Concepts and Algorithms for the Deformation, Analysis, and Compression of Digital Shapes

by
Christoph von Tycowicz

A dissertation submitted to the
Fachbereich Mathematik und Informatik, Freie Universität Berlin
in partial fulfillment of the requirements for the degree of
doctor rerum naturalium
– Dr. rer. nat. –

Freie Universität Berlin

Defended May 5, 2014

Referees:
Prof. Dr. Konrad Polthier, Freie Universität Berlin
Prof. Dr. Mark Pauly, École Polytechnique Fédérale de Lausanne
Dr. Klaus Hildebrandt, Max–Planck–Institut für Informatik

Acknowledgements

It is my pleasure to thank all those who encouraged and supported me during the writing of this thesis. I thank my advisor Konrad Polthier for giving me the opportunity to work in this exciting research area and for providing an excellent research environment in Berlin. I am particularly grateful to Klaus Hildebrandt for inspiring discussions, constructive and critical remarks. This thesis would not exist without him and I am forever thankful for his invaluable advises and constant support. Thank you, Mark Pauly, for reviewing my thesis.

I have been fortunate to collaborate with Christian Schulz and Felix Kälberer. Working with them has been a delight and I am grateful for their tremendous devotion to our common research. I am indebted to the DFG research center Matheon "Mathematics for key technologies" for funding my research. I also wish to thank Jörn Loviscach who introduced me to the field of computer graphics and encouraged me to undertake this PhD. I have learned and benefited so much from the present and former members of the mathematical geometry processing group and I would like to thank all of them for the stimulating discussions and providing an excellent working atmosphere.

Finally, this thesis would not have been possible without my family and friends who accompanied me during the last few years. Words cannot convey my gratitude to my wife Ela for her love and support.

Contents

Acknowledgements **iii**

1 Introduction **1**
1.1 Summary of main achievements 3
1.2 Publications 4
1.3 Overview 5

2 Discrete deformable objects **7**
2.1 Introduction 7
2.2 Deformation energies 13
2.3 Linear vibration modes 20

3 Model reduction of nonlinear dynamical systems **25**
3.1 Introduction 25
3.2 Subspace construction 29
3.3 Approximation of reduced internal forces 39

4 Real-time simulation of deformable objects **51**
4.1 Introduction 51
4.2 Reduced equations of motion 53
4.3 Efficient subspace integration 54
4.4 Results and discussion 56

5 Interactive deformation-based modeling **59**
5.1 Introduction 59

5.2 Reduced-order shape editing 64
5.3 Efficient solver 66
5.4 Results and discussion 68

6 Modal shape analysis beyond Laplacian 75
6.1 Introduction 75
6.2 Vibration signature 78
6.3 Results and discussion 82

7 Compression of adaptive multiresolution meshes 87
7.1 Introduction 88
7.2 Hierarchy coding 91
7.3 Geometry compression 103
7.4 Results and discussion 116

Bibliography 121

Zusammenfassung 135

Verfassererklärung 137

Chapter 1

Introduction

Discrete representations of three-dimensional shapes are of paramount importance in areas such as product design, surgical planning, scientific computing, games and feature film production. To cope with the ever-growing demand in efficient, reliable, and scalable processing tools, solid theoretical and algorithmic foundations have to be developed. To this end, the field of digital geometry processing studies the entire lifecycle of a shape addressing the compression, fairing, parametrization, modeling, and manufacturing of digital shapes to name but a few stages of the processing pipeline. Because of their multiple applications in areas such as computer graphics or architectural geometry, but also due to technological advances such as the availability of low-end 3D printing devices, digital geometry processing has been—and will be—a fast-growing field of research.

An essential element of the geometry processing toolbox is the ability to edit and manipulate digital shapes. Approaches for the modeling of shapes range from purely geometric techniques to physics-based methods that account for material properties and environmental constraints. A major advantage of the latter schemes is that they allow for simple and intuitive user interfaces because they can produce physically meaningful deformations that match the user's experience of how shapes deform. However, treating a detailed digital shape as a physical deformable object is computationally expensive making interactive response times difficult to attain. For example, the dynamics of a deformable object are governed by partial differential equations and deformations of the object attained under static loading are solutions to variational problems. Since analytical solutions for arbitrary shapes are not known, these problems have to be solved numerically. Furthermore, the problems that accurately describe large deformations of an object are nonlinear, effectively prohibiting fully general interactive deformations of shapes with detailed geometry.

In this thesis, we present strategies for the construction of a simplified, low-dimensional model that captures the main features of the original high-dimensional system and is suitable for use in interactive computer graphics applications. Our approach is based on the observation that physical objects do not just deform arbitrarily but rather tend to assume smooth typical deformations. We thus restrict the deformations of objects to properly chosen low-dimensional subspaces that include such characteristic shapes. The resulting reduced systems involve much fewer degrees of freedom and hence hold the promise of superior runtime performance over the original systems but with some loss of accuracy. The subspace constructions we present are automatic and are suitable for use in large deformation scenarios where nonlinear terms of the systems are significant. Additionally, we provide schemes for fast approximation of reduced internal forces for general, nonlinear materials of elastic bodies. Thus, our model reduction techniques achieve runtime costs independent of the geometric complexity of the deformable object. To demonstrate the effectiveness of the new techniques we devise frameworks for real-time simulation and interactive deformation-based modeling. For that purpose, we propose efficient and robust methods for numerical integration and optimization that are tailored to the reduced systems. We evaluate the frameworks in experiments with elastic solids and shells and compare them to alternative approaches.

Physical principles are not only advantageous for the manipulation of shapes but also provide the means to gain insight into a shape. For example, many recent works in shape analysis are based on the heat diffusion process on a surface, which is a parabolic equation involving the Laplace–Beltrami operator. The derived methods profit from properties of the heat equation like isometry invariance and robustness to noise. Applications include segmentation, correspondence finding, shape retrieval, and symmetry detection. Because of its intrinsic nature, the Laplace–Beltrami operator is a powerful tool in applications where (almost) isometric deformations of a shape are considered (almost) identical. However, this property can also be a disadvantage since extrinsic features of a surface, like sharp bends, are of essential importance for some applications. In this thesis, we investigate discrete differential operators that can serve as an alternative to the discrete Laplacians. The operators we consider are derived from the mathematical models that describe the deformations of a shape and hence are sensitive to the extrinsic geometry of curved shapes. Based on these operators, we present the *vibration signature* – a multi-scale signature that serves as a dense feature descriptor and can be used to measure similarities of points on a shape.

Another element of digital geometry processing addressed in this thesis is the compression of digital shapes – an essential approach to achieve compact storage and fast transmission over bandwidth-limited channels. Unlike other types of multimedia, e.g., sound and video, curved shapes do not admit straightforward application of signal processing techniques from the Euclidean setting like the *fast Fourier transform*. However, many of these techniques can be generalized to surfaces with arbitrary topology based on the notion of semiregular meshes (also referred to as multiresolution meshes). These meshes result from successive refinements of a coarse base mesh and are, for example, inherent to multigrid methods for solving differential equations or level-of-detail visualizations in virtual environments. Applying the refinement locally increases the mesh resolution only where it is needed, but at the expense of a non-trivial hierarchical structure. In this thesis, we present a lossless connectivity compression that is adapted to the special characteristics of such adaptive multiresolution meshes. Using information theoretic strategies such as context-based arithmetic coding, we take advantage of structural regularities that are typically present in real-world data. Additionally, we present extensions that exploit correlations of the refinement structure in sequences of time-dependent meshes. We integrate this scheme with a wavelet-based coding of vertex positions for which we provide improved context modeling exploiting intraband and composite statistical dependencies.

1.1 Summary of main achievements

- We propose new model reduction techniques for the *construction of reduced shape spaces of deformable objects* and for the *approximation of reduced internal forces* that accelerate the construction of a reduced dynamical system, increase the accuracy of the approximation, and simplify the implementation of model reduction.
- Based on the model reduction techniques, we propose frameworks for deformation-based modeling and simulation of deformable objects that

are interactive, robust and intuitive to use. We devise efficient numerical methods to solve the inherent nonlinear problems that are tailored to the reduced systems. We demonstrate the effectiveness in different experiments with elastic solids and shells and compare them to alternative approaches to illustrate the high performance of the frameworks.

- We study the spectra and eigenfunctions of discrete differential operators that can serve as an alternative to the discrete Laplacians for applications in shape analysis. In particular, we construct such operators as the Hessians of deformation energies, which are in consequence sensitive to the extrinsic curvature, e.g., sharp bends. Based on the spectra and eigenmodes, we derive the *vibration signature* that can be used to measure the similarity of points on a surface.

- By taking advantage of structural regularities inherent to adaptive multiresolution meshes, we devise a lossless connectivity compression that exceeds state-of-the-art coders by a factor of two to seven. In addition, we provide extensions to sequences of meshes with varying refinement that reduce the entropy even further. Using improved context modeling to enhance the zerotree coding of wavelet coefficients, we achieve compression factors that are four times smaller than those of leading coders for irregular meshes.

1.2 Publications

The results presented in this dissertation were published in highly ranked journals, including ACM Transactions on Graphics, Computer Graphics Forum, and Computer Aided Geometric Design. More specifically, the techniques for model reduction and the derived schemes for interactive deformation-based modeling and real-time simulation have been published in [von Tycowicz et al., 2013] and [Hildebrandt et al., 2011]. The author also presented these results at the top-tier computer graphics conferences SIGGRAPH 2012 and SIGGRAPH Asia 2013. Furthermore, the concepts for modal shape analysis and the resulting vibration signature appeared in [Hildebrandt et al., 2012b] and [Hildebrandt et al., 2010]. In [von Tycowicz et al., 2011] and [Kälberer et al., 2009] we published our compression scheme for adaptive multiresolution meshes. In addition, our techniques address the storage demands inherent to high accuracy numerical simulations and optimal control of time-dependent processes. To demonstrate its efficiency we integrated the connectivity compression with a lossy coding scheme for the compression of finite element solution trajectories in [Götschel et al., 2013].

1.3 Overview

This thesis is organized around seven chapters: this introduction, one background chapter and five core chapters presenting this thesis's contributions. Each of the core chapters starts with an introduction of the addressed problem domain including a detailed overview of the related work. In addition to our technical contributions derived in the core chapters, we provide various experiments and in-depth comparisons to state-of-the-art methods.

This thesis is structured as follows. In the first part of the thesis we develop methods for applications in geometry processing that are based on concepts from continuum mechanics. In Chapter 2, we recall foundations of physically-based discrete deformable objects. Chapter 3 is concerned with the construction of simplified, low-dimensional models that well-approximate the dynamics of geometrically complex, nonlinear deformable objects. More specifically, we present techniques for the efficient construction of reduced shape spaces and for the approximation of reduced deformation energies and their derivatives. We compare our reduction techniques in terms of both accuracy and computational performance to state-of-the-art methods that are based on modal derivatives and greedy cubature optimization. In Chapter 4 we derive a framework for real-time simulation of deformable objects based on our model reduction techniques and demonstrate its efficiency in different experiments with elastic solids and shells. A framework for interactive deformation-based modeling is presented in Chapter 5. We illustrate the performance of our framework in comparisons to leading modelling approaches, i.e., *embedded deformations* and *rigid cells*. In addition to simulation and modeling, we employ the physical models that describe the deformations of a surface for shape analysis. In Chapter 6, we investigate differential operators that are derived from the deformation energy of a shape and hence can serve as alternatives to the Laplacian for applications in modal shape analysis. In the second part of the thesis, we address the compression of digital shapes. In particular, in Chapter 7 we present a lossless compression scheme that is adapted to the special characteristics of adaptively refined, hierarchical meshes. We provide comparisons with leading compression schemes for irregular as well as uniform multiresolution meshes.

Chapter 2

Discrete deformable objects

Figure 2.1: Physically-based deformable objects have widespread application in mature graphics areas, such as the animation of soft bodies.

2.1 Introduction

Physically-based deformable objects play an important role in fields like structural dynamics, surgical planning, and sound synthesis to name a few. Since the seminal work by Terzopoulos et al. [1987] they are an active topic in computer graphics and their use in key areas such as cloth simulation, deformation-based editing and character animation have expanded tremendously. Numerous contributions were made in the past decades and we refer to the excellent surveys [Gibson and Mirtich, 1997; Nealen et al., 2006] for an overview.

The dynamics of a continuous object are governed by the equations of continuum mechanics—a system of partial differential equations that depend on time and spatial coordinates. There exist a multitude of approaches to discretize real-world, deformable solids such as the finite element method (FEM), the finite differences method, mass-spring systems or meshfree methods. Each of these methods possess both benefits and drawbacks and have particular applications. However, mesh-based (approximate) continuum models, i.e., models that treat the object as a continuum with mass and energies distributed throughout, are usually considered to be more accurate and reliable than those modeling the solid using particle systems, cf. Gibson and Mirtich [1997]. Continuum models expect the entire space occupied by the object to be represented by a mesh, e.g., by decomposing the object into tetrahedra. Typically, the deformable model is defined by the undeformed configuration of the mesh (also called the rest state or initial configuration) and by a set of material parameters that specify how the object deforms under applied forces.

Many discretizations are tailored to a particular type of mesh and admissible deformations while others, e.g., FEM, can be applied to various representations. In this introduction, we focus on simplicial complexes with continuous, piecewise linear shape functions. This setting is particularly popular in computer graphics and is the one most commonly used by engineers for solving second-order problems with conforming FEMs [Ciarlet, 2002]. In the remainder of this section, we first introduce simplicial complexes and then derive the notion of a shape space.

2.1.1 Simplicial complexes

Triangle and tetrahedral meshes—or more generally simplicial complexes—are among the most commonly used data structures to represent all kinds of shapes in computer graphics and computational geometry. Compared to the purely geometric description found in many textbooks, e.g. [Munkres, 1996; Lee, 2010], practical mesh processing clearly differentiates between the combinatorial properties of a mesh and its geometric shape. Here, we present such a distinguished treatment that allows us to represent a larger variety of shapes including objects with self-intersections.

In topology, simplicial complexes are used to construct polyhedra by "gluing together" points, lines segments, triangles, and their higher dimensional counterparts. These n-dimensional building blocks are called simplices. We begin with a combinatorial point of view by introducing abstract simplices and abstract simplicial complexes.

Definition 1 *An* ***abstract simplicial complex*** *is a collection $\mathcal{K}$ of finite nonempty sets, such that if s is an element of $\mathcal{K}$, so is every nonempty subset of s. The element s is called an* ***abstract simplex*** *and any nonempty subset of s is called a* ***face*** *of s. The dimension of s is one less than the number of elements of s; the* ***vertex set*** *$\mathcal{K}^0$ is the union of all 0-dimensional simplices of $\mathcal{K}$.*

For now, we consider points as abstract entities and do not care about their specific nature. By combining one or more abstract points we specify a simplex and if two simplices share common points they are glued together. In particular, two s-dimensional simplices are said to be **adjacent** to each other if they share a common $s-1$ dimensional face.

The **dimension** of a simplicial complex is the maximum dimension of its simplices. A k-dimensional complex is said to be **pure** if it contains only simplices that are faces of k-dimensional simplices. Henceforth, we will restrict our attention to **finite** complexes, i.e., complexes that are themselves finite sets.

We perform the transition from the combinatorial to the geometric point of view by defining the geometric representatives associated to the abstract simplices. First, the set $\{p_0, \dots, p_k\}$ of points in $\mathbb{R}^n$ is said to be **geometrically independent** if and only if the vectors $p_1 - p_0, \dots, p_k - p_0$ are linearly independent.

Definition 2 *A* ***k-simplex*** *$\sigma^k = [p_0, \dots, p_k]$ is the convex hull of $k+1$ geometrically independent points $p_0, \dots, p_k$ in $\mathbb{R}^n$, i.e.,*

$$\sigma^p = \left\{ x \in \mathbb{R}^n \mid x = \textstyle\sum_{i=0}^{k} \mu_i p_i \text{ where } \mu_i \geq 0 \text{ and } \sum_{i=0}^{k} \mu_i = 1 \right\}.$$

Any simplex spanned by a nonempty subset of $\{p_0, \dots, p_k\}$ is called a ***face*** *of σ^p. The number k is called the* ***dimension*** *of the simplex.*

With the notion of geometric k-simplices we can now obtain a geometric shape from an abstract simplicial complex by relating abstract simplices to geometric ones.

Definition 3 *A **(geometric) simplicial complex** K in $\mathbb{R}^n$ is a collection of simplices in $\mathbb{R}^n$ such that*

1. *Every face of a simplex of K is in K.*
2. *The intersection of any two simplices of K is a face of each of them.*

If V is the vertex set of a simplicial complex K, let $\mathcal{K}$ be the collection of all subsets $\{a_0, \ldots, a_k\}$ of V such that the vertices $a_0, \ldots, a_k$ span a simplex of K. Then $\mathcal{K}$ is an abstract simplicial complex and is called the **vertex scheme** of K. On the other hand, as K relates a geometric shape to the abstract simplicial complex it is also referred to as a **geometric realization** of $\mathcal{K}$. Any geometric realization of an abstract simplicial complex into a Euclidian space induces a topology on the simplicial complex. Explicitly, let $|K| \subset \mathbb{R}^n$ denote the union of all simplices in K. We define a topology on $|K|$ as the natural topology as a subspace of $\mathbb{R}^n$. The space $|K|$ is called the **underlying (topological) space** of K. For our complexes to describe deformable objects, we carry the notion of a topological manifold over to the context of a simplicial complex.

Definition 4 *A **simplicial k-manifold** K in $\mathbb{R}^n$ is a k-dimensional simplicial complex whose underlying space $|K|$ is a topological submanifold of $\mathbb{R}^n$.*

In particular, we call a simplicial 2-manifold a **simplicial surface** and a simplicial 3-manifold a **simplicial volume**. Furthermore, an abstract simplicial complex is said to be manifold if it is the vertex scheme of a simplicial manifold.

In computer graphics and computational geometry practice, digital shapes are typically represented by meshes, where geometric positions are treated separately from the connectivity of the vertices. This distinction allows for self-intersections and even degenerated shapes, so this possibility should be taken into account when working with discrete deformable objects. To this end, we introduce the notion of a simplicial mesh:

Definition 5 *A **simplicial mesh** $N = (\mathcal{K}, f)$ in $\mathbb{R}^d$ is an abstract simplicial manifold $\mathcal{K}$ together with a correspondence $f : \mathcal{K}^0 \to \mathbb{R}^d$ such that for each $\{a_0, \ldots, a_k\} \in \mathcal{K}$ the points $f(a_0), \ldots, f(a_k)$ are geometrically independent.*

The correspondence f can be extended to a (unique) mapping f' of any geometric realization K of $\mathcal{K}$ that is linear in each simplex. Note that although this mapping is an immersion of each simplex in K, it can cause non-adjacent simplices to intersect and hence K might lose its property of being a simplicial complex under the application of f'. For this reason we induce the topology of N from a geometric realization K, but stick to the mesh's metric induced by the corresponding mapping f'.

2.1.2 Shape space

Throughout the thesis, we will assume $\mathbb{R}^3$ to be the ambient space of the deformable object though many of the presented ideas and concepts can be applied equally well in other dimensions. Let $\Omega \subset \mathbb{R}^3$ be the volumetric domain that represents the *reference configuration* of a deformable object. A deformation $\phi : \Omega \to \mathbb{R}^3$ is a sufficiently smooth and orientation preserving immersion of the object. Points in Ω are denoted by X and are called *material points*, while *spatial points* in $\mathbb{R}^3$ are denoted by $x = \phi(X)$. Then, a *motion* of the object is a time-dependent family of configurations $\phi(X,t)$. The *velocity* of a material point X is regarded as a vector emanating from x and is defined by $V(X,t) = (\partial/\partial t)\phi(X,t)$. The velocity viewed as a function of (x,t) is called the *spatial velocity* and is denoted by $v(x,t)$. Note that $v(x,t) = V(X,t)$, where $x = \phi(X,t)$. In order to derive a discrete dynamical model with finite degrees of freedom, we approximate the continuous object using a simplicial mesh N and additionally restrict the possible deformations to a finite dimensional space. In particular, we consider the space $F(N)$ of functions mapping $N \to \mathbb{R}$ that are continuous on N and linear on each simplex of N. This space falls into the category of finite element spaces. We refer to [Ciarlet, 2002] for a more detailed introduction to piecewise polygonal function spaces on simplicial domains.

Functions in $F(N)$ are spanned by the Lagrange basis functions $\{\varphi_0, \ldots, \varphi_{n-1}\}$ corresponding to the set of vertices $\{p_0, \ldots, p_{n-1}\}$ such that for each vertex p_i there is a function φ_i with $\varphi_i(p_j) = \delta_{ij}\ \forall i, j \in \{0, \ldots, n-1\}$. Thus a deformation $\phi \in F(N)$ of the object is determined by the displacements of the n vertices and has the unique representation

$$\phi(X) = X + \sum_{i=0}^{n-1} u_i \varphi_i(X) \quad \forall X \in N,$$

where $u_i = \phi(p_i) \in \mathbb{R}^3$. The displacements of the vertices can be written in one $3n$-vector $u = (u_0, \ldots, u_{n-1})$ called the *nodal vector*. Hence, we can identify the space of all possible deformations of the object with $\mathbb{R}^{3n}$. This space is called the *shape space* or *configuration space* and we denote it by $\mathcal{C}$. We would like to mention that the ideas presented in this section are also valid for different choices of shape functions, e.g., piecewise quadratic or subdivision-based finite element spaces. Note that n is the dimension of the function space and hence can be different from the number of vertices.

The velocity V of an object is an element of the tangent space $T_\phi\mathcal{C}$ at the deformed shape $\phi(N)$. Since $\mathcal{C}$ equals $\mathbb{R}^{3n}$, the tangent space $T_\phi\mathcal{C}$ can be identified with $\mathbb{R}^{3n}$. We write $\dot{u} = (\dot{u}_0, \ldots, \dot{u}_{n-1})$ to denote the nodal vector that represents V in the Lagrange basis.

In the context of dynamical systems, a natural choice for a metric on $\mathcal{C}$ is given by the kinetic energy $E_{\text{kin}}(v) = {}^1\!/_2 \int_{\phi(N)} \rho(x) \langle v(x), v(x)\rangle \,\mathrm{d}v$, where $\rho(x)$ is the mass density of the body. By the polarization identity, the metric $g(z,w)$ for tangent vectors $z, w \in T_\phi\mathcal{C}$ is then given by $g(z,w) = {}^1\!/_4(E_{\text{kin}}(z+w) - E_{\text{kin}}(z-w))$. Using the Lagrangian basis functions $\{\psi_0, \ldots, \psi_{n-1}\}$ on $\phi(N)$, i.e., $\psi_i \circ \phi = \varphi_i \;\forall i \in \{0, \ldots, n-1\}$, the kinetic energy can be expressed as

$$E_{\text{kin}}(\dot{u}) = \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\langle \dot{u}_i, \dot{u}_j\rangle \int_{\phi(N)} \rho\, \psi_i\psi_j \,\mathrm{d}v = \dot{u}^T M \dot{u},$$

where M is called the mass matrix. Let us assume from now on that for $t = 0$ the body is in the reference configuration, i.e., $\phi(\,\cdot\,, 0)$ is the identity, and let $\bar{\rho}(X) = \rho(x, 0)$ denote the mass density at the rest state. If $\rho(x,t)$ obeys the *conservation of mass* principle, then $\bar{\rho}(X) = \det(D\phi)\rho(x,t)$ for all $x = \phi(X,t)$ (see [Marsden and Hughes, 1994]), where $D\phi$ is the 3×3 matrix of partial derivatives of ϕ with t held constant and is called the *deformation gradient*. By the change of variables formula and the conservation of mass, the kinetic energy can be expressed as $E_{\text{kin}}(V) = {}^1\!/_2 \int_N \bar{\rho}(X) \langle V(X), V(X)\rangle \,\mathrm{d}V$, i.e., an integral over the reference configuration that depends on $\bar{\rho}$ and V instead of ρ and the spatial velocity. Thus, the mass matrix M does not depend on the deformation of the object. For completeness, we would like to note that away from the closed set of rank-deficient configurations ϕ, i.e., deformations with at least one degenerate simplex, the space $\mathcal{C}$ equipped with $g(\cdot,\cdot)$ is a Riemannian manifold.

Other choices of scalar products are possible as well. For example, omitting the conservation of mass by assuming $\rho \equiv 1$ for all ϕ yields the discrete L^2 *– product* for vector fields on $\phi(N)$ as a scalar product on $T_\phi\mathcal{C}$ and was used for the numerical integration of gradient flows of geometric functionals like the area (mean curvature flow) or the Willmore energy, see [Dziuk, 1991;

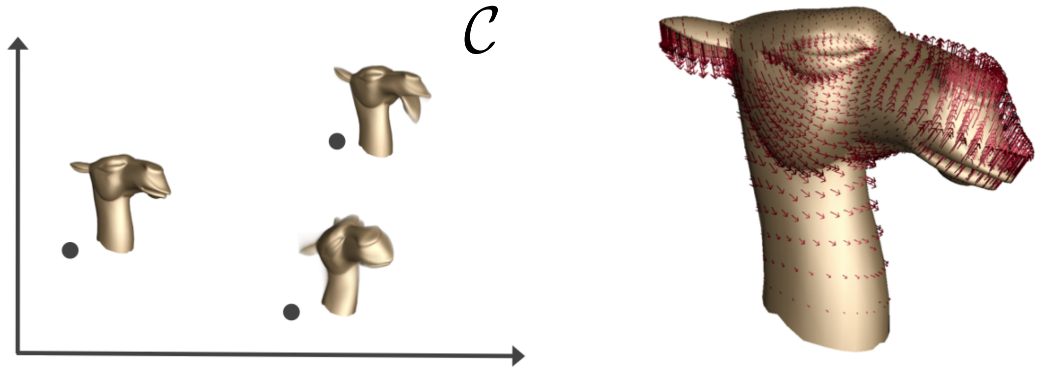


Figure 2.2: (Left) Each point in the shape space $\mathcal{C}$ corresponds to a deformation of the object. (Right) We can interpret an element of $T_\phi\mathcal{C}$ as a vector field on $\phi(N)$, which assigns a vector in $\mathbb{R}^3$ to every point $x = \phi(X)$.

Clarenz et al., 2004]. Gradient flows of geometric functionals with respect to alternative choices of scalar products are explored in [Eckstein et al., 2007]. Furthermore, Kilian et al. [2007] construct different semi-Riemannian metrics that are tailored to certain design and modeling task, e.g., shape exploration and morphing. However, as we are concerned with dynamical systems of deformable objects, we will employ the metric derived from the kinetic energy with conservation of mass throughout this thesis.

2.2 Deformation energies

In this section, we consider discrete deformation energies defined for objects whose reference configurations are represented by simplicial volume meshes N in $\mathbb{R}^3$. For simplicity we assume that N is embedded in $\mathbb{R}^3$, i.e., it is itself a simplicial volume in $\mathbb{R}^3$ and hence $|N| \subset \mathbb{R}^3$. We will start this introduction with basic notions and concepts from 3D elasticity and then present a concise introduction to the theory of thin shells. As a supplement to this introduction we highly recommend the textbooks [Marsden and Hughes, 1994] and [Bonet and Wood, 2008].

Deformation energies measure the deformation of a mesh from a reference configuration, i.e., a function on the shape space that, for every configuration in the shape space, provides a quantitative assessment of the magnitude of the deformation from the rest state. More precisely, we formulate the energy as $E : \mathcal{C} \to \mathbb{R}$, which implies that the energy is uniquely determined by the deformation $\phi \in \mathcal{C}$. This notation reflects the essential hypothesis that the deformation energy does not depend on the deformation history, i.e., it does not matter along which path the object was deformed to reach its current configuration. This absence of hysteresis effects in the energy is a property called *hyperelasticity* and is closely related to the fact that internal forces of hyperelastic objects are *conservative*, i.e., the total work done by the elastic forces depends solely on the initial and final configurations and not on the trajectory. On the contrary, inelastic materials dependent on the deformation history, e.g., materials that underwent plastic deformations will not return to their original shape after the external forces have been released; thus the energy before and after plastic deformation will be different for the same shape.

In continuum mechanics, the severity of a deformation is measured on a local scale as different parts of a deforming object undergo different shape changes. This approach leads to an energy density function $\Psi(\phi, X)$ that measures the deformation energy per *unit undeformed volume* on an infinitesimal domain $\mathrm{d}V$ around a material point X. The first step to deriving the energy density Ψ is to define a quantitative descriptor for the severity of a given deformation called *strain*. As the strain is intended to gauge the behavior of ϕ in an infinitesimal neighborhood of a material point $\hat{X}$, it is reasonable to approximate ϕ by its first-order Taylor expansion: $\phi(X) \approx \phi(\hat{X}) + D\phi|_{\hat{X}}(X - \hat{X})$. Constant terms in the approximation correspond to translations of the region around $\hat{X}$ and hence cause no change in strain energy. Thus, the energy density should be a function of the deformation gradient $D\phi$ and X alone, i.e., $\Psi(\phi, X) = \Psi(D\phi|_X, X)$. We refer to [Marsden and Hughes, 1994], in particular Theorems 2.4 and 2.6, for a rigorous derivation of the dependencies of Ψ and the corresponding constitutive axioms. Despite its descriptive power, the deformation gradient still contains information that is unrelated to shape change. In particular, if ϕ represents a rigid rotation R of the whole body, we have $D\phi = R$. Thus, $D\phi$ is not invariant under rigid motions. However, by the polar decomposition from linear algebra, we can uniquely factorize the deformation gradient as $D\phi = RU = VR$, where R is again a rotation matrix, and U and V are symmetric, positive-definite matrices called *right* and *left stretch tensors*. Indeed, $C = (D\phi)^T D\phi = U^T R^T R U = U^2$ is invariant under rigid body motions and is called the *right Cauchy-Green tensor*. From the viewpoint of differential geometry, C is the pullback of the first fundamental form (or metric tensor) of the deformed to the reference configuration. The actual strain caused by a deformation can thus be measured in terms of C based on the Green-Lagrange strain tensor $\mathcal{E} = {}^1/_2(C - \boldsymbol{1})$, where $\boldsymbol{1}$ is the metric tensor of the reference configuration and is represented by the 3×3 identity matrix.

Both C and $\mathcal{E}$ depend solely on the deformation and are thus purely geometric measures. However, the response of a material to stimuli depends on the physical properties of that material such as compressibility or resistance to shear. These physical traits are encoded by the *stress* of a material which, in general, depends on strain (history), strain rate, and other field quantities. The stress at a point is a measure of force per unit area acting on an elemental area $\mathrm{d}s$. This concept applies not only to points on the boundary of a solid but can also be considered for surfaces created by (imaginarily) slicing the deformed body with a plane. The force per unit area acting on $\mathrm{d}s$ is called the *stress vector* and depends on the direction and magnitude of the force as well as on the orientation of the surface on which it acts. The stress vector acting on a plane perpendicular to n is denoted by $\tau(n)$. By Cauchy's stress theorem, the stress vector is a linear function of n and can thus be written as $\tau(n) = n \cdot \sigma$, where σ is a second order tensor field called the Cauchy stress tensor. Apart from the Cauchy stress tensor there are also other commonly encountered formulations of stress that differ in whether they refer to stress vectors and surface area elements measured in the reference state or in the deformed body. In particular, the second Piola–Kirchhoff stress tensor S is measured in the deformed body but referred to in the material coordinates X. Thus, S is commonly used together with the Green–Lagrange strain $\mathcal{E}$ that is also a function of X. Both stress measures are related by the so-called Piola transformation [Bonet and Wood, 2008] and for small deformation problems the difference between them disappears.

With stress and strain measures at hand, we can formulate the strain energy density in material coordinates as $\Psi(D\phi|_X, X) = S : \mathcal{E}$, where the colon denotes tensor contraction. We can then obtain the total potential energy of elastic forces by integrating the strain energy density over the entire domain N:

$$E(\phi) = \int_N \Psi(D\phi|_X, X)\,\mathrm{d}V = \int_N S : \mathcal{E}\,\mathrm{d}V.$$

The mathematical formulation that relates stress to strain for a hyperelastic material is called its constitutive model. In general, these constitutive equations can not be calculated using physical laws and are thus derived by either fitting the model to experimental measurements or by arguing about the underlying structure of a material. They must, however, satisfy certain physical principles such as the condition of objectivity, or material frame indifference. Materials that exhibit direction-dependent properties, i.e., they are more "stretchable" in some directions than in others, are said to be *anisotropic*. An *isotropic* body, on the other hand, is one for which every material property in all directions at a point is the same. If the material properties vary with the position within the body, the material is called *heterogeneous*. Otherwise, if the properties are the same throughout the body it is called *homogeneous*. A heterogeneous or homogeneous material can be isotropic or anisotropic. In general, the strain-stress relation can take many different forms and a detailed discussion is beyond the scope of this thesis. Here we introduce hyperelastic, isotropic materials for which stress is assumed to depend linear on strain. However, the methods presented in this thesis are not limited to such linear materials and, in particular, we provide experimental results from the materially nonlinear Mooney–Rivlin model. The linear strain-stress relation is also referred to as the generalized Hooke's law and is formulated as $\sigma = \mathbf{C} : \epsilon$, where $\mathbf{C}$ is the fourth-order stiffness tensor. We can think of $\mathbf{C}$ as a $3 \times 3 \times 3 \times 3$ multi-dimensional array. Due to symmetry inherent to the strain, stress and stiffness tensor, there are only 21 independent coefficients in $\mathbf{C}$. Moreover, for isotropic media $\mathbf{C}$ depends only on two independent parameters λ, μ referred to as the Lamé constants.

St. Venant–Kirchhoff material Linear materials are inherent to the field of *linear elasticity* that provides efficient models for bodies undergoing small deformations where nonlinear terms can be neglected. However, combining the linear, isotropic model with the rotationally invariant Green-Lagrange strain $\mathcal{E}$ gives rise to a geometrically nonlinear, yet simple model that exhibits plausible material response in many large deformation scenarios. In particular, this model is called the St. Venant–Kirchhoff material and it is given by the relation

$$S = \lambda \mathrm{tr}(\mathcal{E}) \mathbf{1} + 2\mu\mathcal{E}, \tag{2.1}$$

where S is the second Piola–Kirchoff stress tensor and $\mathbf{1}$ denotes the 3×3 identity matrix. As $D\phi$ depends linearly on ϕ, both $\mathcal{E}$ and S are quadratic in ϕ. Therefore, the potential energy of the St. Venant–Kirchhoff material always takes the shape of a fourth-order polynomial function of ϕ. The St. Venant–Kirchhoff model is very popular in computer graphics due to its simplicity and the benefits it offers over (geometrically) linear elastic models. However, its scope is limited to applications that do not involve large volume compressions. If a St. Venant–Kirchhoff elastic body is compressed, the restorative force initially grows but peaks once a critical compression threshold is reached. Further compression beyond this point will cause the restorative force to decrease and even vanish if the object is squashed down to zero volume. Increasing the compressive load will then cause the body to invert and create elastic forces that pushes it towards complete inversion. On the other hand, if the body is stretched the restorative force will grow arbitrarily large as expected.

Therefore, this material typically produces visually plausible results for large deformations with moderate volume compression.

2.2.1 Thin shells

The analysis of flexible shells and plates is an old and well-studied subject and it has seen many solutions. The different approaches proposed in the literature and relevant references thereof are too numerous to list here. We refer the reader to the excellent books [Reddy, 2007; Ciarlet, 2000] and the extensive references therein.

A plate is a planar structural element with small thickness compared to its planform dimensions. Typically, the thickness is less than one-tenth of the smallest in-plane dimension. Unlike plates, shells are not characterized by a flat undeformed configuration but relax to a curved shape when unstressed. Plates can thus be seen as a degenerate class of shells. Aluminium cans, car bodies, fingernails, ship hulls and eggs are everyday examples of thin shells. Because of the small thickness-to-width ratio, it is often not necessary to model shells using three-dimensional elasticity equations. The aim of shell theory is to replace the three-dimensional problem by a presumably much simpler two-dimensional one.

Typically the undeformed body of a shell with uniform thickness $h > 0$ is parametrized by a system $\{y_1, y_2, y_3\}$ of curvilinear coordinates:

$$\mathbf{\Theta}(y_1, y_2, y_3) := \boldsymbol{\theta}(y_1, y_2) + y_3 a_3(y_1, y_2) \quad \text{for all} \quad y \in \bar{\omega} \times [-h/2, h/2]\,,$$

where $\Sigma = \boldsymbol{\theta}(\bar{\omega})$ describes the parametrized middle surface and $\bar{\omega} \subset \mathbb{R}^2$. Let $\boldsymbol{\theta} \in C^3(\bar{\omega}; \mathbb{R}^3)$ be an immersion and let $\partial_\alpha := \partial/\partial y_\alpha$, then the two vectors

$$a_\alpha := \partial_\alpha \boldsymbol{\theta} \quad \alpha \in \{1, 2\}$$

are linearly independent at all points $(y_1, y_2) \in \bar{\omega}$ and thus span the tangent plane to Σ at $\boldsymbol{\theta}(y_1, y_2)$. The unit vector given by

$$a_3 := \frac{a_1 \times a_2}{\|a_1 \times a_2\|}$$

is normal to the middle surface at any point of Σ. The covarient basis vectors of the shell are then given by

$$\begin{aligned} g_\alpha &= \partial_\alpha \mathbf{\Theta} = a_\alpha + y_3 \partial_\alpha a_3 \quad \alpha \in \{1, 2\}, \\ g_3 &= \partial_3 \mathbf{\Theta} = a_3, \end{aligned}$$

and the corresponding components of the covarient metric tensor (or first fundamental form of $\boldsymbol{\Theta}$) are $g_{ij} = g_i \cdot g_j$ for $i, j \in \{1, 2, 3\}$ (recall that $\partial_\alpha a_3$ is in the tangent plane to Σ at $\boldsymbol{\theta}(y_1, y_2)$ and therefore $g_{3\alpha} = g_{\alpha 3} = 0$ and $g_{33} = 1$).

Such a parametrized shell can now be modelled as a three-dimensional problem by considering displacements of the points in $\boldsymbol{\Theta}(\bar{\omega} \times [-h/2, h/2])$. However, to achieve a dimensional reduction assumptions are made as to the form of the displacement field or stress field. Here we confine our attention to the Kirchhoff-Love theory of thin shells which is based on the following kinematic assumptions:

1. Straight lines perpendicular to the middle surface (i.e., transverse normals) in the undeformed configuration remain straight after deformation.
2. The transverse normals remain perpendicular to the middle surface after deformation.
3. The thickness of the shell does not change (i.e., the transverse normals are inextensible).

Under these assumptions, any displacement field $\eta : \bar{\omega} \to \mathbb{R}^3$ of the middle surface uniquely determines the deformed configuration of the shell which can then be parametrized by

$$\boldsymbol{\Theta}^\eta(y_1, y_2, y_3) := \boldsymbol{\theta}^\eta(y_1, y_2) + y_3 a_3^\eta(y_1, y_2),$$

where $\boldsymbol{\theta}^\eta := \boldsymbol{\theta} + \eta$ and a_3^η are the deformed middle surface and its normal, respectively. The Green-Lagrange strain tensor $\mathcal{E}$ for such a parametrized shell measures the difference between the metric tensors of the undeformed and deformed configurations of the shell and its components are

$$\mathcal{E}_{ij} = {}^1/_2(g_{ij}^\eta - g_{ij}),$$

where g_{ij}^η denotes the components of the covarient metric tensor of the deformed shell. By virtue of the assumed Kirchhoff-Love kinematics, the $\mathcal{E}_{\alpha\beta}$ for $\alpha, \beta \in \{1, 2\}$ are the only non-zero components of $\mathcal{E}$ and may be deduced from the middle surface as

$$\mathcal{E}_{\alpha\beta} = \underbrace{{}^1/_2(a_{\alpha\beta}^\eta - a_{\alpha\beta})}_{G_{\alpha\beta}} + y_3 \underbrace{(b_{\alpha\beta}^\eta - b_{\alpha\beta})}_{R_{\alpha\beta}} + {}^1/_2 y_3^2 (e_{\alpha\beta}^\eta - e_{\alpha\beta}),$$

where the components of the first, second and third fundamental forms of $\boldsymbol{\theta}^\eta$ and $\boldsymbol{\theta}$ are denoted by $a_{\alpha\beta}^\eta$ and $a_{\alpha\beta}$, $b_{\alpha\beta}^\eta$ and $b_{\alpha\beta}$, and $e_{\alpha\beta}^\eta$ and $e_{\alpha\beta}$, respectively.

Therefore, since the three fundamental forms of a surface are linearly dependent (cf. [Toponogov, 2006, Theorem 2.5.1]), the change in metric tensor $G_{\alpha\beta}$ and change in curvature tensor $R_{\alpha\beta}$ furnish a complete description of the deformation of the shell. In particular, $G_{\alpha\beta}$ encodes the stretching and shearing of the middle surface, whereas $R_{\alpha\beta}$ describes the bending of the middle surface. Assuming a linear constitutive law, the elastic energy of the shell can be formulated as in [Terzopoulos et al., 1987] by

$$E(\eta) = \frac{1}{2}\int_{\Sigma} \left(\|G\|^2_{\text{mem}} + \|R\|^2_{\text{bend}}\right) \mathrm{d}A,$$

where $\mathrm{d}A = \sqrt{\det(a_{\alpha\beta})}\mathrm{d}y_1\mathrm{d}y_2$ denotes the area element of Σ and $\| \ \|_{\text{mem}}$ and $\| \ \|_{\text{bend}}$ are certain matrix (semi)norms that measure the membrane strain energy density and the bending strain energy density, respectively, and hence encode the thickness and material properties of the shell.

Discrete shells Discretizations of the thin shell energy depend only on a representation of the middle surface and its corresponding shape space. It is clear from the form of the bending term that in a finite element discretization the displacement fields must necessarily have square integrable first and second derivatives, i.e., they belong to the Sobolev space $H^2(\bar{\omega}, \mathbb{R}^3)$. A major difficulty of this requirement is guaranteeing the continuity across neighboring elements which typically requires the introduction of additional variables like partial derivatives. Cirak et al. [2000] circumvent such difficulties by introducing shape functions based on the notion of subdivision surfaces that achieve the required continuity due to their nonlocal nature.

An alternative approach to discretize the shell equations is based on principles of discrete differential geometry and promises physically plausible, yet computationally efficient, discrete shell models. Let N denote the simplicial surface mesh that represents the middle surface of the shell and let $\mathcal{C}$ be the corresponding shape space spanned by continuous, piecewise linear shape functions (as introduced in the previous section). Then, the *Discrete Shells* model [Grinspun et al., 2003] defines a discrete constitutive model using shape operators derived from discrete geometric ideas. Analogous to the continuous case, the energy that governs this model of thin shells is a weighted sum of a flexural energy and a membrane energy,

$$E(u) = \kappa_{\text{bend}}\, E^F(u) + \kappa_{\text{mem}}\, (E^L(u) + E^A(u)). \tag{2.2}$$

The weights κ_{bend} and κ_{mem} are the bending and membrane stiffness and reflect the material properties to be simulated, e.g., in cloth simulation the membrane energy usually gets a high weight due to the stretch resistance of cloth. The discrete bending (or flexural) energy is given as a summation over the internal edges of N:

$$E_F = \frac{3}{2} \sum_i \frac{\|\bar{e}_i\|^2}{\bar{A}_{e_i}} \left(\theta_{e_i} - \bar{\theta}_{e_i}\right)^2, \tag{2.3}$$

where θ_{e_i} is the dihedral angle at the edge e_i, A_{e_i} is the combined area of the two triangles incident to e_i and $\|e_i\|$ is the length of the edge. The quantities $\|\bar{e}_i\|$, $\bar{A}_{e_i}$, and $\bar{\theta}_{e_i}$ are measured on the reference configuration of the surface. The membrane energy consists of two terms: one measuring the stretching of the edges,

$$E_L = \frac{1}{2} \sum_i \frac{1}{\|\bar{e}_i\|} (\|e_i\| - \|\bar{e}_i\|)^2, \tag{2.4}$$

and one measuring the change of the triangle area A_i

$$E_A = \frac{1}{2} \sum_i \frac{1}{\bar{A}_i} (A_i - \bar{A}_i)^2. \tag{2.5}$$

Here the second sum runs over the triangles of N. Alternatively, more recent works like [Gingold et al., 2004; Heeren et al., 2012] employ the standard constant strain triangle (CST) formulation that admits to represent the membrane energy using the well-known Koiter's shell model [Koiter, 1966].

2.3 Linear vibration modes

Vibration theory is concerned with oscillatory motion of physical systems and provides ways to compute the vibration modes of a deformable object. We refer to [Shabana, 1997] for a survey of the theory of vibrations of both discrete and continuous systems. The oscillatory motion of a system depends on boundary conditions, the geometry of the object as well as its material, and can be observed around stable equilibrium configurations of the object with respect to internal and external forces. Let $\bar{u} \in \mathcal{C}$ be such a stable equilibrium. Then a small disturbance of the system from equilibrium results only in a small bounded motion about $\bar{u}$. As the acting forces are in equilibrium, the motion in the immediate neighborhood of $\bar{u}$ is governed by the linear terms in a Taylor expansion of the forces. Assuming conservative systems and vanishing derivatives of external forces, the vibrations of the object are determined by the eigenvalues and eigenmodes of the Hessian of the deformation energy E at the configuration $\bar{u}$.

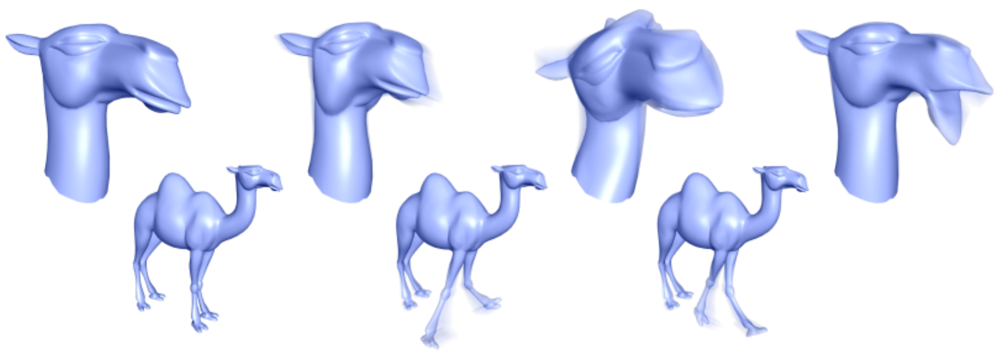


Figure 2.3: Visualization of vibration modes derived from the discrete thin shell energy. In each row the left most image shows the rest state followed by deformations captured by a vibration mode.

The Hessian of a deformation energy (or more generally of a function) does not only depend on the differentiable structure of $\mathcal{C}$ but also on the metric on $\mathcal{C}$ and hence is a quantity from Riemannian geometry.

We denote by ∂E_u the $3n$-vector containing the first partial derivatives of E at u and by $\partial^2 E_u$ the matrix containing the second partial derivatives at u. The matrix $\partial^2 E_u$ is also referred to as the tangent stiffness matrix. We would like to emphasize that ∂E_u and $\partial^2 E_u$ do not depend on the metric on $\mathcal{C}$, whereas the gradient and the Hessian of E do. The gradient of E at u is given by

$$\mathrm{grad}_u E = M^{-1} \partial E_u. \tag{2.6}$$

The Hessian of E at a mesh u is the self-adjoint operator that maps any tangent vector $v \in T_u\mathcal{C}$ to the tangent vector $\mathrm{hess}_u E(v) \in T_u\mathcal{C}$ given by

$$\mathrm{hess}_u E(v) = \nabla_v \mathrm{grad}_u E, \tag{2.7}$$

where ∇ is the covariant derivative of $\mathcal{C}$.

Hessian computation In general, it is a delicate task to derive an explicit representation of the Hessian of a deformation energy and often only approximations of the Hessian are available. As pointed out in the preceding section, deriving the metric of $\mathcal{C}$ from the kinetic energy with conservation of mass leads to a constant mass matrix. The Hessian of E therefore takes the following form

$$\mathrm{hess}_u E = M^{-1} \partial^2 E_u, \tag{2.8}$$

as we do not need to consider derivatives of the metric. Since there can be no motion of bodies while the kinetic energy remains zero, the metric of $\mathcal{C}$ is always positive definite. Thus, the mass matrix is also positive definite and its inverse exists.

Eigenvalue problem To obtain the eigenmodes $\Phi_i \in T_{\bar{u}}\mathcal{C}$ of $\text{hess}_{\bar{u}}E$, we need to solve the generalized eigenvalue problem

$$\partial^2 E_{\bar{u}}\, \Phi_i = \omega_i^2\, M\, \Phi_i. \tag{2.9}$$

Since both $\partial^2 E_{\bar{u}}$ and M are symmetric and positive definite ($\bar{u}$ is a stable equilibrium), all eigenvalues ω_i^2 are positive real numbers, and their square roots ω_i are called *natural frequencies* of the system. We would like to remark that if the system admits motions that cause no change in energy E (e.g., rigid motions), $\partial^2 E_{\bar{u}}$ is only positive semi-definite and its kernel is the set of all vectors in $T_{\bar{u}}\mathcal{C}$ tangent to those motions. An important property of the eigenmodes, also called vibration modes or mode shapes, is their orthogonality with respect to both $\partial^2 E_{\bar{u}}$ and M (see [Shabana, 1997]). Its utility for the solution of dynamical systems will become apparent in the remainder of this section.

The eigenvalue problem (2.9) typically involves large but sparse matrices. Fast solvers for these problems are discussed in [Saad, 1992] and an example of a software package that specializes in such problems is Arpack (see [Lehoucq et al., 1998]).

Vibration modes and gradient flow To illustrate the concept of eigenmodes of the Hessian of a deformation energy, we look at the vibrations of a mesh in a force field induced by the energy. For simplicity, we consider the case of free vibrations. In general, the dynamics of a time-dependent mesh $u(t)$ in the space $\mathcal{C}$ is governed by a system of non-linear second-order ODEs of the form

$$M\ddot{u}(t) = F(t, u(t), \dot{u}(t)),$$

see [Baraff and Witkin, 1998]. Here, the mass matrix M represents the physical mass of u and F represents the acting forces. We consider the force field that has E as its potential, i.e.,

$$F(t, u(t), \dot{u}(t)) = -\partial E_{u(t)}.$$

In the case of free vibrations, this is the only force. In a more general setting, we could include damping and exterior forces, see [Pentland and Williams, 1989]. The equations that govern the motion of a time-dependent mesh $u(t)$ during free vibration are

$$\mathrm{grad}_{u(t)}E + \ddot{u}(t) = 0, \tag{2.10}$$

where we use the definition of the gradient, Equation (2.6), to simplify the formula. Since we are interested in meshes u that are (arbitrarily) close to $\bar{u}$, we expand the force $\mathrm{grad}_u E$ into a Taylor series around $\bar{u}$. Using $\partial E_{\bar{u}} = 0$ (no force acts at the equilibrium $\bar{u}$) we get

$$\mathrm{grad}_u E = \mathrm{hess}_{\bar{u}}E(u - \bar{u}) + \mathcal{O}(\|u - \bar{u}\|^2). \tag{2.11}$$

Then, if we omit the second order term in (2.11) and combine (2.10) and (2.11), we get

$$\mathrm{hess}_{\bar{u}}E\, v(t) + \ddot{v}(t) = 0, \tag{2.12}$$

where $v(t) = u(t) - \bar{u}$. This is a system of second-order linear ODEs that are coupled by $\mathrm{hess}_{\bar{u}}E$. To solve the system we consider a normalized eigenbasis B of $\mathrm{hess}_{\bar{u}}E$. Written in such a basis, both matrices $\partial^2 E_{\bar{u}}$ and M are diagonal matrices and Equation (2.12) takes the form

$$\Lambda\, w(t) + \ddot{w}(t) = 0, \tag{2.13}$$

where w is the representation of v in the basis B and Λ is a diagonal matrix that contains the eigenvalues. The system (2.13) is decoupled and can be solved row by row. Each row describes an oscillation around $\bar{u}$ with frequency ω_i in the direction of the eigenmode Φ_i corresponding to the eigenvalue ω_i^2. This means, that the eigenmodes of problem (2.9) describe the vibration modes of the mesh $\bar{u}$ (with respect to the deformation energy E). The vibrations of a physical system are usually not free, but are affected by damping forces. Common models for such forces are Rayleigh damping, see [Hauser et al., 2003], and, even simpler, mass damping, see [Pentland and Williams, 1989]. We would like to mention that if Rayleigh (or mass) damping forces are added to the system, it still has the same vibration modes, see [Hauser et al., 2003].

Another possibility to interpret the eigenmodes of the Hessian is to consider the gradient flow governed by the energy E that is given by the system

$$\mathrm{grad}_{u(t)}E + \dot{u}(t) = 0 \tag{2.14}$$

of first-order ODEs. It describes the evolution of an object in a velocity field given by the negative gradient of the energy. Initial data for the equation is a position of the mesh $u(t)$ for some time t_0. Using Equation (2.11), we derive the linearized gradient flow

$$\mathrm{hess}_{\bar{u}} E \, v(t) + \dot{v}(t) = 0. \tag{2.15}$$

Analogously to Equation (2.12), this system can be decoupled employing the orthogonality property of the eigenmodes of $\mathrm{hess}_{\bar{u}} E$.

Chapter 3

Model reduction of nonlinear dynamical systems

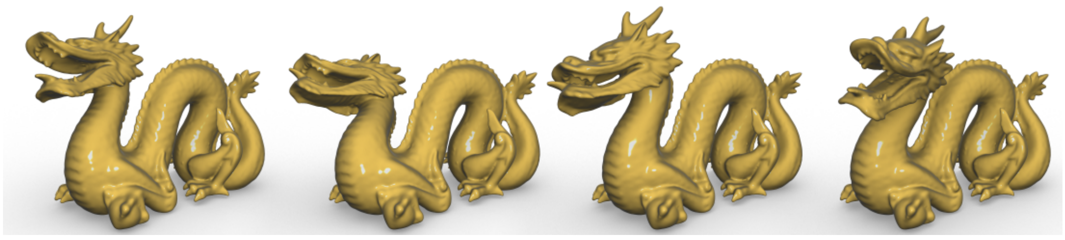


Figure 3.1: Reduced simulation of a geometrically nonlinear deformable object with 92k tetrahedra computed at over 120Hz after about four minutes of preprocessing (see Table 3.3).

3.1 Introduction

Methods for real-time simulations of deformable objects based on model reduction have received much attention in recent years. These schemes construct a low-dimensional approximation of the dynamical system underlying a simulation and thereby achieve a runtime that depends only on the complexity of the low-dimensional system. We focus on two problems of reduced nonlinear systems: the *subspace construction* and the *efficient approximation of the reduced forces*. We propose new techniques for both problems aiming at accelerating the construction of the approximate dynamical system, increasing the accuracy of the approximation, and simplifying their implementation. Based on this, we implement schemes for real-time simulations of deformable objects in Chapter 4 and for deformation-based editing of triangular or tetrahedral meshes in Chapter 5. Beyond these two applications, the developed techniques are potentially useful for other applications including the acceleration of large simulations and the reduction of constrained spacetime optimization problems, e.g., for motion design.

Subspace construction Subspace construction based on linear modal analysis has become standard practice for the dimension reduction of linear second-order dynamical systems. However, for nonlinear systems, such a basis cannot capture the effects of the nonlinearities, which for the simulation of deformable objects leads to artifacts for large deformations. We propose a simple, yet effective, technique for extending modal bases and demonstrate that the resulting subspaces can better represent the nonlinear behavior of deformable objects. The idea is to use linear transformations of $\mathbb{R}^3$ to create new basis vectors. In contrast to common usage, we do not apply the linear transformations to deform the geometry directly, but to "deform" the linear vibration modes. The resulting new basis vectors depend not only on the geometry but also on the material properties of the deformable object. Using this strategy, modal bases are extended in such a way that the spanned subspaces better approximate large deformations. In our experiments, we found the resulting effect comparable to that achieved by adding modal derivatives to linear modal bases. Benefits of the proposed technique are that the construction is fast and simple to implement.

Approximation of reduced forces In addition to dimension reduction, the real-time simulations of deformable objects require a scheme for efficiently approximating the nonlinear reduced forces. Here, we consider two approaches to the resolution-independent approximation of reduced forces: one that follows the *optimized cubature* introduced by An et al. [2008], and one that is based on coarsening the discrete geometry.

Typically, interior forces of a discrete deformable object can be written as a sum whose summands depend only on the deformation of a local neighborhood of the object, e.g., a triangle or a tetrahedron. The idea of using a cubature-based approximation is to exploit the correlations between these summands. The dimension reduction restricts the system to a small number of degrees of freedom, which in turn strengthens the correlations. The strategy is to select a small number of summands and to approximate the reduced forces by a linear combination of these summands. The subset and weights are determined through an optimization procedure in which the approximation error on an automatically generated set of training poses is minimized. This is a constrained best subset selection problem. In Section 3.3.1, we devise a new scheme for efficiently solving this problem, which is based on recent advances in the field of sparse approximation. Our strategy for solving the subset selection problem is substantially different from that used in [An et al., 2008]. They use a greedy strategy that iteratively constructs the selection set by successively adding one entity per iteration. In contrast, our scheme constructs a complete selection set in the first iteration and the whole selection set can be changed in subsequent iterations. We demonstrate in a number of examples that our scheme can produce a significantly smaller approximation error at lower computational costs and is able to achieve a given training error with a smaller selection set.

In addition to optimized cubature, we propose a scheme for force approximation that is based on a second reduced shape space for a low-resolution version of the object. By construction, the two reduced shape spaces are isomorphic and we can use the isomorphism to pull the energy from the shape space of the simplified mesh to the shape space of the full mesh. Thus, the coarse mesh provides means to exploit the spatially local coherence, while optimized cubature exploits the coherence in the global motion of the object.

Related work

Using subspaces constructed from linear vibration modes to accelerate the integration of linear second-order dynamical systems is a standard technique with a long tradition [Pentland and Williams, 1989]. Still, the question of how this technique can be generalized to nonlinear systems is an active area of research. One strategy is to compute the vibration modes around several states and to use the span of the union of these modes. The drawback of this approach is the high computational cost for solving several eigenvalue problems. An alternative approach is to enrich the modal basis with *modal derivatives* [Idelsohn and Cardona, 1985; Barbič and James, 2005; Hildebrandt et al., 2011; Tiso, 2011]. Roughly speaking, a modal derivative, which is computed from a pair of modes, describes how one mode changes when the object is deformed in the direction of the other mode. A derivation of the modal derivatives as well as a discussion of their relation to energy descent directions will be covered in Section 3.2.1. Another very common approach for creating a basis for nonlinear systems is to construct reduced spaces based on a principal component analysis of a set of observations [Krysl et al., 2001].

After dimension reduction, the cost for evaluating the nonlinear interior forces of a deformable object is still high as it requires computing and projecting the unreduced forces. *Optimized cubatures* have been successfully applied for approximating the interior forces of different types of hyperelastic materials for elastic solids [An et al., 2008] and thin shells [Chadwick et al., 2009]. Recently, they were also used for reduced fluid re-simulations [Kim and Delaney, 2013]. Computing the cubature points requires solving a complex optimization problem: a best subset selection problem. To solve the problem, current schemes use a greedy strategy that incrementally builds the selection set. An alternative to force approximation is the exact evaluation of the reduced forces. For linear materials, e.g., the St. Venant–Kirchhoff model of elastic solids, the forces are cubic polynomials on the shape space. In this case, the coefficients of the restriction of the polynomials to the reduced space can be precomputed [Barbič and James, 2005]. This yields an exact representation of the forces in the reduced space and evaluation costs that depend only on the size of the subspace. However, the number of coefficients to be precomputed and evaluated at runtime grows quartically with the dimension of the reduced space.

In addition to real-time simulations, model reduction has been used to accelerate large simulations [Kim and James, 2009] and for controlling the motion of deformable objects [Barbič and Popović, 2008; Barbič et al., 2009, 2012; Hildebrandt et al., 2012a] as well as characters [Safonova et al., 2004] and fluids [Treuille et al., 2006; Wicke et al., 2009]. Moreover, in [Hahn et al., 2012], simulations in reduced spaces obtained from animators' rigs were considered with the goal of simplifying the integration of simulation into the traditional animation pipeline. Subset selection based on training poses has also been used for facial articulation and global illumination by Meyer and Anderson [2007].

Alternative approaches for real-time simulations of deformable objects in a reduced space are to use *modal warping* [Choi and Ko, 2005] and *rotation strain coordinates* [Huang et al., 2011]. These schemes integrate a linearized system in modal coordinates and *warp* the solutions to counteract artifacts produced by the linearization.

In geometry processing, deformable objects are used for editing shapes. In such a *deformation-based editing* system (see [Botsch and Sorkine, 2008; Botsch et al., 2010] and references therein) a user can select parts of a geometry as handles and translate and rotate them in space. The system automatically deforms the shape so that the handles interpolate or approximate the specified positions. This is done by computing static equilibrium states of the deformable object subject to constraints or external forces that represent the user's input. A major advantage of deformation-based editing over traditional modeling techniques, like NURBS or subdivision surfaces, is that many complex editing tasks can be described by few constraints. This allows for efficient and simple click-and-drag user interfaces. To obtain an interactive editing system for larger models, methods that accelerate the computation based on space deformation [Sumner et al., 2007; Botsch et al., 2007; Ben-Chen et al., 2009], skinning [Jacobson et al., 2012], and model reduction [Hildebrandt et al., 2011] have been proposed. Recently, reduced deformable objects were used to create a system for modeling simulation-ready plants [Zhao and Barbič, 2013]. The deformation based editing system we implemented to test the proposed subspaces and force approximation is detailed in Chapter 5.

3.2 Subspace construction

Simulating large-deformation dynamics in the full, unreduced shape space $\mathcal{C}$ of an object leads to nonlinear and high-dimensional systems that are challenging to solve even if interactive rates are not mandatory. Although deformable objects can, in general, assume arbitrary configurations in $\mathcal{C}$, their motion tends to be confined to a low-dimensional subspace of certain characteristic, low-energy states. Our goal is therefore to find an affine subspace V of $\mathcal{C}$ that is both low-dimensional and able to capture natural deformations of the object. Furthermore, we are interested in subspace constructions that require no advance knowledge of run-time forcing or pre-simulation of the unreduced system and hence can be performed in an automatic preprocess. The choice of subspace depends on the geometry, boundary conditions and the material of the object. An established method in engineering is to use low-frequency vibration modes (see Section 2.3) at the rest state of the object and is known as *linear modal analysis*. The motivation to use these modes is that we are searching for displacements in $\mathcal{C}$ with the least resistance to deformation, and since the internal forces vanish at the rest state the low-frequency eigenmodes of the Hessian point into the directions in $\mathcal{C}$ that (locally) cause the least increase in elastic strain energy of the object. Linear modal analysis provides quality deformation subspaces for small deformations as is typical in sound simulation. However, for deformable objects undergoing large deformations, these reduced spaces cannot capture the nonlinearities. This is reflected in a large increase of strain energy (and visible artifacts) when such a state is projected onto the subspace. The effect is illustrated in Figure 3.7, in which editing results computed in different subspaces are shown. A strategy to remedy this problem is to enrich the basis of linear vibration modes by a second set of deformations that compensate for the artifacts.

Let us assume that $\bar{u} \in \mathcal{C}$ is a static and stable equilibrium for some constant external force. If the object is excited (by a small enough stimulus), it vibrates around $\bar{u}$. The vibration modes Φ_i and frequencies ω_i can be computed by solving the sparse generalized eigenvalue problem 2.9. The modal bases we use as a starting point for our construction of a reduced space consists of the 15-20 eigenmodes with the lowest frequencies. To compute the modes, we use a shift-and-invert Lanczos scheme, see [Saad, 1992].

3.2.1 Modal derivatives

Let V be the linear span of the union of two sets V_1 and V_2 of vectors, where V_1 contains low-frequency eigenmodes of the Hessian of a deformation energy E at the rest state $\bar{u}$. To extend the reduced space spanned by V_1, we collect vectors that point into energy descent directions at points in $\bar{u}$+Span(V_1) in the set V_2. Assume we are at some point u in $\bar{u}$+Span(V_1). Then, an effective descent direction in the full space $\mathcal{C}$ would be the Newton direction $-\partial^2 E_u^{-1} \partial E_u$ at u, which is the direction in which a Newton solver would perform a line search.

To derive an approximation for the energy descent directions, we consider the Taylor series of the Newton direction around $\bar{u}$ given by

$$-\partial^2 E_u^{-1}\partial E_u = -\partial^2 E_{\bar{u}}^{-1}\partial E_{\bar{u}} - \partial(\partial^2 E_{\bar{u}}^{-1}\partial E_{\bar{u}})(v) - \frac{1}{2}\partial^2(\partial^2 E_{\bar{u}}^{-1}\partial E_{\bar{u}})(v,v) + \ldots \quad (3.1)$$

where $v = u - \bar{u}$. Since $\partial E_{\bar{u}} = 0$ at the rest state ($\bar{u}$ is a minimum), the first term of the right-hand side vanishes, and we have

$$\partial(\partial^2 E_{\bar{u}}^{-1}\partial E_{\bar{u}}) = \partial^2 E_{\bar{u}}^{-1}\partial^2 E_{\bar{u}} = \boldsymbol{1},$$

which shows that the second term reduces to $-v$. Indeed, $\partial^2 E_u^{-1}\partial^2 E_u$ equals the identity matrix at all u, which implies

$$0 = \partial(\partial^2 E_{\bar{u}}^{-1}\partial^2 E_{\bar{u}}) = \partial(\partial^2 E_{\bar{u}}^{-1})\partial^2 E_{\bar{u}} + \partial^2 E_{\bar{u}}^{-1}\partial^3 E_{\bar{u}},$$

where $\partial^3 E_{\bar{u}}(\cdot,\cdot,\cdot)$ is the third-rank tensor containing the third partial derivatives of E at $\bar{u}$. Using these observations, we can show

$$\begin{aligned}\partial^2(\partial^2 E_{\bar{u}}^{-1}\partial E_{\bar{u}}) &= \partial(\partial(\partial^2 E_{\bar{u}}^{-1})\partial E_{\bar{u}} + \partial^2 E_{\bar{u}}^{-1}\partial^2 E_{\bar{u}})\\ &= \partial(\partial^2 E_{\bar{u}}^{-1})\partial^2 E_{\bar{u}}\\ &= -\partial^2 E_{\bar{u}}^{-1}\partial^3 E_{\bar{u}}.\end{aligned}$$

We can show now that the Taylor series of the Newton direction indeed satisfies

$$-\partial^2 E_u^{-1}\partial E_u = -v + \frac{1}{2}\partial^2 E_{\bar{u}}^{-1}\partial^3 E_{\bar{u}}(v,v) + \mathcal{O}(\|v\|^3) \tag{3.2}$$

where $\partial^3 E_{\bar{u}}(v,v)$ stands for the vector we obtain by plugging in v twice into $\partial^3 E_{\bar{u}}(\cdot,\cdot,\cdot)$ (and transpose the resulting linear form).

Based on the insights of Equation (3.2), we define vectors Ψ_{ij} as the solutions to the equation

$$\partial^2 E_{\bar{u}}\Psi_{ij} = \partial^3 E_{\bar{u}}(\Phi_i,\Phi_j). \tag{3.3}$$

Since $\partial^3 E_{\bar{u}}(\cdot,\cdot,\cdot)$ is symmetric, we have $\Psi_{ij} = \Psi_{ji}$. Furthermore, if the deformable object is not equality constrained, the first six linear modes span the (linearized) rigid motions, and the translations have vanishing derivatives. Hence, from the first r linear modes, we can construct at most $((r-3)^2 + (r-3))/2$ and $(r^2+r)/2$ linear independent vectors Ψ_{ij} by solving Equation (3.3) for unconstrained and equality constrained objects, respectively. We collect all the Ψ_{ij} obtained from pairs of linear modes from V_1 in the set V_2. Then, at every point $u \in \bar{u}+\mathrm{Span}(V_1)$ the vector $\partial^2 E_{\bar{u}}^{-1}\partial^3 E_{\bar{u}}(v,v)$, which appears in Equation (3.2), is in $\mathrm{Span}(V_2)$. Therefore the affine space spanned by V_1 and V_2 contains an approximation up to the third order in $\|u-\bar{u}\|$ of the Newton direction (3.2) at every point $u \in \bar{u}+\mathrm{Span}(V_1)$. For completeness, we would like to mention that Equations (3.2) and (3.3) are only defined up to the kernel of $\partial^2 E_{\bar{u}}$, which for most deformation energies of unconstrained objects are the linearized rigid motions. However, since the kernel of $\partial^2 E_{\bar{u}}$ is contained in $\mathrm{Span}(V_1)$, the constructed reduced space is independent of the choice of vectors Ψ_{ij} that solve Equation (3.3).

Furthermore, we would like to remark that the construction of the vectors Ψ_{ij} is analogous to the simplified modal derivatives introduced by Idelsohn and Cardona [1985], though their formulation does not use a potential energy and is derived by differentiating the generalized eigenvalue problem (2.9). For this reason we call the Ψ_{ij} the modal derivatives. Examples of vibration modes and modal derivatives of a simple shape (the bar) and of a complex shape (the Chinese dragon) are shown in Figures 3.2 and 5.2. A physical interpretation

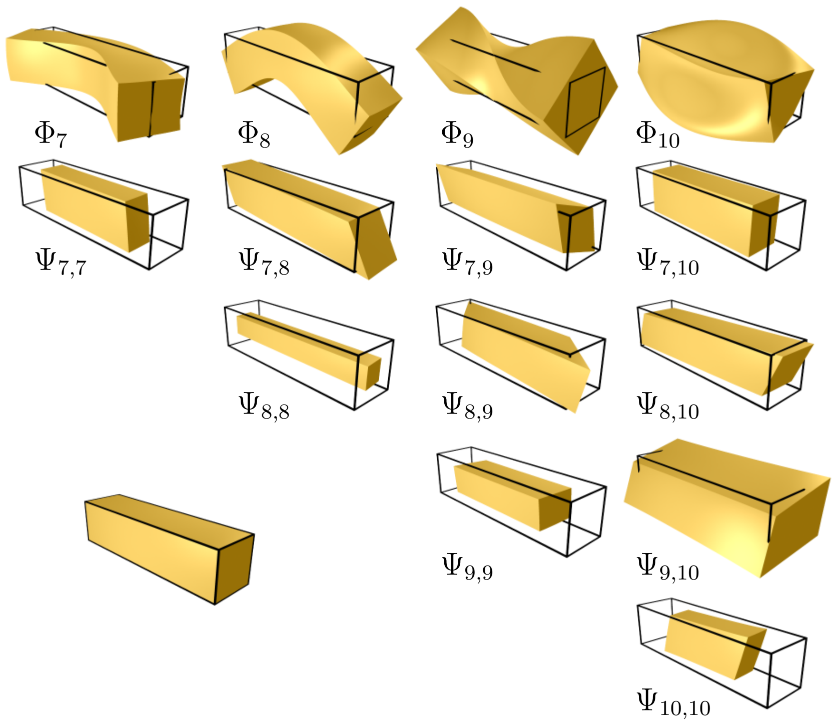


Figure 3.2: Vibration modes and modal derivatives of the bar are shown: vibration modes in the top row and corresponding modal derivatives below. We leave out the first six modes because they span the linearized rigid motions.

of the modal derivatives is that Ψ_{ij} describes how the mode Φ_i changes (in the first order) when the object is deformed in the direction Φ_j.

Deriving formulae for the partial derivatives of a deformation energy E by hand can be laborious and time consuming. Thus, automatic differentiation libraries can help immensely to reduce the effort needed for implementation and debugging. While we employ an implementation for St. Venant–Kirchhoff deformable solids that provides all necessary quantities, we use the *ADOL-C* [Griewank et al., 1996] library to compute the second and third order partial derivatives for the *Discrete Shells* energy. The memory costs for the tensor $\partial^3 E_{\bar{u}}(\cdot,\cdot,\cdot)$ become prohibitive for highly detailed models. However, we do not need to compute the full tensor; only the restriction of it to $\mathrm{Span}(V_1)$. Furthermore, the energies used in our experiments are defined as sums over components which allows us to save on the main memory by directly reducing the third derivatives of the individual summands. To compute the modal derivatives, we need to solve the linear equation (3.3) several times. Since the matrix $\partial^2 E_{\bar{u}}$ is the same in all these linear systems, it is sufficient to compute a sparse factorization of the matrix once and to use it to solve all the systems.

To investigate the ability to well-approximate large, nonlinear deformations in reduced spaces from linear vibration modes and modal derivatives, we compute deformations of a bar model in subspaces of varying size. Figure 3.3 indicates that even remarkably low-dimensional subspaces contain substantial nonlinear content to prescribe large deformations, e.g., the result in a 30-dimensional space is already visually close to the unreduced reference solution of the bar. However, the performance of the reduced spaces depends crucially on the strategy used to sample directions from $\mathrm{Span}(V_2)$ as the number of modal derivatives quickly becomes prohibitive. For example, Barbič and James [2005] provide an example of a 5-dimensional subspace that contains extreme twists of a bar by hand-picking "twist" linear modes and their derivatives.

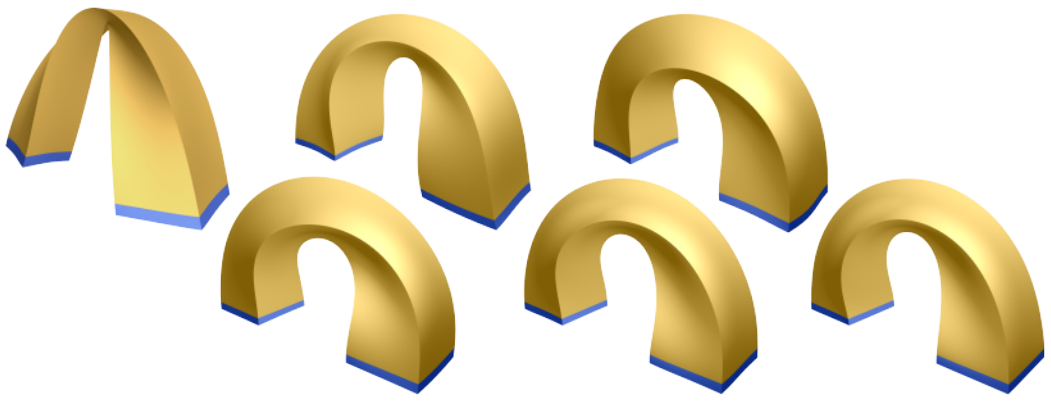


Figure 3.3: A comparison of the results produced with reduced spaces with varying sizes is shown. Number of linear modes and modal derivatives from left to right: (8,6), (10,20), (15,52), (20,110), (30,270), full space.

3.2.2 Extended modal bases

In this section we propose an alternative technique for extending modal bases that is simple, yet effective. The idea is to add new vectors to the basis that are obtained by applying certain linear transformations to the vectors in the modal basis. This provides the subspace with additional degrees of freedom that can compensate for artifacts which would appear in the space spanned by the modal basis. Our experiments show that the effect we achieve through this construction is comparable to that of enriching a modal basis with modal derivatives.

Extended modal bases In the continuous setting, an element of the modal basis is a vector field that assigns a vector in $\mathbb{R}^3$ to every point of the deformable object (see Figure 2.2). Similarly, we can interpret a modal vector $\Phi_i \in \mathbb{R}^{3n}$ as a discrete vector field consisting of n vectors in $\mathbb{R}^3$, e.g., an $\mathbb{R}^3$-vector at every node of a triangle or tet mesh. To construct a new vector field from a mode Φ_i, we consider a linear map on $\mathbb{R}^3$ and apply it (the same linear map) to all n $\mathbb{R}^3$-vectors of the mode. For example, we fix an angle and an axis in $\mathbb{R}^3$ and rotate all the n vectors by the same angle about the same axis.

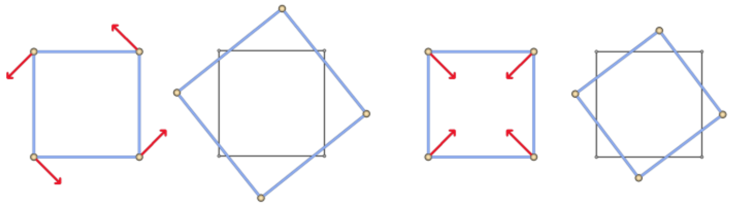


Figure 3.4: Linearization of rotations (*first image*) introduce artifacts (*second image*). By rotating the vectors of the linearized rotation by 90° (*third image*), one obtains a vector field that can be used to compensate for the artifacts (*fourth image*).

Let us look at the simple example shown in Figure 3.4. It illustrates that extending a basis using this construction can help to compensate for linearization artifacts. We consider a $2D$ object in the plane (a square in the figure) and a vector field pointing into the direction of a linearized rotation (around some center c). Moving the vertices of the object along the direction of the vector field, rotates the object, but at the same time produces an artifact: the volume of the object is increased. Now we construct a second vector field by rotating every vector of the linearized rotation by 90°. Then, any rotation of the object around c can be exactly described as a linear combination of the two vector fields.

Motivated by such examples, we propose using this construction for extending modal bases. If a modal basis consists of r modes, then the extended basis can have at most $9r$ vectors since there are only nine linear independent linear maps on $\mathbb{R}^3$. The scheme for extending a modal basis is outlined in Algorithm 1. In the first step, $9r$ vectors are constructed based on the r modes. Then, the (modified) Gram–Schmidt process is applied to get an orthonormal basis of the corresponding reduced space. A basis in the space of linear maps on $\mathbb{R}^3$ is formed by the nine matrices B_{kl} that have only one non-vanishing entry, which takes the value 1. By applying each of the matrices B_{kl} to the n $\mathbb{R}^3$-vectors of a mode, we obtain the desired nine new $\mathbb{R}^{3n}$-vectors. The construction of all $9r$ vectors can be implemented in four nested for-loops. The first loop (index i in Algorithm 1) iterates over all modes, the second (index j) over all n $\mathbb{R}^3$-vectors of the ith mode, and the third and fourth (indices k and l) over the 9 matrices B_{kl}. The body of the inner loop multiplies the matrix B_{kl} by the jth $\mathbb{R}^3$-vector of the ith mode and inserts the result into the corresponding vector of the extended basis. This requires only one assignment since after a multiplication with B_{kl}, a $\mathbb{R}^3$-vector has only one non-zero component. The construction of the $9r$ vectors requires only $9rn$ assignments and, therefore, is fast compared to the construction of the modal basis. In Algorithm 1, the $\mathbb{R}^{3n}$-vectors successively list the x-coordinates the n $\mathbb{R}^3$-vectors, then the n y-coordinates, and finally the z-coordinates.

If the object is unconstrained, the first six eigenmodes span the translations and the linearized rotations of the object. In this case, we modify the construction. In the first step, we apply the extension strategy to the modes $\{\Phi_7, \Phi_8, \dots, \Phi_r\}$. In the second step, we add 12 basis vectors that span all affine transformations, i.e., linear transformations and translations, of the object to the basis. Together, the extended modal basis has a maximum of $9r$ (constrained object) or $9(r-6)+12$ (unconstrained object) vectors. The number of elements in the basis can be smaller if the generated vectors are linearly dependent. Examples of vectors in the extended modal basis are shown in Figure 3.5.

There are also alternative schemes for computing a basis that spans the same reduced space. For example, one could use another basis of the space of linear

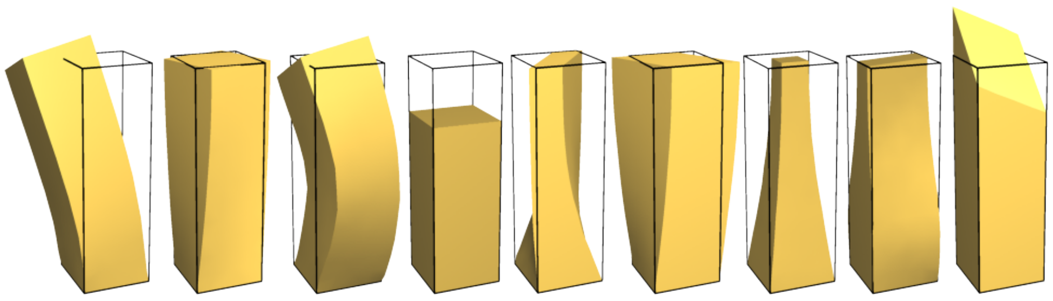


Figure 3.5: Three linear modes (*left*) and two vectors obtained through our basis extension for each of the three modes are shown.

```
Algorithm: Extend Modal Basis
Data: $\Phi_1, \Phi_2, \dots, \Phi_r \in \mathbb{R}^{3n}$
Result: $e_1, e_2, \dots, e_d \in \mathbb{R}^{3n}$
allocate $b_1, b_2, \dots, b_{9r} = 0 \in \mathbb{R}^{3n}$;
for $i = 1, 2, \dots, r$ do
    for $j = 1, 2, \dots, n$ do
        for $k = 1, 2, 3$ do
            for $l = 1, 2, 3$ do
                $b_{9(i-1)+3(k-1)+l}[(l-1)n + j] = \Phi_i[(k-1)n + j]$;
            end
        end
    end
end
$\{e_i\} \leftarrow$ orthonormalize($\{b_i\}$);
```

Algorithm 1: Construction of the extended modal basis.

maps on $\mathbb{R}^3$ to construct the $9r$ vectors. We chose to use the basis B_{kl} because the sparsity of the matrices B_{kl} reduces the number of required arithmetic operations.

Most of our experiments consider homogeneous materials. Still, since our construction of a reduced space includes a modal basis, it is material-aware. Figure 3.6 shows an example of a deformation of an inhomogeneous block made from two different materials. The reduced space adapts to the inhomogeneous material. For comparison, we show the unreduced (reference) solution.

3.2.3 Results and comparison

To evaluate how well our extended modal bases can represent large deformations, we compare simulation and editing results obtained in different subspaces: modal bases, extended modal bases, and modal bases augmented with modal derivatives. In particular, we conducted the experiments using the schemes for real-time simulation and interactive modeling derived in Chapter 4 and 5, respectively. For the examples of static states shown in Figure 3.7 and the simulation of a dragon model shown in Figure 3.1, Table 3.1 lists the relative L^2-error, i.e., the L^2-norm of the difference of the reduced and the unreduced reference solutions divided by the area/volume of the rest state. The configurations that define the four static states are taken from [Botsch and Sorkine, 2008], where they were tested using different linear editing schemes,

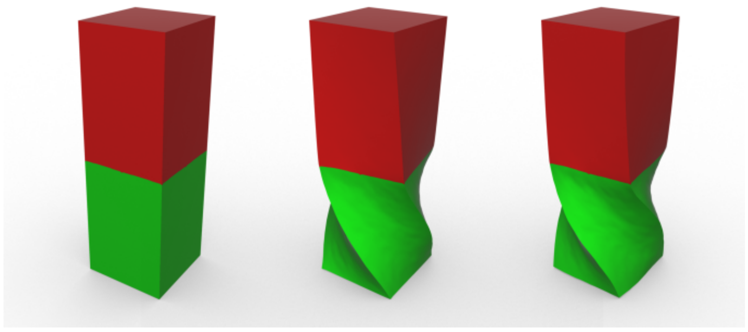


Figure 3.6: Deformations of an object with inhomogeneous material. Left: initial state with color-coding of Young's modulus (red and green denote values of 8×10^4 and 1×10^4, respectively); middle: unreduced solution; right: solution in a 138-dimensional subspace.

e.g., based on linearized thin shells. None of the linear schemes could deal with all four poses without developing visible artifacts.

Model	#v	d	Our	M. D.	Linear
Bar	6k	93	0.0109	0.0113	0.0851
Cactus	5k	93	0.0131	0.0057	0.0933
Cylinder	5k	93	0.0089	0.0080	0.0565
Plate	40k	93	0.0109	0.0136	0.0266
Dragon	26k	135	0.0170	0.0225	0.0440

Table 3.1: **Subspace Fidelity**. From left to right: number of vertices, subspace dimension, L^2-norms of the differences between the unreduced solution to those obtained in subspaces from our construction, modal derivatives, and linear modes only. See Figure 3.7 for shell examples and Figure 3.1 for snapshots from the simulation of the dragon.

For the comparisons, we used subspaces of the same dimension. In most cases, we used 15 eigenmodes to construct the extended modal basis. This yields 93- or 135-dimensional spaces depending on whether the object is constrained or unconstrained, see Section 3.2.2. For some unconstrained objects, we used 20 eigenmodes, resulting in 138-dimensional subspaces. To construct subspaces of equal dimension using modal derivatives, we start with the same number r of linear modes, compute all modal derivatives, and choose the required number of basis vectors from the span of the computed derivatives. For example, for an unconstrained object and $r = 15$, all modal derivatives are used.

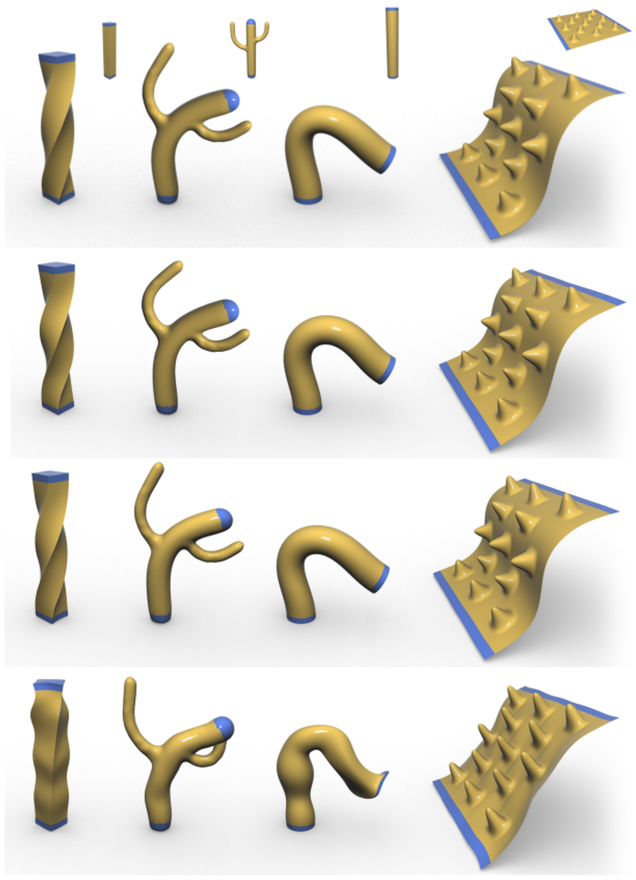


Figure 3.7: Comparison of deformations found in 93-dimensional subspaces. Reference solution in unreduced space and rest states as superscripts (top row), our reduced space (second row), linear modes and modal derivatives (third row), only linear modes (bottom row).

For all examples, our construction as well as the spaces enriched with modal derivatives yield better approximations of the unreduced reference solution than the spaces constructed only from linear modes. The results obtained with our construction and with modal derivatives are of comparable approximation quality. We want to remark that this comparison is limited to hyperelastic, isotropic, and homogeneous materials and the ratio of linear modes and modal derivatives described above.

We found the results promising since compared to modal derivatives the proposed basis extension from Section 3.2.2 is easier to implement and faster to execute. To illustrate the lowered computational cost, we list computation times for the construction of the linear modes, our extended modes, and the

Model	#v	d (r)	Φ	Our	M. D.
Chinese Dragon	130k	138 (20)	102s	8s	407s
Dinosaur	56k	135 (15)	38s	3s	169s
Elephant	25k	135 (15)	17s	1s	81s
Dragon	26k	135 (15)	9s	1s	28s
Vertebrae	16k	135 (15)	7s	1s	21s

Table 3.2: **Subspace construction times** measured on a 2012 MacBook Pro Retina 2.6GHz. From left to right: number of vertices, subspace dimension (number of used vibration modes), time for computation of vibration modes, our extended modes and modal derivatives.

modal derivatives in Table 3.2. Of course, these times may vary depending on what solvers are used for the computation of the eigenmodes and the modal derivatives. To compute the eigenmodes, we use our own implementation of the shift-and-invert Lanczos method. Since the computation of the modal derivatives involves solving a number of linear systems with the same matrix, we use a sparse direct solver for the systems (reusing the factorization). Additionally, we set up the right-hand sides of the systems in parallel.

3.3 Approximation of reduced internal forces

Applying dimensional model reduction to complex, nonlinear systems can significantly improve the runtime performance. However, evaluating the reduced internal forces $\bar{F}^{int}$ by computing the unreduced forces F^{int} and projecting it into the subspace becomes expensive when the mesh size increases. To make the computational costs independent of the size of the full problem, an efficient evaluation of the reduced internal forces is needed. In the remainder of this chapter, we present two techniques that provide approximations of the reduced internal forces that are independent of the resolution of the unreduced system and thus hold the promise of superior runtime performance.

3.3.1 Optimized cubature

The *optimized cubature*, proposed by An et al. [2008], constructs an approximation of the reduced forces based on a best subset selection problem. We follow this approach and set up a similar optimization problem in Equation (3.8).

However, we propose a completely different strategy to solve the best subset selection problem in Section 3.3.1. A comparison of running times and accuracy of the proposed scheme and the greedy solver used in [An et al., 2008] on a set of different models and parameter settings can be found in Table 3.3.

We assume that the internal forces can be written as a sum of m summands

$$F^{int}(u) = \sum_{i=1}^{m} f_i^{int}(u), \tag{3.4}$$

where any f_i^{int} depends only on a local neighborhood, e.g., the four vertices for a tetrahedron. A wide range of discretizations, e.g., finite elements, represent $F^{int}(u)$ in such a form.

Let $\{b_i\}_{i\in\{1,2,...,d\}}$ be a basis of a d-dimensional subspace V of $\mathcal{C} \simeq \mathbb{R}^{3n}$, then the matrix $U = [b_1, b_2, ..., b_d] \in \mathbb{R}^{3n\times d}$ maps the reduced coordinates q in V onto the corresponding displacement vector $u \in \mathbb{R}^{3n}$,

$$u = U\,q. \tag{3.5}$$

Combining Equations (3.4) and (3.5) the reduced force $\bar{F}^{int}$ can be expressed in terms of the reduced coordinates

$$\bar{F}^{int}(q) = \sum_{i=1}^{m} U^T f_i^{int}(Uq) = \sum_{i=1}^{m} \bar{f}_i^{int}(q),$$

where $\bar{f}_i^{int}(q) = U^T f_i^{int}(Uq)$. The idea behind the approximation of $\bar{F}^{int}$ is to exploit the correlations between the $\bar{f}_i^{int}$s. We carefully select a subset of the $\bar{f}_i^{int}$s and assign a non-negative weight w_i to any selected $\bar{f}_i^{int}$. Then, the approximate reduced force is a linear combination of the selected $\bar{f}_i^{int}$s. Formally, we store the weights and indices of the selected $\bar{f}_i^{int}$s in a sparse m-dimensional vector w. An nonzero entry w_i in w, means that $\bar{f}_i^{int}$ is in the selection set and has the weight w_i. Then, we define

$$\bar{F}^{int}(q) \approx \tilde{F}_w^{int}(q) = \sum_{i\in \mathrm{supp}(w)} w_i \bar{f}_i^{int}(q), \tag{3.6}$$

where supp(w) denotes the support w, e.g., the set of indices with non-zero entries of w. The motivation for restricting the weights to be positive is that we want to preserve the structure of F^{int} and reduce the potential of overfitting the model function to the training data.

Determining a good subset of components and weights, is a non-trivial task for general, nonlinear materials and geometrically complex objects. We deal

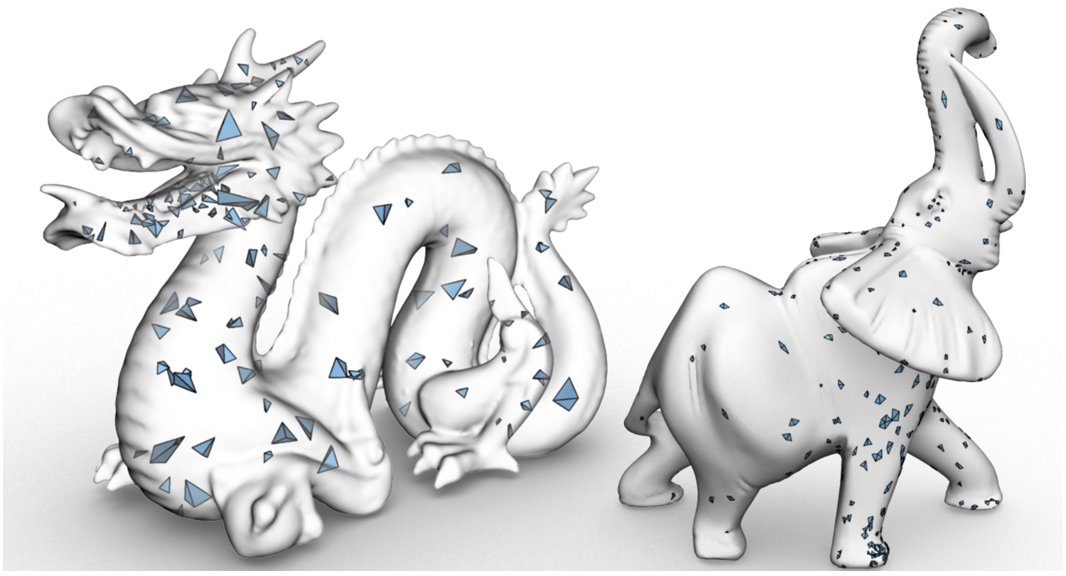


Figure 3.8: Optimal set of components used for the approximation of the reduced forces. Left: tets in a solid model of the dragon; right: edge flaps in the shell model of the elephant (see Table 3.3).

with this problem by using a data-driven approach that *learns* the best subset and weights from a set of T automatically generated training poses (and corresponding forces). Let $\{q_t\}_{t=1,\dots,T}$ be the set of training poses. Then, the relative fitting error of $\tilde{F}_w^{int}$ to the training forces is

$$\epsilon(w) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\frac{\epsilon_t(w)^2}{\left\|\bar{F}^{int}(q_t)\right\|^2}}, \tag{3.7}$$

where $\epsilon_t(w) = \|\bar{F}^{int}(q_t) - \tilde{F}_w^{int}(q_t)\|$ is the absolute approximation error to $\bar{F}^{int}(q_t)$. The problem of finding a sparse approximation can now be stated in terms of an optimization problem in which we try to minimize the fitting error

$$\min_{w} \epsilon(w)^2 \quad \text{subject to} \quad w \geq 0, |\text{supp}(w)| \leq s, \tag{3.8}$$

where $|\text{supp}(w)|$ denotes the cardinality of $\text{supp}(w)$. As $\epsilon(w)^2$ depends quadratically on w, (3.8) is a best subset selection problem with non-negativity constraints. This can be seen by reformulating the error as $\epsilon(w) = \frac{1}{\sqrt{T}}\|Aw - b\|$, where $A \in \mathbb{R}^{dT \times m}$ and $b \in \mathbb{R}^{dT}$ are defined by

$$A = \begin{bmatrix} \frac{\bar{f}_1^{int}(q_1)}{\|\bar{F}^{int}(q_1)\|} & \cdots & \frac{\bar{f}_m^{int}(q_1)}{\|\bar{F}^{int}(q_1)\|} \\ \vdots & \ddots & \vdots \\ \frac{\bar{f}_1^{int}(q_T)}{\|\bar{F}^{int}(q_T)\|} & \cdots & \frac{\bar{f}_m^{int}(q_T)}{\|\bar{F}^{int}(q_T)\|} \end{bmatrix}, \; b = \begin{bmatrix} \frac{\bar{F}^{int}(q_1)}{\|\bar{F}^{int}(q_1)\|} \\ \vdots \\ \frac{\bar{F}^{int}(q_T)}{\|\bar{F}^{int}(q_T)\|} \end{bmatrix}.$$

In the remainder of this section, we provide details of the individual steps needed to estimate an optimal subset, i.e., a solution to (3.8). The complete optimization scheme is summarized in Algorithm 2.

```
**Algorithm:** NN-HTP
**Data**: $f$, initial guess $w^0$, tolerance $\tau$
**Result**: optimal subset $w^*$
$w^0 \leftarrow \mathcal{H}_s^+(w^0)$;
**for** $i = 1, 2, \dots$ **do**
    (Lazy) evaluate $\nabla f(w^i)$;
    **if** $\|\nabla f(w^i)\| \le \tau$ **then return** $w^i$;
    Determine $\mathcal{S}^i$ via (3.10) using $\mathcal{H}_s^+$;
    Determine $\mu^i$ via (3.11);
    $w^{i+1} \leftarrow \mathcal{H}_s^+(w^i - \mu^i \nabla_{\mathcal{S}^i} f(w^i))$;
    $\mathcal{X}^{i+1} \leftarrow \mathrm{supp}(w^{i+1})$;
    **if** $\mathcal{X}^{i+1} = \mathcal{X}^i$ **then return** $w^i$;
    $w^{i+1} \leftarrow \arg\min_{\{v | \mathrm{supp}(v) \subseteq \mathcal{X}^{i+1}, v \ge 0\}} f(v)$;
**end**
```

Algorithm 2: Non-negativity-constrained hard thresholding pursuit.

Optimal subsets via sparse approximation Best subset selection problems are ubiquitous in a variety of disciplines and much research has been devoted to their efficient solution. In particular, their presence in the field of compressed sensing received significant attention and triggered further advancements, one of them being the *normalized iterative hard thresholding* (NIHT) algorithm [Blumensath and Davies, 2010].

Let $f(w) = T\epsilon(w)^2$ be the objective function. Then, an iteration of the NIHT algorithm is given by

$$w^{i+1} = \mathcal{H}_s(w^i - \mu^i \nabla f(w^i)), \tag{3.9}$$

where $\nabla f(w^i) = 2A^T(Aw^i - b)$ and $\mathcal{H}_s$ is a combinatorial projection that sets all but the s largest (in magnitude) entries of a vector to zero. Increased stability over the traditional hard thresholding algorithm is due to choosing the step length μ^i adaptively at every iteration. Before we proceed to the estimation of μ^i, we remark that the support of w^{i+1} will be necessarily contained in the $2s$-element set

$$\mathcal{S}^i = \operatorname{supp}(w^i) \cup \operatorname{supp}(\mathcal{H}_s(-\nabla_{\mathcal{I}\setminus\operatorname{supp}(w^i)} f(w^i))). \tag{3.10}$$

Here $\nabla_{\mathcal{K}} f$ denotes the gradient of f with all entries not in the set $\mathcal{K}$ set to zero and $\mathcal{I} = \{1, \dots, m\}$. Based on S^i, the step length is computed to be

$$\mu^i = \frac{\|\nabla_{\mathcal{S}^i} f(w^i)\|^2}{\|A\nabla_{\mathcal{S}^i} f(w^i)\|^2}. \tag{3.11}$$

Despite its simplicity, under certain conditions, NIHT has a linear rate of convergence and can approximate sparse solutions with near optimal accuracy. However, NIHT can be further accelerated (see [Cevher, 2011]). One of the most effective acceleration methods is the *hard thresholding pursuit (HTP)* by [Foucart, 2011] which adds a second step to the NIHT algorithm. In every iteration, the projected gradient descent step in (3.9) is preceded by the low-dimensional minimization

$$w^{*,i+1} = \arg\min_{\{v|\operatorname{supp}(v)\subseteq\operatorname{supp}(w^{i+1})\}} f(v). \tag{3.12}$$

We propose a novel algorithm, called *non-negativity-constrained HTP* (NN-HTP) that extends HTP by an efficient treatment of non-negativity constraints. First, we define a new projection operator $\mathcal{H}_s^+(w)$ that projects w to the new feasible region, i.e., ignores all negative entries and keeps only the s largest ones. To satisfy the constraints, we replace the projection operator in (3.9) by $\mathcal{H}_s^+$. Accordingly, we account for the adjustments in (3.9) by applying $\mathcal{H}_s^+$ in (3.10) to ensure the correct support estimation.

The second step is to incorporate the non-negativity constraints in the s-dimensional minimization (3.12). Let $A' \in \mathbb{R}^{dT\times s}$ denote the submatrix of A containing only the columns corresponding to elements in $\operatorname{supp}(w^i)$ and $w' \in \mathbb{R}^s$ the associated subvector of w^i. Then, we can determine the minimizer $w^{*,i+1}$ by solving the non-negative least squares (NNLS) problem $A'w' = b$ with $w' \geq 0$. In our experiments, these NNLS problems proved to be too stiff for projection-based solvers like L-BFGS-B [Morales and Nocedal, 2011]. Therefore, we employ the efficient FNNLS solver by Bro and De Jong [1997], an active set solver that accelerates the solution of the normal equations by precomputing the matrix $A'^T A'$.

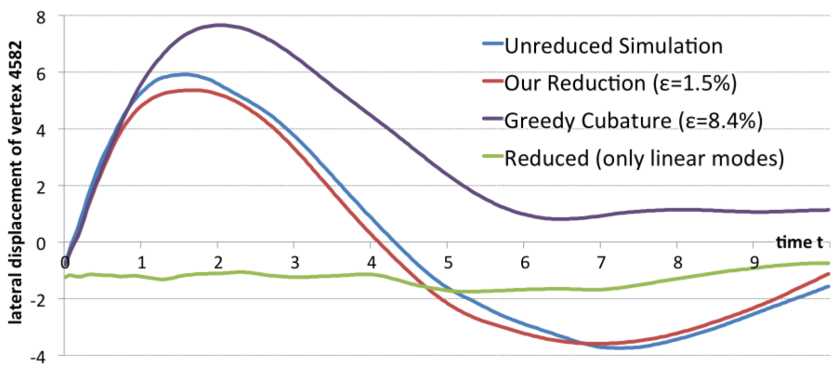


Figure 3.9: **Runtime accuracy.** Lateral displacement of a vertex at the head of the dragon model (see Figure 3.1 and Table 3.3) for the full and different reduced simulations.

Lazy optimization In every iteration of our NN-HTP solver, at most s new components can enter the subset. Considering all remaining components as candidates comes at a significant computational cost; especially since matrix A requires a storage size of $O(dTm)$ and is therefore evaluated only partially when needed.

To reduce the costs of the subset optimization, we introduce a parameter c that specifies the number of considered components. This parameter allows us to trade-off convergence speed against iteration costs of our NN-HTP solver. In every iteration, we randomly choose c of the remaining components and calculate only the corresponding columns of A. In fact, we found that considering only a subset can even increase the convergence of the algorithm. For our examples, we achieved the best results with $c \approx 5s$.

Automatic training pose generation Given the material and a subspace deformation model for a deformable object we can generate training poses automatically by randomly sampling points in the subspace. To acquire a material-aware sampling that roughly balances the ratio of configurations with high and low strain energy, we perform the sampling in the basis $\bar{\Phi}_1, \bar{\Phi}_2, \dots, \bar{\Phi}_d$ of vibrations modes in the reduced space. In particular, $q_t = \sum_i^d \text{rnd}(\bar{\omega}_i^{-1})\bar{\Phi}_i$, where $\text{rnd}(\bar{\omega}_i^{-1})$ generates normally distributed random numbers with standard deviations proportional to the inverse of the frequency $\bar{\omega}_i$.

Results and comparison to greedy cubature The total time needed to set up the reduced deformable models is dominated by the subspace con-

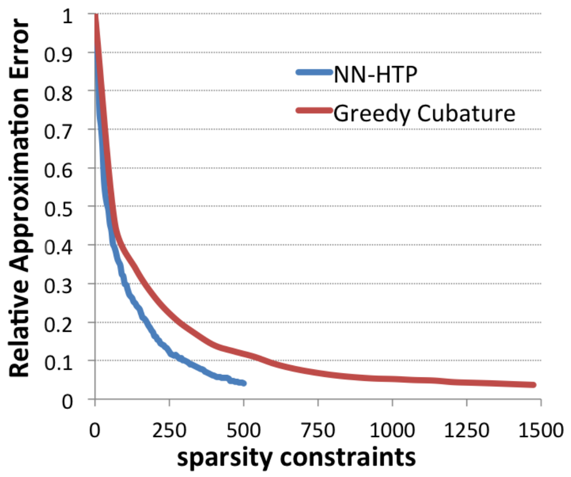


Figure 3.10: Achieved subset optimization error with increasing sparsity constraints for NN-HTP and the Greedy Cubature by [An et al., 2008]. The plot shows optimization results for the elephant shell model in a 135-dimensional subspace (cf. Table 3.3 for $s = 500$).

struction (cf. Section 3.2.2), training force evaluation and subset optimization. Table 3.3 provides the times for each of the steps. Figure 3.8 visualizes optimal sets of components, i.e., tetrahedra and edge flaps used to approximate the St. Venant–Kirchhoff material and the Discrete Shells energy, respectively. Clearly, component selection is not based on purely geometric criteria. In particular, components tend to cluster in certain regions of the objects, e.g., the jaw of the dragon and the right foreleg of the elephant; both regions proved to exhibit more versatile deformation behavior as compared with regions having fewer components.

An advantage of the proposed solver compared to the greedy strategy of An et al. [2008] is that the NN-HTP scheme reduced the number of iterations required to construct a selection set. The greedy strategy adds one component to the selection set in every iteration, hence, requires as many iterations as the cardinality of the selection set. In contrast, the NN-HTP builds a complete selection set in the first iteration and then updates this set at every iteration. In our experiments, we typically needed ten or less iterations to construct the selection set. As a consequence, the number of NNLS problems that need to be solved to construct a selection set decreases. In addition, we use the FNNLS routine to solve the NNLS problems, which we found more effective than the Lawson–Hanson algorithm, see [Lawson and Hanson, 1974], used by An et al. [2008]. We want to remark that, independent from our work, Kim and Delaney [2013] proposed a modified greedy cubature construction that reduces the number of iterations and uses FNNLS.

Table 3.3 provides a comparison of our optimization algorithm (NN-HTP) to the greedy optimization approach by An et al. [2008] (using the implemen-

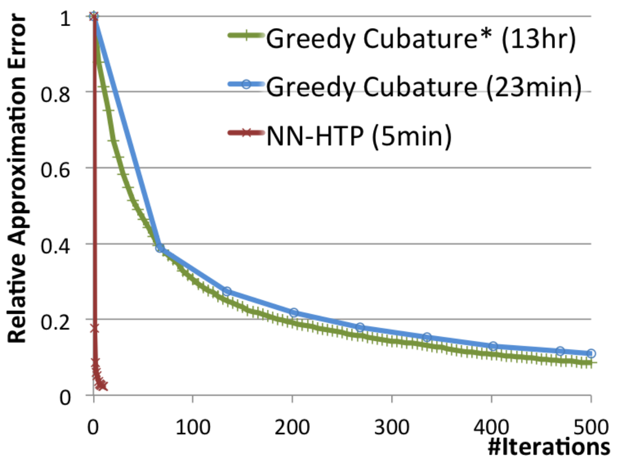


Figure 3.11: Comparison between our NN-HTP and Greedy Cubature by [An et al., 2008] for the dinosaur model (see Table 3.3). (∗) Disabling subset training in greedy solver increases runtime immensely with only marginal improvements in convergence.

tation by Steven An with enabled parallelization). For the comparisons we used St. Venant–Kirchhoff and Mooney–Rivlin materials and Discrete Shells. The results illustrate the advantages of NN-HTP—it produces a significantly smaller approximation error in considerably less time. Furthermore, it is able to achieve a given training error with a smaller selection set yielding force approximations with considerably higher runtime performance. In particular, Figure 3.10 extends the elephant experiment from Table 3.3 by showing approximation errors as a function of the sparsity constraints. Greedy cubature reaches 3.7% training error with a sparsity constraint of 1474 components (almost three times as many as needed by NN-HTP). Tests of the greedy algorithm were performed using runtime-favoring configurations as reported in [An et al., 2008] ($|\mathcal{C}| = 100, T_s = 10$). However, as shown in Figure 3.11, disabling the *subset training* only slightly improves the approximation error from 11.0% to 8.5% but significantly increases runtime. On the other hand, NN-HTP is able to find a solution with 2.3% error in only ten iterations. The benefit of a more accurate force approximation for simulation is illustrated in the accompanying video and Figure 3.9. The video contains a sequence that compares the full simulation of the dragon model with reduced simulations using force approximations computed by our method and by greedy cubature with the same cardinality constraints (for details, see Table 3.3). Furthermore, the figure shows the lateral displacement of the mesh vertex with the largest initial velocity for the different simulations.

3.3.2 Coarsening of elastic materials

An alternative strategy to approximate internal forces is to exploit the spatially local coherence of an elastic material by using a coarse geometric representation of the underlying deformable object. For this, we construct a low resolution version N^s of the triangulation N that represents the initial complex shape. In addition, we simplify the subspace basis $\{b_i\}$. Each b_i is a vector field on N, we compute corresponding vector fields b_i^s on the simplified mesh N^s and define the reduced shape space of the simplified mesh as $V^s = \bar{u}^s + \text{Span}\{b_i^s\}$. Every $u \in V$ has a unique representation in the basis $\{b_i\}$, $u = \bar{u} + \sum_i q_i b_i$, and the linear map given by $\bar{u} + \sum_i q_i b_i \to \bar{u}^s + \sum_i q_i b_i^s$ is an isomorphism of the reduced shape spaces V and V^s. To approximate the reduced internal force of the configuration $\bar{u} + \sum_i q_i b_i \in V$, we can compute the reduced internal force for the coarse mesh $\bar{u}^s + \sum_i q_i b_i^s$.

Let v^s be a vertex of the coarse mesh and let $\{v_1, v_2, ..., v_n\}$ be the set of vertices on the fine mesh that are collapsed to v^s. We construct the simplified basis vectors b_i^s by setting $b_i^s(v^s) = \frac{1}{n}\sum_k b_i(v_k)$. If the initial mesh is strongly irregular, it is reasonable to include the masses of the vertices into this averaging process. In particular, we use an edge-collapse scheme for simplicial surfaces to generate the coarse triangulation. Edge-collapse schemes implicitly generate a map from the vertices of the fine mesh to the vertices of the coarse mesh: for every vertex of the fine mesh there is exactly one vertex on the coarse mesh to which it has been collapsed.

We emphasize that though we approximate the dynamical behavior of a coarse mesh, the fine mesh varies in a space spanned by nice and smooth basis vectors that are computed from the fine mesh. Actually, the simplified mesh is never shown to or handled by the user, therefore, we call it the *ghost*.

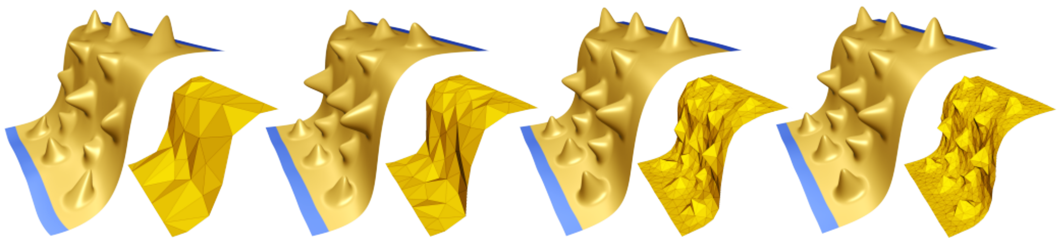


Figure 3.12: Editing results using approximations based on different ghost sizes. Number of vertices (faces) from left to right: 37 (50), 67 (100), 283 (500), and 544 (1000). The ghosts are shown in the bottom row.

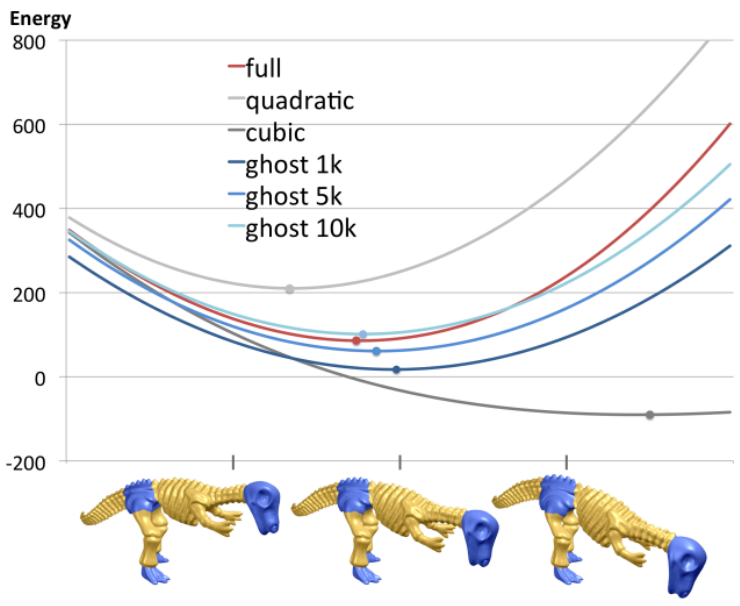


Figure 3.13: Different approximations of the potential energy $\mathcal{E}$ in a one-dimensional affine subspace of the shape space are shown as graphs. The minima of the energies are indicated by dots. Graphs of the full energy, a second-order and a third-order Taylor series of $\mathcal{E}$, and approximations using ghosts with 1k, 5k, and 10k vertices are shown.

Results Figure 3.13 shows a comparison of different approximations of the potential energy $\mathcal{E}$ as used in our deformation-based modeling framework from Chapter 5. Internal forces (and their potential) are based on the Discrete Shells energy. The full energy, a second-order (quadratic) and a third-order (cubic) Taylor series of $\mathcal{E}$ around the rest state $\bar{u}$, and approximations using ghosts with 1k, 5k, and 10k vertices are shown as graphs over a one-dimensional affine subspace of the shape space. To illustrate which subspace was used, we attach images that show shapes in the subspace to the x-axis of the image. The results of our modeling method depend on the location of the minima rather than on the values of the energy, therefore, we added dots to the graphs that indicate the location of the minima. The figure illustrates the experimental observation that our technique to approximate the energy using a ghost mesh produces a smaller approximation error than a Taylor expansion up to second or third order. Furthermore, it demonstrates that the approximation error reduces with increasing size of the ghost mesh.

If necessary, the number of vertices of the coarse mesh can thus be increased to improve the approximation quality. Still, the quality of the force approximation scheme is directly connected to the insensitivity of the discrete energy against simplification of the mesh. Furthermore, objects with high structural complexity or non-homogeneous material require meshes fine enough to resolve the fine-scale heterogeneities to capture the proper material response. Kharevych et al. [2009] present a coarsening method that generates dynamically similar coarse meshes based on homogenization theory. However, their scheme is limited to linear elasticity, and, to the best of our knowledge, an extension to (geometrically) nonlinear deformation energies is still an open problem.

	Model	m	d (r)	U	T	b	s	$\tilde{F}_w^{int}$	Stp	$\frac{\text{Stp}}{\text{s}}$	NN-HTP		Greedy Cub.	
											error	time	error	time
shell	Bunny	104310	135 (15)	26s	2k	125s	800	3.7ms	10.7ms	93	2.9%	593s	12.2%	2.2h
	Ch. dragon	389994	138 (20)	110s	1k	243s	250	1.5ms	4.9ms	203	7.4%	69s	17.4%	266s
	Dinosaur	168576	135 (15)	41s	1k	101s	500	3.0ms	7.9ms	126	2.3%	223s	11.0%	1293s
	Elephant	75000	135 (15)	18s	1k	46s	500	2.8ms	10.1ms	99	3.7%	205s	11.0%	1356s
	VW Beetle	52645	300 (300)	67s	2k	227s	800	9.7ms	20.6ms	49	2.8%	1473s	12.1%	2.7h
solid	Aorta*	35551	138 (20)	6s	1k	73s	200	1.0ms	4.4ms	245	2.1%	22s	10.9%	182s
	Dragon*	92386	135 (15)	10s	1k	182s	150	0.9ms	3.7ms	272	1.5%	63s	8.4%	130s
	Menger†	393216	135 (15)	36s	1k	332s	100	0.5ms	2.0ms	509	2.7%	8s	7.0%	116s
	Neptune†	691434	135 (15)	55s	1k	396s	200	0.8ms	2.4ms	424	2.8%	41s	25.6%	148s
	Skull*	156157	300 (300)	186s	1k	470s	250	3.5ms	9.6ms	104	2.3%	66s	8.4%	606s
	Vertebrae*	70447	15 (15)	7s	1k	117s	30	0.1ms	0.2ms	5059	1.5%	1s	7.6%	6s

Table 3.3: **Statistics** measured on a 2012 MacBook Pro Retina 2.6GHz. From left to right: number of components, dimension of subspace (number of linear modes used), time for computing extended modal basis, number of training poses, time for computation of training forces, posed cardinality constraints, time to evaluate force approximation, time for one step of implicit Newmark integration, time-stepping rate of subspace integration, achieved subset optimization error and optimization time for our scheme and the scheme from [An et al., 2008]. Experiments are based on *Discrete Shells* [Grinspun et al., 2003] and finite elements discretizations of the St. Venant–Kirchhoff ($*$) and Mooney–Rivlin (†) material models. (3% error and max. 10 iterations for NN-HTP as stopping criteria for all runs except the example of the dragon, which converged to 1.5% after 18 iterations.)

Chapter 4

Real-time simulation of deformable objects

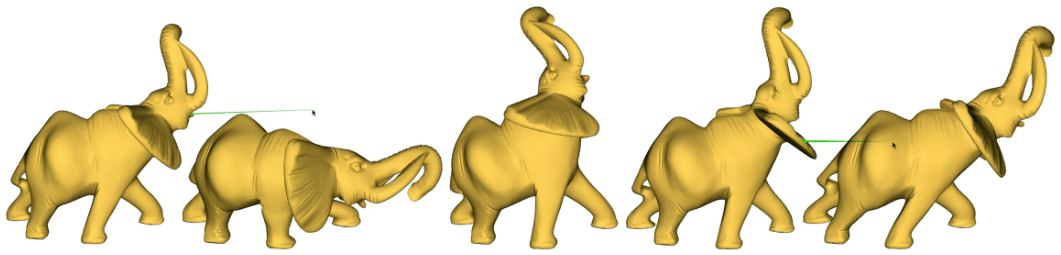


Figure 4.1: User interaction with a real-time simulation of an elephant model (see Table 3.3).

4.1 Introduction

The simulation of nonlinear dynamical systems is notorious for being computationally demanding. Therefore, to keep up with ever larger and more complex models ever more elaborate methods are needed. Based on the results of the previous chapter, we present a fast and robust scheme for the simulation of nonlinear deformable objects based on model reduction that is suitable for interactive applications.

Related work

Since the pioneering work of Terzopoulos et al. [1987], which introduced deformable objects in computer graphics, numerous strategies were devised to address the problem of efficiency, robustness, and scalability. Schemes for the numerical treatment of the nonlinear systems have seen great improvements through the use of fast and stable integration schemes [Baraff and Witkin, 1998; Müller et al., 2001] as well as multiresolution techniques [Debunne et al., 2001; Grinspun et al., 2002; Capell et al., 2002] that employ adaptive refinement of the simulation guided by, e.g., the local amount of deformation.

Another string of contributions focuses on simplified physical models to reduce computational requirements, leading to a revival of linear elasticity [Bro-Nielsen, 1997; James and Pai, 1999]. Corotational methods [Müller et al., 2002; Müller and Gross, 2004; Georgii and Westermann, 2008; Hecht et al., 2012] extend linear elasticity models by exploiting local frames of reference at each node or element. Geometric nonlinearity is then injected into the simulation which compensates for the well-known limitations of linear models for large deformations. However, corotational methods require extracting rotations for each node or element at every timestep, e.g., by performing a polar decomposition of the deformation gradient, incurring computational costs that depend on the resolution of the deformable object.

A simple, yet efficient, way to simulate highly resolved models at low costs is to embed the complex shape in a coarse volumetric mesh (also referred to as cage). Such cage-based methods [Sederberg and Parry, 1986; Müller et al., 2004; Wojtan and Turk, 2008] animate the coarse mesh to induce deformations of the complex embedded shape and hence provide cheap approximations of its dynamical behavior. As the cage is typically defined purely based on the space occupied by the object, it neither uses the information on the spatial distribution of material parameters nor accounts for fine-scale geometric details.

As pointed out earlier, model reduction techniques construct a low-dimensional approximation of the dynamical system underlying the simulation and thereby achieve a runtime that is independent of the resolution of the simulated mesh. In recent years, model reduction has seen widespread application in computer graphics and we refer the reader to the introduction of Chapter 3 for an overview of related work.

4.2 Reduced equations of motion

In this section, we briefly review the basics of model reduction for the physical simulation of deformable objects and introduce our notation. For a recent introduction to model reduction for the simulation of deformable objects, we refer to [Sifakis and Barbic, 2012].

There are different physical models of deformable objects and various ways to discretize them. We keep our presentation general so that it covers a broad class of discrete deformable objects; the specific setting used for our experiments is treated in Section 4.4. We consider a discrete deformable object with n degrees of freedom, i.e., the nodal vector u encoding a displacement is n-dimensional and the configuration space $\mathcal{C}$ can be identified with $\mathbb{R}^n$. For example, for simplicial meshes in $\mathbb{R}^3$ equipped with continuous and piecewise linear shape functions, n is three times the number of free vertices.

The dynamics of a deformable object are described by the equations of motion, which read

$$M\,\ddot{u}(t) = F(t, u(t), \dot{u}(t)),$$

where F represents the acting forces and M is the positive definite and symmetric mass matrix. The forces F are a superposition of internal deformation forces $F^{int}(u(t))$ of the elastic shape, external forces $F^{ext}(t, u(t), \dot{u}(t))$, and damping forces $F^{damp}(u(t), \dot{u}(t))$.

Dimension reduction restricts the configuration space to a d-dimensional subspace V of $\mathbb{R}^n$. Explicitly, we employ the extended modal basis construction that is treated in Section 3.2.2. Let $\{b_i\}_{i\in\{1,2,\ldots,d\}}$ be a basis of V, then the matrix $U = [b_1, b_2, \ldots, b_d] \in \mathbb{R}^{n\times d}$ maps the reduced coordinates q in V onto the corresponding displacement vector $u \in \mathbb{R}^n$,

$$u = U\,q.$$

The reduced equations of motion are

$$\bar{M}\,\ddot{q}(t) = \bar{F}(t, q(t), \dot{q}(t)), \tag{4.1}$$

where

$$\bar{M} = U^T M U \text{ and } \bar{F}(t, q(t), \dot{q}(t)) = U^T F(t, Uq(t), U\dot{q}(t))$$

are the reduced mass matrix and the reduced forces.

Since $d \ll n$, the reduced system in Equation (4.1) can be integrated much faster than the unreduced equation of motions, albeit with some loss of accuracy. In addition, we employ our cubature-based force approximation (see

Section 3.3.1) that depends on d instead of n and thus yields simulation costs independent of the resolution of the unreduced system.

In our implementation, we model the damping forces based on linear Rayleigh damping

$$\bar{F}^{damp}(q(t), \dot{q}(t)) = \left(\alpha_1 \bar{M} + \alpha_2 \bar{K}\right) \dot{q}(t),$$

where $\bar{M}$ and $\bar{K}$ are the reduced mass and tangent stiffness matrices. The two parameters α_1 and α_2 can be adjusted to control the damping of the low- and high-frequency components of the deformation.

To enable a user to interact with the deformable object, we provide a simple click-and-drag interface that allows to pull at vertices of the (boundary) surface. More precisely, we apply external forces to the clicked vertices that are modeled as ideal Hookean springs connecting the vertices and the mouse cursor.

4.3 Efficient subspace integration

To simulate the dynamics of a reduced deformable object, the system (4.1) is numerically integrated over time. Due to the nonlinear forcing terms and the high-frequency components contained in our large-deformation model, explicit schemes can be hard to control as numerical stiffness causes the integrator to become unstable. Because stability is crucial for interactive applications, we opt for a second-order accurate, implicit Newmark integrator.

Let q_i, $\dot{q}_i$, and $\ddot{q}_i$ denote the solutions to Equation (4.1) at time t, then the Newmark method consists of the following equations:

$$\bar{M}\,\ddot{q}_{i+1} = \bar{F}^{ext}_{i+1} + \bar{F}^{int}(q_{i+1}) - \bar{C}\,\dot{q}_{i+1} \tag{4.2}$$

$$q_{i+1} = q_i + \delta t \dot{q}_i + \frac{\delta t^2}{2}\left[(1-2\beta)\ddot{q}_i + 2\beta \ddot{q}_{i+1}\right] \tag{4.3}$$

$$\dot{q}_{i+1} = \dot{q}_i + \delta t\left[(1-\gamma)\ddot{q}_i + \gamma \ddot{q}_{i+1}\right] \tag{4.4}$$

where $\bar{C}$ is the reduced damping matrix and q_{i+1}, $\dot{q}_{i+1}$, and $\ddot{q}_{i+1}$ are the approximations of $q(t+\delta t)$, $\dot{q}(t+\delta t)$, and $\ddot{q}(t+\delta t)$. The vector $\bar{F}^{ext}_{i+1}$ denotes the reduced external force at time $t + \delta t$ and is assumed to be known. Equation (4.2) is the reduced equation of motion in terms of the approximate solutions, and Equations (4.3) and (4.4) describe the evolution of q, $\dot{q}$, and $\ddot{q}$ using finite differences. The Newmark family includes many well-known and established methods each corresponding to a particular choice of the parameters β and γ.

Some classical methods and their properties are summarized in [Hughes, 2000]. In particular, we use the *average acceleration method* ($\beta = {}^1/_4$, $\gamma = {}^1/_2$), which is unconditionally stable and widely used in structural dynamics.

There are several possible ways to implement the Newmark method. One form of implementation eliminates $\dot{q}_{i+1}$ and $\ddot{q}_{i+1}$ from (4.2) using (4.3) and (4.4) which yields the relation

$$\left(\frac{1}{\beta\delta t^2}\bar{M} + \frac{\gamma}{\beta\delta t}\bar{C}\right)q_{i+1} + \bar{F}^{int}(q_{i+1}) = \underbrace{\bar{F}^{ext}_{i+1} + \bar{M}\tilde{a} + \bar{C}\tilde{v}}_{b}, \tag{4.5}$$

where $\tilde{a} = \frac{1}{\beta\delta t^2}q_i + \frac{1}{\beta\delta t}\dot{q}_i + (\frac{1}{2\beta} - 1)\ddot{q}_i$ and $\tilde{v} = \frac{\gamma}{\beta\delta t}q_i + (\frac{\gamma}{\beta} - 1)\dot{q}_i + (1 - \frac{\gamma}{2\beta})\delta t\ddot{q}_i$. Note that in this implementation the right-hand side b in (4.5) does not entail stiffness computations. Advancing the solution of the dynamical system to time $t + \delta t$ then essentially consists of solving the equilibrium equations (4.5), i.e., finding reduced coordinates q for which

$$R(q) = \left(\frac{1}{\beta\delta t^2}\bar{M} + \frac{\gamma}{\beta\delta t}\bar{C}\right)q + \bar{F}^{int}(q) - b = 0^d,$$

where $R(\cdot)$ is a d-dimensional, nonlinear function describing the residual. Solutions to $R(q) = 0^d$ can be computed iteratively using Newton-like methods which construct estimates q^{j+1} by solving for roots of an affine approximation $R'(q) = R(q^j) + B(q^j)(q - q^j)$. For example, adopting the Jacobian matrix $\partial R(q^j)$ of the residual as $B(q^j)$ yields the well-known Newton-Raphson scheme. One step of the Newmark subspace integration is summarized in Algorithm 3.

Solving the nonlinear system of equations to determine q_{i+1} is the most expensive operation in each step of the implicit Newmark integration. In our experiments, we found a quasi-Newton solver that constructs an approximation of the inverse of the Jacobian matrix on the fly to be very effective for this. Explicitly, we used a variant of *Broyden's method* (see [Dennis and Schnabel, 1987]), which achieves superlinear convergence, but in each iteration merely requires the evaluation of the internal forces, a low-rank matrix update, and matrix-vector operations. Using the force approximation from Section 3.3.1, all these operations can be performed at $O(d^2)$ cost. To achieve a warm start, we compute the inverse Jacobian at the rest state during the preprocess and use it as a preconditioner for the nonlinear system. An alternative to the quasi-Newton scheme would be to use a Newton-Raphson solver.

```
Algorithm: Newmark method
Data: $q_i$, $\dot{q}_i$, $\ddot{q}_i$, $\bar{F}^{ext}_{i+1}$, max. iterations $j_{max}$, tolerance $\tau$, $\delta t$
Result: $q_{i+1}$, $\dot{q}_{i+1}$, $\ddot{q}_{i+1}$
$q_{i+1} \leftarrow q_i$;
Compute right-hand side $b$ of (4.5);
for $j = 0, \dots, j_{max}$ do
    Evaluate $R(q_{i+1})$;
    if $\|R(q_{i+1})\| \leq \tau$ then break;
    Evaluate $B^{-1}(q_{i+1})$;
    $q_{i+1} \leftarrow q_{i+1} - B^{-1}(q_{i+1})R(q_{i+1})$;
end
Determine $\dot{q}_{i+1}$ and $\ddot{q}_{i+1}$ via (4.3) and (4.4);
```

Algorithm 3: One-step algorithm for the reduced equations of motion.

4.4 Results and discussion

We evaluate the presented system using finite element discretizations of the St. Venant–Kirchhoff and Mooney–Rivlin material for simplicial volume meshes as well as the *Discrete Shells* model [Grinspun et al., 2003] for simplicial surface meshes.

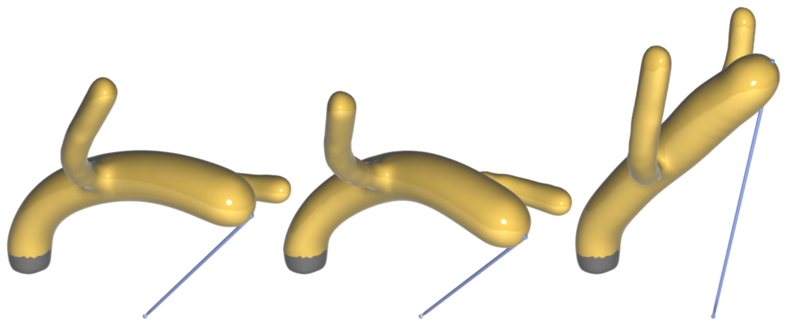


Figure 4.2: Comparison of simulations of a spring pulling at a deformable object. From left to right: reference full simulation, simulation using the proposed space reduction and force approximation, results of a nonlinear simulation in a reduced space spanned by linear modes. Reduced spaces have the same dimensions.

We provide runtime statistics of reduced simulations for various geometries and parameter settings in Table 3.3. This includes timings for the evaluation of the reduced forces and one step of an implicit Newmark integration. In addition, the resulting time-stepping rate of subspace integration (as average measured over 1k timesteps with constant external loads) is listed. Depending on the parameter settings, the reduced simulations achieve rates of 50-5000 implicit Newmark integration steps per second which shows the suitability of our simulation system for real-time applications.

The reduced internal forces of a single component (tet or edge flap) can be evaluated at $O(d)$ cost. As in [An et al., 2008], we observed that the number of components needed to achieve a given error tolerance grows linearly with the subspace dimension. Thus, our reduced-order simulation achieves force approximation independently of the full shape space at $O(d^2)$ cost.

To evaluate the fidelity of the reduced simulation, we measured the deviation of the reduced from the reference-full simulation for a dragon model with 92k tets (see Figure 3.1). The average L^2-distance over all frames is listed in Table 3.1. Additionally, we show both simulations in the accompanying video and plot the lateral displacement of the mesh vertex with the largest initial velocity (a vertex at the side of the head) in Figure 3.9. The example illustrates that the proposed scheme is able to closely match the unreduced trajectory while featuring superior runtime performance—a more than 6000-fold increase in simulation speed after only about four minutes of preprocessing.

Chapter 5

Interactive deformation-based modeling

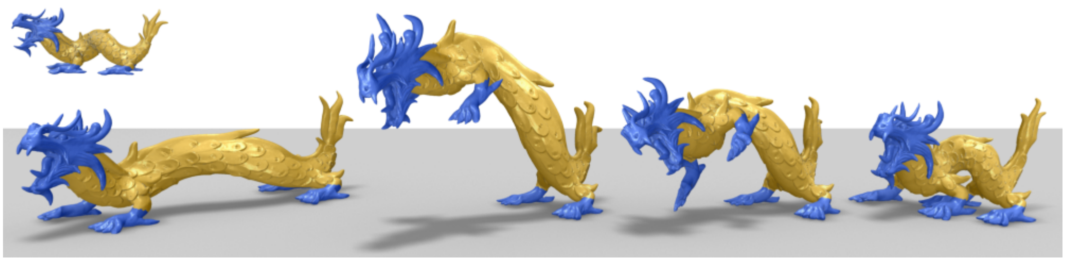


Figure 5.1: Large deformations of the Chinese dragon model (130k vertices) computed by our modeling framework in a 130-dimensional shape space using a 1k ghost.

5.1 Introduction

In recent years, a special focus in geometric modeling has been on schemes for deformation-based surface editing. A major advantage of such schemes over traditional modeling techniques like NURBS or subdivision surfaces is that typical modeling operations can be described by few constraints. This allows for efficient and simple click-and-drag user interfaces. To provide intuitive usability, the computed deformations must be physically meaningful to match the user's intuition and experience on how shapes deform. This leads to nonlinear optimization problems that, to achieve interactivity, have to be solved within fractions of a second. The surfaces to be edited are often rich in detail and thus of high resolution. We distinguish between local and global deformations. Local deformations are restricted to a small area with the rest of the surface fixed. The resulting optimization problems are of small scale; still, it is challenging to solve them at interactive rates. The focus of this work is on global shape deformations which lead to optimization problems that, due to their size, cannot be solved at interactive rates. Since interactivity is of paramount importance, the challenge is to design methods that find as-good-as-possible approximations at a low computational cost.

Instead of manipulating the surface directly, recent schemes for global interactive deformation-based editing manipulate a part of the ambient space that surrounds the surface and therefore implicitly edit the surface. The deformations of the space are often induced by a cage, which in turn is manipulated by a deformation-based editing scheme. The advantage of this concept is that the size of the optimization problem now depends on the resolution of the cage and is independent of the resolution of the surface.

In this chapter we derive a framework for deformation-based modeling that is interactive, robust, intuitive to use, and works with various deformation energies. The main idea is to reduce the complexity of the optimization problem in two ways: by using a low-dimensional shape space and by approximating the energy and its gradient and Hessian. The motivation for the space reduction is the observation that a modeling session typically requires only a fraction of the full shape space of the underlying model. We construct a reduced shape space as the linear span of two sets V_1 and V_2 of vectors. The set V_1 comprises the linearized vibration modes at the rest state of the object corresponding to the lowest frequencies (see Section 2.3). The span of these vectors contains the deformations that locally cause the least increase of energy and is therefore well-suited to generate small deformations. However, large deformations in $\text{Span}(V_1)$ often develop artifacts and thus have high energy values. To improve the representation of large deformations in the reduced space, we collect modal derivatives that correspond to vibration modes in V_1 in the set V_2. These directions are constructed using the third partial derivatives of the underlying deformation energy. In Section 3.2.1 we provide a derivation the modal derivatives in terms of a Taylor expansion of the Newton descent direction. For the approximation of the energy and its derivatives, we propose a scheme based on a second reduced shape space for a simplified mesh (in Section 3.3.2). By construction, the two reduced shape spaces are isomorphic and we can use the isomorphism to pull the energy from the shape space of the simplified mesh to the shape space of the full mesh. Finally, the resulting reduced problem is independent of the resolution of the mesh and our experiments show typical modeling operations, including large deformations, that are reasonably well approximated. To solve the reduced optimization problem, we use a quasi-Newton method that maintains an approximation of the inverse of the Hessian and generates descent directions at the cost of gradient evaluations while still producing superlinear convergence.

Our approach is an alternative to space deformation schemes. Space deformations are controlled by some object, e.g., a cage around the surface (other objects like a volumetric mesh or a skeleton have been used as well). Then, the set of possible deformations of the surface depends on the cage and a space warping scheme. In contrast, our method does not depend on an artificial cage and a space warping scheme, but the subspaces we consider depend on geometric properties of the surface. A resulting advantage of our approach is that we do not need to deal with interpolation artifacts that many space warping schemes create. Furthermore, our scheme does not need to construct a cage, which often is a manual process. Instead the preprocess of our scheme is automatic and the computed basis can be stored on a hard disc with the surface. The subspaces our method produces are effective: we demonstrate that even 67-dimensional spaces can produce good approximations for typical modeling operations. In contrast, the coarsest cages that are used for space deformation have 200-500 vertices, hence generate a 600-1500 dimensional shape space. In addition, our approach is flexible. We can use the same energies as in the unreduced case, like *PriMo* [Botsch et al., 2006] or *as-rigid-as-possible* [Sorkine and Alexa, 2007]. The approximation quality of the energy is adjustable and the size of the reduced space can be increased. Increasing both parameters will lead to the exact solution of the unreduced problem. Both parameters, quality of energy approximation and size of reduced space are independent.

Related work

Deformation-based modeling of surfaces describes shape editing operations relative to an initial surface, e.g., a surface generated by a 3d-scanner. Such methods are driven by a deformation energy, i.e., a function on a shape space of surfaces that, for every surface in the shape space, provides a quantitative assessment of the magnitude of the deformation of the surface from the initial surface. Methods for deformation-based modeling can be classified into three categories: linear, non-linear, and space deformation schemes. In addition,

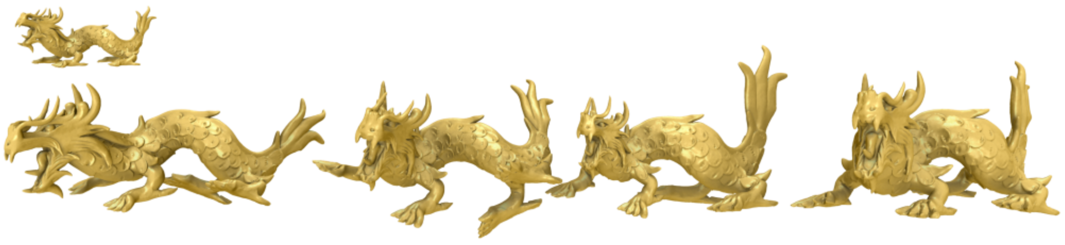


Figure 5.2: Linear vibration modes and modal derivatives of the *Discrete Shells* energy on the Chinese dragon model. Figure shows: the rest state (top left), two linear modes (left), and two modal derivatives (right).

our work is linked to dimension reduction in physical simulations and modal analysis in geometry processing.

Linear surface modeling Linear methods for surface modeling employ a quadratic deformation energy. Such energies are based on linearized thin shells, or, alternatively, on Laplacian coordinates or differential coordinates. Energies based on Laplacian coordinates assess the magnitude of a deformation of a mesh from an initial mesh by summing up the (squared norms of the) deviations of the local Laplace coordinates (which can be seen as discrete mean curvature vectors) at all vertices. The recent survey [Botsch and Sorkine, 2008] provides a detailed overview of linear approaches and includes a comparison of various schemes. The main advantage of linear methods is that the minimization problem to be solved is comparably simple. For example, if the constraints are also modeled as a quadratic energy, as in [Lipman et al., 2004; Sorkine et al., 2004; Nealen et al., 2005] for example, the deformed surface can be computed by solving a sparse linear system. These methods are designed for small deformations around the initial surface and often produce unintuitive results for large deformations.

Non-linear surface modeling Physical models of elastic shells are strongly non-linear and discretizations yield stiff discrete energies. The resulting optimization problems are challenging, especially if real-time solvers are desired. The non-linear PriMo energy [Botsch et al., 2006, 2007] aims at numerical robustness and physically plausible solutions. The idea is to extended the triangles of a mesh to volumetric prisms, which are coupled through elastic forces. During deformations of the mesh the prisms are transformed only rigidly, which increases the robustness of the energy since the prisms cannot degenerate. As an alternative to elastic shells, non-linear methods based on Laplacian coordinates have been proposed. One idea, which can be found in several approaches, is to measure the change of the length of the Laplacian coordinates instead of measuring the change of the full vector. Dual Laplacian editing [Au et al., 2006] iteratively solves quadratic problems and after each iteration rotates the prescribed Laplacian coordinates to match with the surface normal directions of the current iterate. Huang et al. [Huang et al., 2006] describe the prescribed Laplacian coordinates at each vertex in a local coordinate system that is used to update the direction of the prescribed coordinates. Pyramid coordinates [Kraevoy and Sheffer, 2006] can also be seen as non-linear rotation-invariant Laplacian coordinates. For any vertex v there is a rotation that minimizes, in a least squares sense, the distance between the 1-ring of v on the initial and on the actual surface. The *as-rigid-as-possible* energy [Sorkine and Alexa, 2007] is a weighted sum of these minima over all vertices. Recently, Chao et al. [Chao et al., 2010] proposed to use the *distance between the differential of a deformation and the rotation group* as a principle for a geometric model for elasticity. This model includes a material model with standard elastic moduli (Lamé parameters) and is connected to the Biot strain of mechanics. The connection of this model of elasticity to energies used in geometric modeling, like the *as-rigid-as-possible* energy, opens the door to an analysis of the link of these energies and the Biot strain. The drawback of using non-linear energies for surface modeling is that directly solving the resulting minimization problem is costly, thus interactive performance is limited by the size of the meshes.

Space deformation Instead of deforming the surface directly, the idea of space deformation methods is to deform the ambient space around the surface and, therefore, implicitly also the surface. To control the space deformations, a cage is built around the surface. The interior of the cage is described by a boundary representation, i.e., every point in the interior of the volume is represented by coordinates that relate to the vertices of the cage. When the cage is deformed, the coordinates assign a new location to every point inside the cage. Some of the different boundary representations that have been proposed for this purpose are: mean value coordinates [Ju et al., 2005], harmonic coordinates [Joshi et al., 2007], and Green coordinates [Lipman et al., 2008; Ben-Chen et al., 2009]. The advantage of space deformations is that the complexity of the cage is independent of the resolution of the surface. Since it is inconvenient to model the cage directly, the cage can be modeled by a surface deformation scheme [Huang et al., 2006]. Alternatively, space deformations

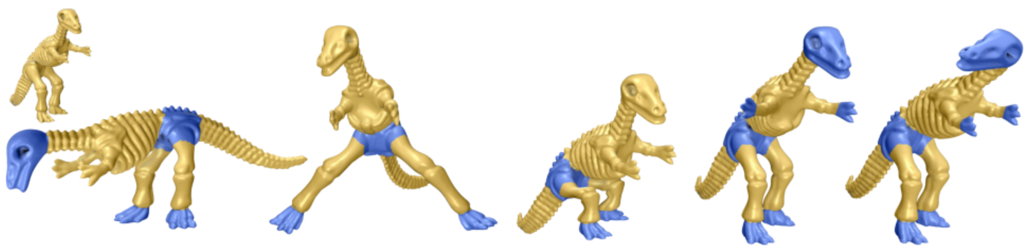


Figure 5.3: Large deformations of the dinosaur model (56k vertices) computed by our method using the *as-rigid-as-possible* energy as the objective functional. We use a 130-dimensional shape space (20 linear modes and 110 modal derivatives) and a ghost with 1k vertices.

can be induced by a skeleton [Shi et al., 2007], a volumetric mesh [Botsch et al., 2007], or a graph structure [Sumner et al., 2007; Adams et al., 2008].

5.2 Reduced-order shape editing

Deformation-based editing allows a user to model a shape by first selecting arbitrary regions of the shape as handles and then translating and rotating them in space. The modeling system automatically computes shapes that follow the handles. This is realized by treating the shape as a deformable object and translating the user input into forces that act on the object. We assume that the internal forces are conservative and denote the potential (deformation energy) by $E(u)$. Our method can work with any energy that is defined for a shape space of meshes and has continuous third derivatives at the reference configuration. However, the quality of our energy approximation scheme is directly connected to the insensitivity of the energy against simplification of the mesh. In our experiments, we used two different energies: the *Discrete Shells* (see Section 2.2 for a definition) and the *as-rigid-as-possible* energy. The external forces that encode the user constraints are based on Hookean zero-length springs, which act on the handles and pull them towards a prescribed position. We denote the potential of the springs by $E^C(u)$ and set

$$\mathcal{E}(u) = E(u) + \kappa_C E^C(u). \tag{5.1}$$

The deformed shape is the minimizer

$$\arg\min_{u \in \mathcal{C}} \mathcal{E}(u), \tag{5.2}$$

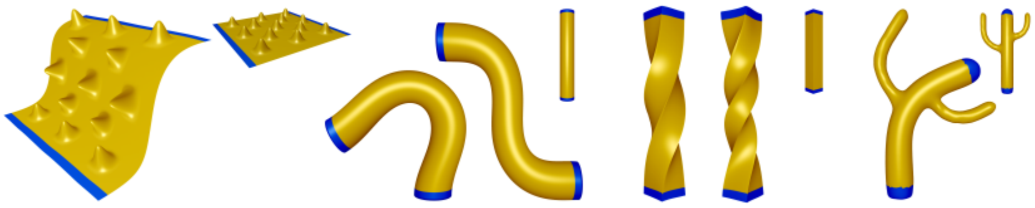


Figure 5.4: Approximation quality of the reduced space with 67 dimensions in all images is demonstrated on the test suite of models and poses introduced in [Botsch and Sorkine, 2008]. Two even larger deformations have been added.

where $\mathcal{C}$ denotes the shape space of the discrete deformable object. For small weights κ_C the constraints are soft and allow for some flexibility, and for larger values of κ_C the constraints tend towards equality constraints. Since the geometries are often highly resolved, e.g., 3D-scan data, this is a high-dimensional nonlinear optimization problem.

This framework for shape editing is linked to physical simulation of elastic shapes. The dynamics of a time-dependent configuration $u(t)$, which represents an elastic shape, is described by a system of second order ODE's of the form

$$M\ddot{u}(t) = F(t, u(t), \dot{u}(t)), \tag{5.3}$$

where F represents the acting forces and M is the mass matrix, cf. Section 4.2. The forces F are a superposition of internal deformation forces $F^{int}(u(t))$ of the elastic shape, external forces $F^{ext}(t, u(t), \dot{u}(t))$, and damping forces $F^{damp}(\dot{u}(t))$. In our setting, the deformation energy E is the potential energy of the internal forces: $F^{int}(u(t)) = -\partial E(u(t))$. In addition, the external forces equal the negative gradient of the energy $\kappa_C E^C$, $F^{ext}(t, u(t), \dot{u}(t)) = -\kappa_C \partial E^C(u(t))$. The physical energy of the system is the sum of the kinetic energy $T(\dot{u}(t))$ and the potential energy $V(u(t))$, where V in our case equals the energy $\mathcal{E}$ defined in Equation (5.1). Due to damping, the system dissipates energy:

$$\frac{d}{dt}(T + V) \leq 0. \tag{5.4}$$

This inequality is strict for all t with $\|\dot{u}(t)\| \neq 0$. For $t \to \infty$ the kinetic energy vanishes and $u(t)$ takes the position of a (local) minimum of $\mathcal{E}$. This means that the deformations our scheme computes are static equilibrium states of the deformable object under the external forces that incorporate the user constraints.

For general meshes, solving the optimization problem, Equation (5.2), is too costly to achieve interactive rates. Even for coarse meshes with 5k-10k vertices, it is challenging to reach a rate of 1fps. Analogous to the simulation system in Chapter 4, we can use model reduction to get an interactive framework for deformation-based editing. Thus, we reduce the complexity of the problem by restricting the optimization to an affine subspace V of the shape space $\mathcal{C}$. Then, the reduced optimization problem is

$$\arg\min_{q \in V} \mathcal{E}(Uq), \tag{5.5}$$

where the matrix U maps the reduced coordinates q in V to the corresponding configurations $u \in \mathcal{C}$. We will write $\bar{\mathcal{E}}(q)$ to denote the reduced energy $\mathcal{E}(Uq)$. An adequate space V should be as small as possible but still contain reasonable approximations of the desired deformations. To construct such a subspace, we employ the linearized vibration modes together with modal derivatives as described in Section 3.2.1.

In our experiments, we often did not use all modal derivatives Ψ_{ij}. We found that using half of the number of possible Ψ_{ij} is a good tradeoff between approximation quality and size of the reduced space. For example, the 67-dimensional basis that we used for many figures includes 52 linearly independent Ψ_{ij}, computed in two *for*-loops over i and j, thus favoring Φ_is with lower frequencies.

In our scheme, the generation of the reduced space is part of a preprocess, before the actual modeling session. To obtain a subspace that is suitable for various types of user interaction, we exclude the energy E^C from this process. As a consequence, the constraints, e.g., the positions of handles, can be modified without forcing a recomputation or adjustment of the subspace.

5.3 Efficient solver

The reduced optimization problem in Equation (5.5) is low-dimensional. However, due to the nonlinearity, it is still challenging to solve at interactive rates. In this section, we derive an optimization scheme that requires only evaluations of the reduced energy and its first partial derivatives, but, unlike steepest descent, features a superlinear convergence. Henceforth, we will consider a mass-orthogonal basis of the subspace, i.e., $U^T M U = \mathbf{1}$.

Solving the reduced optimization problem (5.5) is related to finding the roots of the nonlinear system of equation arising from implicit Newmark integration. Since the Hessian of the reduced energy is the Jacobian matrix of the

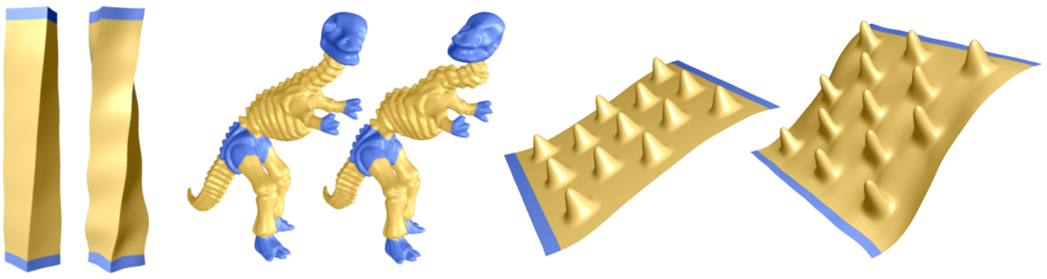


Figure 5.5: Deformation results produced in reduced spaces spanned by the first 130 *linear modes*. For each model a small and a large deformation is shown. The larger deformation is produced with the same constraints as used in Figures 5.3 and 5.4.

d-dimensional system of equations $\frac{\partial}{\partial q}\bar{\mathcal{E}}(q) = 0^d$, it could be approximated with the same techniques used for implicit subspace integration. However, we would neglect important properties inherent to the minimization problem, e.g., the Hessian of the energy is symmetric. In addition, the reduced Hessian is low-dimensional and dense. An efficient quasi-Newton solver which incorporates these properties and again approximates the inverse Hessian (saving the costs of linear system solves) is the *BFGS method* (see [Dennis and Schnabel, 1987]).

Instead of computing the approximate inverse Hessian from scratch at each iteration, the BFGS method uses the change of the gradients at the recent step to update the matrix. Explicitly, the inverse Hessian update is given by

$$B_{k+1} = (\mathbf{1} - \rho_k s_k y_k^T) B_k (\mathbf{1} - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

where B_k is the approximate inverse Hessian at iteration k, $y_k = \partial\bar{\mathcal{E}}_{k+1} - \partial\bar{\mathcal{E}}_k$ the change of gradients, $s_k = q_{k+1} - q_k$ the change of position, and $\rho_k = 1/y_k^T s_k$. The classic BFGS method uses the identity $\mathbf{1}$ as the initial matrix B_1; this means it starts as a gradient descent and becomes more Newton-like during run time. To achieve a warm start of our solver, we compute the inverse of the reduced Hessian at the initial mesh once during the preprocess and use this matrix as the initial approximate inverse Hessian B_1 in the interactive phase. In our experiments, the BFGS solver (with warm start) requires a similar number of iterations to reach a local minimum as a Newton solver, which computes the full Hessian in each iteration (see Figure 5.6).

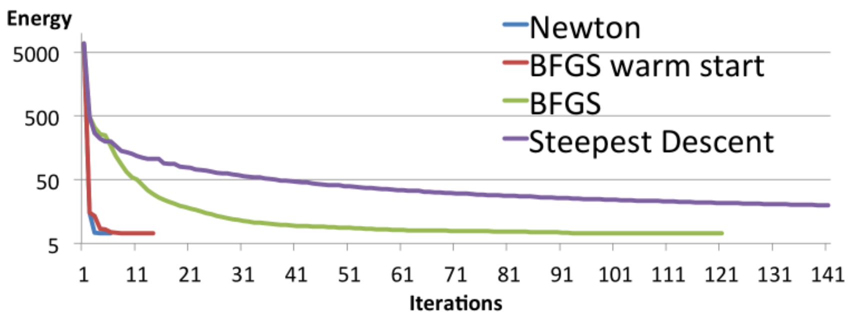


Figure 5.6: Comparison of the performance of different optimization schemes: Newton's method, BFGS with and without warm start, and steepest descent.

5.4 Results and discussion

Eigenvibrations of an elastic shape with small amplitude often look like natural deformations of the shape, as illustrated in Figure 5.2, and shape spaces constructed from linear modes are well-suited to approximate small deformations. But for larger deformations, approximations in such spaces often develop distortions. This is illustrated in Figure 5.5, which shows results obtained in spaces created only by linear modes for small and larger deformations. Our concept to extend the shape space by adding the modal derivatives Ψ_{ij} largely improves the quality of the results. The large deformations shown in Figure 5.5 can be compared to the results shown in Figures 5.3 and 5.4 that are produced with the same poses but in spaces constructed from linear modes and modal derivatives. Examples of vibration modes and modal derivatives are shown in Figures 5.2 and 3.2. Results produced by our method in a reduced space with 67 dimensions (15 linear modes and 52 modal derivatives) on a set of typical test deformations are shown in Figure 5.4. Considering the small size of the reduced space, even the large deformations are astonishingly well approximated. The results shown in Figure 5.4 can be compared with (unreduced) results of various schemes, including PriMo, shown in a comparison table in [Botsch and Sorkine, 2008]. Our scheme is not limited to the *Discrete Shells* energy, but works with other shape deformation energies as well. To use it with other energies, it suffices to exchange the objective functional used for the optimization; if desired, the computation of the modes and modal derivatives can be done with other energies as well. Figure 5.3 shows results produced with the *as-rigid-as-possible* energy as objective functional.

In our experiments, the modeling framework runs robustly on various models, for small and large deformations, and with different parameter settings, like

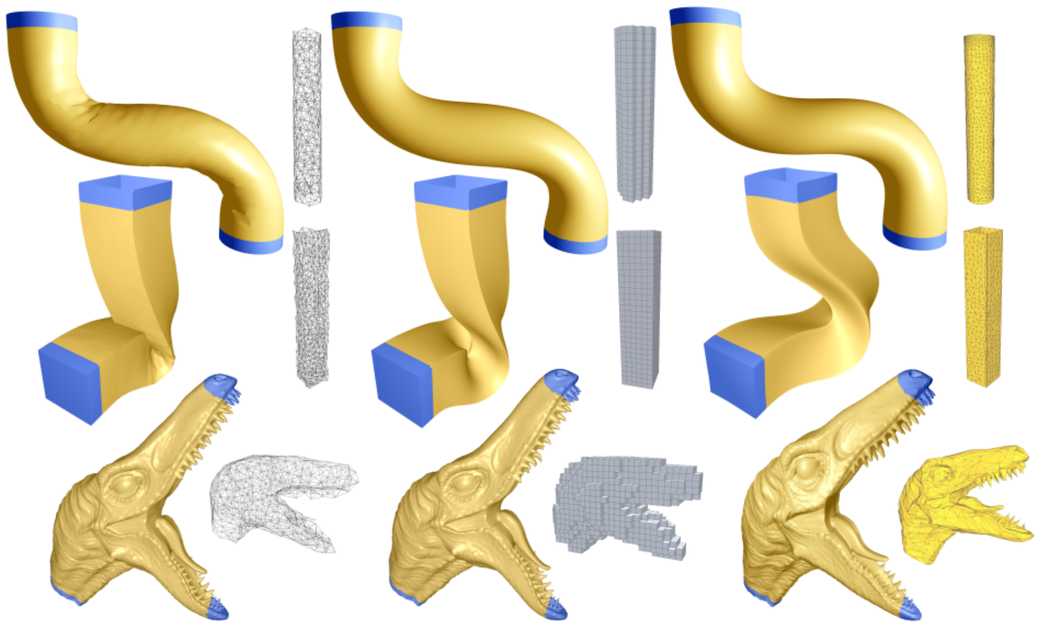


Figure 5.7: Comparison of results of the Embedded Deformations and the Rigid Cells scheme with our method.

the dimension of the reduced space and the resolution of the coarse mesh. Our model reduction has an enormous effect in increasing the stability and reducing the stiffness of the optimization problem. Reasons for this effect are that the reduced shape spaces are low-dimensional and spanned by smooth vector fields that point into directions in which the energy increases slowly. To demonstrate the stabilizing effect of our model reduction, we choose the *Discrete Shells* energy for most of our experiments, instead of the numerically more stable *PriMo* energy or *as-rigid-as-possible* energy. All figures are produced with both material parameters of the *Discrete Shells* energy, κ_{bend} and κ_{mem} in Equation (2.2), equal to 1 and we scaled each surface such that the longest edge of the bounding box has length 10. Figure 5.8 shows the ghosts used to produce Figures 5.1, 5.3, and 5.4. All ghosts are irregular and coarse meshes. We show experiments with various sizes of the reduced shape spaces and the ghosts in Figures 3.12 and 3.3.

Running times Table 5.1 shows running times of the configurations we used to produce the figures and running times for modeling the Chinese dragon model with varying parameters. This demonstrates that our framework pro-

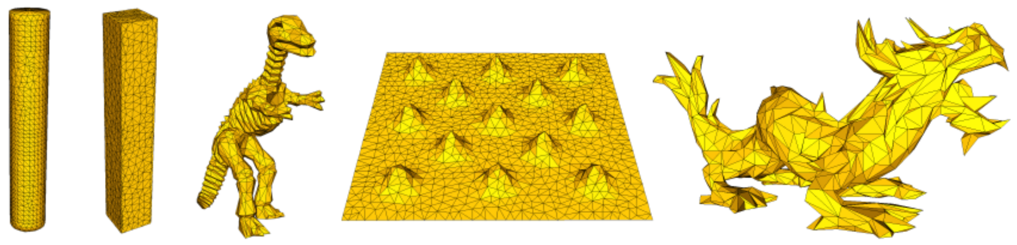


Figure 5.8: The ghosts that were used to produce Figures 5.1, 5.3, and 5.4 are shown.

duces interactive rates even for large meshes with 100k+ vertices. The time needed to solve the optimization problem mainly depends on the size of the reduced space and on the resolution of the coarse mesh. In our experiments, the time needed for one Newton iteration was between 4 and 33ms. During the interactive-modeling phase the constraints, which implement the user input, vary continuously. Therefore, we do not completely solve each optimization problem, but we update the constraints after either a fixed number of iterations is exceeded or an optimality criterion is satisfied. We use the optimality criterion discussed in [Gill et al., 1982], which, for a given ϵ, checks conditions on the change in energy $\bar{\mathcal{E}}$, the convergence of the sequence $\{q_k\}$, and the magnitude of the gradient:

$$\begin{aligned} \bar{\mathcal{E}}_{k-1} - \bar{\mathcal{E}}_k &< \epsilon(1 + |\bar{\mathcal{E}}_k|) \\ \|q_{k-1} - q_k\|_\infty &< \sqrt{\epsilon}(1 + \|q_k\|_\infty) \\ \|\partial\bar{\mathcal{E}}_k\|_\infty &< \sqrt[3]{\epsilon}(1 + |\bar{\mathcal{E}}(q_k)|). \end{aligned} \tag{5.6}$$

In our experiments, we choose a maximum number of 5-10 iterations between updates of the constraints, which yields frame rates of 10+fps. After 5-10 iterations the optimality criterion is usually satisfied with ϵ between 10^{-3} and 10^{-4}. Still, we set $\epsilon = 10^{-6}$ to allow for further iterations if the constraints are not modified. This criterion is usually satisfied after about 15 Newton iterations. The reason that the running time for the *as-rigid-as-possible* energy (last row of Table 1) is much longer than the others is that our current implementation of the energy and gradient evaluation of this energy is very inefficient. Figure 5.6 demonstrates the performance of different optimization schemes. It illustrates that our solver, BFGS with warm start, needs a similar number of iterations to converge as a Newton solver and shows that a BFGS without warm start still converges much faster than steepest descent. Steepest descent required 5806 iterations to converge and still did not reach the same energy level as the Newton or the BFGS solver. The drawback of our approach is that we need to generate the reduced shape space in a preprocess before the actual modeling session can start. In our prototype implementation, which leaves much room for optimization, the preprocess for the 40k bumpy plane took $5\frac{1}{2}$ minutes and for the Chinese dragon model with 130k vertices it took almost 20 minutes. Most of the time is spent on calculating the modal derivatives. But, for every model we only need to compute the reduced basis once and it can be stored with the mesh. Then, after loading, the modeling session can start almost immediately. Also, choosing new handles or changing the resolution of the coarse mesh does not require recomputation of the basis. Alternatively, the extended modal bases presented in Section 3.2.2 can be used instead of modal derivatives. Figure 5.9 shows editing results in a 138-dimensional subspace from extended modes of the same Chinese dragon model used in Figure 5.1. For this example of a hyperelastic, isotropic and homogeneous material the reduced space constructed using extended modes produces results of comparable quality, but at considerably less computational costs. Note that the timings in Table 3.3 and 5.1 are obtained on different hardware—we refer to the results in Table 3.2 for a more unbiased comparison of subspace construction times.

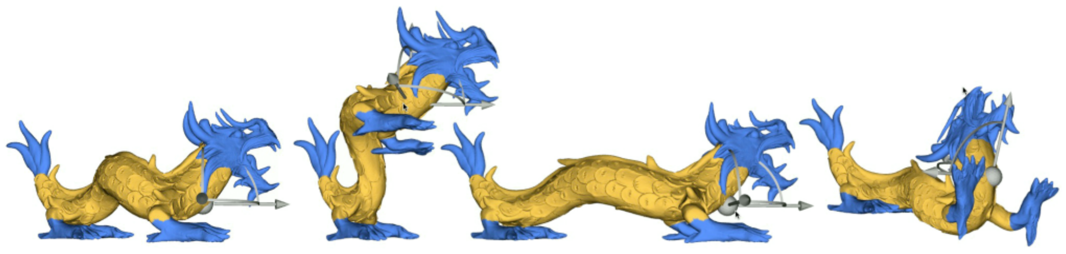


Figure 5.9: Interactive deformation-based editing of a Chinese dragon model in a subspace constructed using the extended modal bases (see Table 3.3 for statistics).

Comparison to previous work We compare the results of our method with two state-of-the-art deformation schemes: *Embedded Deformations* [Sumner et al., 2007] and *Rigid Cells* [Botsch et al., 2007]. The implementations of the methods were kindly provided by their respective authors. Figure 5.7 shows deformations of the cylinder, the bar, and the head of the raptor model. The graph of *Embedded Deformations*, the cell complex of *Rigid Cells*, and the ghost of our method are shown on the right of every deformed model. The left column shows results of *Embedded Deformations* (with a graph of 200 vertices for the cylinder and raptor and 400 for the bar), the middle column of *Rigid Cells* (with 650 cells for the cylinder, 576 cells for the bar, and 1318 cells for the

Model	#Vert.	Dim.	Ghost	Solve	Df.	Total	Prep.	Fig.
Bumpy plane	40k	67	1k	5	3	53(10)	325	5.4
Cylinder	5k	67	1k	5	1	51(10)	38	5.4
Bar	6k	67	1k	5	1	51(10)	48	5.4
Cactus	5k	67	5k	32	1	161(5)	44	5.4
Ch. Dragon	130k	67	0.5k	4	8	48(10)	1067	
Ch. Dragon	130k	67	1k	5	8	58(10)	1067	
Ch. Dragon	130k	67	2.5k	14	8	78(5)	1066	
Ch. Dragon	130k	67	5k	33	8	173(5)	1064	
Ch. Dragon	130k	130	0.5k	7	13	69(8)	1185	
Ch. Dragon	130k	130	1k	10	13	93(8)	1185	5.1
Dinosaur	56k	130	1k	10	6	86(8)	511	
Dinosaur (ARAP)	56k	130	1k	72	6	366(5)	511	5.3

Table 5.1: **Statistics** measured on a 2010 MacBook Pro with a 2.66GHz CPU. From left to right: number of vertices, dimension of reduced space, number of vertices of the ghost, time in milliseconds for one BFGS iteration, time for mapping reduced solution into full shape space, time for full optimization (with maximum number of iterations), time in seconds for the preprocess, and figure that shows the configuration.

raptor), the right column of our method (with a 67-dimensional shape space for the cylinder and bar, a 130-dimensional space for the raptor, and a ghost of 1000 vertices for all). Compared to *Embedded Deformations*, the results our method produces are visually more appealing since *Embedded Deformations* produces some noise artifacts. The results of *Rigid Cells* are comparable to those of our method, but our method is considerably faster (see Table 1). There are three reasons for this. First, *Rigid Cells* requires an expensive interpolation scheme (using radial basis functions) to avoid noise artifacts, which we do not need. Second, our method decouples the approximation quality of the energy (the size of the ghost) from the size of the reduced space, which allows us to use smaller reduced spaces while keeping a reasonable approximation quality of the energy; and, third, *Rigid Cells* is using volume meshes, which are typically larger than surface meshes.

Local deformations A general problem of model reduction schemes for shape modeling is that detail editing is either impossible or requires special treatment. Figure 5.10 and the two rightmost images of Figure 5.3 demonstrate that a certain degree of locality is possible with our scheme, e.g., the head of the dinosaur can turn around or the arm can move without affecting

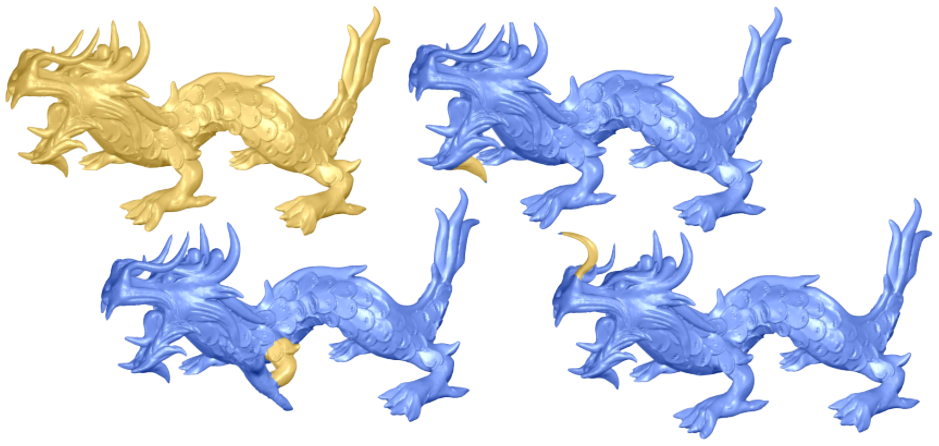


Figure 5.10: Local deformations of Chinese dragon in a 130-dimensional shape space. The reference model is shown on the top-left.

other parts of the body. But Figure 5.10 also shows (left image of the bottom row) that local deformations can introduce artifacts, in the shown example the mouth opens and lower jaw increases in size when the horn below the mouth is edited. Creating a method that integrates local and global edits by extending the subspace basis on the fly is still an open problem. For reduced simulations, a method that extends a reduced space during runtime, e.g., to represent collisions more accurately, was recently proposed by Harmon and Zorin [2013]. An alternative approach would be to integrate local editing methods, e.g., *PriMo* or *as-rigid-as-possible*, to seamlessly switch between modeling of details, and global modeling that preserves these detail edits. A benefit of our method for such an integration is that the same energy can be used for both local and global editing.

Chapter 6

Modal shape analysis beyond Laplacian

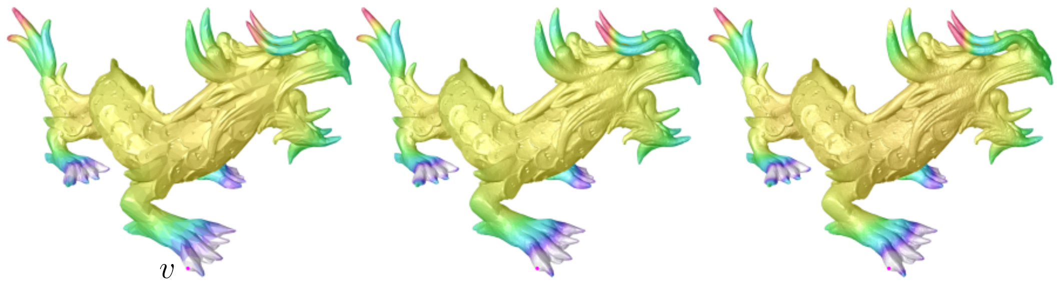


Figure 6.1: The vibration distance on the Chinese dragon model in various resolutions: 10k, 50k, and 130k vertices. Distance to vertex v in continuous coloring: from white being similar to red being dissimilar. Results are visually close indicating that distances can be well-approximated on coarse meshes.

6.1 Introduction

The spectrum and the eigenfunctions of the Laplace–Beltrami operator of a surface have stimulated much recent work in shape analysis and geometry processing ranging from parametrization, segmentation, and symmetry detection to shape signatures and mesh filtering. Such methods profit from properties of the eigenfunctions of the Laplace–Beltrami operator. For example, on a curved surface they form an orthogonal basis of the space of L^2-functions on the surface. Furthermore, the Laplacian depends only on the metric of the surface, hence the eigenvalues and eigenfunctions are invariant under isometric deformations of the surface. However, there are disadvantages as well. For example, a consequence of the invariance to isometric deformations is an insensitivity to extrinsic features of the surface, like sharp bends, that are essential for some applications.

In this chapter we investigate operators, whose eigenmodes and spectra can serve as alternatives to the spectrum and modes of the Laplacian for applications in geometry processing and shape analysis. On the one hand, the eigenfunctions of these operators share properties with the eigenfunctions of the Laplacian, *e.g.*, they form an orthogonal basis of a space of variations of the surface. On the other hand, there are fundamental differences, *e.g.*, these eigenfunctions depend (not only on intrinsic quantities but also) on the extrinsic curvature of the surface. In particular, we consider the linear vibration modes and frequencies given as the eigenmodes and eigenvalues of the Hessian of a deformation energy.

On a planar domain, the eigenfunctions of the Laplacian serve as a model for the vibration modes of a flat plate (Chladni plates). For curved surfaces more elaborate models are required to describe the vibration modes of a surface. We consider a physical model that describes vibration modes of a simplicial surface mesh as introduced in Chapter 2.

As application of the vibration modes, we propose a (multi-scale) signature, the *vibration signature*, to which we associate a (multi-scale) pseudo-metric on the surface. The resulting *vibration distance* can be used as a similarity measure on the surface. We prove a lemma that relates the vibration signature to the linearized gradient flow of the deformation energy (see Section 2.3). This gives further insight on the choice of the weights for the signature and reveals the geometry behind the signature.

Related work

Recently, we have seen a boom of papers that use the eigenvalues and eigenfunctions of the Laplace–Beltrami operator as an ingredient in algorithms in geometry processing and shape analysis. An overview of this development can be found in the survey by Zhang et al. [2010] and in the course notes of a

Siggraph Asia 2009 course held by Lévy and Zhang [2009]. Here, we briefly outline the work that has been most relevant to our approach.

The spectrum of the Laplace–Beltrami operator of a Riemannian manifold contains a significant amount of information about the manifold and the metric. Though it does not fully determine the Riemannian manifold, it can be used as a powerful shape descriptor of a class of isometric Riemannian manifolds. Reuter et al. [2005, 2006] use the spectrum of the Laplace–Beltrami operator to construct a fingerprint of surfaces, which they call the *Shape-DNA*. By construction this fingerprint is invariant under isometric deformations of a surface. Among other applications the Shape-DNA can be used for shape matching, copyright protection, and database retrieval. Rustamov [2007] developed the *Global Point Signature (GPS)*, a signature that can be used to classify shapes up to isometry. Based on the GPS, Ovsjanikov et al. [2008] developed a method for the detection of global symmetries in shapes. Dong et al. [2006] present an elegant technique that uses the Morse–Smale complex (and the quasi-dual complex) of a carefully chosen Laplace eigenfunction to generate a coarse quadrangulation of a surface mesh. This approach was extended by Huang et al. [2008], who design a least-squares optimization routine that modifies the selected Laplace eigenfunction (and hence its Morse–Smale complex) and provides the user with control of the shape, size, orientation, and feature alignment of the faces of the resulting quadrangulation. The computation of the spectrum and eigenfunctions of the Laplacian is a delicate and computationally expensive task, even for medium sized meshes. Vallet and Lévy [2008] propose an efficient shift-and-invert Lanczos method and present an implementation that is designed to handle even large meshes. Using the eigenfunctions of the Laplacian, one can compute the heat kernel of the surface. Sun et al. [2009] propose the heat kernel signature, a surface signature based on the heat kernel which they use to derive a measure for the geometric similarity of different regions of the surface. Due to its construction, this measure is invariant under isometric deformations of the surface. Independent of this work, Gebal et al. [2009] propose a similar signature, named the *Auto Diffusion Function*, and use it for mesh skeletonization and segmentation. In the context of shape matching and retrieval, [Dey et al., 2010] use persistent extrema of the heat kernel signature to construct a robust and efficient pose-oblivious matching algorithm for 3D shapes. Given a corresponding pair of points on two shapes, [Ovsjanikov et al., 2010] use the heat kernel to construct an isometric map between the shapes which allows them to find intrinsic symmetries and match partial, incomplete or isometrically deformed shapes.

Modal analysis is a well-established technique in structural mechanics and mechanical engineering that aims at computing the modes and frequencies of an object during vibration. In graphics, it is mainly used to speed up physical simulations (see [Pentland and Williams, 1989; Hauser et al., 2003; Barbič and James, 2005; Choi and Ko, 2005]). Recently, Huang et al. [2009] use vibration modes of a surface to decompose it into physically meaningful parts. They compute the modes of the surface from the Hessian of a simplification of the *as-rigid-as-possible* deformation energy, which was proposed by Sorkine and Alexa [2007].

In physical simulation, thin shell models describe the dynamics of a thin flexible structure that has a curved undeformed configuration. For example, in cloth simulation thin shells are used to describe folds and wrinkles [Bridson et al., 2003]. Common discrete models [Baraff and Witkin, 1998; Bridson et al., 2003; Grinspun et al., 2003; Garg et al., 2007] describe the middle surface of a thin shell by a mesh and measure the bending of the surface at the edges of the mesh. Of particular interest for this work is the model of Grinspun et al. [2003] that uses a discrete energy to simulate thin shells (see Section 2.2 for details).

6.2 Vibration signature

In this section we introduce the *vibration signature*: a multi-scale surface signature based on the vibration modes of the surface with respect to the *Discrete Shells* energy. The construction of the vibration signature follows the construction of the heat kernel signature defined in [Sun et al., 2009].

The modal signatures we consider are multi-scale signatures, which take a positive scale parameter t as input. For every t such a signature is a function on the mesh, i.e., it associates a real value to every vertex of the mesh. Let v be a vertex of a simplicial surface mesh with reference configuration $u \in \mathcal{C}$ and let t be a positive value. Then, we define the *vibration signature* of u at vertex v and scale t by

$$S_t^{Vib}(v) = \sum_j e^{-\lambda_j t} \left\| \Phi_j(v) \right\|_{\mathbb{R}^3}^2 , \tag{6.1}$$

where λ_j and Φ_j denote the eigenvalues and the L^2-normalized vector-valued vibration modes of a configuration u with respect to the *Discrete Shells* energy. The value $\|\Phi_j(v)\|_{\mathbb{R}^3}$ describes the displacement of the vertex v under the L^2-normalized vibration mode Φ_j. For a fixed t the vibration signature of v measures a weighted average displacement of the vertex over all vibration modes, where the weight of the j^{th} eigenmode is $e^{-\lambda_j t}$. The weights depend on the eigenvalues and on the scale parameter. For increasing λ, the function $e^{-\lambda t}$ rapidly decreases, *e.g.*, the modes with smaller eigenvalue receive higher weights than the modes with large eigenvalues. Furthermore, for increasing t all weights decrease, and, more importantly, the weights of the vibration modes with smaller eigenvalues increases relative to the weights of the modes with larger eigenvalues.

In addition to Equation (6.1), the vibration signature can be computed using the linearized gradient flow, which we defined in Equation (2.15). Before we explain this point of view in the next lemma, we introduce some notation. Let $\{b_1, b_2, b_3\}$ be the standard basis of $\mathbb{R}^3$ and $\alpha \in \{1,2,3\}$. We denote by δ^v_α the vector field in $T_u\mathcal{C}$ that satisfies

$$\langle \delta^v_\alpha, \Phi \rangle_{L^2} = \langle \Phi(v), b_\alpha \rangle_{\mathbb{R}^3} . \tag{6.2}$$

for all $\Phi \in T_u\mathcal{C}$. Explicitly δ^v_α is given by

$$\delta^v_\alpha = M^{-1} e^v_\alpha,$$

where e^v_α is the vector field on u that satisfies $e^v_\alpha(v) = b_\alpha$ and vanishes at all other vertices. If a diagonal mass matrix is used, then $\delta^v_\alpha = \frac{1}{m_v} e^v_\alpha$. Here m_v is the diagonal entry of the mass matrix corresponding to the vertex v; in the case of the lumped mass matrix, m_v is a third of the combined area of all triangles adjacent to v.

Lemma 1 *Let v be a vertex of u, and let $x_1(t)$, $x_2(t)$, and $x_3(t)$ be the solutions of the linearized gradient flow equation (2.15) with initial values $x_\alpha(0) = \delta^v_\alpha$. Then, the vibration signature satisfies*

$$S^{Vib}_{2t}(v) = \sum_{\alpha=1}^{3} \| x_\alpha(t) \|^2_{L^2} \tag{6.3}$$

for all t.

Proof. Using the vibration modes Φ_j we can derive the following explicit representation of the functions $x_\alpha(t)$:

$$x_\alpha(t) = \sum_j \langle \delta^v_\alpha, \Phi_j \rangle_{L^2} e^{-\lambda_j t} \Phi_j = \sum_j \langle \Phi_j(v), b_\alpha \rangle_{\mathbb{R}^3} e^{-\lambda_j t} \Phi_j,$$

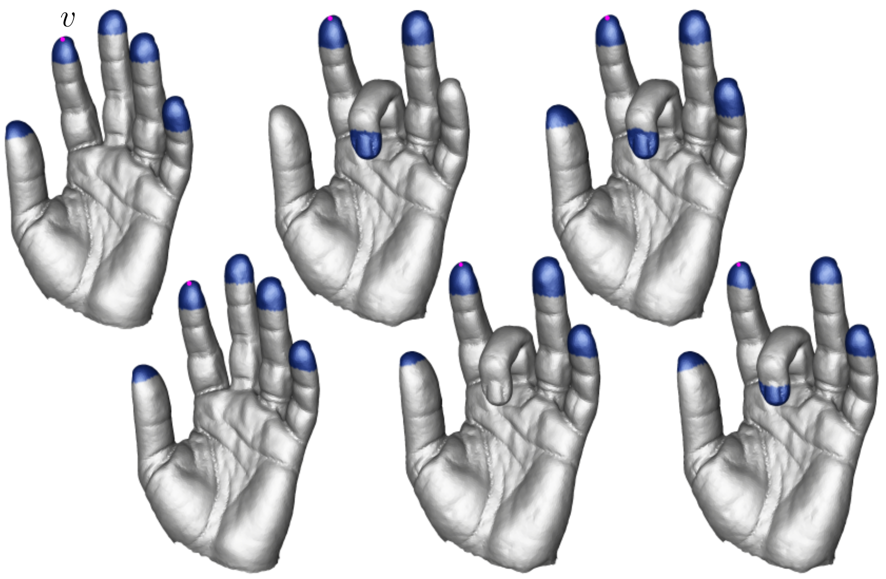


Figure 6.2: Vertices (blue) similar to vertex v based on heat kernel signature [Sun et al., 2009] (top row) and our vibration signature (lower row). Left and right column depict similarity based on a small range of t's and middle column on a large range of t's.

where we use Equation (6.2) in the last equality. Then, from the property that the vibration modes Φ_j are orthonormal, we get

$$\sum_{\alpha=1}^{3} \|x_\alpha(t)\|_{L^2}^2 = \sum_j e^{-2\lambda_j t} \sum_{\alpha=1}^{3} \langle \Phi_j(v), b_\alpha \rangle_{\mathbb{R}^3}^2 = \sum_j e^{-2\lambda_j t} \|\Phi_j(v)\|_{\mathbb{R}^3}^2 = S_{2t}^{Vib}(v),$$

which proves the lemma. ■

We can interpret the initial value condition $x_\alpha(0) = \delta_\alpha^v$ as an initial deformation of the surface. Then $x_\alpha(t)$ describes how this deformation evolves under the linearized gradient flow; for $t \to \infty$, the surface reaches the rest state of the energy. The signature measures for every t the sum of the L^2-norms of $x_1({}^t/_2)$, $x_2({}^t/_2)$, and $x_3({}^t/_2)$. The signature is independent of the choice of an orthonormal basis $\{b_1, b_2, b_3\}$ in $\mathbb{R}^3$ (which determines the initial value conditions). The lemma gives a motivation for the choice of the weights $e^{-\lambda_j t}$, that appear in the definition of the vibration signature. Furthermore, the lemma shows that one can compute the signature without computing the eigenmodes and spectrum first. An algorithm based on Equation (6.3) would not be efficient for our purposes, since we would need to solve the diffusion equation for every vertex of the mesh. Still, if the goal is to evaluate the signature only at certain vertices

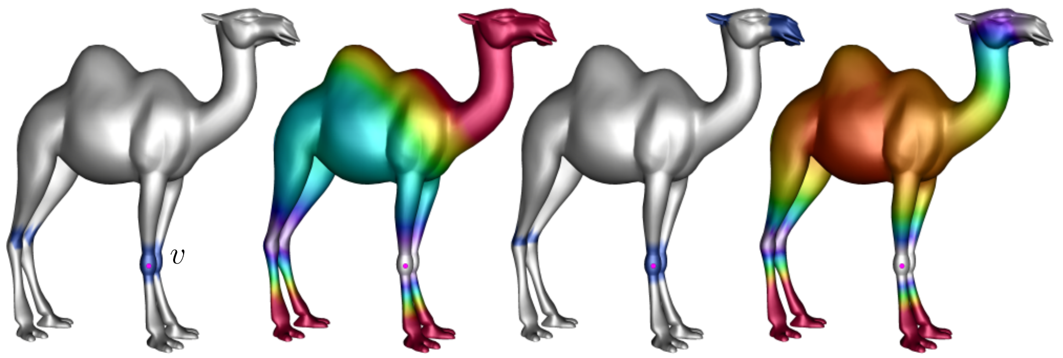


Figure 6.3: Comparison of two similarity measures. Distance to vertex v in binary as well as continuous coloring based on our vibration signature (left most) and the heat kernel signature (right most).

or for small time values, a computation based on Equation (6.3) can be more effective than a scheme that involves solving the eigenvalue problem.

Multi-scale distance From the vibration signature we can construct the following (multi-scale) pseudo-metric on the mesh: let v, $\tilde{v}$ be vertices of the mesh u, then we define

$$\delta_{[t_1,t_2]}(v,\tilde{v}) = \left(\int_{t_1}^{t_2} \left(\frac{S_t^{Vib}(v) - S_t^{Vib}(\tilde{v})}{\sum_k e^{-\lambda_k t}} \right)^2 \mathrm{d}\log t \right)^{\frac{1}{2}}. \tag{6.4}$$

By construction, for any pair of scale values $t_1 < t_2$, $\delta_{[t_1,t_2]}$ is positive semi-definite and symmetric, and one can show that it satisfies the triangle inequality. We call this pseudo-metric constructed from the vibration signature S_t^{Vib} the *vibration distance.*

The idea behind the construction of the pseudo-metric is to use the integral $\int_{t_1}^{t_2} (S_t^{Vib}(v) - S_t^{Vib}(\tilde{v}))^2 \mathrm{d}t$. However, the actual definition additionally includes two heuristics. First, since for increasing t the values $S_t^{Vib}(v)$ decreases for all v, we normalize the value $S_t^{Vib}(v) - S_t^{Vib}(\tilde{v})$ by dividing it by the discrete L^1-norm of S_t^{Vib},

$$\left\| S_t^{Vib} \right\|_{L^1} = \sum_k e^{-\lambda_k t}.$$

Second, for a fixed vertex v, the signature $S_t^{Vib}(v)$ varies more for small values of t compared to large t. To increase the discriminative power of the pseudo-metric, we associate a higher weight to the small t and a lower weight to the

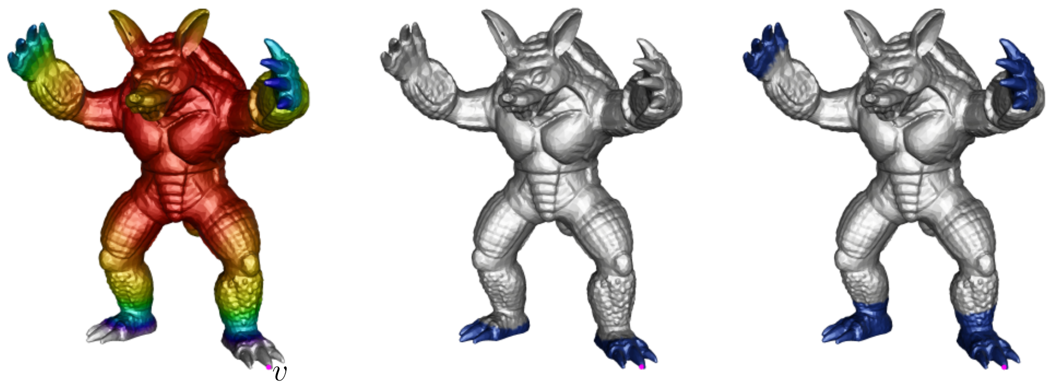


Figure 6.4: Vibration distance to the marked vertex v of the Armadillo model in three colorings: continuous coloring from white being similar to red being dissimilar to v and binary colorings with two different thresholds where blue vertices are similar to v.

larger t. We achieved this by using a weighted integral with weight function $\mathrm{d}\log t = \frac{1}{t}\mathrm{d}t$. This is a Riemann–Stieltjes integral that we evaluate numerically based on a uniform decomposition of the logarithmically scaled interval $[t_1, t_2]$.

6.3 Results and discussion

We experiment with the vibration modes of the *Discrete Shells* energy as well as, for comparison, the eigenfunctions of the *cotan*-Laplace operator [Pinkall and Polthier, 1993]. As a discrete L^2-scalar product we use the diagonal (or lumped) mass matrix M. The diagonal entry in the i^{th} row of the matrix is a third of the combined area of the triangles adjacent to the i^{th} vertex of the mesh. To compute the eigenmodes of a mesh, we solve the generalized eigenvalue problem (2.9). Since M is a diagonal matrix, this problem can be transformed into a standard eigenvalue problem as described in [Vallet and Lévy, 2008]. Then, we solve the resulting standard eigenvalue problem with the shift-and-invert Lanczos scheme described in [Saad, 1992]. For most examples and applications we do not need to compute the full spectrum, but only the lower part of the spectrum.

Figure 2.3 shows eigenvibrations with respect to the *Discrete Shells* energy. The images on the left (top and bottom row) show the reference mesh and each of the other images visualizes a vibration mode. The *Discrete Shells* energy is a weighted sum of a flexural and a membrane energy. If we decrease the weight of the membrane energy, i.e. κ_{mem} in Equation (2.2), the resulting vibration modes include stretching and squashing of the surface (see Figure 2.3 top row second and third images). In contrast, if we put a large weight on the membrane energy, the resulting eigenmodes try to preserve the metric. Examples of such modes are given in Figure 2.3 top row fourth image, bottom row second and third images.

To examine the properties of the vibration signature S_t^{Vib} defined in Equation (6.1) we compare it to the heat kernel signature (HKS) introduced in [Sun et al., 2009]. As noted in Section 6.2, $S_t^{Vib}(v)$ encodes the vibration behavior of a vertex v on multiple scales, i.e., vertices that oscillate with similar intensity throughout the eigenmodes will be close in terms of the vibration distance $\delta_{[t_1,t_2]}(\cdot,\cdot)$. We illustrate this property in Figure 6.4 for the Armadillo model (16k vertices). On the left we color plot the vibration distance $\delta_{[t_1,t_2]}(v,\cdot)$ to the marked vertex v. Two further binary colorings are given, colorizing vertices that are closer to v than a threshold in blue and the other vertices in white. For a small threshold the vertices on both feet are close to v; increasing the threshold causes parts of the hands to be colored in blue as well.

Figure 6.2 provides another comparison of S_t^{Vib} to the HKS. Every image of the hand model (40k vertices) depicts the vertices that are closer to the marked vertex v. In the first column similar results are achieved for HKS and S_t^{Vib}. Since the HKS is constructed using the spectrum and eigenfunctions of the Laplacian, the signature depends only on intrinsic properties of the surface. Thus the signature is incapable of differentiating between isometrically deformed parts of a surface. The vibration signature however is sensitive to extrinsic information and hence represents an alternative to the HKS. This characteristic of S_t^{Vib} becomes especially apparent in the second column of Figure 6.2. Here the middle finger of the hand is almost isometrically deformed. The HKS cannot distinguish this situation from the undeformed one; hence it recognizes the tips of the three longest fingers of the hand as similar to vertex v. As the deformation alters the vibration behavior of the bent finger, S_t^{Vib} detects only the tips of the unbent ones. Like the HKS, the vibration distance can be evaluated at different scales (different choices of $[t_1, t_2]$). Choosing smaller values of t increases the weights (cf. Equation (6.1)) for eigenmodes with higher frequency. Therefore, more local vibrations described by these eigenmodes contribute more to the vibration distance. An example is shown on the right side of the lower row of Figure 6.2. For smaller t, $\delta_{[t_1,t_2]}(v,\cdot)$ captures vibrations of the fingertips as well and thus classifies the vertices on all tips as similar to v.

In Figure 6.3 we provide the final comparison of the vibration signature and the HKS for the camel model (10k vertices). The vibration distance shown on the left, finds both pairs of knees (at the forelegs and at the hind legs) to be the closest to vertex v. For the HKS, shown on the right, the results are not as intuitive: the vertices at the mouth and ears of the camel are closer to the vertex v than the vertices at the hind legs, even closer than the vertices at the knees of the hind legs. This behavior of the HKS was the same at different scales and it is difficult to interpret the results. An indication for this behavior can be found by inspecting the Fiedler vector, which is the eigenfunction of the discrete Laplacian associated to the lowest (non-zero) eigenvalue. Of all eigenfunctions, this one gets the highest weight in the heat kernel distance. On the camel model, the Fiedler vector has one type of extrema, e.g., its minima, at tips of the toes of the hind legs at the tip of the tail and the other type of extrema, e.g., its maxima, at the tips of the toes of the forelegs, at the tips of the ears, and the tip of the snout. The function values at the tips of the ears and the tip of the snout are about the same as the function values at the knees of the forelegs. Hence, the contribution of this eigenfunction to the heat kernel distance is almost zero. This behavior repeats at some of the higher modes.

Implementation details and timings The computation of the eigenmodes and eigenvalues of an energy splits in two steps: setting up the Hessian matrix and solving the eigenproblem. To compute the signatures and distances, we additionally need to evaluate formulae (6.1) and (6.4). The time required for the construction of the Hessians is negligible compared to solving the sparse generalized eigenvalue problem. One way to solve such a generalized eigenproblem is to transform it into a standard eigenvalue problem. In our case the mass matrix is a positive definite diagonal matrix. Therefore such a basis transformation requires only rescaling of the Lagrange basis vectors. Details for this procedure can be found in [Vallet and Lévy, 2008]. To compute the signatures and distance, only a fraction of the lower part of the spectrum is required, because the weights $e^{-\lambda_j t}$ rapidly decrease with increasing eigenvalue. Typically the first 300 eigenvalues and modes yield a faithful approximation of the signatures and distances. To efficiently compute a lower portion of the spectrum and its corresponding eigenvectors we employ the shift-and-invert Lanczos method which does not need the inverse matrix explicitly. Instead only a matrix vector product has to be provided which can be evaluated by solving a linear system of equations. We solve these systems using the sparse direct solver implemented in *MUMPS* (see [Amestoy et al., 2001]). Once the eigenvalues are computed, the evaluation of the signatures and distances is relatively fast. To discretize the integral in Equation (6.4), we use a numeric quadrature. We place the samples of the interval $[t_1, t_2]$ such that they are equidistant on the logarithmic scale, which yields equal weights for all points in the quadrature.

In our experiments, we found that coarse meshes already provide a good approximation of the eigenmodes of the energy. Hence, for applications where computation time is crucial, it seems reasonable to first sub-sample the mesh and to compute the spectrum and eigenvectors of the simplified model. Figure 6.1 shows similarity results of the vibration distance for the Chinese dragon model at different mesh-resolutions. Although the model is simplified significantly, the similarity results still resemble the results of the fine mesh, indicating that the signature and the distance can be well-approximated on coarse meshes. Table 6.1 provides timings for computing the vibration signature of different versions of the Chinese dragon model.

Model	#Vertices	Hessian	Eigenproblem	S_t^{Vib}
Ch. dragon (Fig. 6.1, left)	10k	1 s	58 s	4 s
Ch. dragon (Fig. 6.1, middle)	50k	4 s	326 s	30 s
Ch. dragon (Fig. 6.1, right)	130k	11 s	1122 s	100 s

Table 6.1: **Timings** for the Chinese dragon model measured on a 2010 MacBook Pro with a 2.66GHz CPU. From left to right: model, number of vertices, timings in seconds for constructing the Hessian, solving the eigenproblem, and computing the vibration signature at all points.

Future work Linear vibration modes provide a quality basis for small deformations away from the reference configuration and our results in the preceding chapters show their utility for the dimensional model reduction for dynamical systems. In addition, the vibration modes and more generally operators derived from Hessians of corresponding deformation energies hold the promise of spawning novel methods in shape analysis and geometry processing. Explicitly, we started experimenting with the vibration modes for the problem of spectral quadrangulations. In contrast to the modes of the Laplacian, the modes of the energies considered in this work are sensitive to the extrinsic curvature of the surface. Our goal is to produce quadrangualtions that align with salient surface features.

Chapter 7

Compression of adaptive multiresolution meshes

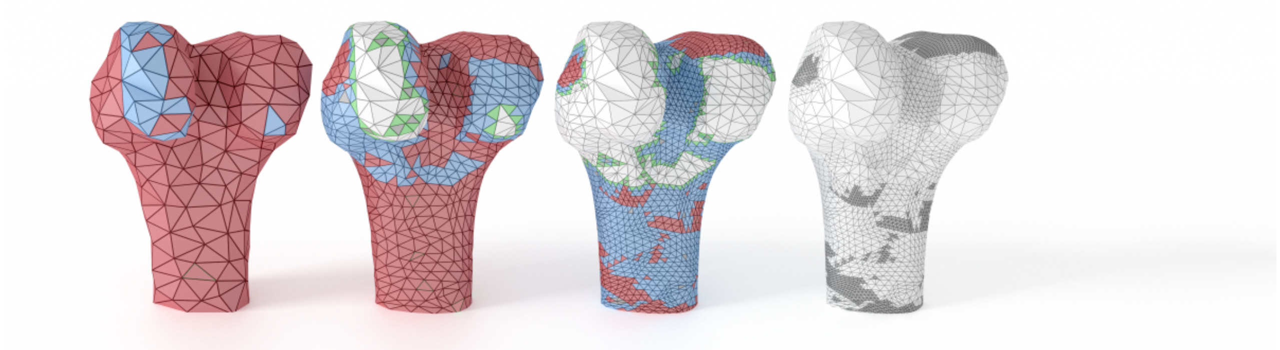


Figure 7.1: Adaptive refinement of a bone model. Elements are colored according to our coding scheme. We store one bit for each blue and red triangle, specifying whether it is refined or not. These bits are sufficient to reconstruct the connectivity of the model. All other triangles can be reconstructed using rules of the refinement scheme.

7.1 Introduction

Multiresolution meshes, i.e., meshes obtained through successive subdivision of a coarse base complex, are commonplace in a variety of areas such as the movie industry, computer aided design and in numerical simulations. The computing power of today's computer systems and the availability of advanced modeling software make it easy to generate grids with up to several million vertices. Storing those meshes in a raw data format is notoriously expensive due to the sheer amount of data. This is where compression comes into play.

Adaptive refinement, i.e., the introduction of detail only where it is needed, is an essential strategy to master the processing, rendering, transmission and storage of such meshes. For uniform refinement, the connectivity of the mesh can be represented by a coarse base complex and the number of subdivision levels. In contrast, adaptively refined meshes exhibit a non-trivial hierarchical structure. To the best of our knowledge, the lossless compression of adaptive hierarchies has not been researched into before.

Generally, compression comes in two stages: The first is a lossy stage, where essential information of the input is extracted and negligible data is dropped. The data decimation is then followed by lossless encoding in which the remaining data is transcoded into a compact byte stream, typically using entropy coding like Huffman or arithmetic coding.

In view of mesh coding, the mesh data consists of connectivity, geometry, and possibly attribute data such as colors and texture coordinates. 3D mesh coders are often referred to as *lossless* if they preserve the original connectivity of the mesh, even if floating point data of coordinates and attributes are truncated to a fixed precision. This tolerance within the "lossless" category may be due to the fact that geometry data will never be free from errors, and errors introduced by truncating the least significant bits of a `float` value are often negligible compared to noise, discretization, and quantization errors during mesh acquisition.

Lossy mesh coders consider the connectivity of the mesh as auxiliary information that does not contribute to the shape of the model. In the same manner, tangential positions of vertices within the surface are regarded as negligible. The intention of those coders is usually the best reproduction of the shape with respect to some distance norm within a given byte limit. The mesh is often remeshed to a semi-regular mesh that allows wavelet analysis to compress the geometry data.

We consider connectivity compression a vital issue since the outcome of many algorithms from geometry processing and numerical analysis depend on an exact reproduction of connectivity—think of animation or morphing. In our work we assume that the data reduction has already taken place. Our input models are hierarchical models that are adaptively refined. We assume that important details have been carefully selected and negligible ones pave been pruned by some criterion, be it by preservation of shape, normals, visual impact, or by some numerical criteria. The use of a lossy black box encoder is prohibitive if no further detail should be lost. Such situations arise for example in optimal control problems with time-dependent PDEs where several frames with varying refinement structure have to be stored. In this case, the base mesh is stored just once, with different refinement stages for each time step. The storage of view-dependent refinements of objects in virtual environments creates a similar situation.

Related work

Numerous compression schemes for surface meshes have been developed for single-rate coding (compressing the whole mesh in a region-growing fashion) as well as progressive coding (encoding the model from coarse to fine). On the single-rate side Edgebreaker [Rossignac, 1999] and the method of Touma and Gotsman [1998] are the most prominent coders for triangle meshes which have spawned a wealth of variants and improvements. Among the best-performing variants for connectivity compression is the early-split coder of Isenburg and Snoeyink [2006] and the optimized Edgebreaker encoding of Szymczak [2002]. These coders profit from mesh regularity and are able to push the bit rate well below the Tutte limit [Tutte, 1962] of roughly 3.24 bits per vertex. Many triangle mesh coders have been generalized to polygonal meshes, such as Isenburg's method [Isenburg, 2002] which extends the Touma-Gotsman coder.

The FreeLence [Kälberer et al., 2005] and Angle Analyzer [Lee et al., 2002] algorithm exploit correlation between connectivity and geometry by accessing already encoded geometry data when encoding connectivity and vice versa, allowing it to push the bit rates below that of [Isenburg and Snoeyink, 2006] and [Szymczak, 2002]. While FreeLence is especially performant in the triangular case, Angle Analyzer outperforms it for quadrangular meshes.

For progressive transmission, models are often simplified or remeshed [Lee et al., 1998; Guskov et al., 2000] to generate a simple base mesh from arbitrary connectivity meshes. In this context, wavelet-based coding has proven itself as one of the most efficient approaches for compression. Wavelet transforms recursively construct lower resolution approximations of a given input model, decorrelating high- and low-frequency geometry data. The difference of the details to predictions based on the coarse data are stored as wavelet coefficients. These typically feature a smaller entropy than the original data, yielding superb compression rates.

Wavelet-based coding schemes exist for both irregular and semi-regular meshes. The first group is based on mesh simplification methods that progressively remove vertices which cause the smallest distortion. Bits are written to identify a vertex within the mesh as well as its geometric position in order to be able to reconstruct the original mesh. Prominent coders of this group are [Isenburg and Snoeyink, 1999; Alliez and Desbrun, 2001; Valette and Prost, 2004].

The best results for geometry compression, however, have been achieved for semi-regular meshes. Based on stencils from subdivision schemes, efficient wavelet transforms have been derived. The best known wavelet-based coder in this category is the progressive geometry compression (PGC) codec by Khodakovsky et al. [2000] adapting the renowned zerotree coding scheme [Shapiro, 1993] from image compression. A wealth of descendants have been proposed extending PGC to different types of meshes [Khodakovsky and Guskov, 2003], resolution scalability [Avilés et al., 2005] and efficient embedded quantization [Payan and Antonini, 2005]. However, these coders only aim at the compression of the geometry and do not allow lossless reconstruction of the connectivity even for meshes generated through adaptive refinement. (Although the normal mesh compression by Khodakovsky and Guskov [2003] is able to generate such meshes, the adaptivity is controlled by the coder, thus neglecting any original criteria.)

Adaptively refined hierarchies also play a major role in point cloud compression. In [Botsch et al., 2002] and many follow-ups, a bounding box is adaptively refined up to a certain degree, and the leaf cells at the finest level then represent the point positions. Although these methods also compress adaptive trees, we propose optimization strategies specialized for the situations inherent to surface meshes, such as non-trivial root domains, red-green conformity, and balanced refinement. On the contrary, point cloud compression schemes utilize characteristics of surface geometry. For instance, nearby points are expected to be nearly coplanar, as done in [Schnabel and Klein, 2006].

7.2 Hierarchy coding

In this section we explain how we encode the hierarchical structure of adaptive multiresolution meshes and their connectivity. First we will explain the concepts of adaptive triangular and quadrilateral refinement, before we outline our encoding procedure in Section 7.2.2. Sections 7.2.3 to 7.2.6 then elaborate on the details.

7.2.1 Adaptive refinement schemes

In the field of subdivision surfaces and FEM a variety of refinement schemes have been devised. Among them, the dyadic split (cf. Butterfly [Dyn et al., 1990] or Loop [Loop, 1987]) for triangle surfaces and the face split (cf. Catmull–Clark [Catmull and Clark, 1978]) for polygonal surfaces are widely spread. The dyadic split operation divides each triangle into four congruent subtriangles. First, new vertices are introduced at each edge midpoint dividing the edges into two. Connecting the three edge midpoints within each triangle then splits the triangle into four. Consequently, this scheme is often referred to as 1-to-4 split. The face split can be applied to elements of arbitrary degree. New vertices are inserted not only at edge midpoints but also at the center of the element. Then the vertices at the edge midpoints are connected to the center vertex, thus splitting an n-gon into n quadrilaterals. Figure 7.2 illustrates both the dyadic and face split refinements.

Unlike *global mesh refinement* where all elements in the mesh are subdivided to obtain a finer mesh, *adaptive* or *local mesh refinement* performs the split operation only for selected elements. The transition between elements of different refinement levels requires extra care since a new vertex is inserted on an edge of the refined element, but the coarse neighboring face still contains the edge undivided. These irregular vertices are called *hanging nodes.* To resolve the non-conforming situations between elements of various refinement grades, the adjacent unrefined elements must also be refined. To maintain locality of the conformization, non-conformal elements must be split without introducing additional hanging nodes (see Figure 7.3, top). In the finite element community, triangles introduced to resolve hanging nodes are called *green* triangles, whereas elements that are generated by the dyadic split are called *red* triangles (hence the name *red-green refinement* [Bank et al., 1983]). Several conformization schemes exist for quadrilateral hierarchies, some of them introducing non-quadrilateral elements. We implemented the scheme described

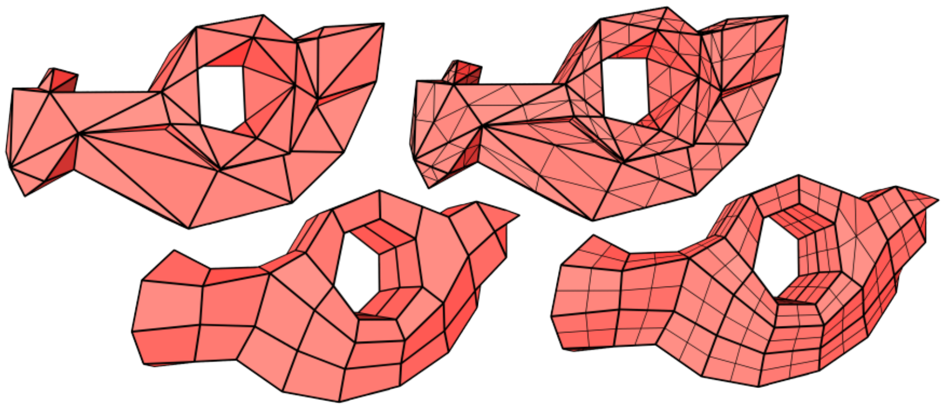


Figure 7.2: Uniform refinement of coarse domains using the dyadic split (top) and face split (bottom).

in [Settgast et al., 2004] (see Figure 7.3, bottom). Following the aforementioned terminology, elements inserted by these schemes will also be referred to as *green elements*.

Since green elements are less shape regular than their parents, the adaptive refinement is usually restricted to *balanced meshes* where the refinement level of neighboring elements must not differ by more than one level. This bounds the number of hanging nodes on each edge to one, preventing the refinement of green elements and therefore ever thinner faces.

The refinement strategy yields a canonical hierarchical structure where each split element acts as a parent for the new sub-elements. Therefore each element of the coarse base mesh will have an associated tree that specifies its refinement. We refer to the entities of the trees as *nodes* to underline the parental relationship between elements at different resolutions of the mesh. For a split element we assign the sub-element incident to the n-th vertex as the n-th child of the related node. In the triangle setting, the remaining central sub-element will be the fourth child. Figure 7.4 shows three root elements with associated refinement trees as well as the represented adaptive grid.

7.2.2 Encoding

Akin to existing progressive coders we separately encode the root grid from the hierarchy. Typically the root grid is described by a small, carefully laid out mesh that can be compressed well using single-rate coders. Our prototype uses

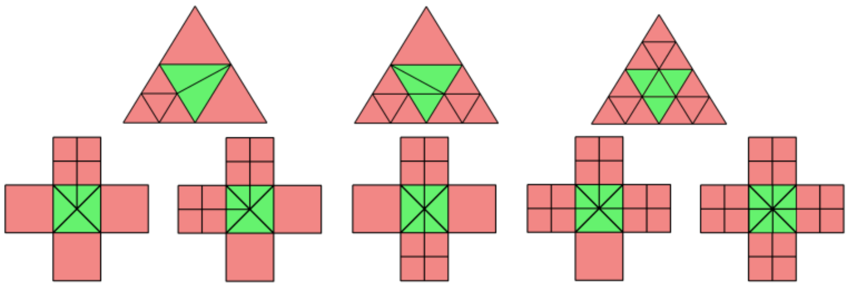


Figure 7.3: Conformization of hanging nodes. Up to symmetry, these are the only configurations of green elements that can occur in *balanced meshes*.

FreeLence [Kälberer et al., 2005] and PMC [Isenburg, 2002] to losslessly encode the triangular and quadrilateral root grids, respectively. This compression will generally alter the order of the base domain elements as well the as their local indexing, i.e., the ordering of references to vertices. To ensure the compatibility of the refinement hierarchy, the root grid and its associated refinement forest is matched to the reconstructed deterministic output of the FreeLence decoder so that refinement trees can be bijectively mapped to root elements without coding of further information.

Starting from the root grid, encoding the refinement hierarchy is sufficient to reconstruct the connectivity of the mesh at any level of detail. Provided that the encoder and decoder agree on a common node traversal strategy, the refinement hierarchy can be stored with one bit per node where each bit specifies whether or not the node has children (is refined). Exceptions are made when the conformization schemes leaves freedom of choice, e.g., triangles with two hanging nodes (see Figure 7.3). The other possibility to resolve this non-conforming situation arises by flipping the diagonal edge. In practice, the concrete conformization is often determined exclusively by local indexing. Since we change the local indexing during the compression of the base mesh, the original indexing is lost. Therefore we need to store additional symbols to determine the conformizations. For the triangle hierarchies these will be one bit per triangle with two hanging nodes. Using a conformization scheme that introduces only quadrilaterals (see e.g., [Schneiders, 1996]) at most one bit for each border between regions of different refinement must be coded for quadrilateral hierarchies. The conformization scheme of Settgast et al. [2004] on the other hand has no such ambiguities.

We entropy code these conformization bits, but found that they were virtually incompressible without knowing the exact implementation of the grid manager.

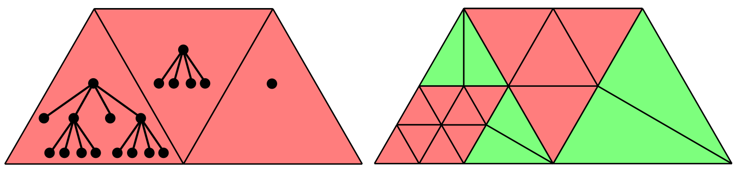


Figure 7.4: Base mesh with refinement trees and its corresponding red-green refined hierarchical mesh.

If, however, a geometric criterion is used to determine the conformizations, we can omit these bits altogether. The same is true if the application does not expect any green elements and the conformization is constructed on the fly, for example, the *Progressive Geometry Compression* software from Caltech [Khodakovsky et al., 2000].

Due to the deterministic conversion of the hierarchical structure to a bit stream we can estimate an upper bound for the code size. Let us begin with hierarchies generated by face splits. Except for the root level, each parent is split into four children. Therefore, the number of bits in the stream will sum to one bit per leaf plus one $\frac{1}{4}$ bit for their parents, a $\frac{1}{4^2}$ bit for their grandparents and so on until reaching a $\frac{1}{4^{d-1}}$ bit for the first level, where d is the depth of the leaf in the tree. Additionally, we have to add a $\frac{1}{4^{d-1}}\frac{1}{n}$ bit for the n-gon contained in the root level. We can write this sum as

$$\sum_{i=0}^{d-1}\frac{1}{4^i}+\frac{1}{4^{d-1}}\frac{1}{n}=\frac{4}{3}+\frac{1}{4^{d-1}}\left(-\frac{1}{3}+\frac{1}{n}\right)\leq\frac{4}{3}$$

bits per leaf. This bound also holds for the triangle hierarchies since they are composed entirely of quad trees so the n in the last term will be 4. Since the number of leaves in the hierarchy is no greater than the number f of elements of the finest resolution, our stream will contain no more than $\frac{4}{3}f$ bits. So far we did not account for symbols needed to resolve multiple possibilities of conformization. In these cases we have to store additional bits, but in fact, we already counted multiple green elements in f that where represented by just one node in the hierarchy.

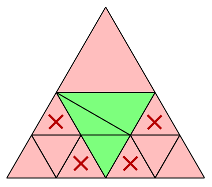
To maintain the progressivity of the generated bit code we traverse the hierarchy breadth-first, so that we can successively visit the nodes at a certain depth in all trees before switching to the next finer level. Finally, the generated bit stream is entropy encoded. In the following sections the algorithm is explained in more detail.

7.2.3 Redundant symbols

We can further improve the hierarchy compression by culling out nodes from the bit stream whose state can be implicitly reconstructed. As the hierarchy is directly related to the mesh, the mesh constraints implied by the refinement scheme are mirrored in the hierarchy's structure. These dependencies can be exploited by the following extensions.

Hanging nodes Some implementations of adaptive refinement prohibit more than a certain number of hanging nodes per element. In these setups, the hanging nodes are conformized by applying the usual split operation, introducing hanging nodes in adjacent coarse elements. During compression, if all neighbors of the current node are already processed, the number of hanging nodes within the element will be known to the coder. Hence, elements that exceed the maximum number of allowed hanging nodes can be split immediately and no symbol has to be coded. Anyhow, we do not need to treat these cases explicitly since they will be handled by our coder without overhead (cf. Section 7.2.4).

1-Regularity As mentioned before, adaptive refinement produces balanced meshes. There will be at most one hanging node per side of an element. Moreover, since the nodes of the hierarchy are conquered level-wise, we already know whether the neighbors of the node in question are green elements that resolved a non-conforming situation in the parent level. As a consequence, nodes representing faces adjacent to coarse green elements cannot be refined and can thus be skipped by the encoder.


Marked faces can't be further refined due to green faces in the parent level.

Uniform refinement Uniformly refined meshes exhibit a featureless hierarchical structure—the whole forest can be described by a single scalar that specifies the overall height of the refinement trees. Because many meshes in practice are uniformly refined to a certain degree, we exploit this property to reduce the number of code symbols. We store a single byte encoding the degree of uniform refinement separately, allowing the coder to skip all nodes on coarser levels.

Stream truncation Note that decoding a 0 from the stream has no effect on the hierarchy while a 1 causes a refinement of the current node (or its associated element, respectively). As the refinement forest is conquered in a breadth-first manner, nodes at the finest level are visited last, thus concluding the output stream. These are all 0 entries and are only needed for *closing* the forest, i.e., generating a valid hierarchical structure. This structure can be constructed by the decoder without using these entries. Therefore the encoder omits the finest nodes from the output and even truncates 0's written after the last 1 as these can be implied (cf. [Said, 2004]). The decoder thus simply skips nodes for which no bit was stored, i.e., the code contains no further symbols that can be read).

Model	f	#n	0s	1-Reg.	Unif.	#Left	Code Size
bones	5622	5k	8%	16%	0.0%	76% (4k)	2k (2k)
fandisk3	86092	96k	34%	23%	0.0%	43% (41k)	23k (41k)
feline	254044	303k	37%	16%	0.8%	46%(139k)	60k(139k)
femur	8944	11k	26%	13%	0.0%	60% (6k)	2k (5k)
heat_transfer	96412	115k	5%	15%	0.6%	79% (91k)	29k (83k)
horse	96368	113k	37%	18%	0.2%	46% (52k)	25k (51k)
rabbit	68506	79k	23%	21%	1.3%	55% (43k)	21k (43k)
venus	138672	155k	40%	24%	0.0%	36% (56k)	36k (50k)
blade	51157	36k	16%	36%	0.0%	49% (17k)	10k (17k)
fandisk4	24595	31k	24%	3%	0.0%	73% (23k)	1k (18k)
fertility	192635	129k	14%	37%	0.0%	49% (64k)	43k (64k)
shoulder	108361	77k	16%	34%	0.4%	49% (38k)	23k (38k)
rockerarm	30747	27k	30%	21%	6.8%	42% (11k)	5k (11k)
torso	54918	39k	12%	36%	0.7%	52% (20k)	12k (20k)
Average	72111	87k	23%	22%	0.8%	54% (43k)	21k (42k)

Table 7.1: **Removal of redundant symbols.** From left to right: number of faces, number of tree nodes, i.e., the number of binary decisions the decoder has to make, percentage of bits that can be omitted by dropping trailing zeros, exploiting 1-regularity, and storing the number of levels of uniform refinement, percentage (actual number) of bits that have to be coded, and the size of the compressed code in bits with (and without) the use of context groups.

Encoding the degree of uniform refinement in combination with the omission of trailing zeros guarantees that not a single symbol ends up in the output when a uniformly refined mesh is compressed. The results of the number of symbols that have to be coded for our benchmark models are shown in Table 7.1. Among the described optimizations, *stream truncation* and *1-regularity* perform best and contribute most to the reduction of symbols. Because *uniform refinement* only affects the coarsest levels only a few symbols are omitted for our adaptive test models. Overall, the number of symbols that have to be coded averages out to around half the number of nodes.

7.2.4 Context groups

With the steps in the previous section, we used the rules of the refinement scheme to eliminate code symbols in the cases where the refinement scheme leaves no room for choice. The steps above reduce the binary representation of the refinement tree to a compact, redundancy-free representation without even looking at the characteristics of the particular input model. Models that appear in practice, however, *do* show certain characteristics. Just like two adjacent pixels in a digital photograph are likely to be similar, the refinement grades in hierarchical meshes typically tend to be locally similar.

Luckily, our forest structure admits the definition of neighbors, which lets us easily determine the effects of locality. We call two nodes within one level of the hierarchy *adjacent* if their element counterparts in the mesh of that level share an edge. Due to the locality of the refinement depth, the split information of two adjacent nodes is highly correlated, so the refinement state of the neighbor node is a valuable hint. For instance, 23k of the 96k nodes of the fandisk model have children, which gives each hierarchy triangle the probability of 24 % of being split. Given the knowledge that a particular neighbor is a leaf, the probability of being subdivided drops to 7 %. If all three direct neighbors are leaves, that probability is a mere 1.2 %.

Let X be the binary stream of split data. As shown by Shannon [1948], the *entropy*

$$H(X) = \sum_{i=0}^{1} -p(i)\log(p(i)),$$

measured in bits per symbol, is the information content of X. It poses a lower bound for the code length of any encoding of the message X, where $p(0)$ and $p(1)$ are the probabilities of X being 0 or 1, respectively. Good entropy coders, such as arithmetic coding [Witten et al., 1987], approach this lower bound in the limit and are thus optimal in that sense.

If we do have additional information about X, for instance the state of the neighbor elements, the code length can be reduced. Let Y be an information source that is correlated to X (in our case $y \in Y$ describes a particular configuration of refinement states of the neighbor elements). The amount of information that actually has to be stored is measured by the conditional entropy

$$H(X|Y) = \sum_{y \in Y} p(Y{=}y) \sum_{i=0}^{1} -p(i|Y{=}y)\log p(i|Y{=}y),$$

that is, the amount of *new* information in X, given that we already know Y. If X and Y are correlated, $H(X|Y)$ is strictly less than $H(X)$.

In our implementation, we use *context groups* as a simple measure to take advantage of the correlation of hierarchy elements. Recall that we specify with one bit whether the current node is refined as we traverse the nodes in the trees level by level. Whenever a binary code symbol is produced, we check the status of the neighbors. If the neighbor has already been visited during traversal, its refinement status will be known to the decoder. Thus we can use the refinement status of already processed neighbors as the definition of contexts: a neighbor can either be *refined* (⟁), *not refined* (△), or it has not been processed before (?).

The status of the neighbor elements define the context group in which the symbol is encoded. We write symbols of different context groups in separate arrays, which are entropy coded independently. With arithmetic coding, each context group y will compress to

$$H(X|Y{=}y) = \sum_{i=0}^{1} -p(i|Y{=}y)\log(p(i|Y{=}y))$$

in the limit. The total code size per symbol is obtained by averaging the entropies individual contexts weighted by their respective probabilities,

$$\sum_{y\in Y} p(Y{=}y)H(X|Y{=}y) = H(X|Y),$$

which proves that contexts are an appropriate tool to capture *all* the mutual information that is inherent in the correlation of neighboring elements.

So far we have not specified exactly how we define the contexts. The contexts arise from the number of ⟁, △, and ? neighbors of a hierarchy node. We write (x, y, z) to denote the context with x ⟁ situations, y △ situations, and z ? situations. For the triangle hierarchy setting, all of the possible cases are listed in Table 7.2.

7.2.5 Traversal order

In this section we review the compression rates within the single context groups and discuss the impact of the hierarchy traversal strategy on them.

The level-wise traversal of the hierarchy is essential to a progressive coding. Still it leaves freedom to choose any iteration of nodes within each level. This

Group	Naïve Traversal			Improved Traversal		
(◬,△,?)	#symb	%1's	bits/symb	#symb	%1's	bits/symb
(0, 3, 0)	13606	2%	0.123	681	10%	0.517
(1, 2, 0)	5603	37%	0.954	30	50%	1.400
(2, 1, 0)	5913	82%	0.693	91	78%	0.945
(3, 0, 0)	12146	100%	0.003	685	100%	0.041
(0, 2, 1)	14314	6%	0.340	29789	5%	0.289
(1, 1, 1)	6373	43%	0.992	12079	72%	0.860
(2, 0, 1)	14130	99%	0.049	33227	98%	0.129
(0, 1, 2)	17256	10%	0.487	40996	23%	0.772
(1, 0, 2)	18520	94%	0.342	20944	96%	0.222
(0, 0, 3)	30855	55%	0.994	1	100%	5.000
Culled	164356		0	164549		0
Total	303072		0.23	303072		0.20

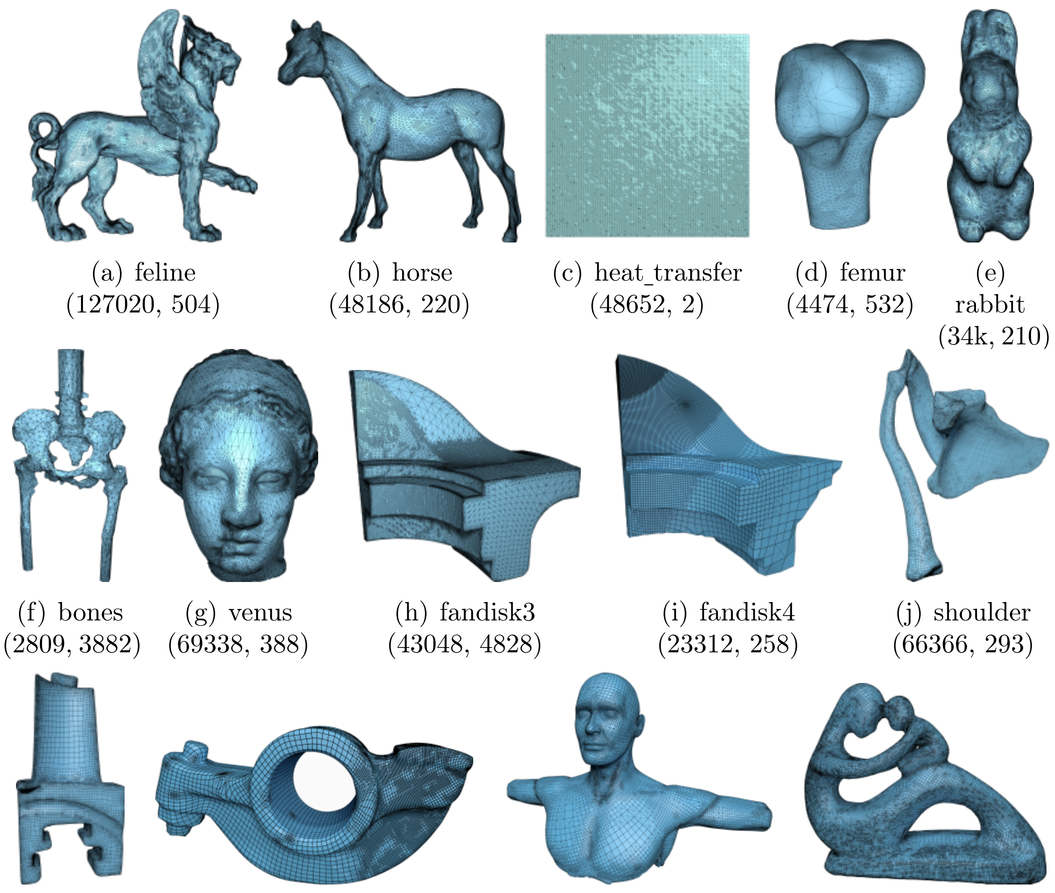
Table 7.2: Population of the context groups for the naïve and improved traversal strategy on the *feline* model. For each strategy we provided the number of symbols in each context group, the percentage of 1's among those symbols, and the efficiency in bits per symbol.

choice directly affects the distribution of the symbols over the context groups as the context of a node solely depends on the refinement state of its neighbors and therefore on the fact whether these have been already visited.

A customary traversal scheme would visit the children of each node in a fixed ordering. Table 7.2 shows the distribution of the symbols in each context group for one of our test models. Here *naïve traversal* refers to a strategy where children are visited in order.

In our implementation the elements incident to the parent's vertices are visited first. Additionally, local indexing of parent elements determines the order of its children. Hence context group $(0, 0, n)$, where none of the neighbors is known, contains the most entries. This group, though, is virtually incompressible as we cannot takte advantage of mutual information. The same holds for context groups where the extra information is rather ambiguous, e.g., $(1, 1, 1)$, $(1, 2, 0)$, and $(2, 1, 0)$ in the triangle setting. On the contrary, the other context groups perform very well but are less populated.

The positive effects of an adaptive traversal order have been observed in Angle-Analyzer [Lee et al., 2002] for surface meshes. Also, Schnabel and Klein [2006] optimize the octree traversal in each level. In this spirit, we designed a new



(a) feline (127020, 504)
(b) horse (48186, 220)
(c) heat_transfer (48652, 2)
(d) femur (4474, 532)
(e) rabbit (34k, 210)
(f) bones (2809, 3882)
(g) venus (69338, 388)
(h) fandisk3 (43048, 4828)
(i) fandisk4 (23312, 258)
(j) shoulder (66366, 293)
(k) blade (30866, 497)
(l) rockerarm (22171, 368)
(m) torso (33334, 265)
(n) fertility (113097, 265)

Figure 7.5: The test set used in our experiments (the number of vertices at the finest resolution, and the number of elements of the base mesh).

traversal scheme (*Improved Traversal* in Table 7.2) to redistribute the symbols and maximize the overall compression. Instead of ordering the children in a fixed manner, we first iterate over every tree and collect all nodes at a certain depth. This allows for a global optimization of the level-wise node traversal.

The principle of our algorithm is to maximize the mutual information that can be exploited for encoding each node. To this end, we prioritize each node by the entropy of its current context. Therefore, nodes that already profit from encoded neighbors will be conquered first, which in turn provides more information to its unprocessed neighbors. Clearly all nodes that are skipped by the coder due to optimizations from Section 7.2.3 feature a zero entropy and will hence be traversed before all other nodes. A promising candidate for prioritization of the remaining nodes is to use the actual entropies of the individual contexts. The greedy strategy traverses nodes in the context with lowest entropy first in order to minimize the overall compression rate. Experiments showed that this strategy already achieves significant improvements in compression. Learning the entropies of the contexts, however, is expensive in terms of compression performance as well as computational cost. Once the learning phase is settled, the algorithm sticks with a fixed prioritization of contexts. To remove all effects of the learning process of the contexts' entropies from the traversal, we additionally investigated fixed priorities for the nodes, i.e., a fixed order of contexts during the search for the next node to conquer.

We looked at different orderings:

- increasing by the learned (settled) entropies of contexts,
- increasing by the number of unknown neighbors, and
- decreasing by the difference of known coarse and known refined neighbors.

In the case of ties we also tried all possible permutations.

The tests revealed that we can substantially improve the performance of the traversal by skipping the learning process. Although we could not identify a single strategy that performed best on the entire set, ordering the contexts as in Table 7.2 (increasing by number of unknown neighbors and breaking ties increasing by the number of known refined neighbors) constantly provided good results. Therefore we chose this ordering as the default for our coder and for all the presented tests.

As a result of our choice, the context group $(0, 0, n)$ contains almost no entries—in fact, it will always be comprised of one symbol if the mesh represents a connected surface. The nodes are thus conquered in a region-growing manner, so nodes whose neighbors are all known become extremely rare (*cf.* group $(3, 0, 0)$, $(2, 1, 0)$, and $(1, 2, 0)$ in Table 7.2). Furthermore, the traversal directly affects the nodes' order which causes the change in the number of culled out symbols. Reviewing the final compression rates on our test models shows an average improvement of 7 %.

Model	#Frames	Average	Total	Diff.
bunny	21	585 (1002)	12280 (21046)	-42%
heat_transfer	11	420 (1125)	4621 (12376)	-63%
cloth	15	459 (472)	6888 (7085)	-3%

Table 7.3: Compression results for time-varying sequences. Left to right: Model, number of frames in the sequence, average code size per frame in bytes for dynamic (static) coder, total code size in bytes for dynamic (static) coder, and the difference in code size between dynamic and static coder.

7.2.6 Time-varying sequences

As we observed, we can profit from the correlations between adjacent elements. We can profit in the same way when we have a time-varying series of refinements of a common base mesh. Here, we assume that the degree of refinement varies only a little from one frame to another, just as one assumes smooth transitions for animated geometries or videos.

When processing a hierarchy node in a series, we query the state of the corresponding node in the previous time frame, which can give us one of three states:

- it was a node with children,
- it is a leaf node, or
- it didn't exist

in the previous frame. Thus, the number of contexts triples if we also include the status of the previous frame in the contexts.

If the refinement trees don't vary much between the time steps, then contexts corresponding to the first case will be mainly filled with ones, while the latter two will primarily contain zeros. Thus, grids which equal their preceding frame can be stored at no cost, aside from a small overhead due to the use of more contexts. On the contrary, if the grids are not correlated, the entropy of the individual contexts can never be worse than in the static setting, since symbols from one context are simply spread to three, maintaining or improving symbol possibilities. Table 7.3 shows the results of the time series adaption applied to three time-varying hierarchies.

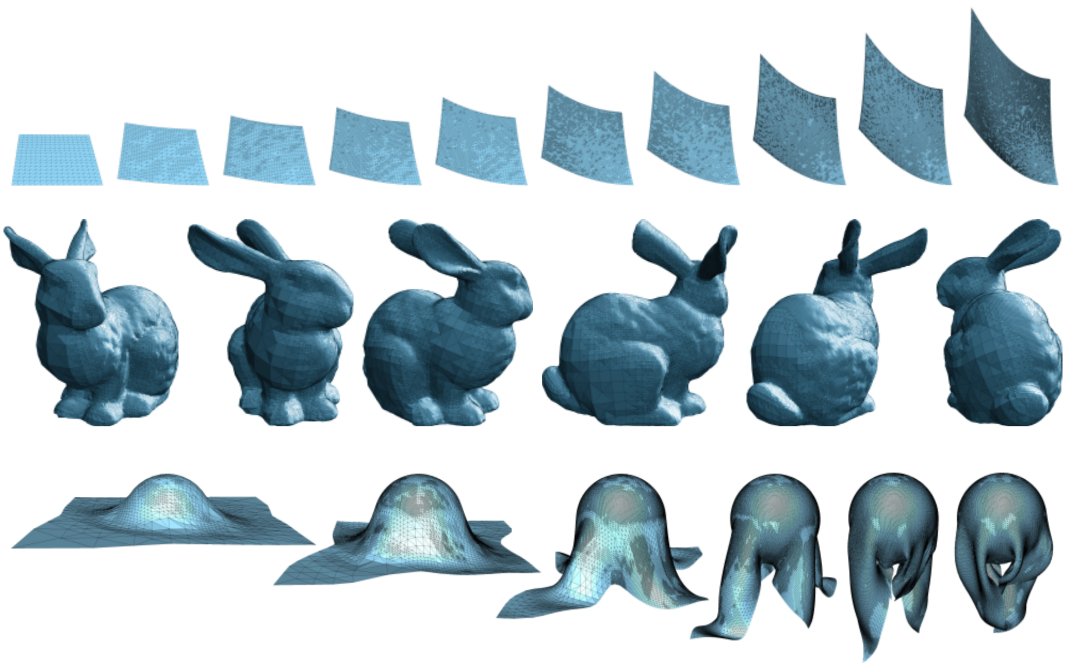


Figure 7.6: The three test sequences used. (Top) Planar domain refined driven by heat transfer computations (showing temperature as z-component). (Middle) View-dependent refinement taking curvature, visibility and silhouette into account (as proposed in [Settgast et al., 2004]). (Bottom) Cloth simulation with curvature-dependent refinement.

7.3 Geometry compression

Thus far, we presented strategies to encode the base domain of a mesh as well as the hierarchical structure inherent to the adaptive refinement. In this section we extend our coder to the compression of the geometry of adaptive hierarchies based on progressive coding schemes. First we will introduce the concept of multiresolution analysis for surfaces with arbitrary topology followed by a description of zerotrees as a method for the coding of wavelet decompositions. Then we present a context model for entropy coding to exploit interband and intraband dependencies between wavelet coefficients.

7.3.1 Multiresolution analysis

Multiresolution analysis and wavelets have proven to be powerful tools for use in numerical analysis and signal processing. Their power stems from their ability to decompose complicated functions into coarse approximations together with high-frequency details called wavelet coefficients. A multiresolution analysis of a measurable function f with finite energy defined over the domain M, i.e., $f \in L_2(M)$, consists of an infinite chain of nested linear function spaces $\{\varnothing\} = V^{-1} \subset V^0 \subset V^1 \subset \cdots \subset V^j \subset L_2(M)$, so that $\bigcup_{j\geq 0} V^j$ is dense in $L_2(M)$. Here, the subspace V^j contains the approximation of f at resolution j, with the approximation quality increasing as j increases. Given an inner product $\langle\cdot,\cdot\rangle$, we can define orthogonal complement spaces

$$W^j := \{f \in V^{j+1} \mid \langle f, g\rangle = 0,\ g \in V^j\},$$

that encode the differences between two successive levels of resolution. A crucial building block of the multiresolution analysis is the construction of the wavelets, i.e., determining basis functions ψ_i^j for the spaces W^j. Intuitively, we would like the wavelets ψ_i^j to be orthogonal to the basis functions $\varphi_{i'}^j$ of V^j. This condition, referred to as *semi-orthogonality*, ensures that when decomposing a function $f^{j+1} \in V^{j+1}$ into a low-resolution part $f^j \in V^j$ and a detail part $h^j \in W^j$, f^j is the unique function in V^j minimizing the least-squares residual $\langle f^{j+1} - f^j, f^{j+1} - f^j\rangle$. An even stricter condition, called *full orthogonality*, additionally requires the wavelets to form orthogonal bases for the complement space W^j, that is $\langle\psi_i^j, \psi_{i'}^j\rangle = 0$ unless $i = i'$. (Note, that orthogonality between wavelets of different resolutions already follows from the semi-orthogonality.) However, it is not always possible to construct (semi-)orthogonal wavelets that are also locally supported, thus causing obvious practical disadvantages. Hence, dropping orthogonality requirements entirely can often be more convenient. This less restrictive form of wavelets merely requires the spaces W^j to be some complement of V^j in V^{j+1} and is called *biorthogonal*.

Classical wavelet constructions are based on translates and dilates of one function and are thus limited to simple domains such as intervals and rectangles. However, such constructions, called *first generation wavelets*, can no longer be used in more general settings such as wavelets on bounded domains or, as in our case, wavelets on surfaces. To analyze data that live on surfaces of arbitrary topological type, we need wavelets that are intrinsically defined on the surface and are adapted to a measure on it. On the other hand we would like to keep all the desirable properties like local support in space and frequency. These goals can be met by omitting translation and dilation in the wavelet construction yielding the *second generation wavelets*.

Constructions of second generation wavelets exist for both irregular and semi-regular meshes. For the latter case, the pioneering work of Lounsbery et al. [1997] provides a theoretical foundation for developing multiresolution analysis based on surface subdivision schemes. For the adaptive hierarchies this construction is the canonical choice; subdivision schemes also start with a coarse control polyhedron M^0 and recursively generate finer meshes $M^1, M^2, \dots$ that converge to some limit surface M^∞. In each subdivision step, the geometry of M^{j+1} is computed by linearly blending the vertices of M^j, where the weights for the blending depend entirely on the connectivity of the mesh. Thus, we can write the subdivision step as

$$\mathbf{X}^{j+1} = \mathbf{P}^j \mathbf{X}^j, \tag{7.1}$$

where $\mathbf{X}^j$ is a matrix containing the x, y, and z coordinates of the vertex i in the ith row, and $\mathbf{P}^j$ is a non-square matrix containing the blending weights. Numerous subdivision schemes have been proposed and we refer the reader to the course notes of Zorin et al. [2000] for an overview of the most prominent schemes and their properties. Naturally, we require the subdivison scheme to be based on the same refinement operator used to generate the hierarchies (cf. 7.2.1). Furthermore, we need the scheme to be local, stationary, continuous, and uniformly convergent. For this setting, Lounsberry et al. show the existence of nested linear spaces $V^j(M^0)$ that are adapted to the root grid M^0 in that they consists of functions having M^0 as the domain. In particular, we can define the scalar basis functions for $V^j(M^0)$ to be

$$\varphi_i^j := \left(\lim_{s\to\infty} \prod_{k=j}^{s} \mathbf{P^k} \right) \mathbf{e}_i^j, \tag{7.2}$$

where $\mathbf{P}^k$ is as in Equation (7.1), and $\mathbf{e}_i^j$ is a vector containing 1 as the i-th entry and zero for all others. Since the subdivision scheme is required to be local and uniformly convergent for any choice of control points, the basis functions must exist and will exhibit the same properties, e.g. continuity or smoothness, as the limit surface. From Equation (7.2) we see that each of the basis functions φ_i^j can be written as a linear combination of the functions φ_i^{j+1}. Hence they are refinable. It is convenient to write this statement in matrix form as

$$\Phi^j = \Phi^{j+1} \mathbf{P}^j, \tag{7.3}$$

where Φ^j denotes the row matrix of basis functions φ_i^j. Thus, the linear spaces $V^j(M^0)$ associated with the mesh M^0 are indeed nested.

The next step in the construction is to determine the basis functions for the complement spaces, i.e., a set of wavelets $\Psi^j = \{\psi_1^j, \psi_2^j, \dots\}$ spanning $W^j(M^0)$.

A straightforward choice for the wavelets ψ_i^j is to use the subset $\mathcal{N}^{j+1}$ of basis functions in $V^{j+1}(M^0)$ which are associated to new vertices (introduced when subdividing M^j). More concretely, we define $\mathcal{N}^{j+1}$ to be:

$$\mathcal{N}^{j+1} := \{\varphi_i^{j+1} | i \in \mathcal{I}^{j+1} \setminus \mathcal{I}^j\},$$

where $\mathcal{I}^k$ is the set of vertex indices of mesh M^k and $\mathcal{I}^k \subset \mathcal{I}^{k+1}$. Similarly, we can denote the basis functions φ_i^{j+1} associated to old vertices (already present in M^j) as $\mathcal{O}^{j+1}$. This way we can write Φ^{j+1} in Equation (7.3) in block matrix form as $(\mathcal{O}^{j+1}\mathcal{N}^{j+1})$ and get

$$\Phi^j = (\mathcal{O}^{j+1}\mathcal{N}^{j+1}) \begin{pmatrix} \mathbf{O}^j \\ \mathbf{N}^j \end{pmatrix},$$

where $\mathbf{O}^j$ and $\mathbf{N}^j$ are the corresponding submatrices of $\mathbf{P}^j$. It is easy to show that $\mathcal{N}^{j+1}$ and Φ^j together span $V^{j+1}(M^0)$ if and only if $\mathbf{O}^j$ is invertible which is true for most primal subdivision methods. In fact, for interpolating subdivision schemes $\mathbf{O}^j$ is the identity matrix. Because $\mathcal{N}^{j+1}$ is in general not orthogonal to Φ^j, this basis will not produce best least-squares approximations. Lounsberry et al. therefore present two constructions to improve the approximation quality. The first is to orthogonalize the functions $\mathcal{N}^{j+1}$ by subtracting out their projection into $V^j(M^0)$. However, this leads in general to globally supported wavelets, causing the multiresolution analysis and synthesis to require quadratic time. The second construction enforces the wavelets to be locally supported and thus addresses the issue of linear time analysis and synthesis. To restrict the support of the wavelet ψ_i^j *a priori*, we require it to be a linear combination of $\varphi_i^{j+1} \in \mathcal{N}^{j+1}$ and some basis functions of $V^j(M^0)$:

$$\psi_i^j = \varphi_i^{j+1} + \sum_{i' \in \mathcal{I}_i^j} \alpha_{i',i}^j \varphi_{i'}^j, \tag{7.4}$$

where $\mathcal{I}_i^j \subset \mathcal{I}^j$ and the coefficients $\alpha_{i',i}^j$ are such that the least-squares norm of the projection of ψ_i^j into $V^j(M^0)$ is minimal. Typically $\mathcal{I}_i^j$ consists of indices of functions in Φ^j that have overlapping support with φ_i^{j+1}. Although this biorthogonal wavelet construction satisfies the linear time requirements (at least for interpolating subdivision schemes), the construction still requires the evaluation of inner products between all basis functions of $V^j(M^0)$ which can cause high preprocessing demands.

Schröder and Sweldens [1995] present an alternative construction extending the idea of Lounsberry et al. In Equation (7.4) we started with function $\varphi_i^{j+1} \in \mathcal{N}^{j+1}$ as a wavelet and improved its approximation properties by blending it with the basis functions of $V^j(M^0)$. This idea can be generalized by posing other constraints on the coefficients $\alpha^j_{i',i}$. Then we can construct biorthogonal wavelets with custom properties. For example, we can require the wavelet to have vanishing moments, i.e., the integral of that wavelet multiplied with a certain polynomial P_k is zero, by solving the equation

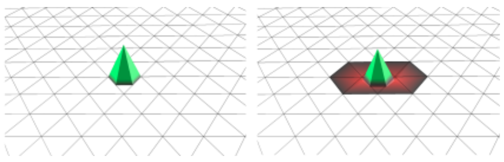
$$0 = \langle \varphi_i^{j+1}, P_i \rangle + \sum_{i' \in \mathcal{I}_i^j} \alpha^j_{i',i} \langle \varphi^j_{i'}, P_k \rangle. \tag{7.5}$$

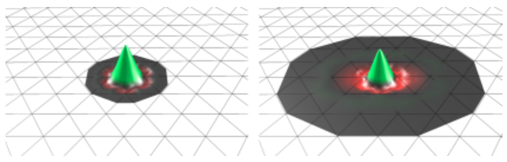
This process can be seen as "lifting" the simple wavelet φ_i^{j+1} to a more efficient one which is why it is referred to as the *lifting scheme*. In general, the lifting scheme can be applied to any basic multiresolution analysis to construct a more performant one. In the compression community it is common practice to use lifting to ensure that the wavelets have at least one vanishing moment, i.e., that the wavelets have a vanishing integral. Consequently, setting P_k in Equation (7.5) to the constant function, the inner products reduce to the integrals of the basis functions over M^0 and we can find suitable lifting coefficients as

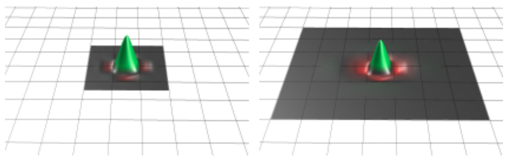
$$\alpha^j_{i',i} = \frac{\int_{M^0} \varphi_i^{j+1}}{|\mathcal{I}_i^j| \int_{M^0} \varphi^j_{i'}}, \tag{7.6}$$

where $|\mathcal{I}_i^j|$ is the number of basis function used to lift φ_i^{j+1}. This way the resulting wavelets will have a vanishing integral. In general, lifting does not improve the ability of the basis to compress, but it causes smaller errors in the decomposition of the signal and thus improves the rate-distortion curves. The presented construction already possesses the advantage that we no longer have to evaluate the inner products between all basis functions Φ^j. However, we still have to compute the integrals of all functions. Different schemes have been proposed in the literature, most of them resorting to numerical quadrature of the functions. Since most of the wavelet transforms are designed to analyze the geometry of surfaces of the same topological type as M^0 but not necessarily with the same metric, the transform is usually constructed based on a combinatorial inner product that equally weights all elements in M^0.

In our implementation, we compute the wavelet transform based on interpolatory subdivision schemes. For the triangle hierarchies we use the butterfly subdivision [Dyn et al., 1990] and for quadrilateral hierarchies we use the scheme in [Kobbelt, 1996]. For the construction of the wavelet ψ_i^j associated to the midpoint of the parent edge (v_k, v_l), we lift the function φ_i^{j+1} using the two basis functions associated to the endpoints of that edge, i.e., $\mathcal{I}_i^j := \{k, l\}$. Quadrilateral hierarchies additionally contain wavelets corresponding to vertices inserted in the midpoint of elements. In these cases, we define $\mathcal{I}_i^j := \{o, p, q, \dots\}$, where


(a) Linear wavelet.


(b) Butterfly wavelet.


(c) Kobbelt wavelet for edge midpoint.

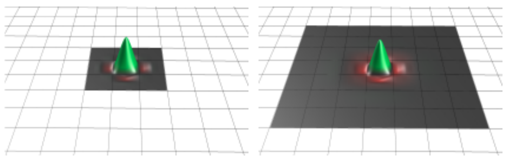

(d) Kobbelt wavelet for face midpoint.

Figure 7.7: Graphs of unlifted (left) and lifted (right) wavelets based on linear triangular (a), Butterfly (b), and Kobbelt's (c) and (d) subdivision scheme. The coarse control polyhedron M^0 is shown as wireframe. Elements in the support of the wavelet are colored according to function values (values increase as the color goes from red to gray to green).

$(v_o, v_p, v_q, \dots)$ is the parent element for vertex v_i. Due to our choice to use a combinatorial inner product there exists a recurrence relation between integrals of scaling functions at different resolutions. In particular, since triangle areas shrink by a factor of 4 under our subdivision schemes, we can compute the lifting weights in Equation (7.6) as $\alpha^j_{i',i} = 1/(4|\mathcal{I}^j_i|)$. Although this does not account for extraordinary points in the support of the scaling functions, we use these weights as an approximation to keep the computational cost for the transform at a minimum. Figure 7.7 shows graphs of wavelets for the different types.

Once the coefficients in level j are computed, we adjust their scaling by a factor of 2^{-j}. This scaling results from L_2-normalizing the subdivision basis functions. It is commonly applied in compression to compensate for their shrinking support on finer levels. Note that the scaling provides higher accuracy on coarser levels and causes stronger quantization for coefficients on finer levels.

7.3.2 Embedded coding of wavelet coefficients

The multiresolution analysis presented in Section 7.3.1 can be used to decompose a function $f \in V^{n+1}(M^0)$ into a lower resolution part in $V^n(M^0)$ and a detail part in $W^n(M^0)$:

$$f = \sum_{i \in \mathcal{I}^{n+1}} a_i^{n+1} \varphi_i^{n+1} = \sum_{i \in \mathcal{I}^n} a_i^n \varphi_i^n + \sum_{i \in \mathcal{I}^{n+1} \setminus \mathcal{I}^n} c_i^n \psi_i^n .$$

Applying this decomposition successively, we can represent f entirely in terms of the wavelet coefficients $\{c_i^j\}$. This representation facilitates compression since a great amount of correlation is removed during the transformation. In particular, the wavelet coefficients capture to which extent function values fail to be similar within a particular region. Therefore, the distribution of wavelet coefficients is usually centered around zero with their magnitudes decaying at finer resolutions. This characteristic of the coefficients is the key to progressive compression and allows for embedded coding.

The general principle of embedded coding is to encode the most important information—which yields the largest error reduction—first. This way, the decoder can stop the reception at any point and reconstruct the data with the least amount of distortion possible with that number of transmitted bits. Hence, an embedded code contains all lower rate codes "embedded" at the beginning of the bit stream.

Considering the error in the decoded function $\hat{f}$ to be the squared L_2-norm of the difference $f - \hat{f}$, i.e.,

$$\int_{M^0} (f - \hat{f})^2 = \sum_{j=-1}^{n} \sum_{i \in \mathcal{I}^{j+1} \setminus \mathcal{I}^j} (c_i^j - \hat{c}_i^j)^2 \int_{M^0} (\psi_i^j)^2 ,$$

the coefficients with the largest magnitude cause the most decrease in error since the wavelets are normalized and the $\{\hat{c}_i^j\}$ are initialized with zero. Coefficients with larger magnitude should thus be sent first. We can even extend this approach to the binary representation of $\left|c_i^j\right|$ by ranking the bits according to the decrease in error they are causing. This ranking induces a bitplane transmission in which bits are transmitted in passes with the most significant ones coded in the first pass. Assume the coefficients to be uniformly quantized in the interval $(-2T_0, 2T_0)$, i.e., each coefficient is represented in terms of a sign bit and *refinement bits* $b_k(c_i^j)$, where $c_i^j = sign(c_i^j) \sum_k b_k(c_i^j) T_k$, and $T_k = T_0/2^k$. In the first pass, the embedded coding only transmits the sign and location for *significant coefficients*, i.e., whose magnitude exceeds the threshold T_0. Then, in subsequent passes, the threshold is halved and locations and signs of coefficients that became significant with respect to T_k are transmitted. Each of these *sorting passes* is followed by a *refinement pass* in which the refinement bits for significant coefficients from previous passes are sent.

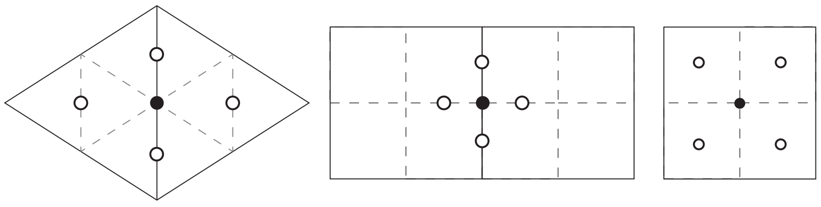


Figure 7.8: Parent-child relationships in triangular and quadrilateral hierarchies. Solid circles indicate parents; empty circles indicate children.

A main feature of the described transmission scheme is that the locations of wavelet coefficients do not have to be transmitted. If encoder and decoder agree on a common traversal of coefficients, then the information that has to be transmitted reduces to the results of magnitude comparisons (significance bits) and the refinements bits. Due to their decay properties, we can further organize coefficients into sets such that, with high probability, all coefficients are below threshold. Therefore, performing set-based significance tests, only a few bits are sufficient to determine the location of significant coefficients.

A sophisticated scheme for the partitioning of sets and the conveyance of significance information is the SPIHT encoding [Said et al., 1996] that is based on zerotrees. To adopt SPIHT to wavelet transforms for adaptive multiresolution meshes we construct a tree organization of the coefficients by employing the hierarchical relationship inherent to faces and edges of the considered meshes. For our triangle hierarchies, coefficients have a one-to-one association with edges of the coarser level. For quadrilateral hierarchies there are two types of associations: vertices inserted at face midpoints are associated with these faces, whereas vertices on edge midpoints are associated to these edges. The resulting parent-child relationship of coefficient is depicted in Figure 7.8.

Based on the hierarchy of the coefficients, we can define the following sets to be used in the SPIHT encoding:

- $\mathcal{O}(c_i^j)$: set of all children of c_i^j,
- $\mathcal{D}(c_i^j)$: set of all descendants of c_i^j,
- $\mathcal{L}(c_i^j) = \mathcal{D}(c_i^j) \setminus \mathcal{O}(c_i^j)$,
- $\{c_i^0\}$: set of all root coefficients.

The significance test for a set $\mathcal{T}$ of coefficients can be written as the function

$$S_k(\mathcal{T}) = \begin{cases} 1 & \max\limits_{c_i^j \in \mathcal{T}} \left|c_i^j\right| \geq T_k, \\ 0 & \text{otherwise.} \end{cases}$$

To simplify notation, we write $S_k(c_i^j)$ for the significance test of the set containing the single coefficient c_i^j.

Thus, if $S_k(\mathcal{T}) = 0$, all coefficients in a set are known to be insignificant. On the contrary, if $S_k(\mathcal{T}) = 1$, the encoder and decoder must split $\mathcal{T}$ and perform the test for the subsets. To keep the size of the code as small as possible, sets are split so that subsets expected to be insignificant contain many coefficients whereas subsets expected to be significant contain only a few coefficients. The set partitioning is performed until all significant coefficients are contained in a single-coefficient set and hence are identified as significant. In particular, the set partitioning rules are:

- $\{c_i^0\}$ and $\left\{\mathcal{D}(c_l^0) \;\middle|\; c_l^0 \in \{c_i^0\}, \mathcal{D}(c_l^0) \neq \varnothing\right\}$ are the initial sets,
- if $\mathcal{D}(c_i^j)$ is significant then it is split into $\mathcal{L}(c_i^j)$ and one single-coefficient set for each child $c_l^k \in \mathcal{O}(c_l^j)$,
- if $\mathcal{L}(c_i^j)$ is significant then it is split into $\mathcal{D}(c_l^k)$ for each child $c_l^k \in \mathcal{O}(c_l^j)$.

Algorithm 4 presents a pseudo-code for the implementation of the described embedded coding scheme. Lists are used to keep track of significant coefficients (L_{SC}), insignificant coefficients (L_{IC}) and insignificant sets (L_{IS}). Each set is represented by the corresponding coefficient c_i^j that is marked as *type A* if the set is $\mathcal{D}(c_i^j)$ or as *type B* if the set is $\mathcal{L}(c_i^j)$. Note that the **foreach**-statement in line 8 also enumerates sets that are added to L_{IS} during its execution. As long as the encoder and decoder initialize the lists L_{IC} and L_{IS} with the same ordering of coefficients, just knowing the outcome of the significance test and the refinement bits is sufficient to decode the magnitudes of the coefficients. To derive the corresponding decode algorithm it suffices to replace *output* by *input*. Additionally, with every received bit, the decoder has to update the reconstructed coefficients $\{\hat{c}_i^j\}$. When the decoder receives that $S_k(c_i^j) = 1$ in the k-th sorting pass, it sets $\hat{c}_i^j = \pm 1.5 \times T_k$ according to the additionally received sign. Similarly, the decoder adds or subtracts T_{k+1} to $\hat{c}_i^j$ depending on the refinement bit $b_k(c_i^j)$ received in the k-th refinement pass.

```
Algorithm: SPIHT
Data: Number of passes $n$, coefficients $\{c_i^j\}$
output $n$;
$L_{SC} \leftarrow \varnothing$, $L_{IC} \leftarrow \{c_i^0\}$, $L_{IS} \leftarrow \{c_i^0 \mid \mathcal{D}(c_i^0) \neq \varnothing\}$;
for $k \leftarrow 0, \dots, n-1$ do
    // $k$-th sorting pass
    foreach $c_i^j$ in $L_{IC}$ do
        output $S_k(c_i^j)$;
        if $S_k(c_i^j) = 1$ then
            output sign($c_i^j$) and move $c_i^j$ from $L_{IC}$ to $L_{SC}$;
8   foreach $c_i^j$ in $L_{IS}$ do
        if $c_i^j$ is of type $A$ then
            output $S_k(\mathcal{D}(c_i^j))$;
            if $S_k(\mathcal{D}(c_i^j)) = 1$ then
                foreach $c_l^q \in \mathcal{O}(c_i^j)$ do
                    output $S_k(c_l^q)$;
                    if $S_k(c_l^q) = 1$ then
                        output sign($c_l^q$) and add $c_l^q$ to $L_{SC}$;
                    else add $c_l^q$ to $L_{IC}$;
                if $\mathcal{L}(c_i^j) \neq \varnothing$ then move $c_i^j$ to end of $L_{IS}$ as type B;
                else remove $c_i^j$ from $L_{IS}$;
        else// $c_i^j$ is of type B
            output $S_k(\mathcal{L}(c_i^j))$;
            if $S_k(\mathcal{L}(c_i^j)) = 1$ then
                add each $c_l^q \in \mathcal{O}(c_i^j)$ to $L_{IS}$ as type A;
                remove $c_i^j$ from $L_{IS}$;
    // $k$-th refinement pass
    foreach $c_i^j \in L_{SC}$ do
        if $c_i^j$ did not become significant in this pass then
            output $b_k(c_i^j)$;
```

Algorithm 4: Embedded coding of wavelet coefficients based on set partitioning (cf. [Said et al., 1996]).

7.3.3 Entropy coding

The zerotree-based coding presented in Section 7.3.2 already provides efficient compression rates at low computational cost. As with any other coding scheme, the efficiency of the zerotree coder can be further improved by entropy-coding its output. In this Section we will employ context-conditioning for the coding of significance bits to exploit the correlation between nearby wavelet coefficients. Sign and refinement bits, on the other hand, show much less correlation and the gain achieved by entropy coding usually does not outweigh the additional computational cost.

There are two forms of statistical dependencies between wavelet coefficients: intraband and interband correlation. Intraband correlation describes the correlation between coefficients within the same subband whereas the interband correlation refers to the dependencies between parent and child coefficients. Following an information theoretic analysis of the statistical dependencies between wavelet coefficients, Denis et al. [2010] recently observed that the intraband correlation is systematically stronger than the interband correlation for multiresolution meshes. Moreover, to fully exploit the correlation between coefficients, composite coding designs incorporating both intraband and interband correlation must be employed.

To exploit intraband correlation, the SPIHT coder conveys significance information for children of one parent as a single symbol. Different context models are used based on the number of insignificant children $m \in \{1, 2, 3, 4\}$, where context m encodes an alphabet of 2^m symbols. Each context is thus conditioned on the fact that a certain number of children are already significant. However, as depicted in Figure 7.9, there are coefficients within the same subband (squares) that are closer than the sibling coefficients in the hierarchy

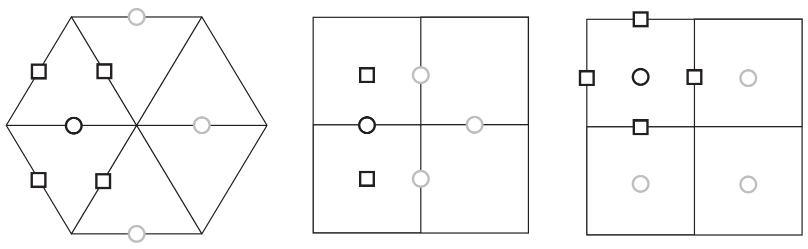


Figure 7.9: The neighbors (squares) that are used for context-based coding of significance bits. The black circle marks the current coefficient and gray circles represent sibling coefficients that are grouped in the SPIHT coder.

(light gray). Expecting a stronger correlation between direct neighbors within a subband, we construct context groups based on their significance information. We follow the ideas of our connectivity coding and use the number of significant, insignificant and unprocessed neighbor coefficients. In particular, we define four context groups representing the configurations:

- all processed neighbors are significant,
- all processed neighbors are insignificant,
- all neighbors are unprocessed, and
- there are significant and insignificant neighbors.

This choice of context groups can be used for various mesh types and is robust if the number of neighbors varies due to boundaries, adaptivity, or irregular face degrees. In fact, we can even extend the context groups for entropy coding of the sets $\{\mathcal{D}(c_i^j)\}$ and $\{\mathcal{L}(c_i^j)\}$. Thus, by coding the different set types separately, we triple the number of contexts to twelve. Neighbors of a set are again sets of the same type and are counted significant if their respective set significant test is positive, i.e., $S_k(\mathcal{T}) = 1$.

The introduced coding can be further improved by also exploiting their interband correlation. To this end, we incorporate the significance information of coefficients in the parent level. This can be achieved by additionally doubling the number of contexts for coefficients and sets $\{\mathcal{D}(c_i^j)\}$. The choice of the context then additionally depends on the significance of the coefficient's parent. For a set $\mathcal{D}(c_i^j)$, we switch the context if the coefficient c_i^j is significant, i.e., $S_k(c_i^j) = 1$. For a set $\mathcal{L}(c_i^j)$, no doubling of contexts is needed; if all parent coefficients $c_l^q \in \mathcal{O}(c_i^j)$ are insignificant $\mathcal{L}(c_i^j)$ will always be significant.

7.3.4 Geometry of AMR meshes

So far, we described a wavelet-based coding scheme for scalar functions defined on multiresolution meshes. Nonetheless, the presented scheme is not limited to this setting and in this section we will see that minor extensions are sufficient to efficiently compress the geometry of adaptive hierarchies.

To encode the geometry of a polygonal surface, we first have to extend our scheme to vector-valued functions. Various strategies for generalizing the concept of significance and signs to the vector case have been examined [Khodakovsky et al., 2000] but did not outperform the simple, yet effective strategy to treat each component individually. However, vector-valued coefficients typically exhibit a fair amount of correlation between their components and we can improve the compression by choosing a proper transform to make the components more statistically independent. Furthermore, the choice of a transform should employ knowledge of the distortion metric. The simple measure used as motivation for the scalar coding scheme in Section 7.3.2 is typically less appropriate for vector-valued functions. The human visual system, for example, is less sensitive to the chrominance than the luminance of a color. Representing colors by their chrominance and luminance therefore not only improves decorrelation but also allows for coarser quantization of chrominance components without a significant loss of the overall perceptual quality. Similarly, it has been observed that wavelet coefficients of the geometry tend to point into normal directions. We can reduce this correlation of the components by representing them in a local frame induced by the tangent plane of the surface. Moreover, common distortion measures for the geometry of surfaces are less sensitive to tangential errors than normal errors, which justifies an increased precision for the coding of the normal components. In our implementation, we use a quantization that is four times finer for the normal components since this factor has been reported as being reasonable in the literature.

Another aspect that has to be considered is the adaptivity of the multiresolution meshes. We not only know the maximal level at which detail coefficients occur, we also know where in the hierarchy there are no coefficients due to adaptivity. The zerotree coder can thus simply skip these coefficients, hence no bit will be transmitted. Additionally, we have to handle non-existent coefficients when determining the context of significance bits for neighbors – we consider these coefficients as unprocessed.

Because zero coefficients are particularly well handled by the zerotree coder, an alternative approach is to fill up the hierarchy of wavelet coefficient with zeros producing a uniformly refined tree structure, as done in the normal mesh compression presented in [Khodakovsky and Guskov, 2003]. Although the impact on the code size for the geometry is relatively small, the data necessary to reconstruct the adaptivity is more costly than our proposed method, especially if the mesh refinement is not geometry related.

In our experiments we use a common threshold T_0 for all the individual components. In particular, we choose T_0 as the greatest magnitude of a coefficients' components. Remember that the normal components have been scaled up to achieve increased precision. The number of bit planes is the same for all components and has been chosen to achieve L_2 errors that are comparable to those caused by uniform 12-bit quantization.

For the progressive transmission of multiresolution meshes it is important to interleave the connectivity and geometry data. This enables the decoder to reconstruct an intermediate mesh from any prefix of the stream. Therefore a common mesh traversal strategy has to be chosen for the geometry and hierarchy compression. Nevertheless, the traversal used by our coder can be changed to any progressive conquering of the mesh in order to facilitate a special geometry coder.

7.4 Results and discussion

We measured the performance of our hierarchy and geometry compression for a test set comprised of the 14 models shown in Figure 7.5. Furthermore, we evaluated the extension of the hierarchy coding to time-varying refinements for the three sequences depicted in Figure 7.6. The test set contains triangular and quadrangular hierarchies that come from various sources in different application areas. The uniform remeshes of the feline, horse, rabbit and venus models are taken from the PGC data set of Khodakovsky et al. The respective adaptive versions have been constructed using local coarsening based on various curvature measures. Bones and heat_transer result from the numerical solution of optimal control problems computed at the Zuse Institute Berlin. The femur model, courtesy of Oliver Sander, was refined according to criteria of a two-body contact problem in a human knee joint. All quadrilateral multiresolution meshes have been generated using QuadCover [Kälberer et al., 2007] parameterizations. Again, the adaptive versions are constructed by coarsening according to curvature criteria except for the fandisk4 and the rockerarm model that are based on heat diffusion and thin shell simulation. Since bones, heat_transfer, and femur are not generated with our own hierarchy manager we additionally have to remember how non-conforming situations are resolved to be able to losslessly reconstruct the connectivity for these models. These orientation bits have been entropy coded and the impact on the code size is reflected in the presented results.

Table 7.1 details the impact of the individual strategies for hierarchy coding presented in Section 7.2. In particular, it shows the efficiency of **stream truncation** and **1-regularity** as well as the context-based coding. On the other hand, there are only a few symbols that can be omitted due to uniform refinement since our test models feature strongly varying refinements. Though the stream truncation exploits the rather simple observation that trailing 0 symbols are not needed for the reconstruction, it still has a significant impact on

	Triangle Mesh		PMC			Wavemesh			PGC			Our		
		v	geo	con	L_2	geo	con	L_2	geo	con	L_2	geo	con	L_2
adaptive	bones	2809	20.63	2.34	47	21.60	4.44	47	–	–	–	17.81	1.84	44
	fandisk3	43048	11.15	1.53	38	11.94	2.87	37	–	–	–	5.15	0.58	27
	feline	127020	10.18	1.23	37	10.33	2.40	37	–	–	–	2.28	0.48	36
	femur	4474	17.12	1.32	38	18.53	2.75	38	–	–	–	13.89	0.71	24
	heat_trans	48652	2.22	1.21	0	3.57	2.52	0	–	–	–	0.07	0.64	4
	horse	48186	11.30	1.32	40	11.39	2.37	40	–	–	–	3.16	0.53	40
	rabbit	34255	12.94	1.47	43	13.28	2.81	43	–	–	–	5.78	0.64	32
	venus	69338	12.60	1.50	33	12.39	2.60	33	–	–	–	5.12	0.52	26
uniform	fandisk3	154498	4.47	0.03	40	4.40	0.04	28	1.99	0.01	26	1.72	0.01	24
	feline	258046	5.34	0.01	37	2.64	0.01	31	1.14	0.00	34	1.09	0.00	34
	horse	112642	5.51	0.01	40	3.45	0.01	32	1.35	0.00	40	1.30	0.00	40
	rabbit	107522	5.65	0.01	43	3.90	0.01	43	1.93	0.00	32	1.88	0.00	32
	venus	198658	5.67	0.01	35	3.63	0.01	40	2.04	0.00	30	1.96	0.00	30
	Avg. (adaptive)		12.27	1.49	35	12.88	2.85	34	–	–	–	6.67	0.74	29
	Avg. (uniform)		5.33	0.01	39	3.54	0.02	35	1.69	0.00	32	1.59	0.00	32

Table 7.4: Connectivity and geometry rates (columns *con* and *geo*) for triangle hierarchies in bits per vertex. Column v lists the number of vertices, and L_2 the root mean square errors reported by `metro` in units of 10^{-6} with respect to the bounding box diameter.

the entropy of the mesh, i.e., keeping the zeros while entropy coding expanded the codes by 17%. Together the strategies of Section 7.2.3 nearly halve the number of symbols and generate a compact representation of the hierarchy. Due to a uniform distribution of the symbols, the compact binary representation is almost incompressible which is elucidated by the values in parentheses in Table 7.1. However, employing knowledge about the mesh structure to derive context models and thus exploit mutual information inherent in the refinement improves the entropy coding by approximately 50%.

Table 7.2 lists the performance of the individual context groups in terms of compression rates. It reveals huge gaps between the performance of the different groups that can be attributed to the mutual information inherent to each group. Table 7.2 also provides a comparison to the distribution of symbols resulting from our improved traversal (cf. Section 7.2.5) illustrating the shift of symbols among the groups. As a consequence, the mutual information available for the coding of each symbols is balanced and an average compression gain of 7% is achieved for our test set.

The compression rates for both the connectivity as well as the geometry are combined in Table 7.4 for the triangular respectively Table 7.5 for the quadrangular test models. Although the connectivity is trivial in the uniform case, we

	Quad Mesh		PMC			Our		
		v	geo	con	L_2	geo	con	L_2
adaptive	blade	36358	9.31	2.14	55	3.67	0.28	36
	fandisk4	23657	5.50	0.33	58	2.73	0.07	39
	fertility	134351	8.11	2.22	55	1.59	0.32	47
	rockerarm	24460	8.52	1.46	61	3.50	0.23	44
	shoulder	77573	8.80	2.08	56	1.75	0.31	59
	torso	39086	9.23	2.10	58	2.64	0.31	54
uniform	blade	127234	4.48	0.00	55	1.02	0.00	52
	fandisk4	263682	3.61	0.00	52	0.51	0.00	57
	fertility	274426	4.15	0.00	55	0.67	0.00	52
	rockerarm	300546	5.06	0.00	61	1.20	0.00	62
	shoulder	94208	4.10	0.00	56	0.41	0.00	57
	torso	269826	4.03	0.00	58	0.34	0.00	45
	Avg. (adaptive)		8.11	1.72	57	2.65	0.25	49
	Avg. (uniform)		4.24	0.00	56	0.69	0.00	54

Table 7.5: Connectivity and geometry rates for quadrilateral hierarchies in bits per vertex.

provide geometry coding results for all models for which the finest level geometry information was also available. We provide comparisons to the single-rate Polygonal Mesh Compression (PMC) [Isenburg, 2002] which achieves state-of-the-art compression rates for both triangular and quadrilateral meshes. Additionally, for the progressive, wavelet-based compression of triangle meshes, we present compression results of Wavemesh and PGC. For Wavemesh we use the current version 2.1 which augments the irregular multiresolution scheme with the ability to recognize and exploit uniform subdivision connectivity. PGC is a well-proven, state-of-the-art geometry compression scheme which we extended to adaptive and quadrilateral hierarchies. Although PGC is limited to uniform multiresolution meshes we include it in the comparison to evaluate the efficiency of our context modeling.

The comparisons are guided by the reconstruction error introduced by uniform 12-bit quantization which was used for the PMC codec. The parameters for Wavemesh, PGC and our coder have then been chosen to achieve a comparable quality in terms of distortion. In particular, we set Wavemesh to uniform 12-bit quantization for the adaptive models and zerotree encoding with lifting radius 1 for the uniform ones. The root grids needed by PGC and our coder have been compressed using uniform 12-bit quantization, too. To compute the error we use the root mean square error as reported by *metro* [Cignoni et al., 1998] in units of 10^{-6} with respect to the bounding box diameter.

The experiments clearly demonstrate the efficiency of our compression scheme for the coding of the hierarchy as well as the geometry. Regarding the hierarchy compression, our scheme outperforms Wavemesh by an average factor of 3.8 respectively PMC by average factors of 2.0 for the triangular and 6.9 for the quadrangular meshes. Furthermore, the introduced improved context models enhance the PGC geometry compression by 7% on average for the uniform triangle hierarchies. Moreover, the extension to adaptive hierarchies surpasses PMC and Wavemesh significantly by average factors between 2.0 and 3.0.

In Table 7.3 we list results of our hierarchy coding for the time-varying sequences with and without the enhancements for utilization of temporal coherence. Overall, the extensions to the static compression scheme, can significantly improve the compression rates. This especially applies to cases with highly structured refinements as found in data sets constructed during numerical simulation. On the other hand, if the refinement is very irregular and varies a lot (cf. cloth), the dynamic coding is on par with the static version.

Despite having a non-optimized prototype implementation, the times for loading and parsing the uncompressed models from hard drive dominate the execution times in our tests. In particular, encoding and decoding of the connectivity requires only a single traversal of the hierarchy in which each node is touched once. All operations needed for processing a node like finding neighbors and children, arithmetic coding of a symbol, and updating a priority queue can be done in constant time. Note that the priority queue can be split into separate doubly linked lists (one for each priority). Although, the geometry compression involves multiple passes, coding time is still linear in the number of wavelet coefficients and bit planes.

Further extensions and future work We took an in-depth look at the connectivity compression of multiresolution meshes. Additionally, for the coding of the geometry we extended the proven PGC scheme to adaptive triangular and quadrilateral hierarchies. In the literature, many tweaks have been reported to further increase the geometry compression performance, for example, the exploitation of correlations between sign bits, design of different wavelet filters, and rate-distortion optimization. Many of these can be applied directly to our geometry coding scheme.

Extending the context modeling for the significance bits of coefficients to incorporate the state in prior time frames enables us to take advantage of temporal coherency in time-varying sequences. We expect similar gains for the geometry compression as for the connectivity part. Furthermore, the ideas presented for both the hierarchy and geometry compression can be transferred to the coding of adaptive tetrahedral and hexahedral multiresolution meshes and pose an interesting direction for future work.

Bibliography

Adams, B., Ovsjanikov, M., Wand, M., Seidel, H.-P., and Guibas, L. J. (2008). Meshless modeling of deformable shapes and their motion. In *Proc. of Symposium on Computer Animation*, pages 77–86.

Alliez, P. and Desbrun, M. (2001). Progressive compression for lossless transmission of triangle meshes. In *Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 195–202. ACM.

Amestoy, P. R., Duff, I. S., Koster, J., and L'Excellent, J.-Y. (2001). A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41.

An, S. S., Kim, T., and James, D. L. (2008). Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.*, 27(5):165:1–165:10.

Au, O. K.-C., Tai, C.-L., Liu, L., and Fu, H. (2006). Dual Laplacian editing for meshes. *IEEE TVCG*, 12:386–395.

Avilés, M., Morán, F., and García, N. (2005). Progressive lower trees of wavelet coefficients: Efficient spatial and SNR scalable coding of 3D models. *Advances in Mulitmedia Information Processing-PCM 2005*, pages 61–72.

Bank, R., Sherman, A., and Weiser, A. (1983). Some refinement algorithms and data structures for regular local mesh refinement. *Scientific Computing, Applications of Mathematics and Computing to the Physical Sciences.*

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH*, pages 43–54.

Barbič, J., da Silva, M., and Popović, J. (2009). Deformable object animation using reduced optimal control. *ACM Trans. Graph.*, 28:53:1–53:9.

Barbič, J. and James, D. L. (2005). Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.*, 24(3):982–990.

Barbič, J. and Popović, J. (2008). Real-time control of physically based simulations using gentle forces. *ACM Trans. Graph.*, 27(5):163:1–163:10.

Barbič, J., Sin, F., and Grinspun, E. (2012). Interactive editing of deformable simulations. *ACM Trans. Graph.*, 31(4):70:1–70:8.

Ben-Chen, M., Weber, O., and Gotsman, C. (2009). Variational harmonic maps for space deformation. *ACM Trans. Graph.*, 28(3).

Blumensath, T. and Davies, M. (2010). Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE J. Sel. Topics Signal Process.*, 4(2):298 –309.

Bonet, J. and Wood, R. (2008). *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Levy, B. (2010). *Polygon Mesh Processing*. AK Peters.

Botsch, M., Pauly, M., Gross, M., and Kobbelt, L. (2006). PriMo: Coupled prisms for intuitive surface modeling. In *Proceedings of Eurographics/Siggraph Symposium on Geometry Processing*, pages 11–20.

Botsch, M., Pauly, M., Wicke, M., and Gross, M. (2007). Adaptive space deformations based on rigid cells. In *Computer Graphics Forum*, volume 26(3), pages 339–347. Proceedings of Eurographics.

Botsch, M. and Sorkine, O. (2008). On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graphics*, 14(1):213–230.

Botsch, M., Wiratanaya, A., and Kobbelt, L. (2002). Efficient high quality rendering of point sampled geometry. In *Proc. of the Eurographics Workshop on Rendering*, pages 53–64.

Bridson, R., Marino, S., and Fedkiw, R. (2003). Simulation of clothing with folds and wrinkles. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36.

Bro, R. and De Jong, S. (1997). A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401.

Bro-Nielsen, M. (1997). Fast finite elements for surgery simulation. In *Proc. Medicine Meets Virtual Reality 5 (MMVR-5'97)*.

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. (2002). A multiresolution framework for dynamic deformations. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 41–47, New York, NY, USA. ACM.

Catmull, E. and Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350 – 355.

Cevher, V. (2011). On Accelerated Hard Thresholding Methods for Sparse Approximation. In *Wavelets and Sparsity XIV*, volume 8138 of *Proc. SPIE*.

Chadwick, J. N., An, S. S., and James, D. L. (2009). Harmonic shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Trans. Graph.*, 28(5):119:1–119:10.

Chao, I., Pinkall, U., Sanan, P., and Schröder, P. (2010). A simple geometric model for elastic deformations. *ACM Trans. Graph.*, 29:38:1–38:6.

Choi, M. G. and Ko, H.-S. (2005). Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Trans. Vis. Comput. Graphics*, 11(1):91–101.

Ciarlet, P. G. (2000). *Mathematical Elasticity - Volume III: Theory of Shells*, volume 29 of *Studies in Mathematics and Its Applications*. North Holland.

Ciarlet, P. G. (2002). *Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Cignoni, P., Rocchini, C., and Scopigno, R. (1998). Metro: Measuring error on simplified surfaces. *Comput. Graph. Forum*, 17:167–174.

Cirak, F., Ortiz, M., and Schröder, P. (2000). Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072.

Clarenz, U., Diewald, U., Dziuk, G., Rumpf, M., and Rusu, R. (2004). A finite element method for surface restoration with smooth boundary conditions. *Comput. Aided Geom. Des.*, 21:427–445.

Debunne, G., Desbrun, M., Cani, M.-P., and Barr, A. H. (2001). Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 31–36, New York, NY, USA. ACM.

Denis, L., Satti, S., Munteanu, A., Cornelis, J., and Schelkens, P. (2010). Scalable intraband and composite wavelet-based coding of semiregular meshes. *IEEE Transactions on Multimedia*, 12(8):773–789.

Dennis, J. and Schnabel, R. (1987). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics. SIAM.

Dey, T. K., Li, K., Luo, C., Ranjan, P., Safa, I., and Wang, Y. (2010). Persistent heat signature for pose-oblivious matching of incomplete models. *Computer Graphics Forum*, 29(5):1545–1554.

Dong, S., Bremer, P.-T., Garland, M., Pascucci, V., and Hart, J. C. (2006). Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066.

Dyn, N., Levine, D., and Graphory, J. A. (1990). A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph.*, 9(2):160–169.

Dziuk, G. (1991). An algorithm for evolutionary surfaces. *Numer. Math. 58*, pages 603–611.

Eckstein, I., Pons, J.-P., Tong, Y., Kuo, C.-C. J., and Desbrun, M. (2007). Generalized surface flows for mesh processing. In *Symposium on Geometry Processing*, pages 183–192. Eurographics Association.

Foucart, S. (2011). Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM J. Numer. Anal.*, 49(6):2543–2563.

Garg, A., Grinspun, E., Wardetzky, M., and Zorin, D. (2007). Cubic Shells. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 91–98.

Gebal, K., Bærentzen, J. A., Aanæs, H., and Larsen, R. (2009). Shape analysis using the auto diffusion function. *Computer Graphics Forum*, 28(5):1405–1413.

Georgii, J. and Westermann, R. (2008). Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*, pages 11–19.

Gibson, S. F. and Mirtich, B. (1997). A survey of deformable modeling in computer graphics. Technical Report MERL-TR-97-19, Mitsubishi Electric Research Labortories, Cambridge, Massachusetts.

Gill, P. E., Murray, W., and Wright, M. H. (1982). *Practical Optimization*. Academic Press.

Gingold, Y., Secord, A., Han, J. Y., Grinspun, E., and Zorin, D. (2004). A discrete model for inelastic deformation of thin shells. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Götschel, S., von Tycowicz, C., Polthier, K., and Weiser, M. (2013). Reducing memory requirements in scientific computing and optimal control. Technical Report 13-64, ZIB, Takustr.7, 14195 Berlin. URN: urn:nbn:de:0297-zib-42695, submitted.

Griewank, A., Juedes, D., and Utke, J. (1996). Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Softw.*, 22(2):131–167.

Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P. (2003). Discrete Shells. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 62–67.

Grinspun, E., Krysl, P., and Schröder, P. (2002). CHARMS: A simple framework for adaptive simulation. *ACM Trans. Graph.*, 21(3):281–290.

Guskov, I., Vidimče, K., Sweldens, W., and Schröder, P. (2000). Normal meshes. In *SIGGRAPH '00 Proceedings*, pages 95–102.

Hahn, F., Martin, S., Thomaszewski, B., Sumner, R., Coros, S., and Gross, M. (2012). Rig-space physics. *ACM Trans. Graph.*, 31(4):72:1–72:8.

Harmon, D. and Zorin, D. (2013). Subspace integration with local deformations. *ACM Trans. Graph.*, 32(4):107:1–107:10.

Hauser, K. K., Shen, C., and O'Brien, J. F. (2003). Interactive deformation using modal analysis with constraints. In *Graphics Interface*, pages 247–256.

Hecht, F., Lee, Y. J., Shewchuk, J. R., and O'Brien, J. F. (2012). Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans. Graph.*, 31(5):123:1–123:13.

Heeren, B., Rumpf, M., Wardetzky, M., and Wirth, B. (2012). Time-discrete geodesics in the space of shells. *Comp. Graph. Forum*, 31(5):1755–1764.

Hildebrandt, K., Schulz, C., von Tycowicz, C., and Polthier, K. (2010). Eigenmodes of surface energies for shape analysis. In *Proceedings of Geometric Modeling and Processing*, pages 296–314.

Hildebrandt, K., Schulz, C., von Tycowicz, C., and Polthier, K. (2011). Interactive surface modeling using modal analysis. *ACM Trans. Graph.*, 30:119:1–119:11.

Hildebrandt, K., Schulz, C., von Tycowicz, C., and Polthier, K. (2012a). Interactive spacetime control of deformable objects. *ACM Trans. Graph.*, 31(4):71:1–71:8.

Hildebrandt, K., Schulz, C., von Tycowicz, C., and Polthier, K. (2012b). Modal shape analysis beyond Laplacian. *Computer Aided Geometric Design*, 29(5):204–218.

Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.-Y., Teng, S.-H., Bao, H., Guo, B., and Shum, H.-Y. (2006). Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, 25(3).

Huang, J., Tong, Y., Zhou, K., Bao, H., and Desbrun, M. (2011). Interactive shape interpolation through controllable dynamic deformation. *IEEE Trans. Vis. Comput. Graphics*, 17(7):983–992.

Huang, J., Zhang, M., Ma, J., Liu, X., Kobbelt, L., and Bao, H. (2008). Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.*, 27(5):1–9.

Huang, Q., Wicke, M., Adams, B., and Guibas, L. (2009). Shape decomposition using modal analysis. *Comp. Graph. Forum*, 28(2):407–416.

Hughes, T. (2000). *The finite element method: Linear static and dynamic finite element analysis*. Dover Publications.

Idelsohn, S. R. and Cardona, A. (1985). A reduction method for nonlinear structural dynamic analysis. *Computer Methods in Applied Mechanics and Engineering*, 49(3):253 – 279.

Isenburg, M. (2002). Compressing polygon mesh connectivity with degree duality prediction. In *Graphics Interface Conference Proceedings*, pages 161–170.

Isenburg, M. and Snoeyink, J. (1999). Mesh collapse compression. In *Proceedings of SIBGRAPI'99*, pages 27–28.

Isenburg, M. and Snoeyink, J. (2006). Early-split coding of triangle mesh connectivity. In *Graphics Interface Proceedings*.

Jacobson, A., Baran, I., Kavan, L., Popović, J., and Sorkine, O. (2012). Fast automatic skinning transformations. *ACM Trans. Graph.*, 31(4):77:1–77:10.

James, D. L. and Pai, D. K. (1999). Artdefo: Accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 65–72, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Joshi, P., Meyer, M., DeRose, T., Green, B., and Sanocki, T. (2007). Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26(3).

Ju, T., Schaefer, S., and Warren, J. (2005). Mean value coordinates for closed triangular meshes. In *ACM Transaction on Graphics*, pages 561–566.

Kälberer, F., Nieser, M., and Polthier, K. (2007). QuadCover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384.

Kälberer, F., Polthier, K., Reitebuch, U., and Wardetzky, M. (2005). Freelence - coding with free valences. *Computer Graphics Forum*, 24(3):469–478.

Kälberer, F., Polthier, K., and von Tycowicz, C. (2009). Lossless compression of adaptive multiresolution meshes. In *Proceedings*.

Kharevych, L., Mullen, P., Owhadi, H., and Desbrun, M. (2009). Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph.*, 28(3):51:1–51:8.

Khodakovsky, A. and Guskov, I. (2003). Compression of normal meshes. In *Geometric Modeling for Scientific Visualization*, pages 189–206. Springer-Verlag.

Khodakovsky, A., Schröder, P., and Sweldens, W. (2000). Progressive geometry compression. In *SIGGRAPH '00 Proceedings*, pages 271–278.

Kilian, M., Mitra, N. J., and Pottmann, H. (2007). Geometric modeling in shape space. *ACM Trans. Graph.*, 26(3).

Kim, T. and Delaney, J. (2013). Subspace fluid re-simulation. *ACM Trans. Graph.*, 32(4):62:1–62:9.

Kim, T. and James, D. L. (2009). Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph.*, 28:123:1–123:9.

Kobbelt, L. (1996). Interpolatory subdivision on open quadrilateral nets with arbitrary topology. 15:409–420.

Koiter, W. (1966). On the Nonlinear Theory of Thin Elastic Shells. *Proc. Konik. Ned. Akad. Wetensch.*, 69:1–54.

Kraevoy, V. and Sheffer, A. (2006). Mean-value geometry encoding. *International Journal of Shape Modeling*, 12(1):29–46.

Krysl, P., Lall, S., and Marsden, J. E. (2001). Dimensional model reduction in non-linear finite element dynamics of solids and structures. *Int. J. Numer. Meth. Eng.*, 51(4):479–504.

Lawson, C. L. and Hanson, R. J. (1974). *Solving Least Square Problems.* Prentice Hall.

Lee, A. W. F., Sweldens, W., Schröder, P., Cowsar, L., and Dobkin, D. (1998). MAPS: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH Proceedings.*

Lee, H., Alliez, P., and Desbrun, M. (2002). Angle-analyzer: A triangle-quad mesh codec. *Computer Graphics Forum*, 21:383–392.

Lee, J. (2010). *Introduction to Topological Manifolds.* Graduate Texts in Mathematics. Springer.

Lehoucq, R. B., Sorensen, D. C., and Yang, C. (1998). *ARPACK users' guide - solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.* Software, environments, tools. SIAM.

Lévy, B. and Zhang, H. (2009). Spectral mesh processing. In *ACM SIGGRAPH ASIA Courses*, pages 1–47.

Lipman, Y., Levin, D., and Cohen-Or, D. (2008). Green coordinates. *ACM Trans. Graph.*, 27(3):1–10.

Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rössl, C., and Seidel, H.-P. (2004). Differential coordinates for interactive mesh editing. In *Shape Modeling International*, pages 181–190.

Loop, C. (1987). Smooth subdivision surfaces based on triangles, Thesis. Utah University, USA.

Lounsbery, M., DeRose, T. D., and Warren, J. (1997). Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.*, 16:34–73.

Marsden, J. and Hughes, T. (1994). *Mathematical foundations of elasticity.* Dover Civil and Mechanical Engineering Series. Dover Publications Inc.

Meyer, M. and Anderson, J. (2007). Key point subspace acceleration and soft caching. *ACM Trans. Graph.*, 26(3).

Morales, J. L. and Nocedal, J. (2011). Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.*, 38(1):7:1–7:4.

Müller, M., Dorsey, J., McMillan, L., Jagnow, R., and Cutler, B. (2002). Stable real-time deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 49–54, New York, NY, USA. ACM.

Müller, M. and Gross, M. (2004). Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.

Müller, M., McMillan, L., Dorsey, J., and Jagnow, R. (2001). Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 113–124, New York, NY, USA. Springer-Verlag New York, Inc.

Müller, M., Teschner, M., and Gross, M. (2004). Physically-based simulation of objects represented by surface meshes. In *Proceedings of the Computer Graphics International*, CGI '04, pages 26–33, Washington, DC, USA. IEEE Computer Society.

Munkres, J. (1996). *Elements of Algebraic Topology.* Advanced Book Program. Westview Press.

Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836.

Nealen, A., Sorkine, O., Alexa, M., and Cohen-Or, D. (2005). A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147.

Ovsjanikov, M., Mérigot, Q., Mémoli, F., and Guibas, L. (2010). One point isometric matching with the heat kernel. *Computer Graphics Forum*, 29(5):1555–1564.

Ovsjanikov, M., Sun, J., and Guibas, L. (2008). Global intrinsic symmetries of shapes. *Computer Graphics Forum*, 27(5):1341–1348.

Payan, F. and Antonini, M. (2005). An efficient bit allocation for compressing normal meshes with an error-driven quantization. *CAGD*, 22(5):466–486.

Pentland, A. and Williams, J. (1989). Good vibrations: Modal dynamics for graphics and animation. *SIGGRAPH Comput. Graph.*, 23(3):207–214.

Pinkall, U. and Polthier, K. (1993). Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36.

Reddy, J. (2007). *Theory and Analysis of Elastic Plates and Shells*. Series in Systems and Control Series. Taylor & Francis Us.

Reuter, M., Wolter, F.-E., and Peinecke, N. (2005). Laplace-spectra as fingerprints for shape matching. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*, pages 101–106.

Reuter, M., Wolter, F.-E., and Peinecke, N. (2006). Laplace-Beltrami spectra as “Shape-DNA” of surfaces and solids. *Computer-Aided Design*, 38(4):342–366.

Rossignac, J. (1999). Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, pages 47–61.

Rustamov, R. M. (2007). Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 225–233.

Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press.

Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23:514–521.

Said, A. (2004). Introduction to arithmetic coding-theory and practice. *Hewlett Packard Laboratories Report*.

Said, A., Pearlman, W. A., and Member, S. (1996). A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250.

Schnabel, R. and Klein, R. (2006). Octree-based point-cloud compression. In Botsch, M. and Chen, B., editors, *Symposium on Point-Based Graphics 2006*. Eurographics.

Schneiders, R. (1996). Refining quadrilateral and hexahedral element meshes. In *5th International Conference on Grid Generation in Computational Field Simulations*. CRC Press.

Schröder, P. and Sweldens, W. (1995). Spherical wavelets: Efficiently representing functions on the sphere. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 161–172. ACM.

Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160.

Settgast, V., Müller, K., Fünfzig, C., and Fellner, D. (2004). Adaptive tesselation of subdivision surfaces. *Computers & Graphics*, 28(1):73–78.

Shabana, A. (1997). *Vibration of Discrete and Continuous Systems*. Number 2 in Mechanical Engineering Series. Springer.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27.

Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficients. In *IEEE Transactions of Signal Processing*, volume 41, pages 3445–3462.

Shi, X., Zhou, K., Tong, Y., Desbrun, M., Bao, H., and Guo, B. (2007). Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph.*, 26(3):81.

Sifakis, E. and Barbic, J. (2012). FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pages 20:1–20:50, New York, NY, USA. ACM.

Sorkine, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 109–116.

Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Symposium on Geometry Processing*, pages 175–184.

Sumner, R. W., Schmid, J., and Pauly, M. (2007). Embedded deformation for shape manipulation. In *ACM Trans. Graph.*, volume 26(3).

Sun, J., Ovsjanikov, M., and Guibas, L. J. (2009). A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28(5):1383–1392.

Szymczak, A. (2002). Optimized edgebreaker encoding for large and regular triangle meshes. In *DCC '02 Proceedings*, page 472, Washington, DC, USA. IEEE Computer Society.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. In *Proc. of ACM SIGGRAPH*, pages 205–214.

Tiso, P. (2011). Optimal second order reduction basis selection for nonlinear transient analysis. In Proulx, T., editor, *Modal Analysis Topics*, volume 6 of *Conference Proceedings of the Society for Experimental Mechanics Series*, pages 27–39. Springer New York.

Toponogov, V. (2006). *Differential Geometry of Curves and Surfaces: A Concise Guide*. Birkhäuser Boston.

Touma, C. and Gotsman, C. (1998). Triangle mesh compression. In *Graphics Interface Conference Proceedings*.

Treuille, A., Lewis, A., and Popović, Z. (2006). Model reduction for real-time fluids. *ACM Trans. Graph.*, 25:826–834.

Tutte, W. (1962). A census of planar triangulations. *Canadian Journal of Mathematics*, 14:21–38.

Valette, S. and Prost, R. (2004). Wavelet-based progressive compression scheme for triangle meshes: Wavemesh. *IEEE Transactions on Visualization and Computer Graphics*, 10(2).

Vallet, B. and Lévy, B. (2008). Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*.

von Tycowicz, C., Kälberer, F., and Polthier, K. (2011). Context-based coding of adaptive multiresolution meshes. *Computer Graphics Forum*, 30(8):2231–2245.

von Tycowicz, C., Schulz, C., Seidel, H.-P., and Hildebrandt, K. (2013). An efficient construction of reduced deformable objects. *ACM Trans. Graph.*, 32(6):213:1–213:10.

Wicke, M., Stanton, M., and Treuille, A. (2009). Modular bases for fluid dynamics. *ACM Trans. Graph.*, 28:39:1–39:8.

Witten, I. H., Neal, R. M., and Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

Wojtan, C. and Turk, G. (2008). Fast viscoelastic behavior with thin features. *ACM Trans. Graph.*, 27(3):47:1–47:8.

Zhang, H., van Kaick, O., and Dyer, R. (2010). Spectral mesh processing. *Computer Graphics Forum*, 29(6):1865–1894.

Zhao, Y. and Barbič, J. (2013). Interactive authoring of simulation-ready plants. *ACM Trans. Graph.*, 32(4):84:1–84:12.

Zorin, D., Schröder, P., DeRose, T. D., Kobbelt, L., Adi, L., and Sweldens, W. (2000). Subdivision for modeling and animation. In *ACM SIGGRAPH Courses*.

Zusammenfassung

Physikalisch basierte Methoden zur Modellierung statischer und dynamischer Eigenschaften flexibler Objekte sind weit verbreitet in den Bereichen der Computeranimation, geometrischen Modellierung und numerischen Simulation. Ein wesentlicher Vorteil dieser Methoden gegenüber rein geometrischen Verfahren liegt in ihrer Fähigkeit, mit nur wenigen Vorgaben realistische Deformationen zu produzieren. Da die zugehörigen Systeme jedoch hochdimensional und zudem nichtlinear sind, lassen sich interaktive Raten für allgemeine, unreduzierte Deformationen nicht erreichen. In dieser Arbeit beschäftigen wir uns mit der Entwicklung effizienter Algorithmen für die Konstruktion von simplifizierten, niedrigdimensionalen Modellen, die das Originalsystems approximieren und sich für die Verwendung in interaktiven Anwendungen eignen. Insbesondere präsentieren wir Techniken zur Konstruktion reduzierter Konfigurationsräume deformierbarer Objekte sowie der Approximation der reduzierten inneren Kräfte. Die vorgestellten Techniken sind automatisch und eignen sich zur Darstellung großer Verformungen in denen nichtlineare Terme des dynamischen Systems relevant sind. Um die Effektivität unserer Reduktionsstrategien zu demonstrieren, entwickeln wir Systeme für die Echtzeitsimulation sowie für die interaktive geometrische Modellierung elastischer Körper und Schalen und vergleichen diese mit alternativen Lösungen.

Die in der Geometrieverarbeitung verwendeten spektralen Methoden zur Analyse gekrümmter Flächen basieren fast ausnahmslos auf dem Laplace–Beltrami Operator. Sie profitieren von Eigenschaften wie der Invarianz unter Isometrien oder der Robustheit gegenüber verrauschten Daten. Wir betrachten in dieser Arbeit alternative Differentialoperatoren deren Spektren und Eigenvektoren das Schwingungsverhalten digitaler Formen charakterisieren. Die diskreten Operatoren sind sensitiv gegenüber extrinsischen Features (wie scharfe Kanten) und ermöglichen somit zusätzliche Einblicke in die Geometrie. Basierend auf diesem Konzept entwickeln wir die *Vibration Signature* – eine Multiskalensignatur, die erlaubt die Ähnlichkeit von Punkten einer Fläche zu bestimmen.

Diese Arbeit beschäftigt sich zudem mit der Kompression von digitalen Formen. Zu diesem Zweck entwickeln wir Strategien zur verlustfreien Komprimierung adaptiv verfeinerter polygonaler Flächen. Unter Ausnutzung struktureller Regularitäten der hierarchischen Netze erreichen wir hierbei Kompressionsraten, die die führender Verfahren um ein Vielfaches übertreffen. Zusätzlich erweitern wir die *Zerotree*–Kodierung um die Verwendung statistischer Abhängigkeiten innerhalb und zwischen Frequenzbändern der Waveletkoeffizienten.

Verfassererklärung

Gemäß § 7 (4) der Promotionsordnung versichere ich hiermit, diese Arbeit selbständig verfasst zu haben. Ich habe alle bei der Erstellung dieser Arbeit benutzten Hilfsmittel und Hilfen angegeben.