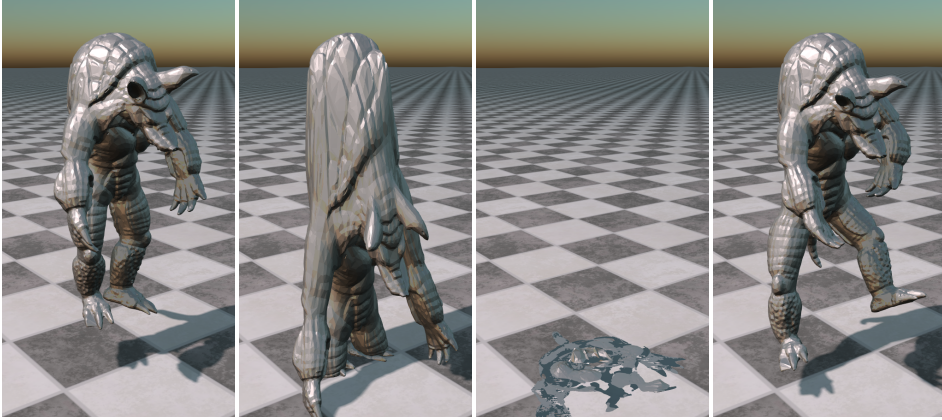


Figure 3.6: We are able to interactively change stiffness parameters in a running simulation. From left to right we scale the predefined weights by 1, 0.1, 0 and again 1. This also demonstrates that our method is able to recover from a fully flattened state.

3.8. CONCLUSION

We presented a method for real-time simulation of deformables that combines the benefits of Projective Dynamics and hyper-reduction techniques in one framework. The resulting scheme is robust, general and efficient. It enables real-time simulation of high-resolution meshes in specifically designed subspaces with up to 4k degrees of freedom while keeping precomputation times low. We provide examples that include deformable solids and shells, user interaction, collision constraints, external forces, varying material stiffness, and recovery from degenerate configurations.

Limitations and challenges While we handle collisions with rigid objects, one limitation of our current implementation is that self-collisions and collisions between multiple deforming meshes of the deformable object are not handled. It would be interesting to integrate techniques like *collision certificates* [64], *bounded-normal trees* [65] and *pose-space cubature* [66] to the proposed approach.

The subspaces used for positions and constraint projections are not directly suited for simulating cloth, since the assumption of deformations that vary continuously across the mesh is no longer valid: When bending resistance is very low or not simulated at all, sharp creases and noisy features start appearing, which can not be faithfully reconstructed using the presented subspaces. Here, performing full simulations and creating subspaces for positions and constraint projections from snapshots might prove beneficial. Initial tests showed that such subspaces are very specific to the task that was performed in the full simulation (*i.e.* how the cloth was grabbed and deformed and what

collision constraints and external forces were present) and do not enjoy the generality of our method.

The Projective Dynamics method has been generalized to handle fluids by Weiler et al. [22], but the loss of any connectivity renders our reduction method unable to handle particles. It is an interesting question whether the reduction can be adjusted such that it can be applied to Projective Fluids.

Lastly, the recent generalizations of Projective Dynamics to nonlinear constitutive materials in [25] and [23, 24] offer a direction to extend our hyper-reduced method in a similar fashion.

REFERENCES

- [1] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan, *Fast simulation of mass-spring systems*, ACM Trans. Graph. **32**, 214:1 (2013).
- [2] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, *Projective dynamics: Fusing constraint projections for fast simulation*, ACM Trans. Graph. **33**, 154:1 (2014).
- [3] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera, *An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations*, Comptes Rendus Mathematique **339**, 667 (2004).
- [4] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM Journal on Scientific Computing **32**, 2737 (2010).
- [5] A. Jacobson, I. Baran, J. Popović, and O. Sorkine, *Bounded biharmonic weights for real-time deformation*, ACM Trans. Graph. **30**, 78:1 (2011).
- [6] Y. Wang, A. Jacobson, J. Barbić, and L. Kavan, *Linear subspace design for real-time shape deformation*, ACM Trans. Graph. **34**, 57:1 (2015).
- [7] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, *An efficient construction of reduced deformable objects*, ACM Trans. Graph. **32**, 213:1 (2013).
- [8] X. Wu, R. Mukherjee, and H. Wang, *A unified approach for subspace simulation of deformable bodies in multiple domains*, ACM Trans. Graph. **34**, 241:1 (2015).
- [9] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, *Position based dynamics*, J. Vis. Comun. Image Represent. **18**, 109 (2007).
- [10] J. Stam, *Nucleus: Towards a unified dynamics solver for computer graphics*, in *Proc. IEEE International Conference on Computer-Aided Design and Computer Graphics* (2009) pp. 1–11.
- [11] M. Müller, B. Heidelberger, M. Teschner, and M. Gross, *Meshless deformations based on shape matching*, in *Proc. ACM SIGGRAPH* (2005) pp. 471–478.
- [12] A. R. Rivers and D. L. James, *Fastlsm: Fast lattice shape matching for robust real-time deformation*, ACM Trans. Graph. **26** (2007).

- [13] M. Müller, *Hierarchical position based dynamics*, in *Proc. Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (2008).
- [14] M. Müller and N. Chentanez, *Solid simulation with oriented particles*, *ACM Trans. Graph.* **30**, 92:1 (2011).
- [15] T.-Y. Kim, N. Chentanez, and M. Müller-Fischer, *Long range attachments - a method to simulate inextensible clothing in computer games*, in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012) pp. 305–310.
- [16] M. Macklin, M. Müller, and N. Chentanez, *Xpbd: Position-based simulation of compliant constrained dynamics*, in *Proc. ACM Motion in Games* (2016) pp. 49–54.
- [17] M. Macklin and M. Müller, *Position based fluids*, *ACM Trans. Graph.* **32**, 104:1 (2013).
- [18] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin, *Strain based dynamics*, in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2014) pp. 149–157.
- [19] J. Bender, D. Koschier, P. Charrier, and D. Weber, *Position-based simulation of continuous materials*, *Computers & Graphics* **44**, 1 (2014).
- [20] J. Bender, M. Müller, and M. Macklin, *Position-based simulation methods in computer graphics*, in *EUROGRAPHICS 2017 Tutorials* (2017).
- [21] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross, *Example-based elastic materials*, *ACM Trans. Graph.* **30**, 72:1 (2011).
- [22] M. Weiler, D. Koschier, and J. Bender, *Projective fluids*, in *Proc. ACM Motion in Games* (2016) pp. 79–84.
- [23] R. Narain, M. Overby, and G. E. Brown, *Admm \supseteq projective dynamics: Fast simulation of general constitutive models*, in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2016) pp. 21–28.
- [24] M. Overby, G. E. Brown, J. Li, and R. Narain, *Admm \supseteq projective dynamics: Fast simulation of hyperelastic models with dynamic constraints*, *IEEE Trans. Vis. and Comp. Graph.* **23**, 2222 (2017).
- [25] T. Liu, S. Bouaziz, and L. Kavan, *Quasi-newton methods for real-time simulation of hyperelastic materials*, *ACM Trans. Graph.* **36**, 23:1 (2017).
- [26] H. Wang, *A chebyshev semi-iterative approach for accelerating projective and position-based dynamics*, *ACM Trans. Graph.* **34**, 246:1 (2015).
- [27] A. Pentland and J. Williams, *Good vibrations: modal dynamics for graphics and animation*, in *Proc. of ACM SIGGRAPH* (1989) pp. 215–222.
- [28] M. G. Choi and H.-S. Ko, *Modal warping: Real-time simulation of large rotational deformation and manipulation*, *IEEE Trans. Vis. Comput. Graphics* **11**, 91 (2005).

- [29] J. Barbič and D. L. James, *Real-time subspace integration for St. Venant-Kirchhoff deformable models*, ACM Trans. Graph. **24**, 982 (2005).
- [30] J. Huang, Y. Tong, K. Zhou, H. Bao, and M. Desbrun, *Interactive shape interpolation through controllable dynamic deformation*, IEEE Transactions on Visualization and Computer Graphics **17**, 983 (2011).
- [31] Z. Pan, H. Bao, and J. Huang, *Subspace dynamic simulation using rotation-strain coordinates*, ACM Trans. Graph. **34**, 242:1 (2015).
- [32] Y. Yang, W. Xu, X. Guo, K. Zhou, and B. Guo, *Boundary-aware multidomain subspace deformation*, IEEE Trans. Vis. and Comp. Graph. **19**, 1633 (2013).
- [33] Y. Yang, D. Li, W. Xu, Y. Tian, and C. Zheng, *Expediting precomputation for reduced deformable simulation*, ACM Trans. Graph. **34**, 243:1 (2015).
- [34] C. Brandt and K. Hildebrandt, *Compressed vibration modes of deformable bodies*, Computer Aided Geometric Design **52–53**, 297 (2017).
- [35] P. Krysl, S. Lall, and J. E. Marsden, *Dimensional model reduction in non-linear finite element dynamics of solids and structures*, Int. J. Numer. Meth. Eng. **51**, 479 (2001).
- [36] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt, *Sparse localized deformation components*, ACM Trans. Graph. **32**, 179:1 (2013).
- [37] B. Gilles, G. Bousquet, F. Faure, and D. K. Pai, *Frame-based elastic models*, ACM Trans. Graph. **30**, 15:1 (2011).
- [38] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine, *Fast automatic skinning transformations*, ACM Trans. Graph. **31**, 77:1 (2012).
- [39] S. S. An, T. Kim, and D. L. James, *Optimizing cubature for efficient integration of subspace deformations*, ACM Trans. Graph. **27**, 1 (2008).
- [40] T. Kim and J. Delaney, *Subspace fluid re-simulation*, ACM Trans. Graph. **32**, 62:1 (2013).
- [41] D. Harmon and D. Zorin, *Subspace integration with local deformations*, ACM Trans. Graph. **32**, 107:1 (2013).
- [42] J. Barbič, F. Sin, and E. Grinspun, *Interactive editing of deformable simulations*, ACM Trans. Graph. **31**, 70:1 (2012).
- [43] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Modal shape analysis beyond Laplacian*, Computer Aided Geometric Design **29**, 204 (2012).
- [44] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, *Animating deformable objects using sparse spacetime constraints*, ACM Transactions on Graphics (TOG) **33**, 109:1 (2014).

- [45] S. Li, J. Huang, F. de Goes, X. Jin, H. Bao, and M. Desbrun, *Space-time editing of elastic motion through material optimization and reduction*, ACM Trans. Graph. **33**, 108:1 (2014).
- [46] H. Xu, Y. Li, Y. Chen, and J. Barbic, *Interactive material design using model reduction*, ACM Trans. Graph. **34**, 18:1 (2015).
- [47] J. N. Chadwick, S. S. An, and D. L. James, *Harmonic shells: A practical nonlinear sound model for near-rigid thin shells*, ACM Trans. Graph. **28**, 119:1 (2009).
- [48] F. Hahn, B. Thomaszewski, S. Coros, R. W. Sumner, F. Cole, M. Meyer, T. DeRose, and M. H. Gross, *Subspace clothing simulation using adaptive bases*, ACM Trans. Graph. **33**, 105:1 (2014).
- [49] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, *Subspace gradient domain mesh deformation*, ACM Trans. Graph. **25** (2006).
- [50] K. Hildebrandt and K. Polthier, *On approximation of the Laplace–Beltrami operator and the Willmore energy of surfaces*, Computer Graphics Forum **30**, 1513 (2011).
- [51] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, *Real-time nonlinear shape interpolation*, ACM Trans. Graph. **34**, 34:1 (2015).
- [52] P. von Radziewsky, E. Eisemann, H.-P. Seidel, and K. Hildebrandt, *Optimized subspaces for deformation-based modeling and shape interpolation*, Computers & Graphics **58**, 128 (2016).
- [53] C. Brandt, C. von Tycowicz, and K. Hildebrandt, *Geometric flows of curves in shape space for processing motion of deformable objects*, Computer Graphics Forum **35** (2016).
- [54] P. Musialski, T. Auzinger, M. Birsak, M. Wimmer, and L. Kobbelt, *Reduced-order shape optimization using offset surfaces*, ACM Trans. Graph. **34**, 102:1 (2015).
- [55] X. Chen, C. Zheng, and K. Zhou, *Example-based subspace stress analysis for interactive shape design*, IEEE Trans. Vis. and Comp. Graph. **23**, 2314 (2017).
- [56] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr, *Dynamic real-time deformations using space & time adaptive sampling*, in *Proc. ACM SIGGRAPH* (2001) pp. 31–36.
- [57] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović, *A multiresolution framework for dynamic deformations*, in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002) pp. 41–47.
- [58] C. Wojtan and G. Turk, *Fast viscoelastic behavior with thin features*, ACM Trans. Graph. **27**, 47:1 (2008).
- [59] L. Kharevych, P. Mullen, H. Owhadi, and M. Desbrun, *Numerical coarsening of inhomogeneous elastic materials*, ACM Trans. Graph. **28**, 51:1 (2009).

- [60] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure, *Preserving topology and elasticity for embedded deformable models*, ACM Trans. Graph. **28**, 52:1 (2009).
- [61] O. Rémillard and P. G. Kry, *Embedded thin shells for wrinkle simulation*, ACM Trans. Graph. **32**, 50:1 (2013).
- [62] M. Campen, M. Heistermann, and L. Kobbelt, *Practical anisotropic geodesy*, Computer Graphics Forum **32**, 63 (2013).
- [63] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, *A simple geometric model for elastic deformations*, ACM Trans. Graph. **29**, 38:1 (2010).
- [64] J. Barbič and D. L. James, *Subspace self-collision culling*, ACM Trans. Graph. **29**, 81:1 (2010).
- [65] S. C. Schwartzman, J. Gascón, and M. A. Otaduy, *Bounded normal trees for reduced deformations of triangulated surfaces*, in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009) pp. 75–82.
- [66] Y. Teng, M. A. Otaduy, and T. Kim, *Simulating articulated subspace self-contact*, ACM Trans. Graph. **33**, 106:1 (2014).
- [67] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, <http://eigen.tuxfamily.org> (2010).
- [68] L. Dagum and R. Menon, *Openmp: An industry-standard api for shared-memory programming*, IEEE Comput. Sci. Eng. **5**, 46 (1998).
- [69] J. Nickolls, I. Buck, M. Garland, and K. Skadron, *Scalable parallel programming with cuda*, Queue **6**, 40 (2008).
- [70] M. Garland and P. S. Heckbert, *Surface simplification using quadric error metrics*, Proc. ACM SIGGRAPH, 209 (1997).

APPENDIX

3.A. IMPLEMENTATION DETAILS

We implemented our Hyper-Reduced Projective Dynamics method using C++, specifically the *Eigen* library [67] to handle all linear algebra operations, OpenMP [68] to handle the parallel execution of constraint projections and CUDA [69] when mapping reduced coordinates directly to vertex positions in OpenGL buffers (this is only done for meshes with more than 15k vertices). Constraint projections were implemented as detailed in [2], with the exception of the bending projection, where we additionally ensure that the dot product between the outer normal of a fixed adjacent triangle and the mean curvature vector keeps the same sign throughout the simulation (this prevents the mesh from permanently inverting along edges where heavy buckling occurs).

User interaction Instead of using position constraints to handle user interactions, we simply modify the vector \mathbf{s} (which can be interpreted as the desired positions in the next time step, disregarding inner forces), by moving vertices close to the mouse parallel to the camera's viewing plane when click and drag actions are performed. We ignore vertices whose positions are not being evaluated (*i.e.* are not part of the sampled constraints), such that this operation remains independent of the full resolution as well. We then obtain the subspace vector $\tilde{\mathbf{s}}$ by interpolating the positions of all vertices on sampled constraints into the subspace. This method of handling user interaction also prevents us from having to introduce inactive position constraints which unnaturally change the system's behavior.

Collision handling In presence of collision constraints, one tries to minimize the energy stated in equation (3.2) subject to inequality constraints on the vertex positions, which come from anticipated collisions of the mesh with static objects. To circumvent both the detection of collisions for every vertex of the mesh in every local/global iteration, as well as adding temporary inequality constraints to the global step solve, we employ the following approximation of this optimization: In every frame of the simulation, once the reduced coordinates of the desired vertex positions $\tilde{\mathbf{s}}$ have been evaluated, we check, for one vertex at each sampled constraint, if there are violated collisions for this vertex and if so, compute the projection of this vertex to a collision free position. Note that the actual positions of these vertices needed to be evaluated anyway, since they are located at sampled constraints. In our reference implementation we used a simple projection method to obtain collision free positions for the vertices and to introduce repulsion and friction, we scale the tangential velocities and normal velocities to the collision plane of vertices that were not collision free by constant factors. This keeps the complexity all computation steps independent of the mesh resolution and offers an efficient and visually convincing way to let the mesh interact with static objects. Collisions with

thin features and sharp edges cannot be captured well by our approach, since the subset of vertices for which we check and resolve collisions is too coarse to support them. More involved collision resolving strategies and models for friction and repulsion can of course be applied, and, if desired, they can exploit the benefit of handling these for only a subset of the vertices and interpolating the effect via a projection to the subspace.

3.B. BLEND-SKINNING SUBSPACE CONSTRUCTION

Here we detail the construction of a subspace that mimics the degrees of freedom available in linear blend-skinning techniques. Given k handles with associated weights w_i^j per vertex and handle, a skinning transformation of a rest shape $\mathbf{q}_0 \in \mathbb{R}^{n \times 3}$ is obtained via

$$\begin{pmatrix} \hat{q}_i^x \\ \hat{q}_i^y \\ \hat{q}_i^z \end{pmatrix} = \sum_j w_i^j \mathbf{A}_j \begin{pmatrix} q_i^x \\ q_i^y \\ q_i^z \\ 1 \end{pmatrix}, \quad (3.10)$$

where the \mathbf{A}_j are the k chosen affine transformations (3×4 matrices) for each handle and (q_i^x, q_i^y, q_i^z) is the i -th row of \mathbf{q} . From this, one can deduce the subspace matrix

$$\mathbf{U}_j = \begin{pmatrix} q_0^x \cdot w_0^j & q_0^y \cdot w_0^j & q_0^z \cdot w_0^j & w_0^j \\ q_1^x \cdot w_1^j & q_1^y \cdot w_1^j & q_1^z \cdot w_1^j & w_1^j \\ \dots & \dots & \dots & \dots \end{pmatrix} \quad (3.11)$$

$$\mathbf{U} = (\mathbf{U}_1 \mid \dots \mid \mathbf{U}_k) \quad (3.12)$$

and interpret the entries of the affine transformations as subspace coordinates

$$\tilde{\mathbf{q}} = \begin{pmatrix} \mathbf{A}_0^T \\ \dots \\ \mathbf{A}_{k-1}^T \end{pmatrix} \quad (3.13)$$

Then, the transformations (3.10) can be concisely written as $\hat{\mathbf{q}} = \mathbf{U}\tilde{\mathbf{q}}$. In effect this means that for each handle, or set of weights, we get 4 subspace vectors and 12 degrees of freedom, as the subspace naturally decouples x , y and z coordinates, which goes along well with the decoupled system of the global step of Projective Dynamics.

3.C. SUBSPACE FOR CONSTRAINT PROJECTIONS

The construction above can be extended to subspaces for vectors of unassembled constraint projections. Here we assume the weights w_i^j are defined at the elements associated to the constraints for which this construction is used (e.g. at the tetrahedrons for volume preservation constraints). In addition, we have the evaluated constraint projections $\mathbf{p}_i(\mathbf{q}_0) \in \mathbb{R}^{p \times 3}$ of the rest shape at each element, which now replace the role of the vertex positions. That is, the subspace coordinates are given by the entries of k' transformations matrices $\mathbf{A}_j \in \mathbb{R}^{3 \times 4}$, from which we get new constraint projections via the transformations

$$\hat{\mathbf{p}}_i = \sum_j w_i^j \mathbf{A}_j \begin{pmatrix} \mathbf{p}_i \\ \mathbf{1} \end{pmatrix}, \quad (3.14)$$

where $\begin{pmatrix} \mathbf{p}_i \\ \mathbf{1} \end{pmatrix}$ is simply the $(p+1) \times 3$ matrix formed by attaching a row of ones to \mathbf{p}_i . Let

$$\mathbf{p}_i = \begin{pmatrix} p_i^{x,0} & p_i^{y,0} & p_i^{z,0} \\ \dots & & \\ p_i^{x,p-1} & p_i^{y,p-1} & p_i^{z,p-1} \end{pmatrix} \quad (3.15)$$

Then the subspace matrix \mathbf{V} is defined as follows:

$$\mathbf{V}_j = \begin{pmatrix} p_0^{x,0} \cdot w_0^j & p_0^{y,0} \cdot w_0^j & p_0^{z,0} \cdot w_0^j & w_0^j \\ \dots & & & \\ p_0^{x,p-1} \cdot w_0^j & p_0^{y,p-1} \cdot w_0^j & p_0^{z,p-1} \cdot w_0^j & w_0^j \\ p_1^{x,0} \cdot w_1^j & p_1^{y,0} \cdot w_1^j & p_1^{z,0} \cdot w_1^j & w_1^j \\ \dots & & & \end{pmatrix} \quad (3.16)$$

$$\mathbf{V} = (\mathbf{V}_1 \mid \dots \mid \mathbf{V}_k) \quad (3.17)$$

That is, $V \in \mathbb{R}^{ep \times 4k'}$, where e is the number of elements associated to the constraint projections, p is the size of the constraints and k' was the chosen number of handles. Then, we can write the transformations (3.14) as

$$\begin{pmatrix} \hat{\mathbf{p}}_0 \\ \dots \\ \hat{\mathbf{p}}_e \end{pmatrix} = \mathbf{V} \begin{pmatrix} \mathbf{A}_0^T \\ \dots \\ \mathbf{A}_{k'-1}^T \end{pmatrix} \quad (3.18)$$

3.D. COMPARISON TO SIMPLIFICATION SCHEMES

In the following we want to highlight the difference of our hyper reduction scheme to an approach in which the mesh is simplified, fully simulated and then the deformation is mapped to the original mesh using an *up-sampling matrix*.

Specifically, we simplify the surface of the original mesh using an algorithm similar to the method by [70], implemented such that all vertices of the simplified mesh lie on vertices of the original mesh. Using this correspondence, we create a matrix that maps displacements of the vertices of the simplified mesh to displacements of the vertices of the original mesh. There are many options to choose this matrix, and we found that no choice fixes the problems that we will address below. For the following experiment, as an upsampling matrix, we use the base matrix of our subspace construction, using the vertices of the simplified mesh as the samples on the original mesh from which the weights are computed (as described above, they are in direct correspondence). Note that we cannot add rotational degrees of freedom, since these are unknowns that cannot be determined from the vertex positions of the simplified mesh directly. This means, that the position of each vertex on the original mesh is a weighted combination of the positions of the vertices of the simplified mesh, where the weights are based on proximity to those samples.

In Figure 3.7, we show frames of the simulations using our hyper-reduction scheme and the simplification scheme described above. For our method, we use the parameters listed in Table 3.1 for the squid mesh. To make the comparison as fair as possible,

we used the same number of degrees of freedom for both our method and the method described above. That is, we simplify the surface mesh, such that the tetrahedralized version has 720 vertices (and 1490 tetrahedrons). This results in both methods having almost identical computation timings per iteration: Our method has some overhead compared to a full simulation on the same number of degrees of freedom due to solving the fitting problem (9), while the simulation on the simplified mesh has to evaluate significantly more constraint projections (1490 as opposed to 1000 for our method).

3

The experiment clearly demonstrates the issues with approaches that rely on mesh simplification. Firstly, the up-sampled results of the simulation on the simplified mesh suffer from strong linearization artifacts. Since no rotational degrees of freedom are solved for, and are thus unavailable, the linear relationship between the vertices of the simplified mesh and the original mesh becomes obvious. In contrast, we carefully choose degrees of freedom that can capture both local and global rotations and translations and solve the variational problem directly in these coordinates. This leads to an optimal solution to the equations of motion within the subspace, as opposed to an approximation that relies on a simulation on a simplified mesh. Secondly, all details that are no longer visible or well represented in the simplified mesh, will not be part of the simulation and do not show any elastic behavior. Instead they are simply displayed as stiff extensions of the rough features still visible on the simplified mesh. This is a fundamental difference to our approach of approximating the dynamics of the original mesh via the proposed constraint projections fitting method. This can be seen when closely the smaller features of the squid mesh that still show individual and elastic dynamics in our hyper reduced simulation.

Note, that our subspace construction is might not be the best suited method to generate a matrix for the up-sampling of vertex displacements and other approaches can yield better results. One problem of our approach is that we use the same support radius for the influence of each vertex of the simplified mesh. Therefore it is difficult to find a good trade-off between local influence and global coverage of the weights. The construction of a good up-sampling matrix is not trivial and not the goal of our approach. Moreover, the issues addressed above will hold for any type of linear up-sampling approach: detail that is not available on the simplified mesh can never take part in the reduced dynamics and a linear up-sampling approach can never be artifact free.

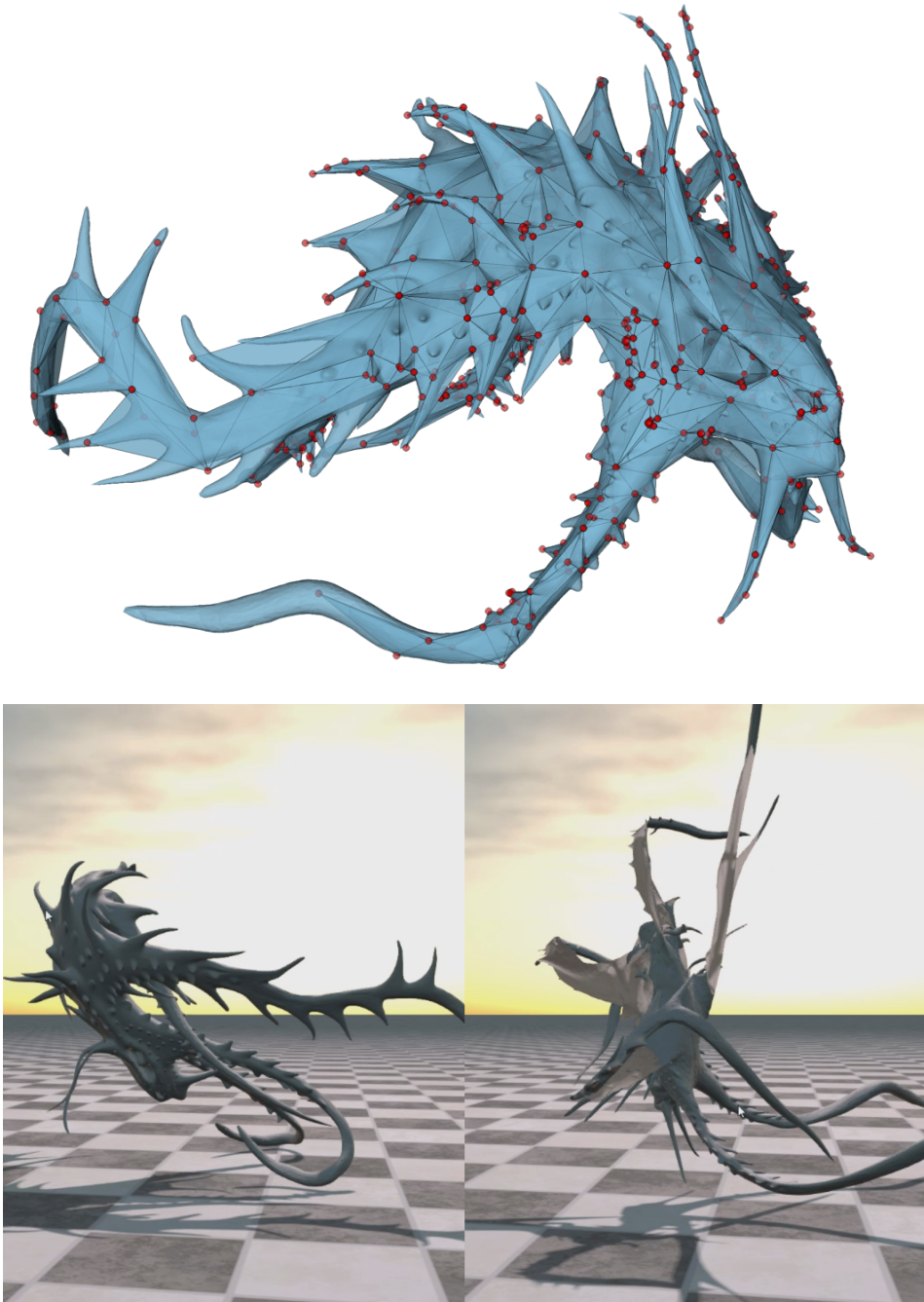


Figure 3.7: Top: The original squid mesh with the edges and vertices of the simplified mesh embedded. Bottom: Frames of the simulation using our hyper reduced scheme (left) and of the simulation using the simplified mesh, up-sampled to the vertices of the original mesh (right).

4

GEOMETRIC FLOWS OF CURVES IN SHAPE SPACE

This chapter is based on the publication *Geometric Flows of Curves in Shape Space for Processing Motion of Deformable Objects* by Christopher Brandt, Christoph von Tycowicz and Klaus Hildebrandt, published in *Computer Graphics Forum* in 2016.

4.1. OVERVIEW

Many problems in geometric modeling and more generally in graphics are dealing with deformable, flexible or non-rigid shapes. The idea of *geometric modeling in shape space*, introduced by Kilian et al. [1], is to equip the manifold of shapes relevant for a problem with a Riemannian metric and to use the resulting geometric structure on such a shape space for modeling tasks. A Riemannian metric on a shape space provides a quantitative measure for the deformation of shapes and concepts from Riemannian geometry, like the Riemannian exponential map and parallel transport, have been applied for designing powerful tools for modeling tasks such as shape deformation and interpolation, shape space exploration, deformation transfer, shape correspondences, and the design of measures of shape similarity. Due to the non-linear nature of shape spaces, geometric modeling in shape space leads to high-dimensional non-linear problems that have to be solved. For example, evaluating the distance between two shapes requires computing a geodesic in shape space. Therefore, efficient solvers for these optimization problems are of central importance.

The main contributions of this chapter are twofold. The first contribution is a novel approach for processing motion and animations of non-rigid shapes. We regard sequences of deformations of shapes as curves in shape space and use the geometric structure on shape spaces to transfer concepts from curve processing in \mathbb{R}^n to the processing of motion of deformable shapes. Following this principle, we introduce a geometric flow for curves in a shape space of meshes. The flow smooths a curve by decreasing its length in shape space. Our analysis of the flow shows that the limits are discrete geodesics in

shape space (as defined in [2]). The definition of the flow involves elastic shape averaging. In every iteration, every shape of the curve is replaced by a weighted average shape of the shape itself and its predecessor and successor. Since the limits are geodesics, the flow establishes a connection between shape averaging (or interpolation) and geodesics in shape space.

Based on the flow, we devise a scheme for the fairing of curves in shape space. The fairing scheme shortens the (shape space) length of the curve and thereby decreases the energy stored in the deformations between consecutive shapes. This means that the scheme is using knowledge of how elastic objects deform to faithfully filter the motion. For example, artifacts, like shrinkage of parts of an object, are avoided because the formation of such artifacts would require additional deformation energy. As of yet, no other temporal filter for mesh sequences with such properties has been introduced. We apply the scheme for removing jittering artifacts in motion capture data and for smoothing non-differentiable transitions that occur when concatenating different motions of an object.

The second main contribution is a model reduction approach for the efficient computation the flow. After a preprocess, the scheme has a computational cost that is independent of the (spatial) resolution of the meshes to be processed. Since the processing of sequences of meshes results in high-dimensional optimization problems, this method is essential for an efficient processing of curves in shape space. Moreover, the scheme provides a novel algorithm for the computation of geodesics in shape space. Compared to timings reported in previous work, this algorithm significantly accelerates the computation. Additionally, it allows for computing geodesics with much higher temporal resolution than previous approaches, which is due to the fact that our scheme performs only local operations in the temporal domain and the dimensional reduction in the spatial domain. Our experiments indicate that the combination of spatial reduction and higher temporal resolution yields a better approximation of the continuous geodesics.

We use the fast computation of geodesics for constructing nonlinear “Bézier curves” in shape space that are controlled by sets of poses. The curves are generated by applying de Casteljau’s algorithm in shape space. This example follows the construction of Bézier curves in shape spaces of images that was recently introduced by Effland et al. [3].

Using Riemannian geometry on shape spaces for geometric modeling tasks is a powerful concept. Crucial for these methods is the efficient computation of geodesics. We are convinced that the proposed reduced-order method (as it allows for faster computation and for higher spatial and temporal resolutions) is a step forward in this development.

4.2. RELATED WORK

Riemannian metrics on shape spaces of curves proved to be effective for various problems in Computer Vision. We refer to the textbook of Younes [4] for an in-depth discussion.

Riemannian metrics on the shape space of triangular surface meshes (with a fixed connectivity) have been introduced by Kilian et al. [1]. These metrics measure the stretching of the edges of the triangles, hence, the metric distortion of the surface. Heeren et al. [2, 5] propose a metric that in addition to the metric distortion measures the change



Figure 4.1: An illustration of a discrete geodesic in shape space consisting of 256 poses of a shape with 40k vertices is shown.

of bending of the surface. Their framework includes a model of elastic materials, which leads to Riemannian metrics on spaces of elastic shells. A Riemannian metric on the space of elastic solids was introduced by Wirth et al. [6]. Kurtek et al. [7, 8] introduce Riemannian metrics on spaces of surfaces parametrized over the unit sphere. Berkels et al. [9] introduce an approach for computing geodesic regression curves in shape spaces. Important for the application is the efficient computation of the geodesics between pairs of points in these shape spaces. This requires solving non-linear optimization problems, which are high-dimensional as the search space consists of curves in shape space. Specialized multi-grid Newton's solvers have been developed for this problem in [1, 2].

The problem of shape interpolation (or averaging, morphing, blending) has many applications in graphics. Early work concerned the morphing of planar shapes [10, 11]. For recent work on the interpolation of planar shapes, we refer to [12] and references therein. Approaches for the interpolation of surface meshes are either based on lin-

earized deformation models, like Poisson reconstruction [13, 14] and linear rotation invariant coordinates [15], or on non-linear deformation models [16–20]. Linearized deformation models are limited to small deformations, see [21] for a detailed discussion. Fast approximation algorithms for the shape interpolation problem have been proposed. Fröhlich and Botsch [18] use a combination of mesh coarsening and deformation-transfer to avoid solving the shape interpolation problem for the fully-resolved surfaces. A model reduction approach that yields real-time computation times for shape interpolation has recently been introduced by von Tycowicz et al. [20]. Compared to the Riemannian structure on the shape space, shape interpolation is a simpler concept. For example, elastic shape averaging does not lead to a distance measure that satisfies the triangle inequality [22]. A comparison of results of shape interpolation techniques and geodesics can be found in Section 4.7. Since our discrete flow is based on shape interpolation and has geodesics in shape space as its limits, this chapter establishes a connection between shape interpolation and geodesics, which were separate concepts before.

Smoothing filters for mesh sequences are typically applied directly to the trajectories of the individual vertices. Vlastic et al. [23] use bilateral filter in the temporal domain for each of the vertex trajectories. Li et al. [24] smooth the frames of an animation using a mix of constraints from points on the current, next and previous frames. These filters smooth the motions of the individual vertices, but neglect the shape formed by the set of vertices. Thus they are unable to prevent unnatural deformations of the shape. An example is shown in Section 4.7.

Related to shape interpolation and geodesics in shape space is the problem of keyframe interpolation in computer animation. The spacetime constraints paradigm, introduced by Witkin and Kass [25], provides a variational framework for physically-based keyframe interpolation. The goal is to help animators in creating plausible motion by combining physical simulation with keyframe interpolation. Spacetime optimization of the motion of deformable objects has been considered in [26, 27]. Recently, schemes for interactive editing of simulations and animations [28, 29] and for creating motion using sets of partial keyframes [30] have been proposed.

4.3. BACKGROUND: DEFORMATION ENERGIES AND SHAPE AVERAGING

In this section, we briefly introduce deformation energies and the elastic shape averaging. Our presentation restricts to the discrete case. For an introduction to elasticity, we refer to [31] and for a background on elastic shape averaging to [20, 32].

Discrete deformation energies We consider a deformable object consisting of a hyperelastic material. A material is elastic if the (inner) forces acting on the object depend only on the current configuration and are independent of the deformation path and speed. This means that the forces can be described by a vector field on the space of configurations of the object. The material is hyperelastic if this field is conservative. Then, the function whose negative gradient equals the force field, is the deformation energy. This function is only determined up to a constant. The constant is chosen such that the neutral configuration stores no deformation energy.

In this chapter, we restrict our attention to the discrete setting and consider triangle meshes for representing elastic shells and tetrahedral meshes for elastic solids. After the choice of materials and a discretization, the discrete deformation energy is a function

$$E : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}.$$

Here n is the number of degrees of freedom of the discrete object. In our setting, we keep the connectivity of the meshes fixed and n is three times the number of vertices. The first entry specifies the neutral configuration and the second the deformed configuration. The value $E(\bar{x}, x)$ measures the energy stored in the deformable object when it is deformed from the neutral configuration \bar{x} to the configuration x . In our experiments, we are using *Discrete Shells* [33]. However, other deformation energies like *PriMo* [34], *As-Rigid-As-Possible* [35, 36], or finite elements discretizations of elastic solids or shells could be used as well.

Elastic shape averaging We consider a set of $\mu + 1$ example configurations $\{y_0, y_1, \dots, y_\mu\}$ and positive weights $\{\omega_0, \omega_1, \dots, \omega_\mu\}$. Elastic shape averaging, introduced by Rumpf and Wirth [32], provides a way to compute weighted averages of the examples. The weighted average shape is defined as the configuration that minimizes the weighted sum of the energies $E(y_i, y)$ and $E(y, y_i)$

$$A(\omega_0, y_0, \dots, \omega_\mu, y_\mu) = \operatorname{argmin}_{y \in \mathbb{R}^n} \sum_{i=0}^{\mu} \omega_i (E(y_i, y) + E(y, y_i)). \quad (4.1)$$

This is a non-linear and elasticity-based approach for shape averaging that has a number of desirable properties. For example, the scheme can deal with larger deformations and the weighted average shape does not change if the example shapes are rigidly transformed.

4.4. DISCRETE CURVE FLOW IN SHAPE SPACE

In this section, we introduce a curve smoothing flow in shape space and discuss its application to the fairing of curves in shape space.

Curve smoothing flow in shape space We consider a discrete curve in shape space given by a sequence of $m + 1$ configurations (x_0, x_1, \dots, x_m) . The curve can either be closed or have a boundary. In the latter case, we fix the first and last configurations. In the case of a closed curve, the indices are to be read modulo $m + 1$.

The discrete curve smoothing flow is defined by the iterative procedure

$$\begin{aligned} x_i^0 &= x_i \\ x_i^{k+1} &= A\left(\frac{\tau}{2}, x_{i-1}^k, 1 - \tau, x_i^k, \frac{\tau}{2}, x_{i+1}^k\right). \end{aligned} \quad (4.2)$$

The parameter $\tau \in (0, 1]$ controls the size of the steps. It can be a fixed value or varied in every step. As we will discuss in the next section, controlling the stepwidth allows to guarantee that every step decreases an energy whose minimizers are discrete geodesics

in shape space. In every iteration, the flow deforms every shape of the curve towards an average of the shape itself and its two neighbors and thereby smoothes the deformations between successive shapes of the curve.

Curve fairing in shape space The smoothing flow combines two smoothing effects. It decreases the (shape space) length of the curve (this is discussed in the following section) and it regularizes the parametrization by equalizing the lengths of the individual segments. Decreasing the shape space length smoothes the curve in a way that avoids the formation of artifacts (like linearization artifacts or shrinkage of parts of the shape) because this would require additional deformation energy and hence make the curve longer. For example, the limit of a closed curve is a static “mean” shape (Figure 4.2 illustrates this). The second effect means that the curves evolve towards a more uniform motion in which the deformations between successive shapes store the same amount of energy. If this effect is not desired, it can be reduced by altering the weights for the shapes x_{i-1}^k and x_{i+1}^k in (4.2). For example, one can use the weights

$$\frac{\tau l_{i-1}}{l_{i-1} + l_i} \quad \text{and} \quad \frac{\tau l_i}{l_{i-1} + l_i}$$

where $l_i = \sqrt{E(x_i, x_{i+1})}$, in order to better preserve the original proportions of the energy stored in the deformations between successive shapes.

4.5. ANALYSIS OF THE FLOW AND THE COMPUTATION OF GEODESICS

In this section, we analyze the relation of the smoothing flow and geodesics in shape space. First, we show that the stationary points of the flow are discrete geodesics. Then, we prove that the flow decreases an energy whose minimizers are discrete geodesics (as defined in [2]). As a consequence, the discrete flow can be used for the computation of discrete geodesics.

Stationary points of the flow As a first step, we characterize the stationary points of the flow in the following lemma. By $\partial_1 E(x, y)$ and $\partial_2 E(x, y)$ we denote the derivatives of the energy E with respect to the first and the second argument.

Lemma 3 *A stationary point (x_0, x_1, \dots, x_m) of the discrete flow (4.2) satisfies*

$$\partial_1 E(x_i, x_{i-1}) + \partial_2 E(x_{i-1}, x_i) + \partial_1 E(x_i, x_{i+1}) + \partial_2 E(x_{i+1}, x_i) = 0 \quad (4.3)$$

for all interior shapes x_i .

Proof. Assume (x_0, x_1, \dots, x_m) is a stationary curve. This means

$$x_i = A\left(\frac{\tau}{2}, x_{i-1}, 1 - \tau, x_i, \frac{\tau}{2}, x_{i+1}\right)$$

for all (interior) i . Using (4.1), we see that x_i has to satisfy

$$\begin{aligned} \frac{\tau}{2} (\partial_1 E(x_i, x_{i-1}) + \partial_2 E(x_{i-1}, x_i)) + (1 - \tau) (\partial_1 E(x_i, x_i) + \partial_2 E(x_i, x_i)) \\ + \frac{\tau}{2} (\partial_1 E(x_i, x_{i+1}) + \partial_2 E(x_{i+1}, x_i)) = 0. \end{aligned}$$

This implies (4.3) since $E(x, x) = 0$ for all configurations x . \square

Limits are discrete geodesics in shape space Different Riemannian metrics on the spaces of shapes have been defined. We consider the physically-based metric introduced by Heeren et al. [2]. It uses viscous dissipation required to deform physical objects for measuring the distance of shapes. After a spatial discretization (which is the setting considered here) the discrete geodesics (x_0, x_1, \dots, x_m) are defined as the minimizers of the functional

$$\sum_{i=1}^m (E(x_{i-1}, x_i) + E(x_i, x_{i-1})) \quad (4.4)$$

for fixed configurations x_0 and x_m and with respect to variations of the other configurations x_i . The relation of the energy (4.4) to the Riemannian distance on the (continuous) shape space is that $E(x_i, x_{i+1})$ is a second-order approximation of the squared Riemannian distance between x_i and x_{i+1} , see [2]. The Euler-Lagrange equation satisfied by the minimizers of (4.4) is exactly equation (4.3), which is satisfied by the stationary points of the discrete flow.

This shows that discrete geodesics are stationary points of the flow. However, this does (in general) not guarantee that curves evolve towards geodesics. The following lemma shows that for every configuration of the curve, a small enough step of the flow decreases the energy (4.4). As a consequence, if we control the stepwidth, the limits of the flow are discrete geodesics in shape space.

Lemma 4 *For any non-stationary curve (x_0, x_1, \dots, x_m) and small enough $\tau > 0$, an iteration of the flow (4.2) decreases the energy (4.4).*

Proof. Let us consider τ as a variable and denote the next iterate by $(x_0^+(\tau), x_1^+(\tau), \dots, x_m^+(\tau))$. To prove the claim, we show that the derivative

$$\frac{\partial}{\partial \tau} (x_0^+(\tau), x_1^+(\tau), \dots, x_m^+(\tau)) \quad (4.5)$$

at $\tau = 0$ points into a descent direction of the energy (4.4). From the definition of the flow, (4.2), it follows that $(x_0^+(\tau), x_1^+(\tau), \dots, x_m^+(\tau))$ satisfies

$$\begin{aligned} \frac{\tau}{2} (\partial_1 E(x_i^+(\tau), x_{i-1}) + \partial_2 E(x_{i-1}, x_i^+(\tau))) + (1 - \tau) (\partial_1 E(x_i^+(\tau), x_i) \\ + \partial_2 E(x_i, x_i^+(\tau))) + \frac{\tau}{2} (\partial_1 E(x_i^+(\tau), x_{i+1}) + \partial_2 E(x_{i+1}, x_i^+(\tau))) = 0 \end{aligned} \quad (4.6)$$

We use these equations as an implicit function that determines $x_i^+(\tau)$. To compute (4.5), we need the derivatives of the left-hand side of (4.6) at $\tau = 0$ with respect to τ and x_i^+ . The former is

$$\begin{aligned} \frac{1}{2} (\partial_1 E(x_i^+(0), x_{i-1}) + \partial_2 E(x_{i-1}, x_i^+(0))) \\ + \partial_1 E(x_i^+(0), x_{i+1}) + \partial_2 E(x_{i+1}, x_i^+(0)) \end{aligned} \quad (4.7)$$

and the latter is

$$\partial_1 \partial_1 E(x_i^+(0), x_i) + \partial_2 \partial_2 E(x_i, x_i^+(0)). \quad (4.8)$$

The configuration $x_i^+(0)$ equals x_i . Plugging this into (4.7) and (4.8), we see that (4.7) is the gradient direction of (4.4) at x_i . Furthermore, the matrices $\partial_1 \partial_1 E(x_i, x_i)$ and $\partial_2 \partial_2 E(x_i, x_i)$ are positive definite (modulo rigid transformation of the shape) because $E(x_i, x_i)$ is a minimum of E for variations of the first and of the second argument. This implies that (4.8) is positive definite. The implicit function theorem implies that the derivatives in (4.5) satisfy

$$\begin{aligned} & (\partial_1 \partial_1 E(x_i, x_i) + \partial_2 \partial_2 E(x_i, x_i)) \frac{\partial}{\partial \tau} x_i^+(0) \\ &= \frac{1}{2} (\partial_1 E(x_i, x_{i-1}) + \partial_2 E(x_{i-1}, x_i) + \partial_1 E(x_i, x_{i+1}) + \partial_2 E(x_{i+1}, x_i)). \end{aligned} \quad (4.9)$$

Since the right-hand side is the gradient direction of (4.4) and the matrix $\partial_1 \partial_1 E(x_i, x_i) + \partial_2 \partial_2 E(x_i, x_i)$ on the left-hand side is positive definite, $\frac{\partial}{\partial \tau} x_i^+(0)$ points into a descent direction of (4.4). This implies that for small enough τ an iteration of the flow will decrease the energy (4.4). \square

4

4.6. EFFICIENT COMPUTATION OF THE FLOW

Integrating the discrete flow requires solving a number of shape averaging problems. We use a reduced-order technique for this, which combines dimensional reduction in the spatial domain and a scheme for the efficient evaluation of the reduced deformation energy and its gradient. Before we introduce the reduction strategy, we first discuss an asymmetry in the elastic potential and its effect on the definitions of elastic shape averaging and geodesics in shape space.

Remark on shape averaging and geodesics The elastic potentials are not symmetric in their two arguments, *i.e.* in general we have $E(x, y) \neq E(y, x)$. This means the energy stored in an object with rest shape y that is deformed into configuration x is not the same as the energy stored in an object with rest shape x that is deformed into configuration y . Because of this asymmetry, we have defined the elastic shape averaging using both terms $E(y, y_i)$ and $E(y_i, y)$ in (4.1). As an alternative, one can use only one of the terms to define the averaging. For example, in [20, 32] only the terms $E(y_i, y)$ are used. Then the averaging is

$$A(\omega_0, y_0, \dots, \omega_\mu, y_\mu) = \operatorname{argmin}_{y \in \mathbb{R}^n} \sum_{i=0}^{\mu} \omega_i E(y_i, y). \quad (4.10)$$

In the same spirit, we define the geodesics in shape space using $E(x_{i-1}, x_i)$ and $E(x_i, x_{i-1})$ in (4.4). In [2], only the terms $E(x_{i-1}, x_i)$ were used to define geodesics. This introduces a slight asymmetry in the definition. The discrete geodesic from shape x to y is not exactly the same as that from shape y to x . However, the difference is small and reduces with temporal refinement of the geodesic.

We have used the symmetric definitions involving both of the energy terms for shape averaging and geodesics in shape space in Sections 4.4 and 4.5 because in this case the connection between averaging and geodesics in shape space established by the proposed discrete smoothing flow is exact. The geodesics are exactly the limits of the flow. This makes the presentation simpler and easier accessible.

In our experiments, we have not noticed significant differences between the results obtained with the different definitions, which matches the observations reported in [32]. Experiments concerning the issue can be found in Table 4.2 in Section 4.7. The computation, of course, is faster if only one of the energy terms is used. Therefore, we used (4.10) for shape averaging in most of our experiments.

Dimensional reduction For the dimensional reduction, we are restricting the variations of every shape to a low-dimensional affine subspace of \mathbb{R}^n . We have used two subspace constructions for our experiments. If the input is a curve in shape space, a good candidate is the affine span of all shapes of the curve. This space can be represented as a linear space attached to one of the shapes. To further reduce the dimension, we select one shape x_σ and compute a principle component analysis (PCA) of the displacement vectors to all other shapes. The new affine space is the d -dimensional linear subspace spanned by the left singular vectors (of the matrix formed by the displacement vectors) with the highest singular value attached to x_σ . This affine space does not contain all shapes anymore. Therefore, we use for every shape x_i the affine span of this space and the vector representing x_i itself. To represent these spaces, we only need to store one subspace basis, which is augmented with the missing vector (the difference of the shape x_i and x_σ) for each shape at runtime. The additional vectors can be orthonormalized against the basis in the preprocess.

If the curve to be processed is very coarse, *e.g.* less than 20 frames, using on the affine span of the shapes would provide enough flexibility. In such a case, we use the flow tangent directions, $\frac{\partial}{\partial t} x_i^+(0)$ in equation (4.9) for every shape as an additional input for constructing the space. In the case, that only two shapes x and y are given and we want to compute a geodesic joining them, we follow the subspace construction in [20]. The starting point is the affine space spanned by the two shapes. This space is enriched with additional vectors. First, two vectors, v_1 and v_2 , obtained from linearizing the deformation from x to y and from y to x are computed. Then, further vectors are generated using a Krylov sequences that involves the Hessians of the elastic potentials, the mass matrices and v_1 and v_2 . For details, we refer to the original work.

Energy and force approximation In addition to dimensional reduction, we are using a scheme for the efficient approximation of the reduced energy and force. Since we are working in a reduced space, the deformations of the individual vertices are correlated. The approximation schemes aim at exploiting this structure. Different schemes have been proposed in the literature.

We adapt the mesh coarsening technique introduced in [37] to our setting. The idea is to create a coarse mesh and a subspace \tilde{V} for the coarse mesh that is isomorphic to the subspace V for the fine mesh. To approximate the energy at a point in V , the energy of the coarse mesh at the corresponding point in \tilde{V} is evaluated. To evaluate the gradient of the energy, the gradient of the coarse mesh is projected onto \tilde{V} . The corresponding vector in V is the vector that has the same reduced coordinates (however this vector is not explicitly computed, as we only work with the reduced coordinates). Explicitly, we use an edge-collapse scheme to generate a coarse version \tilde{x}_σ of the selected shape x_σ . Edge-collapse schemes implicitly generate a map from the vertices of the fine to the vertices

Table 4.1: Data for the experiments. $|V|$ = size of subspace, T_s = seconds per step on average, T_p = seconds for precomputation, *: flow with restoring force.

Animation	#verts.	#verts. ghost	#shapes	$ V $	#steps	T_s	T_p
Twisting Block	450	(not used)	6	24	10	0.26	3.44
Bending Block	450	(not used)	155	20	10/50/200	1.02	0.16
Centaur	15768	1252	138	15	50	2.13	12.08
Finger Geodesic (short)	2046	(not used)	5	14	214	0.03	3.51
Finger Geodesic (long)	2046	1252	81	7	115	0.17	3.48
Elephant Geodesic	39969	1246	256	14	270	1.09	82.18
Hand Linear Artifacts	6094	1252	49	14	100	0.92/0.94*	0.56
Hand Temporal Noise	6094	1252	49	14	100	0.91/0.94*	0.54
Motion Capture	2502	1252	91	20	50	1.81	0.34

of the coarse mesh, see [37] for details. We use this map to get subspace basis vectors for the coarse mesh.

An alternative approach for reduced force approximation is the optimized cubature [38, 39]. A second alternative is polynomial restriction [40], which allows for exact evaluation of a finite elements discretization of St. Venant–Kirchhoff materials for elastic solids at costs depending only on the subspace dimension. By combining the dimensional reduction and the reduced energy and force approximation, we obtain a computational cost for the integration of the flow that is independent of the resolution of the meshes.

Solving the reduced problem To solve the reduced problems, we use the BFGS scheme (see [41]). This is a quasi-Newton scheme that maintains an approximation of the inverse Hessian of the objective functional. Approximating the inverse Hessian avoids costly solving of a linear system to get the Newton direction. It is efficient to initialize the BFGS scheme with an inverse Hessian approximation to get a warm start. In the pre-process, we once compute the inverse Hessian (of the energy of the ghost mesh) of the mean shape of the predecessor and successor (on the initial curve) for every shape.

For the computation of the geodesic between two shapes, we use a coarse-to-fine strategy in the temporal domain. Starting with the two boundary shapes, we perform a two-step procedure. First the temporal domain is refined by inserting a fixed number (two or three in our experiments) of new shapes between every pair of successive shapes x_i and x_{i+1} . These shapes are initialized as interpolating shapes between x_i and x_{i+1} . Secondly, the geometric flow (4.2) is iterated until the squared norm of the reduced gradient of (4.4) is below 1^{-8} times the degrees of freedom.

4.7. APPLICATIONS AND EXPERIMENTS

Details of the experiments conducted in this chapter and computation times are shown in Table 4.1. The implementation was done in Java and the experiments were performed on a custom laptop (Intel Core i7-4600U, 2.1GHz). The precomputation times shown in the table (T_p) include the construction of the subspace, the initialization of the deformation energies and the initialization of the minimizer (computing initial Hessian approximations as a warm start for the BFGS minimizer). The precomputation time is

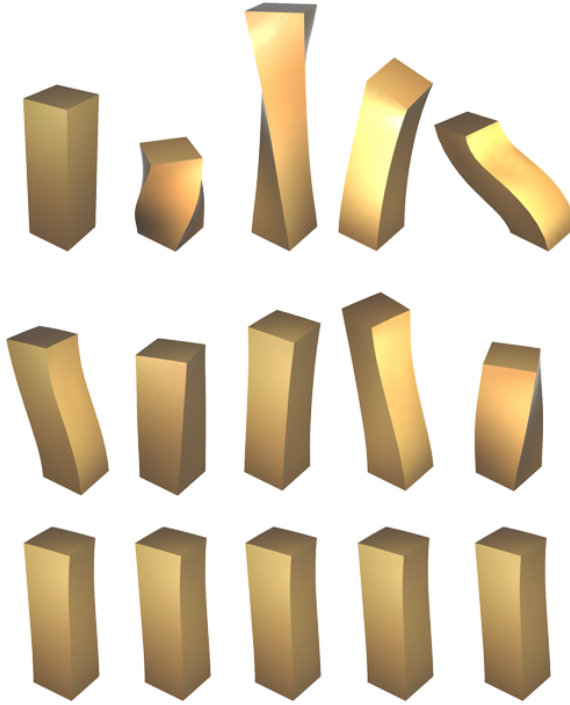


Figure 4.2: A periodic curve evolves to a constant curve. Top: Original sequence, middle: after 1 smoothing step, bottom: after 10 smoothing steps.

significantly lower when the subspace can be constructed from a PCA on the shapes of the input curve, and not via the more involved subspace construction from [20]. For our experiments, we used the Discrete Shells energy [33] as the elastic energy E , where we set the parameters to $k_B = 1$ and $k_L = k_A = 1/2$ (following the notation from that paper).

Basic examples The twisting block sequence (Figure 4.2) is meant as a simple demonstration of how a periodic sequence converges to a single point (or constant curve) when we perform several smoothing steps, akin to the contraction of closed curves to single points under the curve shortening flow. Since the original sequence consists of few shapes, the limit is reached after a few iterations (explicitly after 10 iterations with $\tau = 0.6$).

We demonstrate the ability of our smoothing technique to get rid of sudden jumps in the object's velocity (*i.e.* C^1 -discontinuities in the temporal domain): the bending block sequence shows a block starting from a bent-over position, getting into an upright position and back into a bent-over position, this time bending to a different direction (cf. Figure 4.3). The animation has a visible jump in the velocity around the frame where the block stands upright, since there is a sudden change in the direction of the motion. Applying smoothing steps to this animation leads to a smoother motion: the block does not

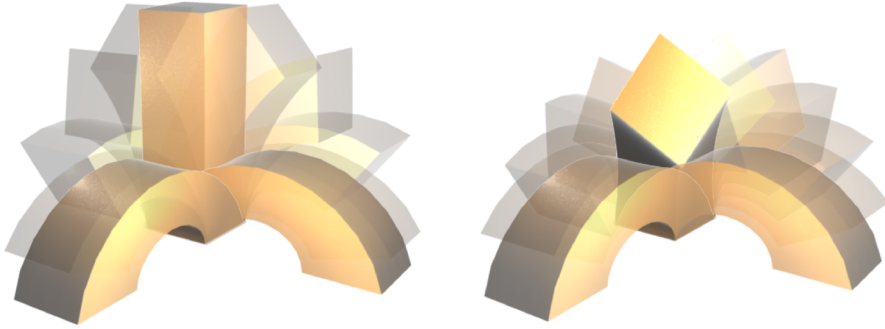


Figure 4.3: The bending block sequence (exhibiting a C^1 -discontinuity) before smoothing (left) and after 50 smoothing iterations (right).

4

get into a fully upright position anymore, and the direction of the motion evenly changes across the full animation.

The usefulness to this type of animation smoothing becomes clear when used on a more complex animation: the centaur animation (cf. Figure 4.4) was created by concatenating interpolation curves to 6 successive poses of the centaur shape. This results in an animation with visible jumps in the moving direction at the input poses. After some steps of the discrete flow, to the animation is visibly smoother and without discontinuities. This can be made precise in the following sense: we can look at the part of the gradient of the energy functional (4.4) of the animation (as a curve in shape space) that corresponds to a certain frame of the animation, in particular at the norm of the l.h.s. of (4.3). In case of the centaur sequence, we can make the observation that the norms of these individual parts of the gradient are much larger at the discontinuities than at the other shapes. This leads to large spikes in the norm of the gradient parts corresponding to these frames. In Figure 4.5, we plot the norm of these individual parts of the gradients after various iterations of smoothing. One can observe how the spikes become less sharp while smoothing the animation.

Computing geodesics As explained in Section 4.5, we can use our technique to compute geodesics. The advantage of only having to solve optimization problems involving one unknown shape at a time and the application of the model reduction techniques lead to a significant speed-up when comparing to the times stated in [2]: when computing the discrete geodesic of the finger mesh on 8 shapes, Heeren et al. report a computation time of 628s for a multilevel mesh. For the same setting, our technique needed 6,42s, including the precomputation time (subspace construction and computation of initial Hessians). To justify our reduced-order modeling, we computed the same geodesic using the full-order model in the space \mathbb{R}^n and compared the lengths of both curves, which differed by less than 0.1% of the length of the full geodesic, as well as the L_2 -distances of all shapes, which differed by less than 1^{-3} % of the summed lengths of the shape vectors.

The dimensional reduction of the spatial domain and the fast computation times al-



Figure 4.4: Applying our flow to concatenated animations results in a visually smooth animation without visible “corners” in the motion.

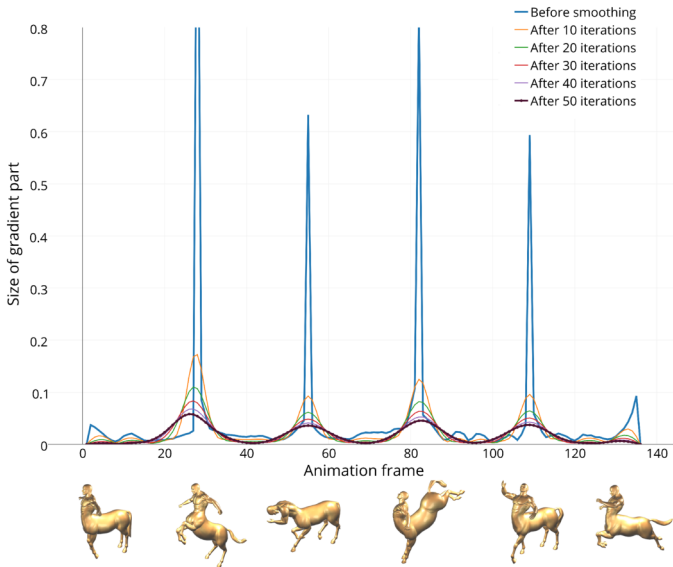


Figure 4.5: The norm of the energy gradients of the individual shapes (left-hand side of (4.3)) for the centaur sequence after various smoothing iterations.

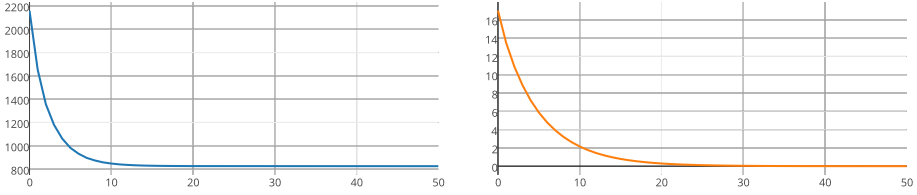


Figure 4.6: Plot of the squared curve length (blue) and size of the gradient of functional (4.4) (orange) while smoothing a sequence of 12 elephant shapes (initialized as the interpolation curve). The x -axis denotes the number of smoothing iterations in both plots.

4

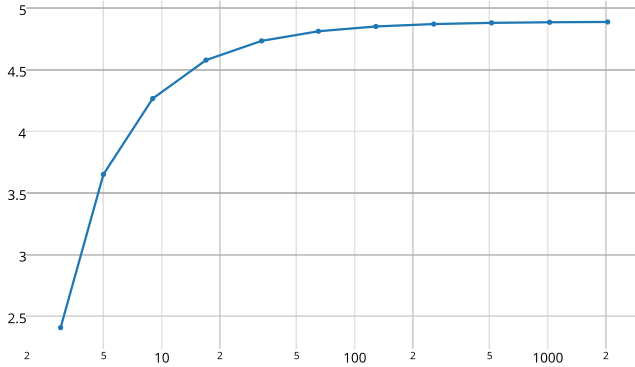


Figure 4.7: Plot of the length of the bending block geodesic (y -axis) with a varying number of intermediate shapes (x -axis).

low for computing discrete geodesics with finer time discretizations: the finger geodesic on 64 shapes leads to a computation time of 19.55s, the elephant geodesic (cf. Figure 4.1) on 256 shapes (40k vertices per shape) leads to a computation time of 376s.

To demonstrate the convergence of our flow to a geodesic, we plot the squared length of the curve and the length of the gradient of functional (4.4) in Figure 4.6 for the computation of the elephant geodesic on 100 shapes, where we initialize the geodesic with 100 interpolation shapes right away (instead of using the adaptive scheme). The plot shows that the length of the curve decreases monotonically.

While the loss of precision due to model reduction is very low, we actually gain accuracy by being able to compute discrete geodesics with more shapes. In Figure 4.7, we plot the length of the geodesic between the bent over and upright block on a varying number of shapes. The plot demonstrates that the length differs significantly when comparing a geodesic with 10 to a geodesic with 100 shapes. This shows the benefit of computing geodesics on many shapes.

De Casteljau algorithm in shape space The fast computation of geodesics makes it possible to evaluate geometric constructions that require repeated computation of geodesics. As one example, we construct nonlinear “Bézier curves” in shape space by executing the de Casteljau algorithm with respect to the shape space metric. This means the straight lines used in Euclidean space are replaced by geodesics in shape space. In [3], Effland et al. applied the de Casteljau algorithm in a shape space of images to obtain smooth curves from a few input images, we refer to this paper for further details on the algorithm.

To compute a point of the “Bézier curves” controlled by four shapes, we need to compute the three geodesics between each successive pair of boundary shapes once and three additional geodesics per shape of the final curve. Figure 4.8 show an example of such a “Bézier curves” with four control poses of a block shape. The computed curve consists of 32 shapes and each auxiliary geodesic was also computed on 32 shapes. The computation time for the whole procedure (which requires the computation of almost 100 geodesics) was 241s. This could be drastically reduced by using coarser time discretizations for the auxiliary geodesics.

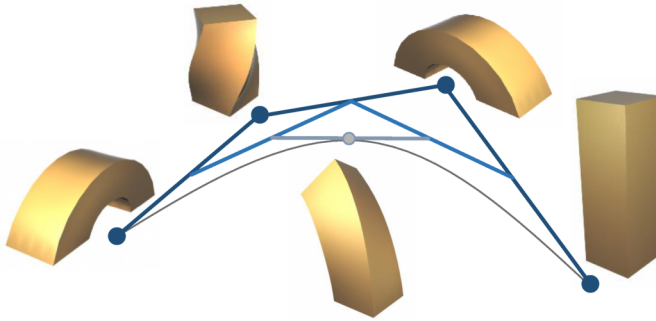


Figure 4.8: Visualization of the de Casteljau algorithm in shape space: the straight lines are geodesics between shapes.

Denosing and animation repair Our smoothing technique is also able to enhance and repair noisy data: an animation of a hand bending its fingers with temporal noise becomes completely denoised after 100 smoothing steps. The shape of the fingers remains plausible even after applying a large amount of smoothing. However, the motion itself has also changed: before smoothing, the index finger and thumb touched, whereas after the smoothing, the fingers stay far away from each other throughout the whole animation.

If the flow is used for noise removal, curve shortening can lead to over-smoothing. To reduce this effect, we use a restoring force that pushes the curve towards its initial state. To achieve this, we modify the flow equation (4.2):

$$y_i^0 = x_i$$

$$y_i^{k+1} = A\left((1-\rho)\frac{\tau}{2}, y_{i-1}^k, (1-\rho)(1-\tau), y_i^k, (1-\rho)\frac{\tau}{2}, y_{i+1}^k, \rho, x_i\right),$$

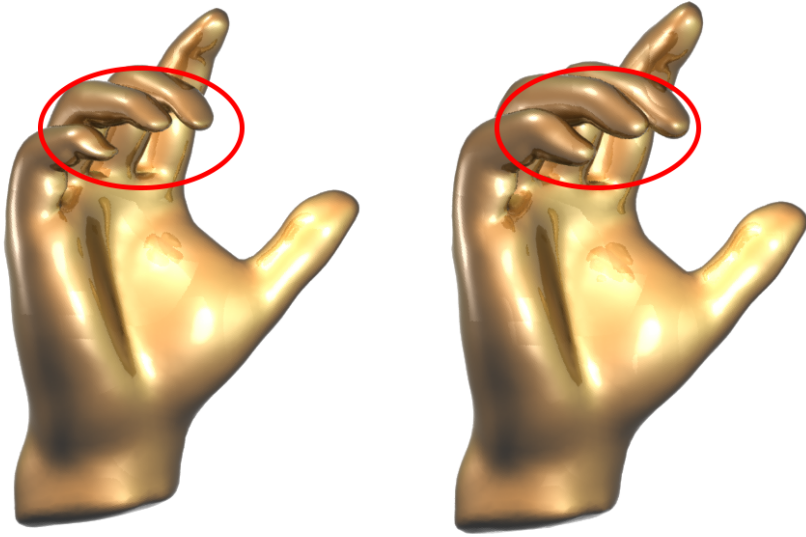


Figure 4.9: Applying our flow to animations exhibiting artifacts can remove them: An animation that contains frames with strong artifacts (left, visible on the fingers). After smoothing the animation, the shape looks artifact-free again, while the motion is kept intact due to the restoring force.

that is, in addition to the current shape and its neighbors, we take the original shape into the local averaging processes. The parameter $\rho \in [0, 1]$ controls the strength of the restoring force. The restoring force can be used for all shapes of the curve or just for some selected shapes.

With this force, the animation is denoised, while articulation of the motion remains intact. We perform a similar experiment with an animation obtained by linear blending of a coarse set of keyframes (cf. Figure 4.9). By adding a restoring force to the keyframes, we are able to remove the linear-blending artifacts, while keeping the overall motion intact.

In addition, we tested our method on motion capturing data from Gall et al. [42], which exhibits temporal and spatial noise. Again, using our smoothing technique with restoring force, we are able to acquire a smooth animation, which at the same time keeps its main characteristics. An advantage of using our smoothing flow to denoise motion capturing data is that each smoothing step regularizes the motion of the whole mesh instead of individual vertex positions. This implies that unwanted deformations and artifacts due to strong noise can be filtered without smoothing the mesh itself.

Averaging operators As discussed in Section 4.6, we used the formulation (4.10) of the shape averaging in most of our experiments to enhance computation speed. We observe that the choice of the operator used for shape averaging, (4.1) or (4.10), does not lead to significantly different geodesics. Table 4.2 shows the results of an experiment performed in this regard: We computed three different geodesics using both formulations (4.1) and

	Bending Block	Centaur	Elephant
Length of (4.10)-geodesic	0.3680	142.1023	368.5427
Length of (4.1)-geodesic	0.3648	141.7774	348.3585
Length of interpolation curve	37.4360	155.9907	613.1286
Rel. L_2 -error of geodesics	5.2375^{-5}	1.4173^{-5}	4.2512^{-5}

Table 4.2: Geodesics computed using the symmetric (4.1) and the asymmetric averaging operator (4.10).

(4.10). We state the energy (4.4) of these geodesics (with the energy of the interpolation curve as a comparison) as well as their relative L_2 -error. This error was computed by first registering each pair of corresponding shapes via a best rigid fit, and then taking the ratio of the norm of the difference of both geodesics (interpreted as vectors in $\mathbb{R}^{n \cdot m}$) and the norm of the (4.1)-geodesic.

COMPARISONS

For comparisons to the timings and accuracy of computing geodesics using our technique as opposed to [2] see the paragraph above.

To prevent confusion, we first want to stress that our flow formulation is based on shape averaging, which is closely related to shape interpolation. This being the case, every interpolation technique can be used to define a smoothing flow using our formulation. However, not every interpolation scheme exhibits the properties of elastic shape averaging required to prove existence of a metric and convergence of our flow to geodesics in shape space.

Nonlinear shape interpolation While our model reduction and flow formulation allow for a fast computation of geodesics, interpolation curves can be produced even faster. The reason is that each interpolating shapes can be computed directly (without the need to compute all interpolating shapes). This cannot be done for geodesics, as we always have to compute the whole curve. Hence, the optimization problem for geodesics is more involved. On the other hand, the more complex problem couples the shapes of a geodesic to each other, which is not the case for shape interpolation. For example, if the computations of interpolating shapes end in different local minima, we can get a stuttering interpolation curve. Due to the coupling of the shapes such effects are smoothed out during the computation of geodesics.

Poisson-type shape interpolation Compared to Poisson interpolation curves (cf. [14]), our technique does not suffer from problems that arise in Poisson interpolation when

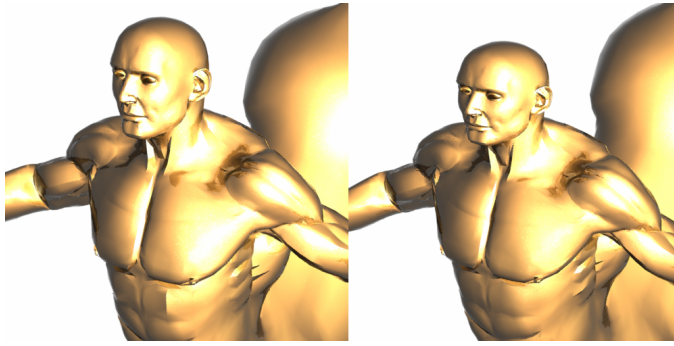


Figure 4.10: A frame of the smoothed centaur sequence. Left: Our technique. Right: bilateral filtering of the vertex trajectories [23], where the head appears to be shrunken.

4

elements are rotated by more than 180 degrees. Also, linear interpolation of the vertex positions leads to strong artifacts, even for very small deformations.

Comparison to other smoothing and denoising techniques Temporal filtering of the vertex positions, as proposed in [23], also leads to a smoother motion, but since a lot of filtering is required, the shape undergoes unnatural deformations, similar to linearization artifacts, as can be seen in Figure 4.10.

4.8. CONCLUSION

In this chapter, we are proposing techniques for the processing of curves in shape space. In particular, we introduce a discrete geometric flow for curves in shape space. The flow iteratively computes local weighted average shapes and thereby decrease the magnitude of the deformation between consecutive shapes of the curve. Based on the flow, we design a novel type of smoothing filter for motions and animations of shapes. In contrast to previous work, the filter only smoothes the deformations between the shapes and thereby minimizes the distortion of the shapes themselves. One application of this filter is the smoothing of motions and animations of objects. We use the filter for reducing jittering artifacts in motion captured data and for smoothing transitions that appear when different motions are concatenated.

Our analysis shows that the flow converges to geodesics in shape space. To compute the flow, we propose a reduced-order scheme that combines a dimensional reduction with a scheme for reduced energy and force approximation. The approach significantly accelerates the computation of geodesics in shape space. In addition, it allows for finer temporal discretizations, which improves the approximation quality. We think that these two benefits are important for an effective processing of curves in shape space. We demonstrate results obtained with our scheme for the computation of geodesics that blend between two shapes as well as the computation of nonlinear “Bézier curves” in shape space that are controlled by a coarse polygon in shape space.

Future work We introduce techniques for the processing of curves in shape space. We think that this is a promising direction for processing motion and animation and we expect to see more algorithms that transfer techniques from the processing of curves in \mathbb{R}^n to curves in shape space. For example, analogous to the example of De Casteljau's algorithm, curve subdivision schemes like corner cutting could be transferred to shape space. Another example is the fairing of curves in shape space. The proposed smoothing filter is the first of this kind and we expect that more filtering techniques for curves in Euclidean space will be transferred to filters for motion and animation of deformable shapes.

Furthermore, we think that reduced-order modeling has a great potential for geometry processing in shape space and other applications using Riemannian metrics on shape spaces. Fast approximation algorithms for shape space computations can be designed and larger data sets can be processed.

With a growing market for devices which are able to directly capture deforming geometry, processing of motion and animation becomes more and more important. The concept of curves in shape space provides powerful and theoretical sound tools for processing this data (in particular for template-based approaches).

REFERENCES

- [1] M. Kilian, N. J. Mitra, and H. Pottmann, *Geometric modeling in shape space*, ACM Trans. Graph. **26**, 64:1 (2007).
- [2] B. Heeren, M. Rumpf, M. Wardetzky, and B. Wirth, *Time-discrete geodesics in the space of shells*, Comp. Graph. Forum **31**, 1755 (2012).
- [3] A. Effland, M. Rumpf, S. Simon, K. Stahn, and B. Wirth, *Bézier curves in the space of images*, in *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 9087, edited by J.-F. Aujol, M. Nikolova, and N. Papadakis (Springer International Publishing, 2015) pp. 372–384.
- [4] L. Younes, *Shapes and Diffeomorphisms* (Springer, 2010).
- [5] B. Heeren, M. Rumpf, P. Schröder, M. Wardetzky, and B. Wirth, *Exploring the geometry of the space of shells*, Comp. Graph. Forum **33**, 247 (2014).
- [6] B. Wirth, L. Bar, M. Rumpf, and G. Sapiro, *A continuum mechanical approach to geodesics in shape space*, Int. J. Comput. Vision **93**, 293 (2011).
- [7] S. Kurtek, E. Klassen, J. Gore, Z. Ding, and A. Srivastava, *Elastic geodesic paths in shape space of parameterized surfaces*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**, 1717 (2012).
- [8] S. Kurtek, A. Srivastava, E. Klassen, and H. Laga, *Landmark-guided elastic shape analysis of spherically-parameterized surfaces*, Computer Graphics Forum **32**, 429 (2013).

- [9] B. Berkels, P. T. Fletcher, B. Heeren, M. Rumpf, and B. Wirth, *Discrete geodesic regression in shape space*, in *Energy Minimization Methods in Computer Vision and Pattern Recognition* (Springer, 2013) pp. 108–122.
- [10] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, *2-d shape blending: An intrinsic solution to the vertex path problem*, in *SIGGRAPH* (1993) pp. 15–18.
- [11] M. Alexa, D. Cohen-Or, and D. Levin, *As-rigid-as-possible shape interpolation*, in *SIGGRAPH* (2000) pp. 157–164.
- [12] R. Chen, O. Weber, D. Keren, and M. Ben-Chen, *Planar shape interpolation with bounded distortion*, *ACM Trans. Graph.* **32**, 108:1 (2013).
- [13] R. W. Sumner and J. Popović, *Deformation transfer for triangle meshes*, *ACM Trans. Graph.* **23**, 399 (2004).
- [14] D. Xu, H. Zhang, Q. Wang, and H. Bao, *Poisson shape interpolation*, in *Symp. on Solid and Phys. Mod.* (2005) pp. 267–274.
- [15] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, *Linear rotation-invariant coordinates for meshes*, *ACM Trans. Graph.* **24**, 479 (2005).
- [16] A. Sheffer and V. Kraevoy, *Pyramid coordinates for morphing and deformation*, in *Proceedings of Symposium on 3D Data Processing, Visualization, and Transmission* (2004) pp. 68–75.
- [17] T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann, *Multi-scale geometry interpolation*, *Comp. Graph. Forum* **29**, 309 (2010).
- [18] S. Fröhlich and M. Botsch, *Example-driven deformations based on discrete shells*, *Comp. Graph. Forum* **30**, 2246 (2011).
- [19] Z. Levi and C. Gotsman, *Smooth rotation enhanced as-rigid-as-possible mesh animation*, *Visualization and Computer Graphics*, *IEEE Transactions on* **21**, 264 (2015).
- [20] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, *Real-time nonlinear shape interpolation*, *ACM Trans. Graph.* **34**, 34:1 (2015).
- [21] M. Botsch and O. Sorkine, *On linear variational surface deformation methods*, *IEEE Transactions on Visualization and Computer Graphics* **14**, 213 (2008).
- [22] M. Rumpf and B. Wirth, *Variational methods in shape analysis*, in *Handbook of Mathematical Methods in Imaging* (Springer, 2011) pp. 1363–1401.
- [23] D. Vlastic, I. Baran, W. Matusik, and J. Popović, *Articulated mesh animation from multi-view silhouettes*, in *ACM Trans. Graph.*, Vol. 27 (ACM, 2008) p. 97.
- [24] H. Li, L. Luo, D. Vlastic, P. Peers, J. Popović, M. Pauly, and S. Rusinkiewicz, *Temporally coherent completion of dynamic shapes*, *ACM Trans. Graph.* **31**, 2 (2012).
- [25] A. Witkin and M. Kass, *Spacetime constraints*, *ACM SIGGRAPH* **22**, 159 (1988).

- [26] J. Barbič, M. da Silva, and J. Popović, *Deformable object animation using reduced optimal control*, ACM Transactions on Graphics **28**, 53:1 (2009).
- [27] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Interactive spacetime control of deformable objects*, ACM Trans. Graph. **31**, 71:1 (2012).
- [28] J. Barbič, F. Sin, and E. Grinspun, *Interactive editing of deformable simulations*, ACM Trans. Graph. **31**, 70:1 (2012).
- [29] S. Li, J. Huang, F. de Goes, X. Jin, H. Bao, and M. Desbrun, *Space-time editing of elastic motion through material optimization and reduction*, ACM Trans. Graph. **33**, 108:1 (2014).
- [30] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, *Animating deformable objects using sparse spacetime constraints*, ACM Transactions on Graphics (TOG) **33**, 109:1 (2014).
- [31] J. E. Marsden and T. J. R. Hughes, *Mathematical Foundations of Elasticity* (Dover Publications, 1994).
- [32] M. Rumpf and B. Wirth, *A nonlinear elastic shape averaging approach*, SIAM J. on Imaging Science **2**, 800 (2009).
- [33] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder, *Discrete Shells*, Symposium on Computer Animation, 62 (2003).
- [34] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt, *PriMo: Coupled prisms for intuitive surface modeling*, in *Symposium on Geometry Processing* (2006) pp. 11–20.
- [35] O. Sorkine and M. Alexa, *As-rigid-as-possible surface modeling*, in *Symposium on Geometry Processing* (2007) pp. 109–116.
- [36] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, *A simple geometric model for elastic deformations*, ACM Trans. Graph. **29**, 38:1 (2010).
- [37] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Interactive surface modeling using modal analysis*, ACM Trans. Graph. **30**, 119:1 (2011).
- [38] S. S. An, T. Kim, and D. L. James, *Optimizing cubature for efficient integration of subspace deformations*, ACM Trans. Graph. **27**, 1 (2008).
- [39] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, *An efficient construction of reduced deformable objects*, ACM Trans. Graph. **32**, 213:1 (2013).
- [40] J. Barbič and D. L. James, *Real-time subspace integration for St. Venant-Kirchhoff deformable models*, ACM Trans. Graph. **24**, 982 (2005).
- [41] J. Nocedal and S. J. Wright, *Numerical Optimization (2nd edition)* (Springer, 2006).
- [42] J. Gall, C. Stoll, E. D. Aguiar, C. Theobalt, B. Rosenhahn, and H. peter Seidel, *Motion capture using joint skeleton tracking and surface estimation*, in *IEEE CVPR* (2009).

5

COMPRESSED VIBRATION MODES OF ELASTIC BODIES

This chapter is based on the publication *Compressed Vibration Modes for Elastic Bodies* by Christopher Brandt and Klaus Hildebrandt, published in Computer Aided Geometric Design in 2017.

5.1. OVERVIEW

Vibration modes and their frequencies are fundamental for analyzing and simulating the dynamics of physical objects. For example, the modes can be used as a basis for model reduction approaches. In this chapter, we introduce an approach for the compression and localization of vibration modes of elastic bodies. The *compressed vibration modes* have a localized support while preserving properties of the natural vibration modes, e.g. they form an orthonormal system in the space of configurations of an elastic body and the low-frequency modes correspond to low-energy deformations. The degree of localization can be controlled by a continuous parameter μ .

For applications, such as reduced-order simulations, the localization provides benefits. As such, the introduced basis can be seen as a building block for model order reduction approaches for the simulation of hyperelastic materials.

The vectors describing the compressed vibration modes of a discrete object are sparse, which results in less memory requirements for storing a basis and fewer arithmetic operations for adding or scaling the vectors. In our experiments, we used the compressed vibration modes for reduced-order simulation and deformation-based shape modeling. A second aspect is that the localization of the vibration modes adds a novel aspect to the modal analysis of deformable objects. Our experiments illustrate that the compressed modes localize in a structured way. We see potential in exploring this structure for modal and shape analysis and using it for applications. As a first step in this direction, we use the compressed modes for shape segmentation into functional parts.

To define the compressed vibration modes, we first characterize the natural vibra-

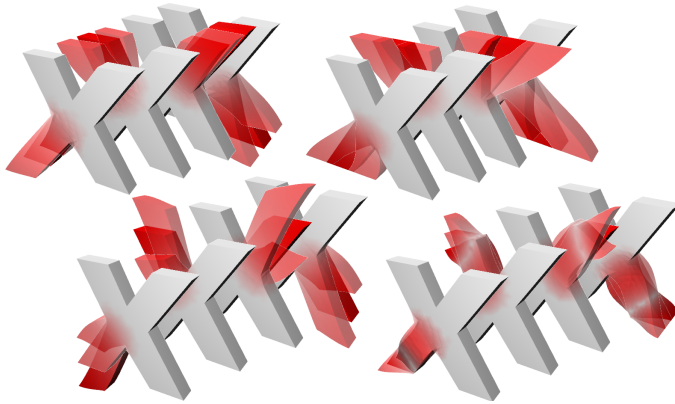


Figure 5.1: The first sixteen compressed vibration modes of an X-shaped mesh ($\mu = 10$). We grouped the same modes (up to symmetry) for each of the four 'legs'.

5

tion modes as the minimizers of an optimization problem, then we add a L^1 regularizing term to the objective to enforce compression. The idea of using L^1 regularization to localize modes of variational problems was introduced in [1] and applied to Schrödinger's equation. The compressed vibration modes, we introduce, specialize this idea to the localization of modes of vibration of elastic bodies. There are significant differences in how we define and compute the compressed modes compared to [1] and other work on the compression of modes like [2–4]. Whereas they compute all compressed modes in one optimization, we devise a reformulation of the L^1 regularized eigenproblem, which allows for computing the modes sequentially. This results in computation timings that scale linearly in the number of modes, whereas previous methods scale superlinearly. Moreover, to compute the modes, we devise an algorithm for solving the L^1 regularized optimization problem, which involves a convexification of a constraint as well as a linearization of the L^1 term. The algorithm allows to stably compute a large number of modes, as the runtime for each additional mode remains (almost) constant. The shown experiments demonstrate the benefits of this computational method over the ADMM method for the computation of compressed modes. An additional point is that our reformulation of the compressed eigenproblem allows for solving the L^1 constrained version of the problem. This formulation has the benefit over the L^1 regularized problem that the L^1 norm remains constant for all modes. This aspect has not been treated in prior work and seems difficult to achieve with prior problem formulations.

5.2. RELATED WORK

Vibration modes Vibration modes are fundamental for describing, analyzing and simulating the dynamics of objects. The low-frequency vibration modes correspond to low-energy deformations of an object, which makes spaces spanned by low-frequency vibration modes attractive for reducing the computational cost of simulations and optimizations. In recent work in graphics, vibration modes are used for reduced-order models

for deformation-based editing [5], deformable object simulation [6–9], sound synthesis [10, 11], and spacetime control of animations [12–15]. Besides the dimensional reduction of simulations and optimizations, vibration modes are used for the analysis of objects [16, 17] and the segmentation of objects into functional parts [18]. One drawback of using vibration modes for dimensional reduction is that the resulting basis vectors are dense, which can be problematic when a large basis is used or applications, like games, impose strict limits on the memory available for the reduced simulation. This problem has been addressed in [19] by applying a data compression scheme for storing the basis vectors. We present, for the first time, a localization of vibration modes, which allows for creating bases of low-energy deformations that are sparse. One resulting benefit for reduced-order methods is a reduced memory requirement.

L^1 regularization for eigenvalue problems *Compressed modes for variation problems* were introduced by Ozoliņš et al. [1] and used for computing localized bases for Schrödinger’s equation. For the numerical computation of the compressed modes, they used a *splitting orthogonality constraint (SOC)* scheme. The approach was extended to the computation of compactly supported multiresolution bases for the Laplace operator on planar domains by Ozoliņš et al. [20] and the sparse approximation of differential operators in the Fourier domain by Mackey et al. [21]. Compressed eigenfunctions of the Laplace–Beltrami operator of curved surfaces, called *compressed manifold modes*, were considered by Neumann et al. [2] and used for mesh segmentation and functional correspondences. For computing the compressed manifold modes, they proposed a scheme based on the alternating direction method of multipliers (ADMM, [22]) and demonstrated that it outperforms the SOC scheme. Boscaini et al. [3] used the compressed manifold modes for building class-specific descriptors for non-rigid shapes. Houston [23] proposed a natural ordering for the compressed manifold modes along with an adaptation of the algorithm that is reported to significantly reduce the number of ADMM iterations required in the optimization. Since the definition of compressed modes includes a unit L^2 norm constraint, the feasible set for the optimization is not a linear space, but a curved manifold. A generic algorithm for L^1 regularized minimization problems over manifolds called the manifold alternating direction method of multipliers (MADMM) was introduced by Kohn et al. [4] and used for the computation of compressed manifold modes. Concurrent to our work, Bronstein et al. [24] proposed a discretization of the L^1 norm for linear Lagrange finite elements on meshes and an iteratively reweighted least-squares scheme for computing compressed manifold modes. In this chapter, we opt for a different approach for computing the compressed vibration modes using convexification of the constraints and linearization of the L^1 term by duplication of the variables.

ℓ_0 regularized eigenvalue problems Sparsity in finite dimensional eigenvalue problems can be enforced by regularization with a ℓ_0 “norm”, which counts the number of non-zero entries in a vector. A recent scheme for solving the resulting *sparse eigenvalue problems* is the *truncated power method* introduced by Yuan et al. [25]. In order to compute sparse eigenvectors of a symmetric matrix with less than k non-zero entries, the matrix is iteratively applied to some initial vector, and subsequently all but the k largest

absolute values are set to 0. Unfortunately such a simple approach is not suited in our scenario since we are interested in finding approximations of continuous functions with localized support, and the ℓ_0 “norm” of a vector representing such a function in some discretization scheme does not measure the size of the support since the mapping between the function and the vector depends on the underlying mesh. If weights, which consider the area associated to each vertex, are used to account for the underlying mesh, the complexity of the problem rises and methods like the truncated power method are no longer applicable.

L^1 regularization for empirical eigenvalue problems Related to the construction of subspaces using compressed modes is the problem of constructing localized bases that span a given space of deformations. Meyer et al. [26] considered spaces specified by an animator’s rig and compute localized, not necessarily orthogonal bases of the rig space via Varimax rotations. It is demonstrated that the resulting basis vectors localize in a structured way. Neumann et al. [27] introduced a scheme for computing localized bases in a spaces of captured facial deformations using a L^1 regularized principal component analysis. In contrast to the approach proposed here, a large sequence of input shapes is required and no elastic energies are taken into account. As a result, this method reduces and localizes a set of input deformations, as opposed to producing a new set of previously unknown deformations based on an analysis of the elastic energy of an object.

Sparsity in geometry processing Sparsity enforcing regularization has also been used in the context of shape deformation. Gao et al. [28] introduced localization encouraging deformation energies. Specifically, they modify the As-Rigid-As-Possible energy by replacing terms measuring least-squares deviations from the rest configuration by terms involving p -norms, which, for small p , tends to localize the deformations when using the modified energy in modeling tasks (*i.e.* minimizing it under soft or hard constraints). More examples, where ℓ_1 or ℓ_0 regularization has been employed for geometry processing tasks, surface smoothing [29] and optimal spline approximation [30, 31]. For a recent survey on compressed sensing for geometry processing, we refer to [32].

5.3. BACKGROUND: DEFORMATION ENERGIES AND VIBRATION MODES

In this section, we introduce basic concepts concerning deformation energies and vibration modes. Due to space restrictions, we consider only the discrete setting and the concepts needed to define the compressed vibration modes in the next section. For more background on elasticity and finite element discretization, we refer to the textbook [33].

In the following, we will deal with triangle surface meshes and tetrahedral volume meshes with fixed connectivity $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{T})$ (vertices, edges, faces and, in case of tetrahedral meshes, tetrahedrons) and initial vertex positions given by a vector $\mathbf{x} \in \mathbb{R}^{3|\mathcal{V}|}$. We refer to the mesh with these vertex positions as the *rest configuration*. We express a deformation of the mesh via a vector $\mathbf{u} \in \mathbb{R}^{3|\mathcal{V}|}$, which stacks the displacements of each vertex in world coordinates.

A discrete deformation energy $E_{\mathcal{M}, \mathbf{x}} : \mathbb{R}^{3|\mathcal{V}|} \rightarrow \mathbb{R}$ is a measure for the effort it takes to deform an object from its rest configuration into a deformed state given by some displacement vector \mathbf{u} , *i.e.* the new vertex positions are given by $\mathbf{x} + \mathbf{u}$, while assuming that the object consists of a hyperelastic material. The energy E depends on the connectivity of the mesh, its rest configuration and material properties. For our experiments, we used a finite element discretization of *St. Venant–Kirchhoff materials* [33] for simulating elastic solids and *Discrete Shells* [34] for elastic shells.

In order to define a L^2 scalar product and L^1 and L^2 norms on the space of displacements of a mesh, we associate a mass m_i to every vertex. We assume that the material of the elastic bodies has a constant density ρ , and, for thin shells, we assume a constant thickness ϵ . Then, for elastic solids, the mass m_i equals ρ times a fourth of the combined volume of all tets containing vertex v_i , and, for thin shells m_i equals $\rho \epsilon$ times a third of the combined area of the triangles containing v_i . The *mass matrix* \mathbf{M} is the $3|\mathcal{V}| \times 3|\mathcal{V}|$ diagonal matrix that stacks the masses on the diagonal (in the same order as the displacements are stacked). The L^2 scalar product is

$$\langle \mathbf{u}, \mathbf{v} \rangle := \mathbf{u}^T \mathbf{M} \mathbf{v}, \quad (5.1)$$

with corresponding L^2 norm $\|\mathbf{u}\|_2 = \langle \mathbf{u}, \mathbf{u} \rangle^{\frac{1}{2}}$, and the L^1 norm is

$$\|\mathbf{u}\|_1 := \sum_{i=0}^{3|\mathcal{V}|} M_{ii} |u_i|, \quad (5.2)$$

where M_{ii} is the i^{th} entry on the diagonal of \mathbf{M} and u_i the i^{th} entry of \mathbf{u} . In [2] the ℓ_1 -norm was used for computing compressed manifold modes, *i.e.* the masses associated to the vertices were not considered when computing the sparsity regularizer. We illustrate the effect of the different 1-norms on the compressed vibration modes in Figure 5.2, where we demonstrate that our method is robust against remeshing of the object.

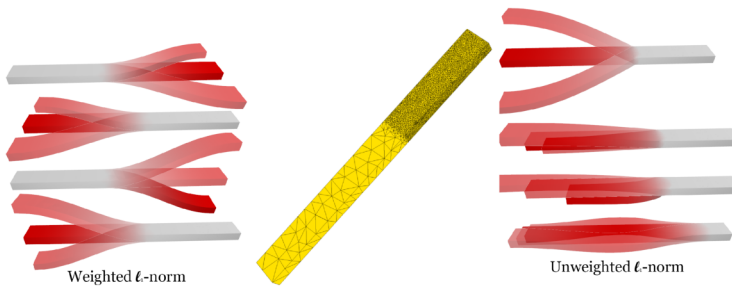


Figure 5.2: Comparison of the first four modes when using the mass-weighted discrete L^1 norm (left) when optimizing for compressed vibration modes of an irregular bar mesh (middle) and the (unweighted) L^1 norm (right) (which was used in [2]).

Natural vibration modes We consider the modes of vibration of an elastic object at its rest configuration. The Hessian \mathbf{H} of \mathbf{E} at \mathbf{x} is the matrix of second derivatives of \mathbf{E} with

respect to displacements of the object, *i.e.* $\mathbf{H}_{ij} = \frac{\partial^2}{\partial u_i \partial u_j} E_{\mathcal{M},\mathbf{x}}(0)$. At the rest configuration, the Hessian is a symmetric and positive semi-definite matrix. The vibration modes are the eigenvectors and the frequencies are the square roots of the eigenvalues of the generalized eigenvalue problem

$$\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{M}\mathbf{u}_i. \quad (5.3)$$

The problem can be re-written as the following sequence of optimization problems:

$$\mathbf{u}_i := \begin{cases} \arg \min_{\mathbf{u}} \mathbf{u}^T \mathbf{H} \mathbf{u} \\ \text{subject to } \mathbf{u}^T \mathbf{M} \mathbf{u} = 1 \text{ and } \forall j < i : \mathbf{u}^T \mathbf{M} \mathbf{u}_j = 0 \end{cases}$$

We will use this definition as the starting point for defining the compressed vibration modes.

5.4. COMPRESSED VIBRATION MODES

Vibration modes are in general global and thus a displacement in direction of any mode deforms the whole object. Since reduced order methods need to store the whole subspace basis, this is a drawback for applications imposing strict bounds on the memory requirements, *e.g.* when the GPU memory is shared for different computational tasks (Games or VR) or larger bases or bases for multiple objects are needed (real-time simulation). Moreover, we found that localized deformations are interesting and natural quantities of the mesh: they localize in a structured way and thus give a useful tool for analyzing the mesh and exploring the space of deformations. Thus, we aim at finding a sparse, *i.e.* localized basis for deformations of a mesh, which represent a trade-off between optimality w.r.t. the deformation energy $E_{\mathcal{M},\mathbf{x}}$ and sparsity, *i.e.* few non-zero entries per vector.

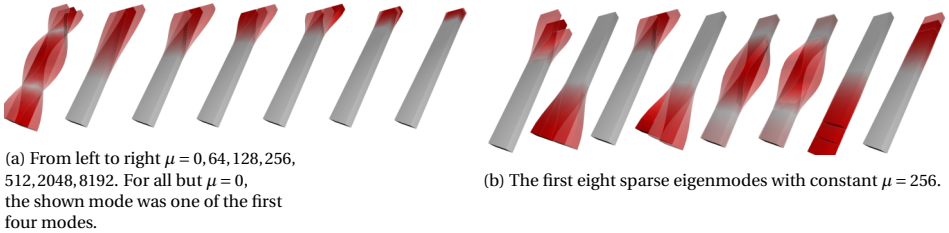


Figure 5.3: Compressed vibration modes of a tetrahedral bar mesh.

To this end, we define the compressed modes as solutions \mathbf{u}_i of the following sequence of optimization problems:

$$\mathbf{u}_i := \begin{cases} \arg \min_{\mathbf{u}} \mathbf{u}^T \mathbf{H} \mathbf{u} + \mu \|\mathbf{u}\|_1 \\ \text{subject to } \mathbf{u}^T \mathbf{M} \mathbf{u} = 1 \text{ and } \forall j < i : \mathbf{u}^T \mathbf{M} \mathbf{u}_j = 0 \end{cases} \quad (5.4)$$

Note that this is a direct extension of the iterative definition for natural vibration modes as given above, with the addition of a sparsity regularizer: the term $\mu \|\mathbf{u}\|_1$ represents a

trade-off between minimizing the elastic energy introduced by the mode, and its sparsity. Higher μ results in more localized deformations, see Figure 5.3a. By iteratively adding orthogonality constraints, the resulting modes cover a broad range of deformations, see Figure 5.3b. The compressed vibration modes strongly depend on the structure of the underlying object and thus analyzing their support reveals some of the structure of the object in return (this will be demonstrated in Section 5.7, where we use compressed vibration modes for object segmentation). It is not obvious how to choose the parameter μ such that a certain localization is achieved. We show several examples for sets of sparse vibration modes and discuss the effects of different sparsity parameters μ in Section 5.8. Additionally, we introduce an alternative definition for compressed vibration modes, where the L^1 norm is being constrained instead of regularized, below.

The definition (5.4) for compressed vibration modes has several differences to the previous definitions of compressed modes as given in [1–4] (aside from us looking at localized vibrations of elastic bodies for the first time). They look for a full set of vectors in a single optimization problem, constraining the whole set to be orthonormal, whereas we define them as solutions of a sequence of smaller optimization problems, while adding an orthogonality constraint in each iteration. This has the advantage that each of the minimization problems has a comparatively low number of variables. When trying to compute a large number of modes for high resolution meshes, solving for all modes at once becomes prohibitive. Note that for $\mu = 0$ both formulations yield the solutions to the generalized eigenvalue problem. Another advantage is that the choice of the sparsity parameter μ becomes consistent: the sparsity of the modes does not depend on the number of modes K but only on μ , whereas in [2], the same choice of μ would lead to different sparsity for bases of different sizes. Another difference in our definition is that we use a proper discretization of the L^1 norm of piecewise linear displacements, as opposed to the unweighted vector ℓ_1 norm used in [2].

5.5. COMPUTATION OF COMPRESSED VIBRATION MODES

The optimization problem (5.4) is non-convex due to the constraint on the L^2 norm of \mathbf{u} and the L^1 term is non-differentiable. For solving this type of problem, a Splitting Orthogonality Constraint and an Alternating Direction Method of Multipliers scheme have been proposed in [1] and [2]. A comparison of the two approaches in [2] indicates that the ADMM scheme is more effective. In our experiments, see Section 5.8, ADMM did not produce satisfying results for the computation of compressed vibration modes, neither in terms of computation times, nor in terms of convergence to an acceptable minimum. Therefore, we opt for a novel optimization scheme, which we refer to as *iterated convexification for compressed modes* (ICCM). It combines a convexification of the problem with the classical approach of turning the L^1 term into a linear term and linear inequality constraints by duplicating the variables. We will describe both steps in detail below. ICCM is parameter free and its implementation is straight-forward, given a quadratic programming library.

Convexification We replace the constraint $\mathbf{u}^T \mathbf{M} \mathbf{u} = 1$, by a hyperplane constraint $\mathbf{u}^T \mathbf{M} \mathbf{c}_0 = 1$, where \mathbf{c}_0 is initialized as a random, normalized vector. Essentially this replaces the constraint that the deformation should be on the unit sphere, with the constraint that the deformation should be on a plane that is tangential to the unit sphere. After we find a solution \mathbf{u} with this constraint, we set $\mathbf{c}_1 = \mathbf{u} / \|\mathbf{u}\|_{L^2}$ and solve the optimization problem again, this time with the hyperplane constraint $\mathbf{u}^T \mathbf{M} \mathbf{c}_1 = 1$. This is being repeated until some convergence criterion is met. In our experiments we used a lower bound on the deltas between the energy values of the last and current solution as the stopping criterion. After finding a mode, we initialize the next optimization problem by again using a random hyperplane constraint \mathbf{c}_0 , which additionally has the property that the problem admits a feasible solution, *i.e.* $\forall j < i : \mathbf{c}_0^T \mathbf{M} \mathbf{u}_j = 0$. While this convexification might lead to finding a local instead of a global minimum, it provides a fast and suitably stable way to solve a large scale, non-convex and non-differentiable optimization problem. The performance and consistence of this convexification is being further analyzed in Section 5.8.

5

Linearization of the L^1 term After convexifying the L^2 constraint, each mode is computed via solving a sequence of convex optimization problems. In order to turn these into quadratic programs, we get rid of the non-differentiable part introduced by the L^1 regularizer by using non-negative variables, that is,

$$\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^- \quad \mathbf{u}^+, \mathbf{u}^- \geq 0, \quad (5.5)$$

where the inequality is meant component wise. This is a widely used approach, first proposed in Tibshirani et al. [35]. With this change of variables the L^1 term can be bounded by a linear term

$$\|\mathbf{u}\|_1 \leq \mathbf{1}^T \mathbf{M} \mathbf{u}^+ + \mathbf{1}^T \mathbf{M} \mathbf{u}^-. \quad (5.6)$$

After expressing the energy functional and all constraints in terms of \mathbf{u}^+ and \mathbf{u}^- , as well as adding the non-negativity constraints (5.5) and employing the convexification above, the optimization problem (5.4) turns into an inner and outer sequence of quadratic programs. In each inner iteration we adapt the convex hyperplane constraints until a convergence criteria is met:

$$\mathbf{u}_{i,k} := \begin{cases} \arg \min_{\mathbf{u}=\mathbf{u}^+-\mathbf{u}^-} \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \begin{pmatrix} \mathbf{H} & -\mathbf{H} \\ -\mathbf{H} & \mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix} \\ \quad + \mu (\mathbf{1}^T \mathbf{M} \mathbf{u}^+ + \mathbf{1}^T \mathbf{M} \mathbf{u}^-) \\ \text{subject to } \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{c}_k \\ -\mathbf{c}_k \end{pmatrix} = 1 \\ \quad \text{and } \forall j < i : \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{u}_j \\ -\mathbf{u}_j \end{pmatrix} = 0 \\ \quad \text{and } \mathbf{u}^+, \mathbf{u}^- \geq 0 \end{cases} \quad (5.7)$$

$$\mathbf{c}_{k+1} = \mathbf{u}_{i,k} / \|\mathbf{u}_{i,k}\|_2.$$

After that we add the last mode from this sequence to the set of computed sparse vibration modes, *i.e.*

$$\mathbf{u}_i := \mathbf{u}_{i,k} / \|\mathbf{u}_{i,k}\|_2$$

and then start the inner iteration above again, with the additional orthogonality constraint induced by the new mode. Note that the linear term that bounds the L^1 norm from above, (5.6), is actually equal to the L^1 norm for solutions of (5.7), since otherwise one could trivially construct a feasible lower energy solution (see the Appendix 5.A where we show this for the L^1 constrained formulation). Thus, (5.6) is a valid, linear, replacement for the non-differentiable term $\|\mathbf{u}\|_1$.

Each inner iteration of ICCM amounts to solving a quadratic program which can be done using highly efficient specialized solvers (in our implementation we used *Mosek*). Typical iteration times and full times for computing a set of modes can be taken from Figure 5.14 and Table 5.4. Table 5.3 shows a comparison to the computation times resulting from ADMM as implemented in [2]. We will analyze the resulting compressed vibration modes and discuss stability and computation times of the algorithm in more detail in Section 5.8.

5.6. L^1 CONSTRAINED VIBRATION MODES

In addition to using L^1 regularization to enforce compression of the modes, we can consider a L^1 constrained formulation: we remove the term $\mu\|\mathbf{u}\|_1$ from the objective functional and instead add the constraint $\|\mathbf{u}\|_1 = s$ for some $s \in \mathbb{R}^{>0}$. The resulting optimization problem is only well-posed, if there are solutions with unit L^2 norm and L^1 norm lower or equal to s . Whether feasible solutions exist for a given value of s can be checked by a numerical optimization software such as *MOSEK*. Also note that the correctness of the linearized L^1 term used in our optimization has to be shown for the constrained version, the proof can be found in Appendix 5.A. For the L^1 constrained problem, the inner iterations of ICCM remain quadratic programs and the algorithm results in comparable computation times.

The attractiveness of the L^1 constrained version becomes apparent when analyzing the behavior of the elastic term $\mathbf{u}^T \mathbf{H} \mathbf{u}$. For many meshes, the vibration modes have energy levels with vastly different magnitudes. Thus, the L^1 regularizer has a different effect on the sparsity of the support of the compressed vibration modes: While the first modes have the desired sparsity pattern, later modes might have a dense support when using the same value of μ . This can be seen in Figure 5.4 where, in the L^1 regularized case ($\mu = 255$), the support of the first eight modes covers roughly one third of the mesh, but higher frequency modes have a support that covers roughly 90% of the mesh. They are essentially dense modes, which shows that the L^1 term does not have significant influence on the optimization. If we fix the value of the sparsity term of the first mode in the L^1 regularized case and use it as the value s in the L^1 constrained case (here $s = 11.5$), we are able to compute a set of modes with consistent sparsity pattern. The plots in Figure 5.3b show the respective values of the L^1 and elasticity terms. From this it can be seen that a regularization is not suited since the elastic terms explode for higher frequencies. This prevents us from having to search for a suited value of μ when computing each mode, which would be infeasible given the computation times of the modes.

We want to remark that while ICCM can be used for computing a constrained version of the compressed modes, it is not clear how to employ previously proposed minimization schemes to solve the constrained version. To the best of our knowledge, this is the first time a L^1 constrained version for compressed modes has been considered. In [36], Rostamov et al. propose an L^1 constrained optimization problem to obtain smooth functions on meshes, *Multiscale Biharmonic Kernels*. Their motivation to use an L^1 constraint is similar to ours: the parameter that controls the magnitude of the L^1 -term controls the localization of the solutions directly. For this problem, no non-convex constraint is present, such that the minimization can be directly formulated as a quadratic program with linear (in)equality constraints.

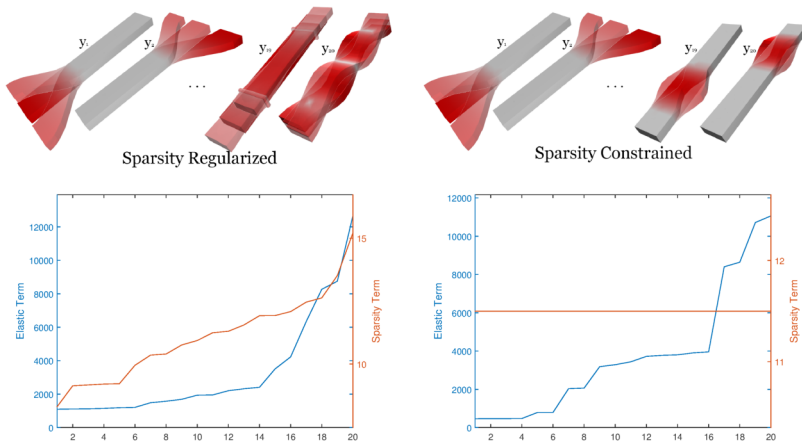
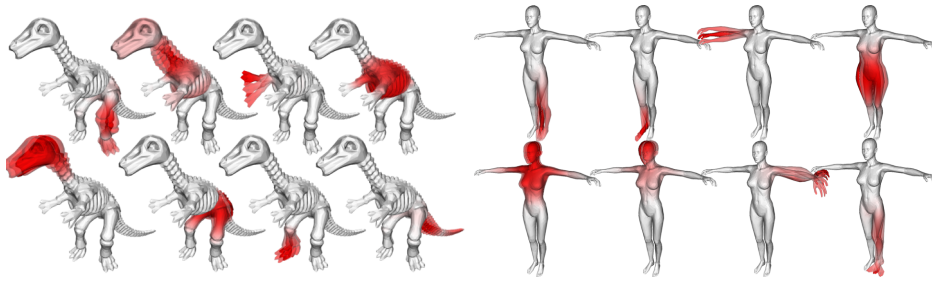


Figure 5.4: Left: The first, second, 19th and 20th L^1 regularized compressed vibration modes for $\mu = 255$. Right: The first, second, 19th and 20th L^1 constrained compressed vibration modes for $s = 11.5$, which is the value of the L^1 norm of the first L^1 regularized vibration mode for $\mu = 255$. The plots show the values of the elastic and sparsity terms in both cases.

Note, that the previous definitions of compressed modes takes all K modes into account in a single optimization problem, and thus applies the L^1 regularization to the whole matrix containing the modes. The problem of finding a suitable parameter μ remains similar: when computing a set of K and a set of \tilde{K} modes, it is not clear how to choose the values of μ , such that a similar sparsity pattern is achieved. Also note that the problem of varying degrees of localization for a fixed value of μ but a larger number of modes K is not specific to our setting of compressed vibration modes and we think that other methods using compressed eigenmodes will benefit from the constrained formulation.

Lastly, we want to clarify that both (the regularized and the constrained) formulations are valid and there are situations in which either might be preferable. When using the L^1 constrained formulation, we specify an approximate volume (or area) that is covered by the support of the vibration mode, which might not be desirable: in Figures 5.5a and 5.5b we see modes whose supports cover vastly different amounts of area or volume



(a) Eight of the first fifty compressed vibration modes on the dinosaur model (tetrahedral mesh, St.-V.-K. energy, $\mu = 20$). (b) Eight of the first fifty compressed vibration modes on the Victoria mesh (surface mesh, thin shells energy, $\mu = 0.0001$).

Figure 5.5: Two sets of compressed vibration modes for different geometry types and elastic energies.

respectively, but we intuitively convey these parts as functional units of the mesh.

5.7. APPLICATIONS

Before we cover the analysis of our optimization scheme, compare to previous methods and show examples for sets of compressed vibration modes, we will cover two applications that benefit from the localized deformations that have been introduced in this Chapter.

Compressed deformation bases, elastic simulation and deformation A general advantage of compressed vibration modes is that they can be stored efficiently by storing a list of the n non-zero indices along with the non-zero entries instead of storing all $N \gg n$ entries. This results in a significant reduction of the space required to store the deformation basis, which becomes essential in applications that impose strict memory requirements, *e.g.* when the GPU memory needs to be shared by different processes (games or VR) or when large or multiple reduced bases have to be stored at the same time (interactive simulations). Moreover, mapping from the subspace given by a sparse basis to the full space is cheaper (given a certain sparsity) than mapping from a dense basis to the full space since sparse matrix vector products can be computed more efficiently. In Table 5.4, we list the relative memory size of the sparse basis compared to the dense basis. We store the sparse deformations as a list of integers representing the indices of the non-zero vectors along with the three Doubles that represent the displacement vector. The dense deformations are simply stored as $N \cdot 3$ Double values representing all displacement vectors.

This raises the question of whether in a reduced-order simulation a basis of compressed vibration modes can compete with a dense basis of the same dimension. To test this, first we used the compressed vibration modes in an elastic simulation setup. In the following it is convenient to use the change of coordinates

$$\hat{\mathbf{u}} := \mathbf{M}^{1/2} \mathbf{u}, \quad (5.8)$$

under which the mass matrix becomes identity. Likewise, let $\hat{\mathbf{H}} = \mathbf{M}^{-1/2} \mathbf{H} \mathbf{M}^{-1/2}$, *i.e.* the

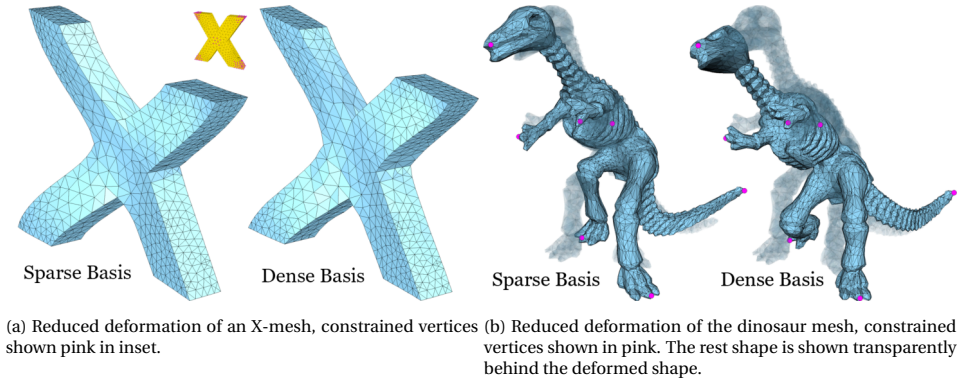


Figure 5.6: Reduced deformation using a basis of sparse and dense vibration modes respectively, for more details see Section 5.7.

5

elasticity Hessian expressed in these coordinates.

We consider the linearized equations of motion induced by the elasticity Hessian,

$$\ddot{\mathbf{u}} + \beta \hat{\mathbf{H}} \dot{\mathbf{u}} + \hat{\mathbf{H}} \mathbf{u} = 0, \quad (5.9)$$

where $\beta \in [0, 1]$ is a damping parameter. We reduce (5.9) using the basis of compressed vibration modes $\hat{\mathbf{U}} = (\hat{\mathbf{u}}_1, \dots, \hat{\mathbf{u}}_K)$, that is, $\hat{\mathbf{U}}$ is the $3|\mathcal{V}| \times K$ matrix which contains the compressed vibration modes, expressed in terms of the changed coordinates, as columns. Let $\hat{\mathbf{H}} = \hat{\mathbf{U}}^T \mathbf{H} \hat{\mathbf{U}}$ and let $\Phi = (\phi_1, \dots, \phi_K)$ be the matrix containing the eigenvectors of $\hat{\mathbf{H}}$ as its columns and $\Lambda = \text{diag}(\lambda_i)$ be the diagonal matrix containing the eigenvalues of $\hat{\mathbf{H}}$. We then consider the reduced, decoupled system

$$\ddot{\mathbf{x}} + \beta \Lambda \dot{\mathbf{x}} + \Lambda \mathbf{x} = 0. \quad (5.10)$$

These systems can be solved, given appropriate initial conditions, by using the closed form solutions for each dimension individually. As a post-processing step, we apply *rotation-strain warping*, see [37], to the results of the linearized simulation. For our experiments, we implemented a tool for interactive simulation, where initial velocities can be injected into the system by clicking on vertices of the mesh. When comparing results obtained in subspaces of the same dimension constructed from compressed and natural vibration modes, we obtain results of comparable quality, while the compressed basis takes about 20% of the memory of the dense basis.

By design the compressed modes \mathbf{u} induce low-energy deformations, *i.e.* $\mathbf{u}^T \mathbf{H} \mathbf{u}$ is small. Due to the localization, the compressed modes are less efficient than the natural vibration modes. However, we expect that the energy of the compressed modes converges to that of the natural modes when μ converges to zero. We provide experimental evidence in Figure 5.7, where we show that the energy $\mathbf{u}^T \mathbf{H} \mathbf{u}$ of the lowest compressed vibration mode (blue) (which is constrained to be orthogonal to the linearized rigid motions) converges to the eigenvalue of the first natural vibration (again orthogonal to the rigid motions) (orange) when μ goes to zero. A visual evaluation of this convergence can be seen in Figure 5.3a.

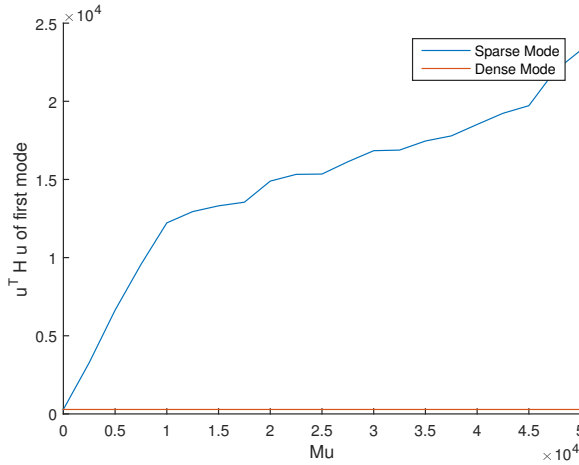


Figure 5.7: Convergence of sparse vibration modes to natural vibration modes as μ goes to 0.

As a second application of the compressed basis, we use compressed vibration modes in a reduced deformation-based shape editing setup. In Figure 5.6, we compare the results of such modeling sessions, using a basis of compressed and dense vibration modes of various sizes (10, 20 and 30 modes for the X-mesh; 20, 40 and 60 modes for the dinosaur).

Mesh	# vertices	K	Energy using compressed modes	Energy using dense modes
X	1098	10	2137.93	424.82
		20	287.33	292.27
		30	109.01	192.17
Dinosaur	27664	20	53.76	372.20
		40	20.47	152.46
		60	12.12	43.93

Table 5.1: Energy values of optimized static state solutions for the deformations shown in Figure 5.6.

More precisely, we minimize a weighted sum of the St.-Venant-Kirchhoff energy and a least squares term that drags a subset of the vertices to user specified positions. Instead of minimizing this energy over the full space, we optimize over the subspaces spanned by the compressed and dense vibration modes respectively. The compressed basis requires about 35%/20% of the storage space of the dense basis for the X-mesh and dinosaur mesh, respectively. The energy levels of the optimized solutions are listed in Table 5.1. It is remarkable that in almost all cases a lower energy deformation was found in the subspace spanned by compressed vibration modes.

Elastic segmentation The compressed vibration modes provide us with information about the dynamics of an object: the supports of the modes mark areas which are well suited as segments of the object that undergo deformations while the rest of the shape remains still. The size of these areas, *i.e.* the semantic level that should be covered by the modes, can be controlled via the parameter of the L^1 regularizer, as can be seen, for example, in Figures 5.3a and 5.12. This makes the compressed vibration modes well-suited to aid in segmentation tasks, where elastic behavior of the mesh should be considered. We implemented a simple segmentation algorithm, where we segment the mesh according to the supports of the modes by only considering a mode if there is not already another mode that covers it by more than 90%. We decide whether a tetrahedron belongs to the segment defined by a mode, if the deformation vector for this mode is larger than that of other modes that cover a nearby segment. Results of this segmentation scheme can be seen in Figure 5.8, where we show its performance on a volumetric and a surface mesh, using the modes from the experiments shown above. In both cases, $K = 50$ compressed modes were produced and the steps above lead to the depicted segmentations. The number of segments depends on the parameter μ and the elastic properties of the shape itself. For the dinosaur we end up with 10 segments, while for the centaur we get 18. In both cases, we can see some triangles and tetrahedrons that do not belong to any partition, or segments with rough borders. This could easily be fixed by extending the steps above, but we wanted to highlight the simplicity of this algorithm which already yields a useful segmentation that is based on elastic properties of the object.

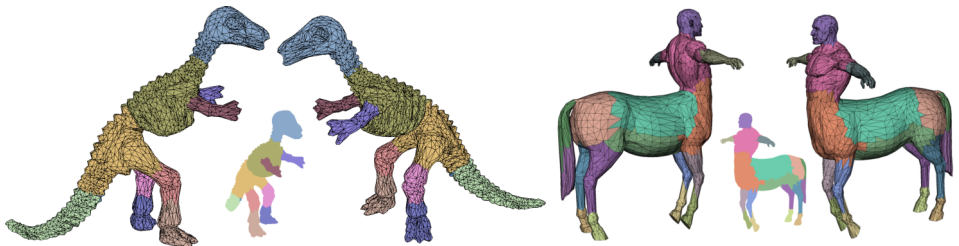


Figure 5.8: A simple segmentation scheme using our compressed vibration modes applied to the dinosaur (volumetric) and the centaur (surface) mesh. See Section 5.7 for details.

Segmentation using elasticity has previously been proposed by Huang et al. [18], where the natural vibration modes of an elastic object were weighted and used in a variant of k -means clustering to partition the mesh. The goal of the segmentation is to be able reproduce the 'likely, or typical' deformations of the vibration modes by using rigid transformations on each of the resulting segments. In contrast, we directly produce a set of likely, or typical sparse deformations whose support we use as the segments.

5.8. EXPERIMENTS

Consistency of ICCM A property of ICCM is that it will always converge to some local minimum, up to a desired precision: the sequence of functional values of the $\hat{\mathbf{u}}_{i,k}$ in (5.7) is monotonically decreasing for each fixed i and increasing k . Indeed, when minimizing



Figure 5.9: Comparison between the compressed manifold modes produced by our scheme (left) and Neumann et al. [2] (right). Both methods yield the same results up to the ordering and sign changes, while the computation times of our method are significantly lower.



Figure 5.10: Compressed vibration modes produced by using ADMM for $\mu = 10, 20, 30, 50, 75$. See Figure 5.1 for the output of our method.

the functional to find $\hat{\mathbf{u}}_{i,k}$, the previous solution $\hat{\mathbf{u}}_{i,k-1}$ is within the hyperplane to which the search space is restricted in that iteration, by construction. If a mode with lower energy value is found within the hyperplane, the functional value of the normalized solution will be strictly smaller since the functional is strictly convex. Moreover, each of the sub-problems can be solved robustly and up to a desired precision since the interior point methods provided by solvers like *MOSEK* offer various optimality guarantees for quadratic programs with inequality constraints. For ADMM, such convergence guarantees are not available, and indeed, as stated above, the density, size and structure of the matrices, as given in our setting of compressed vibration modes, prevented ADMM from converging to a meaningful solution for simple models, which brought up the necessity to develop an alternative optimization scheme.

For $\mu = 0$, the compressed vibration modes coincide with the natural vibration modes that are solutions of the generalized eigenvalue problem $\mathbf{H}\mathbf{u} = \lambda\mathbf{M}\mathbf{u}$. In this case, our algorithm can be seen as a scheme that uses convexification to solve an eigenvalue problem and it is important to verify that the solutions of our approach coincide with the ground truth (which can be obtained using reliable eigen-solvers). Indeed, for all meshes used in our experiments, we were able to find the first 20 vibration modes among the first 25 modes produced using our minimization scheme. The reason that we do not get a one-to-one correspondence to the ground truth is that we use a convexification that approximates the constraint $\|\mathbf{u}\|_2 = 1$. This may lead to finding a local minimum, instead of the global minimum. In our experiments, we were still able to find consistent sets of compressed vibration modes, by computing slightly more modes and cutting off the highest frequency modes.

In case $\mu > 0$, we first compare to the compressed manifold modes shown in Neumann et al. [2]: in Figure 5.9 we show the output of their adapted ADMM optimization scheme and our proposed ICCM scheme. We were able to reproduce the same modes, up to changes in the order in which they were found. This, in combination with the enhanced computation timings shown below, shows ICCM is a valid alternative to ADMM for the problem of finding compressed modes. Moreover, ADMM did not converge after 20000 iterations to any suitable solution when solving for compressed vibration modes: we used the elasticity Hessian and the corresponding mass matrix in the ADMM implementation of Neumann et al. [2] and received the modes shown in Figure 5.10, which aren't meaningful solutions to the optimization problem. See Figure 5.1 for a comparison to the output of our method.

Note that for $K > 1$, ICCM and ADMM try to solve two slightly different optimization problems (computing a sequence of modes or computing all modes at once), so the following quantitative comparison of the two algorithms was performed for $K = 1$. Additionally, for this comparison only, ICCM used the unweighted L^1 norm, such that the objective values will be comparable. We performed the comparison on various meshes and different triangulations. The values of μ were chosen such that the resulting modes had a support of about 20% of the mesh area. We used a randomized mode for both methods as an initialization (in ICCM we need an initial hyperplane constraint, ADMM needs an initial value to start an iterative procedure) and ran both algorithms 100 times with different random initializations. In Table 5.2 we list both the average as well as the best objective values found. From those values we conclude that in setups where ADMM does converge (which is not always the case, see above), both algorithms perform equally well. Thus, we met our goal of creating an algorithm that is able to compute vibration modes consistently and whose computation times scale linearly in the number of modes that should be produced.

Mesh	# vertices	μ	Obj. value of optimum for ICCM (average,best)	Obj. value of optimum for ADMM [2] (average,best)
Hand	868	0.0001	4.1755E-4, 5.5042E-4	4.2055E-4, 6.0002E-4
Hand	4054	0.0001	9.4314E-4, 1.4113E-3	9.4731E-4, 1.1586E-3
Fertility	4994	0.0002	2.1663E-2, 2.5329E-2	2.1683E-2, 2.6280E-2
Fertility	9994	0.00005	1.2519E-2, 1.3222E-2	1.2531E-2, 1.3815E-2
Bunny	34834	0.0001	5.1078E-2, 6.4719E-2	5.1078E-2, 7.0158E-2

Table 5.2: Comparison of objective values of the optima computed using our algorithm (ICCM) and using ADMM as proposed and implemented in Neumann et al. [2].

Lastly, note that ADMM requires the choice of a penalty parameter, for which Neumann et al. use an automatic adjustment strategy. It is unclear whether different strategies might lead to meaningful results the examples where ADMM did not converge. Our proposed algorithm is parameter free and converged in all examples for all choices of sparsity regularization and number of modes.

Compressed vibration modes and compressed manifold modes are unstable in their

order: usually there is a large set of compressed vibration modes with almost the same objective value, so small changes of the mesh, even isometric deformations, lead to a different order of the sparse eigenmodes. Therefore, none of our applications rely on a precise ordering of the modes. Note, however, that the modes can always be ordered after a certain number has been computed. [23] provides a natural ordering for compressed manifold modes that can be extended to compressed vibration modes as well. In our experiments, we always computed about 10%-25% more modes than required, ordered them and cut off the superfluous modes. This way we were able to get a consistent set of the K lowest compressed vibration modes.

In Figure 5.1, we show the first sixteen modes of an X-shaped volume mesh. The modes were computed in the order that is shown there, and putting them into groups of four shows how the symmetry of the mesh is properly reflected in the modes: each mode appears four times, once for each leg of the X.

Sparsity control Examples for compressed vibration modes, where the sparsity term was used as a regularizer are shown in Figures 5.1, 5.3a, 5.3b, 5.12, 5.5a and 5.5b. In Figure 5.3a, we show how the bar can be segmented into parts of arbitrary size (limited by the mesh resolution) by tuning the sparsity parameter μ . For each of those parts, we get multiple vibrations that span a space of deformations for that part of the mesh. In case of the dinosaur mesh shown in Figure 5.5a it is remarkable how the support of the modes is concentrated around parts of the skeleton that we would intuitively categorize as segments.

While for the rest of the experiments we used tetrahedral volume meshes with the St.-Venant-Kirchhoff energy, in Figures 5.12 and 5.5b we show that our method is not limited to this setup: there, compressed vibration modes for triangle surface meshes are shown, where the discrete shells energy was used as the underlying deformation energy. Figure 5.12 also demonstrates how we can localize the modes in different levels of scaling: for $\mu = 0.01$ we get deformations of single fingers as compressed vibration modes and for $\mu = 10^{-4}$ we get deformations of the legs and arms.



Figure 5.11: Sparsity constrained vibration modes of a hand mesh.

In Figure 5.11, we show compressed vibration modes where we constrained the L^1 norm such that the first mode approximately covers one of the fingers. Shown are the first ten modes. Higher modes exhibit more complicated vibrations of the fingers and other parts of the hands are only covered by very high modes. This shows the tendency of

compressed vibration modes to concentrate on parts that are easier to deform in isolation (*i.e.* the energy of these deformations is lower than that of comparably sized other parts).

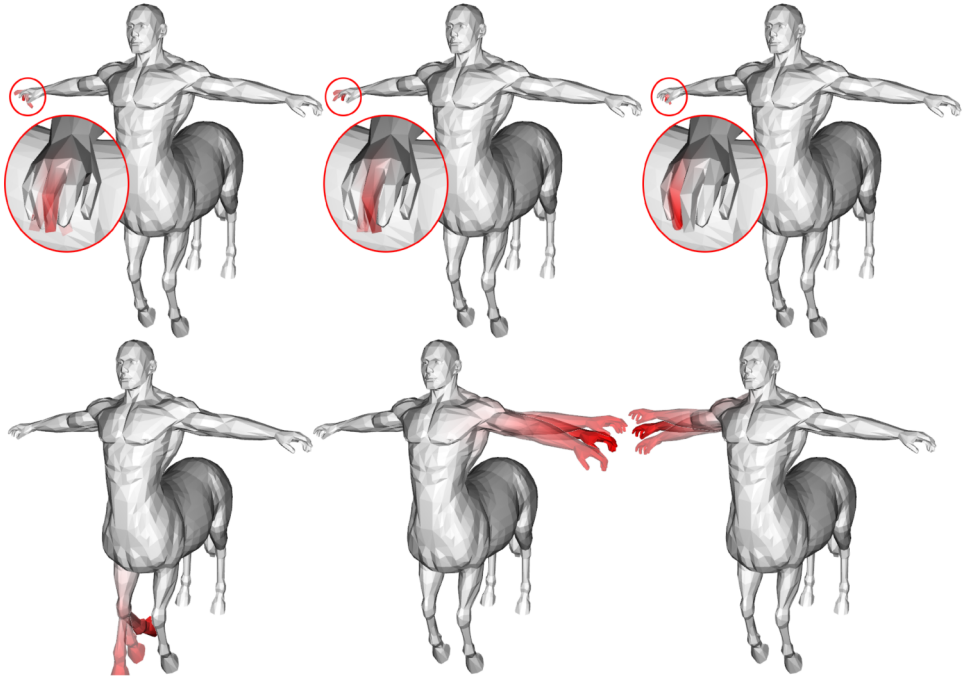


Figure 5.12: A surface mesh of a centaur with compressed vibration modes of the discrete shells energy. From left to right we show the first modes three for $\mu = 0.01$ and then the first three modes for $\mu = 10^{-4}$

In Figure 5.13, we show an advantage of both the correct approximation of the L^1 norm as well as the sparsity constrained formulation: we show that modes computed on a model with 28k vertices are similar to those computed on a simplification of that model with 5k vertices when using the same parameter to constrain the L^1 norm. The L^1 norm is only a meaningful measure when taking into account the volume associated to each vertex sample (*i.e.* using the entries of \mathbf{M} when computing the L^1 norm). Also, there is no guarantee that the elastic energies of both models behave the same, and thus a L^1 regularization can not be expected to have a similar effect on the localization of the modes. A L^1 constraint, however, still leads to a similar support of the modes on the two versions of the mesh, independently of the energy levels induced by the elastic energies on them.

Timings We first show that the computation times of the proposed ICCM scheme scale better when the number of modes, K , increases. Therefore, we compare the timings of our iterative optimization scheme as opposed to using ADMM to find a full set of modes at once. For the latter we use the implementation provided by the authors of [2]. Since ADMM did not converge to satisfying solutions when applied for sparse vibration modes (see previous paragraph), we compare the timings of ICCM and ADMM for the compu-

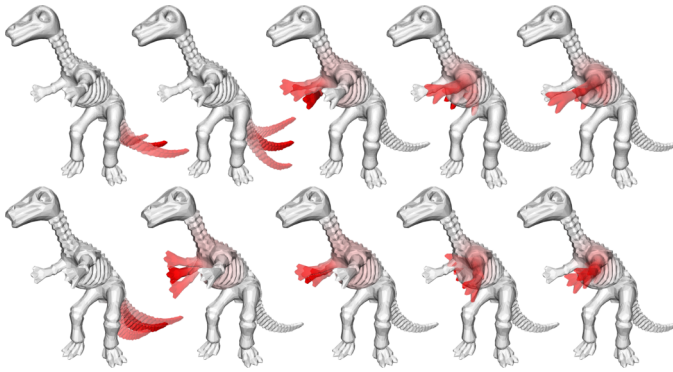


Figure 5.13: The first five sparsity constrained vibration modes on a high resolution and low resolution version of the dinosaur mesh (same value of s for both meshes).

tation of compressed manifold modes (*i.e.* sparse eigenmodes of the surface Laplacian). We used the same custom laptop to measure computation times for both algorithms and use the convergence criteria proposed in [2] for ADMM. When computing the modes using ICCM, we stopped the inner iterations to find a mode, when the changes in the objective functional were below 10^{-8} times the current value of the objective functional. As shown before, in Figure 5.9, for the right values of μ in each algorithm the produced modes are visually indistinguishable (after sign flips and reordering). The resulting timings are listed in Table 5.3. We provide visual comparisons of the computed modes in Figure 5.9 and as inset images in Table 5.3.

For some models and values of K , ADMM did not converge after 20k iterations using the convergence criteria given in the implementation by Neumann et al. [2], we marked these timings in the Table. We get faster computation times in general for smaller models and have a better scaling in computation times when computing a larger number of modes. For large meshes and low values of K , ADMM benefits from the fact that the iterations can be performed at comparably low costs, but as the size of the problem grows,



Mesh	# vertices	times for $K = 10$ (us / ADMM [2])	times for $K = 40$ (us / ADMM [2])	times for $K = 80$ (us / ADMM [2])
Hand See Figure 5.9	868	2.26s / 4.17s	12.47s / 144.64s*	23.25s / 435s*
Fertility 	4994	32s / 30s	123s / 422s	254s / 899s
Bunny 	34834	1007s / 312s	4072s / 3459.82s	8110s / 17205s*

Table 5.3: Comparison of computation times for compressed manifold modes when using ICCM (our scheme) versus the modified ADMM algorithm, as proposed in [2]. The inset pictures show the first three modes of each method, first ours, then ADMM.

*: ADMM did not converge after 20k iterations under the convergence criteria given in the implementation by Neumann et al.

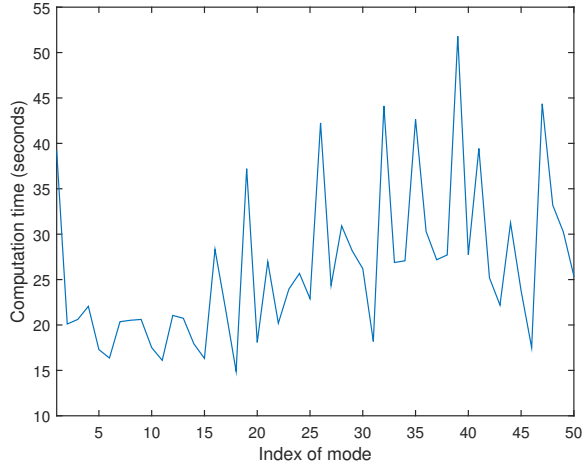


Figure 5.14: Times for the computation of each individual mode of the dinosaur model shown in Figure 5.13, lower row, when using ICCM.

each iteration becomes more costly, and more iterations are required for the method to converge. Thus, ICCM eventually outperforms ADMM as the number of desired modes grows. With our scheme, for each additional mode, we have to solve another sequence of quadratic programs with an additional equality constraint (orthogonality to the last mode). Whenever an additional equality constraint is added to the system, a change of variables can be performed, such that the variables automatically fulfill the constraints. As a result, the computation time per mode remains approximately constant, even for higher modes. In contrast, when using ADMM, as proposed in [2], the number of modes K that have to be computed has to be fixed before starting the computation and the number of variables is $N \cdot K$, where N is the number of vertices of the mesh. During minimization several steps are being performed that scale superlinear in the number of modes. Additionally, in our tests, ADMM requires more iterations in order to converge when the number of modes is high.

An overview of the computation times for several examples of compressed vibration modes can be found in Table 5.4. The computation times strongly depend on the size of the model and the number of modes, but also on the shape itself and the resulting elasticity Hessian. However, as pointed out before, the time to compute each mode is roughly the same, which means that computing a large number of modes is entirely possible. This is shown in Figure 5.14 where we show the computation times for each individual mode of the dinosaur mesh.

Computation of the Hessian Before (compressed) vibration modes can be computed the matrix \mathbf{H} has to be set up. In the following, we list several options on how to compute the Hessian. For a general discrete elastic energy, once the evaluation of the energy with

Mesh	# vertices	constrained?	parameter	average time per mode	average # of iterations per mode	rel. size of sparse versus dense basis
Bar (Fig. 5.3)	418	no	$\mu = 512$	1.97s	12.06	38%
Bar (Fig. 5.4)	418	yes	$s = 11.5$	2.1s	11.02	38%
Centaur (Fig. 5.12)	2645	no	$\mu = 0.01$	16.13s	8	2%
Hand (Fig. 5.11)	2584	yes	$s = 6.46$	14.72s	17.51	17%
Dinosaur lo-res. (Fig. 5.13, lower row)	4713	no	$\mu = 20$	25.17s	10.22	16%
Dinosaur hi-res. (Fig. 5.13, upper row)	27664	yes	$s = 6.5$	518.5s	13.4	16%

Table 5.4: Computation times for different sets of compressed vibration modes when using our proposed algorithm.

respect to a fixed rest-shape \mathbf{x} and a variable displacement vector \mathbf{u} is implemented as a function that takes an array of $3|\mathcal{T}|$ values and returns a single value denoting the energy associated to this displacement, one can use automatic differentiation (*e.g.* ADOL-C, see [38]) in order to obtain the Hessian matrix. Often, however, the Hessian can be evaluated faster by using explicit formulas. For energies that use discrete bending forces, Tamstorf et al. [39] documented closed form expressions for the Hessian of the bend angle, which allows explicit formulas for the energy Hessians of a large class of elastic energies. The Hessian of the discretized St. Venant–Kirchhoff energy for tetrahedral meshes is available as part of the Vega FEM library [40]. Also note, that for our purposes, we only require the evaluation of the energy Hessian at the rest configuration. A simple formula for the Hessian at the rest configuration for a certain class of deformation energies is detailed in [16].

5.9. CONCLUSION

We introduce compressed vibration modes of elastic bodies, which are orthonormal systems of displacements of objects that induce local and low-energy deformations. For the computation of the modes, we devise a novel minimization scheme which proves to be stable and the resulting computation time scales linearly in the number of modes, as opposed to previous methods. The compressed vibration modes are shown to be intuitively controllable via either tuning the sparsity regularizer or imposing a sparsity constraint (the latter of which has not been done before for comparable modes). The modes are shown to be stable under refinement and a correction of the L^1 term leads to invariance of the triangulation.

We show how the modes can extend applications such as reduced elastic simulation and deformation or mesh segmentation. Their compressible structure is attractive for cases in which memory limitations are present (*e.g.* games and VR, where GPU memory has to be handled efficiently). Moreover they tend to exhibit a semantic structure, in the sense that the support of the modes is often located on intuitive parts of objects, such as hands, arms, legs or fingers, depending on the choice of μ .

Challenges and limitations We are convinced that compressed vibration modes can be a useful tool in various areas of geometry processing and we can see direct benefits for simulation, shape space exploration and mesh analysis. While computation times

scale linearly, the timings per mode for detailed meshes is quite high, and does not allow for interactive re-computation of the modes, under different parameters or different rest configurations. Thus, it might be worthwhile to further investigate alternative L^1 optimization schemes, or to provide a multi-resolution scheme for the proposed ICCM optimization.

Another direction of future work might be to exchange the L^1 regularization by a regularization that penalizes the volume of the support of the modes. This would lead to a mass weighted ℓ_0 problem. Solving such problem poses a challenging problem.

REFERENCES

- [1] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher, *Compressed modes for variational problems in mathematics and physics*, Proceedings of the National Academy of Sciences **110**, 18368 (2013).
- [2] T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker, *Compressed manifold modes for mesh processing*, Comput. Graph. Forum **33**, 35 (2014).
- [3] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, *Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks*, Computer Graphics Forum **34**, 13 (2015).
- [4] A. Kovnatsky, K. Glashoff, and M. M. Bronstein, *MADMM: a generic algorithm for non-smooth optimization on manifolds*, arXiv preprint arXiv:1505.07676 (2015).
- [5] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Interactive surface modeling using modal analysis*, ACM Trans. Graph. **30**, 119:1 (2011).
- [6] K. K. Hauser, C. Shen, and J. F. O'Brien, *Interactive deformation using modal analysis with constraints*. in *Graphics Interface*, Vol. 3 (2003) pp. 16–17.
- [7] J. Barbič and D. L. James, *Real-time subspace integration for St. Venant-Kirchhoff deformable models*, ACM Trans. Graph. **24**, 982 (2005).
- [8] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, *An efficient construction of reduced deformable objects*, ACM Trans. Graph. **32**, 213:1 (2013).
- [9] Y. Yang, D. Li, W. Xu, Y. Tian, and C. Zheng, *Expediting precomputation for reduced deformable simulation*, ACM Trans. Graph. **34**, 243:1 (2015).
- [10] J. N. Chadwick, S. S. An, and D. L. James, *Harmonic shells: A practical nonlinear sound model for near-rigid thin shells*, ACM Trans. Graph. **28**, 119:1 (2009).
- [11] D. Li, Y. Fei, and C. Zheng, *Interactive acoustic transfer approximation for modal sound*, ACM Trans. Graph. **35**, 2:1 (2015).
- [12] J. Barbič, F. Sin, and E. Grinspun, *Interactive editing of deformable simulations*, ACM Trans. Graph. **31**, 70:1 (2012).
- [13] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Interactive spacetime control of deformable objects*, ACM Trans. Graph. **31**, 71:1 (2012).

- [14] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, *Animating deformable objects using sparse spacetime constraints*, ACM Transactions on Graphics (TOG) **33**, 109:1 (2014).
- [15] S. Li, J. Huang, F. de Goes, X. Jin, H. Bao, and M. Desbrun, *Space-time editing of elastic motion through material optimization and reduction*, ACM Trans. Graph. **33**, 108:1 (2014).
- [16] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Eigenmodes of surface energies for shape analysis*, in *Proceedings of Geometric Modeling and Processing* (2010) pp. 296–314.
- [17] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Modal shape analysis beyond Laplacian*, Computer Aided Geometric Design **29**, 204 (2012).
- [18] Q. Huang, M. Wicke, B. Adams, and L. Guibas, *Shape decomposition using modal analysis*, Computer Graphics Forum **28**, 407 (2009).
- [19] T. R. Langlois, S. S. An, K. K. Jin, and D. L. James, *Eigenmode compression for modal sound models*, ACM Trans. Graph. **33**, 40:1 (2014).
- [20] V. Ozoliņš, R. Lai, R. Cafilisch, and S. Osher, *Compressed plane waves yield a compactly supported multiresolution basis for the Laplace operator*, Proceedings of the National Academy of Sciences **111**, 1691 (2014).
- [21] A. Mackey, H. Schaeffer, and S. Osher, *On the compressive spectral method*, Multi-scale Modeling & Simulation **12**, 1800 (2014).
- [22] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning **3**, 1 (2011).
- [23] K. Houston, *Compressed manifold modes: Fast calculation and natural ordering*, arXiv preprint arXiv:1507.00644 (2015).
- [24] A. M. Bronstein, Y. Choukroun, R. Kimmel, and M. Sela, *Consistent discretization and minimization of the L^1 norm on manifolds*, arXiv preprint arXiv:1609.05434 (2016).
- [25] X.-T. Yuan and T. Zhang, *Truncated power method for sparse eigenvalue problems*, Journal of Machine Learning Research **14**, 899 (2013).
- [26] M. Meyer and J. Anderson, *Key point subspace acceleration and soft caching*, ACM Trans. Graph. **26**, 74:1 (2007).
- [27] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt, *Sparse localized deformation components*, ACM Trans. Graph. **32**, 179:1 (2013).
- [28] L. Gao, G. Zhang, and Y. Lai, *L^p shape deformation*, Science China Information Sciences **55**, 983 (2012).

- [29] L. He and S. Schaefer, *Mesh denoising via l_0 minimization*, ACM Trans. Graph. **32**, 64:1 (2013).
- [30] H. Kang, F. Chen, Y. Li, J. Deng, and Z. Yang, *Knot calculation for spline fitting via sparse optimization*, Computer-Aided Design **58**, 179 (2015).
- [31] C. Brandt, H.-P. Seidel, and K. Hildebrandt, *Optimal spline approximation via l_0 -minimization*, Computer Graphics Forum **34**, 617 (2015).
- [32] L. Xu, R. Wang, J. Zhang, Z. Yang, J. Deng, F. Chen, and L. Liu, *Survey on sparsity in geometric modeling and processing*, Graphical Models **82**, 160 (2015).
- [33] J. Bonet and R. D. Wood, *Nonlinear Continuum Mechanics for Finite Element Analysis* (Cambridge University Press, 2008).
- [34] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder, *Discrete shells*, in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003) pp. 62–67.
- [35] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodological) , 267 (1996).
- [36] R. M. Rustamov, *Multiscale biharmonic kernels*, Computer Graphics Forum **30**, 1521 (2011).
- [37] J. Huang, Y. Tong, K. Zhou, H. Bao, and M. Desbrun, *Interactive shape interpolation through controllable dynamic deformation*, IEEE Transactions on Visualization and Computer Graphics **17**, 983 (2011).
- [38] A. Griewank, D. Juedes, and J. Utke, *Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Softw. **22**, 131 (1996).
- [39] R. Tamstorf and E. Grinspun, *Discrete bending forces and their Jacobians*, Graphical Models **75**, 362 (2013).
- [40] F. S. Sin, D. Schroeder, and J. Barbič, *Vega: Non-linear fem deformable object simulator*, Computer Graphics Forum **32**, 36 (2013).

APPENDIX

5.A. ANALYSIS OF THE ICCM SCHEME FOR THE L^1 CONSTRAINED OPTIMIZATION

As part of ICCM, we bound the weighted L^1 norm of a mode $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$ with $\mathbf{u}^+, \mathbf{u}^- > 0$ from above, via the linear term $\ell(\mathbf{u}^+, \mathbf{u}^-) = \mathbf{1}^T \mathbf{M} \mathbf{u}^+ + \mathbf{1}^T \mathbf{M} \mathbf{u}^-$. When computing sparse vibration modes via ICCM, in case the L^1 term is used as a regularizer, the minima found in each iteration have an L^1 norm that is equal to $\ell(\mathbf{u}^+, \mathbf{u}^-)$. In the following, we prove that this also holds for the L^1 constrained problem. Note that $\|\mathbf{u}^+ - \mathbf{u}^-\| = \ell(\mathbf{u}^+, \mathbf{u}^-)$ if for each i it is either $\mathbf{u}_i^+ = 0$ or $\mathbf{u}_i^- = 0$, *i.e.* we need to show that for no coordinate both the positive and the negative part of the variables are greater than 0.

In the constrained optimization, after having computed the first $i - 1$ modes, the inner loop of ICCM has the following form (where \mathbf{c} is the current hyperplane, used to convexify the L^2 constraint):

$$(\mathbf{u}^+, \mathbf{u}^-) := \begin{cases} \operatorname{argmin}_{\mathbf{u}^+, \mathbf{u}^-} \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \begin{pmatrix} \mathbf{H} & -\mathbf{H} \\ -\mathbf{H} & \mathbf{H} \end{pmatrix} \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix} \\ \text{subject to } \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ -\mathbf{c} \end{pmatrix} = 1 \\ \text{and } \forall j < i: \begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{u}_j \\ -\mathbf{u}_j \end{pmatrix} = 0 \\ \text{and } \mathbf{u}^+, \mathbf{u}^- \geq 0 \\ \text{and } \ell(\mathbf{u}^+, \mathbf{u}^-) = s \end{cases} \quad (5.11)$$

In the following we make the assumptions that there is indeed a feasible solution (in particular s is large enough to allow $\begin{pmatrix} \mathbf{u}^+ \\ \mathbf{u}^- \end{pmatrix}^T \mathbf{M} \begin{pmatrix} \mathbf{c} \\ -\mathbf{c} \end{pmatrix} = 1$) and that we choose s smaller than the L^1 norm of the solution of the same problem without the constraint $\ell(\mathbf{u}^+, \mathbf{u}^-) = s$. We will now show the following: either we have that $\ell(\mathbf{u}^+, \mathbf{u}^-) = \|\mathbf{u}^+ - \mathbf{u}^-\|_1$ or there is a mode $\tilde{\mathbf{u}}$ with L^1 norm smaller than s but with the same value for the quadratic term. Indeed, suppose that there is j , such that $\mathbf{u}_j^+ > \mathbf{u}_j^- > 0$ (the case $\mathbf{u}_j^- > \mathbf{u}_j^+ > 0$ can be handled equivalently). Let $\tilde{\mathbf{u}} = \tilde{\mathbf{u}}^+ - \tilde{\mathbf{u}}^-$, where $\tilde{\mathbf{u}}_i^+ = \mathbf{u}_i^+$ and $\tilde{\mathbf{u}}_i^- = \mathbf{u}_i^-$ for $i \neq j$ and $\tilde{\mathbf{u}}_j^+ = \mathbf{u}_j^+ - \mathbf{u}_j^-$ and $\tilde{\mathbf{u}}_j^- = 0$. Note that $\ell(\tilde{\mathbf{u}}_j^+, \tilde{\mathbf{u}}_j^-) = \|\tilde{\mathbf{u}}^+ - \tilde{\mathbf{u}}^-\|_1$ and that $\tilde{\mathbf{u}}$ still satisfies all constraints from (5.11) and that it has the same value for the quadratic term (this is clear by construction

of the energy functional and constraints after splitting the variables into positive and negative parts).

This however would imply, that in the optimization problem (5.11), when we replace the constraint $\ell(\mathbf{u}^+, \mathbf{u}^-) = s$ by the constraint $\|\mathbf{u}^+, \mathbf{u}^-\|_1 = s$, this constraint would not be active, which contradicts our assumption, that s was chosen smaller than the L^1 norm of the solution of (5.11) without the constraint $\ell(\mathbf{u}^+, \mathbf{u}^-) = s$.

6

OPTIMAL SPLINE APPROXIMATION VIA ℓ_0 -MINIMIZATION

This chapter is based on the publication *Optimal Spline Approximation via ℓ_0 -Minimization* by Christopher Brandt, Hans-Peter Seidel and Klaus Hildebrandt, published in Computer Graphics Forum in 2015.

6.1. OVERVIEW

In all previous chapters, we used triangular or tetrahedral meshes to approximate smooth 3d shapes. In this chapter, we will consider curve-like data, that is, d -dimensional data that smoothly varies over a parameter. Examples are hand-drawn 2-dimensional curves, 3-dimensional outlines of CAD drafts or high dimensional motion data that varies in time. While such data can be approximated by a polygonal structure like meshes, it is often beneficial to consider analytic approximations of such data, such as splines.

Splines are widely used in graphics for the approximation of functions and parametrized curves. They combine an efficient representation of “smooth” functions by few parameters and with good approximation properties: under refinement, interpolating splines converge to a (smooth enough) function in various norms.

Here we consider the question of *optimal approximating splines* in a general setting, where not only variations of the spline parameter, but also of the number of spline segments and the locations of the knots are allowed. That is, we are looking for a compromise between approximation error and the number of spline segments. Splines with a low number of segments offer reduced order descriptions of the original data, and as such facilitate and accelerate the processing of it. As such, the optimal spline approximations considered in this chapter can be seen as a reduction approach for curve-like data. The resulting description of the data not only reduces the dimensionality of subsequent processing applications, but also facilitates the evaluation of functionals on the data.

In the following, we will derive a discrete version of the regularized optimal spline approximation problem, in which the domain of the spline is uniformly sampled and the feasible set is restricted to splines whose knots are grid points. We show that the discrete problem can be written as an ℓ_0 -regularized minimization problem. In this setting, splines are represented by their function values (and depending on the type of spline also their derivatives) at the grid points. The main ingredient to our modeling of the optimization problem is a linear operator on the space of functions on the grid that has the property that the ℓ_0 -norm of the image of a function agrees with the number of segments of the corresponding spline, which interpolates the function values. We show how such an operator can be constructed for different families of splines including B-splines, composite Bézier curves, splines in tension, and wiggly splines.

In recent years, the ℓ_0 -regularization of optimization problems has received much attention and many efficient algorithms for approximating the solutions have been proposed. Formulating the optimal spline approximation problem as an ℓ_0 -minimization problem allows us to use this rapidly growing pool of algorithms for computing optimal splines. Moreover, we see a connection between optimal spline approximation and recent schemes for image and geometry denoising, which yield a similar type of optimization problem. We have tested various algorithms for solving the ℓ_0 -regularized optimization problem and propose a variant of the recent scheme by Patrascu and Necoara [1]. We tested our implementation for spline approximation of planar and space curves and for spline conversion of motion capture data.

6.2. RELATED WORK

Optimal splines. The computation of optimal approximating splines comes at different complexities depending on what parameters of a spline are varied. Typically, only the parameters the spline depends linearly on are optimized, which leads to linear least-squares problems. For B-splines, this means that a knot vector is fixed and the remaining parameters are optimized. Latest solvers for this problem can compute highly accurate solutions within few milliseconds even for a large number of points [2]. However, the approximation can be greatly improved by treating the knots as free variables, see [3–5] for some early work in this direction. The task is to find an optimal knot vector, in the sense that the best approximating spline on this knot vector yields the lowest approximation error among all other splines with the same number of knots. Strategies to find an optimal knot vector can be roughly classified in three categories:

- The knot vector is iteratively extended using heuristics, such as integrated discrete curvature [6], largest accumulated \mathcal{L}_2 error on a segment [2], or others [7–9]. These methods often require initial knot vectors, user-defined error bounds and other parameters.
- An existing spline gets simplified by iteratively removing knots from the knot vector (*knot removal*), see e.g. [10]. In each iteration, the knots get ranked via some heuristic criterion and then greedily removed, while maintaining some prescribed proximity to the input curve.
- The knots are regarded as additional free variables (subject to appropriate con-

straints) and the approximation error of the best spline using these knots is minimized. This, however, leads to a very involved minimization problem (cf. [3]), and the latest attempts at solving it use genetic algorithms, which interpret the knot vectors as populations which undergo mutations [11–14] or apply particle swarms [15].

Our method differs from these approaches in that we are neither limited to B-splines nor is there a need to use explicit geometric constructions or any heuristics in our formulation to find the target set of points which we use to create the spline approximating the input curve. Furthermore, we do not need to fix the number of knots, but keep it variable. By discretizing the parameter interval, we are able to reformulate the optimal spline approximation problem as a ℓ_0 -regularized quadratic minimization problem. This opens the door to using recent approximation algorithms for ℓ_0 -minimization for optimal spline approximation. For arc splines—curves consisting of a finite sequence of circular arcs and line segments—a geometric approach for approximating a sequence of points by a G^1 -continuous arc spline with a minimum number of segments has been recently proposed by Maier [16]. Contemporaneous to our research, Kang et al. [17] proposed an equivalent reformulation of the optimal spline problem for the case of B-splines in one dimension. However, they replace the ℓ_0 -regularized functional by an ℓ_1 -regularized functional. We further discuss and compare the ℓ_0 - to the ℓ_1 -regularization in Section 6.5.

Optimal polylines. Optimal approximation by polylines has a more local character: changing the position of knots will have no global effect on the solution, as is the case for splines, where differentiability conditions introduce global dependencies. This local property of polylines (and other types of curve which are only demanded to be continuous) allows the use of Dynamic Programming algorithms, which can give optimal approximations (given either a maximum number of segments, a cost per segment or a maximum approximation error in the ℓ_2 or ℓ_∞ norm) by polylines [18, 19], line segments and circular arcs [16, 20], or piecewise polynomials [21].

ℓ_0 -Minimization. Sparsity-regularized and constrained convex optimization problems are the focus of recent research, see *e.g.* [1, 22, 23]. Though such optimization problems are known to be *NP*-hard, various efficient approximation algorithms have been proposed in recent years, see [1, 24–27] and references therein. Whereas typically sparsity is demanded in the variable which is minimized, in our case, we want to apply the regularization on a linear transform of the variable. This type of minimization has been considered in recent sparse image processing [26, 28] and geometry processing [29] applications. The problem has been analyzed and several algorithms have been proposed [26, 27, 30]. We applied, adapted and compared several of these recently proposed algorithms to approach our specific ℓ_0 -regularized minimization problem.

6.3. OPTIMAL SPLINE APPROXIMATION VIA ℓ_0 -MINIMIZATION

Our approach for optimal spline approximation can be applied for various types of splines, including B-splines and composite Bézier curves. In this section, we first outline a continuous version of the optimization problem and then introduce the ℓ_0 -minimization problem, which is a discrete version of the continuous problem. We explicitly describe

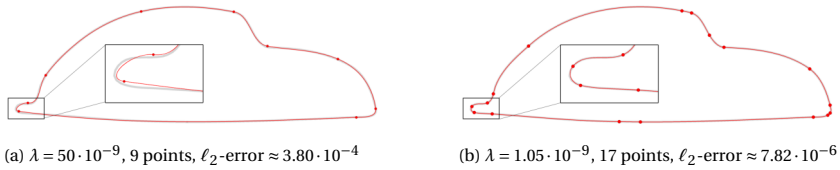


Figure 6.1: Cross section of a Beetle, 500 points, approximated using composite cubic Bézier curves generated with our method.

the optimization problem for C^2 cubic B-splines and composite cubic Bézier curves. Finally we discuss the generalization to other types of splines.

Continuous problem. Let us consider a parametrized curve $c: I \rightarrow \mathbb{R}^d$. Here I is either $[0, 1]$ in the case of curves with boundary, or $[0, 1]_{/0 \sim 1}$ (the unit interval with identified boundary) in the case of closed curves. Our goal is to construct a spline $s: I \rightarrow \mathbb{R}^d$ that is an optimal trade-off between the approximation of c on the one hand and the number of spline nodes on the other hand (a *node* being an interpolation point together with the corresponding parameter value). For this, we consider the functional

$$\mathcal{E}(s) = \|c - s\|_{\mathcal{L}_2}^2 + \lambda v(s) \quad (6.1)$$

where $v(s)$ is the number of nodes of s and $\lambda \in \mathbb{R}_+$. Then an optimal approximating spline is a minimizer of the functional among the set of all splines of one type. The parameter λ provides control of the complexity (number of nodes) of the spline. The optimal spline has the property that there is no spline \tilde{s} with $v(\tilde{s}) > v(s)$ such that the approximation error $\|c - \tilde{s}\|_{\mathcal{L}_2}^2$ is decreased by more than $(v(\tilde{s}) - v(s))\lambda$.

B-spline functions. We consider a discretized version of the optimization problem. For simplicity of presentation, we describe the case of C^2 cubic B-spline approximation of functions first, and generalize to other types of splines and the approximation of curves in \mathbb{R}^d later.

Let $t_1 < t_2 < \dots < t_N$ be a dense uniform partition of the interval I . The data we want to approximate is a set of N function values $p_i \in \mathbb{R}$, with corresponding parameter values $t_i \in I$, e.g. a sampling of the continuous input function. A C^2 cubic B-spline can be characterized as the unique (C^2 -continuous) cubic spline that interpolates a set of \tilde{N} nodes $(\tilde{t}_i, q_i) \in I \times \mathbb{R}$ and satisfies certain boundary conditions. Computing the corresponding control points amounts to solving a linear system of equations. Here, we will represent a cubic spline by the choice of the interpolating points and times. To be able to optimize over a set of cubic splines with a variable number of segments and variable nodes, we restrict the \tilde{t}_i to be values of the aforementioned partition of I . With this restriction, any cubic spline s can be represented by the vector $q \in \mathbb{R}^N$ listing the images $q_i = s(t_i) \in \mathbb{R}$ for all t_i . For any q , we call those q_i that are not nodes of the spline corresponding to q the *inner points* of q .

The inner points can be characterized as follows: Let $s(t)$ be a cubic spline, interpolating the nodes (t_i, q_i) , i.e. $s(t_i) = q_i$, with some prescribed boundary conditions at t_1 and t_N , and now assume that $s'''(t)$ is continuous at $t = t_j$ for some $j \in 2, \dots, N-1$. Then the cubic polynomials $s(t)|_{t \in [t_{j-1}, t_j]}$ and $s(t)|_{t \in [t_j, t_{j+1}]}$ are the same, since the values and all three derivatives at $t = t_j$ agree. This means that the cubic splines interpolating the

nodes $\{(t_1, q_1), \dots, (t_1, q_N)\}$ and $\{(t_1, q_1), \dots, (t_1, q_N)\} \setminus (t_j, q_j)$ are the same. Conversely, if for some t the third derivative s''' has a discontinuity, then $s(t)$ must be a node of s . So we have that q_j is an inner point if, and only if, the third derivatives of s from the left and from the right of t_j agree. This condition is linear in the coordinates representing the spline, so there is a linear operator $C : \mathbb{R}^N \rightarrow \mathbb{R}^N$ with the property that q_j is an inner point, if and only if $(Cq)_j = 0$. The operator C depends on the choice of boundary conditions. In Appendix 6.A, we discuss the construction of this operator explicitly. Given a vector $q \in \mathbb{R}^N$, we can construct the resulting cubic spline by interpolating all nodes of q (those for which $(Cq)_j \neq 0$) by a cubic spline s with appropriate boundary conditions. Then, all inner points of q will lie on this resulting spline.

The number $\nu(s)$ of nodes of the cubic spline s corresponding to a vector $q \in \mathbb{R}^N$ agrees with the ℓ_0 -norm of Cq . Our discrete analog of the energy (6.1) is

$$E(q) = \|p - q\|_{\ell_2}^2 + \lambda \|Cq\|_{\ell_0}, \quad (6.2)$$

where $p \in \mathbb{R}^N$ lists the input data points. Computing an optimal approximating spline amounts to minimizing (6.2) over all $q \in \mathbb{R}^N$ and computing the cubic B-spline s corresponding to the minimizer q . In the case that the interval is $I = [0, 1]$, one can additionally enforce interpolation of the boundary, *i.e.* $q_1 = p_1$ and $q_N = p_N$. Boundary conditions on the derivatives are incorporated into the operator C and the reconstruction of the spline s from a vector q .

B-splines curves. To approximate curves in \mathbb{R}^d , our construction remains almost unchanged: the curve samples p_i and our spline representation q_i are elements of \mathbb{R}^d and we write the list of points p and q as $N \times d$ matrices. This way, the energy (6.2) remains the same except that the ℓ_2 norm gets replaced by the Frobenius norm $\|\cdot\|_F$, and the ℓ_0 term is interpreted row-wise, *i.e.* $\|Cq\|_{\ell_0}$ counts the number of rows of $Cq \in \mathbb{R}^{N \times d}$ which contain non-zero entries.

Composite cubic Bézier curves. For composite cubic Bézier curves (CCBC), the derivation is similar to that of B-splines. These curves are composed of cubic polynomials, but instead of demanding the second derivative to be continuous, first derivatives at all nodes can be prescribed. As for the B-splines, we represent CCBCs with a variable number of nodes, by storing $q_i = s(t_i)$ for all t_i of the uniform partition of I . However, for representing the CCBCs, we additionally store the derivatives $q'_j = s'(t_j)$. Then, any CCBC whose nodes are a subset of the partition t_1, t_2, \dots, t_N can be reconstructed.

For initialization, the q'_i s can be estimated from the input data p or explicitly computed if the input is a parametrized curve. Then, we search for an optimized set of points and derivatives (q, q') , which means that the conditions for points to be inner points are posed on q and q' . The characterization of the inner points for CCBCs is similar to that for B-splines. However, in contrast to the case of B-splines, the second derivative is not guaranteed to be continuous, and thus we have to pose two conditions on inner points, namely that the second and third derivatives are continuous at the corresponding t_i . Again, these conditions are linear and can be formulated via a linear operator $C : \mathbb{R}^{2N} \rightarrow \mathbb{R}^{2N}$, such that p_j is an inner point, if $(C(p, p'))_k = 0$ and $(C(p, p'))_l = 0$, where k and l are the indices of the rows in $C(p, p')$ which are associated to the condition that the second and third derivatives are continuous at t_j . The construction of the operator is discussed in more detail in Appendix 6.A. Using it, we can formulate the energy

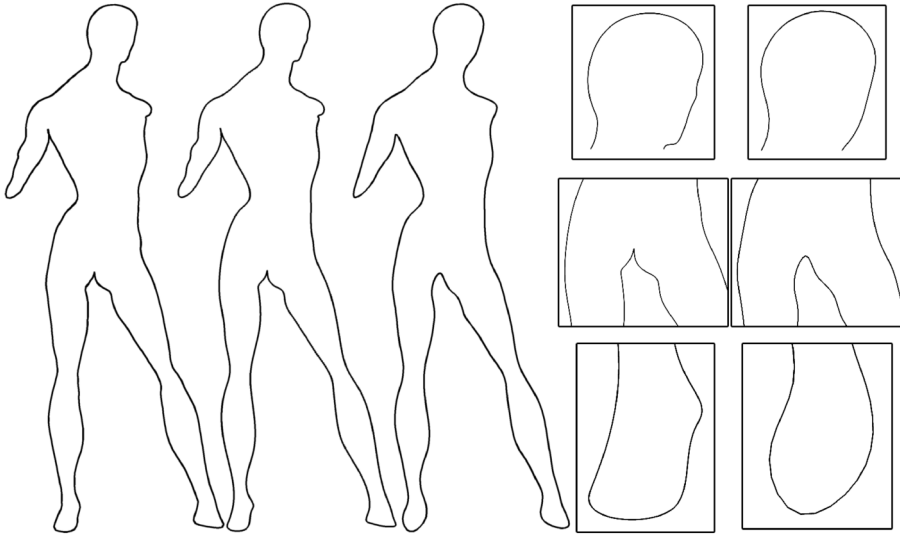


Figure 6.2: Comparison of our method to equidistant interpolation points. Left: input (500 points), middle: our method, cubic spline through 100 points, right: cubic spline through 100 equidistant points. On the right side are magnified regions of our result (left boxes) and the equidistant result (right boxes).

6

associated to CCBCs as

$$E(q, q') = \|p - q\|_F^2 + \lambda \|C(q, q')\|_{\ell_0}, \quad (6.3)$$

where the ℓ_0 term is now counting the number of points for which one or more of the $2 \cdot d$ conditions are not satisfied. To compute an optimal CCBC, we minimize the energy (6.3) over all q and q' and reconstruct the CCBC from this data. Boundary conditions can be enforced by imposing equality constraints onto q_1 , q_N , q'_1 , and q'_N .

Other spline types. B-splines and composite Bézier curves of higher order can be treated in the same vein as above, by introducing more conditions per point and potentially additional variables.

More types of splines can be covered by the following construction: suppose, that the interpolating spline through the nodes (q_i, t_i) is the minimizer of some energy $E : C^k([0, 1]) \rightarrow \mathbb{R}$, subject to the interpolation constraints. For example, cubic splines are minimizers of the quadratic functional $F(f) = \int_0^1 \|f''(t)\|^2$, $f \in C^2$, subject to $f(t_i) = q_i$. Furthermore let $s_q : [0, 1] \rightarrow \mathbb{R}^d$ be the interpolating spline through the points q (for some fixed parameter values t_i) and $\hat{E}(q) = E(s_q)$. Then the gradient mapping of \hat{E} can be used as the operator C , since if the gradient is 0 at a point, the interpolation constraint of this point can be left out, yielding the same minimizer. Thus it is an inner point. Applying this construction to cubic splines, yields the same matrix C as constructed in Appendix 6.A. Another example for such a spline type are splines in tension [31, 32], which are minimizers of the quadratic functional $\int_0^1 (f''(t))^2 + \mu \int_0^1 (f'(t))^2$, $f \in C^2$, subject to $f(t_i) = q_i$. In Figure 6.3, we demonstrate the effect of varying μ and using our method to produce optimal approximating splines in tension. Wiggly splines [33–35] interpolating the nodes

(q_i, t_i) are minimizers of $\int_0^1 (f''(t) + \delta f'(t) + \xi f(t))^2$, $f \in C^2$, subject to $f(t_i) = q_i$, and thus there is a corresponding matrix C characterizing inner points for wiggly splines.

To construct the matrix C for these types of splines, one first needs to find the matrix A which maps the point coordinates to the coefficients of the interpolating spline. This matrix will depend on the t_i , the parameters (μ for splines in tension and δ, ξ for wiggly splines) and possible boundary conditions. Since the energy characterizing the splines is quadratic, it comes from a linear system of equations describing the relation between interpolation points and spline coefficients. Then we get $C = A^T B A$, where B maps the spline coefficients c to the value of the energy via $c^T B c$.

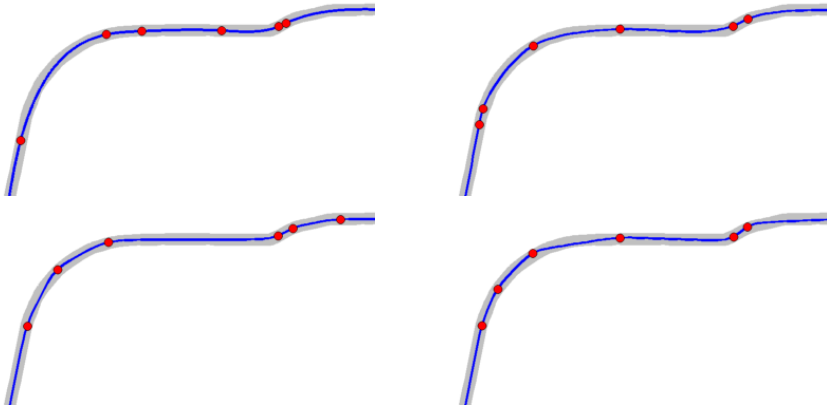


Figure 6.3: Testing our method on splines in tension. In reading order, we set $\mu = 0, 0.1, 0.5$ and 1 . $\lambda = 0.5 \cdot 10^{-9}$ was used in all cases.

6.4. NUMERICAL OPTIMIZATION OF ℓ_0 -REGULARIZED PROBLEMS

In recent years, many approximation algorithms for ℓ_0 -regularized problems have been developed. The classical ℓ_0 -regularized optimization problem is

$$q^{\text{opt}} = \underset{q}{\operatorname{argmin}} f(q) + \lambda \|q\|_{\ell_0} \quad (6.4)$$

where $f(q)$ is a convex function [1, 22, 23]. In our case, however, we want to find solutions where Cq is sparse, *i.e.*

$$q^{\text{opt}} = \underset{q}{\operatorname{argmin}} f(q) + \lambda \|Cq\|_{\ell_0} \quad (6.5)$$

where $f(q)$ is the squared ℓ_2 distance of q to the input points p , and C is a linear operator which might not be invertible—as, for example, is the case for the matrices resulting for cubic B-splines and CCBCs described in the previous sections—so algorithms which solve equation (6.4) cannot be applied directly.

The optimization problem (6.5) is also similar to the optimization problem (5.4) from Chapter 5, but important differences keep us from being able to apply the proposed algorithm ICCM to the problem of optimal splines. Here, we want to minimize an ℓ_0 regularized functional, as opposed to an L_1 regularized functional as in the case of localized

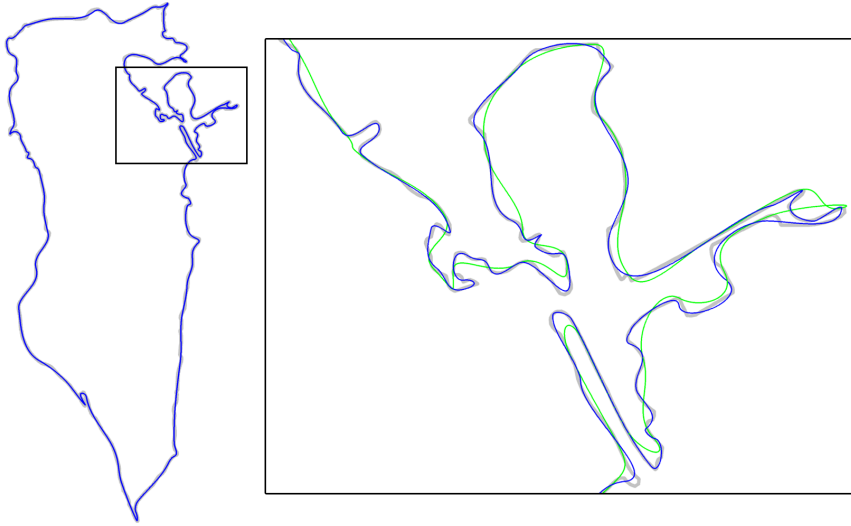


Figure 6.4: Comparison of a CCBC generated by our algorithm to a CCBC on equidistant nodes on the outline of Bahrain (2000 points).

6

vibration modes. Furthermore, ICCM was devised to optimize energy functionals in the presence of manifold constraints. In their absence, ICCM reduces to a single quadratic problem which can be efficiently solved directly. Approximating the ℓ_0 -term by an ℓ_1 -norm will be discussed in the experimental comparisons in Section 6.5.

Luckily, energy functionals of the form (6.5) have received much attention in the graphics and vision community, where they are used for sparse image recovery and smoothing [26–28] and mesh denoising [29].

We implemented and compared several algorithms, including

- a method proposed by Xu et al. [26] (ℓ_0 -gradient-minimization), where an auxiliary variable is introduced and minimization is alternated between two modified versions of (6.5);
- a majorize-minimize subspace algorithm with ℓ_2 - ℓ_0 regularization [27], where the ℓ_0 term is replaced by a differentiable approximation term, and the functional is then minimized using a subspace gradient-descent method with a majorize-minimize step size search;
- a slightly modified hard-thresholding pursuit algorithm inspired by [24, 25], where we performed a change of variables $\tilde{q} = Cq$, computed the gradient of $f(C^{-1}\tilde{q})$ by interpreting $C^{-1}\tilde{q}$ as a least-squares solution of $\tilde{q} = Cq$ subject to $\sum q_i = 1$, and then finding the optimal spline through the points with the largest gradient in these coordinates;
- a greedy matching pursuit algorithm [30], where we iteratively add a small group of points as nodes (by removing the corresponding conditions $(Cq)_i$ from a list of all

constraints), which was best (in terms of the ℓ_2 error) among a random selection of point groups;

- a modified random coordinate-descent method inspired by [1], which we will cover in more detail below;

For most experiments conducted in Section 6.5, the random coordinate-descent algorithm yielded the lowest values of the energy functional. The algorithms that do not directly control which entries of Cq are set to 0 (the first two listed) are problematic, since it is difficult to decide which points to use as nodes in the end.

Thus, our proposed solver uses a modified version of the method introduced by Patrascu and Necoara [1], who aim at minimizing (6.4) by a random coordinate-descent method. Our modified algorithm—aimed to minimize (6.5)—is summarized in Algorithm 4. We avoid a change of variables, since for our matrices C this results in an unstable algorithm. Instead we transform the minimization step in each iteration into solving a linearly constrained quadratic program.

Algorithm 4 Our modified random coordinate-descent method for ℓ_0 -regularized optimization

Input: Set of points p , sampling of the input curve.

Output: A minimizer p^{opt} of (6.5)

$L \leftarrow$ empty list of indices of vertices to be constrained

$\hat{p} \leftarrow p$

while not converged **do**

 Choose an index $k \in \{1, \dots, N\} \setminus L$

$\hat{L} \leftarrow L$

$L \leftarrow L \cup \{k\}$

 Solve $p^{\text{opt}} = \operatorname{argmin}_q \|q - p\|^2$ subject to $(Cq)_k = (0, \dots, 0) \in \mathbb{R}^d \forall k \in L$

if $\|p^{\text{opt}} - p\|^2 - \|\hat{p} - p\|^2 < \lambda$ **then**

$\hat{p} \leftarrow p^{\text{opt}}$

else

$L \leftarrow \hat{L}$

end if

end while

In our final implementation, we choose the random points from a shuffled list of all points and stop after all points have been visited once. (As an optional last step, one can revisit all still unconstrained points again and check if they can be constrained without increasing the energy by more than λ .) This means that we have N steps, and in each step a quadratic functional has to be minimized subject to at most $2Nd$ linear equality constraints. The method is straightforward to implement, and the constrained minimization requires only solving a linear system. The quadratic programs to be solved in each iteration are related since at most one equality constraint is added per iteration. Therefore, instead of solving the whole problem in every iteration, information from the previous iteration can be used to speed up the computation (e.g. updating of the matrix factorization of the previous iteration instead of computing a new factorization). Such

procedures are provided by various optimization libraries—we used MOSEK [36]. The average times for solving the quadratic programs in different scenarios are listed in Table 6.3. Minimizing the ℓ_0 -regularized energy via a random coordinate-descent can be regarded as a variant of traditional knot removal techniques. However, this relation to knot removal techniques is specific to the random coordinate-descent solver. In this sense, the ℓ_0 -formulation we introduce opens a door for using solvers like matching pursuit or ℓ_0 -gradient minimization for knot removal. This could be particularly helpful when removing knots from a spline with a large number of knots since the number of iterations required by the random coordinate-descent (as well as knot removal techniques) depends on the number of samples of the input curve (knots of the input spline). In this case, other solvers for the ℓ_0 problem (e.g. ℓ_0 -gradient-minimization), for which the required number of iterations does not directly depend on the number of sample points, are an alternative.

6.5. EXPERIMENTS AND COMPARISONS

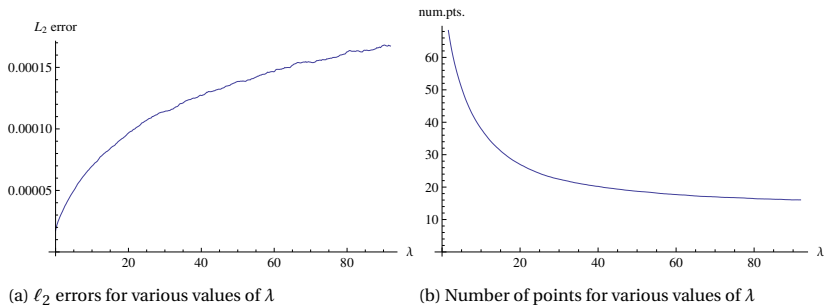


Figure 6.5: Results of using our method on a 150-point curve for $\lambda \in [0, 100 \cdot 10^{-9}]$, for cubic splines.

In our experiments, we scaled the curves to have unit arc length and sampled them equidistantly, densely enough to preserve the details of the input curve. All relevant values of the depicted experiments can be found in Table 6.1. Additionally, in Figure 6.5, we plot the ℓ_2 error and the resulting number of points of the result of our method (using cubic splines) against various values of λ . From this, it can be seen that λ determines the degree of detail in the approximating curve as expected: the higher we set the cost per segment, the fewer points we get, at the cost of higher ℓ_2 errors.

Figures 6.1 and 6.6 show how our method smartly places points in order to reduce the ℓ_2 error. Figure 6.1 shows the effect of decreasing λ : as the cost per segment becomes lower, more segments are placed in favor of improving the approximation of the input curve. We found that $\lambda = 10^{-9}$ gave a good balance between the number of points and accuracy of the approximation for curves of unit arc length, but of course the value has to be adjusted for specific tasks and depending on the demands of applications. While it is possible to adjust the level of detail via controlling the desired number of nodes, we found that it is more intuitive to set a cost per segment, since this is a measure that transports over all curves of the same total length. Figure 6.6 compares cubic splines to CCBCs when approximating a curve using our method with the same value of λ . Note

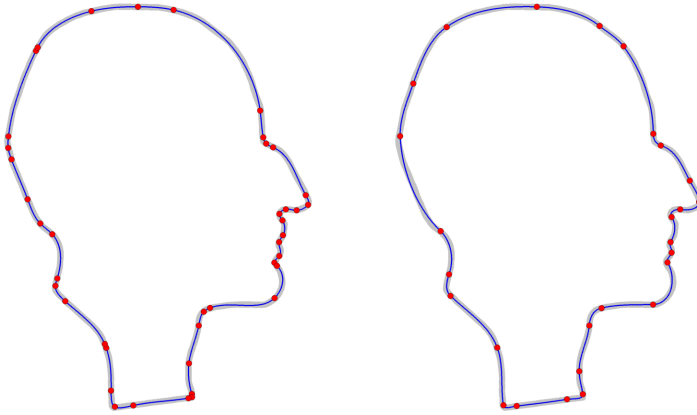


Figure 6.6: Comparison of our approximation method for cubic B-splines (left, 42 points) and CCBCs (right, 27 points) on the silhouette of a bust of Max Planck, 300 points, $\lambda = 4 \cdot 10^{-9}$ for both spline types.

that CCBCs need considerably fewer nodes to achieve the same level of detail. However, at each node we specify the position as well as the first derivative of the spline, instead of just the position.

Name (Fig.)	Type	O	R	$\lambda \cdot 10^9$	ℓ_2 -error	ℓ_∞ -error
Beetle (6.1a)	CCBC	500	9	50	$3.80 \cdot 10^{-4}$	0.00288
Beetle (6.1b)	CCBC	500	17	1.05	$7.82 \cdot 10^{-6}$	0.00289
Max-Planck (6.6, l)	Cubic	300	42	4	$3.87 \cdot 10^{-5}$	0.00318
Max-Planck (6.6, r)	CCBC	300	27	4	$2.82 \cdot 10^{-5}$	0.00383
Silhouette (6.10, m)	Cubic*	500	49	18	$6.75 \cdot 10^{-5}$	0.00510
Silhouette (6.10, r)	Cubic	500	49	5	$4.22 \cdot 10^{-5}$	0.00758
Noisy Max-Planck (6.7, l)	CCBC	300	18	60	$2.11 \cdot 10^{-4}$	0.00759
Noisy Max-Planck (6.7, r)	Cubic	300	26	60	$2.29 \cdot 10^{-4}$	0.00759
Hiragana “wo” (6.8, m)	CCBC**	500	29	1.5	$8.34 \cdot 10^{-5}$	0.00448
Hiragana “wo” (6.8, r)	CCBC	500	39	4	$3.50 \cdot 10^{-5}$	0.00446
Rocker-Arm (6.9, m)	CCBC**	500	26	2	$6.11 \cdot 10^{-5}$	0.00151
Rocker-Arm (6.9, r)	CCBC	500	33	2	$2.04 \cdot 10^{-5}$	0.00237
3D Feature-lines (6.11)	CCBC	2033	154	1000	$6.24 \cdot 10^{-6}$	0.00251

$O = \#$ pts of input curve, $R = \#$ knots of spline, * = weighted ℓ_2 , ** = w/ disc. in 1st deriv.

Table 6.1: Data for the experiments.

Our method is robust to noise: one can adjust λ such that it captures all desired details but not the noise, and the interpolated points will be chosen and arranged such that a smooth curve with minimal ℓ_2 distance is achieved. This is demonstrated in Figure 6.7 for both CCBCs and cubic B-splines: no noise is visible in the results while most of the curve’s characteristics are still intact.

Figures 6.8 and 6.9 show the flexibility of our method: by simply modifying the con-

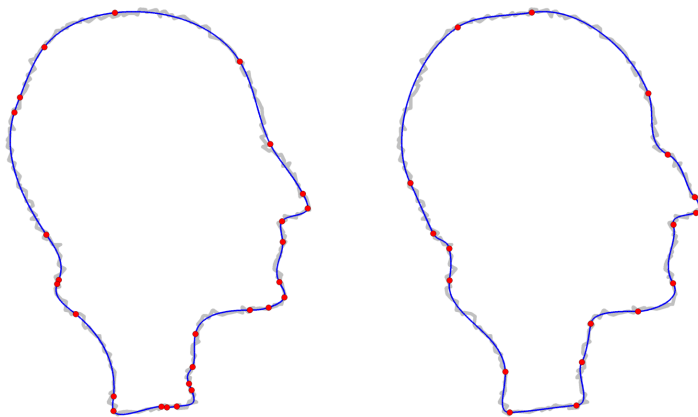


Figure 6.7: The silhouette of a bust of Max Planck with extreme noise (300 points) approximated by a cubic B-spline (left) and a CCBC (right) using our method with $\lambda = 60 \cdot 10^{-9}$.

6

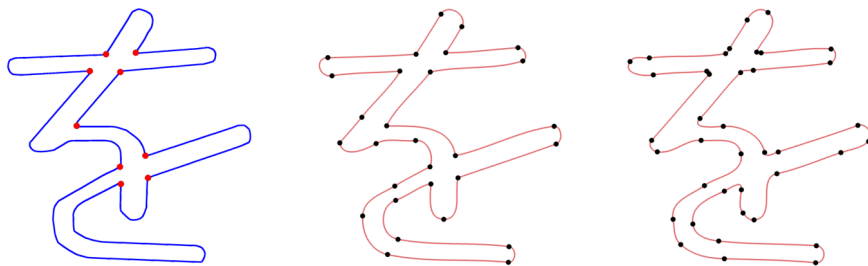


Figure 6.8: The Hiragana “wo” with 500 points and 9 points marked as discontinuities in the first derivative (left). Composite cubic Bézier curves received from our algorithm with (middle, 29 points) and without (right, 39 points) these discontinuities. Both approximations have similar ℓ_∞ error.

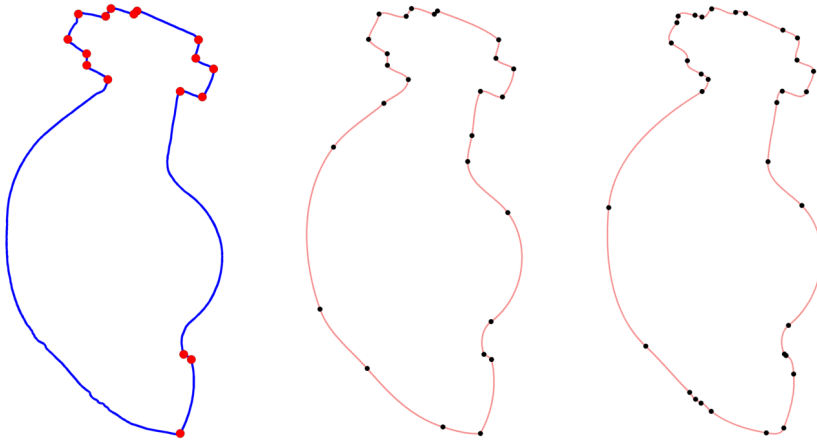


Figure 6.9: A cross section of the rocker-arm model with 500 points and 17 points marked as discontinuities in the first derivative (left). Optimized composite cubic Bézier curves with (middle, 26 points) and without (right, 33 points) these discontinuities, for the same value of λ .

straints of certain points (which means changes to the operator C), we are able to specify points at which we allow the CCBC to have discontinuities in the first derivative, which is very useful if sharp edges need to be modeled, as in these examples. The results are compared to the case where no discontinuities in the first derivative are allowed. In Figure 6.10, we used a weighted ℓ_2 norm: some points were assigned a higher weight such that in the resulting cubic spline we captured the details in these parts very closely, while saving points by approximating the rest only roughly.

In Figure 6.11, we test our algorithm on a large set of three-dimensional densely sampled curves which represent smooth feature lines on a scanned mesh. This example also shows the advantage of the l_0 -regularized minimization: instead of having to choose a fixed number of points for every curve on the mesh, we fix the parameter λ once and obtain a comparable level of detail across all resulting splines.

As another application, we applied our algorithm to motion-capturing data. The data describes the motion of a human performing capoeira over 400 frames (13.3 seconds at 30 frames per second). It contains the position of the midpoint and 3-dimensional angles of 28 joints between various parts of the body, resulting in 89D data (cf. Figure 6.12). First, we approximated the data using one CCBC for the position and one for each angle, resulting in 29 3-dimensional splines. We chose λ such that the approximating splines had 25 nodes on average, which led to an approximation error of $6.51 \cdot 10^{-4}$ (the data was normalized to have unit arc length as well). Visually, the animation generated by the approximating spline is almost indistinguishable from the input animation. Additionally, we used our algorithm to smooth a part of the animation in which the human was visibly trembling: after approximating 100 frames through one 89-dimensional CCBC with 20 nodes, the primary motion of the human was still completely intact while all trembling was gone.

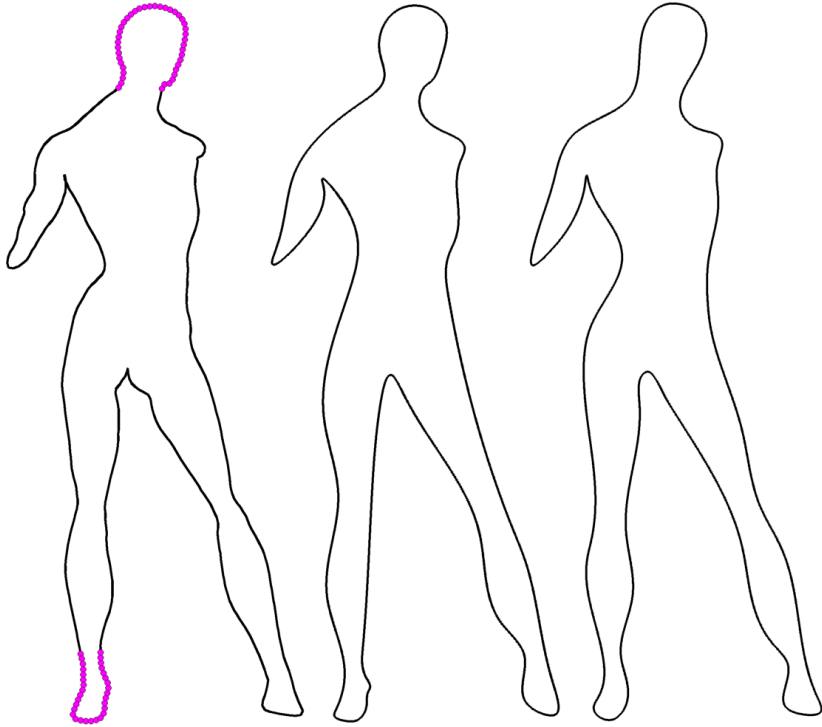


Figure 6.10: Results of our method with weighted ℓ_2 norm. Left: input with marked points in purple, middle: cubic spline through 49 points using weighted ℓ_2 (weight 100 at the marked vertices, 1 elsewhere), right: cubic spline from 49 points using standard ℓ_2 .

Experimental comparisons. We compare the performance of different solvers for the ℓ_0 -regularized optimization problem (6.2). In addition, we compare the results of the ℓ_0 -minimization to results produced by alternative (recent and traditional) techniques for knot optimization. For a set of input curves, we produce cubic splines with the same number of knots using all the methods and measure the resulting ℓ_2 -errors. The results are listed in Table 6.2. In addition to Algorithm 1, we used the following methods:

- A matching pursuit algorithm, where instead of iteratively removing knots, we build the set of unconstrained nodes by greedily unconstraining the best (in terms of lowest resulting energy functional value) cardinality k set, among m randomly chosen cardinality k sets. For the comparisons we used $m = 500, k = 3$ throughout. Note that evaluating the candidate sets in each iteration takes as long as m full iterations in Algorithm 4, which is why this method is only feasible for a small number of desired knots.
- We minimize energy (6.2) using the aforementioned ℓ_0 gradient minimization method proposed by Xu et al. [26]. The direct results of this method often yield large clusters of nodes, which we address by using a cleanup step after the optimization: we

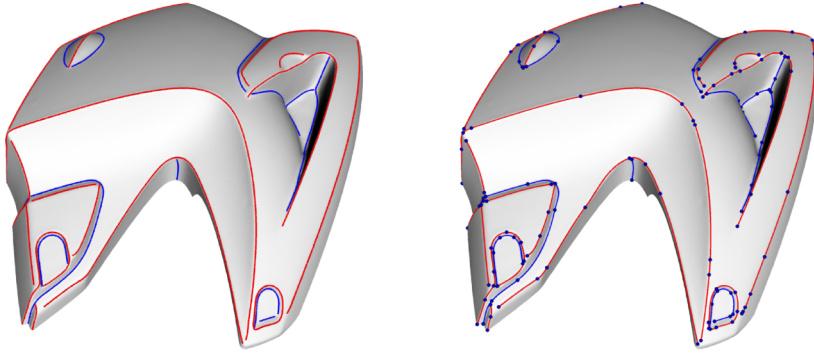


Figure 6.11: Feature lines on a surface, left: input feature lines, right: optimized CCBCs with 154 vertices.

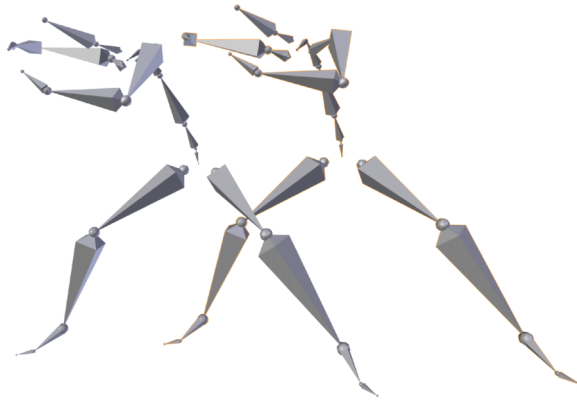


Figure 6.12: A snapshot from the animation which we approximated using CCBCs through few data points (left: input motion, right: approximated motion).

check, for each remaining unconstrained point, whether removing the knot improves the energy. In many cases, the resulting optimizer performs as well as (or sometimes better than) Alg. 4. However, it depends on various parameters that need to be adjusted for each curve.

- When replacing the ℓ_0 -term in energy (6.2) by an ℓ_1 term (as proposed in [17] for the 1-dim. splines), the resulting convex energy can be minimized directly by solving a quadratic program. However, in this formulation, λ can no longer be interpreted as the cost per segment, but it needs to be experimentally adjusted to achieve the desired level of detail. Furthermore, large values in Cp get punished, which is not desired and leads to over-smoothing of the resulting curve. Also, there is no clear way to couple the constraints for the coordinates of the same node, which typically leads to clusters of unconstrained points. These problems can cause bad performance, an example is the large error for the approximation of the Bahrain-curve.

Input curve	#pts	#knots	ℓ_0 optimizer				Other techniques		
			Algorithm 4	Match.Purs.	ℓ_0 -gradient-min.	ℓ_1 -reg.	PSO [15]	LSPIA [2]	MATLAB OPTKNT
Max Planck (Fig. 6.6)	300	48	2.7587 · 10 ⁻⁵	3.6802 · 10 ⁻⁵	2.8661 · 10 ⁻⁵	5.4789 · 10 ⁻⁵	4.4474 · 10 ⁻⁵	3.6980 · 10 ⁻⁵	6.6168 · 10 ⁻⁵
Cross Section (Fig. 6.13)	500	48	1.8336 · 10 ⁻⁵	3.3407 · 10 ⁻⁵	1.3729 · 10 ⁻⁵	4.9466 · 10 ⁻⁵	4.2251 · 10 ⁻⁵	3.6252 · 10 ⁻⁵	6.1692 · 10 ⁻⁵
Silhouette (Fig. 6.10)	500	97	1.2066 · 10 ⁻⁵	1.3967 · 10 ⁻⁵	1.0834 · 10 ⁻⁵	1.2337 · 10 ⁻⁵	1.9649 · 10 ⁻⁵	1.2797 · 10 ⁻⁵	2.5114 · 10 ⁻⁵
Bahrain (Fig. 6.4)	2000	137	1.2070 · 10 ⁻⁵	1.6341 · 10 ⁻⁵	1.5947 · 10 ⁻⁵	3.0374 · 10 ⁻⁵	1.5633 · 10 ⁻⁵	1.2395 · 10 ⁻⁵	1.7437 · 10 ⁻⁵
Hiragana (Fig. 6.8)	500	52	5.6206 · 10 ⁻⁵	5.3876 · 10 ⁻⁵	6.9553 · 10 ⁻⁵	8.3407 · 10 ⁻⁵	5.7682 · 10 ⁻⁵	6.1827 · 10 ⁻⁵	7.2012 · 10 ⁻⁵

Table 6.2: Data for comparisons of different ℓ_0 -minimizers and other spline optimization techniques. For a set of input curves, the ℓ_2 -errors to the resulting cubic splines are listed. In every row, all splines have the same number of knots.

	CCBC	Cubic	Cubic w/ lumped mass matrix
250 pts.	3.2	10.4	2.0
500 pts.	7.4	33.6	4.6
1000 pts.	20.1	79.4	11.9

Table 6.3: Average times (in milliseconds) for solving the quadratic program in each iteration of Algorithm 4

- For the case of B-splines, our ℓ_0 problem is addressing the free-knot optimization problem, so it is natural to compare to recent techniques which aim at directly minimizing the ℓ_2 -error over a fixed number of nodes. Galvez et al. [15] report that this results in a difficult multimodal and multivariate nonlinear optimization problem, and they try to find an optimal knot vector by using particle-swarm optimization (PSO). We apply their algorithm using the proposed parameter values, namely 100 particles, 10 iterations, $\gamma_{1,2} = 2$, $w = 0.9 \dots 0.4$. While the PSO performs well if the number of desired knots is small, it becomes infeasible for a larger number of knots.
- The refinement algorithm proposed in [2] that iteratively adds knots to an approximating cubic B-spline (starting from a cubic B-spline defined on a sparse knot vector) by placing them on the segment of the spline on which the ℓ_2 error is largest, specifically at the parameter value, where the accumulated ℓ_2 error reaches half of the total error on this segment. While performing comparable to our algorithm on curves with fairly constant level of detail, it often produces knots at unnecessary places (see Figure 6.13) for other types of curves.
- As a comparison to a traditional technique, we consult the MATLAB function OPTKNT which computes an “optimal” knot sequence “in the sense of Micchelli/Rivlin/Winograd and Gaffney/Powell” [37].

For all tested examples, one of the ℓ_0 -optimizers yielded the best result, which justifies our reformulation of the optimal spline problem.

6.6. CONCLUSION

We introduce a new approach for computing optimal approximating splines, where spline coefficients, positions of nodes, and the number of spline segments are variable. The approach can be used to optimize different types of splines including B-splines and com-

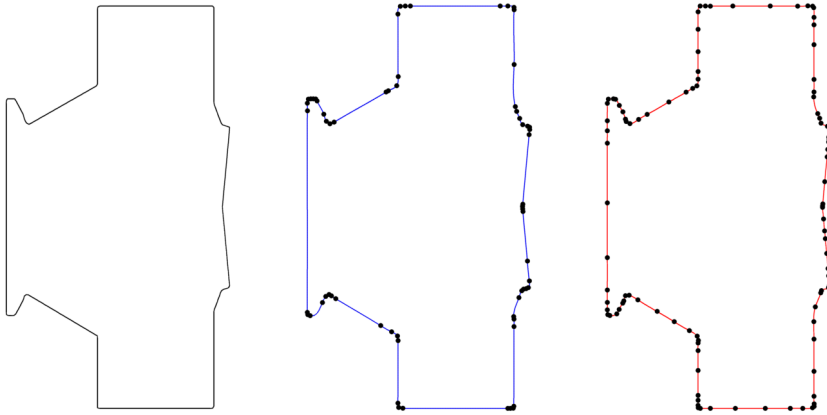


Figure 6.13: A curve approximated by a cubic spline using our algorithm and using a refinement technique from [2], such that both results produce similar ℓ_2 error to the input curve. From left to right: input curve, cubic B-spline from our algorithm with 65 nodes, cubic B-spline from the refinement method in [2] with 102 nodes.

posite Bézier curves. Our modeling of the optimization problem yields a ℓ_0 -regularized quadratic problem for which we devise a solver based on the recent scheme of Patrascu and Necoara [1]. We present results produced with our implementation for the approximation of planar and space curves, and spline conversion of motion capture data.

6

Limitations and challenges. One direction of future work is to find a convex approximation of problem (6.2), *e.g.* using weighted ℓ_1 -norms. This could potentially lead to a very fast tool for optimal spline approximation. A limitation of our current approach is that for B-splines, it cannot deal with multiple knots. One idea of how to integrate multiple knots is to allow for lower regularity than C^2 (for the case of cubic splines) at nodes, but to penalize lower continuity. We leave this problem as future work. Our tool could be used to convert curves (*e.g.* hand-drawn outlines of shapes) into CCBCs (or other splines types), which can be used for curved editing. It would be interesting to experiment with other norms for the approximation of the input curve. The ultimate goal would be to design a norm such that our tool places the nodes in locations where an artists would place them. Another direction for future work is to extend the approach from curves to surfaces.

REFERENCES

- [1] A. Patrascu and I. Necoara, *Random coordinate descent methods for ℓ_0 regularized convex optimization*, IEEE Transactions on Automatic Control **60**, 1811 (2015).
- [2] C. Deng and H. Lin, *Progressive and iterative approximation for least squares B-spline curve and surface fitting*, Computer-Aided Design **47**, 32 (2014).

- [3] C. de Boor, *Good approximation by splines with variable knots*, in *Spline Functions and Approximation Theory* (Springer, 1973) pp. 57–72.
- [4] H. G. Burchard, *Splines (with optimal knots) are better*, *Applicable Analysis* **3**, 309 (1974).
- [5] D. L. B. Jupp, *Approximation to data by splines with free knots*, *SIAM J. Numer. Anal.* **15**, 328 (1978).
- [6] W. Li, S. Xu, G. Zhao, and L. P. Goh, *Adaptive knot placement in B-spline curve approximation*, *Computer-Aided Design* **37**, 791 (2005).
- [7] H. Yang, W. Wang, and J. Sun, *Control point adjustment for B-spline curve approximation*, *Computer Aided Design* **36**, 639 (2004).
- [8] T. Lyche and K. Mørken, *A data-reduction strategy for splines with applications to the approximation of functions and data*, *IMA J. Numer. Anal.* **8**, 185 (1988).
- [9] T. I. Vassilev, *Fair interpolation and approximation of B-splines by energy minimization and points insertion*, *Computer-Aided Design* **28**, 753 (1996).
- [10] T. Lyche and K. Mørken, *Knot removal for parametric B-spline curves and surfaces*, *Comput. Aided Geom. Des.* **4**, 217 (1987).
- [11] X. Zhao, C. Zhang, B. Yang, and P. Li, *Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation*, *Computer-Aided Design* **43**, 598 (2011).
- [12] M. Sarfraz and S. A. Raza, *Capturing outline of fonts using genetic algorithm and splines*, in *Proc. Intern. Conf. on Information Visualisation* (2001) pp. 738–743.
- [13] M. Moriyama, F. Yoshimoto, and T. Harada, *Automatic knot placement by a genetic algorithm for data fitting with a spline*, in *Proc. Shape Model. Intern.* (1999) pp. 162–169.
- [14] F. Yoshimoto, T. Harada, and Y. Yoshimoto, *Data fitting with a spline using a real-coded genetic algorithm*, *Computer-Aided Design* **35**, 751 (2003).
- [15] A. Gálvez and A. Iglesias, *Efficient particle swarm optimization approach for data fitting with free knot B-splines*, *Computer-Aided Design* **43**, 1683 (2011).
- [16] G. Maier, *Optimal arc spline approximation*, *Computer Aided Geometric Design* **31**, 211 (2014).
- [17] H. Kang, F. Chen, Y. Li, J. Deng, and Z. Yang, *Knot calculation for spline fitting via sparse optimization*, *Computer-Aided Design* **58**, 179 (2015).
- [18] J.-C. Perez and E. Vidal, *Optimum polygonal approximation of digitized curves*, *Pattern Recognition Letters* **15**, 743 (1994).

- [19] A. Gribov and E. Bodansky, *A new method of polyline approximation*, in *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, Vol. 3138 (Springer, 2004) pp. 504–511.
- [20] A. Kolesnikov, *Segmentation and multi-model approximation of digital curves*, *Pattern Recognition Letters* **33**, 1171 (2012).
- [21] R. Mann, A. Jepson, and T. El-Maraghi, *Trajectory segmentation using dynamic programming*, in *16th Intern. Conf. on Pattern Recognition*, Vol. 1 (2002) pp. 331–334.
- [22] S. Bahmani, P. Boufounos, and B. Raj, *Greedy sparsity-constrained optimization*, in *Forty Fifth Asilomar Conference on Signals, Systems and Computers* (2011) pp. 1148–1152.
- [23] A. Beck and Y. C. Eldar, *Sparsity constrained nonlinear optimization: Optimality conditions and algorithms*, *SIAM J. Optim.* **23**, 1480 (2013).
- [24] T. Blumensath and M. E. Davies, *Normalized iterative hard thresholding: Guaranteed stability and performance*, *IEEE J. Sel. Topics Signal Process.* **4**, 298 (2010).
- [25] S. Foucart, *Hard thresholding pursuit: An algorithm for compressive sensing*, *SIAM J. Numer. Anal.* **49**, 2543 (2011).
- [26] L. Xu, C. Lu, Y. Xu, and J. Jia, *Image smoothing via L_0 gradient minimization*, *ACM Trans. Graph.* **30**, 174:1 (2011).
- [27] E. Chouzenoux, A. Jezierska, J.-C. Pesquet, and H. Talbot, *A majorize-minimize subspace approach for $\ell_2 - \ell_0$ image regularization*, *SIAM Journal on Imaging Science* **6**, 563 (2013).
- [28] M. Nikolova, M. K. Ng, S. Zhang, and W.-K. Ching, *Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization*. *SIAM J. Imaging Sciences* **1**, 2 (2008).
- [29] L. He and S. Schaefer, *Mesh denoising via L_0 minimization*, *ACM Trans. Graph.* **32**, 64:1 (2013).
- [30] F. Bergeaud and S. Mallat, *Matching pursuit of images*, *Wavelet Analysis and Its Applications* **7**, 285 (1998).
- [31] D. Schweikert, *An interpolating curve using a spline in tension*, *J. Math. Phys.* **45**, 312 (1966).
- [32] M. Hofer and H. Pottmann, *Energy-minimizing splines in manifolds*, *ACM Trans. Graph.* **23**, 284 (2004).
- [33] M. Kass and J. Anderson, *Animating oscillatory motion with overlap: wiggly splines*, *ACM Trans. Graph.* **27**, 28:1 (2008).
- [34] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier, *Interactive spacetime control of deformable objects*, *ACM Trans. Graph.* **31**, 71:1 (2012).

- [35] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, *Animating deformable objects using sparse spacetime constraints*, ACM Transactions on Graphics (TOG) **33**, 109:1 (2014).
- [36] E. D. Andersen and K. D. Andersen, *The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm*, in *High performance optimization* (Springer, 2000) pp. 197–232.
- [37] P. Gaffney and M. Powell, *Optimal interpolation*, in *Numerical Analysis*, Lecture Notes in Mathematics, Vol. 506, edited by G. Watson (Springer Berlin Heidelberg, 1976) pp. 90–99.

APPENDIX

6.A. EXPLICIT CONSTRAINT OPERATORS

Cubic splines. We will now construct the operator C for cubic splines, which will have the property that $(C(q))_j = 0$ if the point q_j is an inner point. To this end, we need the operator A which maps a set of points to some representation of a cubic spline. We represent the cubic spline by its second derivatives q''_j at the points q_j (which determine the spline uniquely), since in this representation all involved operators have simple, precomputable forms. Therefore we define the stiffness matrix S as the circulant matrix having $\frac{h}{6}(2, -1, 0, \dots, 0, -1) \in \mathbb{R}^N$ as its first row and the mass matrix M as the circulant matrix having $\frac{1}{h}(4, 1, 0, \dots, 0, 1) \in \mathbb{R}^N$ as its first row. Then, determining the second derivatives $q'' = (q''_1, \dots, q''_N)$ at the points can be done via solving the system of linear equations $Mq'' = -Sq$. This system assumes that the t_i are equidistant with step size h , i.e. $t_i - t_{i-1} = h$ for $i = 2, \dots, N$, and that the curve is closed, i.e. we want periodic boundary conditions (other boundary conditions can be realized by changing the first and last rows of the matrices M and S). The matrix $A = \frac{6}{h^2} M^{-1} S$ will also be circulant and its entries can be precomputed up to arbitrary precision. Now the cubic spline for $t \in [t_j, t_{j+1})$ is given by

$$s(t) = q_j s_1(t) + q_{j+1} s_2(t) + \frac{h^2}{6} (q''_j (s_1^3(t) - s_1(t)) + q''_{j+1} (s_2^3(t) - s_2(t)))$$

where $s_1(t) = \frac{t_{j+1} - t}{h}$ and $s_2(t) = \frac{t - t_j}{h}$

So for $t \in [t_j, t_{j+1})$ the third derivative of the resulting cubic spline is $s'''(t) = \frac{1}{h} (-q''_j + q''_{j+1})$ and as the condition for the left and right derivatives at $t = t_j$ to agree we get $2q''_j - q''_{j-1} - q''_{j+1} = 0$. Thus, letting $C = SM^{-1}S$, we get the desired linear operator, which has the property that q_j is an inner point iff $(Cp)_j = 0$.

CCBCs. In order to define the operator C which characterizes inner points for CCBCs, we are going to represent CCBCs by their coefficients in the Bernstein basis $(B_k^3)_{k=0, \dots, 3}$, since this again leads to simple forms of the involved operators. It is $B_k^3(t) = \binom{3}{k} t^k (1-t)^{3-k}$ and in case of equidistant sampling with step size h , the CCBC takes the piecewise defined form $s(t) = s_j(t) = \sum_{k=0}^3 b_k^j B_k^3\left(\frac{t-t_j}{h}\right)$ for $t \in [t_j, t_{j+1})$. The coefficients are directly given via the set of points and the prescribed first derivatives (we again assume equidistant step sizes h):

$$b_0^j = q_j, \quad b_1^j = q_j + \frac{h}{3} q'_j, \quad b_2^j = q_{j+1} - \frac{h}{3} q'_{j+1}, \quad b_3^j = q_{j+1}$$

and the second and third derivatives of $s_j(t)$ for $t = t_j$ and $t = t_{j+1}$ are given by

$$s_j''(t_j) = \frac{6}{h^2}(b_2^j - 2b_1^j + b_0^j), \quad s_j''(t_{j+1}) = \frac{6}{h^2}(b_3^j - 2b_2^j + b_1^j)$$

$$s_j'''(t_j) = s_j'''(t_{j+1}) = \frac{6}{h^3}(b_3^j - 3b_2^j + 3b_1^j - b_0^j)$$

An inner point q_j has to satisfy the conditions $s_j''(t_j) = s_{j-1}''(t_j)$ and $s_j'''(t_j) = s_{j-1}'''(t_j)$. From this, for $n = 1$, we define C as the $2N \times 2N$ matrix

$$\begin{pmatrix} 0 & -\frac{4h}{3} & 1 & -\frac{h}{3} & 0 & \cdots & \cdots & 0 & -1 & -\frac{h}{3} \\ 4 & 0 & -2 & h & 0 & \cdots & \cdots & 0 & -4 & -h \\ -1 & -\frac{h}{3} & 0 & -\frac{4h}{3} & 1 & -\frac{h}{3} & 0 & \cdots & \cdots & 0 \\ -2 & -h & 4 & 0 & -2 & h & 0 & \cdots & \cdots & 0 \\ & & & & & \ddots & & & & \\ 0 & \cdots & & & \cdots & 0 & -1 & -\frac{h}{3} & -\frac{4}{3} & 0 & 1 & -\frac{h}{3} \\ 0 & \cdots & & & \cdots & 0 & -1 & -h & 2 & 0 & -1 & h \\ 1 & -\frac{h}{3} & 0 & \cdots & \cdots & 0 & 0 & 0 & -1 & -\frac{h}{3} & -\frac{4h}{3} & 0 \\ -2 & h & 0 & \cdots & \cdots & 0 & 0 & 0 & -2 & -h & 4 & 0 \end{pmatrix}$$

This matrix contains two rows for each point, which expresses the condition that the second and third derivatives are continuous at this point. With this, q_j is an inner point iff $(C(q, q'))_{2j-2} = 0$ and $(C(q, q'))_{2j-1} = 0$, where (q, q') is the vector $(q_1, q'_1, q_2, q'_2, \dots, q_N, q'_N)$ (or, if $d > 1$, a matrix where each column is a vector).

7

CONCLUSION

We presented contributions to the domain of computer graphics in a variety of problems and applications. A central focus was the usage of model reduction approaches to enhance and accelerate numerical algorithms for optimization, simulation and modeling tasks. A goal of such approaches is to enable interactivity of computer graphics applications and tools, independently of the size, complexity or resolution of the input. Many other contributions were made, which are summarized in the conclusions of each individual chapter. Here, we would like to briefly summarize the general ideas and offer some thoughts on possible directions for future work that researches the interplay of computer graphics and model reduction approaches, where we consider a very general interpretation of *model reduction*.

Firstly, we would like to remark that virtually any problem in computer graphics can benefit from reduction approaches. The nature of input data in this field is always discrete: even for analytic representations, such as spline curves and surfaces, a set of control points is required. The amount of data grows with the complexity and level of detail that is desired within the application. Without reduction approaches, this will always limit the level of detail that can be processed interactively, or at least in feasible computation times. Model reduction approaches shift this limitation to a trade-off between accuracy and speed. For example, we showed how to simulate meshes of arbitrary resolution in real time, by approximating the dynamics and thus sacrificing details of the motion of finer details within the mesh. Unless this detail can be observed (*e.g.* by watching the simulation in slow motion or by placing the camera very close to such details) the trade-off is invisible. With this in mind, we hope to see many more areas in computer graphics to employ the benefits of reduction approaches. For example, multi-resolution hierarchies and upsampling approaches are first steps of such techniques in the area of shading and rendering, but a more methodical approach might greatly enhance accuracy and performance here.

The trade-off between approximation accuracy and computation speed gained from

model reduction approaches can be steered. Which details should be preserved and what can be sacrificed is determined by the specific dimensionality reduction and approximation schemes. For example, the tangent fields we model in Chapter 1 are restricted to the lowest eigenfields of the Hodge–Laplace operator. Hence, we only preserve low frequency details of the fields. In this case, this even regularizes the field, as smoothness is the main indicator of quality in this application. If specific, classifiable, high frequency detail is required, it can be added in the form of specific (possibly localized) basis vectors, or artificially added in post processing steps.

In general, the question where to preserve detail and how much data should be added in order to preserve it is complex. It requires us to distinguish desired features from unnecessary (or unwanted) features and noise. This problem is explicitly addressed in Chapter 6, where the user can specify the cost of an additional control point in a spline in terms of errors in the approximation of the original curve. As discussed there, this turns out to be a NP-hard optimization problem, which characterizes the nature of the question.

A different type of approach tries to learn the distinction between noise and desired detail by making use of additional information, provided in the pre-process. For example, if a collection of exemplary deformations of the same shape is available, reducing the dynamics of the shape to a subspace created from a POD of these examples has been shown to achieve plausible dynamics for low-dimensional subspaces. However, such sets of exemplary deformations are rarely provided along with the shape and creating them is a time-consuming and complex endeavor.

In absence of such structured data, one can resort to more sophisticated machine learning approaches that make use unstructured collections of data (as can be found on the internet, where huge databases of shapes are available). Such approaches can try to learn the classification of important and unimportant dimensions from the data by making use of user-provided labels. This has been successfully demonstrated, *e.g.* for image compression. One obstacle to carry such methodology over to the field of geometry processing is the complex nature of the data describing geometry. For example, it is unclear how to compare a collection of meshes found "in the wild", *i.e.* raw triangle meshes, as their descriptions are not compatible. In contrast to images, there is no canonical ordering of the descriptors (vertices and triangles) of the data, such that feature descriptions cannot be easily matched across several different geometries.

To further expand on the direction of machine learning, let us recall the second type of approximation that appeared in many of the presented model reduction approaches of this thesis. That is, the evaluation of elastic forces or other non-linear functionals that appeared in simulation and modeling tasks. Here we can produce data by simply computing pairs of input and output of such functionals. This data can be used in various ways, for example one could learn the relation between partial deformation data and the output of the functional that computes the elastic forces corresponding to the deformation. This would yield a reduced approximation for elastic forces. Note that the input/output pairs correspond to a single geometry and thus we would learn to evaluate

the functional on that specific shape. The cubature we used in the reduction approach in Chapter 4 could be classified as one such learning method. It would be interesting to explore, whether employing CNNs to learn these functionals has a positive impact on performance and approximation quality.

Aside from machine learning, other examples for fields whose current development will be beneficial for the construction of novel or enhanced model reduction approaches are signal processing, in particular compressed sensing (by obtaining better approximations for the optimal choice of data points to keep), stochastics and statistics (by offering better heuristic methods for the approximation of non-linear functionals) and dynamical systems (by offering more insight on how to distinguish and classify important and unimportant dimensions and terms for the long term behavior and general characteristics of solutions).

More computation power gained by ever-growing technological development enables the processing of larger amounts of data at interactive rates. Still, model reduction approaches remain fundamental to the field of computer graphics. For example, with the advent of virtual reality on a consumer level, it becomes more important than ever to enable real-time framerates with an immense level of detail. Model reduction approaches can vastly benefit such applications.

We hope that these impulses, and the ideas provided within this thesis will help to demonstrate the benefits of, and nourish the interest in model reduction approaches in computer graphics.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisor Dr. Klaus Hildebrandt, whose guidance, clear focus and scientific mind led me through my PhD studies with great success. He was always eager to discuss both details and the broader picture of our research projects and constantly encouraged me to bring my work to excellency. Prof. Elmar Eisemann and Dr. Klaus Hildebrandt helped me in making important career and life decisions and were supportive far beyond expectation. I consider myself lucky to have been guided by such friendly, knowledgeable and experienced people.

I would like to thank my other co-authors Christoph von Tycowicz, Leonardo Scandolo and Ahmad Nasikun, whose valuable input helped to shape the publications that make up this thesis. Moreover, I would like to thank all anonymous reviewers, whose helpful comments helped to refine and enhance the publications within this thesis.

My deep gratitude also goes to Jean-Marc Thiery, Christian Schulz, Pablo Bauszat, Niels de Hon, Jing-Tang Liao, Changgong Zhang and many other friends and colleagues who guided my research through deep and interesting discussions and remarks.

To my family and Lara for their unconditional support – it means everything to me.

CURRICULUM VITÆ

Christopher BRANDT

EDUCATION

- 1992–1996 Schiller-Schule Wiesloch, Germany
1996–2006 Englisches Institut Heidelberg, Germany
2007–2008 Dual Studies in Business Informatics
In cooperation with SAP AG
2008–2011 B.Sc. in Mathematics
FU Berlin, Germany
Thesis: Kaskaden Periodenverdoppelder Bifurkationen.
Promotor: Prof. dr. B. Fiedler
2011–2014 M.Sc. in Mathematics
FU Berlin, Germany
Thesis: The Restricted Satisfiability Problem (k, s)-SAT and
Maker-Breaker Games on Hypergraphs.
Promotor: Prof. dr. T. Szabó
2014–2015 Ph.D. Studies in Geometry Processing
Max-Planck-Institut für Informatik, Germany
2015–2019 Ph.D. Studies in Geometry Processing
Delft University of Technology, Netherlands

WORKING EXPERIENCE

- 2006–2007 Red Cross Heidelberg, Germany
2007–2008 Training at SAP AG
As part of Dual Studies in Business Informatics
2011–2012 Research student at the Potsdam Institute for Climate Impact Research, Germany
2012–2013 Research student in the Geometry Processing Group of Prof. K. Polthier

LIST OF PUBLICATIONS

6. **C. Brandt, Elmar Eisemann, Klaus Hildebrandt**, *Hyper-Reduced Projective Dynamics*, ACM Transactions on Graphics **37(4)**, Article 80 (2018).
5. **C. Brandt, Leonardo Scandolo, Elmar Eisemann, Klaus Hildebrandt**, *Modeling n -Symmetry Vector Fields using Higher-Order Energies*, ACM Transactions on Graphics **37(2)**, Article 18 (2018).
4. **C. Brandt, Klaus Hildebrandt**, *Compressed Vibration Modes for Elastic Bodies*, Computer Aided Geometric Design **52-53**, 297–312 (2017).
3. **C. Brandt, Leonardo Scandolo, Elmar Eisemann, Klaus Hildebrandt**, *Spectral Processing of Tangential Vector Fields*, Computer Graphics Forum **36(6)**, 338–353 (2017).
2. **C. Brandt, Christoph von Tycowicz, Klaus Hildebrandt**, *Geometric Flows of Curves in Shape Space for Processing Motion of Deformable Objects*, Computer Graphics Forum **35(2)**, 295–305 (2016).
1. **C. Brandt, Hans-Peter Seidel, Klaus Hildebrandt**, *Optimal Spline Approximation via ℓ_0 -Minimization*, Computer Graphics Forum **34(2)**, 617–626 (2015).