JANICK MARTINEZ ESTURO

SHAPES IN VECTOR FIELDS

SHAPES IN VECTOR FIELDS

METHODS FOR CONTINUOUS DEFORMATIONS AND SURFACE-BASED FLOW VISUALIZATIONS

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur
(Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke Universität Magdeburg.

vorgelegt von:	Dipl.-Inform. Janick Martinez Esturo
geboren am:	29. Juli 1983
in:	Bünde (Krs. Herford)

Gutachter:	Prof. Dr. habil. H. Theisel
	Prof. Dr. M. Botsch
	Prof. Dr. M. Wardetzky

eingereicht am:	21. Mai 2013
verteidigt am:	25. Oktober 2013

LOCATION:
Otto-von-Guericke University
Faculty of Computer Science
Institute of Simulation and Graphics
Magdeburg

Janick Martinez Esturo: *Shapes in Vector Fields* – Methods for Continuous Deformations and Surface-based Flow Visualizations

ACKNOWLEDGMENTS

First of all, I sincerely thank my advisor Holger Theisel for introducing me to the exciting field of computer graphics with his lectures in Bielefeld, and for supporting, guiding, and encouraging me throughout the last years in Magdeburg. Without his inspiration, this project would not have been started. I would also like to express my gratitude to my co-advisor Christian Rössl for his dedication and endless but very fruitful discussions. This work has greatly benefited from his helpful advices and comments. Many thanks also go to my reviewers Mario Botsch and Max Wardetzky for their time and encouragements. It has been a great experience to share their knowledge in interesting discussions. I am very grateful to a lot of people in Magdeburg for providing such an enjoyable and stimulating environment. Special thanks go to Maik Schulze, Dirk J. Lehmann, Alexander Kuhn, Mathias Otto, and Benjamin Wegener. My work has to a large extend been funded by the *Studienstiftung des deutschen Volkes*. I also wish to thank my parents who always encouraged me to pursue my studies and live my passion for research, and Felipe for being such a great brother. Most of all, I wish to thank Sophie for all her support, patience, and love.

ABSTRACT

Geometric shapes are the basic building blocks of any graphics related application. The effective manipulation of shapes is therefore of central interest for many relevant problems. In particular, there is a growing demand for high-quality nonlinear deformations for shape modeling and animation. The application of vector fields that guide a continuous deformation is a practical approach for their computation. It turns out that typically challenging nonlinear problems can be solved in an elegant way using such vector field-based methodologies. This thesis presents novel approaches and prospects for vector field-based *manipulation* of geometric shapes (Part I). Thereafter, also the *definition* of geometric shapes by means of vector fields is examined (Part II).

Depending on the specific shape representation and the concrete modeling problem, different types of vector fields are required: a family of generalized vector field energies is introduced that enables near-isometric, near-conformal, as well as near-authalic continuous deformations of planar and volumetric shapes. It is demonstrated how near-isometric surface and volume-preserving isosurface deformations are computed by a similar framework. Furthermore, an integration-based pose correction method is presented. Based on a generic energy description that incorporates energy smoothness, a conceptual simple but effective generalized energy regularization is proposed, which is not only beneficial for continuous deformations but additionally enhances a variety of related geometry processing methods.

In the second part of the thesis vector fields are not considered to represent deformations anymore. Instead, they are interpreted as flow fields that define characteristic shapes such as stream surfaces: a deformation-based approach for interactive flow exploration and the extraction of flow-tangential and flow-orthogonal surfaces is proposed. It is shown how an unified computational framework yields parametrizations that are particularly useful for surface-based flow illustrations. Finally, an automatic method for the selection of relevant stream surfaces in complex flow data sets is presented that is based on a new surface-based intrinsic quality measure.

The usefulness of the newly developed methods is shown by applying them to a number of different geometry processing and visualization problems.

ZUSAMMENFASSUNG

Geometrische Formen sind die grundlegenden Bausteine jedweder graphischen Anwendung. Die effektive Manipulation dieser Formen ist daher von besonderem Interesse für viele relevante Probleme. Insbesondere herrscht sowohl für die Modellierung als auch für die Animation von Formen ein stetig steigender Bedarf an nichtlinearen Deformationen von hoher Qualität. Die Nutzung von Vektorfeldern, die dabei eine kontinuierliche Deformation leiten, ist ein geeigneter Ansatz für deren Berechnung. Hierbei stellt sich heraus, dass typischerweise anspruchsvolle nichtlineare Probleme in einer eleganten Art und Weise unter Zuhilfenahme solcher Vektorfeld-basierten Methodiken gelöst werden können. Diese Dissertation präsentiert dazu neue Ansätze und Perspektiven für Vektorfeld-basierte *Manipulationen* von geometrischen Formen (Teil I). Darüber hinaus wird die *Definition* von geometrischen Formen durch Vectorfelder näher untersucht (Teil II).

In Abhängigkeit der spezifischen Formrepräsentation und des konkreten Modellierungsproblems werden verschiedene Typen von Vektorfeldern benötigt: Hierfür wird eine Familie generalisierter Vektorfeldenergien eingeführt, die Approximationen von isometrischen, konformen, sowie volumenerhaltenden kontinuierlichen Deformationen von planaren und volumetrischen Formen ermöglicht. Es wird gezeigt, wie Approximationen von isometrischen Oberflächendeformationen sowie volumenerhaltenden Isoflächendeformationen in einem ähnlichen System berechnet werden. Darüber hinaus wir eine integrations-basierte Methode zur Korrektur von Posen präsentiert. Auf der Grundlage einer generischen Energiebeschreibung, welche Energieglattheit mit berücksichtigt, wird eine konzeptionell einfache aber effektive generalisierte Energieregularisierung vorgeschlagen, die sich nicht nur als nützlich für kontinuierliche Deformationen erweist, sondern zusätzlich eine Vielzahl an verwandten Geometrieverarbeitungsmethoden verbessert.

In dem zweiten Teil der Dissertation werden Vektorfelder nicht mehr zur Repräsentation von Deformationen genutzt. Stattdessen werden sie als Strömungsfelder interpretiert, die charakteristische geometrische Formen, wie beispielsweise Stromflächen, definieren: Dazu wird ein deformations-basierter Ansatz zur interaktiven Exploration von Strömungen und zur Extraktion von tangentialen und orthogonalen Oberflächen vorgeschlagen. Es wird gezeigt, wie durch einen einheitlichen numerischen Ansatz Parameterisierungen berechnet werden, die besonders nützlich für Oberflächen-basierte Strömungsillustrationen sind. Abschließend wird auf Grundlage eines neuen Oberflächen-basierten intrinsischen

Qualitätsmaßes eine Methode zur automatischen Selektion von relevante Stromflächen in komplexen Strömungsdatensätzen präsentiert.

Der Nutzen der neu entwickelten Methoden wird demonstriert, indem sie auf eine Reihe von unterschiedlichen Geometrieverarbeitungs- und Visualisierungsproblemen angewendet werden.

PUBLICATIONS

Parts of this thesis appeared previously in the following peer-reviewed publications and submissions:

[MRF*11] Martinez Esturo J., Rössl C., Fröhlich S., Botsch M., Theisel H.: Pose correction by space-time integration. In *Proc. VMV* (2011), EG, pp. 33–40. (Cited on page 98.)

[MRT10] Martinez Esturo J., Rössl C., Theisel H.: Continuous deformations of implicit surfaces. In *Proc. VMV* (2010), EG, pp. 219–226. (Cited on page 81.)

[MRT12] Martinez Esturo J., Rössl C., Theisel H.: Continuous deformations by isometry preserving shape integration. *Springer LNCS (Proc. Curves and Surfaces 2010) 6920*, 1 (2012), 456–472. (Cited on page 64.)

[MRT13a] Martinez Esturo J., Rössl C., Theisel H.: Generalized metric energies for continuous shape deformation. *Springer LNCS (Proc. Curves and Surfaces 2012) 8177*, 1 (2013), 135–157. (Cited on page 45.)

[MRT13b] Martinez Esturo J., Rössl C., Theisel H.: Smoothed quadratic energies on meshes. *ACM Trans. Graph.* (2013), (to appear). (Cited on page 121.)

[MSRT13a] Martinez Esturo J., Schulze M., Rössl C., Theisel H.: Global selection of stream surfaces. *Comput. Graph. Forum (Proc. Eurographics) 32*, 2 (2013), 113–122. (Cited on pages 169 and 178.)

[MSRT13b] Martinez Esturo J., Schulze M., Rössl C., Theisel H.: Poisson-based tools for flow visualization. In *Proc. PacificVis* (2013), IEEE, pp. 241–248. (Cited on pages 148, 153, 156, and 160.)

CONTENTS

1

INTRODUCTION

For decades media have been dominated by text only. Then multi-media was introduced in form of images, sounds, and videos. Lately, geometric models of object *shapes* have emerged as an important new form of digital media. Digital representations of geometric shapes are ubiquitous. They are becoming even more important in various economical application areas such as industrial design, entertainment, and e-commerce, where sophisticated digital methods based on geometric data help to improve products and reduce their costs. In the same way, whole scientific fields like visualization, computational medicine, and mechanical engineering are to a great extend concerned with the study of different representations of real or simulated geometric shapes for the discovery of novel scientific results. The importance of geometric shapes comes with the need for appropriate data structures and algorithms that are required for their effective processing. In computer science, the field of *digital geometry processing* examines methods for geometric shapes that are both efficient and scalable. Computing with geometric shapes lies at the core of geometric modeling and processing. A variety of effective methods is known for, e.g., the acquisition, analysis, and storage of shapes [BKP*10].

This thesis is concerned with the *manipulation* of geometric shapes, i.e., the purposeful modification of shape geometry. Modifications can either be performed interactively by a user for persistent shape modeling, or be performed automatically for shape optimization. Shape modeling is a challenging topic of digital geometry processing because the oftentimes involved and computationally expensive deformation methods need to be performed at interactive rates and be hidden behind intuitive modeling metaphors. Shape optimization, on the other hand, is generally an offline process but has to guarantee similar properties in terms of robustness and accuracy of the solution.

The manipulation of shapes is tightly coupled to their representation and the used deformation paradigm. Typically, a shape is viewed as a set of points and represented according to the available data and the intended application. Shapes are generally represented either explicitly or implicitly, and a representation is

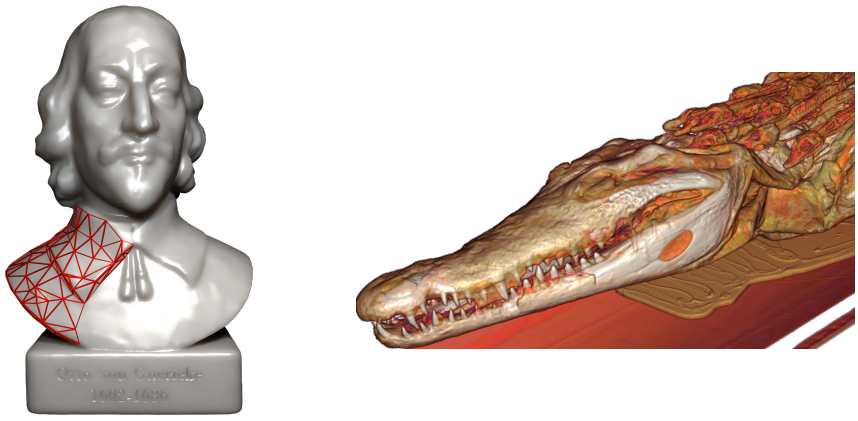


Figure 1: Shape Representations. Laser range scanning of the Otto-von-Guericke bust results in the shape represented by an *explicit* triangular surface mesh (left). The crocodile mummy is a volume data set obtained by CT scanning and defines the shape of (volume rendered) *implicit* level set surfaces (right).

chosen in a problem-specific way. Two examples are shown in Figure 1. Independent of the shape representation is the deformation paradigm. The classic paradigm considers deformations that are maps of a shape (or its surrounding space) to a deformed version, which are computed as a single deformation step. Linear and nonlinear single-step deformations are well-researched.

In this work, we study the class of shape manipulations that are *continuous*, i.e., represented as a continuously parameterized family of shape deformations. Continuous deformations have received less attention, although a number of traditionally hard nonlinear problems like volume-preserving deformations have natural solutions in this framework. An elegant way to define continuous deformations is the usage of *vector fields* that guide the evolving shapes. We consider vector field-based deformations of explicitly and implicitly defined shapes and apply them to both shape modeling and shape optimization. Continuous deformations are the main focus of the first part of this thesis. We propose new vector field energies and corresponding deformation types and present novel applications for continuous deformations.

For vector field-based deformations the guidance velocity fields are not given in the first place but are generally computed in a problem-specific way, e.g., to induce a low amount of shape distortion. However, the abstract vector field concept can also represent different kinds of fields: vector fields are also particularly well suited to represent the velocity field of complex flows. The analysis of properties of these vector fields is of special interest for various scientific fields. In contrast to the first part of this thesis, flow vector fields are either measured or simulated. In particular, they are given a priori and are not modified for the analysis. The field of *flow visualization* focuses on the computation of more abstract and characteristic representations of flow fields to facilitate the analysis of complex flow phenomena. Among the different types of flow visualization

techniques the class of geometry-based methods relies on geometric shapes to convey flow field properties. Generally, these shapes are *defined* by the flow vector field. Flow defined shapes will be considered in the second part of this thesis where we apply geometry processing techniques to improve geometry-based flow visualization.

1.1 THESIS STRUCTURE

We continue to give a brief overview of the structure of the following chapters and summarize the main contributions of this work. This thesis is separated into two parts. We consider vector field-based shape manipulations in the first part that consist of the following chapters:

- Chapter 2 introduces continuous deformations and reviews related work on shape deformations with a focus on continuous methods.
- Chapter 3 studies vector field-based and GPU-accelerated deformations of planar and volumetric shapes. Vector fields are obtained by variational optimization of a new family of generalized vector field energies that supports near-isometric, near-conformal, and near-authalic deformations as well as nonhomogeneous and anisotropic behavior.
- Chapter 4 introduces a method for near-isometric continuous surface deformations. These GPU-accelerated deformations are of higher quality compared to related nonlinear methods.
- Chapter 5 considers continuous deformations of implicitly defined isosurfaces. Vector field-based deformations are computed by a new GPU-based backward integration scheme and are guaranteed to preserve the volume and topology of every isosurface.
- Chapter 6 applies continuous deformations to pose databases for the automatic correction of geometric artifacts like local self-intersections. Path line integration is performed in vector fields that are reconstructed using a new radial basis function center selection and GPU-accelerated factorization update scheme.
- Chapter 7 introduces a conceptually simple but very effective generic energy regularization scheme that is not only applicable to continuous deformations but to a variety of geometry processing methods. Energy regularization is easy to implement and significantly improves results of a wide range of variational approaches.

In the second part of this thesis, we present interactive and automatic methods for geometry-based flow visualization that focus on surface-based techniques:

- Chapter 8 introduces the flow visualization setting and reviews related work on surface-based visualization techniques.
- Chapter 9 applies Poisson-based deformations for flux optimization to extract flow-tangential and flow-orthogonal surfaces, which are well-suited for interactive flow exploration. Flow-aligned parametrizations are computed by the same framework and facilitate seed curve computation and enable illustrative flow visualizations.
- Chapter 10 introduces a completely automatic selection method of stream surfaces. Relevant stream surfaces are identified by a new intrinsic quality measure. Representative stream surfaces are computed by a global combinatorial optimization.

We conclude this thesis in Chapter 11. The appendix of this work contains details on discretizations of differential operators of linear functions on simplices and metric deformation errors, which are used throughout this thesis.

1.2 NOTATION

In large parts of this work, we study planar or two-manifold domains (surfaces) $\mathcal{D}_2$ embedded in Euclidean spaces $\mathbb{E}^2$ or $\mathbb{E}^3$, respectively, as well as volumetric domains $\mathcal{D}_3 \subset \mathbb{E}^3$. For convenience, we identify coordinate spaces $\mathbb{E}^p$ by the vector spaces $\mathbb{R}^p$, and use the terms domain and shape interchangeably if the context is clear. We use the following basic notation for simplical discretizations of these domains, which will be extended, if required, in the corresponding chapters: surfaces are discretized by triangular meshes $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, i.e., as sets of vertices $i \in \mathcal{V}$, edges $\mathcal{E} \subseteq \mathcal{V}^2$, and triangles $\mathcal{T} \subseteq \mathcal{V}^3$. We will distinguish between internal edges $\mathcal{E}_i$ and boundary edges $\mathcal{E}_b$. Piecewise linear d-dimensional functions on $\mathcal{M}$ are represented by coefficients at the vertices, e.g., the vertex coordinate function $\mathbf{x}_i \in \mathbb{R}^d$. The collection of all "stacked" coefficients is denoted by a vector without subscript, e.g., $\mathbf{x} \in \mathbb{R}^{d\,|\mathcal{V}|}$. Discretization elements, e.g., $i \in \mathcal{V}$, are identified with indices if the context is clear. Piecewise linear functions over volumetric domains are discretized similarly by tetrahedral meshes. Higher-dimensional meshes will be considered in form of (pure) simplical complexes. We will make no formal distinction between the terms element, simplex, or cell and treat them synonymously if the context is clear.

Some non-instructive derivations of expression equalities were omitted in this work. Instead, they are provided in the accompanying additional material.

We briefly summarize on the notation used throughout this work in the following table:

Notation	Description
e, E; i	Scalar value / function or index / set element, italic letters
$\mathbf{v}$	Column vector, bold lower case letters
$\mathbf{A}$; $\mathbf{F}$	Matrix or tensor, bold upper case letters
$\mathcal{S}$	Set, calligraphic upper case letters
$\lvert e \rvert$; $\lvert \mathcal{S} \rvert$	Absolute value of a scalar or the cardinality of a set
$\lfloor a \rfloor$; $\lceil a \rceil$; $\lfloor a \rceil$	Previous, next, and nearest integer of (fractional) scalar
$\mathbf{v}^{\mathrm{T}}$, $\mathbf{A}^{\mathrm{T}}$	Transpose of a vector or a matrix
$\mathrm{vec}(\mathbf{A})$	Column-wise vectorization of a matrix
$\det(\mathbf{A})$	Determinant of a matrix
$\mathrm{Tr}(\mathbf{A})$	Trace of a matrix
$\mathrm{vol}(\Omega)$	Generalized volume of a (multi-dimensional) point set Ω
$\dim(\Omega)$	Intrinsic dimensionality of a (multi-dimensional) point set Ω
$\mathrm{diag}(\lambda_1, \ldots, \lambda_d)$	Diagonal matrix constructed from coefficients
$\mathrm{diag}^{-1}(\mathbf{A})$	Diagonal of a matrix
$\binom{1}{2}$; $\binom{\mathbf{a}}{\mathbf{b}}$	Coefficient or block (column) vector (round brackets)
$\left(\begin{smallmatrix} 1 & 3 \\ 2 & 4 \end{smallmatrix}\right)$	Coefficient matrix (round brackets)
$\left[\begin{smallmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{B} & \mathbf{D} \end{smallmatrix}\right]$	Block matrix (square brackets)
$\lVert \mathbf{v} \rVert^2 = \mathbf{v}^{\mathrm{T}} \mathbf{v}$	Squared Euclidean distance (squared two-norm) of a vector
$\lVert \mathbf{v} \rVert^2_{\mathbf{N}} = \mathbf{v}^{\mathrm{T}} \mathbf{N} \mathbf{v}$	Squared vector norm parameterized by symmetric positive-definite matrices $\mathbf{N}$
$\lVert \mathbf{A} \rVert^2_F = \mathrm{Tr}\big(\mathbf{A}^{\mathrm{T}} \mathbf{A}\big)$	Squared Frobenius norm of a matrix
$\lVert \mathbf{A} \rVert^2_{\mathbf{N}} = \mathrm{Tr}\big(\mathbf{A}^{\mathrm{T}} \mathbf{N} \mathbf{A}\big)$	Squared matrix norm parameterized by symmetric positive-definite matrices $\mathbf{N}$
$\mathbf{A} \otimes \mathbf{B}$	Kronecker product of two matrices
$\mathbf{I}$, $\mathbf{I}_d$	Square identity matrix / linear identity operator of general or specific dimensionality
$\mathbf{0}$; $\mathbf{1}$	Zero vector or matrix, and vector of ones
δ^{ij}	Kronecker's delta function
$\mathbf{\Lambda}^{i,j}_{r,s}$	Single-entry $r \times s$ matrix that is one at (i, j) and zero elsewhere
$\dot{\mathbf{c}} = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{c}(t)$	Tangent vector of a curve
$u_x = \frac{\partial}{\partial x} u(x, y)$ $u_{xy} = \frac{\partial^2}{\partial x \, \partial y} u(x, y)$	Partial derivatives of multivariate function
$\nabla e(\mathbf{x})$, $\nabla_{\mathbf{x}} e(\mathbf{x}, \mathbf{y})$	Gradient vector of a scalar-valued function w.r.t. all or a subset of variables
$\nabla \mathbf{v}(\mathbf{x})$, $\nabla_{\mathbf{x}} \mathbf{v}(\mathbf{x}, \mathbf{y})$	Gradient tensor (transpose of *Jacobian* matrix / tensor) of a vector-valued function w.r.t. all or a subset of variables

Part I

SHAPES MANIPULATED BY VECTOR FIELDS

2

OVERVIEW OF CONTINUOUS DEFORMATIONS

Persistent inelastic shape manipulation is a classic problem in computer graphics and design. Even though numerous approaches haven been developed in the previous decades, it is still an important and active area of research. Applications for planar shape deformations include, e.g., image warping and cartoon animation. Deformation of $3d$ shapes is used in diverse domains such as in engineering for shape modeling or for the creation of animations in the media industry, but also, e.g., for data registration in medical applications.

A recent trend is the development of *isometric* [SVWG12] and less constrained *near-isometric* deformation methods [KMP07, SBBG11a]. These deformations should preserve angles and area as much as possible. Intuitively, isometry is a good geometric measure for the quality of a deformation: while the shape should accurately satisfy the constraints defining the deformation, it should not unnecessarily stretch or bend. Hence, near-isometric deformations yield intuitive and high-quality results. Also, for a number of related applications the importance of isometric maps is well known [MS04, BBK05b, KFC*08, HSvTP10, SVWG12].

However, this is not free of cost! Roughly speaking, high quality near-isometric deformations come at the price of solving computationally expensive *nonlinear* problems. This is a major issue especially for interactive applications, which are typical in computer graphics and are mandatory for interactive modeling. There is a competition with more efficient linear methods (see, e.g., [BS08]), which are based on simpler, often approximated differential quantities. It is well known that linear methods fail to handle certain deformations: they either cope well with translations or with rotations – but not with both simultaneously. Also, there is no guarantee that the deformation does not induce local folds or self-intersections. We arrive at the conclusion that both, linear and nonlinear methods, have their own right to co-exist in shape deformation frameworks: the user has the choice between fast linear methods at the cost of sacrificing quality, and high-quality nonlinear methods that are significantly harder to compute in terms of computation costs and robustness.

The user has to pay a certain price – higher computation times or smaller data sets – and therefore expects benefits from nonlinear methods. These include not

only geometric properties of the deformation but also other important criteria related to usability. We identify a number of requirements that the computation of near-isometric shape deformations should fulfill and which make their computation a challenging problem:

- The isometric deformation problem is nonlinear. Nevertheless, computation must be *effective* and *robust* to guarantee a globally optimal solution. In addition, computation should be *efficient* enough to enable real-time response to user input.
- Deformations should be *plausible* for aesthetic reasons and for acceptance by the user. This requires suitable measures for the deviation of isometry.
- Deformations must *interpolate* constraints, which can be defined for *any* point of the shape. Approximate satisfaction of "soft constraints" can be tolerated only if arbitrarily small tolerances are possible in principle.
- Deformations must be *smooth* in a sense that the energy or metric error is distributed smoothly over the shape. In particular, the error must not concentrate near positional constraints.
- The final deformations must be independent of the particular partition of the shape or the domain. This implies *resolution and tessellation insensitivity*.
- Ideally, the user can – globally and locally – attenuate isometry such that continuous blends from angle preservation to area preservation are possible. Locally anisotropic behavior is an additional desirable design parameter for the user.
- Deformations must not have singularities, i.e., the gradient of the map from the original to the deformed shape must not vanish. In a discrete setting, this means that elements, triangles or tetrahedra, must not change orientation.
- Ideally, the formulation of the solution should be same for the $2d$ and the $3d$ case. This alleviates implementation.

A variety of nonlinear methods optimize for maximum local rigidity criteria that are related to near-isometric deformations, but only partially fulfill these requirements. For example, such nonlinear near-rigid energies are minimized iteratively with alternating linearizations [LZX*08] or by directly using generic local nonlinear solvers [CPSS10]. Related are approaches that iteratively optimize energies of loosely coupled near-rigid prisms [BPGK06]. These methods all optimize nonlinear energies in a direct *single-step* deformation. According to Lipman [Lip12, Section 6.1], deformations like [LZX*08] might easily get stuck in local minima. Other nonlinear methods are likely to have this property, too, as global nonlinear optimization algorithms are seldom applied to deformation problems. Hence, in general convergence to globally optimal solution cannot be guaranteed by these methods (although results are generally of high quality).

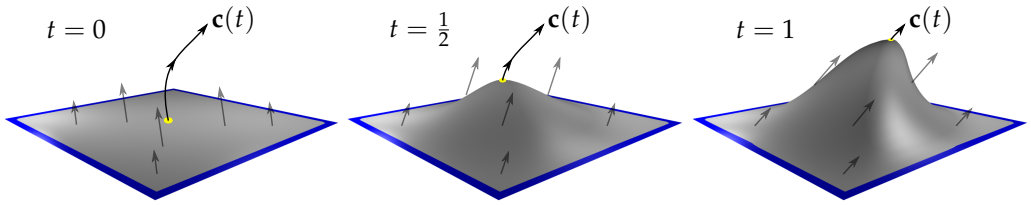


Figure 2: Vector Field-based Continuous Deformation. A small handle region (•) is moved along a guidance curve $\mathbf{c}(t)$, while the border (•) corresponds to the fixed surface part. For every integration step at time t, the current surface geometry and constrained vectors (•) are given, while the remaining vectors (•) are obtained by, e.g., a variational energy optimization.

The problem of direct nonlinear energy optimization can be avoided by relying on vector field-based *continuous deformations* that evolve the deformation over time: the energy minimizer at each infinitesimal time step determines a guidance vector field of the deformation, and the final deformation is obtained as the solution of an ordinary differential equation (ODE). From a technical point of view, iterative solvers for minimizing nonlinear energies are, roughly speaking, replaced by a nonlinear numerical ODE integration method. The latter is a standard numerical problem that is well-understood and can be solved efficiently and reliably. The main benefit of continuous methods consists in the fact that deformation vector field are generally efficient to compute. In particular, we show that it is a linear problem for near-isometric deformations. In addition, and in contrast to single-step methods, all involved energy optimizations yield globally optimal results. Figure 2 illustrates the vector field-based continuous deformation principle.

We emphasize that vector field-based deformations are well suited to optimize for near-isometric deformations in an indirect way, e.g., the methods by Kilian et al. [KMP07] and Solomon et al. [SBBG11a] are formulated continuously. However, not only the hard nonlinear problem of near-isometric deformation can be computed using continuous approaches. In fact, in this work we show that continuous vector field-based deformations are suitable to compute a great variety of different deformation types: for example, with carefully chosen vector fields also volume-preserving deformations can be modeled in straightforward way (see, e.g., Chapters 3 and 5). Moreover, we show how vector field-based deformations are applied to related geometry processing problems like shape optimization (see Chapter 6).

2.1 RELATED WORK

We continue with a summary on methods that are most relevant to the shape manipulation approaches we propose.

2.1.1 Manipulation of Explicit Shapes

A great variety of deformation methods acts on *explicit* shapes. Here, piecewise representations, especially triangle meshes (or, more generally, polygonal surfaces) as well as point-based models have emerged as a de facto standard for geometric shape models. The deformation of a explicitly parameterized shapes consists of a map from the original shape to the deformed shape. *Isometric* maps preserve lengths, which is equivalent to simultaneously preserving angles (*conformal* maps) and area or volume (*authalic* maps). Isometric deformations have been studied extensively in differential geometry and mathematics in general: Efimow [Efi57] theoretically investigates infinitesimal first-order and higher-order deformations. In this work, we consider the vector field-based first-order case. For a rigorous introduction of the differential geometry of such maps we refer to do Carmo [dC92]. Related to shape deformation is parametrization of surfaces, i.e., finding a map between a surface in $3d$ and a planar domain. Naturally, isometry is a desired property for such maps; a pioneering approach is the construction of most-isometric parametrizations by Hormann and Greiner [HG99]. Liu et al. [LZX*08] present hybrid parametrizations that interpolate locally rigid or local similarity transformations, which is similar to the generalized framework for continuous deformations that we propose in Chapter 3. In the following, we consider only shape deformation methods and refer to the survey by Hormann et al. [HPS08] for a discussion of parametrization methods.

LINEAR DEFORMATIONS. Linear shape deformation methods can be modeled in various ways, for instance

- as a variational problem minimizing an energy functional that penalizes certain linearized physically inspired shell-based bending energies (see, e.g., [BK04]),
- as a reconstruction from any kind of differential coordinates (see, e.g., [YZX*04, SCL*04, NSACO05, LSLCO05]),
- as a multiresolution shape decomposition with subsequent detail transfer (see, e.g., [KCVS98, BSPG06]),
- as a projection or Poisson reconstruction after application of a "transformation field" to individual triangles thus over-determining vertex positions (see, e.g., [SP04, XZY*07, ZRKS05, KG08]).

Several methods are closely related and rely on the minimization of certain linear (or potentially linearized) energy functionals, which often results in solving associated Euler-Lagrange equations characterizing an equilibrium state. They all share the goals of feature preservation and establishing smooth transitions towards deformed regions. All of these methods rely on the factorization of a

few or even only a single linear system, a fact that renders these methods interactive. In particular, movement of handles often requires only back-substitution for solving the system. For an in-depth review of linear methods and a discussion of their particular limitations and differences to nonlinear methods we refer to the survey by Botsch and Sorkine [BS08].

NONLINEAR DEFORMATIONS. A popular approach to nonlinear isometry preservation is to restrict deformations locally to near-rigid transformations, i.e., translation and rotation. (Reflection is undesired.) This leads to the notion of the well-established as-rigid-as-possible (ARAP) maps, which where initially introduced for shape interpolation by Alexa et al. [ACL00] and later applied for shape deformation by Igarashi et al. [IMH05] as well as Sorkine and Alexa [SA07]. Until today, there have emerged numerous extensions like [WXXC08, LZX*08, BWG09, SDC09, KFG09, MS11, Lip12, JBK*12, MWCS13], to mention just a few. ARAP approaches minimize a nonlinear energy expressing local rigidity subject to constraints like fixed and displaced points. The classic approach consists in an iterative algorithm that repeatedly estimates local rotations to construct the global deformation until convergence [SA07]. A related iterative algorithm for more general nonlinear geometry processing constraints was proposed by Bouaziz et al. [BDS*12]. Other approaches try to directly optimize the nonlinear problem using dedicated solvers like (quasi) Gauss-Newton solvers [CPSS10]. There are also alternative nonlinear deformation energies that enforce rigidity in form of, e.g., the rest energy of coupled rigid prisms proposed in for of the PriMo method by Botsch et al. for surfaces [BPGK06] and volumes [BPWG07].

Various alternative methods for nonlinear planar, volumetric, or manifold shape deformation exist: Barr [Bar84] and Sederberg and Parry [SP86] pioneered free-form deformations that establish a mapping from the domain onto a warped space. Related are a prominent class of methods that are based on generalized barycentric coordinates and defined on the shape itself or on superposed control structures [HF06, JSW05, JMD*07, LLC08, WBCG09, JBPS11, WBGH11, JWS12, WMZ12]. Besides isometry, here is also a demand for conformal maps, which are produced by none of the above ARAP methods. Nonlinear space deformations are also obtained using moving least-squares techniques [SMW06, ZG07], radial basis functions [BK05b], sweeps of shapes [MJBF02] serving as volume-preserving tools [ACWK04, AWC04], and blended rigid transformations embedded at nodes of space-spanning graphs [SSP07]. Furthermore, there are various hybrid methods, see, e.g., [Coh09]. For a detailed overview and further references we refer to the survey by Gain and Bechmann [GB08]. Nonlinear constraints are also solved on simplified subspaces around the shape [HSL*06] or by cascaded optimizations [KCATLF06, SZT*07]. Hildebrandt et al. employ spectral properties of discrete shell energies for surface modeling and shape space exploration [HSTP11]. Vaxman performs the same operations on planar polyhedral

meshes using fitted affine maps [Vax12]. Furthermore, shapes are often supplemented by additional control structures like local transformations and skeletons for skinning-based deformation propagation [KCvO08, XWY*09, JS11, JBK*12, KS12]. Multiple input shapes are considered by nonlinear example-based methods for skeletal skinning and rigging [LCF00, WPP07, WSLG07, LWP10], as well as shape deformations [SZGP05, DSP06, PJS06, FKY08]. Additionally, more abstract representations of shape features allow for semantical deformations [GSMCO09, BWSK12], semantic deformation transfer [BVGP09], as well as deformations that modify rendering results [RTD*10, MIW13]. The recent survey by Mitra et al. [MWZ*13] gives an overview on structure-aware shape modifications.

Independent of the energy and the particular numerical scheme, the deformation is obtained, either directly or indirectly, as the minimizer of a particular (non-) linear energy that most often depends on the shape geometry at a singular point in time: we refer to these methods as direct *single-step* methods.

CONTINUOUS DEFORMATIONS. Nonlinear *continuous deformations* complement the above single-step methods by also describing all deformations in between the initial and final shape. Such descriptions are useful for, e.g., shape animation and shape space exploration applications. We distinguish between *kinematic* formulations that describe deformations in a purely geometrical way, and *dynamic* formulations that consider deformations that originate from physically motivated forces acting on mass-associated shapes.

The interpolation between shapes is a prominent continuous example: space-time pose interpolation constraints were introduces by Welch and Witkin [WK88]. Additional kinematic interpolation and morphing methods exists for two [RV11] or multiple poses [KG08, WDAH10]. More recently, Cashman and Hormann present a parameterized kinematic representation for pose space exploration [CH12]. Dynamic physics-based simulations can be considered to be a continuous process, see, e.g., the survey of Nealen et al. [NMK*06]. Pentland and Williams pioneered dynamic simulations using modal deformation energy analysis [PW89]. More recently, dynamic continuous simulations of surfaces are often discretized in both space and time and are commonly based on physically motivated thin shell energies [GHDS03, BMWG07]. Example-based dynamic methods employ similar energies [FB11, MTGG11, STC*12, HSvTP12]. A unified formulation for physical simulations of different shape types was proposed by Martin et al. [MKB*10]. Related are physical simulations of cloth [BMF03, GHF*07, EB08, FYK10], elastic rods and threads [BWR*08, BAV*10], and FEM-based (linear) elasticity simulations that often require corotated energies [MDM*02, WBG07, MTPS08, KMBG08, MKB*08, MZS*11, HMT*12, CMT*12].

Vector field-based continuous deformations can be regarded as kinematic methods that integrate shapes defined in higher-dimensional shape spaces. Orig-

inally, curvature flow is performed for surface fairing [DMSB99]. Isometric deformations are guaranteed for integration of exact *Killing vector fields* [1], see, e.g., [Kil88, BBSG09]. Kilian et al. [KMP07] approximate Killing vector fields by a linear optimization. Using a hierarchical space and time discretization they obtain shape space geodesics that yield near-isometric deformations and interpolations of two-manifolds embedded in three-space. More general gradient flows are used by Eckstein et al. [EPT*07] for surface interpolation, matching, and optimization. Wirth et al. [WBRS09] and Heeren et al. [HRWW12] compute geodesics for surface modeling in nonlinear physically-based shape spaces defined by discretized elasticity or thin shell energies, respectively. Shape spaces representing constrained shapes for, e.g., rationalization in freeform architecture are explored by Yang et al. [YYPM11], Barton et al. [BSK*13], and Deng et al. [DBD*13]. Here, tangential first-order shape variations need to be reprojected to the shape space to fulfill prescribed constraints. Solomon et al. [SBBG11a] introduce the notion of *as-Killing-as-possible* (AKVF) vector fields for near-isometric deformations of planar shapes. In contrast to the above approaches, they ensure smoothness by a post-process rather than by a regularization term, and instead of a standard ODE solver, they use planar holomorphic curves as a predictor to construct the trajectories. Von Funck et al. [vFTS06, vFTS07a, vFTS07c, vFTS07b] develop a different family of approaches for continuous kinematic $3d$ surface deformations, which preserve volume by integration of divergence-free vector fields.

2.1.2 Manipulation of Implicit Shapes

There are numerous tasks in computer graphics and geometry processing that are performed preferably on *implicit* shapes like isosurfaces rather than on explicit representations: for instance, changes in shape topology are generally simpler. Self-intersections are avoided by construction. However, for shape modeling there is significantly fewer work on deformation of implicit shapes than for the explicit case. Closely related are volume deformations, which are often modeled by a space deformation as discussed above. For instance, in medical applications this includes non-rigid registration of data sets: the data often describes certain material properties, like soft-tissue, and manipulations are required to be physically-based. We refer to the survey by Chen et al. [CCI*07] for a general and broad review of volume deformations with a discussion on various data representations and applications.

Several classic approaches to modeling with isosurfaces are based on level set methods [OF01]. Museth et al. [MBWB02] define shape editing operations for

1 Vector fields generating isometries are named after the German mathematician Wilhelm Karl Joseph Killing (1847-1923), who made important contributions to the fields of Lie algebras and non-Euclidean geometries [Kil88].

smoothing, offsetting, and blending. Desbrun and Cani-Gascuel [DCG98] use a level set approach to define active implicit surfaces inspired by geometric snakes. Level set methods focusing on explicitly handling topology preservation were presented in [AS05]. As these approaches act only on a particular isosurface, computations can be limited to a narrow band of the scalar field. Various other approaches are physically plausible and employ particle systems and a Lagrangian integration scheme to simulate and animate surface material. In computer graphics, this was pioneered by Desbrun and Cani-Gascuel [DG95], who minimize local volume variations. Another potential goal for such approaches is the emulation of virtual clay [MQW01, CA06]. Alternative sculpting methods implement virtual carving operations [PF01] based on adaptive distance fields [FPRJ00]. Yet a different approach to volume deformation consists in simulating networks of geometric primitives, e.g., using a chain mail analogy [Gib97]. Mullen et al. [MMTD07] present an Eulerian framework for geometry processing problems like smoothing and gradient flow-based surface offsetting. Their method handles multiple isosurfaces simultaneously but does not support surface modeling. A crucial aspect of any deformation method is interactivity: the user requires real-time feedback. The use of modern graphics hardware can speed up computations or is even mandatory to achieve interactive frame rates on volume data [RSSG01, WR01, GW06, SBH07, RWE08]. For a more in-depth overview, we refer to the survey by Hadwiger et al. [HKRW06].

3

VECTOR FIELD-BASED CONTINUOUS DEFORMATION ENERGIES

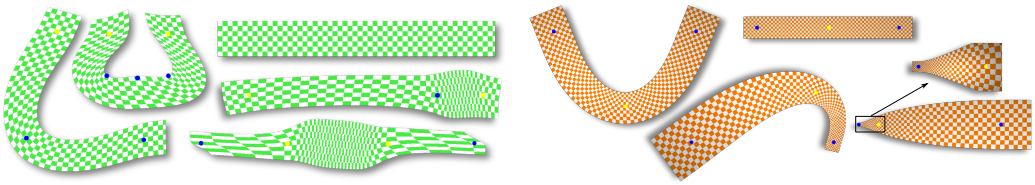
In this chapter, we introduce continuous deformations formally. We derive energies that determine different kinds of vector field-based deformations in $2d$ and $3d$. The main focus will lie on a generalized formulation of vector field energies that determine a one-parameter family of different continuous deformations: the generalized energy describes near-isometric, near-conformal, and close-to-authalic deformations, and supports nonhomogeneous and anisotropic behavior. Our approach is more general – but not more complicated – than previous methods. In particular, we show that the recently proposed planar deformations by Solomon et al. [SBBG11a], which are based on approximate Killing vector fields (AKVFs), constitute a special case of our energy.

3.1 CONTINUOUS SHAPE DEFORMATIONS

In this work, we considere geometric *shapes* that are compact d-dimensional point sets $\mathcal{D}_d \subset \mathbb{R}^d$ of intrinsic dimensionality $\dim(\mathcal{D}_d) \leq d$, e.g., univariate curves $\mathcal{D}_2$ with $\dim(\mathcal{D}_2) = 1$ embedded in $\mathbb{R}^2$. A *continuous shape deformation* is then given as a map

$$\mathbf{f}(\mathbf{x}, t) \colon \mathcal{D}_d^0 \times \mathbb{R} \to \mathbb{R}^d \ ,$$

i.e., a time-parameterized map from an initial shape $\mathcal{D}_d^0$ to a deformed state in $\mathbb{R}^d$. In this work, we primarily consider the practically most relevant dimensions $d = 2$ and $d = 3$ of planar, surface, and volumetric deformations, respectively, but also provide generalizations for higher dimensions $d > 3$. We assume that $\mathbf{f}$ is differentiable. $\mathcal{D}_d^0$ can be considered to be a point in a high-dimensional space and describes the realization of the initial shape in some Euclidean space, e.g., a two-variate surface shape embedded in $3d$. In the literature, this space is often considered as a so-called *shape space* with a choice of a suitable metric [KMP07]. The deformed shape at time t is expressed as the image $\mathbf{f}(\mathcal{D}_d^0, t)$. We use the short notation $\mathcal{D}_d = \mathbf{f}(\mathcal{D}_d^0, t) \equiv \mathbf{x}(t)$ for the deformed shape at the current time



(a) Near-isometric. (b) Near-conformal.

Figure 3: $2d$ Deformation Examples. On the straight strip some vertices were fixed (•) while some vertices were moved (•) (all models have the same scale). The deformations are generated by AMAP (9) and ACAP (13) vector fields. Note the approximate length preservation in (a), and the preservation of angles and the area deviation in (b).

t, which is clear from the context. The univariate curve $\mathbf{x}$ describes the time-parameterized continuous shape deformation.

In this chapter, we concentrate our studies on continuous solid deformations with $\dim(\mathcal{D}_d^0) = d$. Related continuous space deformations will be presented in Chapters 5 and 6. Surface deformations in $3d$ will be studied in Chapter 4.

A powerful way to describe continuous deformations is to define them by kinematic space-time integration along suitable vector fields. We define the velocity tangent vector field of $\mathbf{f}$ as the vector field $\mathbf{v}(\mathbf{x}, t) = \frac{\partial}{\partial t}\mathbf{f}(\mathbf{x}, t)$. Given the initial shape $\mathcal{D}_d^0$ and the vector field $\mathbf{v}$, the deformation $\mathbf{f}$ can be reconstructed by solving the ordinary differential equation (ODE) given as the initial value problem

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{v}(\mathbf{x}, t) \quad \text{with} \quad \mathbf{x}(0) \equiv \mathcal{D}_d^0 \,. \tag{1}$$

Therefore, the deformation of the shape $\mathcal{D}_d^0$ is completely defined by $\mathbf{v}$, and properties of the vector field directly influence properties of the deformation. For example, applying vector fields that are divergence-free in the embedding Euclidean space yields volume-preserving authalic deformations.

3.2 CONTINUOUS METRIC ENERGIES

In the following, we derive conditions on $\mathbf{v}$ that lead to near-isometric, near-conformal, as well as near-authalic maps $\mathbf{f}$ of any dimension. The conditions are characterized as the minimizers of certain energies w.r.t. interpolation constraints on $\mathbf{v}$. Figure 3 shows examples for deformations that were determined by this kind of vector fields. The different energies are characterized by the types of deformations they define.

3.2.1 Characteristic Deformations

For a *single-step* (i.e., not time-dependent) deformation $\mathbf{f}(\mathbf{x})\colon \mathcal{D}_d^0 \to \mathbb{R}^d$ with *deformation gradient tensor field* $\mathbf{D} = \nabla\mathbf{f}$, the first fundamental form $\mathcal{I}$ of $\mathbf{f}$ has the particularly simple form

$$\mathcal{I} = \mathbf{D}^{\mathrm{T}}\,\mathbf{D}\,.$$

Therefore, the *singular values* σ_i of $\mathbf{D}$ are square roots of the *eigenvalues* λ_i of $\mathcal{I}$. Then the following equivalent local properties of the deformation map $\mathbf{f}$ can be shown:

$$\begin{aligned}
&1.\ \mathbf{f} \text{ is isometric} && \Leftrightarrow\ \mathcal{I} = \mathbf{I} && \Leftrightarrow\ \lambda_i = 1 && \Leftrightarrow\ \sigma_i = 1, && (2)\\
&2.\ \mathbf{f} \text{ is conformal} && \Leftrightarrow\ \mathcal{I} = \mu\,\mathbf{I} && \Leftrightarrow\ \frac{\lambda_i}{\lambda_j} = 1 && \Leftrightarrow\ \frac{\sigma_i}{\sigma_j} = 1, && (3)\\
&3.\ \mathbf{f} \text{ is authalic} && \Leftrightarrow\ \det(\mathcal{I}) = 1 && \Leftrightarrow\ \textstyle\prod_{i=1}^{d}\lambda_i = 1 && \Leftrightarrow\ \textstyle\prod_{i=1}^{d}\sigma_i = 1 && (4)
\end{aligned}$$

See, e.g., the work of Floater and Hormann [FH05] for a derivation and discussion in the context of surface parametrizations ($d = 2$). Note that parametrizations can be regarded as single-step deformations between surfaces embedded in $3d$ and planar $2d$ shapes that minimize some form of deviation of these properties. In this chapter, we study corresponding continuous deformations that are described by instantaneous deformation energies in the generating vector fields.

If deformations are parameterized by time t, i.e., we have continuous deformations $\mathbf{f}(\mathbf{x}, t)$ and deformation gradients $\mathbf{D}(\mathbf{x}, t)$, these properties can be differentiated in order to obtain defining conditions on the vector field of the continuous deformation. Specifically, we apply the matrix algebra described by Minka [Min01] to obtain matrix derivatives w.r.t. t: the *differential* $\mathrm{d}\mathbf{y}(\mathbf{x})$ is defined to be the part of $\mathbf{y}(\mathbf{x} + \mathrm{d}\mathbf{x}) - \mathbf{y}(\mathbf{x})$ that is linear in $\mathrm{d}\mathbf{x}$. Differentials are obtained by iteratively applying a set of differentiation rules. After transformation into canonical form the matrix derivative can directly be read off.

3.2.2 Isometric Energies

Exact isometric deformations are only possible for very simple shapes, e.g., for developable surfaces, and a very restrictive set of user-constraints. To enable deformations of a greater variety of shapes without the restriction of user-constraints, the isometric deformation property is generally relaxed to *near-isometric* deformations that should be "close" to exact isometries. Therefore, measures for the *deviation* from isometry are required. Depending on the ansatz of deviation measurement, different types of near-isometric energies are defined. We continue to present two different models: Killing and metric energies. We

also propose a third model of skew-symmetric energies, which turn out to be equivalent to Killing energies.

KILLING ENERGY. The matrix derivative of the isometry property (2) is obtained by deducing the differential w.r.t. t, which gives

$$\mathrm{d}\mathbf{D}^\mathrm{T}\,\mathbf{D} + \mathbf{D}^\mathrm{T}\,\mathrm{d}\mathbf{D} = \mathbf{0}$$

using the product rule $\mathrm{d}(\mathbf{A}\,\mathbf{B}) = \mathrm{d}\mathbf{A}\,\mathbf{B} + \mathbf{A}\,\mathrm{d}\mathbf{B}$ and $\mathrm{d}\mathbf{I} = \mathbf{0}$. This equality has to hold for every time t of the continuous deformation. Specifically, for $t = 0$ we have $\mathbf{D}(\mathbf{x}_0, 0) = \mathbf{I}$ for all $\mathbf{x}_0 \in \mathcal{D}_d^0$, and by using $\mathrm{d}\mathbf{D}^\mathrm{T} = \mathbf{J}$, where $\mathbf{J}(\mathbf{v}, \mathbf{x})$ is the Jacobian of the tangent vector field $\mathbf{v}$ of $\mathbf{f}$, we obtain

$$\mathbf{J}^\mathrm{T} + \mathbf{J} = \mathbf{0} \tag{5}$$

as the condition for $\mathbf{f}$ to be isometric expressed in the vector field $\mathbf{v}$ of the continuous deformation. Equation (5) corresponds to the constraint that exact isometric deformations are generated by infinitesimal instantaneous rotations, as the symmetric part of their vector field Jacobians, which are skew-symmetric then, vanishes.

Using the Frobenius norm the L^2 deviation of (5) over a shape $\mathcal{D}_d$

$$E_{\text{AKVF}}(\mathbf{v}) = \int_{\mathcal{D}_d} \|\mathbf{J}^\mathrm{T} + \mathbf{J}\|_F^2 \,\mathrm{d}\mathbf{x} \tag{6}$$

is called *Killing energy*. Vector fields minimizing E_{AKVF} are called *approximate Killing vector fields* (AKVF). They are used by Solomon et al. [SBBG11a] for the $d = 2$ case to compute *as-Killing-as-possible* (AKAP) planar deformations that are near-isometric. Higher-dimensional cases ($d > 2$) are also well-defined by (6). Note that the Jacobian is linear in the unknown vector field $\mathbf{v}$ as differentiation is a linear operation, i.e., there exists a linear gradient operator $\mathbf{G}$ on $\mathcal{D}_d$ with $\mathbf{J}^\mathrm{T} = \mathbf{G}(\mathbf{v})$. See Appendix A.1 for a derivation of gradient operators on different shape discretizations. Therefore, the energy E_{AKVF} is quadratic in $\mathbf{v}$, and the corresponding variational optimization of E_{AKVF} on an appropriate domain discretization leads to a linear system that can be solved efficiently for the optimal vector field.

METRIC ENERGY. The standard Killing energy (6) uses the Frobenius norm of (5) to measure deviation from isometry. We propose a related energy that measures another form of deviation from isometry that is *not* based on a L^2 deviation of (5). Informally spoken, our energy directly observe an infinitesimally small line segment and measures change of squared length under an infinitesimal integration step in $\mathbf{v}$. This is done for all possible infinitesimal segments, i.e., we integrate the change of length over all possible directions. We call this

energy *metric* as distance variations are measured explicitly. We start with the derivation of the $2d$ case followed by the $3d$ case.

In order to measure the variation of length under integration in $\mathbf{v}$, we consider a $2d$ line segment $\mathcal{S}$ between points $\mathbf{x}_0$ and $\mathbf{x}_1 = \mathbf{x}_0 + r_1\,\mathbf{r}_1$ for a unit direction $\mathbf{r}_1$ and segment length r_1. The flow of $\mathcal{S}$ in $\mathbf{v}$ is given by $\mathbf{x}_0'(h) = \mathbf{x}_0 + \int_0^h \mathbf{v}(\mathbf{x}_0'(s))\,\mathrm{d}s$ and $\mathbf{x}_1'(h) = \mathbf{x}_1 + \int_0^h \mathbf{v}(\mathbf{x}_1'(s))\,\mathrm{d}s$ for some (finite) integration time $h > 0$. This integration induces the (finite) squared length variation

$$d(h) = \|\mathbf{x}_1 - \mathbf{x}_0\|^2 - \|\mathbf{x}_1'(h) - \mathbf{x}_0'(h)\|^2 \,.$$

Since we are interested in instantaneous variations (i.e., the squared length variation of an infinitesimal small line segment during an infinitesimal small integration time) only, we consider the point-wise directional limit

$$d_0(\mathbf{r}_1) = \lim_{r_1 \to 0, h \to 0} \frac{d(h)}{r_1^2\, h} \,.$$

d_0 measures the instantaneous squared length variation at $\mathbf{x}_0$ in the direction $\mathbf{r}_1$. We obtain the point-wise squared metric energy $e_{\text{METR}}(\mathbf{x}_0, \mathbf{v})$ at $\mathbf{x}_0$ by considering all possible line segment directions given by $\mathbf{r}_1(\alpha) = (\cos(\alpha)\,, \sin(\alpha)\,)^{\mathrm{T}}$:

$$e_{\text{METR}}(\mathbf{x}_0, \mathbf{v}) = \frac{1}{2\pi} \int_0^{2\pi} d_0^l(\mathbf{r}_1(\alpha))^2\,\mathrm{d}\alpha \,. \tag{7}$$

It can be shown that (7) has the following closed-form solution that depends only on the Jacobian of $\mathbf{v}$ [1]:

$$\begin{aligned} e_{\text{METR}}(\mathbf{x}_0, \mathbf{v}) &= u_x^2 + v_y^2 + \frac{1}{2}\left(u_y + v_x\right)^2 + \frac{1}{2}\left(u_x + v_y\right)^2 \\ &= c\,\left(\|\mathbf{J} + \mathbf{J}^{\mathrm{T}}\|_F^2 + 2\,\operatorname{Tr}(\mathbf{J})^2\right) \,. \end{aligned} \tag{8}$$

Here, $\mathbf{J} = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}$ denotes the Jacobian matrix of $\mathbf{v} = (u(x,y), v(x,y))^{\mathrm{T}}$ at $\mathbf{x}_0$, $\operatorname{Tr}(\cdot)$ is the trace of a matrix, and $c = 1/4$ is a constant factor that does not influence the optimal solution. The total *metric energy* of a vector field $\mathbf{v}$ on $\mathcal{D}_2$ is now given by

$$E_{\text{METR}}(\mathbf{v}) = \int_{\mathcal{D}_2} e_{\text{METR}}(\mathbf{x}, \mathbf{v})\,\mathrm{d}\mathbf{x} \,. \tag{9}$$

We call vector fields that minimize this energy *approximate metric vector fields* (AMVF). Figure 3a shows examples for deformations that were determined by this kind of vector fields.

1 The proof of equivalence is lengthy but consists only of basic algebraic transformations and is therefore omitted in this work. — We provide "derivations" in form of Maple scripts for all closed-form solutions of integrals in this chapter with the additional material folder `addmaterial/contdef`.

We derive a similar energy for $d = 3$ dimensions using the same ansatz as above for $d = 2$. Again, we integrate over all possible configurations of an infinitesimal integration step of an infinitesimally small line segment between two points $\mathbf{x}_0$ and $\mathbf{x}_1 = \mathbf{x}_0 + r_1 \mathbf{r}_1$ in $3d$. The main difference to the $2d$ case is that angles in the plane now have to be replaced by solid angles to perform local volumetric integration. For the spherical parametrization of the unit direction $\mathbf{r}_1(\alpha, \beta) = (\cos(\alpha)\cos(\beta), \sin(\alpha)\cos(\beta), \sin(\beta))^{\mathrm{T}}$ we obtain the point-wise quadratic metric energy as the integral

$$e_{\textsc{Metr3d}}(\mathbf{x}_0, \mathbf{v}) = \frac{1}{4\pi} \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos(\beta)\ d_0(\mathbf{r}_1(\alpha, \beta))^2 \,\mathrm{d}\beta \,\mathrm{d}\alpha \,, \tag{10}$$

which again has the closed-form solution

$$e_{\textsc{Metr3d}}(\mathbf{x}_0, \mathbf{v}) = c \left(\|\mathbf{J} + \mathbf{J}^{\mathrm{T}}\|_F^2 + 2\ \mathrm{Tr}(\mathbf{J})^2 \right) \tag{11}$$

with $c = 1/30$. Interestingly, the factors of (8) and (11) only differ in the constant c, although their dimensions differ. This property also holds for higher dimensions. The total $3d$ metric energy is then obtained as

$$E_{\textsc{Metr3d}}(\mathbf{v}) = \int_{\mathcal{D}_3} e_{\textsc{Metr3d}}(\mathbf{x}, \mathbf{v}) \,\mathrm{d}\mathbf{x} \,.$$

We again call the minimizers of this energy *approximate metric vector fields* (AMVF). In the following we will use the identifiers Metr and Metr3d synonymously whenever the context is clear.

3.2.3 Conformal Energy

We continue to derive conformal energies that measure the deviation of angles induced by vector field integration. The differential of (3) is given by

$$\mathrm{d}\mathbf{D}^{\mathrm{T}}\mathbf{D} + \mathbf{D}^{\mathrm{T}}\,\mathrm{d}\mathbf{D} = \mathrm{d}\mu\,\mathbf{I} \,.$$

We again evaluate it at $t = 0$, and by setting $\mathrm{d}\mu = \alpha$ we obtain

$$\mathbf{J}^{\mathrm{T}} + \mathbf{J} = \alpha\,\mathbf{I}$$

as the condition for the continuous deformation $\mathbf{f}$ to be conformal. Note that here α, the differential of the scaling factor μ, is an additional degree of freedom stating the fact that instantaneous uniform scaling is conformal for every scaling factor.

We derive a point-wise energy e_{CONF} that measures the L^2 deviation of this conformality condition. The construction of the energy holds for any dimension d from which the important two and three-dimensional special cases can be obtained:

$$\begin{aligned} e_{\text{CONF}} &= \|\mathbf{J}^\mathsf{T} + \mathbf{J} - \alpha\,\mathbf{I}\|_F^2 \\ &= \operatorname{Tr}\Big(\left(\mathbf{J}^\mathsf{T} + \mathbf{J} - \alpha\,\mathbf{I}\right)^\mathsf{T}\left(\mathbf{J}^\mathsf{T} + \mathbf{J} - \alpha\,\mathbf{I}\right)\Big) \\ &= \operatorname{Tr}\Big(\left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right)^\mathsf{T}\left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right)\Big) + \operatorname{Tr}\Big(-2\alpha\left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right) + \alpha^2\mathbf{I}\Big) \\ &= \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_F^2 - 2\alpha\operatorname{Tr}\Big(\mathbf{J}^\mathsf{T} + \mathbf{J}\Big) + d\,\alpha^2 \\ &= \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_F^2 - 4\alpha\operatorname{Tr}(\mathbf{J}) + d\,\alpha^2 \end{aligned} \tag{12}$$

This energy formulation still depends on the scaling factor α. To obtain an expression that is independent of this parameter we consistently set it to the value that minimizes the value of the energy. That is, we solve $\nabla_\alpha\, e_{\text{CONF}} \stackrel{!}{=} 0$ for α, which gives $\alpha = \frac{2}{d}\operatorname{Tr}(\mathbf{J})$. Inserting this result into (12) we obtain

$$e_{\text{CONF}} = \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_F^2 - \frac{4}{d}\operatorname{Tr}(\mathbf{J})^2$$

for the general d-dimensional point-wise conformal energy in the vector field of the continuous deformation. The total *conformal energy* of the vector field is then given by

$$E_{\text{CONF}}(\mathbf{v}) = \int_{\mathcal{D}_d} \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_F^2 - \frac{4}{d}\operatorname{Tr}(\mathbf{J})^2\,\mathrm{d}\mathbf{x}\,.$$

Again, this energy is quadratic in the vector field and is defined for any dimension d. We call vector fields minimizing this energy *approximate conformal vector fields* (ACVF). The important low-dimensional special cases are

$$\begin{aligned} E_{\text{CONF2D}}(\mathbf{v}) &= \int_{\mathcal{D}_2} \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_F^2 - 2\,\operatorname{Tr}(\mathbf{J})^2\,\mathrm{d}\mathbf{x} \quad \text{and} \\ E_{\text{CONF3D}}(\mathbf{v}) &= \int_{\mathcal{D}_3} \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_F^2 - \frac{4}{3}\,\operatorname{Tr}(\mathbf{J})^2\,\mathrm{d}\mathbf{x}\,. \end{aligned} \tag{13}$$

See Figure 3b for example deformations that were determined by vector fields that minimize this type of energy.

3.2.4 Authalic Energy

In order to obtain the vector field condition for the continuous deformation to be authalic, i.e., volume-preserving, we differentiate the left hand side of (4) using the differentiation rule $\mathrm{d}\det(\mathbf{A}) = \det(\mathbf{A})\,\operatorname{Tr}\left(\mathbf{A}^{-1}\,\mathrm{d}\mathbf{A}\right)$:

$$\mathrm{d}\det\Big(\mathbf{D}^\mathsf{T}\mathbf{D}\Big) = 2\,(\mathrm{d}\det(\mathbf{D})\,)\,\det(\mathbf{D})$$

$$
\begin{aligned}
&= 2\left(\det(\mathbf{D})\ \mathrm{Tr}\Big(\mathbf{D}^{-1}\,\mathrm{d}\mathbf{D}\Big)\right)\ \det(\mathbf{D}) \\
&= \mathrm{Tr}\Big(2\ \det(\mathbf{D})^2\ \mathbf{D}^{-1}\,\mathrm{d}\mathbf{D}\Big)
\end{aligned} \tag{14}
$$

Evaluating (4) and (14) once again at $t = 0$, and by using $\mathrm{d}\mathbf{D}^{\mathrm{T}} = \mathbf{J}$ the authalic condition on the vector field simplifies to

$$\mathrm{Tr}(\mathbf{J}) = 0,$$

which states that the vector field has to be divergence-free as $\mathrm{Tr}(\mathbf{J}) = \nabla^{\mathrm{T}}\mathbf{v}$ is the divergence of $\mathbf{v}$. The corresponding L^2 point-wise authalic energy $e_{\textsc{Auth}} = \left(\nabla^{\mathrm{T}}\mathbf{v}\right)^2$ yields the total authalic energy

$$E_{\textsc{Auth}}(\mathbf{v}) = \int_{\mathcal{D}_d}\left(\nabla^{\mathrm{T}}\mathbf{v}\right)^2 \mathrm{d}\mathbf{x}$$

for any dimension d. We call vector fields minimizing this energy *approximate authalic vector fields* (AAVF).

3.2.5 Alternative Isometric Energy

For completeness, we provide a derivation of a different energy form that yields *equivalent* near-isometric minimizers as the Killing energy. The Killing energy (6) measures deviation from skew-symmetry using the L^2 deviation of (5). Alternatively, skew-symmetry can be measured by minimizing the L^2 distance to the closest skew-symmetric matrix in the Frobenius norm, which also gives near-isometric deformations. We do only consider the $2d$ case as this ansatz is more complex to generalize to higher dimensions. Let $\mathbf{S} = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$ be a basis of skew-symmetric tensors in $2d$. Pointwise energies are then given by

$$
\begin{aligned}
e_{\textsc{Skew2D}} &= \|\mathbf{J} - \alpha\,\mathbf{S}\|_F^2 \\
&= \mathrm{Tr}\Big(\mathbf{J}^{\mathrm{T}}\mathbf{J} - \alpha\,\mathbf{J}^{\mathrm{T}}\mathbf{S} - \alpha\,\mathbf{S}^{\mathrm{T}}\mathbf{J} + \alpha^2\,\mathbf{I}\Big) \\
&= \|\mathbf{J}\|_F^2 - 2\alpha\ \mathrm{Tr}\Big(\mathbf{S}^{\mathrm{T}}\mathbf{J}\Big) + 2\alpha^2
\end{aligned}
$$

using the skew-symmetry of $\mathbf{S}$. Solving $\nabla_\alpha\, e_{\textsc{Skew2D}} \stackrel{!}{=} 0$ for α gives $\alpha = \frac{1}{2}\,\mathrm{Tr}\big(\mathbf{S}^{\mathrm{T}}\mathbf{J}\big)$, such that we obtain a point-wise energy of

$$e_{\textsc{Skew2D}} = \|\mathbf{J}\|_F^2 - \frac{1}{2}\,\mathrm{Tr}\Big(\mathbf{S}^{\mathrm{T}}\mathbf{J}\Big)^2, \tag{15}$$

and a total energy of

$$E_{\textsc{Skew2D}}(\mathbf{v}) = \int_{\mathcal{D}_2} \|\mathbf{J}\|_F^2 - \frac{1}{2}\,\mathrm{Tr}\Big(\mathbf{S}^{\mathrm{T}}\mathbf{J}\Big)^2 \mathrm{d}\mathbf{x}\,. \tag{16}$$

Vector fields that minimize this energy are called *approximate skew-symmetric vector fields* (ASSVF). We will show that (6) and (16) have equal minimizers in Section 3.4.3. Due to this equivalence we will not further analyze properties of this energy in this work and continue to give a unification of all other proposed energies.

3.3 A GENERALIZED FAMILY OF ENERGIES

In the following, we relate the near-isometric, near-conformal, and near-authalic energies to derive a generalized energy. It is a one-parameter family of energies that determines smooth blends between the different deformation types.

We define the energy terms

$$q(\mathbf{x},\mathbf{v}) := \|\mathbf{J}(\mathbf{x}) + \mathbf{J}(\mathbf{x})^{\mathrm{T}}\|_F^2 \quad \text{and} \quad r(\mathbf{x},\mathbf{v}) := \mathrm{Tr}(\mathbf{J}(\mathbf{x}))^2 \ . \tag{17}$$

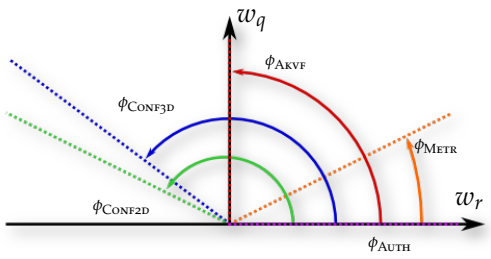
Then all point-wise energies introduced so far can be expressed as linear combinations of q and r. Uniform energy scaling does not change the minimizing vector field. Therefore, we can describe all energies as a one-parameter family of *generalized metric energies* parameterized by a scalar ϕ:

$$E_\phi(\mathbf{v}) = \int_{\mathcal{D}_d} w_q(\phi)\ q(\mathbf{x},\mathbf{v}) \ + \ w_r(\phi)\ r(\mathbf{x},\mathbf{v})\,\mathrm{d}\mathbf{x}\ ,$$

where the weights $w_q(\phi) := \sin(\phi)$ and $w_r(\phi) := \cos(\phi)$ are chosen to have specific ratios. ϕ can vary in the interval $(0,\ \pi - \arctan d/4]$. Then the parameter of the isometric energies is given by $\phi = \phi_{\mathrm{AKVF}} = \pi/2$, resp. $\phi = \phi_{\mathrm{Metr}} = \arctan 1/2$, and the minimizers of $E_{\phi_{\mathrm{AKVF}}}(\mathbf{v})$ and $E_{\mathrm{AKVF}}(\mathbf{v})$, resp. $E_{\phi_{\mathrm{Metr}}}(\mathbf{v})$ and $E_{\mathrm{Metr}}(\mathbf{v})$ are equal. Furthermore, the d-dimensional conformal energy is given by $\phi = \phi_{\mathrm{Conf}} = \pi - \arctan d/4$. The authalic energy is recovered for $\phi = \phi_{\mathrm{Auth}} = 0$. We note that volume preservation is not a sufficient condition for uniquely defining $\mathbf{v}$. However, adding a small amount of q to E_ϕ (i.e., choosing ϕ slightly above zero) gives unique solutions corresponding to near-authalic deformations.

Figure 4 illustrates and summarizes the different choices of ϕ. Note that for $\phi > \pi/2$, E_ϕ contains negative quadratic terms. However, due to the definition of the point-wise conformal energy (12) as a squared matrix norm, it is guaranteed that E_ϕ is non-negative (as long as $\phi \leq \pi - \arctan d/4$), and that an unique minimizer exists. Obviously, general vector fields cannot have multiple characteristics simultaneously. For example, a non-rigid vector field cannot always preserve local angles and volumes at the same time. Vector fields defining continuous deformation inherit this fundamental property from single-step deformations.

We note that in (17) only the symmetric part $\frac{1}{2}(\mathbf{J} + \mathbf{J}^{\mathrm{T}})$ of the vector field Jacobian $\mathbf{J}$ contributes to the energy. The trace-free skew-symmetric part $\frac{1}{2}(\mathbf{J} - \mathbf{J}^{\mathrm{T}})$ does not contribute to the energy as it represents the infinitisimal rotational part of $\mathbf{J}$



Energy	w_q :	w_r	ϕ
METR $2d$ & $3d$	1 :	2	$\arctan 1/2$
CONF $2d$	1 :	-2	$\pi - \arctan 1/2$
CONF $3d$	1 :	$-4/3$	$\pi - \arctan 3/4$
AKVF $2d$ & $3d$	1 :	0	$\pi/2$
AUTH $2d$ & $3d$	0 :	1	0

Figure 4: Energy Parameter Domain for $2d$ and $3d$. The different energies obtained from the general metric energy $E_\phi(\mathbf{v})$ are linear subspaces in the visualized domain of weights w_q and w_r, i.e., every pair of weights in a subspace yields the same energy minimizer. However, energies may not have unique minimizers, like the AUTH energy in the limit.

that induces no distortion. Hence, our generalized family of energies is invariant to superimposed rigid vector fields.

ANISOTROPIC ENERGIES. The energy formulations presented so far are *isotropic* as distortions are measured in an uniform way for every shape direction. We model *anisotropic* energies by replacing isotropic Frobenius norms with anisotropic norms $\|\cdot\|_{\mathbf{B}}^2$ defined by a rank-2 tensor fields of symmetric positive-definite matrices $\mathbf{B}(\mathbf{x})$: $\|\mathbf{A}\|_{\mathbf{B}}^2 = \operatorname{Tr}(\mathbf{A}^T \mathbf{B} \mathbf{A})$. Effectively, this way we modulate the isotropic point-wise energies to obtain anisotropic behavior. For example, the point-wise energy (12) then becomes

$$\begin{aligned} e_{\text{CONF}} &= \|\mathbf{J}^T + \mathbf{J} - \alpha \mathbf{I}\|_{\mathbf{B}}^2 \\ &= \|\mathbf{J}^T + \mathbf{J}\|_{\mathbf{B}}^2 + \operatorname{Tr}\Big(-2\alpha \left(\mathbf{J}^T \mathbf{B} + \mathbf{B} \mathbf{J}\right)\Big) + \operatorname{Tr}\left(\alpha^2 \mathbf{B}\right) \\ &= \|\mathbf{J}^T + \mathbf{J}\|_{\mathbf{B}}^2 - 4\alpha \operatorname{Tr}(\mathbf{B} \mathbf{J}) + \gamma \alpha^2 \\ &= \|\mathbf{J}^T + \mathbf{J}\|_{\mathbf{B}}^2 - \frac{4}{\gamma} \operatorname{Tr}(\mathbf{B} \mathbf{J})^2 \ , \end{aligned}$$

where we have set $\gamma = \operatorname{Tr}(\mathbf{B})$ and used the identity $\operatorname{Tr}(\mathbf{A}\mathbf{B}) = \operatorname{Tr}(\mathbf{B}\mathbf{A})$ together with the solution of $\nabla_\alpha e_{\text{CONF}} \stackrel{!}{=} 0$, which is $\alpha = \frac{2}{\gamma} \operatorname{Tr}(\mathbf{B}\mathbf{J})$. In general, the anisotropic energy factors become

$$q = \|\mathbf{J} + \mathbf{J}^T\|_{\mathbf{B}}^2 \quad \text{and} \quad r = \operatorname{Tr}(\mathbf{B}\mathbf{J})^2 , \tag{18}$$

together with $w_r = -\frac{4}{\gamma}$ for the conformal energies. In the special case of $\mathbf{B} = \mathbf{I}$ the isotropic cases are recovered as then $\|\cdot\|_{\mathbf{B}} \equiv \|\cdot\|_F$ and $\gamma = d$.

A simple way to model the local anisotropic norm field is to attach a local d-dimensional orthonormal basis $\{\mathbf{t}_k\}$ together with local weights $\{\lambda_k\}$, $\lambda_k > 0$ to each shape point. Then $\mathbf{B}$ is given by the spectral decomposition

$$\mathbf{B} = \begin{bmatrix} \mathbf{t}_1 & \cdots & \mathbf{t}_d \end{bmatrix} \operatorname{diag}(\lambda_1, \cdots, \lambda_d) \begin{bmatrix} \mathbf{t}_1 & \cdots & \mathbf{t}_d \end{bmatrix}^T .$$

The weight λ_k indicates the anisotropic contribution of the measured error in direction $\mathbf{t}_k$. Note that the basis directions need to be transformed during deformation according to local shape rotations.

3.4 DISCRETE SETTING

In this section, we provide suitable discretizations of the domains for the vector field energies introduces above. Let $\mathcal{M} = (\mathcal{V}, \mathcal{T})$ be a piecewise linear partition of $\mathcal{D}_d$ (at a particular time t) with sets of vertices $\mathcal{V}$ and cells (or simplices) $\mathcal{T}$ (triangle meshes for $d = 2$ and tetrahedral meshes for $d = 3$). From hereon, we will omit dimension subscripts d whenever the context is clear. Furthermore, let $|\mathcal{V}|$ denote the number of vertices, and $\mathbf{x}_i$ for vertices $i \in \mathcal{V}$ indicate d-dimensional vertex coordinates. The piecewise linear coordinate functions define the shape of the realization of $\mathcal{M}$. We also express vector fields $\mathbf{v}$ as piecewise linear functions on $\mathcal{M}$: $\mathbf{v}$ is given as d-dimensional nodal values $\mathbf{v}_i$, $i \in \mathcal{V}$; we write $\mathbf{v}$ as the a single $(d\,|\mathcal{V}|)$-dimensional column vector $\mathbf{v} = ({\mathbf{v}_1}^{\mathrm{T}}, \ldots, {\mathbf{v}_{|\mathcal{V}|}}^{\mathrm{T}})^{\mathrm{T}}$. Its piecewise constant Jacobian field is given as matrices $\mathbf{J}_c$ on each cell $c \in \mathcal{T}$.

3.4.1 Energy Discretization

In the discrete setting E_ϕ is a quadratic form in the unknown vector field: $E_\phi(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}\,\mathbf{E}_\phi\,\mathbf{v}$. The $d\,|\mathcal{V}| \times d\,|\mathcal{V}|$ matrix $\mathbf{E}_\phi$ is a symmetric positive-semi-definite sparse matrix defining E_ϕ. The rank deficient of $\mathbf{E}_\phi$ depends on ϕ, but for $\phi > 0$ the rank deficit is bounded from above by d^2. Therefore, the vector field has to be prescribed in form of user-specified boundary constraints at at least d vertices for well-defined energy minimizing vector fields. In practice this is no problem as $d < |\mathcal{V}|$ for any mesh and in general $d \ll |\mathcal{V}|$ for relevant meshes.

The matrix $\mathbf{E}_\phi$ can be constructed in an element-wise way. With the Jacobians being constant on each cell, the coefficients of $\mathbf{E}_\phi$ are the sum of permuted matrices $\mathbf{E}_\phi^c$ that capture the local error E_ϕ^c on cell c as

$$\begin{aligned} E_\phi^c(\mathbf{v}) &= \int_{\mathcal{D}^c} w_q(\phi)\; q(\mathbf{x}, \mathbf{v}) \;+\; w_r(\phi)\; r(\mathbf{x}, \mathbf{v})\,\mathrm{d}\mathbf{x} \\ &= \mathrm{vol}(\mathcal{D}^c)\;\left(w_q\,\|\mathbf{J}_c + {\mathbf{J}_c}^{\mathrm{T}}\|_F^2 + w_r\;\mathrm{Tr}(\mathbf{J}_c)^2\right) = {\mathbf{v}_c}^{\mathrm{T}}\,\mathbf{E}_\phi^c\;\mathbf{v}_c\;. \end{aligned}$$

Here, the $d(d+1)$-dimensional vector $\mathbf{v}_c$ is the concatenation of velocities of the vertices of c, $\mathbf{J}_c$ is the constant Jacobian on c, and $\mathrm{vol}(\mathcal{D}^c)$ is the volume of the cell, e.g., triangle area or tetrahedral volume, which weights the constant expressions during integration over the discrete domain $\mathcal{M}$. Explicit expressions for $\mathbf{E}_\phi^c$ are derived in the following section.

The local quadratic forms $\mathbf{E}_\phi^c$ on individual cells can then be assembled into the global quadratic form $\mathbf{E}_\phi$ using an element-wise setup that is similar to the FEM-related stiffness matrix assembly:

$$\mathbf{E}_\phi = \mathbf{P}^\mathrm{T} \left(\sum_{c \in \mathcal{T}} \mathbf{E}_\phi^c \otimes \mathbf{\Lambda}^{c,c}_{|\mathcal{T}|,|\mathcal{T}|} \right) \mathbf{P} \tag{19}$$

Here, $\mathbf{P}$ is an appropriate permutation / replication matrix that selects all $\mathbf{v}_c$ associated with a particular cell, $\otimes$ denotes the Kronecker product, and $\mathbf{\Lambda}^{c,c}_{|\mathcal{T}|,|\mathcal{T}|}$ is a $|\mathcal{T}|$-dimensional square single-entry matrix that is one at (c, c) and vanishes everywhere else. Therefore, $\mathbf{E}_\phi$ is a permuted block diagonal matrix of quadratic forms in which every quadratic form measures the energy contribution of each cell. In general, $\mathbf{E}_\phi$ is highly sparse.

We use interpolation constraints on the flow $\mathbf{v}$. This means that the user provides guidance curves $\gamma_i(t)$ that define the continuous paths of some constrained vertices $i \in \mathcal{V}$. This yields conditions $\mathbf{v}_i(t) = \frac{\mathrm{d}}{\mathrm{d}t}\gamma_i(t)$ as the flow along the curves is defined by their tangents. Not that this includes the special case of "fixed" vertices for which the trajectory is a constant domain point with $\mathbf{v}_i(t) \equiv \mathbf{0}$. Different simple types of user-interaction techniques for the guidance curve specification will be presented in Section 3.5.1.

As $E_\phi(\mathbf{v})$ is quadratic in $\mathbf{v}$, the optimal vector field

$$\tilde{\mathbf{v}} = \operatorname*{argmin}_{\mathbf{v}} E_\phi(\mathbf{v}) \tag{20}$$

that minimizes the generalized energy is efficient to compute as the variational critical point of the energy $\nabla E_\phi(\mathbf{v}) \stackrel{!}{=} \mathbf{0}$, subject to these interpolation constraints, which is a linear problem.

Note that Equations (19) and (20) describe the quadratic form assembly and optimization only conceptually. In Section 3.5 we will discuss how to setup $\mathbf{E}_\phi$ and solve the arising linear systems efficiently in practice.

3.4.2 Energy Expressions

The implementation of the energy minimization on $\mathcal{M}$ requires explicit expressions for $\mathbf{E}_\phi^c$. A straightforward method to obtain terms that are valid for any dimension d is to use vectorizations of the Jacobians. We omit explicit cell indices c in this section because the derivation also holds for the continuous pointwise case. Let $\mathbf{j} = \mathrm{vec}(\mathbf{J})$ be the column-wise vectorization of the Jacobian $\mathbf{J}$, e.g., in the $2d$ case $\left(u_x, v_x, u_y, v_y\right)^\mathrm{T} = \mathrm{vec}(\mathbf{J})$. Note that vectorization is a linear operation such that there is an appropriately permuting linear gradient operator on $\mathcal{D}$ with $\mathbf{j} = \mathbf{G}(\mathbf{v})$. For notational convenience, in the following we will therefore restrict our derivation to $\mathbf{j}$, which is directly related to the unknown vector

field by local gradient operators $\mathbf{G}$. Additionally, we require the d^2-dimensional square commutation matrices $\mathbf{K}_d$ that are defined by $\mathbf{K}_d \operatorname{vec}(\mathbf{A}) = \operatorname{vec}(\mathbf{A}^\mathsf{T})$ for any square d-dimensional matrix $\mathbf{A}$ (see, e.g., the excellent book by Magnus and Heudecker [MN07] for more applications and related matrix calculus concepts). For example, in $2d$ and $3d$

$$\mathbf{K}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{K}_3 = \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \end{pmatrix}.$$

Note that commutation matrices are symmetric $\mathbf{K}_d = \mathbf{K}_d{}^\mathsf{T}$ (and are, in fact, involutory $\mathbf{K}_d = \mathbf{K}_d^{-1}$). As $\mathbf{v}^\mathsf{T}\mathbf{E}_\phi\,\mathbf{v} = \mathbf{v}^\mathsf{T}\mathbf{G}^\mathsf{T}\mathbf{M}_\phi\,\mathbf{G}\,\mathbf{v} = \mathbf{j}^\mathsf{T}\mathbf{M}_\phi\,\mathbf{j}$, we continue to derive expressions for the quadratic forms $\mathbf{M}_\phi$ that describe the local deformation properties. We focus on the anisotropic energies that have isotropic energies as a special case. For the anisotropic factors q and r of (18) we obtain

$$\begin{aligned} q &= \|\mathbf{J}^\mathsf{T} + \mathbf{J}\|_\mathbf{B}^2 \\ &= \operatorname{Tr}\Big(\left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right)^\mathsf{T} \mathbf{B} \left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right) \Big) \\ &= \operatorname{vec}\left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right)^\mathsf{T} (\mathbf{I}_d \otimes \mathbf{B})\ \operatorname{vec}\left(\mathbf{J}^\mathsf{T} + \mathbf{J}\right) \\ &= (\mathbf{j} + \mathbf{K}_d\,\mathbf{j})^\mathsf{T} (\mathbf{I}_d \otimes \mathbf{B})\ (\mathbf{j} + \mathbf{K}_d\,\mathbf{j}) \\ &= \mathbf{j}^\mathsf{T} \left((\mathbf{I}_d \otimes \mathbf{B}) + (\mathbf{I}_d \otimes \mathbf{B})\ \mathbf{K}_d + \mathbf{K}_d\,(\mathbf{I}_d \otimes \mathbf{B}) + \mathbf{K}_d\,(\mathbf{I}_d \otimes \mathbf{B})\ \mathbf{K}_d \right) \mathbf{j} \\ &= \mathbf{j}^\mathsf{T} \underbrace{(\mathbf{I}_{d^2} + \mathbf{K}_d)^\mathsf{T} (\mathbf{I}_d \otimes \mathbf{B})\ (\mathbf{I}_{d^2} + \mathbf{K}_d)}_{\mathbf{Q}} \mathbf{j} \end{aligned}$$

and

$$\begin{aligned} r &= \operatorname{Tr}(\mathbf{B}\,\mathbf{J})^2 = \operatorname{Tr}\left(\mathbf{B}^\mathsf{T}\mathbf{J}\right)^2 \\ &= \left(\operatorname{vec}(\mathbf{B})^\mathsf{T}\ \operatorname{vec}(\mathbf{J})\ \right)^2 = \left(\mathbf{b}^\mathsf{T}\,\mathbf{j}\right)^2 \\ &= \mathbf{j}^\mathsf{T} \underbrace{\mathbf{b}\,\mathbf{b}^\mathsf{T}}_{\mathbf{R}}\ \mathbf{j} \end{aligned}$$

using the symmetry of $\mathbf{K}_d$ and $\mathbf{B}$, $\mathbf{b} = \operatorname{vec}(\mathbf{B})$, and the fundamental identity $\operatorname{Tr}(\mathbf{X}^\mathsf{T}\mathbf{Y}\,\mathbf{Z}) = \operatorname{vec}(\mathbf{X})^\mathsf{T}(\mathbf{I} \otimes \mathbf{Y})\operatorname{vec}(\mathbf{Z})$ that relates matrix traces to vectorizations using the Kronecker product $\otimes$. Finally, due to linearity, we have the fundamental anisotropic quadratic form given as

$$\mathbf{M}_\phi = w_q(\phi)\,\mathbf{Q} + w_r(\phi)\,\mathbf{R}\,. \tag{21}$$

This expression simplifies to

$$\mathbf{M}_\phi = 2\,w_q(\phi)\,(\mathbf{I}_{d^2} + \mathbf{K}_d) + w_r(\phi)\ \operatorname{vec}(\mathbf{I}_d)\ \operatorname{vec}(\mathbf{I}_d)^\mathsf{T}$$

in the isotropic case $\mathbf{B} = \mathbf{I}_d$. Then the fundamental isotropic quadratic forms for $2d$ and $3d$ have the explicit representations

$$\mathbf{M}_\phi = \begin{pmatrix} 4\,w_q + w_r & 0 & 0 & w_r \\ 0 & 2\,w_q & 2\,w_q & 0 \\ 0 & 2\,w_q & 2\,w_q & 0 \\ w_r & 0 & 0 & 4\,w_q + w_r \end{pmatrix}$$

and

$$\mathbf{M}_\phi = \begin{pmatrix} 4\,w_q+w_r & 0 & 0 & 0 & w_r & 0 & 0 & 0 & w_r \\ 0 & 2\,w_q & 0 & 2\,w_q & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2\,w_q & 0 & 0 & 0 & 2\,w_q & 0 & 0 \\ 0 & 2\,w_q & 0 & 2\,w_q & 0 & 0 & 0 & 0 & 0 \\ w_r & 0 & 0 & 0 & 4\,w_q+w_r & 0 & 0 & 0 & w_r \\ 0 & 0 & 0 & 0 & 0 & 2\,w_q & 0 & 2\,w_q & 0 \\ 0 & 0 & 2\,w_q & 0 & 0 & 0 & 2\,w_q & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\,w_q & 0 & 2\,w_q & 0 \\ w_r & 0 & 0 & 0 & w_r & 0 & 0 & 0 & 4\,w_q+w_r \end{pmatrix}.$$

From these fundamental representations the following isotropic $2d$ special cases are obtained (with analogous results for the $3d$ case):

$$\begin{aligned} \mathbf{M}_{\phi_{\text{AKVF}}} &= \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}, & \mathbf{M}_{\phi_{\text{METR}}} &= \begin{pmatrix} 6 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 6 \end{pmatrix}, \\ \mathbf{M}_{\phi_{\text{CONF}}} &= \begin{pmatrix} 2 & 0 & 0 & -2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ -2 & 0 & 0 & 2 \end{pmatrix}, & \mathbf{M}_{\phi_{\text{AUTH}}} &= \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{22}$$

The quadratic form $\mathbf{M}_{\phi_{\text{AUTH}}}$ reveals the singular nature of the authalic energy, which has a three times higher rank deficit compared to all other energy types.

We will obtain related quadratic forms for other geometry processing approaches in the context of energy smoothing in Chapter 7 of this thesis. In fact, we will show that a variety of different modeling methods are, at their core, describable by energies that relate gradients using similar fundamental quadratic forms.

3.4.3 Equivalence of AKVFs and ASSVFs

The equivalence of approximate Killing (AKVF) and skew-symmetric (ASSVF) vector fields is now straightforward to show using the vectorization of (15):

$$e_{\text{SKEW2D}} = \|\mathbf{J}\|_F^2 - \frac{1}{2}\,\text{Tr}\left(\mathbf{S}^\mathsf{T}\mathbf{J}\right)^2$$

$$
\begin{aligned}
&= \operatorname{vec}(\mathbf{J})^{\mathrm{T}} \operatorname{vec}(\mathbf{J}) - \frac{1}{2}\Big(\operatorname{vec}(\mathbf{S})^{\mathrm{T}} \operatorname{vec}(\mathbf{J})\Big)^2 \\
&= \mathbf{j}^{\mathrm{T}}\mathbf{j} - \frac{1}{2}\mathbf{j}^{\mathrm{T}}\mathbf{s}\mathbf{s}^{\mathrm{T}}\mathbf{j} \\
&= \mathbf{j}^{\mathrm{T}} \underbrace{\left(\mathbf{I}_4 - \frac{1}{2}\mathbf{s}\mathbf{s}^{\mathrm{T}}\right)}_{\mathbf{M}_{\text{SKEW2D}}} \mathbf{j}\,,
\end{aligned}
$$

with $\mathbf{s} = \operatorname{vec}(\mathbf{S})$. Hence, the ASSVF quadratic form is given by

$$
\mathbf{M}_{\text{SKEW2D}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .
$$

Therefore, $\mathbf{M}_{\text{SKEW2D}}$ is equal to $\mathbf{M}_{\phi_{\text{AKVF}}}$ in (22) up to a constant scaling factor, which does not influence the optimal vector field that minimizes *both* energies. Note that this derivation yields the matrix identity $\mathbf{I}_4 - \mathbf{s}\mathbf{s}^{\mathrm{T}} = \mathbf{K}_2$, which relates skew-symmetric matrices to commutation matrices in $2d$, and is, to the best of our knowledge, not know in the literature. However, higher-dimensional identity versions are more complex to describe, because a higher number of skew-symmetric basis matrices are required.

3.4.4 Energy Smoothness

The derived energies do not yet enforce smoothness of the solution across the domain. This means that even though we obtain energy minimizers, the residual energy is not distributed smoothly over the shape. In particular, in practice this leads to high energy concentrations near constrained vertices, which results in deformation artifacts after integration. These artifacts are highlighted in Figure 5.

Recently, Lipman [Lip12, Figure 3] also observes this effect for non-continuous single-step as-rigid-as-possible deformations (ARAP) [IMH05, SA07] and proposes a non-linear bounded distortion mapping space to circumvent these artifacts.

This problem was also discussed by Solomon et al. [SBBG11a, Figure 3] for continuous vector field-based deformations that are based on AKVFs. Their solution consists in a post-process: they solve an *additional* linear system in form of a Poisson equation that *diffuses* the error minimizing vector field at the *domain boundary* to construct a smooth vector field at the interior. Essentially, this way the optimal energy-minimizing vector field at the interior of the domain is *disregarded*. We identify three additional problematic consequences: firstly, the extra solving step

effectively halves performance. Secondly, previously defined user-constraints at the shape interior can not be satisfied. In fact, during time-integration, this approach can leads to *significant* drift from the user-defined trajectories. And finally, we show that the total resulting deformation error increases *unnecessarily* due to disregarding large amounts of the optimal vector field (see Section 3.6).

We propose a different approach that is based on *regularization*. We define smoothness as the local quadratic *first-order energy variation*. This way local deformation errors vary smoothly and do not concentrate, e.g., only at the constrained vertices. Minimizing energy variation will then lead to smoother vector fields, which in turn results in smoother total deformations. We continue to derive our formulation of this type of energy smoothing.

Note that the aforementioned deformation artifacts appear for any ϕ of the generalized vector field energy. Therefore, we derive energy smoothness in a generic way that is not only applicable to AKVFs. We use the energy expressions derived in Section 3.4.2

For notational simplicity, we again concentrate on vectorized Jacobians $\mathbf{j}_c = \mathbf{G}_c \mathbf{v}_c$ per cell c in place of the vector field $\mathbf{v}_c$ at c itself due to the linear relation by the local gradient operator $\mathbf{G}_c$. The energy expressions derived in Section 3.4.2 yield the local generalized point-wise energy $e_c = {\mathbf{j}_c}^{\mathrm{T}} \mathbf{M}_c \mathbf{j}_c$. Note that $\mathbf{M}_c$ defines the energy properties in cell c, which depends on both, the energy parameter ϕ_c, and the local anisotropic norm $\mathbf{B}_c$. Then the first-order energy variation is given by $\nabla_{\mathbf{j}_c} e_c = 2\,\mathbf{M}_c \mathbf{j}_c$. Note that this expression is linear in $\mathbf{j}_c$ again.

The local energy of a cell depends only on its constant local Jacobian, i.e., there is energy variation only on the cell boundaries. Let $c_i, c_j \in \mathcal{T}$ be two neighboring cells $\mathcal{D}^i$, $\mathcal{D}^j$ with Jacobians $\mathbf{j}_i$, $\mathbf{j}_j$, and local energy operators $\mathbf{M}_i, \mathbf{M}_j$, respectively. Then we enforce smoothness by minimizing energy variation along common points $\mathcal{B}^{i,j} = \mathcal{D}^i \cap \mathcal{D}^j$, e.g., along a common triangle edge ($d = 2$) or tetrahedron face ($d = 3$). Hence, we regularize by minimizing smoothness energies of the form

$$
\begin{aligned}
E_S^{i,j}(\mathbf{v}) &= \int_{\mathcal{B}^{i,j}} \| \nabla_{\mathbf{j}_i} e_i - \nabla_{\mathbf{j}_j} e_j \|_F^2 \, \mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{B}^{i,j}} \| 2 \begin{bmatrix} \mathbf{M}_i & -\mathbf{M}_j \end{bmatrix} \begin{pmatrix} \mathbf{j}_i \\ \mathbf{j}_j \end{pmatrix} \|_F^2 \, \mathrm{d}\mathbf{x} \\
&= 4 \, \mathrm{vol}\left(\mathcal{B}^{i,j}\right) \begin{pmatrix} \mathbf{j}_i \\ \mathbf{j}_j \end{pmatrix}^{\mathrm{T}} \begin{bmatrix} {\mathbf{M}_i}^{\mathrm{T}} \mathbf{M}_i & -{\mathbf{M}_i}^{\mathrm{T}} \mathbf{M}_j \\ -{\mathbf{M}_j}^{\mathrm{T}} \mathbf{M}_i & {\mathbf{M}_j}^{\mathrm{T}} \mathbf{M}_j \end{bmatrix} \begin{pmatrix} \mathbf{j}_i \\ \mathbf{j}_j \end{pmatrix} \\
&= \left({\mathbf{v}_i}^{\mathrm{T}}, {\mathbf{v}_j}^{\mathrm{T}} \right) \mathbf{E}_S^{i,j} \left({\mathbf{v}_i}^{\mathrm{T}}, {\mathbf{v}_j}^{\mathrm{T}} \right)^{\mathrm{T}}
\end{aligned}
$$

Here, $\mathrm{vol}\left(\mathcal{B}^{i,j}\right)$ is the volume of the common point set, e.g., the common triangle edge length ($d = 2$) or the common tetrahedron face area ($d = 3$), that performs the integration of the constant term, and $\mathbf{v}_i$,$\mathbf{v}_j$ are the concatenated velocities

at the vertices of the neighboring cells. The $2\,d(d+1)$-dimensional quadratic form $\mathbf{E}_S^{i,j}$ measures the squared first-order energy variation between the pair of elements c_i, c_j. $E_S^{i,j}(\mathbf{v})$ is again quadratic in $\mathbf{v}$. The total discrete smoothness energy is then given by the sum over all adjacent cells

$$E_S(\mathbf{v}) = \sum_{\substack{c_i,c_j\in\mathcal{T}:\\ c_i,c_j \text{ adjacent}}} E_S^{i,j}(\mathbf{v}) \; .$$

This energy has the quadratic form $E_S = \mathbf{v}^\mathrm{T}\,\mathbf{E}_S\,\mathbf{v}$ that is assembled similar to (19) with appropriate permutations. It acts as a regularization term in a weighted total generalized energy in the deformation vector field

$$E(\mathbf{v}) = E_\phi(\mathbf{v}) + \alpha\, E_S(\mathbf{v}) \; . \tag{23}$$

Its quadratic form is $E(\mathbf{v}) = \mathbf{v}^\mathrm{T}\left(\mathbf{E}_\phi + \alpha\,\mathbf{E}_S\right)\mathbf{v} := \mathbf{v}^\mathrm{T}\,\mathbf{E}\,\mathbf{v}$. Hence, we compute smooth optimal vector fields

$$\hat{\mathbf{v}} = \operatorname*{argmin}_{\mathbf{v}} E(\mathbf{v})$$

by solving for the critical point of the smoothed energy $\nabla E(\mathbf{v}) \stackrel{!}{=} \mathbf{0}$, subject to the aforementioned user-constraints on $\hat{\mathbf{v}}$. The factor α weights between pure energy minimization ($\alpha = 0$) and energy smoothness ($\alpha > 0$). We use a factor of $\alpha = 0.1$ in all our example deformations. Using this small factor was sufficient to prevent spurious deformation artifact in all our experiment.

Figure 5 illustrates the effect of using the regularization term E_S in two and three dimensions. Note that smoothness of the vector field is preserved for handles in the interior as well as on the boundary of the domain, and that handle constraints are interpolated exactly. The shape boundary varies in the regularized cases but does not introduce higher deformation distortion near the boundary. Additionally, the regularized optimization for $\hat{\mathbf{v}}$ requires the solution of a single linear system. In contrast, the post-process of Solomon et al. [SBBG11a] is only able to interpolate handle constraints at the boundary. The boundary does not vary in their case, as they interpolate the boundary vector field in form of Dirichlet conditions on the additional Poisson optimization. We compare both smoothing techniques empirically in Section 3.6.

The proposed regularization based on energy variation minimization is not only beneficial to prevent deformation artifacts in the context of continuous deformations. In fact, a variety of other geometry processing approaches can be enhanced using a similar technique. We will study energy smoothing with applications to, e.g., single-step deformations or surface parametrizations, in Chapter 7 of this thesis.

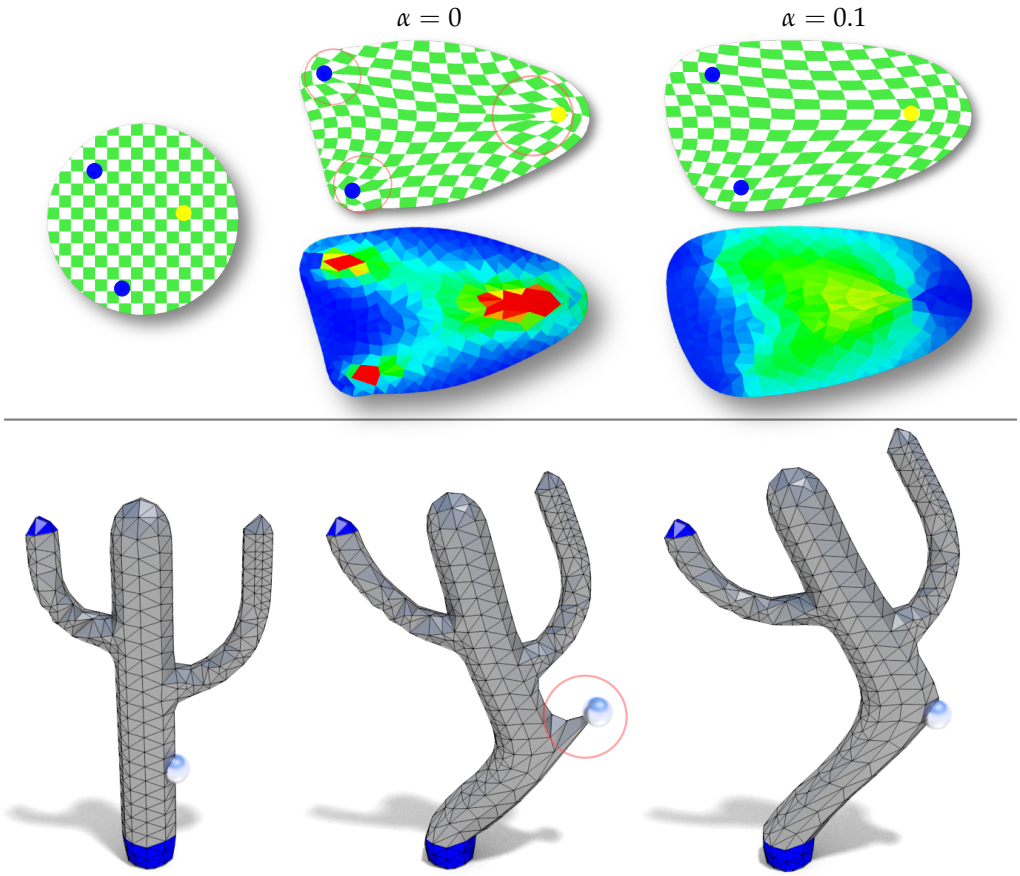
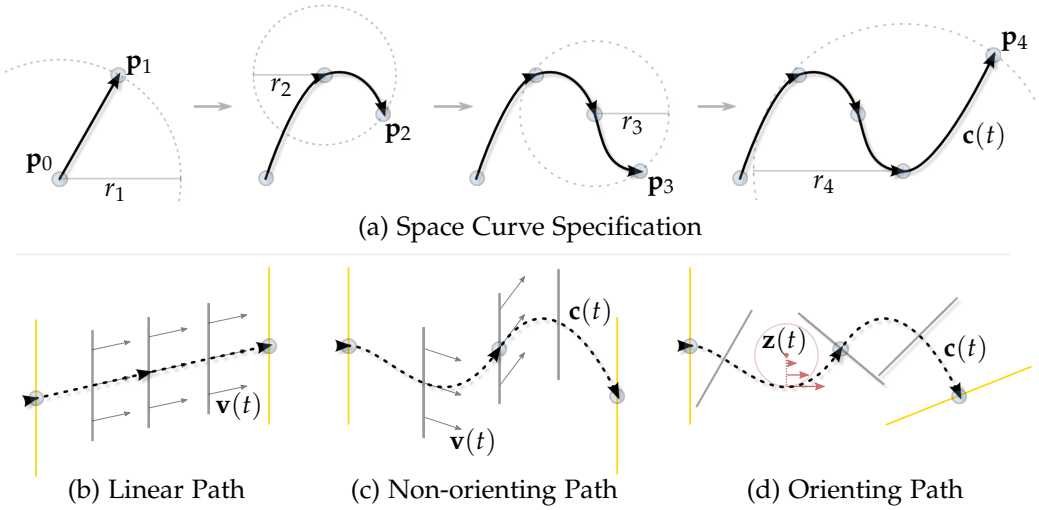


Figure 5: Energy Smoothness. In $2d$ (top) and $3d$ (bottom) minimizers of the generalized metric energy are non-smooth near user-constraints (with fixed vertices •) leading to the highlighted locally distorted deformations (colored images show metric distortions, see Section 3.6). The smoothness energy term yields smooth vector fields and therefore smooth deformations ($\alpha > 0$). Note that both interior (top) and boundary vertices (bottom) can be constrained.

3.4.5 Shape Integration

We are left with the problem of solving a $d\,|\mathcal{V}|$-dimensional ODE numerically: we solve $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_i(t) = \hat{\mathbf{v}}(\mathbf{x}_i(t), t)$ with initial vertex positions $\mathbf{x}_i(0)$, $i \in \mathcal{V}$, using a standard ODE integrator. For every evaluation of the vector field the energy minimizing flow $\hat{\mathbf{v}}$ is computed from the current shape configuration $\mathbf{x}(t)$. $\mathbf{x}(t)$ is the resulting time-parameterized continuous deformation that represents a curve in shape space to which $\hat{\mathbf{v}}$ is tangent.



(a) Space Curve Specification

(b) Linear Path (c) Non-orienting Path (d) Orienting Path

Figure 6: User Interaction Concepts. (a) Space curves $\mathbf{c}(t)$ interpolate user-selected interpolation points $\mathbf{p}_i$ on spheres centered at $\mathbf{p}_{i-1}$ (shown as $2d$ dashed circles) with user-specified varying radii r_i. (b-d) Different path types that propagate single curve constraints to handle regions (●). $\mathbf{z}(t)$ in (d) denotes the local curvature center that defines local rotation fields (●) for orientation preservation.

3.5 IMPLEMENTATION

We continue to provide implementation details of our specific implementation of the proposed generalized energy.

3.5.1 Modeling Metaphor and User Interaction

In contrast to single-step deformation methods, continuous deformations require velocities as boundary constraints (cf. [SBBG11a, KMP07]), i.e., for a subset of vertices (handles) or sets of vertices (handle regions), we prescribe the associated time-dependent vectors. This way we implement the standard handle metaphor for shape modeling: the user selects surface regions which are either fixed, deformable, or displaced by a handle. There are various ways to prescribe velocities. In the simplest case they are provided as zero vectors for fixed vertices. Translations can be modeled by constant velocities, rotations can be expressed by linear flows. A fairly general and intuitive approach is the definition of a space-time curve $\mathbf{c}(t)$ that acts as a trajectory, i.e., velocity along the curve is defined as the tangent vector $\dot{\mathbf{c}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{c}(t)$. The tangents of these curves determine the movement of handle vertices.

From the user's point of view, fixed and handle regions are selected. There is no restriction on the number of such regions or their connectivity. Then arbitrary parametric guidance curves for moving the handles are prescribed.

In particular, we adapt a modeling technique called `ZSpheres` of the modeling package `ZBrush` [9] for guidance curve specification: a sphere is centered on the initial curve point that is usually given by the handle vertex or the barycenter of a handle region. By clicking on the sphere the user selects the next $3d$ interpolation point of the space curve, and a new sphere is centered at this point. By not releasing but holding the mouse button the user can vary the radius of this new sphere. The next interpolation point is selected on this sphere. This way it is simple to specify even complex spaced curves very quickly. Figure 6 (a) illustrates this interaction concept in $2d$. We use cubic C^2-continuous B-Spline curves with centripetal parametrization and natural end conditions for point interpolation and curve reconstruction.

When a single guidance curve is assigned to a larger handle region the user is able to choose from different possible types for the propagation of curve tangent vectors to vector field constraints using "bundles" of trajectories. For simple linear and non-orienting paths the tangent vector is simply duplicated for all vertices of the region, see Figures 6 (b,c). For orientation-preserving paths local rotations around the curve curvature center

$$\mathbf{z}(t) = \mathbf{c} + \frac{\ddot{\mathbf{c}}(\dot{\mathbf{c}}^{\mathrm{T}}\dot{\mathbf{c}})^2 - \dot{\mathbf{c}}(\dot{\mathbf{c}}^{\mathrm{T}}\dot{\mathbf{c}})(\dot{\mathbf{c}}^{\mathrm{T}}\ddot{\mathbf{c}})}{(\dot{\mathbf{c}}^{\mathrm{T}}\dot{\mathbf{c}})(\ddot{\mathbf{c}}^{\mathrm{T}}\ddot{\mathbf{c}}) - (\dot{\mathbf{c}}^{\mathrm{T}}\ddot{\mathbf{c}})^2}$$

and rotation axis direction $\dot{\mathbf{c}} \times \ddot{\mathbf{c}}$ need to be performed. The C^2 curve continuity is required for continuous second curve derivatives $\ddot{\mathbf{c}}(t) = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{c}(t)$ and the corresponding spanning Frenet frames. See the local rotation field (●) in Figure 6 (d) for an example. This technique for orientation-preserving integration was already used by, e.g., von Funck et al. [vFTS07b].

Indeed, our approach is even more general in a sense that we are not restricted to parametric curves for specifying constraints: any piecewise linear time-dependent vector field can be applied to prescribe motion of handles. In particular, twisting and bending of the shape can be modeled easily.

In addition to vector field constraints, the user can model nonhomogeneous energies by changing the scalar parameter ϕ and the tensor field $\mathbf{B}$. This can be done globally or locally per cell, e.g., by a spatial blend (see Figure 12). From the users point of view, near-isometric deformations often behave similarly to real stiff materials, while near-conformal deformations often exhibit strong scaling components towards smaller and larger area.

3.5.2 GPU Implementation

We implemented the most generic energy forms, which are parameterized with per cell energy parameters ϕ and norms $\mathbf{B}$. This generic implementation is able to compute all deformation types introduced above.

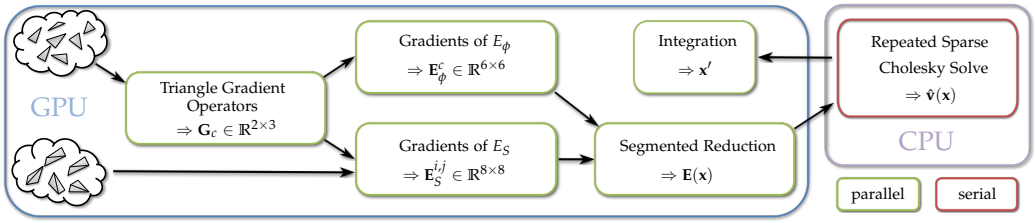


Figure 7: GPU Pipeline. We use the GPU to setup linear systems and perform vector field integration. The linear systems are solved on the CPU using an efficient sparse solver. Operations marked (●) are performed in parallel on the GPU.

Our implementation uses the GPU to accelerate certain steps of our continuous deformation algorithm. Figure 7 provides an overview, with matrix dimensions given for the $2d$ case. In short, the setup of the linear system and the integration of vertices are performed in parallel on the GPU, and the sparse system is solved on the CPU. First, all triangle gradient operators $\mathbf{G}_c$ (see Appendix A.1) are computed in parallel at the current shape configuration $\mathbf{x}(t)$. These are required to compute the energy terms E_ϕ and E_S. Then the energy gradients are computed in parallel by exploiting symmetry for each cell and for each pair of adjacent cells. The results are summed by a parallel segmented reduction operation to give the coefficients of the final linear system [5].

The sparse system is downloaded to CPU memory, where it is solved using a state-of-the-art sparse Cholesky solver that uses an approximate minimum degree preordering to reduce fill-in [CDHR08]. We use a precomputed symbolic factorization of the sparsity pattern of the linear system $\mathbf{E}$, which is filled by the numerical values computed by the GPU. In our experiments this direct system solve is up to four times faster than solving the linear system on the GPU using an iterative sparse conjugate gradient solver (see, e.g., [PTVF07]). Botsch et al. [BBK05a] obtain analogous results for pure CPU implementations of similar solvers on mesh-based energy discretizations. Compared to a pure CPU implementation incorporating the GPU for system setup results in up to three times faster execution times. This is because setup costs are significant as multiple systems need to be solved during integration and the symbolic factorization is amortized for multiple solves of systems with unchanged sparsity pattern.

Finally, shape integration along the optimal flow $\hat{\mathbf{v}}$ is performed on the GPU. The problem of numerical ODE integration is well-understood and a number of possible algorithms exits (see, e.g., [PTVF07]). Straightforward Euler integration yields visually pleasing results even for moderate time-steps. However, the lack of accuracy of this scheme would spoil our overall carefully discretized approach and render this scheme unacceptable. Therefore, we rely on higher-order schemes. Single-step ODE integrators such as Runge-Kutta integration schemes are popular, e.g., for particle tracing in flow visualization approaches (see Part II of this thesis), where vector field sampling is computationally inexpensive. Instead, for numerical integration in our setting we prefer a third-order multi-

step predictor-corrector method with adaptive step size control: in contrast to single-step schemes the predictor–corrector Adams-Bashforth-Moulton scheme takes advantage of results from previous integrations steps. The initialization is provided by a few fourth-order Runge-Kutta steps. The main motivation for choosing this integrator is the relatively fewer number of required costly vector field evaluations compared to single-step methods, if similar approximation errors are assumed. This choice was justified by experiments.

3.6 ANALYSIS AND RESULTS

We continue to provide a further analysis of properties of our approach together with deformation results in $2d$ and $3d$.

ENERGY COMPARISON. We evaluate the angle and volume quality of deformations using the following normalized error terms:

$$E^{2d}_{conf} = \frac{1}{2}\sum_{c\in\mathcal{T}_2}\rho_c(\sigma^c_1-\sigma^c_2)^2 \qquad E^{3d}_{conf} = \frac{1}{3}\sum_{c\in\mathcal{T}_3}\rho_c\sum_{(i,j)\in\mathcal{P}_3}\left(\sigma^c_i-\sigma^c_j\right)^2$$
$$E^{2d}_{auth} = \sum_{c\in\mathcal{T}_2}\rho_c(\sigma^c_1\sigma^c_2-1)^2 \qquad E^{3d}_{auth} = \sum_{c\in\mathcal{T}_3}\rho_c(\sigma^c_1\sigma^c_2\sigma^c_3-1)^2 \ .$$

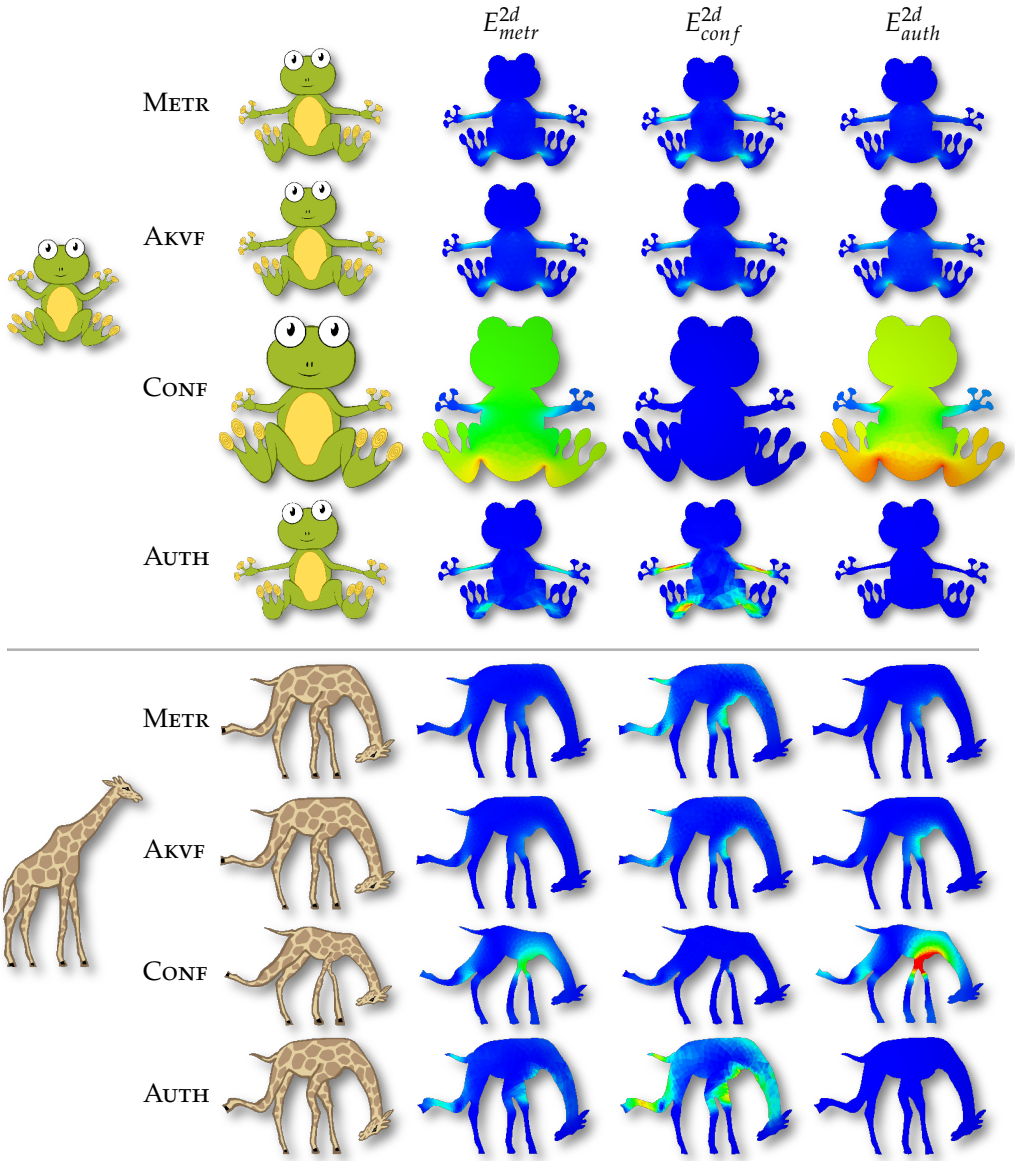
Here, σ^c_i is the i-th singular value of the deformation gradient of cell c, $\rho_c = {}^{\mathrm{vol}(\mathcal{D}^c)}/_{\mathrm{vol}(\mathcal{D})}$ is the relative cell volume (triangle area or tetrahedron volume), and $\mathcal{P}_3 = \{(1,2),(2,3),(3,1)\}$. See Appendix B for a derivation of these errors. To measure metric length-variation errors we introduce the normalized error terms

$$E^{2d}_{metr} = \sum_{c\in\mathcal{T}_2}\rho_c\left((\sigma^c_1-1)^2+(\sigma^c_2-1)^2-\frac{1}{4}(\sigma^c_1-\sigma^c_2)^2\right) \tag{24}$$

$$E^{3d}_{metr} = \sum_{c\in\mathcal{T}_3}\rho_c\left(\sum_{i=1}^{3}(\sigma^c_i-1)^2-\frac{1}{5}\sum_{(j,k)\in P_3}\left(\sigma^c_j-\sigma^c_k\right)^2\right) \tag{25}$$

that are the weighted sum of local solutions of integrals of the form of (7) and (10). In contrast to the previous derivation of vector field energies, the integrand is no point-wise infinitesimal quadratic length variation but the point-wise finite quadratic length variation induced by the deformation map, which is integrated along all possible directions [1]. Note that the metric errors (24) and (25) are weighted combinations of isometric and conformal terms (cf. Equations (81) and (82) of Appendix B). In the optimal case all error terms vanish.

Figure 8 shows error values and error visualizations for two planar deformations. For the AUTH results we used $\phi = \arctan 2^{-9}$. The METR energy generating near-isometric vector fields achieves lowest metric and area distortions at the cost of



Energy	E^{2d}_{metr}		E^{2d}_{conf}		E^{2d}_{auth}	
METR	0.0228	0.0216	0.036	0.0692	0.0155	0.0137
AKVF	0.028	0.0273	0.0211	0.0443	0.033	0.028
CONF	0.752	0.0902	0.0001	0.0098	0.978	0.273
AUTH	0.0297	0.0489	0.1102	0.1864	$5 \cdot 10^{-6}$	$4 \cdot 10^{-6}$

Figure 8: 2*d* Energy Evaluation. Two initial models (left) are deformed using the same boundary constraints for the different energy types. The plots visualize color coded local errors (low error •, high error •). The table gives total errors for each method and each model (frog left / giraffe right column).

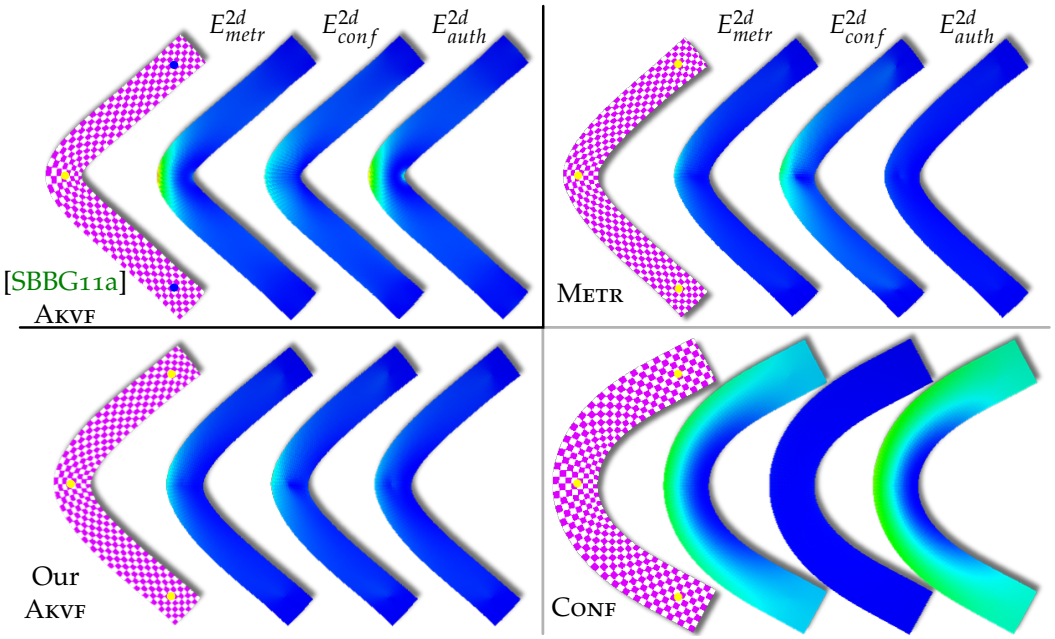
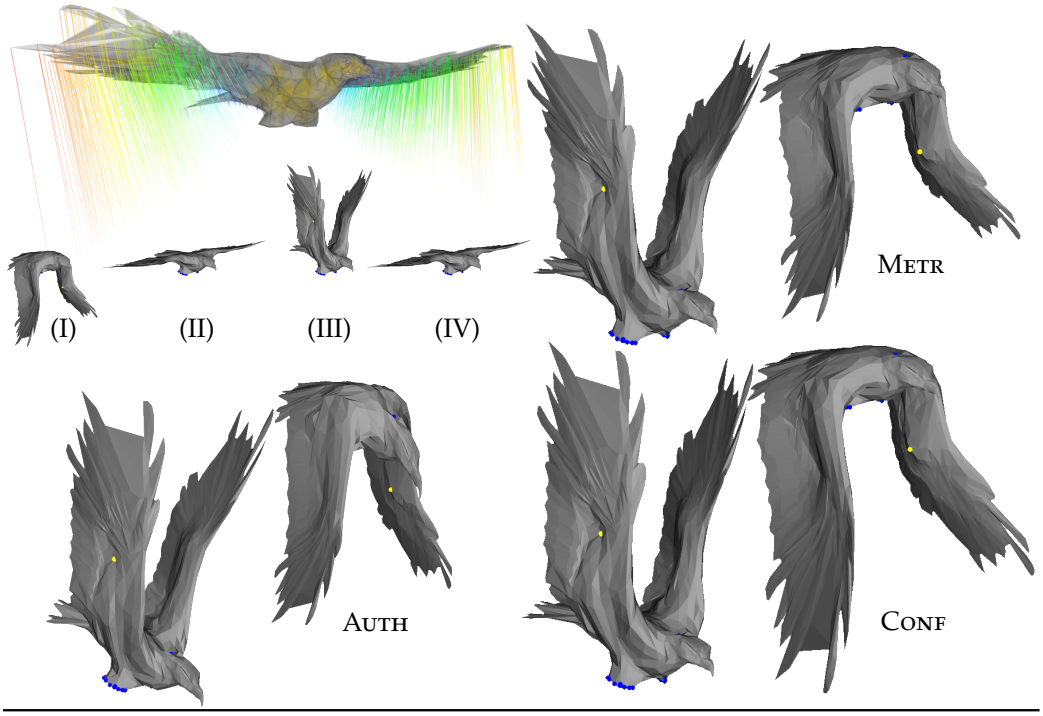


Figure 9: AKVF Comparison. A symmetric strip deformation is used to evaluate our energies and compare them to the original AKVF formulation of Solomon et al. [SBBG11a] who use no energy-based smoothing and only soft constraints. The color coded images visualize local error components (low error •, high error •), which are all scaled equally. For completeness the corresponding deformations using the metric and conformal energies are shown.

change of angle. Deformations based on the AKVF energy show better angle preservation compared to METR, but they also show greater errors in length and area variation. Almost no angle distortion is introduced by ACVFs based on the CONF energy, however, this is at the cost of area errors. The opposite is true for the EQUIA deformation that introduces almost no area error but instead a large angular error. The experiments confirm that the parameter ϕ corresponds to balance between metric and area preservation on the one side and angular preservation on the other side (cf. Figure 4). No deformation can preserve all properties at the same time.

AKVF SMOOTHING COMPARISON. In Figure 9, we compare our energies (including AKVF) to the original method of Solomon et al. [SBBG11a] that uses, to a certain extent, "soft" handle constraints and achieves smoothness by an additional diffusion step of the boundary vector field. The table in the inset gives the corresponding total error values. Note that the softly

Energy	E^{2d}_{metr}	E^{2}_{conf}	E^{2d}_{auth}
[SBBG11a] AKVF	0.084	0.075	0.080
Our AKVF	0.048	0.059	0.042
METR	0.041	0.091	0.018
CONF	0.227	0.001	0.338



(I)/(III)	METR		CONF		AKVF		AUTH	
E^{3d}_{metr}	0.061	0.069	0.172	0.072	0.083	0.091	0.386	0.211
E^{3d}_{conf}	0.356	0.421	0.022	0.027	0.305	0.357	2.357	1.367
E^{3d}_{auth}	0.009	0.007	0.273	0.051	0.028	0.012	$9 \cdot 10^{-4}$	$4 \cdot 10^{-4}$

Figure 10: 3*d* Eagle Deformation. A tetrahedral model of an eagle (top left, with instantaneous vector field) was deformed in an animation of steps (I-IV). The closeups show intermediate steps for different energies. Note the volume variation of the near-conformal deformation, and the distortions of the near-authalic deformation.

constrained vertices drifted significantly. To compensate for this effect and for a fair comparison, the constraints were selected such that the trajectories of all handles (•) end in the (optimally) fixed soft handles (•) after the same integration time. The two AKVF and the METR results look visually similar. However, even for this simple deformation all three error values indicate that our AKVF approach that uses a problem-dependent smoothing term achieves superior deformations of lower error compared to vector field diffusion [SBBG11a].

Figure 10 shows frames of the animated deformation of a volumetric mesh. The wings of the eagle model were deformed symmetrically using three-dimensional AMVFs, ACVFs, as well as with AAVFs with $\phi = \arctan 2^{-9}$. Again, the conformal energy trades volumetric error for angle preservation, while the isometric energy has better length and volume preservation properties at the expense of angular distortion. Best volume preservation but also most angular distortion is

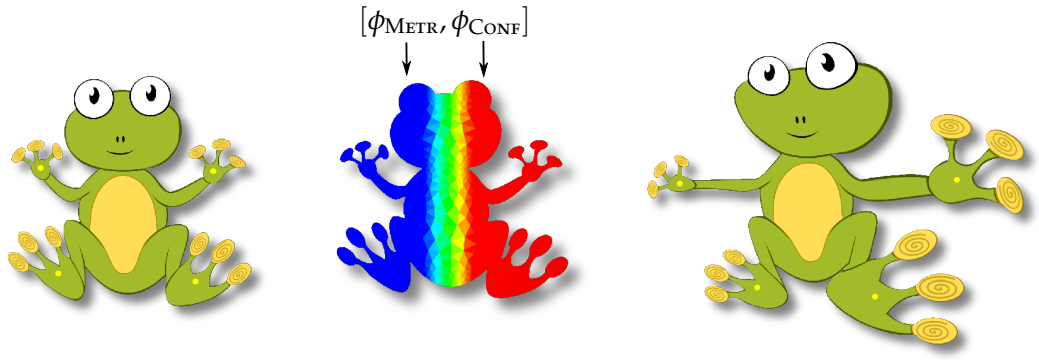


Figure 11: Nonhomogeneous Deformation. A nonhomogeneous parameter ϕ is given as a scalar field in form of a blend from ϕ_{METR} on the left side of the frog's domain to ϕ_{CONF} on the right. Deformation constraints (●) are defined symmetrically on both sides of the model.

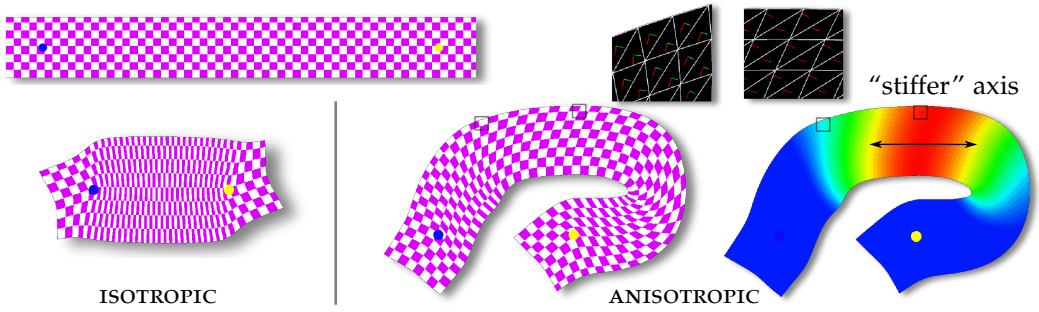


Figure 12: Anisotropic Deformation. An isotropic deformation (left) is compared to a deformation of anisotropic material with a locally "stiffer" axis direction (right). Equal constraints are used together with near-isometric ϕ_{AKVF}. Closeups show local eigenvectors of the tensor field $\mathbf{B}$ (scaled relatively by the eigenvalues).

achieved by authalic deformations. This is also reflected in the error values of the animation steps (I) and (III), which are given in the table of Figure 10. For completeness, we also provide the error values of the AKVF-based continuous deformations of the same $3d$ animation, which is not shown [2].

ENERGY PARAMETERS. Figure 11 shows an example where a nonhomogeneous parameter ϕ is prescribed as a scalar field on the domain. In the example we use a spatial blend from near-isometric (left side) to conformal (right side). Applying symmetric constraints shows the nonhomogeneous effect of ϕ in the resulting deformation.

In Figure 12 we demonstrate the effect of using an anisotropic material compared to an isotropic one. Specifically, we define a region in the center of the strip that is "stiffer" along one prescribed axis modeled by a corresponding tensor field $\mathbf{B}$.

2 Figure 52 in Chapter 7 shows the same model deformed using similar constraints for the AKVF energy.

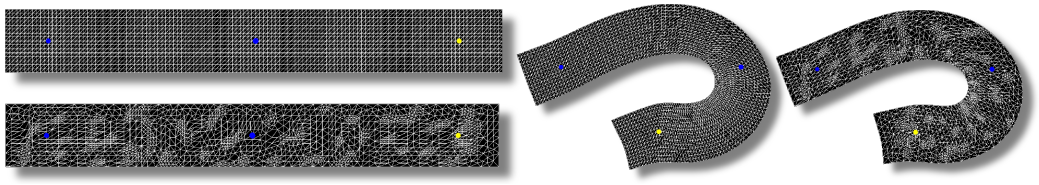


Figure 13: Tessellation Insensitivity. A differently tessellated shape (left) is deformed with the *same* boundary constraints. The deformed *shapes* are insensitive to varying domain discretizations (right).

The material modification leads to two near-isometric deformations of different characteristics for the same boundary constraints. In the isotropic case the whole shape is contracted symmetrically, whereas an asymmetric deformation results from the anisotropic energy.

TESSELLATION INSENSITIVITY. The discretization of both our vector field energies and energy smoothness terms are *integrated* quantities on the discretized domain. We therefore expect that the resulting deformations are insensitive of the domain partition, i.e., of the tessellation, as long as there are enough degrees of freedom available to represent the constrained deformation. This is confirmed by all our experiments in $2d$ and $3d$. See Figure 13 for an example.

MODELING RESULTS. Figure 14 shows initial $2d$ shapes and two deformed versions using AMAP and ACAP deformations. The model size ranges from 5k to 11k vertices and the modeling time was below four minutes in every example. More $2d$ examples are shown in Figures 3a, 3b, and 8. Besides the animation in Figure 10, we show further tetrahedral deformations in Figure 15. Again we have the initial shapes together with AMAP and ACAP deformations. The meshes contain between $1,500$ to 5000 vertices, and are deformed interactively. Again modeling time of an inexperienced user ranges from a few seconds to a few minutes.

None of our tests suffered from stability issues, not even for extreme deformations. In particular, we didn't observe local folds or flips. This is due to the fact that the energy minimizing vector field generally does not vanish and the smoothing term avoids strong local vector field variation. However, we have no formal guarantee that these artifacts will never appear because we do not explicitly avoid them. Artifact avoidance can be guaranteed by nonlinear optimization in suitable spaces at the cost of a more expensive optimization. For example, the bounded-distortion mapping spaces by Lipman [Lip12] guarantee deformation bijectivity (for triangles only) by using a nonlinear constrained conic optimization. Compared to our regularization-based approach, which only requires efficient linear optimization of quadratic energies, this limits the effective mesh sizes below a few hundred vertices. In Chapter 7, we will provide a more

Figure 14: 2*d* Deformation Examples. The triangulated models in the box were deformed using near-isometric AMVFs and near-conformal ACVFs.

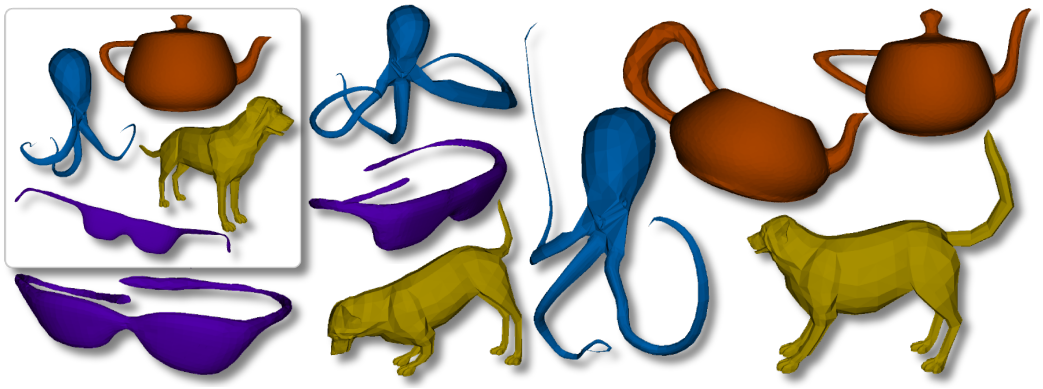


Figure 15: 3*d* Deformation Examples. The tetrahedral models in the box were deformed using near-isometric AMVFs and near-conformal ACVFs.

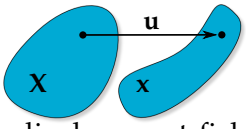
detailed analysis of our regularization approach and an explicit comparison to Lipman's method.

TIMINGS. The following table lists timings of our approach for deformations of the smallest and largest models in 2*d* and 3*d*, respectively. We measured the time for the initial symbolic factorization of the linear system (t_1), system setup time on the GPU (t_2), time to solve the system by the CPU (t_3), and the total time T to perform ten consecutive integration steps. Ten integration steps are sufficient to obtain, e.g., the TOUCAN deformations in Figure 14. Compared to a sole CPU implementation, our parallel system setup using the GPU is up

Model ($\lvert\mathcal{T}\rvert, \lvert\mathcal{V}\rvert$)	$t_1(ms)$	$t_2(ms)$	$t_3(ms)$	$T(s)$
TOUCAN (5.6k, 9.6k)	230	6	40	1.9
CAT (11k, 18k)	642	13	68	3.8
OCTOPUS (1.5k, 5k)	168	19	51	2.9
TEAPOT (5k, 16k)	424	31	118	6.3

to three times faster, even though the system has to be transferred to the CPU before solving it. Timings were measured on an AMD Phenom II 955 quad-core CPU with 3.2GHz clock speed equipped with a NVIDIA GTX 560 Ti GPU with 2 GB of memory running Linux. Our approach is interactive for reasonably sized models. However, as is true for most solvers of nonlinear measures, it also has much higher computational costs compared to linear methods. Please also see the accompanying video of [MRT13a] for additional modeling results [3].

3.7 RELATION TO LINEAR ELASTICITY



Our geometrically-motivated energy formulation can be related to physically-based theory of linear elasticity (see, e.g., [Bra07, MZS*11]). This formalism assumes that a rest configuration with material coordinates $\mathbf{X}$ is deformed by a displacement field $\mathbf{u} = \mathbf{x} - \mathbf{X}$ into a deformed shape $\mathbf{x}$. The deformation results in an isotropic internal potential deformation energy $\psi = \mu\|\boldsymbol{\epsilon}\|_F^2 + \frac{\lambda}{2}\operatorname{Tr}(\boldsymbol{\epsilon})^2$ that depends on the local strain tensor $\boldsymbol{\epsilon}$, which is usually defined using the deformation gradient tensor $\mathbf{F} = \nabla_{\mathbf{X}}\mathbf{x} = \mathbf{H} + \mathbf{I}$ with displacement gradient tensor $\mathbf{H} = \nabla_{\mathbf{X}}\mathbf{u}$. Here, μ and λ are the physical Lamé material constants, which are related to stiffness and volume preservation, respectively. For small displacement gradients the Lagrangian finite strain tensor $\boldsymbol{\epsilon}$ can be approximated by the linearized small strain tensor

$$\boldsymbol{\epsilon} := \frac{1}{2}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{I}\right) = \frac{1}{2}\left(\mathbf{H} + \mathbf{H}^{\mathrm{T}} + \mathbf{H}^{\mathrm{T}}\mathbf{H}\right) \overset{\|\mathbf{H}\|_F \ll 1}{\approx} \frac{1}{2}\left(\mathbf{H} + \mathbf{H}^{\mathrm{T}}\right) ,$$

and the linear elasticity energy becomes

$$\psi = \frac{\mu}{4}\|\mathbf{H} + \mathbf{H}^{\mathrm{T}}\|_F^2 + \frac{\lambda}{2}\operatorname{Tr}(\mathbf{H})^2 .$$

It measures the potential energy of the deformed shape $\mathbf{x}$ relative to the rest configuration $\mathbf{X}$, which is different to our instantaneous deformation energies that does not use the notion of a distant rest pose. Still, in the limit of instantaneous deformations, i.e., $\mathbf{x} \to \mathbf{X}$, we have $\mathbf{H} \to \mathbf{J}^{\mathrm{T}}$, i.e., the displacement gradient becomes the (transposed) vector field Jacobian. Both the physical linear elasticity

3 The video is located in the additional material folder `addmaterial/contdef`.

model and our geometrically motivated energy formulation therefore coincide in this case with the relation of parameters $w_q = \mu/4$ and $w_r = \lambda/2$. However, as we do not need to consider deformed shapes in different coordinate systems, our instantaneous approach doesn't require additional regularization methods like corotational elasticity [MDM*02] to correct artifacts of diverging coordinate systems $\mathbf{X}$ and $\mathbf{x}$ that stem from energy linearization. Additionally, our instantaneous approach is unconditionally stable and we can therefore apply standard explicit ODE solvers for integration and require no, e.g., implicit integration. Moreover, this derivation shows that, e.g., AKAP deformations [SBBG11a] can be regarded as a geometric instantaneous special case of physically-based linear elasticity that describes near-isometric materials. In addition, in this work we provide the parameters for materials that show, e.g., near-conformal behavior, which might not always give physically plausible results.

3.8 DISCUSSION

Most existent geometrically-motivated approaches either optimize for near–isometric [IMH05, SA07, KMP07, BWG09, SBBG11a] or for near–conformal deformations [WBGH11, WMZ12]. In contrast to this, our generalized energy formulation combines both extrema in an integral formulation, and it can be applied the same way in any dimension and especially in $2d$ and in $3d$.

We note that it is certainly possible to directly obtain near-conformal deformations in a single-step deformation by optimizing for deformation gradients that are close to similarities. However, this is not possible for, e.g., near-isometric single-step deformations, as fitting of deformation gradients to closest rotations is a nonlinear problem that usually requires iterative solvers (see, e.g., Chapter 7). Both deformation types are supported in a uniform way by our continuous approach.

In our setting of continuous deformations, all vector field energies that describe near-conformal, near-authalic, and even near-isometric deformations are quadratic, and variational optimizations become simple linear problems. The nonlinearity of the deformation is then provided by solutions of nonlinear ODEs. As ODE integration is a well-understood problem, it tends to become simpler to compute deformations that minimize nonlinear deformation energies, e.g., deviations from perfect isometry. In particular, no specialized optimization algorithms like local-global optimization schemes [SA07, LZX*08], which only converge to local minima, are required. In fact, for a given set of constrained vertex curves $\gamma_i(t)$, our continuous deformations are always globally optimal, because all involved optimizations are convex and vector fields are guaranteed to be globally optimal w.r.t. the boundary constraints. Additionally, this is the reason why our continuous deformations require no convergence criteria [BWG09]. Still, both continuous deformations and nonlinear single-step deformations perform

linearizations of the nonlinear problems, only at different points of the deformation: the optimization for vector fields of our continuous formulation is a linear problem that can be related to the linear optimization for a descent direction of a generic optimization algorithms like Gauss-Newton iterations [FB11]. Both optimization strategies have their merits, but they are also hard to compare directly, as vector field-based methods yield a continuously parameterized family of pure plastic deformations, whereas single-step methods compute a single minimizer of a pure elastic deformation energy.

The results in Solomon et al. [SBBG11a] indicate that their AKVF-based approach yields deformations of superior quality compared to related linear and nonlinear methods. Our approach is not only able to reproduce their results but it shows even better behavior, getting even closer to isometric maps. At the same time our method is less complex and more efficient as we achieve smoothness by a regularization, enable true interpolation constraints, and use ODE solvers that are not restricted to the $2d$ case only. We refer to their work for an error evaluation of planar continuous deformations to other single-step nonlinear planar deformation approaches, i.e., Cauchy-Green coordinates [WBCG09], moving least-squares coordinates [SMW06], and ARAP deformations [LZX*08]. They demonstrate that their continuous AKAP deformations show superior results compared to all other tested planar deformation methods. In Chapter 4, we provide an additional evaluation of surface-based near-isometric continuous deformations to other surface deformation approaches.

Our main feature, however, is the ability to control local deformation types by the single parameter ϕ. We obtain deformations that range from conformal to authalic with AKAP and our model of near-isometric deformations in between. Additionally, we support anisotropic energies for all types of deformations. To the best of our knowledge, this approach is the first geometrically-motivated method that provides such range of vector field-based deformations in a single and concise mathematical framework. Our METR energies are an alternative way to measure deviation from isometry. In a direct comparison to near-isometric AKVF deformations, these new near-isometric energies show a better area preservation at the expense of a slightly higher angle deviation. It is up to the user to select one these near-isometric energy types for a concrete deformation problem. Note that near-authalic deformations have not been studied thoroughly in the literature. Even though it is well known that these maps are not uniquely defined, for $\phi \to 0$ we get close to this limit and obtain meaningful results. However, we also point out that the numerical conditioning of the linear system degrades slightly for this limit. This is not the case for all other deformation types.

It seems that for many relevant shape deformation tasks near-isometry is the desired property. This is because isometric deformations have a plausible and predictable behavior, as they approximate many common deformations of real-world objects. On the other hand, near-conformal deformations seem less intuitive, as deformations with strong volume variation are less common. Still, this

type of deformation is valuable for, e.g., tasks were texture-mapped objects are deformed, because the apparent texture distortions are minimized by the local angle-preservation.

Our continuous deformations do not only depend on the initial and final handle positions as is true for single-step deformations. In fact, the final deformation depends on the whole path of every constraint, i.e., our deformations are *path-dependent*. Depending on the particular application, this property can both be seen as a feature or an artifact. We only state this property here and refer to Chapter 4, where we discuss path-dependency in the context of surface-based continuous deformations in more detail.

LIMITATIONS AND OUTLOOK. Nonlinear methods are expensive, and our method is no exception. Although we use a parallelized GPU implementation, it is impossible to outperform linear methods in terms of computation time. This is a general drawback, and the user must decide if the additional cost is worthwhile to obtain deformations of higher quality. Still, all shown examples were modeled interactively.

So far we consider only the initial value problem for path constrained deformation. It is a more complex problem to solve a boundary value problem to find an energy minimizing path between two poses in shape space, e.g., for interpolation between poses (cf. [KMP07, CH12, HRWW12]). The application of our generalized energies to these types of problems is an interesting direction for further research.

We furthermore want to study the application of our generalized energies for parametrization applications that allow locally varying parametrizations types ranging from isometric to authalic behavior. Moreover, the eigen-spectrum of the energy might allow a multiresolution (in the parameter ϕ) segmentation of shapes.

Until now we consider only solid deformations. Hence, our proposed energies cannot yet be applied to, e.g., explicit deformations of surfaces that are embedded in $3d$ space. This is because the vector field Jacobians capture only the tangential components of the vector field. They do not measure variations normal to the surface. For the same reason approximate Killing vector fields are, until now, considered only tangentially for triangulated surfaces [SBBG11b]. To address this limitation we propose a specialized type of vector field energy that yields near-isometric continuous surface deformations in the next Chapter 4.

3.9 SUMMARY

In this chapter we introduce a new family of generalized metric vector field energies for continuous shape deformations. We obtain near-isometric and near-conformal deformations by integration of approximate isometric and approximate conformal vector fields as special cases of the general energy. Deformations can be nonhomogeneous and local anisotropic behavior is achieved by incorporating varying anisotropic energy norms. Our approach works in any dimension, and we applied it for planar deformations of triangular meshes and volumetric deformations of tetrahedral meshes. The method can easily be integrated into existing tools as it shares the common intuitive user interface where few points are fixed and few points act as handles, which can be dragged along paths in the domain by the user. For the discretization of the energy we applied a first-order smoothness criterion that is energy-aware. Our implementation uses the GPU to achieve interactivity.

4

ISOMETRIC SURFACE INTEGRATION

In the previous chapter we introduced a family of vector field energies that describes different possible deformation types of continuous deformations. The formulation holds for any dimension, but deformations are restricted to solid deformations only. This includes the deformation of planar triangle meshes and volumetric tetrahedral meshes. In practice, however, also the deformation of surfaces, i.e., two-manifolds embedded in three-space, is of special importance. Yet, surface deformations are not supported by the previously introduced energies. Therefore, in this chapter we derive a new vector field energy for $3d$ surface deformations. We restrict the derivation on the most relevant near-isometric deformations.

4.1 NEAR-ISOMETRIC SURFACE ENERGY

As before, we define the continuous deformation of a two-manifold surface $\mathcal{D} \subset \mathbb{R}^3$, $\dim(\mathcal{D}) = 2$ by vector field integration. Vector fields $\mathbf{v}(\mathbf{x})$, $\mathbf{x} \in \mathcal{D}$ are obtained from a variational energy minimization. The energy penalizes vector fields that induce (locally) non-isometric behavior.

The subsequent energy derivation is motivated by the fact that integrating surfaces along perfectly rigid vector fields yields no distortion. Obviously they won't yield a reasonable deformation either. However, *locally* rigid vector fields can be easily constructed and serve as a reference: the closer $\mathbf{v}$ is to a rigid vector field, the more isometric the deformation. We show how this concept can be extended naturally to also incorporate smoothness of the resulting vector fields.

Our model of local instantaneous rigidity that yields near-isometric surface deformations is related to common single-step as-rigid-as-possible (ARAP) surface deformations that optimize for closest rotational deformation gradients (see, e.g., [SA07] and Appendix B). However, ARAP approaches generally have to solve global nonlinear problems for energy minimization, whereas in our instantaneous setting optimization is cheaper and therefore computationally more attrac-

tive. In fact, we show that global variational vector field optimization becomes a linear problem.

4.1.1 Continuous Energy

We consider a $3d$ vector field $\mathbf{r}(\mathbf{x})$ describing a *rigid* vector field, i.e., it can be written as $\mathbf{r}(\mathbf{x}) = \mathbf{r}^t(\mathbf{x}) + \mathbf{r}^r(\mathbf{x}) \times \mathbf{x}$. Here, $\mathbf{r}^t$ and $\mathbf{r}^r$ describe the local translational part and the rotation axis, respectively. Note that even though $\mathbf{r}$ may be defined everywhere in $\mathbb{R}^3$, we evaluate it only on the surface. We define the fitting energy $\tilde{E}$ as the squared difference of $\mathbf{r}$ and $\mathbf{v}$ integrated over the surface $\mathcal{D}$:

$$\tilde{E}(\mathbf{v}, \mathbf{r}) = \int_{\mathcal{D}} \|\mathbf{v}(\mathbf{x}) - \mathbf{r}(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x} \, .$$

Given a current surface $\mathcal{D}$, our goal is to compute the closest rigid vector field $\hat{\mathbf{r}}(\mathbf{v})$ as a function of $\mathbf{v}$ by minimizing $\tilde{E}(\mathbf{v}, \mathbf{r})$ for all rigid fields $\mathbf{r}$:

$$\hat{\mathbf{r}}(\mathbf{v}) = \operatorname*{argmin}_{\mathbf{r}} \tilde{E}(\mathbf{v}, \mathbf{r}) \, . \tag{26}$$

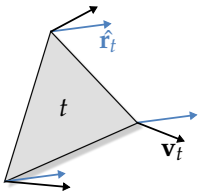
Then we obtain the total isometric vector field energy E as the deviation of $\mathbf{v}$ from $\hat{\mathbf{r}}$:

$$E(\mathbf{v}) = \tilde{E}(\mathbf{v}, \hat{\mathbf{r}}) = \int_{\mathcal{D}} \|\mathbf{v}(\mathbf{x}) - \hat{\mathbf{r}}(\mathbf{v}, \mathbf{x})\|^2 \, \mathrm{d}\mathbf{x} \, . \tag{27}$$

This energy models a measure for the isometric distortion of $\mathcal{D}$ under instantaneous motion along $\mathbf{v}$. With (27) it is evident that $\mathbf{v}$ is a Killing vector field [Efi57, KMP07, SBBG11a] on $\mathcal{D}$ iff $E(\mathbf{v}) = 0$, because in this case the best fitting rigid field $\hat{\mathbf{r}}$ will be identical to $\mathbf{v}$.

4.1.2 Energy Discretization

We discretize the surface $\mathcal{D}$ by triangular meshes $\mathcal{M} = (\mathcal{V}, \mathcal{T})$ with vertices $\mathcal{V}$, and triangles $\mathcal{T}$. Let $\mathcal{E}_i$ denote the set of internal edges. The embedding of the surface in three-space is defined by vertex positions $\mathbf{x}_i \in \mathbb{R}^3$, $i \in \mathcal{V}$. Furthermore, the piecewise linear vector field $\mathbf{v} = \left({\mathbf{v}_1}^{\mathrm{T}}, \ldots, {\mathbf{v}_{|\mathcal{V}|}}^{\mathrm{T}}\right)^{\mathrm{T}}$ is defined by vectors $\mathbf{v}_i \in \mathbb{R}^3$ at each vertex.



Reviewing the situation for a single triangle $t \in \mathcal{T}$ is sufficient to explain the energy discretization. The total energy is obtained by contributions of all triangles. We consider a triangle $t = (1, 2, 3)$ with vertex coordinates $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and associated vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ that define the linear coordinate function and vector field $\mathbf{v}_t(\mathbf{x})$ on t. We associate rigid fields $\mathbf{r}_t(\mathbf{x})$ to each triangle with local

translation and rotation components. Then the fitting energy is defined on each triangle with points $\mathcal{D}_t$ by

$$\tilde{E}_t(\mathbf{v}_t, \mathbf{r}_t) = \int_{\mathcal{D}_t} \|\mathbf{v}_t(\mathbf{x}) - \mathbf{r}_t(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x}, \tag{28}$$

and by evaluation of the integral we obtain the closed-form expression

$$\tilde{E}_t(\mathbf{v}_t, \mathbf{r}_t) = \frac{A_t}{3} \sum_{(i,j)\in\mathcal{P}_3} \|\mathbf{v}_{ij} - \mathbf{r}_t(\mathbf{x}_{ij})\|^2 \, .$$

with triangle area A_t, $\mathbf{x}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$, $\mathbf{v}_{ij} = \frac{1}{2}(\mathbf{v}_i + \mathbf{v}_j)$, and $\mathcal{P}_3 = \{(1,2), (2,3), (3,1)\}$ corresponds to all triangle edges. Hence, the local minimization problem (26) is a linear least-squares problem in the six coefficients of $\mathbf{r}_t$ that define the optimal $\hat{\mathbf{r}}_t$ ($\bullet$ in the inset). It is important to note that these coefficients depend in a nonlinear way on the current triangle coordinates, but are linear in the target vector field $\mathbf{v}_t$. We provide more details on this local optimization problem in Section 4.2.2. Analogously to the continuous case, we obtain the total triangle-based isometric vector field energy E_t by the deviation of the local vector field to the closest rigid vector field

$$E_t(\mathbf{v}_t) = \tilde{E}_t(\mathbf{v}_t, \hat{\mathbf{r}}_t) = \int_{\mathcal{D}_t} \|\mathbf{v}_t(\mathbf{x}) - \hat{\mathbf{r}}_t(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x} \, . \tag{29}$$

Interestingly, E_t is quadratic in the unknown vector field $\mathbf{v}_t$ on t because $\hat{\mathbf{r}}_t$ depends linearly on $\mathbf{v}_t$. The total discretized isometric vector field energy E is then given by the contributions of all triangles as

$$E(\mathbf{v}) = \sum_{(i,j,k)\in\mathcal{T}} E_t([\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]) \, .$$

4.1.3 Energy Properties

The energy $E(\mathbf{v})$ is quadratic in $\mathbf{v}$. Hence, there is a (sparse) quadratic form $\mathbf{E}$ such that $E(\mathbf{v}) = \mathbf{v}^\mathrm{T} \mathbf{E} \mathbf{v}$. This property will allow an efficient optimization that is discussed in Section 4.2. Note also that this is a fundamental difference to common ARAP deformations, where analogous energies in the deformed coordinates are nonlinear (see, e.g., Equation (80) for isometric deformation gradients).

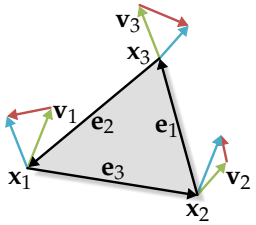
The energy $E(\mathbf{v})$ is also invariant under adding a rigid field to $\mathbf{v}$: let $\hat{\mathbf{r}}$ be the best fitting rigid field to $\mathbf{v}$, and let $\mathbf{p}(\mathbf{x})$ be another arbitrary rigid vector field. Then it is straightforward to show that the best fitting rigid field to the modified target vector field $\mathbf{v}'(\mathbf{x}) = \mathbf{v}(\mathbf{x}) + \mathbf{p}(\mathbf{x})$ is $\hat{\mathbf{r}} + \mathbf{p}$. Furthermore, $E(\mathbf{v}) = E(\mathbf{v}')$.

We emphasize that by construction $E(\mathbf{v})$ measures isometric distortion: using our energy exact isometric deformations of developable surfaces (see Figure 20 (a))

are obtained and indeed yield $E(\mathbf{v}) \equiv 0$ for the whole integration. In particular, this differs from other vector field energy formulations (see, e.g., the related discussion by Kilian et al. [KMP07]).

In the literature (cf. [KMP07, EPT*07]) isometric distortion of a triangle $t = (1, 2, 3)$ under integration of its vertices is usually measured by

$$\begin{aligned} \bar{E}_t(\mathbf{v}) &= h_1^2 + h_2^2 + h_3^2 \qquad \text{with} \\ h_1 &= (\mathbf{x}_3 - \mathbf{x}_2)^{\mathrm{T}} (\mathbf{v}_3 - \mathbf{v}_2) \\ h_2 &= (\mathbf{x}_1 - \mathbf{x}_3)^{\mathrm{T}} (\mathbf{v}_1 - \mathbf{v}_3) \\ h_3 &= (\mathbf{x}_2 - \mathbf{x}_1)^{\mathrm{T}} (\mathbf{v}_2 - \mathbf{v}_1) \ . \end{aligned} \tag{30}$$



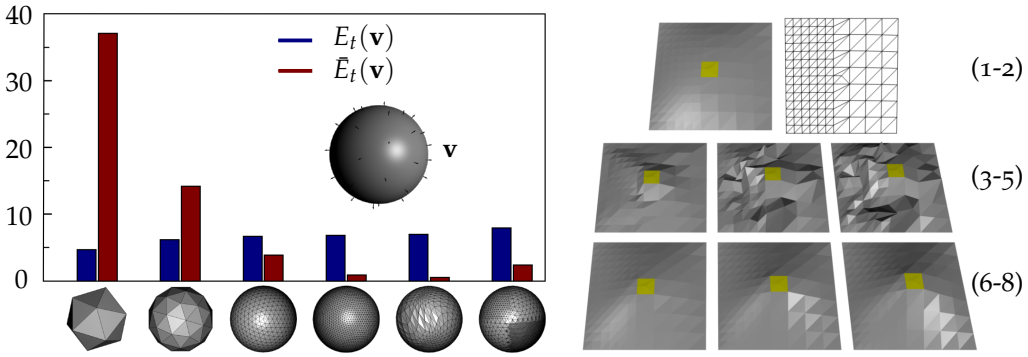
Here, h_k are projections of vector field variation (•) at each edge onto the edge vectors $\mathbf{e}_k$ (•). This way the h_k terms vanish if the vector field at an edge induces no edge length variation. In particular, the measure vanishes if the vector field is sampled from a rigid vector field. Then the summation of $\bar{E}_t$ over all triangles is the global energy to be minimized.

Our energy is related and compatible to this formulation in the sense that it can be written as the quadratic form $E_t = (h_1, h_2, h_3)\ \mathbf{M}\ (h_1, h_2, h_3)^{\mathrm{T}}$ ($\mathbf{M} = \mathbf{I}$ for $\bar{E}_t$): let $e_{ij} = \mathbf{e}_i{}^{\mathrm{T}} \mathbf{e}_j$ and $\alpha = 4\ \mathrm{vol}(\mathcal{D}_t)^2$. Then direct calculation shows that $\mathbf{M}$ is a symmetric quadratic form, which only depends on $\mathbf{x}_t$ and not on $\mathbf{v}_t$:

$$\mathbf{M} = \frac{-1}{144\,\alpha\,(e_{12} + e_{23} + e_{31})} \begin{bmatrix} 3\,e_{23}^2 + 4\,\alpha & 6\,e_{23}\,e_{31} - 4\,\alpha & 6\,e_{31}\,e_{12} - 4\,\alpha \\ 6\,e_{31}\,e_{23} - 4\,\alpha & 3\,e_{31}^2 + 4\,\alpha & 6\,e_{12}\,e_{23} - 4\,\alpha \\ 6\,e_{12}\,e_{31} - 4\,\alpha & 6\,e_{23}\,e_{12} - 4\,\alpha & 3\,e_{12}^2 + 4\,\alpha \end{bmatrix} .$$

Hence, generally no edge or area weighting-scheme exists that turns their energy $\bar{E}_t(\mathbf{v})$ into $E_t(\mathbf{v})$, since $\mathbf{M}$ is generally not diagonal. In contrast to $\bar{E}_t(\mathbf{v})$, and roughly speaking, our energy $E_t(\mathbf{v})$ also incorporate all mixed products $h_i h_j, i \neq j$. The main reason why both measures differ is the fact that we perform a careful discretization of integrated continuous point-wise energies (29), instead of only measuring edge length variations of discretized surfaces without considering the whole shape area (30).

The integration of error quantities on the whole surface is essential to the design of our energy: this way, $E_t(\mathbf{v})$ will be more insensitive to subdivision of triangles or generally of particular parametrization / tessellation. Figure 16a illustrates this property by a simple example and compares $E_t(\mathbf{v})$ to $\bar{E}_t(\mathbf{v})$: we prescribe a vector field and evaluate the energy for different tessellations of the same shape, a unit sphere. Then tessellation-insensitivity requires low *variance* of energy values: our energy $E_t(\mathbf{v})$ shows much lower variance compared to $\bar{E}_t(\mathbf{v})$. We emphasize that geometrically the absolute values are meaningless for this experiment. However, they can physically be interpreted to be the applied membrane strain since isometric deformations are a geometric approximation of real-world, thin surfaces deforming with a very high Young's modulus [NMK*06]. Further



(a) Tessellation Dependence (b) Energy Stability

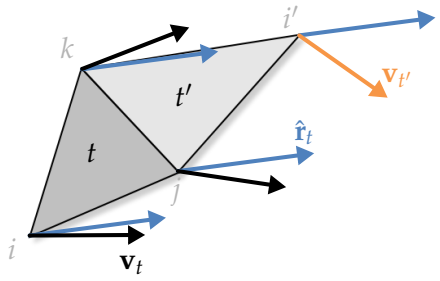
Figure 16: Energy Comparison. (a) Different unit sphere tessellations and values of our energy $E_t(\mathbf{v})$ and $\bar{E}_t(\mathbf{v})$ for the shown normal vector field $\mathbf{v}$. Of importance is the energy *variance* for different tessellations of the same shape, which should be low. (b) The irregularly tessellated test surface (1-2) is Euler-integrated three steps by minimizing $\bar{E}_t(\mathbf{v})$ (3-5), and by minimizing our $E_t(\mathbf{v})$ (6-8).

experiments and a comparison are shown in Figures 20 and 16b (see also Section 4.5). Tessellation-insensitivity is generally an important requirement for many algorithms, and it is also essential for meaningful continuous deformations because coherence for time–dependent deformations is improved.

4.1.4 Smoothness Energy

An energy that is based solely on the preservation of isometry is obviously not sufficient to determine meaningful surface deformation: for instance, folding yields perfect isometric deformations, whereas for shape deformation they are considered unwanted artifacts. The energy $E(\mathbf{v})$ does not exclude, e.g., foldings of developable surfaces (see, e.g., [KFC*08, SVWG12]). Consequently, we require an additional energy term that penalizes discontinuous deformation by enforcing smoothness of the vector field $\mathbf{v}$.

A suitable measure that fits our setting should be derived from existing quantities. We take advantage of the fact that the best fitting rigid vector fields $\hat{\mathbf{r}}$ are defined not only on respective triangles but everywhere in $\mathbb{R}^3$. In particular, we can evaluate $\hat{\mathbf{r}}$ for a certain triangle t on an adjacent triangle t'.



Let triangles $t = (i, j, k)$ and $t' = (k, j, i')$ be adjacent with vertex coordinates $\mathbf{x}_\ell$ and associated vectors $\mathbf{v}_\ell$, $\ell \in \{i, j, k, i'\}$. Furthermore, let $\hat{\mathbf{r}}_t$ (•) and $\hat{\mathbf{r}}_{t'}$ be the best fitting rigid fields on t and t', respectively. Then, loosely spoken, $\hat{\mathbf{r}}_t$ and $\mathbf{v}_{t'}$ should not differ too much for a meaningful deformation. We formalize this by applying $\hat{\mathbf{r}}_t$ to t' (and $\hat{\mathbf{r}}_{t'}$ symmetrically to t, respectively) and using the distance to the local vector field as point-wise energy. Then the vector field smoothness energy at neighboring triangles is given by

$$S_{t,t'}(\mathbf{v}) = \int_{\mathcal{D}_{t'}} \|\mathbf{v}_{t'}(\mathbf{x}) - \hat{\mathbf{r}}_t(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x} \,,$$

where $\mathbf{v}_{t'}$ denotes the linear vector field on triangle t', and $\hat{\mathbf{r}}_t$ is the best fitting rigid field optimized on t. This integral can be evaluated for the equivalent closed-form

$$S_{t,t'}(\mathbf{v}) = \frac{A_{t'}}{3} \sum_{(m,n)\in\{(k,j),(j,i'),(i',k)\}} \|\mathbf{v}_{mn} - \hat{\mathbf{r}}_t(\mathbf{x}_{mn})\|^2$$

with $\mathbf{x}_{mn} = \frac{1}{2}(\mathbf{x}_m + \mathbf{x}_n)$ and $\mathbf{v}_{mn} = \frac{1}{2}(\mathbf{v}_m + \mathbf{v}_n)$. This energy is again quadratic in the unknown vector field $\mathbf{v}$. Using the energy $S_{t,t'}(\mathbf{v})$ between neighboring triangles we define the global vector field smoothness energy S on the entire surface as the sum of all pairwise contributions:

$$S(\mathbf{v}) = \sum_{\substack{t,t'\in\mathcal{T}:\\ t,t' \text{ adjacent}}} S_{t,t'}(\mathbf{v}) + S_{t',t}(\mathbf{v}) \,.$$

In general $S_{t,t'}(\mathbf{v}) \neq S_{t',t}(\mathbf{v})$, which both have to be summed in $S(\mathbf{v})$. The smoothness energy can be written as the (sparse) quadratic form $S(\mathbf{v}) = \mathbf{v}^{\mathrm{T}} \mathbf{S} \mathbf{v}$. Note that Botsch et al. [BPGK06] use a similar principle to match transformations of incident prisms. We compare results of both approaches in Section 4.3.

With the two energies $E(\mathbf{v})$ and $S(\mathbf{v})$ we finally define the total smoothed vector field energy $E^S(\mathbf{v})$ that will be minimized to compute continuous deformations:

$$E^S(\mathbf{v}) = (1 - \omega)\, E(\mathbf{v}) + \omega\, S(\mathbf{v}) \,,$$

with a small weight $\omega > 0$ (see below). This combination of distortion energy and a suitable smoothness term is the surface-based analog to the total generalized space-deformation energy (23).

We motivated the second energy $S(\mathbf{v})$ by the fact that isometry does not always convey enough information for meaningful deformations. We remark that $S(\mathbf{v})$ is required for another reason: in special cases the discretization of $E(\mathbf{v})$ yields a singular or ill-conditioned operator, and $S(\mathbf{v})$ acts as a regularization term. For instance, a planar surface constitutes such a special case, independently of the number of applied user-constrains. Of course, then it applies only to the first integration step – after that the surface is probably no longer planar. However, to ensure robustness of our approach in any possible situation we require $\omega > 0$.

Our experiments show that a rather high value in $(0, 1]$ can be chosen for ω without spoiling minimization of distortion, i.e., the effect of $E(\mathbf{v})$. The reason is that the definition of $S(\mathbf{v})$ retains essential properties of $E(\mathbf{v})$ with the difference of rigid vector field extrapolation to neighboring triangles. This energy-aware smoothness term is reminiscent to the energy-aware smoothness already introduced in Section 3.4.4. In fact, a variety of geometry processing approaches can be enhanced using a similar approach, see Chapter 7.

We close this section with two final remarks: First, the energies $E(\mathbf{v})$ and $S(\mathbf{v})$ are compatible in a sense that comparable quantities are measured, i.e., integrated squared distances of vector fields to rigid vector fields. Second, there is a bias in the weighting as summation is over $|\mathcal{T}|$ triangles for $E(\mathbf{v})$ and over $|\mathcal{E}_i|$ adjacent pairs of triangles for $S(\mathbf{v})$. From the latter relation the Euler–Poincaré characteristic applied to triangle meshes yields $\omega = 1/3$ for an even weighting. We use this value as the default parameter if not noted otherwise.

4.1.5 Shape Integration

In analogy to the solid vector field energies of Chapter 3, we optimize for the piecewise linear vector field $\hat{\mathbf{v}}$ that minimizes the energy functional $E^S(\mathbf{v})$. In contrast to the previous chapter, the energy is *surface-based* and defined on triangular meshes with vertex coordinates $\mathbf{x}$. The vector field $\hat{\mathbf{v}}$ minimizes isometric distortion under integration of $\mathbf{x}$ due to the definition of $E(\mathbf{v})$. Additionally, the contribution of $S(\mathbf{v})$ to $E^S(\mathbf{v})$ accounts for smoothness of $\hat{\mathbf{v}}$.

Our approach to near-isometric surface deformation assumes time-dependent shapes $\mathbf{x}(t)$ and vector fields $\hat{\mathbf{v}}(t)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \hat{\mathbf{v}}(t) \tag{31}$$

(cf. Equation (1)). In every time step t (or generally every point t where the vector field is evaluated) we determine $\hat{\mathbf{v}}(t)$ for $\mathbf{x}(t)$ subject to user-constraints. The associated optimization problem is again linear in the unknowns $\mathbf{v}_i$, $i \in \mathcal{V}$.

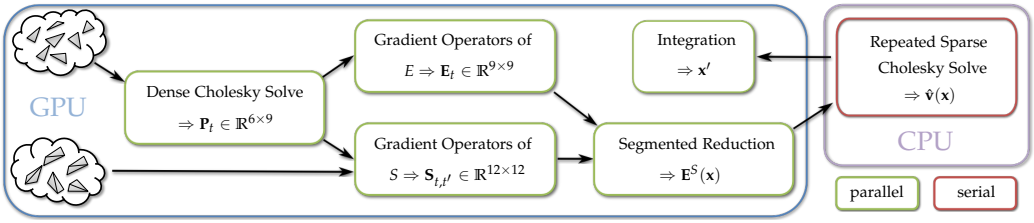


Figure 17: Computational Pipeline. For each triangle on the GPU (●) we compute the parameter mapping matrices $\mathbf{P}_t$, which are used to determine the gradient components of the distortion and smoothness energies E and S. These coefficients are combined in the global sparse quadratic form $\mathbf{E}^S$. All GPU operations are performed in parallel (●). The system is solved serially on the CPU (●) using a precomputed symbolic factorization yielding the optimal vector field $\hat{\mathbf{v}}$.

4.2 IMPLEMENTATION

In this section, we discuss the implementation of each stage of our approach in more detail. These are firstly, the specification of deformations, secondly, the GPU-based linear framework for finding vector fields that minimize our error measures in every time step, and finally, the numerical integration of the shape over time.

4.2.1 Interaction

Deformations are specified by the user in the same way as discussed in Section 3.5.1, i.e., for a subset of vertices guiding curves are specified. This includes prescribed vertex trajectories as well as fixed vertices with vanishing guidance vectors. Twisting and bending motions are modeled by appropriate linear vector fields.

4.2.2 Global Energy Minimization

Energies are quadratic forms in the unknown vector field $\mathbf{v} \in \mathbb{R}^{3|\mathcal{V}|}$. Our goal is to minimize an energy of the form

$$E^S(\mathbf{v}) = \mathbf{v}^T((1-\omega)\,\mathbf{E}(\mathbf{x}) + \omega\,\mathbf{S}(\mathbf{x}))\,\mathbf{v}\,, \tag{32}$$

where $\mathbf{E}$ and $\mathbf{S}$ are global sparse quadratic forms that implement the energies E and S, respectively. The setup of the corresponding global sparse quadratic form $\mathbf{E}^S(\mathbf{x}) = (1-\omega)\,\mathbf{E} + \omega\,\mathbf{S}$ is attended by considerable computational costs, but is inherently parallizable. To guarantee fast execution times we again opt for a combined GPU and CPU approach that is shown in Figure 17. The basic computational pipeline is similar to the implementation discussed in Section 3.5

but includes an additional per-triangle optimization for best-fitting rigid field operators.

BEST RIGID FIELDS. Both energies E and S depend on the evaluation of the best fitting rigid fields $\hat{\mathbf{r}}_t$ on each triangle $t = (i, j, k) \in \mathcal{T}$. Rigid fields are parameterized by the rotational and translational parameters $\mathbf{p}_t = \left(\mathbf{r}_t^{r\mathrm{T}}, \mathbf{r}_t^{t\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^6$. The parameters $\hat{\mathbf{p}}_t$ of the closest rigid vector field are induced by the linear vector field $\mathbf{v}_t = \left(\mathbf{v}_i{}^{\mathrm{T}}, \mathbf{v}_j{}^{\mathrm{T}}, \mathbf{v}_k{}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^9$ on t by the optimization of (26). As this optimization problem is quadratic in $\mathbf{p}_t$, there is a linear map $\mathbf{P}_t\,\mathbf{v}_t = \hat{\mathbf{p}}_t$ that relates the linear vector field to the best rigid vector field. Here, $\mathbf{P}_t \in \mathbb{R}^{6\times 9}$ are *parameter mapping operators* that are determined for each triangle independently using a local optimization: switching to matrix notation, the integrated deviation from a rigid vector field that is described by Equation (28) is expressed as

$$\tilde{E}_t = \|\mathbf{v}_t - \mathbf{R}_t\,\mathbf{p}_t\|^2_{\mathbf{N}_t}\,. \tag{33}$$

Here, $\mathbf{R}_i \in \mathbb{R}^{9\times 6}$ is an operator that evaluates the rigid vector field parameterized by $\mathbf{p}_t$ at each vertex of triangle t. The norm $\mathbf{N}_t$ is the integration operator of squared three-dimensional linear functions on triangle t. Please see Appendix A.2 for a derivation of $\mathbf{N}_t$, which corresponds to the case $\mathbf{N}_{2,3}$ in the notation of that section and is also known as the FEM mass matrix. Let A_t be the area of t, and let $\mathbf{C}_\mathbf{a}(\cdot) = \mathbf{a} \times (\cdot)$ be the 3×3 linear operator that performs a cross product with the vector $\mathbf{a}$. Then $\mathbf{N}_t$ and $\mathbf{R}_t$ are given explicitly by

$$N_{t,\alpha\beta} = \begin{cases} A_t/6 & \text{if} \quad \alpha = \beta \\ A_t/12 & \text{if} \quad \alpha - \beta \mod 3 = 0 \\ 0 & \text{else} \end{cases}, \qquad \mathbf{R}_t = \begin{bmatrix} -\mathbf{C}_{\mathbf{x}_i} & \mathbf{I}_3 \\ -\mathbf{C}_{\mathbf{x}_j} & \mathbf{I}_3 \\ -\mathbf{C}_{\mathbf{x}_k} & \mathbf{I}_3 \end{bmatrix},$$

respectively. Note that both $\mathbf{R}_t$ and $\mathbf{N}_t$ depend on the geometric configuration of t, i.e., its vertex coordinates, but not on the unknown vector field $\mathbf{v}_t$. Then (33) is optimized for $\mathbf{P}_t$ by solving the linear system corresponding to $\nabla_{\mathbf{p}_t} \tilde{E}_t \stackrel{!}{=} \mathbf{0}$, yielding

$$\mathbf{P}_t = \left(\mathbf{R}_t{}^{\mathrm{T}}\,\mathbf{N}_t\,\mathbf{R}_t\right)^{-1}\,\mathbf{R}_t{}^{\mathrm{T}}\,\mathbf{N}_t\,.$$

We perform these independent computations in a parallelized and numerically stable way for each triangle on the GPU by computing one dense Cholesky factorizations of $\mathbf{R}_t{}^{\mathrm{T}}\,\mathbf{N}_t\,\mathbf{R}_t$ and nine corresponding back-substitutions. Note that the computation of $\mathbf{P}_t$ is robust as long as the triangle is not degenerated, i.e., as long as the triangle area and the edge length ratios are bounded from below.

DISTORTION AND SMOOTHNESS GRADIENT OPERATORS. Using the parameter mapping operators we can write Equation (29) in matrix form as

$$E_t = \|(\mathbf{I}_3 - \mathbf{R}_t\,\mathbf{P}_t)\;\mathbf{v}_t\|^2_{\mathbf{N}_t}\,.$$

For the global optimization we require the gradient operator $\mathbf{E}_t \in \mathbb{R}^{9\times 9}$ that maps $\mathbf{v}_t$ to the energy gradient of E_t. It contributes to the global quadratic form $\mathbf{E}$. This operator is obtained by differentiation of E_t w.r.t. $\mathbf{v}_t$, which gives

$$\mathbf{E}_t = 2\left(\mathbf{I}_3 - \mathbf{R}_t\,\mathbf{P}_t\right)^{\mathrm{T}} \mathbf{N}_t \left(\mathbf{I}_3 - \mathbf{R}_t\,\mathbf{P}_t\right) \ .$$

The computation of $\mathbf{E}_t$ is performed in parallel for each triangle on the GPU.

Similarly, we find the gradient operators $\mathbf{S}_{t,t'} \in \mathbb{R}^{12\times 12}$ that contribute to $\mathbf{S}$ for each pair of neighboring triangles t and $t' = (k, j, i')$ by the evaluation of the gradient of

$$S_{t,t'} = \left\| \left(\mathbf{I}_3' - \mathbf{R}_{t'}\,\mathbf{P}_t\,\mathbf{Q}_t\right)\ \mathbf{v}_{t,t'} \right\|^2_{\mathbf{N}_{t'}}$$

w.r.t. $\mathbf{v}_{t,t'}$. Here, $\mathbf{I}_3' = \Big[\mathbf{I}_3, \mathbf{0}\Big]$, and $\mathbf{Q}_t \in \mathbb{R}^{9\times 12}$ is a permutation matrix that selects the vector $\mathbf{v}_t$ that corresponds to t out of $\mathbf{v}_{t,t'} = \left({\mathbf{v}_{t'}}^{\mathrm{T}}, {\mathbf{v}_t}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{12}$ in the correct order. Also computed in parallel on the GPU this yields

$$\mathbf{S}_{t,t'} = 2\left(\mathbf{I}_3' - \mathbf{R}_{t'}\,\mathbf{P}_t\,\mathbf{Q}_t\right)^{\mathrm{T}} \mathbf{N}_{t'} \left(\mathbf{I}_3' - \mathbf{R}_{t'}\,\mathbf{P}_t\,\mathbf{Q}_t\right) \ .$$

LINEAR SYSTEMS. In the last step the final sparse symmetric quadratic form $\mathbf{E}^S$ is constructed as half of the Hessian of E^S in parallel by a weighted segmented reduction operation [SHZO07, 5]. In a single summation step all gradient operators $\mathbf{E}_t$ and $\mathbf{S}_{t,t'}$ that contribute to $\mathbf{E}$ and $\mathbf{S}$, respectively, are summed to give the nonzero coefficients of $\mathbf{E}^S$ by a weighted sums according to

$$E^S_{\alpha\beta} = \frac{1}{2}\frac{\partial^2 E^S}{\partial v_\alpha\,\partial v_\beta} = \frac{1}{2}\left((1-\omega)\,E_{\alpha\beta} + \omega\,S_{\alpha\beta}\right) \ .$$

We minimize $E^S(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}\,\mathbf{E}^S\,\mathbf{v}$ on the CPU subject to user-defined Dirichlet boundary conditions (see Section 3.5.1). The resulting linear systems are symmetric positive-definite and sparse with about 1.5% non-zero entries on average. The linear systems are solved by state-of-the-art direct solvers, namely a sparse Cholesky factorization in combination with an approximate minimum degree preordering to reduce fill-in [CDHR08]. We exploit the fact that the structure of the linear system stays fixed in consecutive minimization steps, which allows the precomputation of a symbolic factorization that strongly accelerates the optimization. Experiments reveal that using this direct CPU solver is two orders of magnitude faster than a GPU-based sparse preconditioned conjugate gradient solver. This property was already observed by Botsch et al. [BBK05a] for CPU-only implementations of these solvers for related discretized energies on triangular meshes.

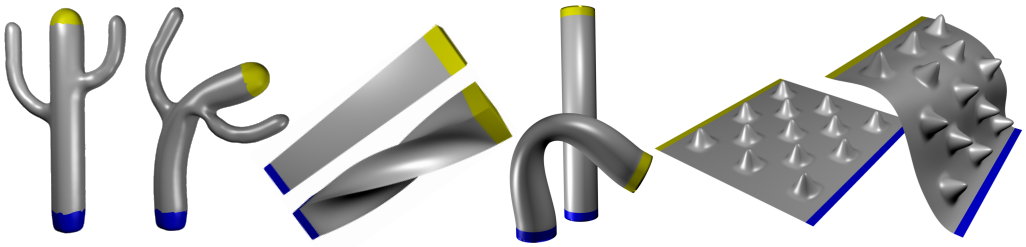
Finally, for the same reasons as already discussed in Section 3.5.2, we solve (31) by a numerical shape integration on the GPU using a single-step Adams-Bashforth-Moulton integration scheme [PTVF07].

4.3 ANALYSIS AND RESULTS

In order to analyze our approach, we apply it to the four benchmark surface deformation problems defined by Botsch and Sorkine [BS08]. The different deformation problems are designed to highlight varying deformation characteristics that are hard to maintain by (linear) single-step approaches simultaneously, e.g., translational or rotational awareness and local detail preservation. Our results are shown in Figure 18 (top) together with the initial benchmark surfaces. For every problem our (nonlinear) near-isometric continuous deformations yield plausible and visually convincing deformations.

Additionally, we compare our deformations to the single-step deformations described in [BPGK06] (PriMo), [BSPG06] (ThinShells), [ZRKS05] (GradientEd), [SCL*04] (LaplacianEd), and [LSLCO05] (RotationInv). The deformation distortion values are summarized in the table of Figure 18. We use the $2d$ deformation errors E^{2d}_{isom}, E^{2d}_{auth}, and E^{2d}_{conf} derived in Appendix B for the considered two-manifold surfaces. For our method, we use four different values of ω. Since the absolute values of the distortions do not have a geometric meaning (because they depend on a particular triangulation), we normalized them by the distortion of our method with $\omega = \frac{1}{3}$. (A number above 100% in the table indicates a higher distortion than for our method with $\omega = \frac{1}{3}$.) We compare the different deformation results of the benchmark shapes and additionally visualize local isometric distortions measured by E^{2d}_{isom} (conformal and authalic errors are similar) in Figure 19. Both the distortion values and the deformation results indicate that our method computes deformations of high quality for the various different surface types. Yet, as our method is nonlinear, a direct comparison to the listed methods is fair only with the nonlinear single-step PriMo approach by Botsch et al. [BPGK06]. Our approach computes deformations that are equally convincing, but haven even lower distortion values compared to PriMo deformations. A possible reason is the lower deformation smoothness of this approach, which is indicated by, e.g., the PriMo Bar example in Figure 19. In contrast, our proposed smoothness energy term measures smoothness globally and seems to give superior results. Also note that our approach performs especially well on the Bumpplane problem, since only at the small junctions of the bumps to the underlying plane minimal distortions are introduced and the remaining surface deforms isometrically with correct detail orientation.

Timings were measured on a Linux 2.6GHz AMD Opteron system with 8GB RAM and a NVIDIA GTX 280 running CUDA. The number of required integration steps for each problem was 13 (Cactus), 44 (Bar), 46 (Cylinder) and 76 (BumpPlane), and the number of vector-field evaluations was roughly twice as many. The first three models can be modified at interactive rates. However, performance is impaired for the very large BumpPlane model. Here, the sparse solver becomes the bottleneck of the optimization due to the size of the arising linear systems, which was not the case for all other examples in this chapter.



CACTUS BAR CYLINDER BUMPPLANE

		PRIMO	THINSHELLS	GRADIENTED	LAPLACIANED	ROTATIONINV	Our Method: $\omega = 1$	$\omega = \frac{2}{3}$	$\omega = \frac{1}{3}$	$\omega = 0.01$
Cactus 70° Bend $5261V$ $10518T$										
E^{2d}_{isom}	(%)	128	215	199	831	242	101	101	100	90
E^{2d}_{auth}	(%)	125	196	194	739	96	96	95	100	81
E^{2d}_{conf}	(%)	167	272	275	1151	243	101	100	100	93
Time	(s)						4	4	4	5
Bar 135° Twist $6084V$ $12106T$										
E^{2d}_{isom}	(%)	411	2874	369	2586	333	109	107	100	32
E^{2d}_{auth}	(%)	130	10168	137	6951	878	107	105	100	55
E^{2d}_{conf}	(%)	331	96	298	1230	336	110	107	100	22
Time	(s)						11	12	13	13
Cylinder 120° Bend $4802V$ $9600T$										
E^{2d}_{isom}	(%)	184	1074	365	783	346	106	104	100	33
E^{2d}_{auth}	(%)	139	715	344	661	33	106	105	100	33
E^{2d}_{conf}	(%)	164	212	340	1016	242	105	104	100	31
Time	(s)						8	8	9	9
Bumpplane Translation $40401V$ $80000T$										
E^{2d}_{isom}	(%)	21478	24163	694283	366961	508844	118	112	100	10
E^{2d}_{auth}	(%)	18070	20611	513386	280151	458046	116	111	100	12
E^{2d}_{conf}	(%)	12911	11329	383755	654916	531962	116	111	100	13
Time	(s)						212	241	275	287

Figure 18: Benchmark Deformations. Standard surface deformation problems specified by Botsch and Sorkine [BS08] with prescribed handle (•) and fixed (•) regions. Our continuous deformation results are computed with $\omega = \frac{1}{3}$ and shown on the top. The table underneath gives distortion values (see Appendix B) and deformation times. All distortion values are relative to our result with $\omega = \frac{1}{3}$ (marked •).

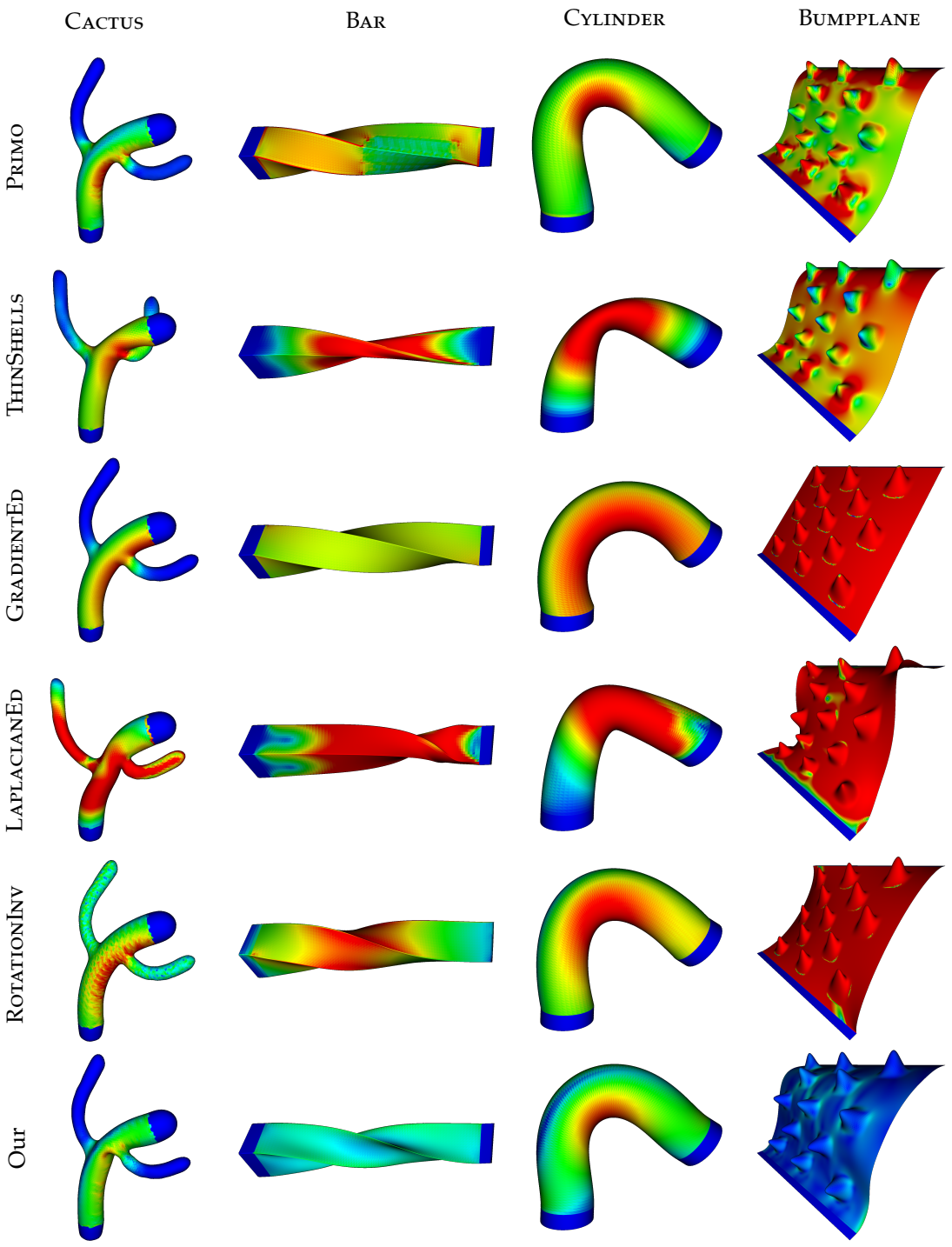


Figure 19: Deformation Comparison. We compare our continuous deformations (bottom, $\omega = \frac{1}{3}$) to other surface deformation methods. The color coding visualizes local isometric distortions measured by E^{2d}_{isom} (see Appendix B). Color scales range from low (•) to high (•) errors and are always scaled equally for each shape.

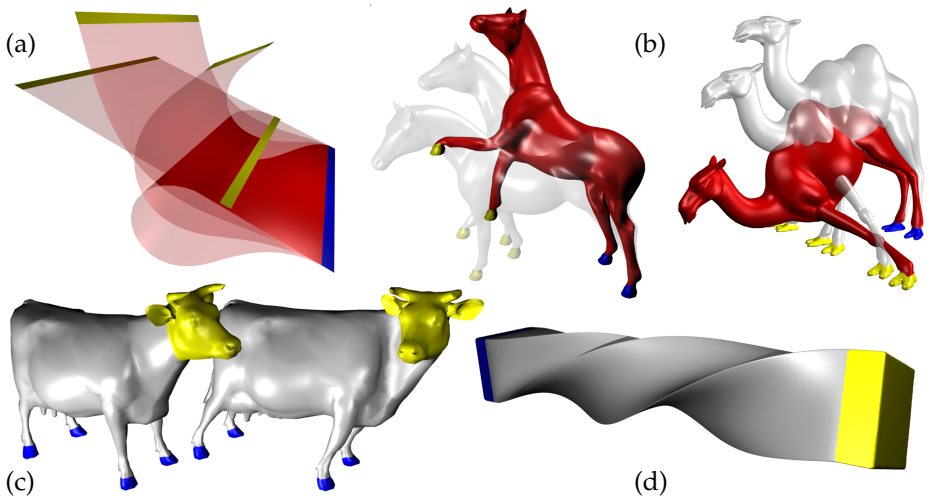


Figure 20: Continuous Surface Deformation Examples. (a) Perfectly isometric deformation of developable plane. (b) Continuous deformations using multiple handles. (c-d) Twisting deformation of the head of the cow and a strong twist of a bar by 280° that is not achievable by single-step deformations.

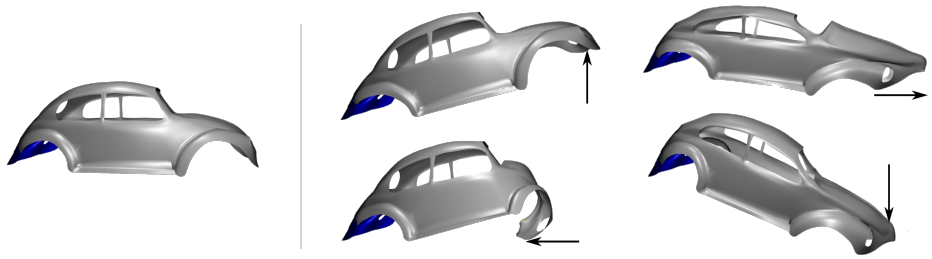


Figure 21: Beetle Car Deformations. The original beetle model (left) is deformed by fixing the rear of the car and moving the handle at the engine hood into four different directions (right).

Further examples of our approach are shown in the Figures 20 and 21. Figure 20 (a) shows a perfectly isometric deformation ($E^{2d}_{isom} \equiv 0$) of a developable surface computed using our approach. Figure 20 (b) shows that deformations can contain different handle paths: the front legs of the animals were moved in different directions, yielding plausible deformations. Figure 20 (c) shows a twisting deformation of the head of a cow model. There, the body leans forward to compensate metric distortion. In Figure 21 a beetle car model is deformed in four antipodal directions giving four rather different deformation results. The accompanying video of [MRT12] shows additional results [1].

The impact of the variation of the weight ω is illustrated in Figure 22. The example shows that too small weights can lead to unpleasant artifacts if defor-

1 The video is located in the additional material folder `addmaterial/isomdef`.

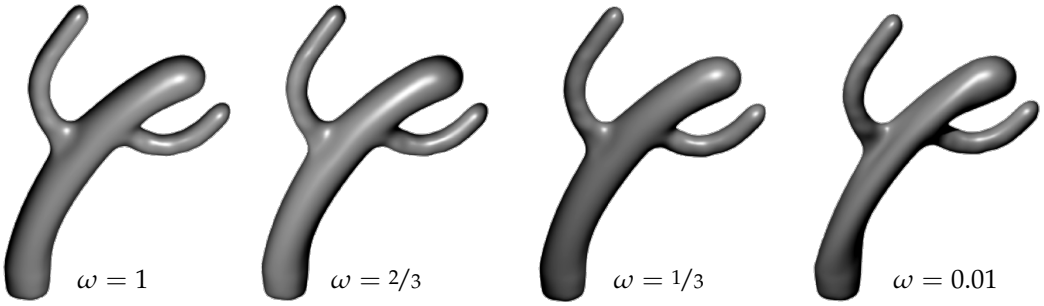


Figure 22: Energy Weighting. Smaller weights ω lead to small distortion but may produce deformation artifacts if the boundary conditions do not allow for near-isometric deformations ($\omega = 0.01$). Slightly higher ω values result in less artifacts and similar global deformations.

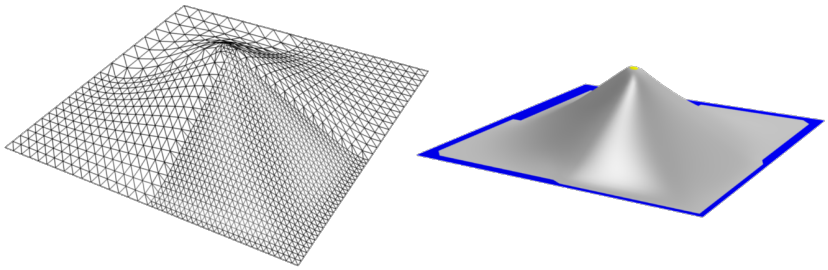


Figure 23: Mesh Resolution Insensitivity. Pulling the handle vertices (•) at the center up vertically produces a smooth deformation (right) independent of the inhomogeneous mesh resolution (left).

mations cannot be perfectly isometric and distortion minimization is enforced at the expense of vector field smoothness in (32).

4.4 SPECTRAL ENERGY PROPERTIES

In addition to the definition of continuous deformations, the norm $\mathbf{E}^S$ of the quadratic form $E^S(\mathbf{v}) = \|\mathbf{v}\|^2_{\mathbf{E}^S}$ has interesting properties in the spectral domain [ZvKD10]: let $(\lambda_i, \mathbf{v}_i)$, $i = 1, \ldots, 3\,|\mathcal{V}|$, be the eigenvalue and eigenvector / discrete eigenfunction pairs of $\mathbf{E}^S$ in ascending order of λ_i. As ${\mathbf{v}_i}^{\mathrm{T}}\,\mathbf{v}_j = \delta^{ij}$, we have $E^S(\mathbf{v}_i) = \lambda_i$. For general surfaces $\mathbf{E}^S$ has a rank deficit of six, i.e., $\lambda_i = 0$, $i = 1, \ldots, 6$, and $\lambda_i > 0$, $i > 6$, due to the definiteness of $\mathbf{E}^S$. The six kernel dimensions correspond to the six degrees of freedom of globally rigid vector fields, such that the corresponding eigenfunction are a basis of such fields. The first twelve eigenfunctions of a simple model are shown in Figure 24.

The lower spectrum of eigenfunctions can be used for shape analysis to detect structurally stiff regions in the shape: intrinsic stiffness corresponds to a low amount of instantaneous motion, which induces low amount of distortion. As

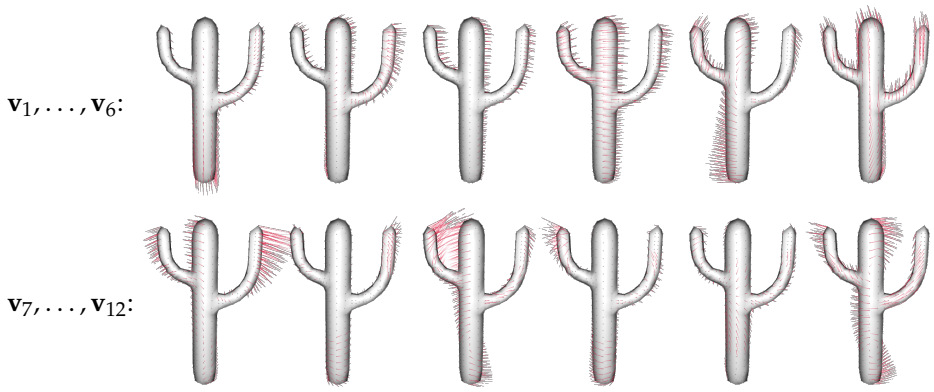


Figure 24: Energy Eigenfunctions. The first twelve eigenfunction $\mathbf{v}_i$ of $\mathbf{E}^S$. The first six eigenfunctions are a basis of globally rigid vector fields, the next greater eigenfunctions describe near-isometric instantaneous intrinsic shape deformations.

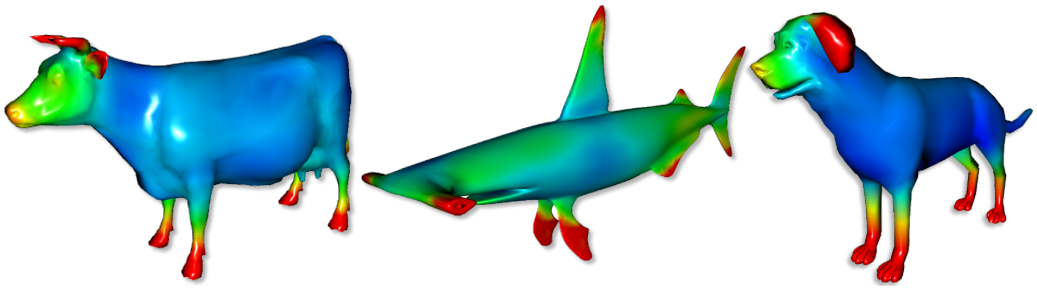


Figure 25: Structural Shape Stiffness. Using the lower spectrum of the first twenty eigenfunctions of the isometric energy the local shape stiffness is computed. Regions have lower (•) to higher (•) tendency to move instantaneously while inducing low amount of distortion.

this type of instantaneous motion is given by the lower spectrum of n eigenfunctions, we can sum the norm of the contribution of the eigenfunction at each vertex for all $i < n$. The resulting scalar field describes the local tendency to move instantaneously while inducting low amount of distortion. In Figure 25 we show three examples of such a shape descriptor.

The lower spectrum can also serve as descriptors of a feature space for, e.g., shape segmentation: for each vertex we measure the norm of each eigenfunction and use the resulting n-dimensional feature vector at each vertex for a segmentation similar to the method of Huang et al. [HWAG09]. Note that their method uses a feature description that is based on a modal analysis of a single-step ARAP energy, whereas our vector fields directly describe first-order instantaneous deformations. Instead of a k-means clustering of the original method we use spectral clustering based on normalized cuts proposed by Shi and Malik [SM00]. This method has the advantage that the normalized cut measure allows for an automatic cluster number selection. Moreover, segmentations generally

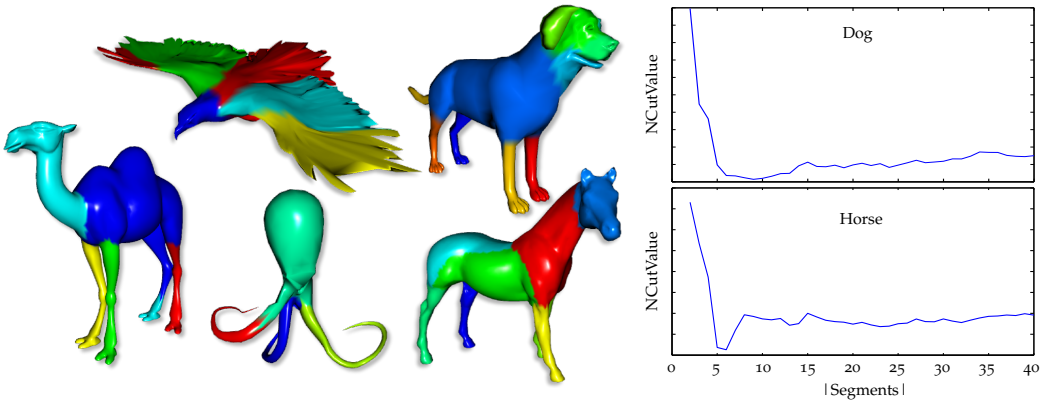


Figure 26: Stiffness-based Segmentation Results. Spectral clustering on energy eigenfunctions as feature vectors is used for region classification. Associated shape regions are colored equally. The graphs (right) show the normalized cuts value for different numbers of clusters for the dog and horse segmentation, respectively.

showed a higher quality compared to k-means clustering. Surface segmentation results using are shown in Figure 26.

4.5 DISCUSSION

We continue to discuss several aspects of our continuous deformation approach.

ISOMETRIC SURFACE ENERGIES. For our approach we developed a new discrete vector field energy that measures instantaneous isometric distortions induced by the field acting on surfaces. The usual approach (30) as used in [KMP07, EPT*07] is not sufficient because of two reasons: first, it does not consider the triangle shapes and sizes. In Figure 16b, we show the impact on the deformation stability in a simple experimental evaluation: for this experiment the non-planar surface $z(x,y) = \frac{1}{2}(1+x)(1-x)(1+y)(1-y)$ was sampled over the interval $[-1,1] \times [-1,1]$ as shown in Figure 16b (1-2). Note that an irregular tessellation was chosen. The deformation was defined by keeping the boundary fixed and translating the (•) region in the direction of the z-axis. Figures 16b (3-5) show three steps of an Euler integration by minimizing $\bar{E}(\mathbf{v})$ defined by (30), while Figures 16b (6-8) shows the same steps by minimizing our $E^S(\mathbf{v})$ with $\omega = 0$ i.e., *without* the smoothness term. Euler integration was chosen to emphasize the artifacts. Even this very small example clearly shows that the measure $\bar{E}(\mathbf{v})$ does not yield acceptable results. We point out that this is not due to missing regularization (the initial surface is not planar), the corresponding linear operators are sufficiently well conditioned.

The second advantage of our energy is that it offers a simple method to incorporate the smoothness of the deformation, i.e., to prevent appearance or disappearance of sharp edges during the deformation. While we do not see a straightforward way to extend (30) in this direction, our measure can easily be extended in this direction as shown in Section 4.1.4.

We conclude this aspect by visualizing the tessellation-insensitivity of our deformation method for a simple example. Figure 23 shows a plane that is triangulated with different resolutions and then deformed trivially: the tessellation has no effect on the resulting surface shape due to the design of our energy, which is a surface-integrated quantity (see also Figure 16a and Section 4.1.3).

RELATION TO ARAP DEFORMATIONS. Our method is based on matching deformation vector fields to closest rigid vector fields. It can therefore be interpreted to be an instantaneous relative of the single-step as-rigid-as-possible (ARAP) deformations that approximate closest rotational deformation gradients (see, e.g., Section 7.4.2). Hence, both approaches minimize similar nonlinear deformation errors in two different ways: our method is based on ODE integration, whereas the ARAP energy is commonly minimized iteratively. The following differences are worth noting: due to the local direct minimization of the nonlinear energy, the converged ARAP result depend on a deformation *initialization*, e.g., a fast to compute linear Laplacian deformation [SA07]. Our approach does not require an initialization by a different deformation method as vector fields are optimized on the currently integrated surface. On the other hand, our method does not directly optimize the distortion of the final deformation, but rather tries to minimize the *variation* of isometry in each integration step. This property is advantageous if an animation of the deformation is also required. However, it may also lead to an accumulation of distortion relative to the original surface, although our experiments in Section 4.3 indicate that the total deformation quality is still high. Another consequence of our model is the dependency of the deformation on the whole path of each handle.

HANDLE PATH DEPENDENCY. Being a continuous method the result of our deformation depends not only on the final position of the constrained vertices, but also on the paths on which they move from starting to final position. This is a significant difference to single-step deformation approaches. Figure 27 illustrates this property. There, the shape (a) is deformed by moving the yellow boundary to the right while keeping the blue boundary fixed (b). The successive reverse deformation (c) gives the original shape with minimal positional variation, i.e., for certain handle paths deformations are *reversible*. Our deformations have this property if only a small amount of distortion is induced by the deformation constraints, e.g., the deformation in (b) is close to an exact isometry. Contrary, by moving the handle first to the left (Figure 27 (d)) ends up in a significantly different shape (e), as the initial deformation induced a certain amount

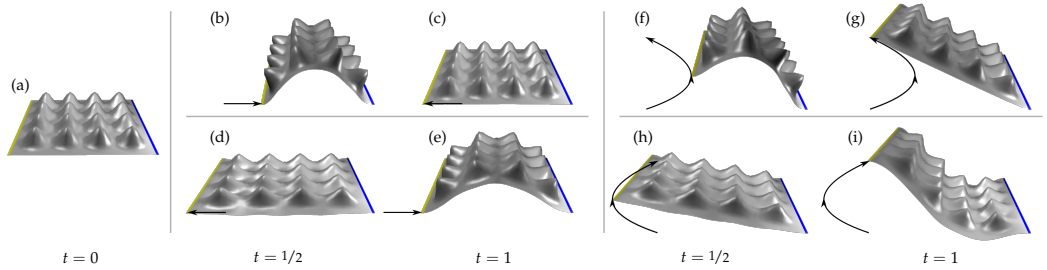


Figure 27: Handle Path Dependency. Deforming the initial surface at $t = 0$ (a) by different handle paths from and to the *same* rest positions results in different deformation at $t = 1$. The images (b-e) show two linear antipodal handle paths, the images (f-i) two parabolic handle paths.

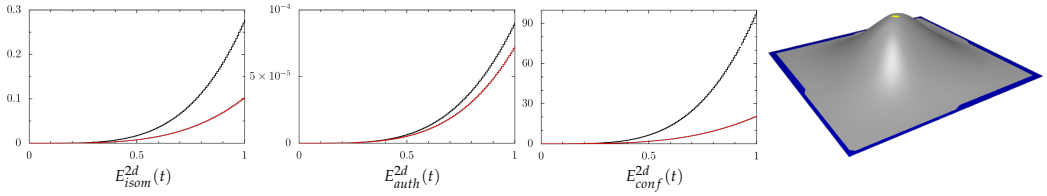


Figure 28: Time-Dependent bi-Laplacian Deformation. Comparison of distortions over time for time-dependent bi-Laplacian deformation (•) and our method (•).

of distortion, which is preserved afterwards. Figures 27 (f-i) show the movement of the handle over two mirrored paths, yielding different final shapes.

While such a path dependence might not always be a desired property, we emphasize that for a number of applications it opens a wider flexibility of the modeling process because it reflects the fact that real materials are never deformed in a perfectly elastic way. The "memory effect" of the deformation gives the look of a combined plastic and elastic deformation of a real material, even though only geometric measures of the surface are considered. Furthermore, it allows to obtain strong twisting with rotation angles greater than π (Figure 20 (d)) and knots [vFTS07b, Figure 14], which are hard to compute by path-independent methods.

COMPARISON TO TIME-DEPENDENT BI-LAPLACIAN. We point out that it is not sufficient to modify existent linear single-step energies to operate in a continuous instantaneous setting to minimize isometric distortion. Consider Figure 28 as an example. For this simple deformation a bi-Laplacian operator (see [BS08]) was discretized for every time step, partial deformations were integrated within the same solver, and the same boundary constraints as in Figure 23 were applied. Comparing results, metric distortions are still significantly higher — they didn't improve much — in comparison to our method. This is not surprising as different energies are minimized. We remark as bottom line that by breaking a discrete bi-Laplacian deformation trivially into a "continuous" deformation

one cannot achieve the same effect as our vector field-based method. Other discrete state-of-the-art methods are also likely to exhibit higher distortion by this strategy compared to our approach, too.

LIMITATIONS. We see the main limitation of our approach in the relatively high computation times: despite the fact that we accelerate our nonlinear approach using the GPU (and the experiments were performed on a relatively slow machine), the technique is far less interactive than state-of-the-art linear frameworks if applied to large models and thus does not scale to very large meshes yet. We also mention that for the linear operators the memory footprint is probably higher. As discussed in Chapter 2, nonlinear deformations of higher quality are generally more expensive to compute. Our method shares this trade off with other nonlinear deformation approaches. Efficiency for meshes of higher resolution could be improved using, e.g., multiresolution or subspace techniques [KMP07, WDAH10, FB11, JBK*12]. However, these accelerations are likely to spoil local deformation quality, which is the main focus of our approach.

The path dependence of our method can as well be seen as a limitation. We claim it is a features and an integral property of continuous vector field-based deformations. However, we are aware that depending on the application path dependence may be also interpreted as an artifact.

4.6 SUMMARY

In this chapter, we made the following contributions: we described a method for continuous integration-based deformations of triangulated surfaces that tries to preserve isometry. Results show significantly lower distortion of length, angles, and area for a set of representative shapes compared to existing standard (linear and nonlinear) deformations. Moreover, the results look visually pleasing. For every time step a piecewise linear vector field is constructed by applying a quadratic energy minimization. The energy minimization consists of two combined phases: a local rigid vector field fitting per triangle and a global vector field optimization that measures the deviation from local rigid fields. Both phases are accelerated using the GPU. Our new energy is surface-integrated and is extended to incorporate smoothness. Our modeling metaphor defines handle paths. Both the final position of the handles and the path influence the deformation.

The most prominent issue for future research is the further improvement of the performance. Another interesting challenge is the boundary value problem of path planning where the optimal path between two poses is determined. To do so the method by Kilian et al. [KMP07] can be extended to use our vector field energy formulation.

5

CONTINUOUS DEFORMATIONS OF IMPLICIT SURFACES

In the previous chapters of this work, we considered continuous deformations of explicit shapes that are represented by triangular and tetrahedral meshes. Alternatively, a common approach in computer graphics is to represent shapes in an implicit form, i.e., as the isocontours of (volumetric) scalar fields.

There are a number of approaches for implicit surface deformation and modeling. Most often, these methods focus on deforming *one* single isosurface. Volume data, or more generally scalar fields, contain much more information than just one isosurface. In fact, there is a whole family of isosurfaces that may represent different kinds of information: in CT or MRI data from medical imaging applications, different isosurfaces describe transitions between different materials (like bone, tissue, or air), in other applications different isosurfaces may contain distance information to a particular isosurface of interest. Generally, for most volume data sets there is more than a single isosurface of interest.

If one is interested in the deformation of one particular isosurface, a generic solution is to extract it, then apply an explicit deformation, and finally perform an implicitation. Such an approach does not regard any volumetric information except for the location of one isosurface. However, if the complete volume is of interest, a good deformation should incorporate the whole field, i.e., it should take care of the shape of *all* isosurfaces.

In this chapter, we present a method for the computation of such deformations. Deformations are again defined continuously by a path line integration of an explicitly constructed divergence-free vector field. We show that they have the following properties: the volume inside each isosurface is preserved during the deformation, no isosurface changes its topology, no new critical points of the volume data set appear or disappear, and a C^1 continuity of the isosurfaces is preserved during the deformation. The desired volume preservation is justified by the fact that many common materials approximately preserve their volume under deformation. At the same time, topology preservation allows to construct complicated shapes with a pre-defined simple topology.

The deformation is computed numerically by an efficient and unconditionally stable backward Lagrangian integration scheme that is performed by the GPU. For interactive real-time modeling, we visualize isosurfaces on the grid of the underlying volume data set. In addition, an exact reconstruction can be used to retrieve an arbitrary isosurface with exact topology and high accuracy. We apply our technique to deform volume data sets and to model complex families of isosurfaces with a pre-defined simple topology.

5.1 CONTINUOUS ISOSURFACE DEFORMATIONS

Isocontours $\mathcal{S}_\alpha$ of a scalar field $s(\mathbf{x})$ in a spatial domain $\mathcal{D}$ are the set of points $\mathcal{S}_\alpha = \{\mathbf{x} \in \mathcal{D} \,|\, s(\mathbf{x}) = \alpha\}$ whose scalar value matches a prescribed isovalue α. Given an initial scalar field $s_0(\mathbf{x})$ over $\mathcal{D}$, we consider continuous deformations over time as the computation of time-dependent scalar fields $s(\mathbf{x}, t)$ with $s(\mathbf{x}, t_0) = s_0(\mathbf{x})$. Then time-dependent scalar fields induce continuous deformations of all $\mathcal{S}_\alpha$. In this chapter, we only consider the $3d$ case $\dim(\mathcal{D}) = 3$, although most concepts generalize to other dimensions, too.

We define the deformation by a $3d$ time-dependent vector field $\mathbf{v}(\mathbf{x}, t)$ describing the transport of the isosurfaces over time. We make use of the concept of a *flow map* $\boldsymbol{\phi}$ of $\mathbf{v}$. The map $\boldsymbol{\phi}$ assigns to each point $\mathbf{x}_0 \in \mathcal{D}$ where a massless particle is seeded at time t_0 the point where it is located at time τ under a kinematic path line integration of $\mathbf{v}$:

$$\boldsymbol{\phi}_{t_0}^{\tau}(\mathbf{x}_0) = \mathbf{x}_0 + \int_{t_0}^{\tau} \mathbf{v}(\mathbf{x}(t), t)\, \mathrm{d}t \tag{34}$$

with $\mathbf{x}(t_0) = \mathbf{x}_0$ and $\dot{\mathbf{x}}(t) = \mathbf{v}(\mathbf{x}(t), t)$. Given s_0 and $\mathbf{v}$, finding the deformation s is equivalent to solving the PDE

$$\frac{\partial}{\partial t} s = -(\nabla s)^{\mathrm{T}} \mathbf{v} \quad \text{with} \quad s(\mathbf{x}, t_0) = s_0(\mathbf{x})\,, \tag{35}$$

where $\nabla\cdot$ is the (spatial) gradient operator. Equation (35) is known as the *fundamental level set equation* [OF01], the solution is an initial value problem.

There are a variety of approaches to solve (35) that are based on a discretization of s and $\mathbf{v}$ in both, space and time [OF02]. Eulerian integration schemes compute s at a time step t_i from s and $\mathbf{v}$ at the time t_{i-1} regarding only specific locations, e.g., grid points. Lagrangian schemes follow the trajectories of particles forward over all time steps, while semi-Lagrangian integration techniques evaluate s at time t_i by a single backward integration step of $\mathbf{v}$. Furthermore, there are hybrid schemes incorporating particle integrations to correct errors in the (Eulerian) integration of the PDE (35).

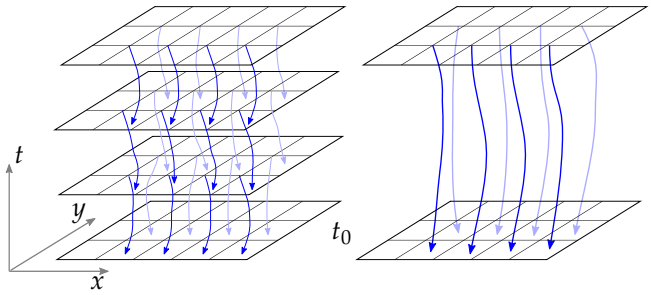


Figure 29: Semi-Lagrangian (left) versus backward Lagrangian scheme (right). Both schemes update s at grid points by integrating backward in time. The semi-Lagrangian scheme, however, does this for each step. In our setting we can apply a fully backward Lagrangian scheme, which requires only a single evaluation (interpolation) of s at t_0. This scheme is simpler, more efficient, and more accurate.

Eulerian schemes often suffer from stability problems as they are only conditionally stable. A common problem with fully Lagrangian schemes is a faithful reconstruction of s since the final particle distribution may be highly non-uniform. This is why semi-Lagrangian schemes are often preferred (e.g., in fluid simulation [Sta99]). They ensure reconstruction by choosing grid points as evaluation points thus reverting to the spatial grid after each time step.

For our application we can rely on a much simpler integration scheme: a *backward Lagrangian integration*. Note that standard methods to solve (35) steming from level set theory and numerical flow simulation often assume that $\mathbf{v}$ and s are *not* independent. In fact, usually the definition of $\mathbf{v}$ incorporates local components of s such as its gradient, Hessian, or the Gaussian and mean curvature of its isosurfaces, leading to the fact that $\mathbf{v}$ at a time t_i is not known until s has been computed in t_i. We emphasize that this is not the case for our approach: we define $\mathbf{v}$ *independently* of s. This allows for using a backward Lagrangian scheme to solve (35): the scalar value at a time t is obtained by a *complete* path line integration back until t_0:

$$s(\mathbf{x}, t) = s_0(\boldsymbol{\phi}_t^{t_0}(\mathbf{x})). \tag{36}$$

This concept is illustrated in Figure 29. Note that for computing $s(\mathbf{x}, t)$, it is not necessary to compute s at any intermediate time steps between t_0 and t. There are two main benefits of the backward Lagrangian scheme: firstly, improved accuracy as we do not suffer from interpolation artifacts that occur for a semi-Lagrangian scheme – the scalar field s is evaluated only once at t_0. Secondly, integration involves fewer data and fewer operations and can be implemented more efficiently.

5.1.1 Deformation Properties

Let $\mathbf{v}$ be a C^1 continuous vector field over $\mathcal{D}$ with the following properties:

- *local support*: $\mathbf{v}$ is non-zero only in some inner region of $\mathcal{D}$ (it is constantly zero at the boundary of $\mathcal{D}$),
- *boundedness*: $\|\mathbf{v}\| < \infty$ and $\|\nabla\mathbf{v}\|_F < \infty$ at any location in $\mathcal{D}$ ($\nabla\mathbf{v}$ is the transposed vector field Jacobian), and
- $\mathbf{v}$ is *divergence-free*, i.e., $\mathrm{Tr}(\nabla\mathbf{v}) = 0$.

Then the deformation s defined by (36) has the following properties:

A. s is *volume-preserving*: the volume inside *every* isosurface remains constant under the deformation,

B. s is *continuity-preserving*: if s_0 is C^1-continuous then s is C^1 as well, and

C. s is *topology-preserving*: no isosurface changes its topology during the deformation.

Property (A) follows directly from the definition of divergence of vector fields [Dav67]. Property (B) has been proven in [vFTS06] for explicit surfaces, the same proof holds for implicit surfaces as well.

Regarding Property (C), we realize that a topology change requires a *critical point* of s, i.e., a point where the scalar field gradient ∇s vanishes [OF02]. Hence, we can rephrase this property as follows: no critical points can appear or disappear during the deformation. All critical points of s are obtained by integrating the critical points of s_0. In order to show Property (C), we observe how ∇s is changing under integration of $\mathbf{v}$ over time:

$$\frac{\partial}{\partial\tau}\nabla s = \lim_{\tau\to t}\frac{\nabla s(\boldsymbol{\phi}_t^\tau(\mathbf{x}),\tau) - \nabla s(\mathbf{x},t)}{\tau - t} = \frac{\partial}{\partial t}\nabla s + \mathbf{H}\,\mathbf{v}\,, \tag{37}$$

where $\mathbf{H}$ denotes the (spatial) Hessian of s. We rewrite the PDE (35) in matrix notation as the scalar product

$$\begin{pmatrix}\mathbf{v}^\mathrm{T} & 1\end{pmatrix}\begin{pmatrix}\nabla s\\ \frac{\partial}{\partial t}s\end{pmatrix} = 0\,. \tag{38}$$

Then computing the gradient of (38) by applying the product rule gives

$$\begin{bmatrix}\mathbf{H} & \frac{\partial}{\partial t}\nabla s\end{bmatrix}\begin{pmatrix}\mathbf{v}\\ 1\end{pmatrix} + \begin{bmatrix}\nabla\mathbf{v}\\ \mathbf{0}^\mathrm{T}\end{bmatrix}^\mathrm{T}\begin{pmatrix}\nabla s\\ \frac{\partial}{\partial t}s\end{pmatrix} = \mathbf{0}\,,$$

Evaluation of the terms and comparison with (37) yields

$$\frac{\partial}{\partial\tau}\nabla s = -(\nabla\mathbf{v})^\mathrm{T}\nabla s. \tag{39}$$

Equation (39) holds everywhere in $\mathcal{D}$ and states that a critical point of s (i.e., $\nabla s = \mathbf{0}$) remains a critical point under the integration of $\mathbf{v}$ (i.e., $\frac{\partial}{\partial \tau}\nabla s = \mathbf{0}$). Conversely, for a non-critical point, ∇s cannot vanish during the integration of $\mathbf{v}$ as otherwise a backward integration starting from the critical point would violate the previous statement.

5.1.2 Definition of the Vector Field

The divergence-free vector field $\mathbf{v}$ steers the deformation. Its definition is not a contribution of this work, since we use the method presented by von Funck et al. [vFTS06], who define deformations of explicit shapes (represented as triangular meshes) by vector field integration. For the sake of completeness, we provide a brief review: essentially, the definition of $\mathbf{v}$ is an interactive process, where $\mathbf{v}$ is defined by three time-dependent scalar fields $e(\mathbf{x}, t)$, $f(\mathbf{x}, t)$, and $r(\mathbf{x}, t)$, together with two thresholds r_i, r_o. The region field r and thresholds r_i and r_o define an inner region of full deformation, a blended intermediate region, and a region of zero deformation. The full deformation is defined by scalar fields p and q as

$$\mathbf{v} = \nabla p \times \nabla q$$

with

$$p(\mathbf{x}) = \begin{cases} e(\mathbf{x}) & \text{if } r(\mathbf{x}) \leq r_i \\ (1-b)\, e(\mathbf{x}) & \text{if } r_i < r(\mathbf{x}) \leq r_o \\ 0 & \text{if } r_o < r(\mathbf{x}) \end{cases}, \quad q(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } r(\mathbf{x}) \leq r_i \\ (1-b)\, f(\mathbf{x}) & \text{if } r_i < r(\mathbf{x}) \leq r_o \\ 0 & \text{if } r_o < r(\mathbf{x}) \end{cases},$$

and $b = b(r(\mathbf{x}))$ is a polynomial blending function with $b(r_i) = 0$, $b(r_o) = 1$, and $\frac{\mathrm{d}}{\mathrm{d}r}b(r_i) = \frac{\mathrm{d}}{\mathrm{d}r}b(r_o) = \frac{\mathrm{d}^2}{\mathrm{d}r^2}b(r_i) = 0$ represented in Bernstein-Bézier form. Vector fields $\mathbf{v}$ constructed this way are guaranteed to be divergence-free [Dav67]. As these vector fields are *independent* of s they allow the usage of the proposed backward Lagrangian integration scheme.

5.1.3 Deformation Types

With the choice of the scalar fields e, f, and r we can define different types of deformations. Since our approach does not focus on a particular isosurface, e, f, r should be chosen to act on a family of isosurfaces simultaneously. For this purpose we adapt the approach presented in [vFTS07b], which uses a spatial curve that guides the deformation, to our setting of deforming a family of implicit surfaces: the user specifies $r(\mathbf{x}, t_0), r_i, r_o$, and a curve $\mathbf{c}(t)$ with $r(\mathbf{c}(t_0), t_0) \leq r_i$ interactively. The region field r and threshold r_i, r_o define the regions of full and blended deformations, respectively, and $\mathbf{c}$ describes the path and orientation of

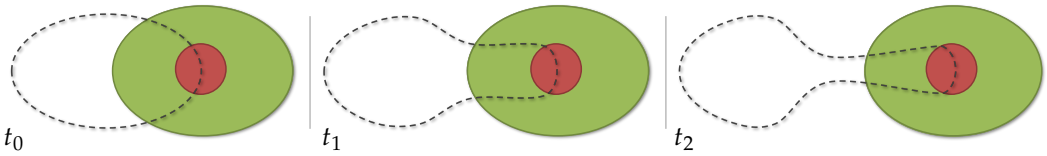


Figure 30: Deformation Principle. The dashed isocontours $s(\mathbf{x}, t) = 0$ are deformed by a translational motion. Shown are three time steps. The deformation region consists of a full deformation (•) and a blended deformation (•) part.

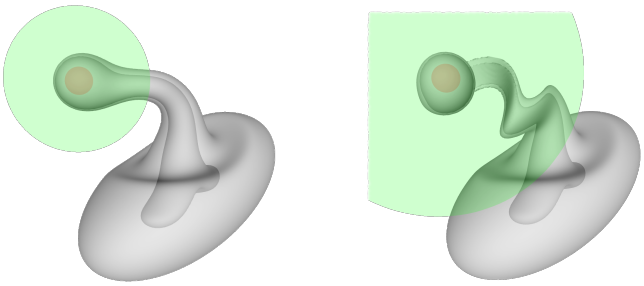


Figure 31: Simple Deformations. Examples for translation along a curve (left) followed by multiple rotations (right). Inner region (•) and intermediate region (•) are highlighted. For each case *two* deformed isosurfaces are shown.

the inner region over time. This can be imagined as sweeping a deformation tool with local support along $\mathbf{c}$. See Figure 30 for an illustration of the basic interaction metaphor. Defining $\mathbf{N}(t) = (\mathbf{t}(t), \mathbf{n}(t), \mathbf{b}(t))$ as the moving normalized Frenet frame of $\mathbf{c}$, we obtain

$$r(\mathbf{x}, t) = r\Big(\mathbf{c}(t_0) + \mathbf{N}(t_0)\,\mathbf{N}(t)^T\,(\mathbf{x} - \mathbf{c}(t))\;,\; t_0\Big)\;,$$

and e, f describe a translation in the direction of the curve tangent $\dot{\mathbf{c}}$ if computed by

$$e = \mathbf{n}^{\mathrm{T}}\,(\mathbf{x} - \mathbf{c}) \quad \text{and} \quad f = \|\dot{\mathbf{c}}\|\mathbf{b}^{\mathrm{T}}\,(\mathbf{x} - \mathbf{c})\;.$$

Optionally, we allow rotational deformation around an axis given by a center $\mathbf{p}$ and a direction $\mathbf{d}$:

$$e = \mathbf{d}^{\mathrm{T}}\,(\mathbf{x} - \mathbf{p}) \quad \text{and} \quad f = \|\mathbf{d} \times (\mathbf{x} - \mathbf{p})\|^2\;.$$

Figure 31 illustrates two examples: a translation in the inner region following a curve $\mathbf{c}$ (left), and a subsequent rotation around an axis through the red center (right).

5.2 IMPLEMENTATION

In this section, we describe the implementation of our continuous isosurface deformation approach. Generally, we discretize scalar fields on uniform grids,

and we use a tricubic C^1 interpolation for evaluation of function values and gradients. For path line integration we apply a fourth-order Runge-Kutta scheme with adaptive step size control (see, e.g., [PTVF07]).

5.2.1 User Interface and Real-Time Visualization

The user first loads an initial scalar field, which is uploaded to the GPU. Then she can set parameters of the tool interactively, e.g., type of deformation and region fields, and perform sweeps in real-time. A particular or multiple isosurfaces are displayed for interactive modeling, the isovalues can be varied freely. In our experiments this approach has shown to be more intuitive than a more general real-time volume rendering approach (see, e.g., [HKRW06] for an overview). Our visualization uses a real-time GPU version of the marching cubes algorithm based on histogram pyramids [DZTS08], which does not show any significant impact on run time of the main algorithm in all our experiments. Marching cubes is performed only if either the scalar field or the isovalue changes, and the result is saved into a vertex buffer for rendering.

5.2.2 Interactive Scalar Field Manipulation

In all our examples we use a grid resolution of 256^3. The maximum resolution is mainly limited by available memory as the whole field is required to persist in GPU memory for visualization. We remark that an out-of-core implementation of our general approach is straightforward, currently it is the marching cubes visualization that requires that all data persists on the GPU. For the deformation approach itself we use s and a temporary buffer with size of the region of interest on the GPU. Both, path line integration and scalar field interpolation are performed by the GPU.

Deformations are specified as control points of the sweep curve $\mathbf{c}$ together with additional properties such as region fields. Deformations can be extended by editing the current curves and adding new ones interactively, restriction provides an undo functionality. The associated parameter sets are sent to the GPU, their memory footprint is not significant. When evaluating $\mathbf{v}$, we exploit the fact that the individual operations have local support, i.e., $\mathbf{v}$ is non-zero only within a region of interest and can therefore be evaluated on a smaller grid.

For practical real-time editing we relax the integration scheme slightly using intermediate results: we partition a sweep along the guiding curve $\mathbf{c}$ (see Section 5.1.3) into time intervals, each of which describes a partial modification. From the arising series of modified scalar fields only the most recent scalar field is stored in a GPU buffer of constant size. This means that we do not integrate all way back to t_0 but only to the last interval bound. However, each edit still

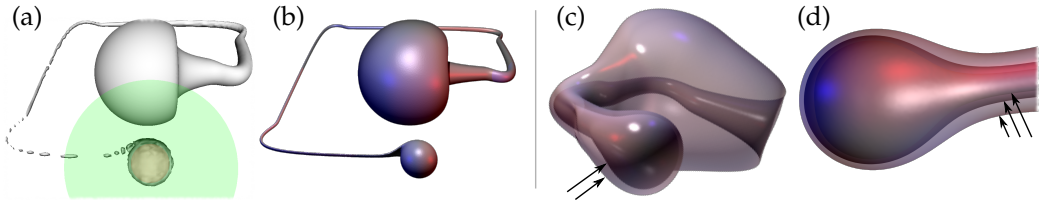


Figure 32: Reconstructions Comparison. (a) An extreme deformation leads to artifacts for marching cubes reconstructions during interactive modeling. (b) The same surface reconstructed exactly. Two (c) and three (d) interfaces (indicated by arrows) that get close to each other are reconstructed correctly.

comprises a significant time interval and potentially many integration steps – we are not switching to a semi-Lagrangian scheme. We use this compromise of buffering intermediate edits to balance accuracy and efficiency. This way we save on the integration process with shorter time intervals that only reach back to the previous partial edit, and we can guarantee real-time response. The price is a slight loss of accuracy due to interpolation of intermediate data. Of course, we can control granularity of time intervals, and we can even invest some extra time and do a full path line back integration to t_0 at any time to visualize a better approximation.

5.2.3 Offline High-Quality Isosurface Extraction

The quality of the rendered isosurfaces depends on the grid resolution for the marching cubes algorithm. We use the same resolution as for computation of $s(t)$ and are generally limited by GPU memory. While this is acceptable for real-time visualization, it is evident that artifacts show up whenever the sampling rate becomes too low, depending on the particular isovalue (see Figure 32 (a)). This is not a failure of our method but a failure of reconstruction. To show this property we extract high-quality isosurfaces using the surface meshing routines provided by the CGAL library [3]. We chose this library because the underlying adaptive meshing algorithm by Boissonnat et al. [BO05] is extremely robust and produces highly accurate triangulations of the isosurface. Even extreme examples with very thin or near overlapping surface parts are reconstructed faithfully, see Figure 32 (b-d). The high-quality isosurfaces are extracted at the cost of using a sequential CPU implementation that takes the recorded parameter sets as input and integrates the path lines. We note that this is a inherently sequential process: the meshing algorithm does not have enough information of the surface and treats evaluation as a black-box component, the so called oracle. We do not have any influence on location and order of evaluation points, which seem highly non-uniform. As a consequence this is clearly an off-line process. We observed that sampling $s(t)$ on a uniform grid and using the CGAL surface mesher on this as input provides a significantly more efficient and fairly accurate recon-

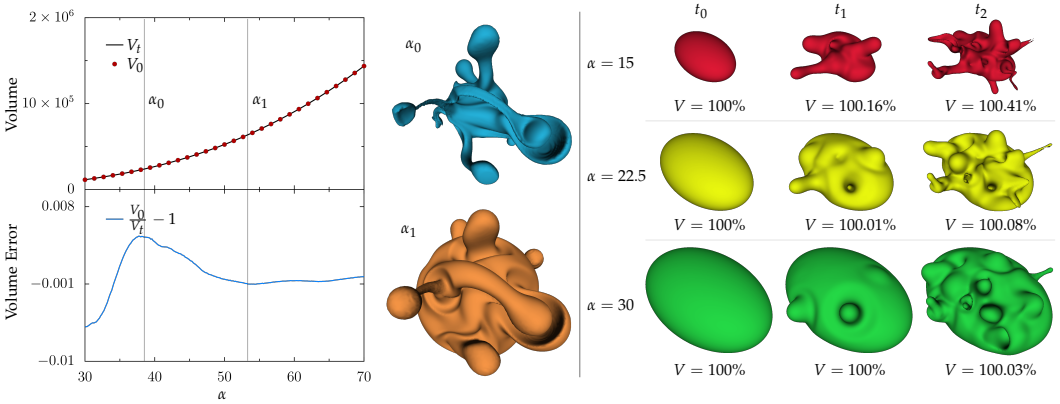


Figure 33: Volume Preservation. Left: Evaluation of the volume V_α of a randomly deformed (•) and original sphere ($V_0 = 4\pi\alpha^3$, •), plotted over varying isovalues α ($\equiv$ radii). The curves are nearly identical. The relative volume differences are shown underneath. Two particular isosurfaces are shown on the right for the isovalues α_0 and α_1. Right: Three isosurfaces at isovalues α shown at times t_0, t_1, t_2. Volume variation is low even for the applied extreme deformations.

struction. However, this compromise is less exact and depends on the sampling rate.

5.3 ANALYSIS

We continue to analyze properties of our approach and its implementation.

VOLUME PRESERVATION. We compare the volume preservation of deformed isosurfaces to ground truth. The exact total volumes V of closed triangulated isosurfaces are computed by signed tetrahedra volumes

$$V = \frac{1}{6} \sum_{(i,j,k) \in \mathcal{T}} \det\left(\begin{bmatrix} \mathbf{x}_i & \mathbf{x}_j & \mathbf{x}_k \end{bmatrix} \right)$$

for sets of triangles $\mathcal{T}$ and vertex coordinates $\mathbf{x}_i$ (see, e.g., [vFTS08]). Even though the volumes are computed from meshes computed interactively by marching cubes (and not by using the high quality extraction), our experiments already confirm volume preservation. Figure 33 (left) plots volume of isosurfaces over their isovalues for an initial sphere and a randomly deformed version (depicted are isosurfaces corresponding to α_0, α_1). The relative error is very low even though reconstruction artifacts come into play for smaller isovalues. Figure 33 (right) shows a similar experiment for a number of different isosurfaces and surface triangulations by marching cubes for the initial surfaces and two particular time steps. Again, volume variation is insignificant and confirms the volume preservation of our approach even for discretized marching cubes isosurfaces.

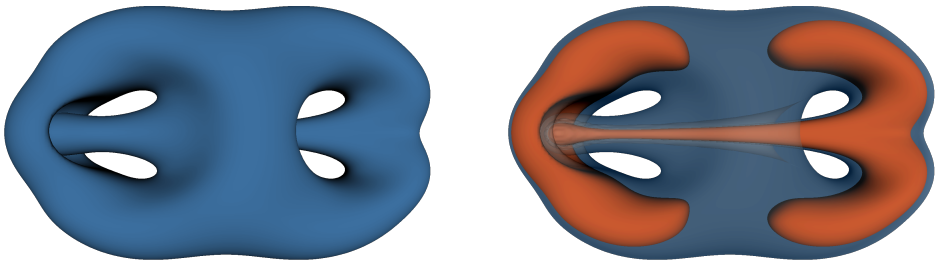


Figure 34: Topology Preservation. A double torus was deformed from its right to its left handle. The deformation preserves its genus-2 topology. At the same time the topology of an inscribed unconnected isosurfaces with lower isovalues (solid •) remains unchanged. (Figure 32 (d) shows a comparable closeup of the large isosurface (•) using exact reconstruction.)

RECONSTRUCTION ACCURACY. The static grid of the GPU implementation may lead to undersampling artifacts. For all our experiments, we could extract high-quality surfaces with correct topology and without any self-intersections. Figure 32 compares marching cubes to reconstruction with [BO05] and shows extreme configurations with hard to reconstruct regions of isosurfaces where interfaces are getting very close to each other. The volume variation of all shown high-quality reconstructions was always below 0.01% [1].

TOPOLOGY PRESERVATION. Figure 34 shows isosurfaces for an extreme deformation of a double torus scalar field. The isosurfaces have different topology at t_0 (a single genus-2 surface and two disconnected genus-0 surfaces are shown), and their topology is preserved over the whole deformation. In fact, topology preservation shown in Section 5.1.1 was confirmed by all our experiments (disregarding discretization artifacts from marching cubes).

PERFORMANCE. We implemented our approach using NVIDIA's CUDA GPU interface. All timings are measured for a NVIDIA GTX280 GPU with 1GB memory and an AMD Opteron processor at 2.6GHz.

$\bar{h}$ / N	Translation			Rotation			Interpolation	
	32	64	128	32	64	128	Linear	Cubic
10^3	0.96	1.36	2.16	0.87	1.21	1.81	0.01	0.01
10^4	1.55	2.41	4.62	1.87	2.15	3.64	0.01	0.02
10^5	7.42	13.74	26.41	6.12	11.16	21.42	0.03	0.04
10^6	64.65	125.28	247.71	48.55	92.63	180.97	0.04	0.05

1 The additional material folder `addmaterial/voldef` contains a video of the approach and high-quality reconstructions of a mesh shown in Figure 32.

The table summarizes integration and interpolation timings. Measured timings (in ms) are listed for N GPU path line integrations that require on average $\overline{h}$ integration steps each (left), and N scalar field interpolations (right). The processing time of cubic interpolation is negligible. Numerical path line integration is the most time consuming part of our algorithm. The average number of adaptive integration steps is 64 in all our examples. Typically, we observe numbers of seven (Figure 32 (a)) up to 250 (Figure 35 (right)) time steps if using intermediate scalar fields. This confirms that even a numbers of 10^6 path lines can be handled efficiently. The main reason is that no time consuming vector field lookup into, e.g., texture space, is required as our vector fields are parameterized and evaluated in closed-form. Finally, we remark that a full backward integration is more expensive and corresponds to a sum over all time steps back to t_0. For most of our examples this is in the order of 1000 steps in total. A single marching cubes surface reconstruction required $14ms$ on average, depending on the number of occupied voxels.

High quality reconstruction is an off-line process. Reconstruction time for the double torus and torus models in Figure 35 was 656 seconds and 843 seconds, respectively. Reconstruction of the Medusa head required 24 minutes.

5.4 APPLICATIONS

Our method can be applied to the following applications. Please also see the accompanying video of [MRT10] for additional interactive results [1].

DESIGNING SMOOTH SURFACES FROM SCRATCH. As a deformation approach our method can be used for general modeling of smooth isosurfaces starting from a smooth field, e.g., representing a family of spheres. The volume preservation property mimics plasticine-like materials in a plausible way and leads intuitive editing. In addition, constant topology is guaranteed for correct results. As a remarkable feature these two properties are satisfied not only for a single isosurface of interest but for all isosurfaces, i.e., the deformation acts on a family of surfaces. Therefore, a whole family of isosurfaces can be modeled simultaneously. Figure 35 shows examples that were modeled in $3-15$ minutes by an inexperienced user.

INTERACTIVE DEFORMATION OF VOLUME DATA. Volume-preserving deformations of a single isosurface result in modifications of all isosurfaces within the inner and blended region fields. Figure 36 shows a deformation of the bonsai volume data set, where the tool was swept to deform the trunk and its neighborhood.

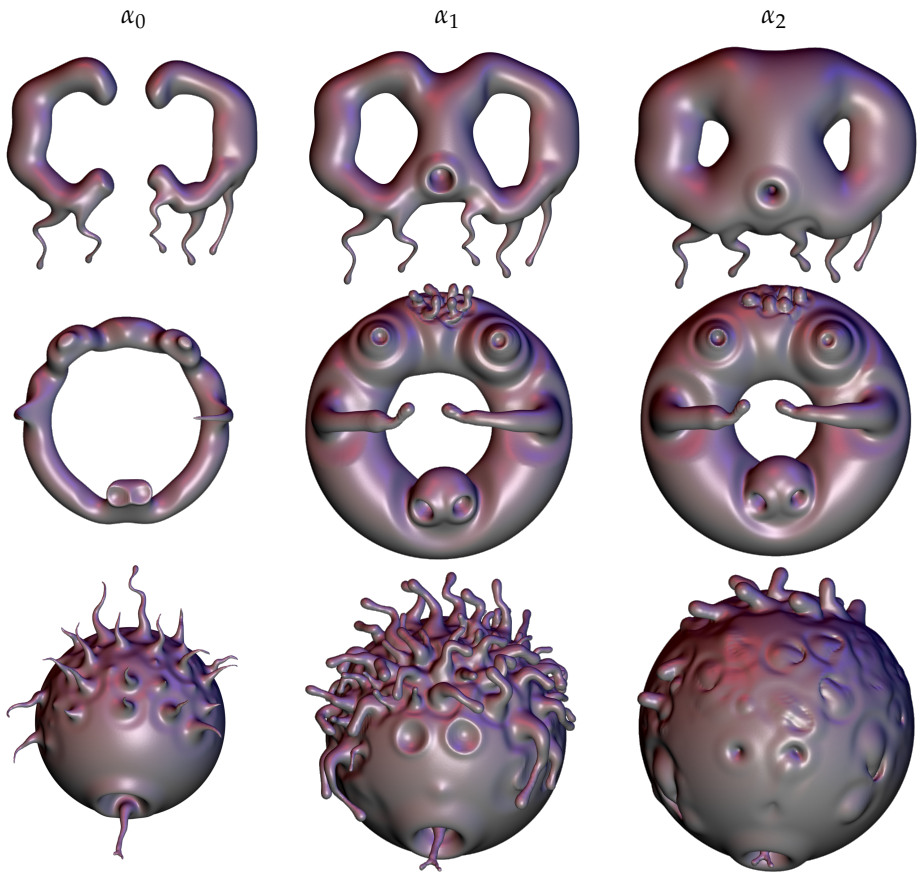


Figure 35: Faces of Different Topology. The images show isosurfaces of three modified scalar fields with $\alpha_i < \alpha_{i+1}$: *all* isosurfaces are deformed simultaneously while their volume and topology are preserved.

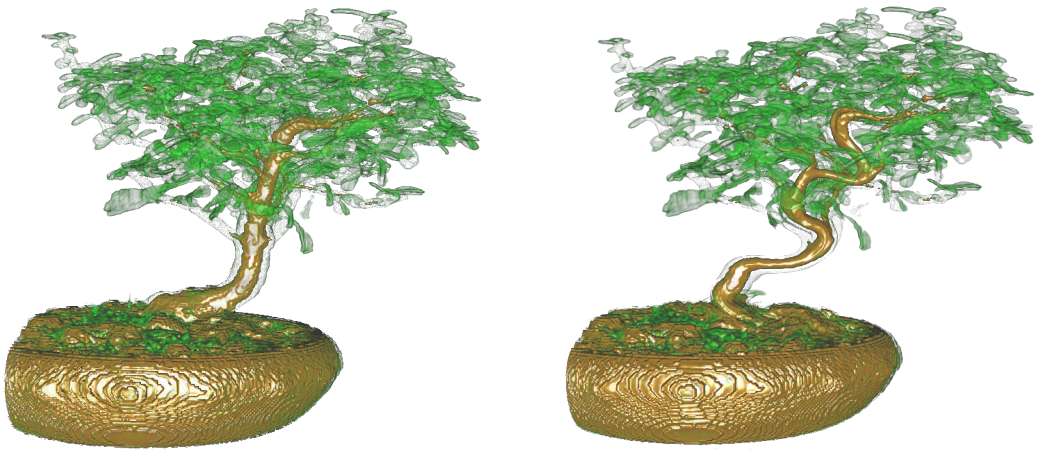


Figure 36: Volume Data Deformation. The volume rendered original bonsai data set (left) is deformed near its trunk (●) (right). Nearby leafs (●) are deformed in a volume-preserving way.

5.5 DISCUSSION

We continue to give a comparison of our approach to existing ones.

There are a number of approaches for deforming implicit surfaces, among them some aiming at the preservation of the volume inside a particular isosurface [CA06]. However, to the best of our knowledge, our approach is the first one that preserves the volume of *all* isosurfaces. Also, topology preservation of all isosurfaces is not addressed so far.

In comparison to other real-time volume deformation techniques [RSSG01, WR01, SBH07] our approach allows for stronger and at the same time more localized deformations. Also, volume and topology preservation is not addressed there.

The approaches closest to us are the methods by von Funck et al. [vFTS06, vFTS07b], who define deformations by integration along similar vector fields. Contrary to us, their methods only work on a single explicit surface, and a forward integration of mesh vertices is used. Resampling of the deformed meshes has to be accounted for explicitly in a post-process.

In volume modeling, a popular classification of approaches distinguishes physically-based and non-physically-based methods. Our approach lies between these classes: although it is not explicitly physically-based, volume preservation is a physically justified property for many materials.

In comparison to well-established integration techniques for level sets, our approach applies a backward Lagrangian integration scheme that cannot be applied to general level sets due to the dependencies of the level sets and the steering field.

LIMITATIONS. Although volume preservation appears to be a natural condition for plausible deformations, there are applications for which volume preservation does not hold. For example, the growing of a tumor in a medical volume data set cannot be addressed by our approach. Also, as we consider the whole volume, well-established isosurface deformation tools, which focus on a particular surface, do not fit into our approach. This includes sculpting techniques like cutting or carving [PF01] as well as volumetric copy and paste techniques [MBWB02].

Since our technique only incorporates the volume of the isosurfaces, metric distortions of an isosurface are not addressed. Moreover, analytically represented implicit surfaces cannot be deformed exactly by the integration scheme without discretizing the defining function.

The limitations mentioned above are inherent to our approach. In addition, there are algorithmic limitations that can be addressed in future research: the fixed

grid resolution of the volume data is a compromise between interactive performance and accuracy and is limited by GPU memory. This also naturally limits the spacial extents of the domain, and artifacts may appear for isosurfaces near grid boundaries unless special care is taken. Additionally, due to reconstruction artifacts, topology preservation may appear corrupted for real-time visualization. Hardware accelerated adaptive grids are a candidate for improvements here. Finally, more advanced deformation tools can be developed. In principle, different choices of the fields e, f, r that stay independent of the underlying scalar field allow for customized tools, but this depends on and has to be set for different application scenarios.

5.6 SUMMARY

In this chapter, we introduced a continuous deformation method for implicitly defined surfaces. Inside of a volume data set *all* isosurfaces in the vicinity of the deformation tool are deformed. The deformation guarantees the preservation of volume, continuity, and topology of each isosurface. For their computation we introduced an efficient and accurate backward Lagrangian integration scheme that is executed on the GPU for interactive modeling results. High-quality surfaces are extracted in a post-process. Our approach can be used to deform measured or simulated volume data sets as well as for the design of complex implicitly defined shapes with a pre-defined topology.

6

POSE CORRECTION BY SPACE-TIME INTEGRATION

The previous chapters motivated that the deformation of a shape from a given reference pose to a variety of new poses is a central task in computer graphics, geometric modeling, and computer animation. In contrast to the proposed continuous shape deformations, several recent approaches incorporate training data in the form of example poses to satisfy the demand for ever more realistic deformations without significantly increasing computation or simulation time: given a rest shape and several example poses, new shape variations are created by custom-tailored interpolation or extrapolation within a suitable shape space [KMP07]. Example-based approaches have been proposed, e.g., for shape interpolation [XZWB06, LSLCO05, KG08, WDAH10, FB11], skeletal skinning [LCF00, WPP07, WSLG07], shape deformations [SZGP05, FKY08, PJS06], as well as physical simulations [FYK10].

The reference model typically has been carefully designed and/or acquired from a physical object, often requiring a lot of manual work to check for and remove geometric inconsistencies, such as degenerate triangles, fold-overs, or self-intersections. The example poses are then created by deforming the rest pose and fine-tuning the result. For this purpose the deformation methods from the previous chapters can be used. In most cases, the rest pose and the example poses share the same structure, i.e., the same mesh connectivity, which we assume to be mandatory in the following.

Unfortunately, in practice many example poses suffer from geometric artifacts – even if the rest pose is a clean mesh. Oftentimes the degeneracies are introduced when deforming the rest pose into the target poses, e.g., by linear blend skinning [MTLT88, LCF00]. For similar reasons, many models obtained from shape-databases also contain geometric inconsistencies. These artifacts are then reproduced (sometimes even amplified) in example-based mesh processing techniques. They do not only corrupt the visual appearance, they may even cause severe numerical problems for more sophisticated nonlinear approaches [FB11], where they slow down or even spoil convergence of optimization schemes.

In this chapter, we apply integration-based continuous deformations in an automatic post-processing technique for detecting and resolving geometric incon-

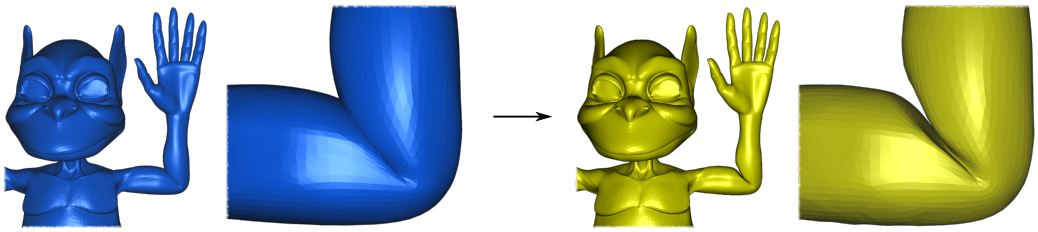


Figure 37: Elbow Deformation Repair. Self-penetrations of an articulated target pose (left •) are corrected by our approach locally *without* altering the connectivity of the underlying mesh (right •).

sistencies in given example poses. In contrast to the previous chapters, deformations are not user-specified but defined by pairs of rest and target poses. In particular, we use vector field-based shape integration because it guarantees the avoidance of, e.g., self-intersections in the target pose and the preservation of mesh connectivity.

Based on a linear (hence robust) mesh interpolation technique, we first generate a set of intermediate key-frames between rest pose and target pose. From those we derive smooth vertex trajectories, to which we fit a $4d$ space-time vector field. Integrating this vector field through time deforms the rest pose mesh into the target shape and corresponds to path line integration of each point of the rest pose. Since path lines never cross each other in space-time [TWHS05], the resulting mesh is free of self-intersections by construction. Regions that initially contained self-intersections are repaired, the remaining parts are faithfully reproduced. Figure 37 shows an example of a repaired mesh.

6.1 BACKGROUND

In this chapter, our general scope is mesh repair, and in our approach we apply vector field-based shape deformations and scattered data interpolation by radial basis functions.

MESH REPAIR. There is a variety of methods for repairing inconsistencies in geometric models like holes, self-intersections, folds, and topological noise in different types of geometric models. Methods can roughly be classified into surface-based methods and volumetric methods (cf. the recent survey by Ju [Ju09]).

Surface-based methods modify the surface meshes directly. In an early work, Bohn [Boh93] proposes a topological shell-closure operator that guarantees orientable surfaces. In order to also repair geometric inconsistencies, Barequet et al. [BDK98] employ heuristics for solving ambiguous geometric configurations. A frequent problem for scanned objects is topological noise. Guskov and

Wood [GW01] remove small handles by a local wave-front mesh analysis and non-separating cuts.

Volumetric methods are based on implicit surface models that guarantee surfaces without self-intersections. For instance, Nooruddin and Turk [NT03] use voxelization to obtain an implicit surface, the repaired surface is generated from iso-contours. Space partitions can lead to efficient algorithms. In [Ju04] an octree space voxelizations of an in/out volume classification is used for feature adaptive mesh repair. Bischoff et al. [BPK05] present an octree-based mesh repair framework which provides high accuracy. A follow-up [BK05a] focuses on locality such that mesh connectivity is modified only locally in the vicinity of inconsistencies. More recently, nested space partitions have been proposed by Campen and Kobbelt [CK10] for model repair.

All these methods operate on a single surface, and all of them modify the internal structure, e.g., the mesh connectivity. In contrast to this, the input to our method consists of poses of the same mesh with fixed connectivity. Only the geometry of the meshes is corrected by a *deformation* approach: our algorithm does not modify mesh connectivity. We remark that there are deformation approaches that suppress intersections directly in the modeling process [BK03] or in a physical simulation of cloths [BWK03]. However, these approaches aim at interactive or physically plausible shape deformation rather than automatic model repair as a post-process.

VECTOR FIELD-BASED DEFORMATIONS. Our approach can be considered as a deformation method that exploits special properties of an underlying continuous deformation vector field. The vector field-based formulation guarantees the deformations to prevent self-intersections. In contrast to the continuous deformation methods of the previous chapters, this vector field is determined automatically by the input poses. We refer to the overviews and references on continuous vector field-based deformations of the previous chapters for more details.

INTERPOLATION BY RADIAL BASIS FUNCTIONS. We apply radial basis functions (RBFs) as a model for space-time vector fields. RBFs are a general and widely used tool for mesh-less scattered data interpolation [Wen04]. They are used in numerous applications on different types of data. Examples related to geometry processing include surface-fitting based on point samples [MYC*01, CBC*01, OBS05], linear elasticity-based simulations [MKB*08] as well as real-time surface deformations [BK05b].

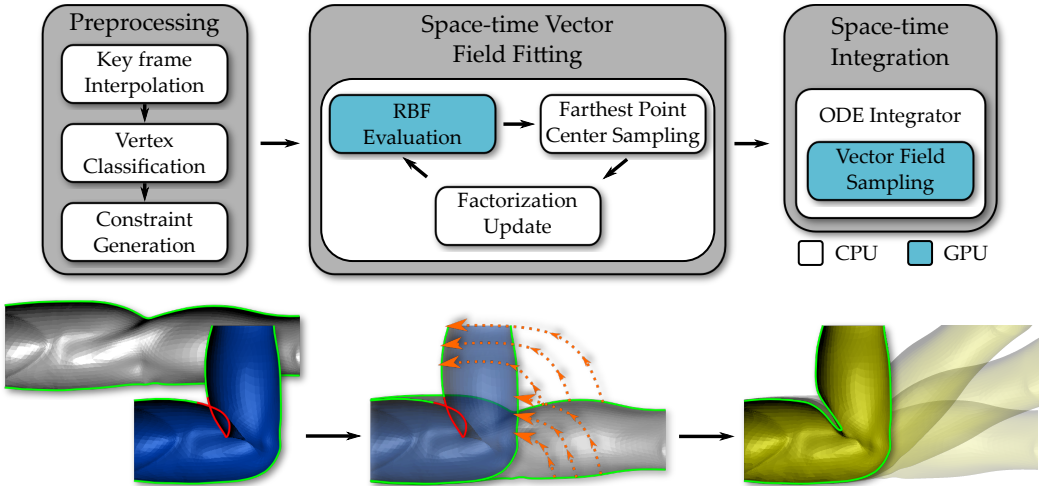


Figure 38: Computational Pipeline. Rest pose (●) and target pose (●) are analyzed for uncritical (●) and critical regions (●), from which space-time constraints are computed. Then a space-time vector field (●) that describes the deformation is fitted using RBFs. Space-time shape integration in this field yields the repaired pose (●). Steps highlighted in ● use GPU hardware acceleration.

6.2 GENERAL IDEA AND OVERVIEW

This section gives an overview of our method, which is also illustrated in Figure 38. Details on the particular steps are provided in the subsequent Section 6.3.

Given are two poses of a surface mesh, a reference mesh and a target mesh, which both share the same connectivity. The target shows defects such as self-intersections, and our goal is to remove defects and to repair the mesh.

We propose a deformation approach to repair the target pose. The deformation is represented by a space-time vector field integration, and we exploit the fact that path lines of integrated surface points do not intersect in space-time (see, e.g., [TWHS05]). This way we obtain surfaces that approximate the given targets and by construction are free of self-intersections. Similar to the isosurface deformations of Chapter 5, the time-dependent vector field will be defined globally in the whole embedding space of the surfaces and is not restricted onto the shape itself as in Chapters 3 and 4.

We want to find these continuous time-dependent vector fields such that they describe the motion of the surfaces under deformation from the reference poses to the target poses. Our approach is based on fitting a smooth vector field to a set of discrete samples. In our case, these samples are tangent vectors of the trajectories of surface points moving in space-time. We obtain the tangent vector samples by first generating a sequence of *key frames* that represent some intermediate surfaces. Then we fit trajectories as C^1-continuous cubic spline

curves at each vertex to obtain continuous curves that describe the motion of each vertex.

Large parts of the target surface do not require any repair or modification. Our goal is to reproduce such regions as good as possible, whereas regions that contain defects in the target pose should be repaired. We achieve this by an automatic *classification* of surface regions. The reproduction of well-behaved surface regions is due to constraints on the vector field fitting. The key idea is that regions with defects do not generate any constraints on the vector field: we use the extrapolation provided by the underlying RBF interpolation model to fill up the vector field smoothly and in an energy minimizing way in the space-time domain where no constraints are available. This can roughly be compared to hole filling in applications to surface reconstruction where RBF-based interpolation is also used successfully (see, e.g., [CBC*01]).

The vector field is represented as a radial basis function over the space-time domain (see, e.g., [TWHS05]). The function interpolates the constraints given by key frames and vertex classification. Then *fitting* the vector field to these constraints boils down to solving a (dense) linear system.

Finally, we *integrate* all vertices of the reference pose through the vector field to obtain the corrected result pose. The integration exploits the property that path lines never cross each other in space-time to guarantee intersection-free time-surfaces.

6.3 ALGORITHMIC DETAILS

Our approach proceeds in three steps: generation of space-time constraints, vector field fitting, and integration. This is illustrated in Figure 38. In this section, we describe the algorithmic steps in detail.

We consider meshes $\mathcal{M}_k$ that share the same connectivity: the surfaces are defined by vertex positions $\mathbf{x}_i \in \mathbb{R}^3, i \in \mathcal{V}$, where $\mathcal{V}$ is the set of vertices. Let $\mathcal{M}_R$ be the reference pose, and $\mathcal{M}_T$, $T \neq R$ is one of the target poses. We assume that the reference $\mathcal{M}_R$ does not contain defects, and that the target poses should be repaired. Note that our goal is to maintain a consistent triangulation over all poses, i.e., the connectivity remains fixed. This is in contrast to other methods that focus on repairing a single mesh like [CK10].

6.3.1 Preprocessing

Our algorithm works on the reference pose $\mathcal{M}_R$ and one single target pose $\mathcal{M}_T$ at a time. It does not require or use data of other target poses. The input

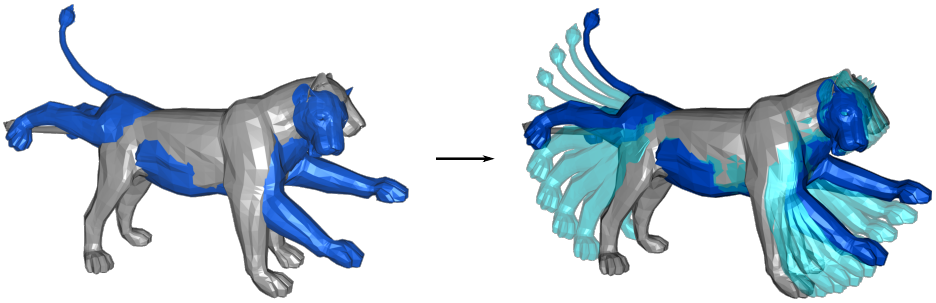


Figure 39: Preprocessing: Key Frame Interpolation. From the aligned reference (•) and one target shape (•) we generate a dense set of intermediate key frame shapes by linear Poisson-based interpolation (transparent •).

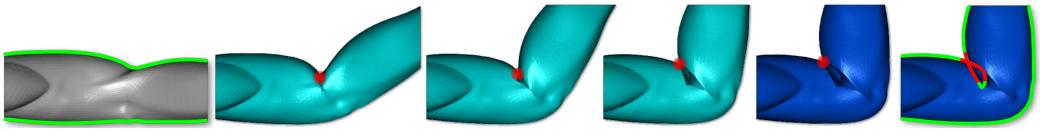


Figure 40: Preprocessing: Vertex Classification. Vertices at self-intersections (• dots) are detected in the intermediate key frames (•) and removed from the set of constrained vertices $\mathcal{V}^{\mathcal{C}}$ (•).

is a pair of meshes $(\mathcal{M}_R, \mathcal{M}_T)$. The preprocessing step consists of key frame interpolation, intersection detection, and constraint generation.

KEY FRAME INTERPOLATION. Our shape interpolation is based on Poisson interpolation in gradient space (see [XZWB06]), because this linear method faithfully reproduces even artifact-tainted poses. However, we could also use any other method for key frame generation (see, e.g., [SP04, KG08, WDAH10]). Note that the key frames may contain defects and artifacts. In fact, if the target pose contains defective regions, the intermediate key frames are likely to have defects, too. Figure 39 shows an example for key frame shapes. Every surface refers to a time step t_k.

VERTEX CLASSIFICATION. We determine the constrained surface points $\mathcal{V}^{\mathcal{C}} \subset \mathcal{V}$ that refer to well-behaved regions as follows: initially, all surface parts are considered well-behaved, and we initialize $\mathcal{V}^{\mathcal{C}} = \mathcal{V}$. Then we generate a number of key frame surfaces and scan these meshes for self-intersections. (We used 11 key frames in all our examples.) We need to detect self-intersections in all intermediate key frames — not only in the target mesh — to capture the global support of evolving self-intersections. The intersection tests are performed efficiently by using a space hierarchy of axis-aligned bounding boxes that enclose the triangles and exact geometric predicates, which are both provided by the CGAL library [3]. Whenever we detect self-intersecting triangles, their vertices and all vertices in their one-ring neighborhood are removed from $\mathcal{V}^{\mathcal{C}}$. Our experiments show that

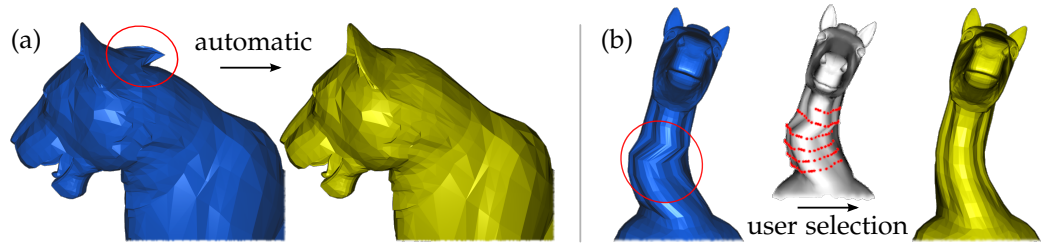


Figure 41: Repair of Common Skinning Artifacts. (a) Badly chosen skinning weights lead to intersection artifacts at the highlighted regions of neighboring skeleton joints (• mesh), which are automatically detected and repaired by our method (• mesh). (b) Other types of skinning artifacts can also be repaired, but they require manual selection (• vertices). Note that the original skinning skeleton is not required for any correction.

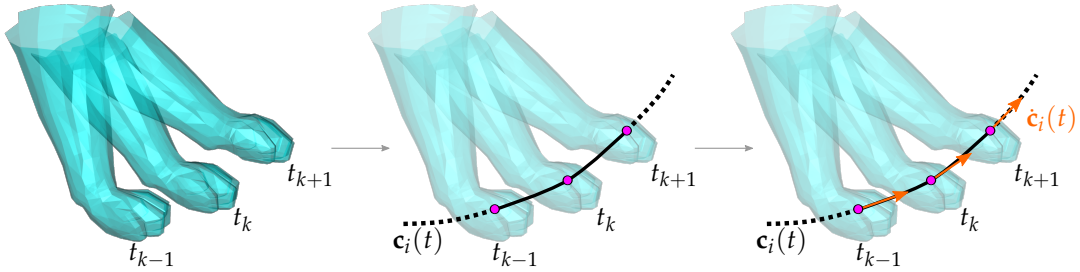


Figure 42: Preprocess: Constraint Sampling. Through each vertex (•) of the key frame shapes at time t_k we fit C^1-continuous splines (•). The tangents to these curves (•) are then used together with the vertex coordinates as space-time constraints for vector field fitting.

the removal of one additional ring of vertices works well in practice. Removing a larger neighborhood can lead to smoother transitions between regions. However, there is a trade-off between smoothness and reproduction. Figure 40 illustrates the process. We emphasize that the classification can be modified manually to correct other types of artifacts that are not detected automatically (see Figure 41 (b) for an example related to skinning artifacts).

SPACE-TIME CONSTRAINT GENERATION. The key frames define trajectories of vertices moving in space-time. For vertex positions $\mathbf{x}_i(t_k), i \in \mathcal{V}$, we fit cubic C^1-continuous spline curves $\mathbf{c}_i$ such that $\mathbf{c}_i(t_k) = \mathbf{x}_i(t_k)$. We use Hermite interpolation with tangent directions approximated by finite differences w.r.t. time. The trajectories $\mathbf{c}_i$ can be evaluated at arbitrary times $t \in [0,1]$, where $\mathbf{c}_i(0)$ and $\mathbf{c}_i(1)$ evaluate to points on the reference and the target surface, respectively. For vector field fitting we require sampling of positions $\mathbf{c}_i(t)$ to place RBF centers in space-time, and sampling of tangents $\dot{\mathbf{c}}(t) := \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{c}(t)$ to evaluate interpolation constraints. Figure 42 illustrates this constraint sampling.

The space-time deformation vector field $\mathbf{u}$ is a $3d$ function on a $4d$ space-time domain: $\mathbf{u}(\mathbf{y})\colon \mathbb{R}^4 \to \mathbb{R}^3$. Given the vertex classification $\mathcal{V}^{\mathcal{C}}$ and smooth trajectories $\mathbf{c}_i$, we can define point-wise interpolation constraints for the vector field function. Let $\mathbf{y}_i(t_k) = (\mathbf{c}_i(t_k), t_k)^{\mathrm{T}} \in \mathbb{R}^4$ be a point in space-time. Then we have the interpolation conditions $\mathbf{u}(\mathbf{y}_i(t_k)) \stackrel{!}{=} \dot{\mathbf{c}}_i(t_k)$. We denote the set of interpolation constraints $\mathcal{C}$ as

$$\mathcal{C} = \left\{ (\mathbf{y}_i(t_k), \dot{\mathbf{c}}_i(t_k)) \;\middle|\; i \in \mathcal{V}^{\mathcal{C}},\ t_k = \tfrac{k}{s},\ k = 0, \ldots, s \right\} .$$

$\dot{\mathbf{c}}_i(t)$ are the tangents to trajectories of vertices $i \in \mathcal{V}^{\mathcal{C}}$. We used $s = 12$ for all our examples. To weight the spatial and temporal dimensions equally we scale all models to the unit sphere before processing.

6.3.2 Space-Time Vector Field Fitting

Computing the space-time vector field that interpolates all constraints in $\mathcal{C}$ is the most crucial step in our method. The quality of the vector field is directly related to the quality of the deformed shapes. Furthermore, fitting of a continuous space-time vector field that is smooth and well-behaved between constrained points dominates the runtime of our approach.

RBF INTERPOLATION. We represent the space-time vector field $\mathbf{u}$ as a sum of radial basis functions (RBFs) that interpolate the constraints: $\mathbf{u}(\mathbf{y}_i(t_k)) \stackrel{!}{=} \dot{\mathbf{c}}_i(t_k)$. Our four-variate three-dimensional RBF model $\mathbf{u}\colon \mathbb{R}^4 \to \mathbb{R}^3$ is defined by a set of centers $\gamma_j \in \mathbb{R}^4$ and associated weights $\mathbf{w}_j \in \mathbb{R}^3$:

$$\mathbf{u}(\mathbf{y}) = \sum_j \mathbf{w}_j \, \phi_j(\mathbf{y}) + \boldsymbol{\pi}(\mathbf{y}) .$$

The function $\phi_j(\mathbf{y}) = \phi\big(\|\gamma_j - \mathbf{y}\|\big)$ is the basis function [1] corresponding to the jth center γ_j. We employ the triharmonic kernel for even dimensions $\phi(r) = r^2 \ln(r)$ in combination with a four-variate quadratic polynomial $\boldsymbol{\pi}(\mathbf{y}) \in \Pi_4^2$, since then the resulting function $\mathbf{u}$ minimizes spline-like fairness energies and provides desirable extrapolation properties [Wen04].

In order to interpolate n constraints $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]^{\mathrm{T}} \in \mathbb{R}^{n\times 3}$ (corresponding to $\dot{\mathbf{c}}_i(t_k)$), we place the centers $\gamma_1, \ldots, \gamma_n$ at the constrained positions (corresponding to $\mathbf{y}_i(t_k)$). We solve the corresponding linear system for the coefficients of the basis functions $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_n]^{\mathrm{T}} \in \mathbb{R}^{n\times 3}$ and the quadratic polynomial $\mathbf{U} \in \mathbb{R}^{15\times 3}$:

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\Pi} \\ \boldsymbol{\Pi}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{W} \\ \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{V} \\ \mathbf{0} \end{bmatrix} .$$

1 The RBF basis functions are often called *basic* functions in the literature [Wen04]. We do not make this distinction in this work.

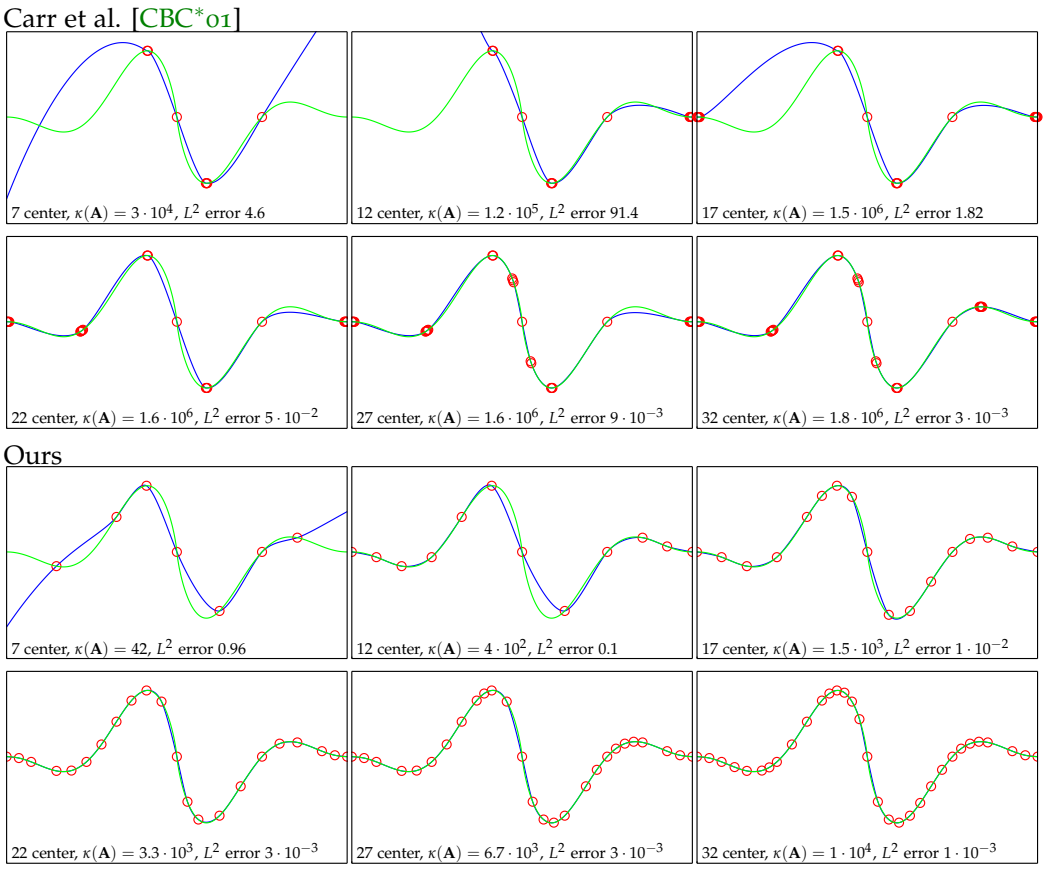


Figure 43: RBF Center Selection. A test function (●) is approximated by an iterative RBF (●) refinement using the center (●) selection strategies of [CBC*01] and our extension. In each step $m = 5$ new centers are selected. No center clusters are generated by our approach. We achieve the same total L^2 approximation accuracy with less centers (22 vs. 32) and a significantly better conditioning κ of the linear system $\mathbf{A}$.

The blocks $\mathbf{\Phi} \in \mathbb{R}^{n \times n}$ and $\mathbf{\Pi} \in \mathbb{R}^{n \times 15}$ are given by $\Phi_{ij} = \phi_j(\gamma_i)$ and $\Pi_{ij} = \pi_j(\gamma_i)$, and $\{\pi_1, \ldots, \pi_{15}\}$ is a basis of the function space Π_4^2. The last 15 equations augment the system with the orthogonality condition $\mathbf{\Pi}^T \mathbf{W} = \mathbf{0}$ (see [CBC*01]). This linear system is dense because of the global support of $\phi(r)$.

RBF CENTER SELECTION. Due to the density of the linear system we are limited to a few thousand centers in practice. Therefore, the selection of centers plays a key role in RBF interpolation. We propose an iterative strategy for center selection, which is an extension of the strategy used in [CBC*01]. The idea is to satisfy a prescribed approximation accuracy ϵ by carefully selecting a subset of interpolation constraints, which requires a certain selection strategy.

Center selection is an iterative process. We start with the two most distant space-time points in $\mathcal{C}$ as the initial set of RBF centers. In each iteration we evaluate

the fitting error, and we stop if the maximum error drops below a threshold ϵ, i.e., if

$$\max_{(\mathbf{y},\mathbf{v})\in\mathcal{C}} \|\mathbf{u}(\mathbf{y}) - \mathbf{v}\| < \epsilon \,.$$

In each iteration, the selection strategy by Carr et al. [CBC*01] consists of enlarging the set of centers by the m space-time points with highest errors. It turns out that this scheme leads to badly conditioned linear systems and requires a high number of total centers for convergence. Instead, we extend the original approach by adding a farthest point sampling step: we select the m most distant points within p candidate center points with highest error, where $p \gg m$. We use a kd-tree to partition the $4d$ space-time domain for efficient farthest point queries [1]. The rationale is to avoid generation of clusters of centers that would lead to ill-conditioned linear systems (see [BK05b]). Our selection strategy does not only respect the range of $\mathbf{u}$ given by the fitting errors, which are minimized, but also the geometry of the domain of $\mathbf{u}$ in form of pairwise center distances, which are maximized. It turns out that this strategy significantly reduces the number of centers that are required to achieve a prescribed error bound ϵ. Additionally, the numerical stability is increased significantly (see Section 6.4 for a quantitative analysis). For all examples we used $m = 128$, $p = 1024$, and a fitting accuracy of $\epsilon = 10^{-4} \cdot d$, where d is the length of the spatial bounding box diagonal of the reference pose. The algorithm terminated for all examples, i.e., the maximum error is reduced until it falls below ϵ.

A comparison of both selection strategies applied to the approximation of an univariate function is shown in Figure 43. Our method avoids center clusters, requires less centers for function approximation and is, at the same time, numerically more stable to compute. Note that in this example the total L^2 approximation error of the selection scheme by [CBC*01] does not even decrease monotonically (see the step from 7 to 12 centers). This is not the case for our method, which decreases errors steadily. Although we cannot guarantee this behavior, all our experiments indicate a similar monotonic error reduction property also for harder interpolation problems.

We remark that numerical stability and convergence rate of iterative space-time vector field fitting are additionally improved by the vertex classification, which excludes contradictory constraints.

BLOCKED FACTORIZATION UPDATE. For the factorization of the resulting linear system we exploit similarity of consecutive iterations and fuse center selection with the factorization of the system. This way, we obtain an efficient factorization update scheme.

In each iteration we update a $(n+15) \times (n+15)$ LU factorization with partial pivoting of the RBF system $\mathbf{A}$ that is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{\Phi}_{0,0} & \mathbf{\Pi}_0 \\ \mathbf{\Pi}_0^T & \mathbf{0} \end{bmatrix} = \mathbf{P}^T \mathbf{L} \mathbf{U} \mathbf{R} .$$

The system matrix $\mathbf{A}$ corresponds to the n selected centers of the previous iteration. The matrices $\mathbf{P}$ and $\mathbf{R}$ are permutations (in the first iteration $\mathbf{R} = \mathbf{I}$), $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular, respectively. Permutations and triangular factors are updated in each iteration.

The goal of factorization update is to reuse this known factorization for the efficient computation of the LU factorization of the new $(n+m+15) \times (n+m+15)$ RBF linear system $\mathbf{A}'$ that is extended by m new RBF centers. It has the form

$$\mathbf{A}' = \begin{bmatrix} \mathbf{\Phi}_{0,0} & \mathbf{\Phi}_{1,0} & \mathbf{\Pi}_0 \\ \mathbf{\Phi}_{1,0}^T & \mathbf{\Phi}_{1,1} & \mathbf{\Pi}_1 \\ \mathbf{\Pi}_0^T & \mathbf{\Pi}_1^T & \mathbf{0} \end{bmatrix} .$$

Blocks indexed by 1 are basis functions evaluated in the current iteration, whereas 0-indexed matrices are reused. We rewrite $\mathbf{A}'$ by first applying global row and column permutations $\mathbf{Q}$ such that we can reuse the factorization of $\mathbf{A}$:

$$\mathbf{A}' = \mathbf{Q}^T \begin{bmatrix} \mathbf{\Phi}_{0,0} & \mathbf{\Pi}_0 & \mathbf{\Phi}_{1,0} \\ \mathbf{\Pi}_0^T & \mathbf{0} & \mathbf{\Pi}_1^T \\ \mathbf{\Phi}_{1,0}^T & \mathbf{\Pi}_1 & \mathbf{\Phi}_{1,1} \end{bmatrix} \mathbf{Q} = \mathbf{Q}^T \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{\Phi}_{1,1} \end{bmatrix} \mathbf{Q} .$$

The permutation $\mathbf{Q}$ swaps the last two rows and columns of block matrices, respectively. We did also substitute the block matrix $\mathbf{C}^T = \begin{bmatrix} \mathbf{\Phi}_{1,0}^T & \mathbf{\Pi}_1 \end{bmatrix}$.

It is now possible to derive the updated LU factorization of $\mathbf{A}' = \mathbf{P'}^T \mathbf{L}' \mathbf{U}' \mathbf{R}'$ by using only identity transformations and the Schur complement $\mathbf{\Phi}_{1,1} - \mathbf{C}^T \mathbf{A}^{-1} \mathbf{C}$ of the center block of $\mathbf{A}'$ [8]:

$$\begin{aligned} \mathbf{A}' &= \mathbf{Q}^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}^T \mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{0} & \mathbf{\Phi}_{1,1} - \mathbf{C}^T \mathbf{A}^{-1} \mathbf{C} \end{bmatrix} \mathbf{Q} \\ &= \mathbf{Q}^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}^T \mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P}^T \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{L}^{-1} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{0} & \mathbf{\Phi}_{1,1} - \mathbf{C}^T \mathbf{A}^{-1} \mathbf{C} \end{bmatrix} \mathbf{Q} \\ &= \mathbf{Q}^T \begin{bmatrix} \mathbf{P}^T \mathbf{L} & \mathbf{0} \\ \mathbf{C}^T \mathbf{R}^T \mathbf{U}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U} \mathbf{R} & \mathbf{L}^{-1} \mathbf{P} \mathbf{C} \\ \mathbf{0} & \mathbf{\Phi}_{1,1} - \mathbf{C}^T \mathbf{A}^{-1} \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{R}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q} \\ &= \mathbf{Q}^T \begin{bmatrix} \mathbf{P}^T \mathbf{L} & \mathbf{0} \\ \mathbf{S} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{T} \\ \mathbf{0} & \tilde{\mathbf{P}}^T \tilde{\mathbf{L}} \tilde{\mathbf{U}} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q} \end{aligned} \tag{40}$$

$$
\begin{aligned}
&= \mathbf{Q}^\mathrm{T} \begin{bmatrix} \mathbf{P}^\mathrm{T}\mathbf{L} & \mathbf{0} \\ \mathbf{S} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}}^\mathrm{T}\tilde{\mathbf{L}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{L}}^{-1}\tilde{\mathbf{P}} \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{T} \\ \mathbf{0} & \tilde{\mathbf{P}}^\mathrm{T}\tilde{\mathbf{L}}\tilde{\mathbf{U}} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q} \\
&= \mathbf{Q}^\mathrm{T} \begin{bmatrix} \mathbf{P}^\mathrm{T}\mathbf{L} & \mathbf{0} \\ \mathbf{S} & \tilde{\mathbf{P}}^\mathrm{T}\tilde{\mathbf{L}} \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{T} \\ \mathbf{0} & \tilde{\mathbf{U}} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q} \\
&= \mathbf{Q}^\mathrm{T} \begin{bmatrix} \mathbf{P}^\mathrm{T} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}}^\mathrm{T} \end{bmatrix} \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}} \end{bmatrix} \begin{bmatrix} \mathbf{P}^\mathrm{T}\mathbf{L} & \mathbf{0} \\ \mathbf{S} & \tilde{\mathbf{P}}^\mathrm{T}\tilde{\mathbf{L}} \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{T} \\ \mathbf{0} & \tilde{\mathbf{U}} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q} \\
&= \underbrace{\mathbf{Q}^\mathrm{T} \begin{bmatrix} \mathbf{P}^\mathrm{T} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}}^\mathrm{T} \end{bmatrix}}_{\mathbf{P}'^\mathrm{T}} \underbrace{\begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \tilde{\mathbf{P}}\mathbf{S} & \tilde{\mathbf{L}} \end{bmatrix}}_{\mathbf{L}'} \underbrace{\begin{bmatrix} \mathbf{U} & \mathbf{T} \\ \mathbf{0} & \tilde{\mathbf{U}} \end{bmatrix}}_{\mathbf{U}'} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q}}_{\mathbf{R}'} .
\end{aligned}
$$

Hence, $\mathbf{L}'$ and $\mathbf{U}'$ are triangular, and $\mathbf{P}'$ and $\mathbf{R}'$ are permutations, as required for the LU factorization of $\mathbf{A}'$. We have made the substitutions $\mathbf{S} = \mathbf{C}^\mathrm{T}\mathbf{R}^\mathrm{T}\mathbf{U}^{-1}$ and $\mathbf{T} = \mathbf{L}^{-1}\mathbf{P}\mathbf{C}$, which require two triangular back-substitutions for the known triangular matrices $\mathbf{L}$ and $\mathbf{U}$. Additionally, we substituted the LU factorization with partial pivoting of the Schur complement $\mathbf{\Phi}_{1,1} - \mathbf{C}^\mathrm{T}\mathbf{A}^{-1}\mathbf{C} = \mathbf{\Phi}_{1,1} - \mathbf{S}\mathbf{T}$, which we denoted by $\tilde{\mathbf{P}}^\mathrm{T}\tilde{\mathbf{L}}\tilde{\mathbf{U}}$. The factorization of this small $m \times m$ matrix is the only costly numerical operation that has to be performed in every iteration. All other operations are multiplications of dense matrices, which can be performed efficiently and parallelized on, e.g., modern CPUs [6] or GPUs [7].

We note that this LU-based factorization update scheme does not take advantage of the symmetry of $\mathbf{A}$ and $\mathbf{A}'$ In fact, the symmetry of the system can be exploited by instead updating a cheaper to compute LDL factorization with partial pivoting $\mathbf{A} = \mathbf{P}^\mathrm{T}\mathbf{L}\mathbf{D}\mathbf{L}^\mathrm{T}\mathbf{P}$ [GVL96]: a derivation similar to (40) yields the update scheme

$$
\mathbf{A}' = \underbrace{\mathbf{Q}^\mathrm{T} \begin{bmatrix} \mathbf{P}^\mathrm{T} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}}^\mathrm{T} \end{bmatrix}}_{\mathbf{P}'^\mathrm{T}} \underbrace{\begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \tilde{\mathbf{P}}\mathbf{S} & \tilde{\mathbf{L}} \end{bmatrix}}_{\mathbf{L}'} \underbrace{\begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{D}} \end{bmatrix}}_{\mathbf{D}'} \underbrace{\begin{bmatrix} \mathbf{L}^\mathrm{T} & \mathbf{S}^\mathrm{T}\tilde{\mathbf{P}}^\mathrm{T} \\ \mathbf{0} & \tilde{\mathbf{L}}^\mathrm{T} \end{bmatrix}}_{\mathbf{L}'^\mathrm{T}} \underbrace{\begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{P}} \end{bmatrix} \mathbf{Q}}_{\mathbf{P}'} \tag{41}
$$

for the updated factorization $\mathbf{A}' = \mathbf{P}'^\mathrm{T}\mathbf{L}'\mathbf{D}'\mathbf{L}'^\mathrm{T}\mathbf{P}'$. Here, a single back-substitution for $\mathbf{S} = \mathbf{C}^\mathrm{T}\mathbf{P}^\mathrm{T}\mathbf{L}^{-\mathrm{T}}\mathbf{D}^{-1}$ has to be performed together with a small $m \times m$ LDL factorization $\tilde{\mathbf{P}}^\mathrm{T}\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^\mathrm{T}\tilde{\mathbf{P}} = \mathbf{\Phi}_{1,1} - \mathbf{S}\mathbf{D}\mathbf{S}^\mathrm{T}$. As LDL factorizations are half as expensive to compute in terms of computational and storage complexity compared to LU factorizations [GVL96] this scheme should generally be preferred. Still, all experiments in this chapter were performed using the previously derived LU factorization.

Note that our fused update and factorization scheme for RBF interpolation is not restricted to our particular problem of vector field fitting. In fact, it can be applied to *any* scattered data approximation problem. Hence, a large number of applications can potentially benefit from our method, too.

6.3.3 Space-Time Shape Integration

The RBF interpolation defines a smooth vector field $\mathbf{u}$ in space-time. Interpolation of space-time constraints ensures both spatial and temporal continuity of $\mathbf{u}$. We integrate this vector field for all vertices of the reference surface starting at time $t = 0$, and we expect an approximation of the target surface at $t = 1$. The idea is to reconstruct the trajectories $\mathbf{c}_i(t)$ of vertices i with path lines in $\mathbf{u}$ starting at $\mathbf{x}_i(0)$. For well-behaved regions, the path lines are close to the original trajectories, and integration ends near $\mathbf{x}_i(1)$. However, as path lines do not intersect at specific point in time, self-intersections cannot be reproduced – instead they are avoided and hence repaired in the final shape.

To show that path lines do not intersect at a specific point in time we lift $\mathbf{u}$ to the $4d$ space-time vector field $\bar{\mathbf{u}} = (\mathbf{u}, 1)^{\mathrm{T}}$, in which the last component represents the time differential. Then path lines of $\mathbf{u}$ are represented as the tangent curves (stream lines) of $\bar{\mathbf{u}}$ after re-projection into ordinary space. As only a single tangent vector is defined in each space-time point, the tangent curves of $\bar{\mathbf{u}}$ cannot intersect each other (except in critical points, which do not exist in $\bar{\mathbf{u}}$ because $\|\bar{\mathbf{u}}\| > 0$), and the ordinary path lines of $\mathbf{u}$ are also guaranteed to be intersection-free at a specific point in time.

For the numerical ODE integration we use a standard fifth-order Runge-Kutta integrator with adaptive step size control (see, e.g., [PTVF07]). This integrator performed well in all our experiments as the fitted vector fields turned out to be very smooth. The computation cost is dominated by evaluation of the RBF. To speed up computations we apply a parallelized implementation on the GPU [5] for basis function evaluation. Compared to a parallelized CPU implementation on a quad-core processor, we obtain a speedup of 320% in double-precision compute mode. We apply the GPU-based RBF basis function evaluation similarly for setting up the linear systems.

We remark that by using global vector field integration our approach can only remove self-intersections that are not already present in the reference pose. Yet these do not impede our method otherwise: after vertex classification we simply re-add the corresponding vertices of the reference pose to $\mathcal{V}^{\mathcal{C}}$. Indeed, this way initial self-intersection are reproduced with no effects on the remainder of our method.

6.4 ANALYSIS AND RESULTS

In this section we analyze properties of our approach, discuss its implementation, and present results. The video accompanying the corresponding publication

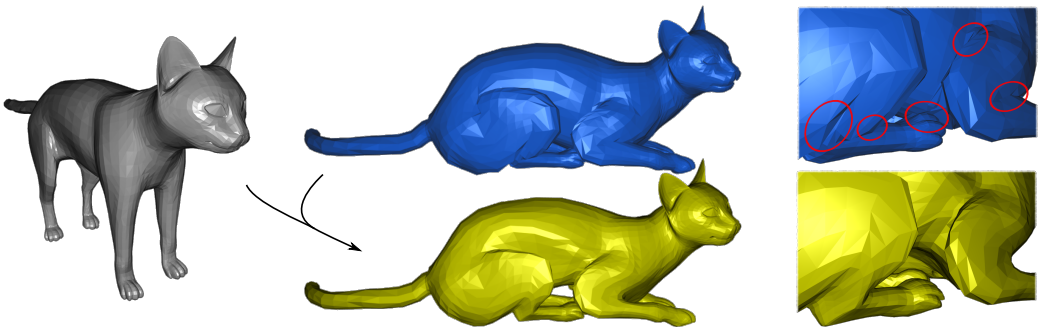


Figure 44: Shape Reproduction. Integrated from the reference pose (●, left) the shape of the target CAT model (●) is correctly reproduced in the non-intersecting regions of the head, neck, back and tail (●, middle). The close-ups show the critical regions (●) corrected by our method (right).

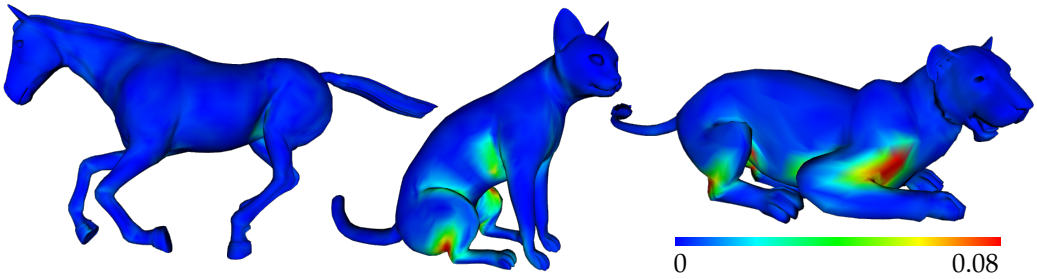


Figure 45: Geometric Differences. Shown are corrected surfaces generated by our method. The color code indicates the euclidean distance to the given target shapes. (Models were scaled to the unit sphere to enable comparison). Shapes differ significantly only in region with defects, which are repaired by our method. The experiment confirms accurate reproduction and smooth transitions to repaired regions.

[MRF*11] shows additional deformation results and examples of continuously deforming time-integrated shapes [2].

TARGET SHAPE REPRODUCTION. Our goal is to repair defects. However, regions of the target mesh without any defects should be reproduced. Our experiments show that our method introduces geometric differences only locally at vertices $i \notin \mathcal{V}^C$. Figure 44 shows an example were the overall shape of the target cat pose is maintained while all self-intersection are corrected. The example in Figure 45 shows color coded local geometric differences of some target models and our repaired models. The regions that show high geometric differences in the visualizations are regions that have been repaired. All other regions are reproduced accurately.

2 The video is located in the additional material folder `addmaterial/pcorr`. Additionally, this folder contains pose correction results of the LION, CAT, and HORSE models, respectively.

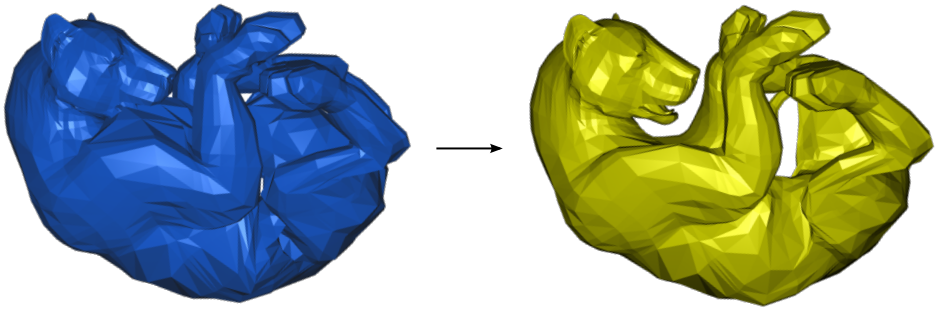


Figure 46: Resolving Complex Self-Intersections. We compare the given target mesh (•) to the repaired mesh (•). All self-intersections that were not already present in the reference pose have been successfully removed. The overall shape is maintained.

Figure 47: Relocation of Intersecting Triangles. Corrected results where triangles in • indicate the locations of intersecting triangles of the initial target pose. If possible, relocation distance is minimal (left) while larger relocations are required for deeper initial penetration distances (right) (see also Figure 46).

RESOLVING INTERSECTIONS. Our continuous deformations of shapes that are guided by a smooth space-time vector field cannot introduce new intersections of the shape during space-time integration. The reason for this is that path lines, trajectories of surface points in space-time, do not intersect. In all our experiments there are no newly introduced self-intersections. Figure 46 shows an example with non-trivial self-intersections. The left target shape contains 523 self-intersecting triangles. In contrast, our result on the right contains 22 self-intersecting triangles in the ears of the lion, i.e., it is not completely intersection-free. However, *all* of these 22 self-intersections are already present in the reference shape. Therefore, our method successfully removes even very complex intersections that had been introduced by initially modeling the target shape. Figure 47 illustrates how the surface regions of $\mathcal{V}^{\mathcal{C}}$ are relocated to their corrected intersection-free rest positions. The corrections are proportional to the penetration depths in order to dissolve the self-intersections correctly.

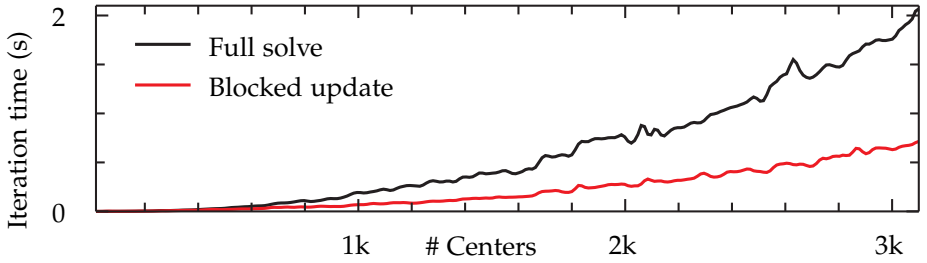
DISCRETIZATION. Integration cannot introduce self-intersections. This property holds strictly only for integration of a continuous surface. In the discrete

setting, we are integrating only vertex positions, such that we have to assume a sufficiently dense sampling of the surface. However, none of our experiments suffered from under-sampling and resulting self-intersections. This issue is discussed similarly in [vFTS06] in a different context of continuous deformations of triangle meshes by divergence-free vector fields.

BENEFITS FOR SUBSEQUENT PROCESSING. The motivation for surface repairing is not restricted to improving visual appearance. Even more important is the improvement and support of other geometry processing methods, which either suffer from defects or strictly require "clean" surface meshes. These are, e.g., methods based on nonlinear optimization. Here, the numerical computations frequently rely on well-behaved distribution of inner and dihedral angles. In our experiments we examine the recent example-driven single-step deformation and interpolation approach by Fröhlich et al. [FB11], who apply Gauss–Newton iterations for nonlinear optimization. Their optimization is sensitive to extreme dihedral angles (corresponding to local self-intersections) leading to increased number of iterations or even failure of convergence. Our experiments show that using the corrected poses the optimization required up to 60% *fewer* iterations for convergence. Moreover, for poses like the lion pose in the Figures 45 and 46, the nonlinear optimization even fails to converge for the original data but converges for our corrected shapes. Other example-driven nonlinear geometry processing methods are likely to benefit from corrected poses, too.

TIMINGS. We compare our RBF center selection scheme (with blocked LU factorization update and GPU-based evaluation) to the scheme used in [CBC*01] experimentally in the following table. The maximal condition number $\kappa(\mathbf{A})$ / average number of required centers / and average total run times for vector

Poses ($p \times \lvert\mathcal{V}\rvert$)	Carr et al. [CBC*01]	Ours
LION (5×5k)	$1 \cdot 10^{10}$ / 12k / $2.2 \cdot 10^3$	$2 \cdot 10^6$ / 2.7k / 13
CAT (5×7k)	$2 \cdot 10^{11}$ / 15k / $3.4 \cdot 10^3$	$2 \cdot 10^7$ / 4.5k / 42
HORSE (10×8k)	$3 \cdot 10^{10}$ / 13k / $3 \cdot 10^3$	$7 \cdot 10^6$ / 4.6k / 43
GOBLIN (86k)	exceeds memory limit	$6 \cdot 10^8$ / 10k / 603



field fitting are listed. The rows show results for different data sets (averages of p different poses) [3]. All times are given in seconds on a Linux PC equipped with a quad core AMD Phenom II CPU running at 3.4GHz, a NVIDIA GTX 460 GPU and 4GB of host memory. Our experiments show that our scheme leads to better conditioning of linear systems, which in turn leads to fewer centers and a significantly faster overall fitting process. Larger models may not even be processable without our scheme due to memory exhaustion, e.g., the GOBLIN model.

The plot underneath the table compares our blocked LU factorization update to traditional full factorization. It demonstrates the runtime speedup achieved when using our block factorization update scheme. Note that the speedup rate grows super-linearly with the number of centers. The LDL-based update scheme (41) is likely to further accelerate RBF fitting.

In all our experiments, the preprocessing stage required on average between 1.5s (LION), 1.9s (HORSE), and 2.3s (CAT) for the smaller models; it required 19s for the GOBLIN. Typical timings for integration are 29s (LION), 32s (CAT), and 39s for the GOBLIN (10k centers). Note that the integration does not only depend on the number of vertices and centers but also on the smoothness of the vector field: for instance, integration required only 19s for the HORSE model.

6.5 DISCUSSION

From a designer's point of view self-intersections may indeed be desirable: they tend to generate a more natural *visual appearance*, especially at joints of neighboring articulated parts (see Figure 37), although the actual surface does not represent the outer hull of a solid object anymore. However, we believe that other kinds of self-intersections (see Figures 41, 44, and 46) require corrections even for visual plausibility. This is especially necessary for multiple poses of a single model, as all interpolations or combinations of these shapes would inherit the visual artifacts. Even more important than visual aspects is the improvement of the triangulation in view of further algorithmic processing stages. This is a central goal of any mesh repair method. Our experiments show that numerical properties of a typical nonlinear mesh blending method like [FB11] are significantly improved.

LIMITATIONS. Our approach has several limitations. Self-intersections that are already present in the reference pose are reproduced, they are not corrected. In order to guarantee absence of self-intersections in the target, we require no

3 The LION, CAT, and HORSE poses were kindly provided by Robert Sumner [SP04].

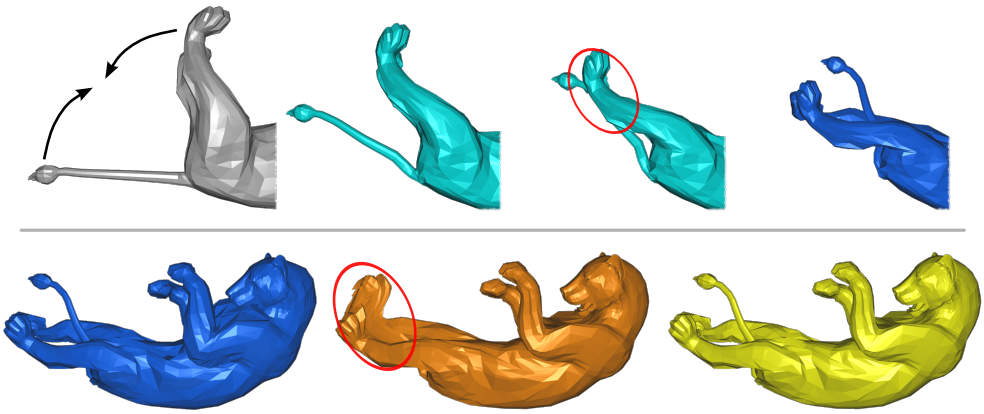


Figure 48: Limitations. The example shows key frames with opposing directions of motion, leading to *global* self-intersections: the lion's tail moves through the leg on the way to the target pose; there is no intersection between tail and leg in the target pose (top). By construction, this situation cannot be reproduced by vector field integration. Instead, integration results in a partially corrected (●, bottom) shape. The remaining regions are copied from the original target (●).

self-intersections in the reference. This is not a severe limitation because the reference surface is usually modeled from scratch, and the additional effort to manually remove defects is often not significant. In any case, an automatic standard mesh repair method can be applied to the reference surface before generating the target poses [Ju09].

Our method reproduces the shape of well-behaved regions. However, the reproduction is only approximately exact and slight variations in the final vertex coordinates are inevitable since the fitted vector field is only exact up to the error bound ϵ. Moreover, the tangent curves are only approximated in a *pointwise* way by the interpolation constraints. Hence, we trade reproduction quality with computation time. Yet, if higher quality reproductions are necessary, the fitting error bound ϵ and the number of interpolation time steps s can always be adjusted accordingly to improve the results. Also, vertices of the integrated poses that are close to their positions in the target pose can be mapped to these locations for exact reproduction.

Guaranteeing absence of self-intersections is a main feature of our method. However, this can also impose a fundamental limitation. Figure 48 shows an example where the key frame interpolation generates a global self-intersection: the lion's tail moves through the leg. Generally, such self-intersections can occur in practice. In a sense the input to interpolation methods like [XZWB06] is ill-posed: these methods usually depend on local surface properties, such as coordinate function gradients; they do not incorporate global (or even semantic) properties. Therefore, global self-intersection cannot be avoided automatically in key frame generation. By construction, our method is unable to reproduce self-intersections in space-time. So in the example, the lion's tail cannot move "through" the leg –

it will always stay in front of the leg while both regions are deformed. In such cases, we ignore the result from surface integration for the affected region and replace it instead by the corresponding region of the input target mesh.

6.6 SUMMARY

We presented a novel automatic approach to mesh repair that is tailored to the correction of a sequence of meshes with same connectivity representing different poses of the same shape. Our aim is to keep the mesh connectivity *fixed*; this is in contrast to existing mesh repair methods. This sort of mesh repair is important not only to suppress visual artifacts but also to improve numerical stability of further mesh processing stages that require compatible meshes.

Our approach to mesh repair is based on continuous mesh deformation. The deformation is defined by space-time integration of a smooth time-dependent vector field. This guarantees that deformations of the reference surface do not introduce self-intersections in the desired target shape. The vector field is fitted to interpolations of pairs of poses. Experiments for a variety of examples confirm the effectiveness of the method. Our approach is efficient thanks to first, an improved strategy to select RBF basis functions, second, a novel blocked RBF factorization update formulation, and third exploiting the GPU. These RBF-related contributions are not specific to our problem of mesh repair and can be beneficial for a variety of other methods, too.

The guaranteed avoidance of self-intersections is a main feature of space-time surface integration. However, this imposes also a limitation as global self-intersections cannot be reproduced. At present, we apply local modifications of the mesh to handle such cases. A challenge for future research is the avoidance of "collisions" that lead to global self-intersections during pose interpolation for key frame generation. Anther possible direction is the extension of the approach to space-time coherent remeshing.

7

SMOOTHED ENERGIES FOR GEOMETRY PROCESSING

The previous parts of this thesis demonstrated that a number of shape manipulation approaches can be computed by energy minimization. In fact, solutions of various problems in computer graphics are modeled in an elegant way by minimizers of problem-specific energies. Of particular interest are quadratic energies as their minimization leads to solving a linear system of equations, a task that is well-understood and relatively inexpensive. For this reason, other types of energies that involve more expensive nonlinear problems are often minimized iteratively, such that every step consists in the minimization of a quadratic energy, e.g., in Quasi-Newton methods. Energy is a scalar quantity, which is typically integrated over a domain. In discrete settings the domain is commonly partitioned into simple elements: the energy is defined locally on each element such that the global energy value consists in the sum of integrals over all elements. The simplest and most wide-spread partitions in computer graphics employ triangular elements for planar or two-manifold domains and tetrahedral elements for volumetric domains (see, e.g., Chapters 3 and 4).

A lot of research has been devoted so far to the definition of energies and to their discretization. In particular, this includes the well-known harmonic and conformal energies and discrete versions of gradient, divergence, and Laplace operators (or their counterparts in discrete exterior calculus). In this chapter, we study the construction of *regularization terms* — a topic that has got considerably less attention in many geometry processing applications. Regularization terms are used for different purposes: first of all, they may enable the solution of rank-deficient ill-posed problems. On the other hand, for well-posed problems they may "pull" the solution towards a certain class of feasible solutions by penalizing "undesired" behavior. A well-known and generic approach is Tikhonov regularization, which gives preference, e.g., to smoother solutions. De facto, this is the standard approach for regularizing the minimization of quadratic energies, especially for sparse operators.

In this chapter, we propose a different view on regularization that leads to a simple and generic construction of problem-specific regularized quadratic energies on triangular or tetrahedral partitions. Instead of demanding smoothness of the

solution, we give preference to solutions with low spatial variation in the *energy* over the domain. This leads to the design of a *problem-specific* regularization that is tailored to the particular energy. This is in contrast to generic Tikhonov regularization methods, which are independent of the energy and hence "problem-unaware". We show that there is a *generic regularization construction* that is formally expressed as change of a norm and that features a simple and generic implementation. We illustrate the recipe for the construction by applying this approach to a number of typical problems in geometry processing. The effect is demonstrated by experimental results, which reveal the practical benefits, e.g., avoidance of artifacts or, in some instances, the computation of solutions that so far were only achieved by costly nonlinear optimization. In particular, our approach does not increase the asymptotic complexity of the solved problems: all optimizations remain linear, only the space of solutions is modified.

Our regularization is inspired by the energy smoothing of Section 3.4.4, which can be considered to be a single specific bottom-up instantiation of our generic concept for the problem of vector field-based shape deformations. In this chapter, we provide a generic top-down regularization construction that is applicable to a broad range of energies modeling various problems.

7.1 BACKGROUND

Regularization is an important tool for solving numerical problems. Typical classes of application are the solution of rank-deficient problems or discrete ill-posed problems that frequently arise for inverse problems. (We refer the reader to, e.g., [Han10] for a rigorous introduction and overview.) The singular value decomposition reveals characteristics of linear operators, and regularization can be expressed as filtering singular values, e.g., truncation in the simplest case. Classic Tikhonov regularization (with $\mathbf{\Gamma} \equiv \gamma\mathbf{I}$ in (46), see next section) can be easily expressed as a "filter" for singular values. In practice, this is often not an option as the dimension of linear operators may become too large. In general, geometry processing applications involve too many degrees of freedom — still, they typically employ sparse linear operators that define an integration of a quadratic energy in a domain. Therefore, the classic formulation — adding a quadratic regularization term — is preferred, and the numerical solution is typically obtained from the normal equations or QR-factorization.

There is a huge body of research on the appropriate discretization of linear operators. This becomes evident especially in the geometry processing context, where the domain is often a two-manifold embedded in $3d$ space: the discretization of the Laplace-Beltrami operator on triangular meshes by, e.g., the well-known cotangent-weights [PP93] was only recently complemented by a formulation for general polygonal meshes by Alexa and Wardetzky [AW11], and there are a number of variants (see, e.g., Wardetzky et al. [WMKG07] for a review).

Interestingly, regularization operators got less attention. In fact, often regularization is not considered at all. Mostly, there is a good reason for this: many problems are not ill-posed, and the energy is carefully designed with a particular interpretation in mind, so from a puristic point of view regularization may spoil the solution. Still, we will show that there are possible benefits of using regularization. Probably, the most widely used "regularization" is the preference of "soft constraints" in least-squares sense, i.e., the introduction of a penalty term instead of elimination (see, e.g., [BS08]). If regularization takes place, then this is typically done by preferring either a low-norm solution or a smooth solution. Both refers to standard Tikhonov regularization and does not take the particular choice of energy into consideration. A typical example is the reconstruction of curves and surfaces, where a smooth solution is preferred. Here, the additional smoothness term accounts for missing data, for instance for parametric spline fitting (see, e.g., [HL93]) or for more complex discrete surface reconstruction (e.g., [ACSTD07]). Eckstein et al. [EPT*07] use regularization for the specific problem of curvature flow design. They construct specific regularizers or priors and show that this corresponds to constructing new inner products or norm alteration. We will get to a similar interpretation for our generic approach.

Examples for ill-posed problems include the reconstruction of (geometric) data by Bayesian statistics, where the so-called prior distribution provides a suitable model of the data. The surface reconstruction in [HAW07], for instance, uses the prior to constrain a "prototype surface". Generally, the prior restricts the solution space and enables a meaningful solution; this can be seen as a regularization as well. Another notable example of a problem that is likely to be ill-posed is the estimation of a linear operator for the refinement of animations: Kavan et al. [KGBS11] transform a quadratic energy into a spectral space where the regularization operator, a spatial low-pass filter, becomes diagonal. Note that we apply our regularization to problems that are generally *not* ill-posed a priori.

The applications shown in this chapter focus on typical geometry processing tasks. Although our approach is general and not restricted to this application domain, we mostly stick to this perspective in the following. We selected a number of standard problems and representative state-of-the-art solutions as examples for employing our approach. Credits to these original approaches as well as brief summaries are given in the respective parts of Section 7.4.

7.2 SMOOTHED ENERGIES

We study problems in form of function spaces $\mathcal{P}$ whose solutions are defined as minimizers of global quadratic energies $E_{\mathcal{P}}$ defined in domains $\mathcal{D}$. These energies have the general form

$$E_{\mathcal{P}}(\mathbf{u}) = \int_{\mathcal{D}} \|\mathbf{e}(\mathbf{u}(\mathbf{x}), \mathbf{x})\|^2 \, \mathrm{d}\mathbf{x} \, . \tag{42}$$

Here, functions $\mathbf{u} \in \mathcal{P}$, $\mathbf{u}\colon \mathcal{D} \to \mathbb{R}^d$ are *feasible solutions* of a specific problem $\mathcal{P}$ with dimension d, and $\mathbf{x} \in \mathcal{D}$ are points in the domain where *local energies* $\mathbf{e}(\mathbf{u}, \mathbf{x})$ corresponding to $\mathcal{P}$ are measured. The domains may be manifolds, e.g., two-manifolds embedded in $3d$. The vector valued $\mathbf{e}$ are linear in $\mathbf{u}$, and their dimension n depends on the concrete problem $\mathcal{P}$. Optimal solutions

$$\mathbf{u}^\dagger = \operatorname*{argmin}_{\mathbf{u}\in\mathcal{P}} \; E_{\mathcal{P}}(\mathbf{u}) \tag{43}$$

are the minimizers of $E_{\mathcal{P}}$, they are obtained as solutions of a linear system. This optimization is generally performed subject to suitable boundary constraints on $\mathbf{u}^\dagger$. Boundary constraints either guarantee unique solutions, or they are imposed by the user to modify the solution.

Many problems in geometry processing are in this class $\mathcal{P}$ because they require minimization of quadratic energies. For example, $\mathcal{D}$ could represent an initial surface, and $\mathbf{u}$ would be the coordinate function of a deformed version of $\mathcal{D}$. In this case $\mathbf{e}$ measures some form of distortion induced by the deformation $\mathbf{u}$ relative to the initial shape given by $\mathbf{x}$. We will show several examples in Section 7.4.

Often energies of the form (42) alone are not sufficient to define desirable solutions, as the local energies $\mathbf{e}$ do not measure smoothness of the solution. This leads to undesirable artifacts in $\mathbf{u}^\dagger$, which are often most noticeable near user-imposed constraints. To alleviate undesired behavior, a common approach is to add an additional *regularization energy* $E_{\mathcal{R}}(\mathbf{u})$ that penalizes non-smooth or in other respects undesirable behavior of $\mathbf{u}^\dagger$:

$$E(\mathbf{u}) = (1-\beta)\, E_{\mathcal{P}}(\mathbf{u}) + \beta\, E_{\mathcal{R}}(\mathbf{u}) \tag{44}$$

$$\mathbf{u}^\star = \operatorname*{argmin}_{\mathbf{u}\in\mathcal{P}} \; E(\mathbf{u})\ . \tag{45}$$

Here, $\mathbf{u}^\star$ denotes the optimal solution of the regularized problem $E(\mathbf{u})$. The amount of regularization is steered by the weight $\beta \in [0, 1)$, i.e., E is a convex combination of $E_{\mathcal{P}}$ and $E_{\mathcal{R}}$. These energies often use forms of *generic Tikhonov regularizers* $E^T_{\mathcal{R}}$ that have the general form

$$E^T_{\mathcal{R}}(\mathbf{u}) = \int_{\mathcal{D}} \|\mathbf{\Gamma}(\mathbf{u}(\mathbf{x}))\|_F^2 \,\mathrm{d}\mathbf{x}\ , \tag{46}$$

and a prominent choice of regularization operator is given by the gradient operator $\mathbf{\Gamma}(\cdot) \equiv \nabla\cdot$ on $\mathcal{D}$ to enforce global smoothness of $\mathbf{u}$ (see, e.g., [Han10]). We identify two major drawbacks of using standard regularizations: first, generic regularizers like those based on a low-pass filter on $\mathbf{u}$ are *independent* of the concrete problem that is defined by the energy $E_{\mathcal{P}}$. We call this property *problem-unawareness*. They often perform poorly or too aggressively in correcting artifacts in $\mathbf{u}^\dagger$ (see Section 7.6). Second, oftentimes the design of effective regularizations is not obvious and requires careful parameter tuning to not attenuate the result of the original energy $E_{\mathcal{P}}$ too severely.

To overcome these drawbacks, we propose a regularization that is based on energy smoothness. We define the quadratic *energy smoothness regularization* as

$$E_{\mathcal{R}}(\mathbf{u}) = \int_{\mathcal{D}} \|\nabla \mathbf{e}(\mathbf{u}(\mathbf{x}), \mathbf{x})\|_F^2 \, \mathrm{d}\mathbf{x}, \tag{47}$$

i.e., as the isotropic squared norm of the first-order energy variation. In contrast to functionals like (46), we do not directly measure variation of the solution $\mathbf{u}$, but variation of the induced local energies $\mathbf{e}$. This way the regularization is tightly coupled and tailored to the original energy (42) and is *problem-specific*. In particular, $E_{\mathcal{R}}$ and standard Tikhonov regularizers $E_{\mathcal{R}}^T$ are generally not identical, which will be discussed in Section 7.6.

We will demonstrate in Section 7.4 that this conceptually simple regularization strategy leads to attractive results for a wide range of problems. Due to the linearity of the gradient operator, the tensor field $\nabla \mathbf{e}$ is also linear in $\mathbf{u}$. Hence, the regularized optimization (45) stays quadratic in $\mathbf{u}$ and is efficient to compute. Moreover, once a discretization of the original problem (43) is set up, the regularization using (47) is very simple to obtain. We continue to show this property in the following section.

7.3 DISCRETIZATION

In the following we consider only two-dimensional domains as the construction is essentially the same for three dimensions. We discretize planar or two-manifold domains $\mathcal{D}$ by triangular meshes $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, i.e., as sets of vertices $i \in \mathcal{V}$, edges $\mathcal{E} \subset \mathcal{V}^2$, and triangles $\mathcal{T} \subset \mathcal{V}^3$. The edges $\mathcal{E}$ can be partitioned into two disjoint sets: interior edges $\mathcal{E}_i$ and exterior edges $\mathcal{E}_e$, the latter positioned at the boundary of $\mathcal{M}$. We write $l(e)$ and $r(e)$ to denote the left and right triangle at an internal edge e, respectively. We consider discrete functions on $\mathcal{M}$ that are piecewise linear and represented by coefficients at the vertices, e.g., the vertex coordinate function $\mathbf{x}_i \in \mathbb{R}^{2/3}$, or problem solutions $\mathbf{u}_i \in \mathbb{R}^d$. All coefficients of a function are "stacked" into a single vector without subscript, e.g., $\mathbf{u} \in \mathbb{R}^{d|\mathcal{V}|}$, or stacked component-wise into a single matrix $\mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times d}$. For a triangle $t = (i, j, k) \in \mathcal{T}$ we denote the stacked coefficients at all vertices of t by $\mathbf{u}_t = \left(\mathbf{u}_i{}^{\mathrm{T}}, \mathbf{u}_j{}^{\mathrm{T}}, \mathbf{u}_k{}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{3d}$, and the component-wise stacked coefficient matrix by $\mathbf{U}_t = \left(\mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k\right)^{\mathrm{T}} \in \mathbb{R}^{3 \times d}$. For convenience, we define the single-entry $r \times s$ matrix $\mathbf{\Lambda}_{r,s}^{i,j}$ that is 1 at (i, j) and zero elsewhere. Finally, we use the notation

$$\|\mathbf{y}\|_{\mathbf{N}}^2 = \mathbf{y}^{\mathrm{T}} \mathbf{N} \, \mathbf{y} \quad \text{and} \quad \|\mathbf{Y}\|_{\mathbf{N}}^2 = \mathrm{Tr}\left(\mathbf{Y}^{\mathrm{T}} \, \mathbf{N} \, \mathbf{Y}\right) \tag{48}$$

for (squared) vector and matrix norms that are defined by symmetric and positive-definite matrices $\mathbf{N}$. A popular example for such a norm is the well-known Mahalanobis distance that is defined by the inverse of a covariance matrix.

TRIANGLE-BASED ENERGIES. The majority of geometry processing approaches discretize continuous energies on a per triangle basis, e.g., Poisson-based methods (see, e.g., [BKP*10]). In these discretizations the local energies $\mathbf{e}$ of $E_{\mathcal{P}}$ are constant on each triangle, and domain integration is performed by area weighting. Most discretizations of (42) for global, purely quadratic triangle-based energies can be expressed in the form

$$E_{\mathcal{P}}(\mathbf{u}) = \|\mathbf{E}\,\mathbf{u} - \mathbf{c}\|^2_{\mathbf{A}_n} \,. \tag{49}$$

Here, $\mathbf{E}(\mathbf{x})$ is a global *energy matrix* of dimension $n\,|\mathcal{T}| \times d\,|\mathcal{V}|$, and $\mathbf{c}(\mathbf{x}) \in \mathbb{R}^{n|\mathcal{T}|}$ is triangle-constant. $\mathbf{E}$ maps feasible solutions $\mathbf{u}$ to $|\mathcal{T}|$ consecutive local n-dimensional vectors $\mathbf{e}_t$, which are constant per triangle t. In most cases, $\mathbf{E}$ is a sparse matrix. In the remainder of this chapter, we assume that all $\mathbf{e}_t$ are defined in a common coordinate system, e.g., the canonical world coordinates in $\mathbb{R}^3$. This can always be ensured by construction. Then $\|\mathbf{e}_t - \mathbf{c}_t\|^2$ is also constant per triangle. Its integration over the mesh is performed by the $n\,|\mathcal{T}|$-dimensional square diagonal matrix $\mathbf{A}_n(\mathbf{x})$ given by $\mathbf{A}_n = \mathbf{I}_n \otimes \mathbf{A}$. Here, $\otimes$ is the Kronecker product, $\mathbf{I}_n$ is the $n \times n$ identity matrix, and $\mathbf{A}(\mathbf{x})$ is the diagonal matrix of triangle areas.

For applications where the coordinate functions of the solutions are decoupled, $\mathbf{E}$ can be expressed as a matrix of dimension $n\,|\mathcal{T}| \times |\mathcal{V}|$, and the vectors $\mathbf{u}$ and $\mathbf{c}$ become matrices $\mathbf{U}$ and $\mathbf{C}$ whose d columns correspond to coordinate functions. The general formulation (49) remains the same with the difference that it expresses a matrix norm (see (48)).

ENERGY SMOOTHNESS. Given a discretized energy in the form (49), the derivation of a discretization of the energy smoothness regularization $E_{\mathcal{R}}$ of (47) turns out to be straightforward. This is because the gradient field $\nabla\mathbf{e}$ vanishes almost everywhere on $\mathcal{M}$ due to the constantness of $\mathbf{e}$ in each triangle: $\mathbf{e}$ varies only across internal edges. Therefore, the tensor $\nabla\mathbf{e}$ only has a directional derivative component orthogonal to each edge at each point of an edge. We estimate these directional derivatives by finite differences. A discretization of the energy smoothness regularization $E_{\mathcal{R}}$ that is directly based on the discrete energy $E_{\mathcal{P}}$ is then given by

$$E_{\mathcal{R}}(\mathbf{u}) = \|\mathbf{D}_n\,(\mathbf{E}\,\mathbf{u} - \mathbf{c})\|^2_{\mathbf{B}_n} \,.$$

Here, $\mathbf{D}_n$ is a sparse $n\,|\mathcal{E}_i| \times n\,|\mathcal{T}|$ discrete differences operator on the local energies. It is given by $\mathbf{D}_n = \mathbf{I}_n \otimes \mathbf{Q}$, where the connectivity matrix $\mathbf{Q}$ is nonzero only at

$$Q_{et} = \begin{cases} 1 & \text{if } l(e) = t \\ -1 & \text{if } r(e) = t \end{cases},$$

for all internal edges $e \in \mathcal{E}_i$ and triangles $t \in \mathcal{T}$. As $\mathbf{e}$ is constant along each side of an edge, the integration of (47) simplifies to a weighted sum of quadratic directional differences of the local energy along each internal edge. This integration

is performed by the $n\,|\mathcal{E}_i|$ square diagonal matrix $\mathbf{B}_n(\mathbf{x})$ given by $\mathbf{B}_n = \mathbf{I}_n \otimes \mathbf{B}$, where $\mathbf{B}(\mathbf{x})$ is the diagonal matrix of edge lengths of all internal edges. Ultimately, this discretization of (47) can be interpreted as measuring the squared differences of local energies along each internal edge, which are weighted by the respective edge lengths.

The discretization of the total regularized energy (44) is then given by

$$E(\mathbf{u}) = (1-\beta)\,\|\mathbf{E}\,\mathbf{u} - \mathbf{c}\|^2_{\mathbf{A}_n} + \beta\,\|\mathbf{D}_n\,(\mathbf{E}\,\mathbf{u} - \mathbf{c})\|^2_{\mathbf{B}_n}\,.$$

Using the energy residual vector $\mathbf{r} = \mathbf{E}\mathbf{u} - \mathbf{c}$, we can transform this formulation to

$$\begin{aligned} E(\mathbf{u}) &= (1-\beta)\,\|\mathbf{r}\|^2_{\mathbf{A}_n} + \beta\,\|\mathbf{D}_n\,\mathbf{r}\|^2_{\mathbf{B}_n} \\ &= (1-\beta)\,\mathbf{r}^{\mathrm{T}}\,\mathbf{A}_n\,\mathbf{r} + \beta\,\mathbf{r}^{\mathrm{T}}\,{\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{B}_n\,\mathbf{D}_n\,\mathbf{r} \\ &= \mathbf{r}^{\mathrm{T}}\,\underbrace{\left((1-\beta)\,\mathbf{A}_n + \beta\,{\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{B}_n\,\mathbf{D}_n\right)}_{\mathbf{W}_n}\,\mathbf{r} = \mathbf{r}^{\mathrm{T}}\,\mathbf{W}_n\,\mathbf{r} = \|\mathbf{r}\|^2_{\mathbf{W}_n} \end{aligned}$$

(the derivation for matrix energy residuals $\mathbf{R} = \mathbf{E}\,\mathbf{U} - \mathbf{C}$ is analog). Hence, we obtain the compact form

$$\begin{aligned} E(\mathbf{u}) &= \|\mathbf{E}\,\mathbf{u} - \mathbf{c}\|^2_{\mathbf{W}_n} \\ \text{with}\quad \mathbf{W}_n &= (1-\beta)\,\mathbf{A}_n + \beta\,{\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{B}_n\,\mathbf{D}_n\,. \end{aligned} \tag{50}$$

β-WEIGHTED NORM. The total regularized energy in (50) has the remarkable property that it measures a *modified* norm of the *same* energy residual $\mathbf{r}$ as (49). In fact, the domain integration represented by $\mathbf{A}_n$ simply has to be replaced by the β-weighted *generalized norm* $\mathbf{W}_n(\mathbf{x}, \beta)$.

To show that the norm $\|\cdot\|^2_{\mathbf{W}_n}$ is well-defined for $\beta \in [0, 1)$, we show that $\mathbf{W}_n$ is positive-definite: assuming non-degenerate meshes, the integration matrices $\mathbf{A}_n$ and $\mathbf{B}_n$ are diagonal with strictly positive entries, i.e., they are positive-definite. The difference operator $\mathbf{D}_n$ is rank deficient, i.e., one or more eigenvalues of ${\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{D}_n$ are equal to zero. This is because a manifold triangle polyhedron always has more (internal) edges than triangles (Euler's formula). Hence, $\mathbf{D}_n$ has more rows than columns, and together with the fact that a discrete difference operator has a nontrivial kernel, its maximum rank is $n\,|\mathcal{T}| - 1$. Therefore, ${\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{B}_n\,\mathbf{D}_n$ is symmetric and positive-*semi*-definite, i.e., $\mathbf{y}^{\mathrm{T}}\,{\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{B}_n\,\mathbf{D}_n\,\mathbf{y} \geq 0$ for $\mathbf{y} \neq \mathbf{0}$. Positive definiteness of $\mathbf{W}_n$ follows from the linearity of sums as for $\mathbf{y} \neq \mathbf{0}$ and $\beta \in [0, 1)$ we have $\mathbf{y}^{\mathrm{T}}\mathbf{A}_n\mathbf{y} > 0$ and

$$\mathbf{y}^{\mathrm{T}}\,\mathbf{W}_n\,\mathbf{y} = (1-\beta)\,\mathbf{y}^{\mathrm{T}}\,\mathbf{A}_n\,\mathbf{y} + \beta\,\mathbf{y}^{\mathrm{T}}\,{\mathbf{D}_n}^{\mathrm{T}}\,\mathbf{B}_n\,\mathbf{D}_n\,\mathbf{y} > 0\,.$$

Therefore, the norm $\|\cdot\|^2_{\mathbf{W}_n}$ is well-defined for $\beta \in [0, 1)$.

For $\beta = 0$, the original energy is recovered, and $\mathbf{W}_n$ only performs energy integration. For $\beta > 0$, $\mathbf{W}_n$ also measures first-order energy variation. Instead of energy residuals, any other triangle-constant function may be integrated using $\mathbf{W}_n$. Generally, $\mathbf{W}_n$ defines a normed function space of triangle-constant functions. Minimization in this space yields functions of low norms, which incorporates low energy variation as steered by β. As the minimization problem is defined by the energy, penalizing energy variation results in a *problem-specific* regularization. In Section 7.5, we show for a number of typical applications that the minimizers $\mathbf{u}^\star$ of problem-specific, smoothed energies E defined by $(\mathbf{E}, \mathbf{c})$ are preferable to solutions without regularization or those from using Tikhonov regularization.

OPTIMIZATION. The minimizer $\mathbf{u}^\star$ of the discretization of (45) given by (50) can be computed by solving for the critical point of $\nabla E(\mathbf{u})$. Using normal equations, this is equivalent to solving the linear system

$$\mathbf{E}^\mathrm{T}\mathbf{W}_n\,\mathbf{E}\,\mathbf{u}^\star = \mathbf{E}^\mathrm{T}\mathbf{W}_n\,\mathbf{c}\,. \tag{51}$$

After a potential elimination of boundary constraints, which might be required due to rank deficiencies of $\mathbf{E}$, the system becomes symmetric positive-definite and can be solved with the Cholesky factorization $\mathbf{L}\,\mathbf{L}^\mathrm{T} = \mathbf{E}^\mathrm{T}\,\mathbf{W}_n\,\mathbf{E}$. Depending on the application, the factorization can be reused for different right-hand sides $\mathbf{c}$. This factorization is efficient also for sparse matrices if an additional fill-in reducing reordering is applied. Obviously, if $\mathbf{E}$ is sparse then the system matrix remains sparse. However, for $\beta > 0$ the number of nonzero entries increases as $\mathbf{W}_n$ is "quasi-banded" and not diagonal anymore, whereas $\mathbf{A}_n$ is. Our experiments indicate that this lower sparseness results in almost no loss in performance (see Section 7.5). This is because the dimension of the solved linear systems is unchanged, and this dominates the asymptotic runtime of linear solvers.

GENERIC IMPLEMENTATION. The generalization of $\|\cdot\|^2_{\mathbf{A}_n}$ to $\|\cdot\|^2_{\mathbf{W}_n}$ is also very beneficial from an implementation point of view: many problems that employ quadratic energies on meshes are already represented in the discretized form (49). Given a model of the problem as $(\mathbf{E}, \mathbf{c})$, adding a problem-specific regularization is nothing more than replacing the integration $\mathbf{A}_n$ by $\mathbf{W}_n$, i.e., to solve (50). The setup of the matrix $\mathbf{W}_n$ is simple, it depends only on the connectivity and coordinates of the domain mesh $\mathcal{M}$. In particular, the setup of $\mathbf{W}_n$ is *independent* of a concrete energy or concrete problem and can be reused. This makes the construction of the regularization a generic function that takes as input β, a mesh $\mathcal{M}$, and a problem $(\mathbf{E}, \mathbf{c})$ with local energy dimensionality n.

VOLUMETRIC DOMAINS. The discretization is not restricted to triangulations of $2d$ domains. The setting is very similar for $3d$ (and even higher-dimensional)

piecewise linear domains. There, a tetrahedral partition of the domain is considered, and the local energies $\mathbf{e}$ are constant functions defined per tetrahedron. Then the diagonal matrix $\mathbf{A}_n$ performs the integration, and the difference operator $\mathbf{D}_n$ is defined for inner triangles, each shared by its left and right tetrahedron. Consequently, the integration of squared energy variation by $\mathbf{B}_n$ is then based on triangle areas instead of edge lengths.

VERTEX-BASED ENERGIES. Until now, we consider only local energies that are constant on each triangle. A similar approach is possible for piecewise linear energies that are defined as nodal values at vertices. Note that it is generally easier and more natural to define energies or errors on a per-triangle basis, whereas it is commonly artificial to do this for vertices. For example, all of our applications in Section 7.4 will only require triangle-based energies. Still, for instance finite element-based energies are often discretized on a per-vertex basis. For completeness, we give a short outline for the vertex-based energy setting, for which our concept is applicable as well.

Quadratic vertex-based energies on triangle meshes can again be written in the general form

$$E_{\mathcal{P}}(\mathbf{u}) = \|\mathbf{E}\,\mathbf{u} - \mathbf{c}\|^2_{\mathbf{M}_n} .$$

Here, $\mathbf{E}(\mathbf{x})$ is the energy matrix of dimension $n\,|\mathcal{V}| \times d\,|\mathcal{V}|$, and $\mathbf{c}(\mathbf{x}) \in \mathbb{R}^{n|\mathcal{V}|}$ is vertex-constant. In contrast to (49), energies are piecewise linear and we perform energy integration by the $n\,|\mathcal{V}| \times n\,|\mathcal{V}|$ *mass matrix* $\mathbf{M}_n(\mathbf{x})$ that integrates each of the n energy components separately (see Appendix A.2 for a derivation of $\mathbf{M}_n$ for $n = 1$). For regularization, we measure the first-order energy variation by integrated energy gradients that are triangle-constant:

$$E_{\mathcal{R}}(\mathbf{u}) = \|\mathbf{G}_n\,(\mathbf{E}\,\mathbf{u} - \mathbf{c})\|^2_{\mathbf{A}_{mn}} .$$

Here, $\mathbf{G}_n$ is the $mn\,|\mathcal{T}| \times n\,|\mathcal{V}|$ gradient operator that assigns each triangle the constant gradient vector of each of the n energy components in a canonical m-dimensional basis, e.g., $m = 2$ for triangle meshes and $m = 3$ for tetrahedral meshes. $\mathbf{G}_n$ can be constructed in a component-wise way by replicating the plain two-variate gradient operator $\mathbf{G}$ on $\mathcal{M}$, see, e.g., (57) in the next section for the $m = 2$ case or Appendix A.1. Integration of the constant squared gradient norms is then again performed by area-based weighting using the diagonal triangle area matrix $\mathbf{A}_{mn}$.

The vertex-based energy regularization can again be expressed compactly as

$$\begin{aligned} E(\mathbf{u}) &= \|\mathbf{E}\,\mathbf{u} - \mathbf{c}\|^2_{\mathbf{W}_n} \\ \text{with} \quad \mathbf{W}_n &= (1-\beta)\,\mathbf{M}_n + \beta\,{\mathbf{G}_n}^{\mathsf{T}}\,\mathbf{A}_{mn}\,\mathbf{G}_n . \end{aligned}$$

Hence, similar to the triangle-based energies, we obtain regularized solutions by optimization in a space of smoother solutions expressed by the modified norm $\mathbf{W}_n$.

We note that different integration operators are commonly used in the literature, e.g., diagonal lumped barycentric mass matrices (see Appendix A.2). These operators can be incorporated into our framework in a similar way. However, depending on the particular integration model, different forms of energy gradient operators need to be discretized. For instance, for constant barycentric integration, gradients need to be estimated along the internal dual edges of the mesh.

7.4 APPLICATIONS

We demonstrate our approach for a number of typical problems $\mathcal{P}$ from geometry processing. The minimization of quadratic energies on meshes is ubiquitous in this application domain. The particular problems, their discretizations and solutions have been addressed in a range of prior work. We reference representative work and describe how the problem-specific discretization $(\mathbf{E}, \mathbf{c})$ is set up. We try to keep this description abstract to emphasize the main differences of the particular settings. We do not recap the setup and elimination of boundary conditions. Boundary conditions are generally "hard" interpolation constraints like, for instance, Dirichlet conditions. Adding our regularization is then a simple and generic procedure as described above. Our examples include linear $2d$ shape deformations, nonlinear and continuous deformations in $2d$ and $3d$, and surface parametrization.

7.4.1 Planar Linear Deformations

Shape deformations $\mathcal{P}$ of two-dimensional planar meshes $\mathbf{x}_i \in \mathbb{R}^2$ are displacement maps $\mathbf{u}(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^2$ with $\mathbf{u}(\mathbf{x}_i) = \mathbf{u}_i$, $i \in \mathcal{V}$ that are piecewise linear on each triangle $t \in \mathcal{T}$: $\mathbf{u}(\mathbf{x}) = \mathbf{F}_t\,\mathbf{x} + \mathbf{t}_t$. A common approach to defining energy minimizing deformations is to measure the difference of a particular deformation from a prototype deformation (see also Appendix B). Then the optimal deformation is given by the energy minimizing solution $\mathbf{u}^\star$. As deformations are piecewise linear, the 2×2 deformation gradient tensor field $\mathbf{F}_t(\mathbf{u})$ is constant on each triangle. Isotropically integrated translation-invariant discrete deformation energies are therefore given by

$$E_{\mathcal{P}}(\mathbf{u}) = \sum_{t \in \mathcal{T}} A_t \left\| \mathbf{F}_t - \mathbf{M}(\mathbf{F}_t) \right\|_F^2 , \tag{52}$$

with triangle areas A_t and closest 2×2 prototype deformation gradient tensors $\mathbf{M}(\mathbf{F}_t)$. Different choices for $\mathbf{M}$ will be discussed in the following paragraphs. As

$\mathbf{F}_t$ is the gradient of $\mathbf{u}$ on a triangle t, it can be computed using a 2×3 gradient operator $\mathbf{G}_t(\mathbf{x})$: $\mathbf{F}_t = \mathbf{G}_t \mathbf{U}_t$. We use a $2d$ gradient operator that computes gradients in a common global coordinate system to be able to apply our regularization. It is given for a triangle $t = (i, j, k)$ by

$$\mathbf{G}_t = \begin{bmatrix} (\mathbf{x}_j - \mathbf{x}_i)^\mathrm{T} \\ (\mathbf{x}_k - \mathbf{x}_i)^\mathrm{T} \end{bmatrix}^{-1} \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \tag{53}$$

(see Appendix A.1 for a derivation). It is convenient to define a linear 4×6 operator $\mathbf{H}_t = \begin{bmatrix} \mathbf{G}_t & \\ & \mathbf{G}_t \end{bmatrix} \mathbf{P}$ with an appropriate permutation $\mathbf{P}$, such that $\mathrm{vec}(\mathbf{F}_t) = \mathbf{H}_t \mathbf{u}_t$ is the column-wise vectorization of $\mathbf{F}_t$, i.e., the stacked gradients of all displacement coordinate functions.

AS-SIMILAR-AS-POSSIBLE (ASAP). We call deformations as–similar–as–possible (ASAP) if they are approximately conformal, i.e., angles should be preserved. Here, we use a characterization that is similar to the one given by Liu et al. [LZX*08]. Discrete $2d$ deformations are conformal in least-squares sense if all deformation gradients $\mathbf{F}_t = \left(\begin{smallmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{smallmatrix}\right)$ are as close as possible to the closest $2d$ similarity matrix $\mathbf{S} \in \left\{ \left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right) \mid a, b \in \mathbb{R} \right\}$ that minimizes $\|\mathbf{F}_t - \mathbf{S}\|_F^2$ (see [BS08]). The similarity minimizing this distance is

$$\mathbf{S}_t^\star = \frac{1}{2} \begin{pmatrix} f_{11} + f_{22} & f_{12} - f_{21} \\ f_{21} - f_{12} & f_{11} + f_{22} \end{pmatrix} .$$

The prototype deformation tensor in (52) is therefore given by the closest similarity $\mathbf{M}(\mathbf{F}_t) = \mathbf{S}_t^\star$. $\mathbf{M}$ is linear in $\mathbf{F}_t$ for this case of approximately conformal deformations. Each triangle-constant energy term of the summation in (52) can thus be written as $\|\mathbf{E}_t \mathbf{u}_t\|^2$ with local 4×6 energy operators

$$\mathbf{E}_t = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \mathbf{H}_t .$$

The local operators $\mathbf{E}_t$ on individual triangles can then be assembled into the single global $4\,|\mathcal{T}| \times 2\,|\mathcal{V}|$ operator $\mathbf{E}$ to give the energy (52) in the general form (49):

$$\mathbf{E} = \left(\sum_{t \in \mathcal{T}} \mathbf{E}_t \otimes \mathbf{\Lambda}^{t,t}_{|\mathcal{T}|,|\mathcal{T}|} \right) \mathbf{Q} \,,\ \mathbf{c} = \mathbf{0} \,. \tag{54}$$

Here, $\mathbf{Q}$ is an appropriate replication matrix that selects all $\mathbf{u}_i$ associated with a particular triangle, and $\mathbf{\Lambda}^{t,t}_{|\mathcal{T}|,|\mathcal{T}|}$ is a single-entry matrix (see Section 7.3). This description of the as-similar-as-possible deformation energy is then suitable for regularization by (50). For this problem, we have $n = 4$ as each triangle has four associated energy components.

LINEARIZED AS-RIGID-AS-POSSIBLE (LARAP). $2d$ deformations that are as-rigid-as-possible (ARAP) are commonly computed by optimizing for deformation gradients that are close to rotations. Due to the nonlinearity of rotations, this optimization generally requires iterative schemes (see Section 7.4.2). Linearization of rotations is used to avoid nonlinear problems. However, this simplification is often considered defective as large rotations lead to linearization artifacts. Still, we show that even this linearized setting gives significantly more competitive results with our proposed regularization scheme.

The rotation $\mathbf{R} \in \left\{ \begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \mid \alpha \in \mathbb{R} \right\}$ that minimizes $\|\mathbf{F}_t - \mathbf{R}\|_F^2$ has an optimal rotation angle of $\alpha^\star = \tan^{-1}(a)$ with $a = f_{12} - f_{21} / f_{11} - f_{22}$, and is given by

$$\mathbf{R}_t^\star = \frac{1}{\sqrt{a^2+1}} \begin{pmatrix} 1 & a \\ -a & 1 \end{pmatrix} . \tag{55}$$

A linearized approximation $\mathbf{R}_t^\dagger \approx \mathbf{R}_t^\star$ is obtained from a Taylor series expansion of $\mathbf{R}_t^\star$ around $\mathbf{I}_2$:

$$\mathbf{R}_t^\dagger = \frac{1}{2} \begin{pmatrix} 2 & f_{12} - f_{21} \\ f_{21} - f_{12} & 2 \end{pmatrix} .$$

Then, we identify $\mathbf{M}(\mathbf{F}_t) = \mathbf{R}_t^\dagger$ as the prototype deformation tensor in (52) that gives a (linearized) quadratic energy that measures deviation from rigidity. Triangle-constant local energies $\|\mathbf{E}_t \, \mathbf{u}_t - \mathbf{c}_t\|^2$ are then given by

$$\mathbf{E}_t = \frac{1}{2} \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \mathbf{H}_t \text{, and } \mathbf{c}_t = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} .$$

Similar to (54), the global energy matrix $\mathbf{E}$ is assembled block-wise of all $\mathbf{E}_t$, and $\mathbf{c}$ is obtained by stacking all $\mathbf{c}_t$. Again, the dimension of local energies is $n = 4$.

7.4.2 Nonlinear Planar Deformations and Poisson-type Problems

The nonlinear as-rigid-as-possible (ARAP) energy is (52) with exact rotations $\mathbf{M}(\mathbf{F}_t) = \mathbf{R}_t^\star$ as deformation gradient prototypes [IMH05]. The optimization of these energies, which are nonlinear in $\mathbf{u}$, is often performed iteratively with alternating linearizations [SA07, LZX*08] or by directly using nonlinear solvers [CPSS10, Lip12]. We focus on the first type of iterative optimization, which is most related to our framework as each step minimizes a quadratic energy. In each iteration k an intermediate solution $\mathbf{u}^k$ is improved by computing the closest rotations to the deformation gradients $\mathbf{F}_t^k$ of $\mathbf{u}^k$ relative to the initial shape

$\mathbf{x}$. Here, $\mathbf{R}_t^\star$ of (55) could be used. A numerically more stable variant is based on the singular value decomposition (SVD) $\mathbf{F}_t^k = \mathbf{S}\,\mathbf{\Sigma}\,\mathbf{T}^\mathrm{T}$, such that $\mathbf{R}_t^\star = \mathbf{S}\,\mathbf{T}^\mathrm{T}$ is the polar decomposition of $\mathbf{F}_t^k$. (There are alternative methods to obtain the polar decomposition.) Furthermore, an assumption requires that there is no reflection, i.e., we have $\det(\mathbf{S}\,\mathbf{T}^\mathrm{T}) = +1$. The computation of optimal rotations is performed locally per triangle and does not involve any regularization. It is the prerequisite for the next step: the definition of a quadratic energy for the $(k+1)$-th iteration step. Therefore, the matrices $\mathbf{R}_t^\star$ are used as the target deformation gradients to reconstruct the coordinates of $\mathbf{u}^{k+1}$ in least-squares sense. Reconstruction errors are triangle-constant due to the constant gradients of $\mathbf{u}$. The corresponding integrated triangle-based reconstruction energy can be decomposed into component-wise functions $\mathbf{U}$:

$$\begin{aligned} E_\mathcal{P}(\mathbf{u}^{k+1}) &= \sum_{t\in\mathcal{T}} A_t \,\|\mathbf{F}_t^{k+1} - \mathbf{R}_t^\star\|_F^2 \\ &= \sum_{t\in\mathcal{T}} A_t \,\|\mathbf{G}_t\,\mathbf{U}_t^{k+1} - \mathbf{R}_t^\star\|_F^2 \\ &= \|\mathbf{G}\,\mathbf{U}^{k+1} - \mathbf{C}^k\|_{\mathbf{A}_n}^2 \,. \end{aligned} \tag{56}$$

This is a global energy similar to (54) with

$$\mathbf{G} = \left(\sum_{t\in\mathcal{T}} \mathbf{G}_t \otimes \mathbf{\Lambda}^{t,t}_{|\mathcal{T}|,|\mathcal{T}|}\right)\mathbf{Q}\,,\quad \mathbf{C}^k = \sum_{t\in\mathcal{T}} \mathbf{R}_t^\star \otimes \mathbf{\Lambda}^{t,1}_{|\mathcal{T}|,1}\,, \tag{57}$$

$n = 2$, and an appropriate replication matrix $\mathbf{Q}$. The global gradient operator $\mathbf{G}$ is sparse with a dimension of $2\,|\mathcal{T}| \times |\mathcal{V}|$, and $\mathbf{C}$ is a dense $2\,|\mathcal{T}| \times 2$ matrix. Again, this energy has the form (49) with

$$(\mathbf{E}, \mathbf{C}) = (\mathbf{G}, \mathbf{C}^k)$$

i.e., a matrix norm (48) is used, and regularization (50) is straightforward. Also, note that system factorization only has to be performed once and can be reused in each iteration as both $\mathbf{G}$ and $\mathbf{W}_n$ are discretized on the initial mesh $\mathbf{x}$. Hence, they are independent of the iteration k. This means that in every step the solution $\mathbf{u}^{k+1}$ is obtained from back-substitution.

The particular energy type with $\mathbf{E} = \mathbf{G}$ representing a discrete gradient operator has a more general interpretation. The discrete energy (56) allows to fit scalar fields $\mathbf{u}$ on $\mathcal{M}$ to prescribed gradients $\mathbf{c}$. In the ARAP case, these are (component-wise) deformations $\mathbf{u}$ and prescribed deformation gradients $\mathbf{R}^\star$. In general, minimizers of this type of energy operator are described by the well-known discrete *Poisson equation* that is in this context equivalent to (51) for $\beta = 0$. Therefore, by using $\mathbf{W}_n$ instead of $\mathbf{A}_n$, *any* Poisson-type energy on triangle meshes can be regularized using our approach. (Poisson-type energies will also be used in the context of surface-based flow visualization in Chapter 9 and reviewed in more detail in Section 9.1.)

7.4.3 Surface Deformations

Not only $2d$ planar, but also $3d$ surface-based deformations can be regularized in the same way in our framework. For example, the planar ARAP deformations of the previous section can directly be extended to $3d$ surface deformations described by $\mathbf{x}_i, \mathbf{u}_i \in \mathbb{R}^3$ and regularized subsequently. The surface-based ARAP energies can be setup using vertex-based one-ring transformations as described by Sorkine and Alexa [SA07]. We provide an alternative formulation that is more triangle-centric.

For regularization, we require 3×3 deformation gradients in a common coordinate system. We compute $\tilde{\mathbf{F}}_t = \mathbf{G}_t \mathbf{U}_t$ using a 3×3 extension of the gradient operator (53) for triangles $t = (i, j, k)$

$$\mathbf{G}_t = \begin{bmatrix} (\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} \\ (\mathbf{x}_k - \mathbf{x}_i)^{\mathrm{T}} \\ \mathbf{n}_t{}^{\mathrm{T}} \end{bmatrix}^{-1} \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \tag{58}$$

where $\mathbf{n}_t$ is the normalized triangle normal of t in the original $\mathbf{x}$ coordinates (cf. Appendix A.1). The gradients of the coordinate functions of $\mathbf{u}$ computed by this operator lie in the triangle planes of $\mathbf{x}$ in $\mathbb{R}^3$. However, closest rotations to $\tilde{\mathbf{F}}_t$ are not well-defined yet, because $\tilde{\mathbf{F}}_t$ are singular in direction $\mathbf{n}_t$. One can combine the ARAP local rotation optimization with the handling of this singularity in a single step by using the 3×3 SVD $\tilde{\mathbf{F}}_t = \tilde{\mathbf{S}} \tilde{\mathbf{\Sigma}} \tilde{\mathbf{T}}^{\mathrm{T}}$: the closest 3D rotation to the deformation gradient is $\mathbf{R}_t^\star = \mathbf{S}\,\mathbf{T}^{\mathrm{T}}$, where $\mathbf{S}$ and $\mathbf{T}$ are derived from $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{T}}$ by replacing their column that corresponds to the vanishing singular value in $\tilde{\mathbf{\Sigma}}$ with $\mathbf{n}_t$ and $\mathbf{n}_t^k$, respectively. Here, $\mathbf{n}_t^k$ is the normalized normal of t in the deformation $\mathbf{u}^k$ at iteration k. Given local rotations optimized this way as target deformation gradients, the global deformation reconstruction in each iteration is analogous to (56) with (57). In fact, the resulting Poisson system is equivalent to (56), except that now $n = 3$ and the $3d$ versions of $\mathbf{G}_t$ and $\mathbf{R}_t^\star$ are used to setup $(\mathbf{E}, \mathbf{C}^k)$. All properties are inherited from the $2d$ case. Most importantly, this way also surface deformations based on $3d$ ARAP energies can directly be regularized.

7.4.4 Continuous Deformations

All deformation examples discussed so far are described by one single map from the initial coordinates $\mathbf{x}$ to the coordinates of the deformed solution $\mathbf{u}$, i.e., by a single-step deformation. This map is obtained directly as the minimizer of some deformation energy. An alternative way to describe deformations are continuous formulations where $\mathbf{u}_i(s) \in \mathbb{R}^2$ represent time-dependent tangent velocity fields at a specific time s (cf. Chapter 3). Deformations are then obtained by integrating

an initial shape $\mathbf{x}$ along $\mathbf{u}(s)$ forward in time, giving time-dependent, continuous deformations $\mathbf{x}(s)$. This requires evaluation of $\mathbf{u}(s)$ on different deformed $\mathbf{x}(s)$, and the vector fields are obtained by optimizing energies in $\mathbf{u}$.

Solomon et al. [SBBG11a] propose discretized energies that are quadratic in $\mathbf{u}$ and yield near-isometric planar deformations. These as-Killing-as-possible (AKAP) energies are of the form

$$E_{\mathcal{P}}(\mathbf{u}(s)) = \sum_{t \in \mathcal{T}} A_t \left\| \mathbf{J}_t + \mathbf{J}_t{}^{\mathrm{T}} \right\|_F^2$$

(cf. the continuous version of Equation (6)). This energy measures the squared distance of the 2×2 vector field Jacobians $\mathbf{J}_t{}^{\mathrm{T}} = \mathbf{G}_t \, \mathbf{U}_t$ from Jacobians of exact isometric vector fields (known as Killing vector fields), which are anti-symmetric. Here, we reuse the gradient operator (53), with the difference that it has to be discretized on the current $\mathbf{x}(s)$ for each evaluation of the vector field. Vector field Jacobians, which represent a different gradient type, are constant per triangle, and therefore the energy is also local per triangle and fits our setting. The local energies $\|\mathbf{E}_t \, \mathbf{u}_t\|^2$ are given by the local 4×6 energy operators

$$\mathbf{E}_t = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \mathbf{H}_t \,, \tag{59}$$

and the global $(\mathbf{E}, \mathbf{c})$ are equivalent to (54) using these local operators and $n = 4$.

The same formulation also holds for the continuous deformation of $3d$ volumetric shapes. In this case we have Jacobians $\mathbf{J}_t \in \mathbb{R}^{3\times3}$, and A_t is the volume of a tetrahedron t.

In fact, the family of generalized anisotropic metric vector field energies proposed in Section 3.3 can be regularized in a similar way: using the energy expressions derived in Section 3.4.2, which are defined by quadratic forms $\mathbf{M}_\phi$, we obtain $\mathbf{E}_t = \mathbf{S}_\phi \, \mathbf{H}_t$ instead of (59), where $\mathbf{S}_\phi$ are (real symmetric) matrix square roots $\mathbf{S}_\phi \, \mathbf{S}_\phi = \mathbf{M}_\phi \left(= \mathbf{S}_\phi{}^{\mathrm{T}} \, \mathbf{S}_\phi \right)$ [1]. For the $2d$ isotropic energy special cases (22) the unique positive-semi-definite square roots are given by

$$\mathbf{S}_{\phi_{\mathrm{AKVF}}} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \qquad \mathbf{S}_{\phi_{\mathrm{METR}}} = \begin{pmatrix} \sqrt{2}+1 & 0 & 0 & \sqrt{2}-1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ \sqrt{2}-1 & 0 & 0 & \sqrt{2}+1 \end{pmatrix},$$

1 The Cholesky factors of $\mathbf{M}_\phi$ cannot be used because $\mathbf{M}_\phi$ is only positive-semi-definite.

$$\mathbf{S}_{\phi_{\text{CONF}}} = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{S}_{\phi_{\text{AUTH}}} = \frac{1}{2} \begin{pmatrix} \sqrt{2} & 0 & 0 & \sqrt{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix}.$$

For general anisotropic and higher-dimensional quadratic forms $\mathbf{M}_\phi$ (cf. Equation (21)) the matrix square root can be computed using, e.g., the algorithm of Higham and Wilkinson [HJW87].

We note that the regularization technique presented in this chapter is closely related to the vector field energy smoothing already presented in Section 3.4.4, where we also minimize first-order energy variation. However, the formulation of this chapter is based on a more general problem description and is therefore applicable to a larger number of energy types, i.e., it is not restricted to deformation vector field energies only.

7.4.5 Surface Parametrizations

The parametrization of a surface mesh can be considered as the computation of a map from the surface coordinates $\mathbf{x}_i \in \mathbb{R}^3$ to coordinates $\mathbf{u}_i = (u_i, v_i)^\mathrm{T} \in \mathbb{R}^2$ in the parametrization domain. This map should minimize some type of shape distortion.

AS-CONFORMAL-AS-POSSIBLE. Discrete near-conformal parametrizations are obtained by satisfying the Cauchy-Riemann conditions $\nabla u = \nabla v^\perp$ in least-squares sense: this yields the least-squares conformal maps (LSCM) by Levy et al. [LPRM02]. (A related approach with identical results was proposed simultaneously by Desbrun et al. [DMA02].) The corresponding discrete and integrated energy can be written as

$$E_\mathcal{P}(\mathbf{u}) = \sum_{(i,j,k)=t\in\mathcal{T}} A_t \left\| \mathbf{G}_t \left(u_i, u_j, u_k\right)^\mathrm{T} - \mathbf{R}_t \, \mathbf{G}_t \left(v_i, v_j, v_k\right)^\mathrm{T} \right\|^2$$

(see, e.g., [BKP*10]). Here, we use the surface-based gradient operator (58) for gradients in a common $3d$ coordinate system, and $\mathbf{R}_t(\cdot) \equiv \mathbf{n}_t \times (\cdot)$ are 3×3 rotation matrices performing $\pi/2$ rotations of vectors in each triangle plane. The corresponding local energies $\|\mathbf{E}_t \, \mathbf{u}_t\|^2$ are given by the 3×6 operators

$$\mathbf{E}_t = \begin{bmatrix} \mathbf{G}_t & -\mathbf{R}_t \, \mathbf{G}_t \end{bmatrix} \mathbf{P}$$

with an appropriate permutation $\mathbf{P}$ for the coordinate-wise gradient computation. The global energy is described by $(\mathbf{E}, \mathbf{c} = \mathbf{0})$, where the linear operator is setup as in (54) but using $\mathbf{E}_t$ as described here with $n = 3$.

AS-RIGID-AS-POSSIBLE. As shown by Liu et al. [LZX*08], the ARAP energy used in Section 7.4.2 can also be employed to compute surface parametrizations with low distortions. To perform the local $2d$ rotation optimization we compute the constant basis functions of a $2d$ gradient operator $\mathbf{G}_t$ in the parametrization domain. This can be done by transforming each triangle $t = (i, j, k)$ separately and isometrically from $3d$ to $2d$ and computing the 2×3 operators by

$$\mathbf{G}_t = \left(\mathbf{S}\,\mathbf{R}_t \begin{bmatrix} \mathbf{x}_j - \mathbf{x}_i & \mathbf{x}_k - \mathbf{x}_i \end{bmatrix} \right)^{-\mathrm{T}} \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} . \tag{60}$$

Here, $\mathbf{R}_t$ is the rotation that transforms $\mathbf{n}_t$ into $(0, 0, 1)^{\mathrm{T}}$ to align each triangle locally with the xy-plane, and $\mathbf{S} = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right)$ projects the transformed edge vectors into this plane. By transforming each triangle by $\mathbf{S}\,\mathbf{R}_t$ we essentially obtain *undeformed* $2d$ reference triangles for the original $3d$ surface triangles. A parametrization $\mathbf{u}^k$ at iteration k then defines 2×2 deformation gradients $\mathbf{F}_t^k = \mathbf{G}_t\,\mathbf{U}_t^k$, and closest rotations $\mathbf{R}_t^\star$ can be fitted as before. The global Poisson-based reconstruction of $\mathbf{u}^{k+1}$ then is *identical* to (56) and (57), which can be regularized with $\beta > 0$. Note that the definition of the global $2d$ gradient operator $\mathbf{G}$ (57) by using (60) does not require a continuous reference mesh in the parametrization domain. In fact, the connectivity of the mesh $\mathcal{M}$ defines continuity of the solution.

7.5 RESULTS

In this section, we present experimental results and provide comparisons. We also refer to the accompaning material of [MRT13b], which contains a `Matlab` reference implementation of our approach, a video showing additional results, as well as all result meshes of this chapter [2]. If not stated otherwise, boundary constraints are "hard" interpolation constraints, which can be interpreted as pins or handles on vertices. They are rendered as blue spheres. The color-codes visualize the local energies $E_{\mathcal{P}}$ per triangle [3]. Note that in general the plain energy cases ($\beta = 0$) correspond to results of the original methods (e.g., published in [LZX*08]). Results with $\beta > 0$ are our regularized versions with *identical* boundary constraints.

Figure 49 shows the effect of our regularization for different types of $2d$ deformations for different shapes. We minimize the ASAP, LARAP (Section 7.4.1), and ARAP (Section 7.4.2) energies for $\beta \in \{0, 0.2, 0.4\}$. The local variation of energy decreases for increasing amount of regularization. The plain energy solutions suffer from high distortion and flipped triangles near the handles for both ARAP energy types, which was also observed by Jacobson et al. [JBK*12, Figure 7]. Regularization corrects these artifacts, and especially the LARAP results are of improved quality. As an additional positive side-effect, global self-intersections of

2 The accompaning material is located in the additional material folder `addmaterial/se`.

3 Some figures are best viewed in the electronic version of the thesis. Please zoom in closely.

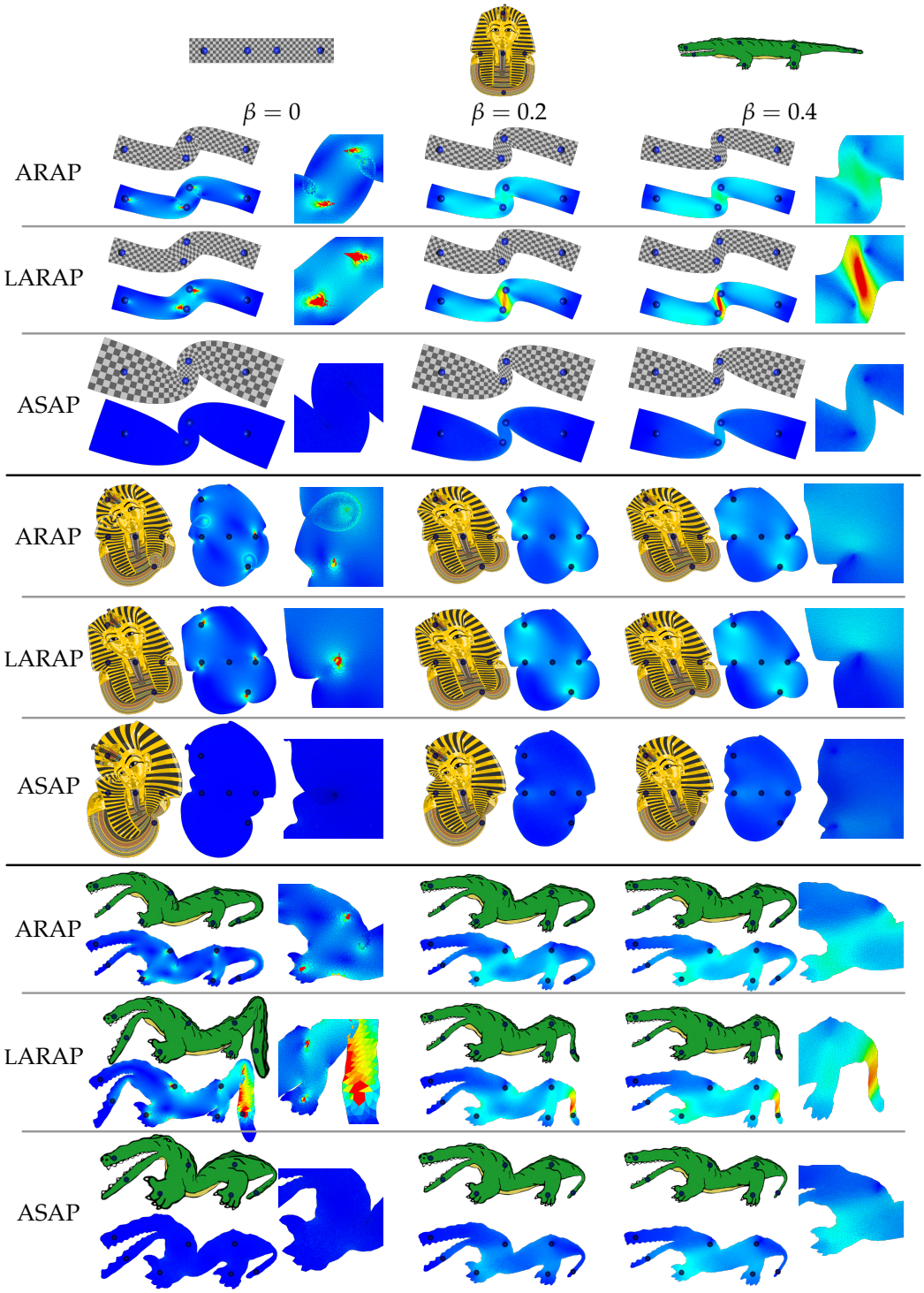


Figure 49: Regularized $2d$ Deformations. The initial shapes (top) are deformed using the different original ($\beta = 0$) and regularized ($\beta > 0$) $2d$ deformation energies. The color-coded images visualize local energies of $E_{\mathcal{P}}$ on each triangle (low •, high • energy). For $\beta \in \{0, 0.4\}$ closeups visualize energies at critical regions that contain high distortions as well as local and global self-intersections.

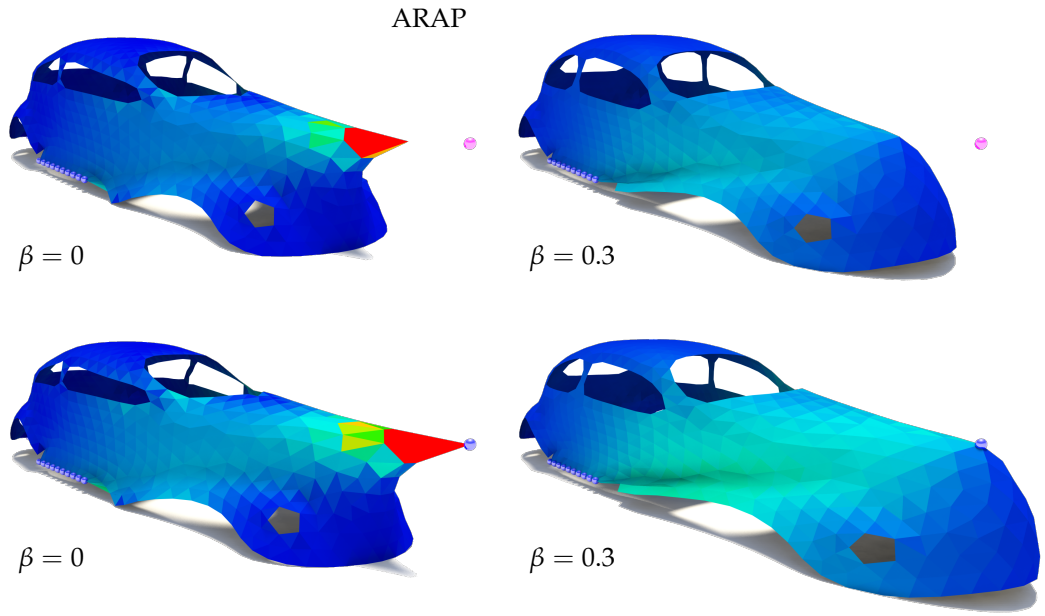


Figure 50: Regularized ARAP Surface Deformation. The side and rear (not shown) of the beetle surface are fixed and a vertex on the engine hood is moved to deform the shape such that the ARAP energy is minimized. We show results for both "soft" (•, top) and interpolating "hard" handle constraints (•, bottom). Our regularized solution is artifact-free at the handles in both cases.

plain deformations, which are most notable in the ASAP case, are repressed in the regularized solutions, although the regularization is not designed to directly prevent this type of artifact.

Figure 50 shows simple examples of $3d$ surface deformations minimizing the ARAP energy. The side and the rear (not shown) of the beetle surface were fixed, and a vertex on the engine hood was moved to deform the shape globally. We show results for both "soft" constraints, which only approximate the handle, and interpolating "hard" handle constraints. In both cases, the regularized energy yields smooth deformations respecting the particular constraints, whereas the original energy suffers from localized artifacts at the handle vertex. The experiment supports the claim that our regularization also works for $3d$ surface deformations. In particular, ARAP surface deformations are enhanced.

Figure 51 shows examples for parametrizations minimizing LSCM and ARAP energies. We compare plain ($\beta = 0$) and regularized ($\beta > 0$) energies. The boundary of the beetle parametrization (left) was fixed in the plane, and the positions of four *interior* vertices were prescribed in the plane. The regularized solution has no flipped triangles near the handles. The gargoyle model (right) was cut open and mapped to the plane. (The same cut-open mesh was used in the original work [LZX*08].) Again, *interior* vertices were fixed. This is an action that is highly relevant in practice for locally "fine-tuning" maps. At the same time this is highly problematic as local "spikes" and fold-overs arise quickly,

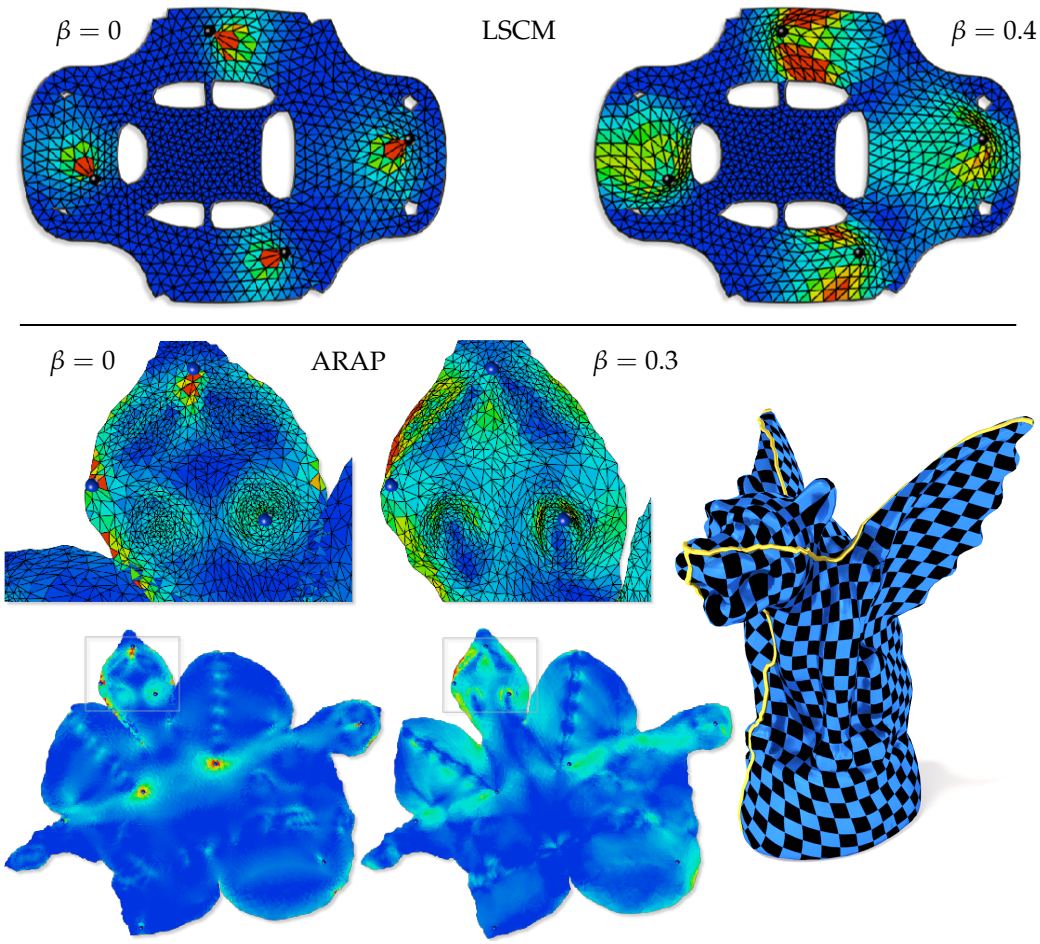


Figure 51: Regularization of Parametrizations. Top: LSCM of the beetle with boundary and four *interior* vertices fixed in the plane. Bottom: ARAP parametrization of the gargoyle with constraints on *interior* vertices. In contrast to the original solutions our regularized solutions are free of triangle flips (see closeups).

which, until now, demanded nonlinear methods. Our regularization ($\beta = 0.3$) handles these problems and provides a valid solution without flipped triangles (see closeups).

We apply our regularization to continuous deformations that minimize the AKAP energy in Figure 52. To emphasize total distortions we color-code isometric distortions E^{nd}_{isom} relative to the original shape (cf. Appendix B). Plain ($\beta = 0$) deformations show notable artifacts identified by Solomon et al. [SBBG11a]. Their filtering method requires the solution of an *additional* Poisson system that introduces additional distortions and only approximates user constraints (cf. Sections 3.4.4 and 3.6). Our method also avoids deformation artifacts. Additionally, we only require a single linear system solve and guarantee exact constraint satisfaction. The tetrahedral eagle model is an example of our regularization method applied to volumetric domains.

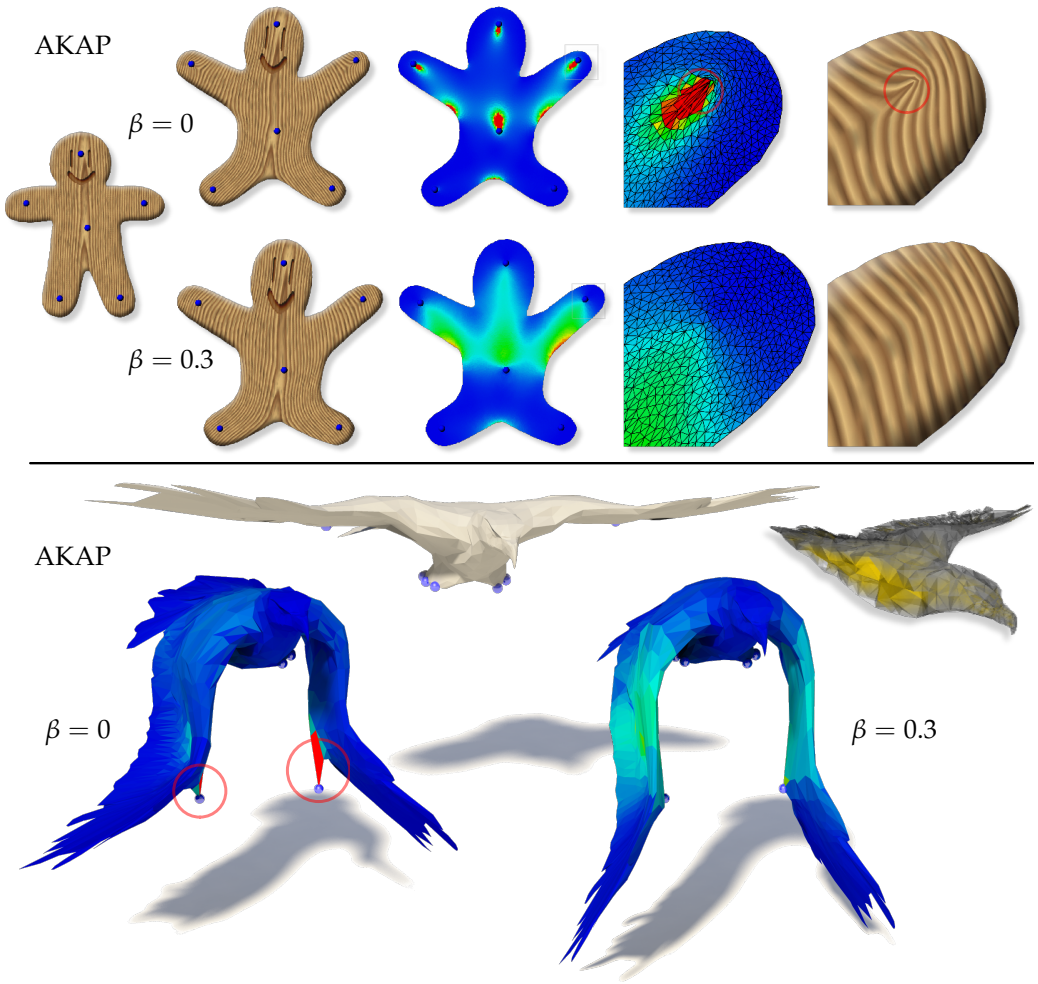


Figure 52: Regularized Continuous Deformations. The AKAP energy is minimized and the resulting vector field is integrated for a planar triangle mesh (top) and a volumetric tetrahedral mesh (bottom). The color-coded visualizations show local isometric distortions E^{nd}_{isom} relative to the rest-pose. Artifacts of the original energies ($\beta = 0$) are highlighted (●).

We compare our approach to standard Tikhonov regularization in Figure 53. The $2d$ giraffe shape is deformed, and a parametrization of the 3-balls surface is computed both by minimizing an ARAP energy. The Tikhonov regularizer is defined as $E^{\mathcal{T}}_{\mathcal{R}}(\mathbf{u}) = \int_{\mathcal{D}} \|\nabla \mathbf{u}(\mathbf{x})\|^2_F \, d\mathbf{x}$, i.e., first-order variation of the coordinate function *solution* $\mathbf{u}$ is penalized. It is applied for $\beta = \frac{1}{4}$ and $\beta = \frac{1}{2}$. We compare to our regularizer $E_{\mathcal{R}}$ for $\beta = \frac{1}{2}$ (right for each example). The undesired effect of Tikhonov regularization is clearly visible: the solution is smoother but it "shrinks". In fact, Tikhonov regularization modifies the solution *even* if handles are not moved at all due to the energy-unawareness. Our proposed regularization does not show this effect because both the energy and the energy-aware regularization terms vanish on the initial pose. Not shown are other typical choices of $\mathbf{\Gamma}$ like penalizing second-order variation ($\mathbf{\Gamma} \equiv \Delta$) or penalizing the magnitude of the solution

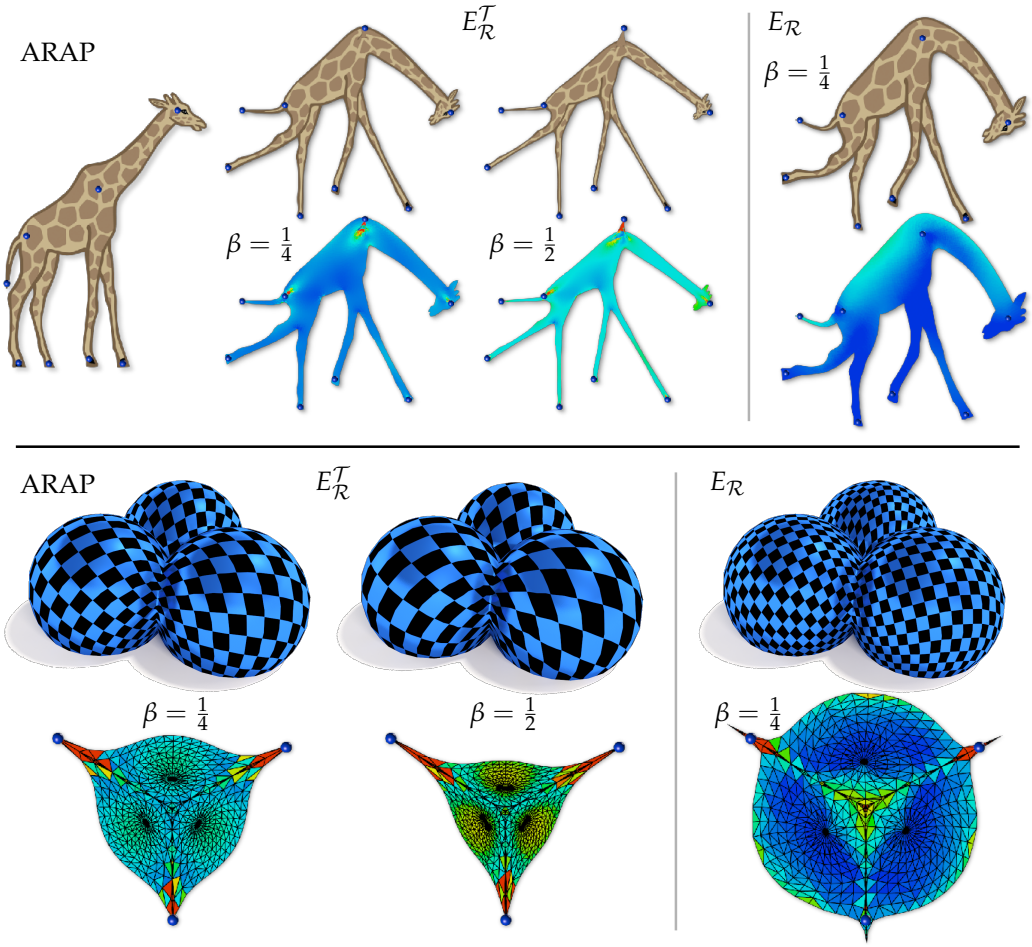


Figure 53: Regularization Comparison. We compare standard Tikhonov regularizer $E_{\mathcal{R}}^{\mathcal{T}}$ with $\mathbf{\Gamma} \equiv \nabla$ for $\beta \in \left\{\frac{1}{4}, \frac{1}{2}\right\}$ and our regularizer $E_{\mathcal{R}}$ for $\beta = \frac{1}{4}$. The ARAP energy is minimized to deform the giraffe and to obtain a parametrization of the 3-balls surface.

($\mathbf{\Gamma} \equiv \mathbf{I}$): they do not remedy undesired behavior either. This general statement holds similarly for minimizing other deformation energies.

Figure 54 compares our results with the nonlinear bounded distortion (BD-)maps proposed by Lipman [Lip12]. We consider near-conformal deformations from minimizing BD-LSCM and ASAP energies and near-rigid deformations from minimizing BD-ARAP and ARAP energies. Without regularization ($\beta = 0$) we obtain the undesired artifacts that were pointed out by Lipman (red circles). Adding our regularization $\beta > 0$ effectively removes artifacts and yields valid shapes without local self-intersections. Interestingly, our solution with the smoother energy distribution for ASAP ($\beta = \frac{1}{4}$) shows significantly less global deformation compared to BD-LSCM. The BD-ARAP and the regularized ARAP deformation show comparable results: regularized ARAP yields a smoother energy distribution and a thinner neck of the dinosaur model. The $\beta = 0.085$ value

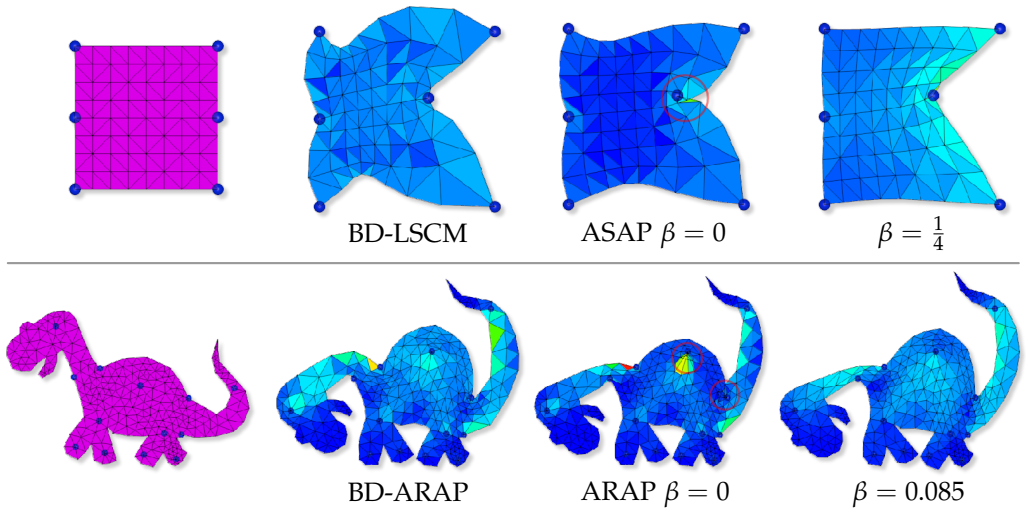


Figure 54: Comparison with BD-maps [Lip12, Figures 2 and 3]. Near-conformal (top) and near-rigid (bottom) deformations of the initial models (•) using bounded distortion mappings and our corresponding regularized energies ($\beta > 0$). Artifacts of the original energies ($\beta = 0$) are highlighted (•).

of this regularization is the smallest regularization parameter for which the deformation is artifact-free. Note that there are constraints on *internal* vertices. This leads to a very visible artifact for the unregularized ARAP ($\beta = 0$). We emphasize that we compare our approach to the original results of [Lip12, Figures 2 and 3], which are very coarsely tessellated because the nonlinear BD-maps optimization is limited to a low number of triangles. Our approach can efficiently be applied to much larger meshes.

The experiment in Figure 55 shows that the proposed regularization is not only problem-specific but also largely independent of the particular tessellation of a mesh. We apply an ARAP deformation to four different meshes that all represent the same shape geometry. Note that this particular shape with thin parts and relatively long boundary is a nontrivial benchmark. The tessellation is challenging as triangle area ratios vary up to $1.7 \cdot 10^3$, and the triangle circumcircle to incircle radius ratio is up to 38.8. The results are similar for all tessellations, we show this for two different values of β. This indicates that our energy discretization in Section 7.3 is reasonable.

7.6 DISCUSSION

We tested our regularization on a number of typical problems in geometry processing and showed experimental results. Our approach fits problems in this application domain, but it is generally not limited to geometry processing.

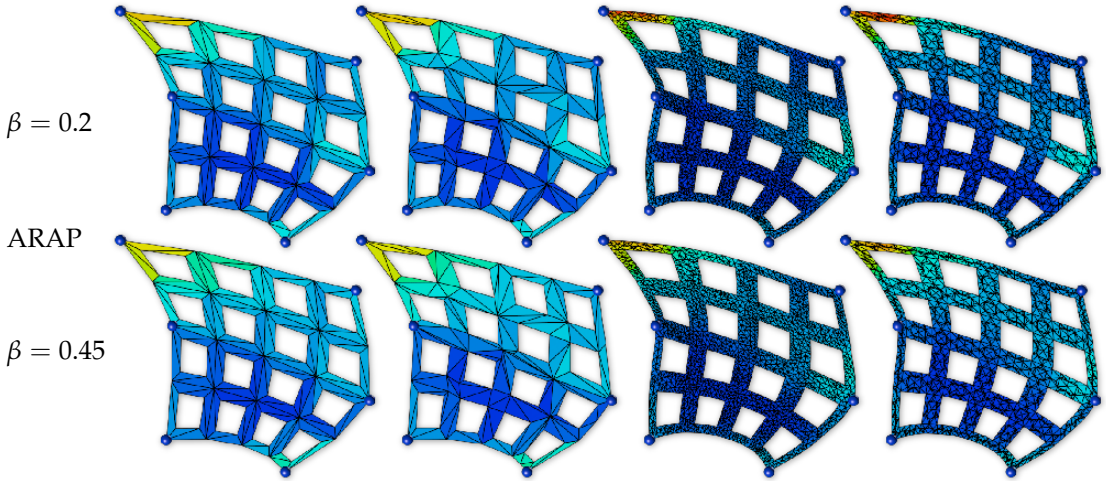


Figure 55: Mesh Dependence. We deform a regular grid that is differently tessellated using the same deformation constraints. We show results for the regularized ARAP energy for two different β values. Other regularized energies show similar results.

TIKHONOV REGULARIZATION. It is noticeable that in particular for these applications, regularization is not considered at all. The reason is probably simple: first, standard Tikhonov regularization is available but it does not fit the problems and yields rather undesired results. Second, most problems are *not ill-posed*, and any regularization might increase the energy $E_\mathcal{P}$ that characterizes the problem. The first argument seems true as indicated by our experiments. This holds also for the very popular Tikhonov-type regularization to prefer "soft constraints", i.e., a penalty term as regularizer, over "hard constraints", i.e., constraint interpolation by elimination, in order to suppress or smooth out artifacts near handles. Soft-constraints, however, have their drawbacks: artifacts only appear later but are not effectively suppressed (see Figure 50), and in addition, the solution may "float" in the domain. We refer to, e.g., [SBBG11a]. There are exceptions to the second argument: for instance, minimizing the AKAP energy [SBBG11a] is an example for a problem that is not ill-posed, though it inherently requires regularization. However, there the standard approach of incorporating smoothness of the solution, i.e., standard Tikhonov regularization, would lead to undesirable results. For this reason, Solomon et al. propose a Poisson-based smoothing step as a post-process.

Hence, in comparison to standard problem-unaware Tikhonov regularization, our problem-specific approach provides the (more) desirable results. Even more, the regularization enables imposing "hard" interpolation constraints not only on the boundary. Our experiments show that constraints *inside* the domain are possible, at least to some extent. This makes minimizers of regularized quadratic energies competitive with solutions from nonlinear optimization. Of course there is no guarantee that, e.g., triangle flips are avoided.

SOBOLEV NORM. Our regularization norm can be interpreted as a weighted Sobolev H^1 norm, i.e., as a norm in the function value and its first-order variation. This norm was used by Eckstein et al. [EPT*07, Section 4] for the specific problem of regularized gradient flow computation. In contrast, we apply similar norms to energies that do not only define gradient flows, but any type of quadratic optimization. This is a key difference that enables the generalized application of our method to a variety of geometry processing problems.

NONLINEAR OPTIMIZATION. Quadratic energies are attractive as models because their minimization is straightforward and numerically efficient. In particular, in geometry processing, operators are often linearized, which is efficient but induces a model error. (Linearization of rotations is an example.) Dropping linearization requires nonlinear optimization. Additionally, the range of problems that can be modeled by quadratic energies is of course limited. Hence, nonlinear models are required or preferred in many settings. A prominent example is imposing general constraints or bounds on variables. This is often done to guarantee properties of the solution. We consider the recent work by Lipman [Lip12] as a typical example: a sophisticated nonlinear model guarantees bounded distortion and validity of piecewise linear (deformation) maps. The type of distortion is generic, and validity refers to the absence of triangle flips. This is much more than can be expected from a linear setting, i.e., quadratic energy minimization, but it requires the more expensive solution of a nonlinear problem, i.e., one has to invest more time and/or reduce the problem size. Note that in this case the bounds on variables, which restrict the solution space, could be interpreted as a nonlinear, (but still problem-unaware) regularization. If a feasible solution exists in this space then it satisfies the guarantees by construction. Having similar guarantees is generally not possible with our setting of quadratic energy minimization. The comparisons by Lipman reveal these problems, they show up early and frequently for typical deformation tasks. This changes, however, for quadratic energies that are regularized by our method. There are still no guarantees, but our experiments show that the regularized solutions get significantly closer to results from nonlinear optimization. And if they break, i.e., they yield an invalid result, this happens relatively late, meaning for extreme deformations imposed by "large" movement of handles.

We conclude that although our regularization cannot provide similar guarantees as nonlinear optimization, minimization in the regularized function space yields competitive results that seem to remain valid for a considerable range of input constraints. This is not possible without or even with standard regularization.

COST. The setup of the matrix $\mathbf{W}_n$ and the regularized normal equations (51) takes more operations than the initial setup (49). More importantly, for sparse operators the new system matrix has more nonzero entries. Our experiments show that there is no significant effect on sparsity of the Cholesky factors $\mathbf{L}$

and on overall timings compared to having no regularization. Moreover, the overhead is on par with that of Tikhonov regularization penalizing first-order variation of the solution.

LIMITATIONS. Our regularization is restricted to quadratic energies on discrete domains. This is a standard scenario for geometry processing. We are not limited to such scenarios but it may be the case that the penalization of energy variation pays off in particular for this sort of applications. The penalization of variation in the solution may be suited better for other applications. In particular, a more general regularization based on the (generalized) singular value decomposition with filtering singular values and the possibility for parameter estimation should be preferred for general ill-posed and inverse problems [Han10]. However, for geometry processing this is generally less suited as global SVD computations turn out to be too expensive.

We show empirically that the minimization of regularized quadratic energies tends to get close to solutions from nonlinear optimization. There is of course no guarantee for such behavior, moreover, there is no a priori optimal choice for β. In particular, the regularized solutions tend to break much later but they will break, i.e., represent an invalid result, for some extremal constraints. Regularization of quadratic energies is therefore not a substitute for nonlinear models with guarantees, but it may complement these models: the regularized solution can be computed quickly for a number of β values. If it is valid (and satisfies any other criteria) it may be used directly. Otherwise, it may serve as an initial guess for a nonlinear solver.

7.7 SUMMARY

Many relevant problems can be formulated as the minimization of a quadratic energy. In this chapter, we proposed a generic construction of *regularized* quadratic energies: the input consists of *any* quadratic energy that is integrated in a discrete domain, which is given as a partition of simplices. The output consists of a new energy that incorporates a *problem-specific* regularization. This construction is generic and can be applied in a straightforward way if the problem is modeled in a standard form (49). Essentially, we can interpret our regularization as a simple change of the norm that is minimized. The key observation of our approach is that the energy, which characterizes the problem, should *explicitly* contribute to the regularization. This makes our regularization different from standard methods based on Tikhonov regularization: it enforces low energy variation instead of a generically smooth solution. We applied our regularization to a number of geometry processing applications. Our experiments reveal the advantages of incorporating regularization, and they indicate that our regularizers are appropriate for various energies and fit a range of different and

relevant problems. Promising directions for further research include the efficient automatic selection of the regularization weight β and the feasibility analysis of higher-order problem-specific regularizers.

Part II

SHAPES DEFINED BY VECTOR FIELDS

8

OVERVIEW OF SURFACE-BASED FLOW VISUALIZATION

In the first part of this thesis, we considered vector field-based *manipulations* of shapes in the context of geometry processing problems. In this second part, we now approach shapes that are *defined* by vector fields.

In particular, we consider vector fields that represent the velocity field of complex flows. These fields are practically relevant as a variety of scientific, engineering, and medical application areas study problems in which $3d$ flow phenomena play a key role for their comprehension and solution. In contrast to the first part of the thesis, the flow vector fields are given a priori and are not computed for a particular shape manipulation. Rather, vector fields are subject to further analysis by extracting shapes as features that are directly related to the underlying flow data. In computer graphics, these problems are studied by the flow visualization research field.

Different flow-defined types of shapes can be used for the visualization of integrated flow features. The most common geometry-based flow visualization techniques either rely on univariate curves or bivariate surfaces. Integral curves like stream lines are defined by steady flows, whereas streak, path, and time lines are defined by unsteady flows. Univariate shapes are well-accepted structures for the visualization of $3d$ vector fields [WHT12].

In this part of the thesis, we focus on bivariate surface-based flow visualization techniques. Integral surface-based approaches consider stream surfaces in steady flows, as well as streak, path, and time surfaces in unsteady flows. Compared to line-based visualizations, integral surfaces are often better suited to reveal laminar flow regions and regions with convergent or divergent flow behavior. Additionally, the perception of surfaces is often superior compared to sets of curves because advanced surface shading techniques can be applied. On the other hand, surface-based approaches generally turn out to be more complex due to the higher intrinsic dimensionality of surfaces. Still, properties of integral surfaces are well-studied. In fact, deep and comprehensive research has been done on the problem how to integrate and how to graphically represent integral surfaces. However, in practice many important professional engineering tools do rarely provide domain experts with such advanced visualization techniques:

a lot of tools (like, e.g., the ANSYS CFD-Post package [2]) provide users only with basic visualization techniques, e.g., iconic or glyph-based, slice-based, or univariate stream line or path line visualizations. Visualizations by integral surfaces are infrequently used in practice despite their advantages over these basic visualizations [MLP*10].

A major reason for the less prominent representation in professional tools might be the more complex interaction that is required for the specification of the surfaces to be visualized: stream surfaces are usually defined by seed curves from which the surfaces are constructed by integration. The exact placement of these curves is based on assumptions and experience of domain experts and is usually performed manually in *interactive* sessions. Most often the choice of seed curve geometry is very limited, e.g., to simple straight line segments or circles. Additionally, not every seed curve is suitable to define stream surfaces: in fact, seed curves that the user erroneously positions tangentially to the flow result in degenerate stream surfaces. These factors make interactive seed curve specification challenging and time-consuming in practice. Moreover, a lot of methods exist for *automatic* seeding of stream lines, whereas only a few limited approaches exist to automatically extract relevant stream surfaces.

Hence, in order to make surface-based visualizations more accessible to use in practice, it is necessary to provide users with additional stream surface specification tools that overcome limitations of traditional interactive seed curve manipulation. Alternatively, completely automatic methods for the selection of most relevant stream surfaces can eliminate the need for manual interaction. Interactive tools are useful if users are only interested in specific data set regions where certain flow phenomena are already expected. Automatic selection methods, on the other hand, are beneficial for flow field exploration and unsupervised flow analysis. Here, no prior knowledge of the data set is required.

To address these stream surface *specification* and *selection* problems, we propose two new *interactive* and *automatic* approaches that facilitate the application of surface-based methods for the detection and analysis of relevant flow features.

In Chapter 9, we apply Poisson-based techniques to the specification and extraction problem of flow-aligned surfaces. Poisson-based methods are well-known in image and geometry processing, but have not yet been applied to flow visualization problems. We support the extraction of both flow-tangential surfaces and flow-orthogonal surfaces. In contrast to traditional integration-based techniques, our method is deformation-based and steered by flux optimization criteria. Surfaces can be positioned interactively by the user. They are deformed in real-time according to the local flow. The same Poisson-based framework can be applied to obtain parametrizations of flow-aligned surfaces. This way nontrivial seed structures for standard integration-based flow visualization methods are simple to specify. Additionally, we present illustrative and animated texture-based flow

visualizations of stream and path surfaces, which are based on the same Poisson-based computational toolkit.

In Chapter 10, we present a fully automatic approach for the global selection of relevant stream surfaces. Stream surface relevance is measured by a new intrinsic surface-based quality criterion that prefers surfaces for which the flow is aligned with principal curvature directions of the shape. This concept corresponds to curvature-based surface fairing known from geometry processing. The problem of seed structure selection can then be reduced to the computation of simple minimal paths in a weighted graph that spans the domain. A global simulated annealing-based optimization method finds smooth seed curves of globally near-optimal stream surfaces. The resulting stream surfaces are very similar to surfaces manually selected by visualization experts.

8.1 RELATED WORK

There is a large body of research on integration-based flow visualization. We continue to summarize on surface and texture-based flow visualization methods that are most related to our approaches. For a complete overview on the flow visualization field we refer to a number of recent surveys: Post et al. review feature and topology-based techniques [PVH*02, PVH*03], Laramee et al. survey dense visualization methods [LHZP05], McLoughlin et al. give a summary on integration-based approaches [MLP*10], and Pobitzer et al. give an overview on topology-based methods for unsteady flows [PPF*11].

8.1.1 Surface-based Flow Visualization

In Chapter 9, we introduce an interactive flux-optimizing and deformation-based method to extract both *flow-tangential* and *flow-orthogonal* surfaces. The classic approach to computing the former are based on *surface integration* — a problem that is well-understood [MLP*10]: Hultquist pioneered the development of stream surface integrators [Hul92], which were extended by Stalling [Sta98] towards critical point handling. More recent techniques focus on higher approximation and tessellation quality [SBH*01, GTS*04, GKT*08, SWS09], GPU implementation [STWE07], and topology-aware methods [PS09, SRWS10]. Schulze et al. present an advanced global time-scaling approach [SGRT12]. Van Wijk [vW93] and Stöter et al. [SWST12] propose implicit stream surface representations. Elaborate approaches for time-dependent path surface integration [STWE07, GKT*08], and streak and time surface integration [vFWTS08, BFTW09, KGJ09, WHT12] have been proposed recently. Edmunds et al. [ELC*12] review surface integration and perception-improving rendering methods. Stream surface integration is also regarded in the context of dynamical systems. The survey of Krauskopf

et al. [KOD*05] provides an overview on the extraction methods of (un)stable manifolds in general vector fields.

Most related to the orthogonal surfaces that we compute by flux maximization are the recently proposed as-perpendicular-as-possible surfaces (APAPs) by Schulze et al. [SRGT12], who integrate along a scaled vector field to obtain orthogonally aligned surfaces. A simpler method that does not consider energy minimization has been proposed before by Palmerius et al. [PCY09]. Additionally, orthogonal structures were used to improve animations [BW08], to seed stream lines on two-manifolds [RPP*09], and they are well-known in the computer vision community in the context of, e.g., shape-from-shading [CLL07].

Seed structures for stream geometries are usually placed manually. Seeds for stream ribbons and particles can be interactively moved around for real-time exploration [KKKW05]. The appearance of stream ribbons and surfaces can be enhanced using illustrative techniques as shown by Born et al. [BWF*10]. Hummel et al. [HGH*10] apply screen-space curvature approximations to enhance integral surface visualizations. Recent approaches also focus on the computation of frame-coherent and perceptually enhanced results for interactive exploration, such as Günther et al. [GBWT11, GRT13] for line renderings. These approaches only focus on the visual representation and not on selecting representative flow features. In Chapter 10, we present a method for the automatic selection of representative stream surfaces.

8.1.2 Texture-based Flow Visualization on Integral Surfaces

Another class of flow visualization techniques are dense texture-based methods. Van Wijk pioneered texture-based flow visualization using spot noise [vW91]. Line integral convolution (LIC) approaches [CL93, SH95] can be used to visualize the flow *on* integral surfaces in a dense way: recent methods range from the generation of texture atlases [LTWH08] to image-space techniques [LGSH06], which can be frame-coherent [HPW*12]. The survey by Laramee et al. [LHD*04] discusses various other texture-based techniques. More abstract illustrative visualizations of stream surfaces were proposed by Born et al. [BWF*10]. In the next chapter we introduce, inter alia, LIC-like flow structure visualizations and illustrations that are defined by globally flow-aligned parametrizations. Our method provides frame-coherence and supports stream and path surfaces as well as animation and requires no flow integration or line integral convolution.

8.2 DEFINITIONS AND NOTATION

In this part of the thesis, we make use of the following formal concepts: given is a 3*d steady* (i.e., not time-dependent) differentiable vector field $\mathbf{v}(\mathbf{x})$ over a

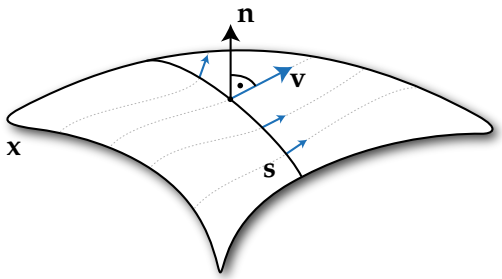
domain $\mathcal{D}$. Let $\mathbf{J}(\mathbf{x})$ be the Jacobian tensor field of $\mathbf{v}$. To ease the formal presentation, we assume $\mathcal{D} = \mathbb{R}^3$ to prevent boundary effects for integration. This is not a limitation: in practice, integration is stopped if the domain boundary is reached.

We again use the *flow map* concept $\boldsymbol{\phi}$ that maps to the location of a massless particle seeded at $\mathbf{x}_0$ after kinematic stream line integration in $\mathbf{v}$ over a time period τ:

$$\boldsymbol{\phi}^\tau(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^\tau \mathbf{v}(\mathbf{x}(t))\,\mathrm{d}t$$

with $\mathbf{x}(0) = \mathbf{x}_0$ and $\dot{\mathbf{x}}(t) = \mathbf{v}(\mathbf{x}(t))$ (cf. the flow map (34) for unsteady flows).

Let $\mathbf{x}(s,t)$ be a regular parametric surface in $\mathcal{D}$, and let $\mathbf{n}(s,t)$ be its unit normal. Then a defining property of $\mathbf{x}$ being a *stream surface* is $\mathbf{v}(\mathbf{x}(s,t))^\mathrm{T}\,\mathbf{n}(s,t) = 0$ for every (s,t) of the surface domain:



One way to construct a stream surface is to start integration in $\mathbf{v}$ from a parametric *seed curve* $\mathbf{s}(s)$. A stream surface $\mathbf{x}$ can then be described in explicit parametric form as the collection of all stream lines starting at $\mathbf{s}$:

$$\mathbf{x}(s,t) = \boldsymbol{\phi}^t(\mathbf{s}(s))\ .$$

Note that $\dot{\mathbf{s}} \times \mathbf{v}(\mathbf{s}) \neq \mathbf{0}$ has to hold for regular parametrizations of non-degenerate stream surfaces.

Closely related are *path surfaces* $\mathbf{x}_{t_0}$ in time-dependent flows $\mathbf{v}(\mathbf{x},t)$. These surfaces are also integrated from a seed curve $\mathbf{s}$ and given as the collection of all path lines seeded on $\mathbf{s}$ at time t_0 using the unsteady flow map (34):

$$\mathbf{x}_{t_0}(s,t) = \boldsymbol{\phi}_{t_0}^t(\mathbf{s}(s))\ .$$

9

INTERACTIVE SURFACE-BASED FLOW VISUALIZATION USING POISSON-BASED TOOLS

In this chapter, we show that *both*, interactive flow-aligning surface deformations and their parametrizations, can be computed efficiently and in a unified way by a Poisson-based optimization framework. Poisson-based methods are well-known with many applications in image and geometry processing. This is the first approach that applies this technique in the context of surface-based flow visualization.

In contrast to existing methods that require seed curve manipulation, our approach is based on direct interaction with entire surfaces that are near flow-aligned. We do *not* perform surface integration but instead propose an interactive surface *deformation-based* method that allows free positioning of surfaces. This way we achieve a more direct interaction with the resulting flow-aligned surfaces. The set of flow-aligned surfaces under consideration does not only contain flow-tangential surfaces, which correspond to standard stream surfaces, but also flow-orthogonal surfaces, which, in general, cannot be obtained by a simple surface integration. Our approach is generalized to computing both types of surfaces by deformations that are steered by flux optimization criteria. Our flow-tangential surfaces are particularly useful for *interactive* flow exploration. Flow-orthogonal surfaces are well-suited to provide seed structures for integration-based methods, because every embedded curve is flow-orthogonal by construction. These curves can be obtained as iso-contours of special flow-aligned parametrizations. Moreover, the same global parametrizations can further be used for, e.g., texture-based visualization, and we show how they define animated and view-independent LIC-like and illustrative renderings.

9.1 POISSON-BASED OPTIMIZATION AND MODELING

Both our deformation and parametrization approaches apply Poisson-based optimizations. For completeness and self-containment, we begin this chapter with a brief review of the general Poisson optimization method and its discretization for piecewise linear functions on triangular meshes.

Poisson-based surface deformations were first proposed by Yu et al. [YZX*04]. They apply the idea of Poisson-based image editing by Perez et al. [PGB03] in the context of geometric modeling. Using their basic technique a number of geometry processing methods were proposed: applications range from user-defined deformations [ZRKS05, SA07], pose interpolation [XZWB06], parametrizations [LPRM02, ZRS05, LZX*08] to deformation transfer [SP04, ZRKS05]. Surface reconstruction [KBH06, ZGHG10] and smoothly shaded drawings [OBW*08] also use related concepts. All these methods are based on similar Poisson-based computations. They differ in the way local gradients are transformed prior to function reconstruction. This chapter shows how gradient transformations are determined in a data-driven way to align surfaces to a flow field using deformations. We also refer to Section 7.4.2, where we study related discretized Poisson-type systems in the context of energy regularization.

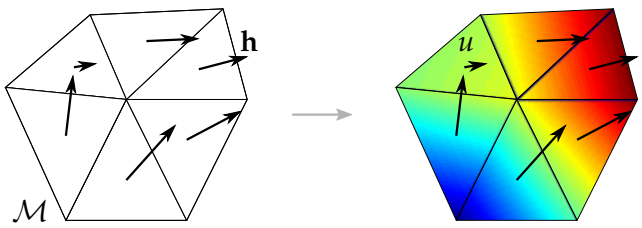
Given is a guidance vector field $\mathbf{h}$ in a two-manifold $\mathcal{M}$ with associated gradient operator ∇. Then Poisson-based methods search for the scalar field u on $\mathcal{M}$ whose gradient ∇u best fits $\mathbf{h}$ in least-squares sense. Formally, the continuous energy

$$e(u) = \int_{\mathcal{M}} \|\nabla u - \mathbf{h}\|^2 \, \mathrm{d}\mathbf{x} \tag{61}$$

is minimized. A computationally attractive and sufficient condition for a minimizer is given by the solution of Poisson's (point-wise) elliptic linear partial differential equation

$$\Delta\, u = \nabla^{\mathrm{T}} \mathbf{h}\,, \tag{62}$$

which has to hold for all $\mathbf{x} \in \mathcal{M}$ and needs to be solved subject to suitable boundary constraints (see, e.g., [AMR88]). In (62), Δ denotes the Laplace-Beltrami operator on $\mathcal{M}$, which can be represented by $\Delta = \nabla^{\mathrm{T}} \nabla$, i.e., as the divergence of the gradient.



We again discretize surfaces $\mathcal{M}$ by triangular meshes $(\mathcal{V}, \mathcal{T})$ defined by sets $\mathcal{V}$ of vertices with coordinates $\mathbf{x}_i \in \mathbb{R}^3$, and sets $\mathcal{T} \subset \mathcal{V}^3$ of triangles. Scalar fields u are piecewise linear functions on triangulated surfaces, i.e., on a triangle $t = (a, b, c)$ with coefficients u_i at the vertices i, we have $u(\mathbf{x}) = \sum_{i \in t} \phi_i{}^t(\mathbf{x})\, u_i$ with barycentric coordinates $\phi_i{}^t(\mathbf{x})$ as linear basis functions (see Appendix A).

The piecewise constant gradient field on t is given by $\nabla u = \sum_{i \in t} \nabla \phi_i{}^t \, u_i$. One way to compute the constant gradients $\nabla \phi_i{}^t$ of the basis functions is to solve the linear system

$$\begin{bmatrix} (\mathbf{x}_a - \mathbf{x}_c)^\mathrm{T} \\ (\mathbf{x}_b - \mathbf{x}_c)^\mathrm{T} \\ \mathbf{n}_t{}^\mathrm{T} \end{bmatrix} \begin{bmatrix} \nabla \phi_a{}^t & \nabla \phi_b{}^t & \nabla \phi_c{}^t \end{bmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} ,$$

where $\mathbf{n}_t$ is the unit normal of the triangle. If all scalar field coefficients u_i are stacked in a vector $\mathbf{u}$, one can assemble a $3\,|\mathcal{T}| \times |\mathcal{V}|$ gradient operator matrix $\mathbf{G}$ from the basis function gradients such that $\mathbf{G}\,\mathbf{u}$ is the vector of stacked scalar field gradients on each triangle (see Appendix A.1 for a derivation).

As scalar field gradients on triangular meshes are constant per triangle, the integral of (61) simplifies to an area-weighted sum of quadratic differences, and we can rewrite the integrated energy as

$$e(\mathbf{u}) = \|\mathbf{G}\,\mathbf{u} - \mathbf{h}\|^2_{\mathbf{A}} \, , \tag{63}$$

where $\mathbf{h}$ is the vector of stacked guidance gradient field vectors per triangle and $\mathbf{A}$ is a $3\,|\mathcal{T}| \times 3\,|\mathcal{T}|$ diagonal inner-product matrix of triangle areas that performs the integration (see Chapter 7). Then the optimal scalar field minimizing $e(\mathbf{u})$ is obtained by solving the linear system $\nabla\, e(\mathbf{u}) \stackrel{!}{=} \mathbf{0}$, which results in the normal equations

$$\mathbf{G}^\mathrm{T}\,\mathbf{A}\,\mathbf{G}\,\mathbf{u} = \mathbf{G}^\mathrm{T}\,\mathbf{A}\,\mathbf{h} \, . \tag{64}$$

This linear system is a discretization of the Poisson Equation (62) on triangular meshes with the discrete Laplace-Beltrami operator matrix $\mathbf{L} = \mathbf{G}^\mathrm{T}\,\mathbf{A}\,\mathbf{G}$ and the discrete divergence operator matrix $\mathbf{D} = \mathbf{G}^\mathrm{T}\,\mathbf{A}$. The matrix $\mathbf{L}$ is sparse and positive-semi-definite, its kernel is spanned by constant functions. Note that this discretization of the Laplace-Beltrami operator is equivalent to the well-known discrete cotangent discretization [PP93] with barycentric area weights (see, e.g., [BS08]).

For Poisson-based surface deformations the gradients of all three scalar coordinate functions are modified, and the coordinates of the deformed mesh are reconstructed by solving Equation (64). More precisely, let $\mathbf{Y}_t$ be the 3×3 matrix of the mesh coordinate gradients of triangle t, $\mathbf{Y}^\mathrm{T} = \left[\mathbf{Y}_1^\mathrm{T}, \ldots, \mathbf{Y}_{|\mathcal{T}|}^\mathrm{T}\right]$ the $3\,|\mathcal{T}| \times 3$ matrix of all stacked gradients, and $\mathbf{X}^\mathrm{T} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{V}|}\right]$ the $|\mathcal{V}| \times 3$ matrix of stacked mesh coordinates such that $\mathbf{Y} = \mathbf{G}\,\mathbf{X}$. Then the per-triangle gradients are modified using *local gradient transformations* $\mathbf{T}_t$ to give $\mathbf{Y}'_t = \mathbf{Y}_t\,\mathbf{T}_t^\mathrm{T}$. Finally, the deformed mesh with coordinates $\mathbf{X}'$, whose gradients best conform to the transformed gradients in least-squares sense, is reconstructed by solving

$$\mathbf{L}\,\mathbf{X}' = \mathbf{D}\,\mathbf{Y}' \, . \tag{65}$$

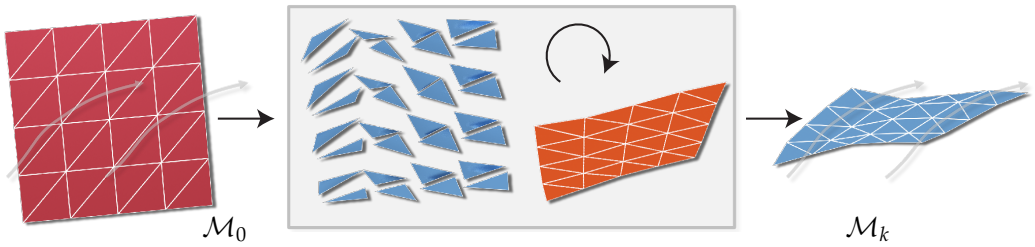


Figure 56: Deformation Principle. An initial surface mesh (•) is iteratively deformed by *conceptually* aligning each triangle individually to the flow and reconstructing the mesh (•) from these transformed gradients until the iteration converges to a flow-aligned mesh (•).

As the gradients are translation-invariant, the coordinates of at least one vertex need to be prescribed when solving this system.

Different applications rely on this general Poisson-based surface reconstruction technique and differ only in the way the local transformations $\mathbf{T}_t$ are specified. Usually, these transformations are combined rotation and scaling operations [YZX*04, SP04, ZRKS05, XZWB06]. In this chapter, we use a flow data-driven specification of the local transformations $\mathbf{T}_t$ to obtain either flow-tangential or flow-orthogonal surfaces.

9.2 INTERACTIVE DEFORMATION-BASED FLOW ALIGNMENT

In this chapter, we provide interactive tools for the explorative analysis of $3d$ steady vector fields $\mathbf{v}(\mathbf{x})$ over a spatial domain $\mathcal{D}$. A popular and well-studied family of methods for the visualization of vector fields are integral surfaces. For steady vector fields these are usually stream surfaces [MLP*10]. As introduced in Section 8.2, stream surfaces are regular surfaces $\mathcal{S} \subset \mathcal{D}$ that are tangential to the flow, i.e., given the normal $\mathbf{n}(\mathbf{x})$ of the stream surface, the local flux condition

$$\mathbf{n}(\mathbf{x})^{\mathrm{T}}\,\mathbf{v}(\mathbf{x}) = 0$$

holds for all points $\mathbf{x}$ on $\mathcal{S}$. Therefore, the *total flux* through a stream surface

$$f = \int_{\mathcal{S}} \mathbf{n}(\mathbf{x})^{\mathrm{T}}\,\mathbf{v}(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

vanishes [1].

Stream surfaces are usually constructed by advancing front integration algorithms that start integration at seed curves, such as straight line segments. Seed curves are usually defined manually by domain experts in an iterative and time

1 Note that there can be non-stream surfaces that minimize the total flux f due to the sign of the integrand. However, the surfaces we compute minimize the total flux by being locally tangential to the flow. Hence, effectively we optimize the absolute flux $\bar{f} = \int_{\mathcal{S}} \left|\mathbf{n}(\mathbf{x})^{\mathrm{T}}\,\mathbf{v}(\mathbf{x})\right| \mathrm{d}\mathbf{x}$.

consuming process [MLP*10]. Moreover, even if users are restricted to simple seed geometries, it is still possible to specify flow-tangential curves that lead to degenerate stream surfaces.

We propose an interaction metaphor that is different to this classic approach and requires neither seed curve specification nor flow integration. Instead, our specification approach is deformation-based and we give the user direct, interactive control over *complete* surfaces that are *flow-aligned*: the basic idea is to start from an initial surface $\mathcal{M}_0$, which is usually a procedurally generated planar triangle mesh. The user positions this surface within the area of interest of the flow domain. Then this surface is iteratively deformed into intermediate surfaces $\mathcal{M}_k$ in a way such that the result converges to a surface that is aligned with the flow. Figure 56 illustrates the basic principle. At any time the user can interactively reposition the surface to more interesting locations. Simultaneously, the shape of the surface adapts according to the altered local flow. We apply Poisson-based deformations and achieve interactivity by pre-factorization of the involved linear operators. In contrast to other Poisson-based deformations, our approach is data-driven in that it is steered by the local flow.

Note that in this chapter we denote both, flow-tangential and flow-orthogonal surfaces, by the uniform term flow-aligned surfaces. The surfaces are distinguished by either minimizing or maximizing flux. Flow-tangential surfaces correspond to stream surfaces, whereas flow-orthogonal surfaces are most useful for the specification of seed structures that are near flow-orthogonal and define, e.g., non-degenerate stream surfaces.

ORTHOGONAL ALIGNMENT. Surfaces that we align orthogonally to the flow *maximize* total flux. Exactly flow-orthogonal surfaces exist only in helicity-free fields, i.e., in field where $\mathbf{v}^{\mathrm{T}}(\nabla \times \mathbf{v}) = 0$ holds in every point [PCY09]. In general vector fields the orthogonality of surfaces is only possible approximately [SRGT12].

Before considering the entire surface mesh, we start with the analysis of aligning a single triangle orthogonally to the flow. The only admissible class of deformations are (rigid) rotations, since we strive to preserve the shape of the triangle and only align it locally to the flow. Let $\mathbf{R}(\mathbf{a}, \gamma) \in \mathrm{SO}(3)$ be the transformation describing a rotation around the axis $\mathbf{a}$ with angle γ, and let $\gamma(\mathbf{p}, \mathbf{q})$ be the enclosing angle between the vectors $\mathbf{p}$ and $\mathbf{q}$. If we assume a linear vector field $\mathbf{v}(\mathbf{x})$ on a triangle $t = (a, b, c)$ given by $\mathbf{v}^t(\mathbf{x}) = \sum_{i \in t} \phi_i{}^t(\mathbf{x})\, \mathbf{v}(\mathbf{x}_i)$, then the flux f_t through t is given by evaluation of the integral

$$f_t = \int_t \mathbf{n}_t{}^{\mathrm{T}} \mathbf{v}^t(\mathbf{x})\, \mathrm{d}\mathbf{x} = \frac{A_t}{3}\, \mathbf{n}_t{}^{\mathrm{T}} \left(\mathbf{v}(\mathbf{x}_a) + \mathbf{v}(\mathbf{x}_b) + \mathbf{v}(\mathbf{x}_c)\right) \ ,$$

with triangle area A_t and normal $\mathbf{n}_t$. Since $\frac{1}{3}(\mathbf{v}(\mathbf{x}_a) + \mathbf{v}(\mathbf{x}_b) + \mathbf{v}(\mathbf{x}_c))$ is the value of the linearized vector field at the center of the triangle, the nonlinear flux through

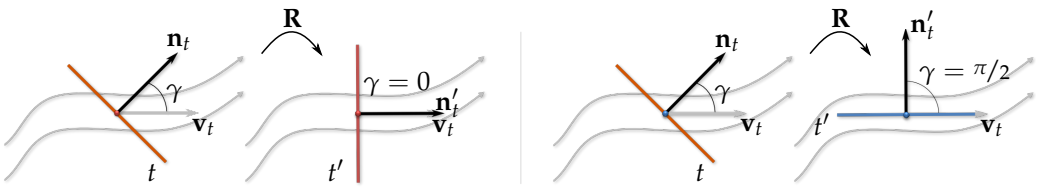


Figure 57: Triangle Alignment. A single triangle (resp. its coordinate function gradients, •) is aligned either orthogonally (left, •) or tangentially (right, •) to the flow by rotations $\mathbf{R}$ that depend on the angle γ.

the triangle can be approximated by dropping the linearity property and evaluating the approximate flux by a single point quadrature that evaluates the vector field only at the triangle center: $f_t \approx A_t \, \mathbf{n}_t{}^{\mathsf{T}} \, \mathbf{v}_t$ with $\mathbf{v}_t = \mathbf{v}(\frac{1}{3}(\mathbf{x}_a + \mathbf{x}_b + \mathbf{x}_c))$. To maximize the flux and align the triangle orthogonally to the flow a rotation that minimizes $\gamma(\mathbf{n}_t, \mathbf{v}_t)$ has to be performed around the axis $\mathbf{a}_t = \mathbf{n}_t \times \mathbf{v}_t$. We therefore rotate the triangle (around its center) by the transformation $\mathbf{R}(\mathbf{a}_t, \gamma(\mathbf{n}_t, \mathbf{v}_t))$, such that the approximated flux f_t is maximized, see Figure 57 (left).

Using Poisson-based deformations this consideration for a single triangle can directly be applied to align triangular meshes to the flow. Instead of transforming the vertex coordinates directly the basic idea is to transform the gradients of the coordinate function of each triangle. This means that the local gradient transformations of each triangle of the mesh are given by

$$\mathbf{T}_t = \mathbf{R}(\mathbf{a}_t, \gamma(\mathbf{n}_t, \mathbf{v}_t))$$

(see Section 9.1). The deformed mesh that optimally approximates these modified gradients (in least-squares sense) is then reconstructed by solving Equation (65). Figure 56 (center) illustrates this concept. In Section 9.6, we show that our method for computing orthogonal surfaces achieves higher flux rates compared to APAP surfaces by Schulze et al. [SRGT12].

TANGENTIAL ALIGNMENT. Only a slight modification of the previous argument is required to directly obtain flow-tangential surfaces in the same framework, as flow-tangential surfaces *minimize* the total flux. Therefore, using

$$\mathbf{T}_t = \mathbf{R}(\mathbf{a}_t, \gamma(\mathbf{n}_t, \mathbf{v}_t) - \pi/2)$$

to minimize the deviation of $\gamma(\mathbf{n}_t, \mathbf{v}_t)$ from $\pi/2$ as the local gradient transformation for each triangle minimizes the flux through each triangle, and approximately flow-tangential surfaces are obtained (see Figure 57 (right)).

INTERACTIVE ITERATIVE DEFORMATION. Surfaces will in general not directly be aligned with the flow if the mesh is reconstructed using (65) together with the proposed local gradient transformations. This is because the gradients of the reconstruction only comply with the prescribed gradients in least-squares

sense. Moreover, and more importantly, the rotations are only flux-optimizing if triangles undergo no translation (unless the vector field is constant). This is because otherwise the local flow after reconstruction differs from the one that was used to determine the rotations. In fact, the reconstruction inevitably has to introduce slight translations for the mesh to be continuous. Therefore, one single reconstruction is, in general, not sufficient to obtain a deformed *and* aligned mesh. Yet, if only *small* deformations are performed, then the flux is iteratively optimized until the mesh converges to an aligned configuration. This is justified by a mild continuity assumption on the vector field, i.e., a matrix norm of the Jacobian of the vector field is bounded. The convergence in interactive sessions is further quantified in Section 9.5.

We perform small deformations by limiting the maximal absolute value of the rotation angle by a constant small value η. Hence, small deformations are performed iteratively and we compute new local gradient transformations for each intermediate deformed configuration $\mathcal{M}_k$. Convergence is achieved if the maximal rotation angle is smaller than a constant value ϵ. This iterative deformation approach is, in essence, similar to other single-step deformation [SA07] and parametrization methods [ZRS05, LZX*08], which also minimize nonlinear measures.

From the user's perspective these iterative deformations seem to be continuous, although the deformations do not comply with the formal definition of continuous deformations of Section 3.1, because we compute no continuous parametrization of the evolving shape. In the iteration we merely only obtain discrete samples of an underlying continuous deformation. This is sufficient for the considered application because deformations generally converge quickly (see Section 9.5), and we are only interested in the final converged result as the intermediate steps only approximate this solution.

STABILIZATION. There are two cases that still need to be handled by corrections of the local gradient transformations: first, triangle area may vary between iterations due to the least-squares reconstruction. We avoid this artifact by performing an additional damped rescaling transformation of the prescribed gradients by using $\mathbf{T}'_t = \sqrt{A_t^0/A_t^k}\,\mathbf{T}_t$ as local gradient transformations, where A_t^k is the area of triangle t in the k-iteration. Second, for tangentially aligned surfaces it is possible that two neighboring triangles converge to a flow-aligned, but oppositely directed folded configuration, because both configurations minimize the local flux. This is due to the fact that local gradient transformations are computed independently of each other. To correct this artifact we prescribe a maximal dihedral angle $\theta_{\max}$. If for two neighboring triangles p and q with dihedral angle θ_{pq} we detect that the angle defect $\delta = \theta_{pq} - \theta_{\max} > 0$, i.e., the dihedral angle is greater than the prescribed maximum angle, we modify the local gradient transformations as $\mathbf{T}'_p = \mathbf{R}(\mathbf{n}_p \times \mathbf{n}_q, \delta/2)\,\mathbf{T}_p$, and $\mathbf{T}'_q = \mathbf{R}(\mathbf{n}_q \times \mathbf{n}_p, \delta/2)\,\mathbf{T}_q$. This way the dihedral angle is minimized in consecutive iterations until the fold is

resolved. We demonstrate the stability of this unfolding technique in the accompanying video corresponding to the publication [MSRT13b] [2]. In the unlikely event of sampling a critical point, i.e., $\mathbf{v}_j = \mathbf{0}$, or if the triangle is located outside of the flow domain we simply set $\mathbf{T}_j = \mathbf{I}$. In all experiments we use $\eta = 0.1$, $\epsilon = 10^{-6}$, and $\theta_{\max} = \pi/2$. These are all parameters of our approach.

NUMERICAL SOLUTION. For the reconstruction of the deformed mesh using (65) the coordinates of at least one vertex need to be prescribed to account for the translation invariance of the Poisson system. We found that fixing vertices of the triangle that is closest to the barycenter of the mesh works well in practice. It is sufficient to use "soft" constraints on these vertices such that the system in (65) becomes positive-definite.

The global iterative deformation can be performed in real-time as each single deformation is very cheap. This is because for each iteration only the right-hand side of (65) varies. In particular, the matrices $\mathbf{L}$ and $\mathbf{D}$ are constant as we discretize these operators on the initial mesh $\mathcal{M}_0$. This approach has two benefits: first, since the system matrix $\mathbf{L}$ (augmented with a "soft" constraints diagonal term $\mathbf{W}^\mathrm{T}\mathbf{W}$ weighted by β [3]) is symmetric positive-definite, we are able to perform one single sparse Cholesky factorization only *once* in a preprocessing step of the interaction. This factorization $\mathbf{R}^\mathrm{T}\mathbf{R} = \mathbf{L} + \beta\,\mathbf{W}^\mathrm{T}\mathbf{W}$ yields the sparse triangular Cholesky factor $\mathbf{R}$. Then only efficient back-substitutions have to be performed in each iteration for updated right-hand sides to update the mesh coordinates and guarantee interactive deformations: $\mathbf{X}'_{k+1} = \mathbf{R}^{-1}\,\mathbf{R}^{-\mathrm{T}}\left(\mathbf{D}\,\mathbf{Y}'_k + \beta\,\mathbf{W}^\mathrm{T}\,\mathbf{X}_c\right)$ with the matrix $\mathbf{X}_c$ of constrained mesh coordinates. Second, in each iteration $\mathcal{M}_0$ is deformed according to the updated gradients, and we conceptually do not deform intermediate meshes. This way possible errors introduced in a single deformation step cannot accumulate in the iteration and the mesh discretization of the deformed surface is based on $\mathcal{M}_0$.

INTERACTION. During the iterative deformation the user can interactively position and orient the current surface $\mathcal{M}_k$ inside the flow domain. In the next iteration the surface is deformed according to the changed local flow. We provide rigid translation and rotation operations and interleave user operations and deformation iteration. Usually deformations converge after a few iterations (quantified experimentally in Section 9.5). The surface can also be grown in a user-defined direction. We use a growing strategy that is similar to the one by Schulze et al. [SRGT12] by computing an offset curve at the boundary that is tangential to the surface. The offset curve is then tessellated to obtain the grown surface. Note that we perform this update operation of mesh coordinates on

2 The video is located in the additional material folder `addmaterial/poiss`.

3 "Soft" constraints are a special form of (algebraic) Tikhonov regularization with $\mathbf{W} \equiv \mathbf{\Gamma}$, where $\mathbf{\Gamma}$ encodes the indices of the constrained vertices (see Chapter 7).

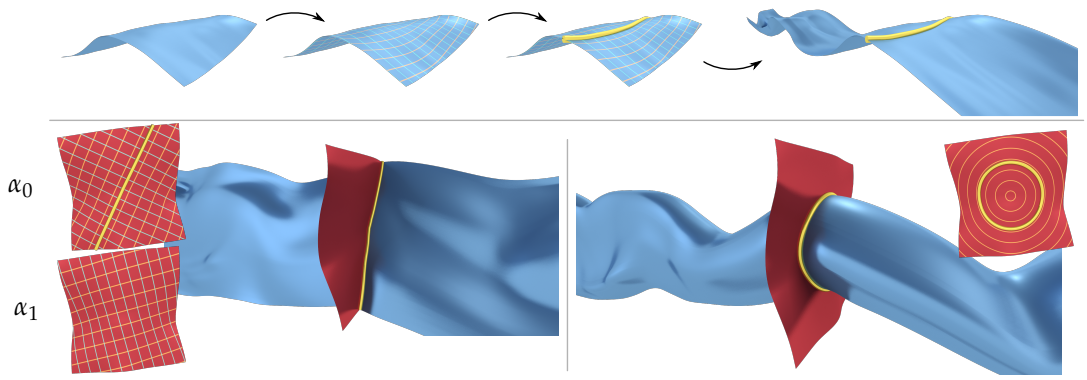


Figure 58: Flow-aligned Parametrizations. Top: an interactively placed flow-tangential surface (●) is parameterized in a flow-aligned way. From an iso-contour (●) a larger exact stream surface is integrated (left to right). Bottom: orthogonally aligned surfaces (●) are parametrized using different angular rotations (left) or using circular geodesics-based distance fields (right).

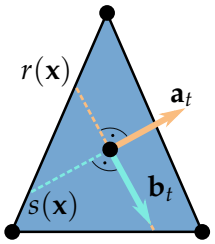
both the current mesh $\mathcal{M}_k$ and the base mesh $\mathcal{M}_0$ to be able to update the differential operators, which are discretized on $\mathcal{M}_0$. However, since growing changes the connectivity of the mesh, the discretized Laplace-Beltrami operator has to be refactored in each growing step. This is a more expensive operation, especially for large meshes. It turns out that growing a tangentially aligned surface in a direction orthogonal to the flow is advantageous for the seed curve extraction discussed in the next section (see Figure 61).

9.3 SURFACE PARAMETRIZATION FOR SEED EXTRACTION

The flow-aligned deformations presented in the previous section are well-suited for interactive exploration of flow data sets. However, as we do not perform mesh adaption during deformation, we cannot formally guarantee exact flow alignment. Nevertheless, the converged surfaces are well-suited to provide *orthogonal* seed curves for an additional front line-based stream surface integration using either classic [Hul92] or more advanced [SGRT12] surface integrators.

Stream surfaces seeded from curves in approximately flow-tangential surfaces coincide locally, but they span a larger part of the domain due to integration. Flow-orthogonal surfaces are even more suited for stream surface seeding as any curve in such a surface is also flow-orthogonal by construction. Seed curves on flow-orthogonal surfaces can either be manually "drawn" by the user (see Figure 61) or computed automatically. To automatically calculate seed curves we first perform different kinds of parametrizations of flow-aligned surfaces. Flow-orthogonal seed curves are then extracted as iso-contours of these parametrizations.

Note that the following computations are based on the same computational framework of Poisson-based scalar field optimization. In fact, our approach to parametrization of flow-aligned surfaces is similar to the surface quadrangulation approach by Bommes et al. [BZK09]. This approach first computes a normalized orthogonal cross field $\mathbf{C}_t = [\mathbf{a}_t, \mathbf{b}_t]$ on each triangle t. If these are interpreted as the parametrization gradient and cogradient, we can directly compute the corresponding parametrization scalar fields. For this a globally integrated parametrization energy



$$e_p(\mathbf{r}, \mathbf{s}) = \|\mathbf{G}\,\mathbf{r} - \mathbf{a}\|_{\mathbf{A}}^2 + \|\mathbf{G}\,\mathbf{s} - \mathbf{b}\|_{\mathbf{A}}^2 = \|\mathbf{G}\,[\mathbf{r}, \mathbf{s}] - [\mathbf{a}, \mathbf{b}]\|_{\mathbf{A}}^2 \tag{66}$$

similar to (63) is minimized for the parametrization scalar fields $r(\mathbf{x}), s(\mathbf{x})$ with stacked coefficients vectors $(\mathbf{r}, \mathbf{s})$. This is equivalent to solving

$$\mathbf{G}^{\mathrm{T}}\mathbf{A}\,\mathbf{G}\,[\mathbf{r}, \mathbf{s}] = \mathbf{G}^{\mathrm{T}}\mathbf{A}\,\mathbf{C} \tag{67}$$

with the $3\,|\mathcal{T}| \times 2$ matrix $\mathbf{C}^{\mathrm{T}} = \left[\mathbf{C}_1^{\mathrm{T}}, \ldots, \mathbf{C}_{|\mathcal{T}|}^{\mathrm{T}}\right]$. Scalar fields need to be constrained at a single vertex c once again. Unless otherwise specified by the user we constrain the vertex nearest to the barycenter onto the parameter origin. In general, only the gradients $\mathbf{a}_t$ need to be determined, and the second orthogonal cross direction follows from $\mathbf{b}_t = \mathbf{a}_t \times \mathbf{n}_t$. For quadrangulations the computation of the guiding cross fields turns out to be the most complex part, because they need to be determined from the shape geometry only [BZK09]. In this work, however, we make use of the flow-alignment property of the considered surfaces, from which guidance fields can directly be determined. Figure 58 illustrates the different parametrization types we propose: tangential, orthogonal, and circular parametrizations.

TANGENTIAL PARAMETRIZATION. For tangentially flow-aligned surfaces a natural choice for the parametrization guidance field $\mathbf{a}_t$ is simply the vector field itself: $\mathbf{a}_t = {}^{\mathbf{P}_t\,\mathbf{v}_t}/_{\|\mathbf{P}_t\,\mathbf{v}_t\|}$ with $\mathbf{P}_t = \mathbf{I} - \mathbf{n}_t\,\mathbf{n}_t^{\mathrm{T}}$. The projection $\mathbf{P}_t$ into the triangle plane is only required if the surface is not yet aligned exactly with the flow. Normalization of the gradient field guarantees that iso-contour lines are near-equidistant on the surface. Iso-contours of $r(\mathbf{x})$ are near-perpendicular to the flow and can therefore be used as seed curves of stream surfaces, whereas iso-contours of $s(\mathbf{x})$ should not be used as seed curves as they are near-tangential to the flow. Note that the extracted iso-contours are general curves, which is a much greater set of possible seed curves compared to the typically used straight line segments. The number of possible selectable stream surfaces is therefore also much higher. This natural Poisson-based parametrization technique for flow-aligned surfaces is applicable to all other integral surfaces as well and is useful in its own right: Figure 59 demonstrates that our normalized parametrizations are less distorted compared to the unnormalized time line/stream line parametrizations, which are generated by common stream surface integrators like [Hul92]. We use this type of parametrization to compute LIC-like visualizations in Section 9.4.

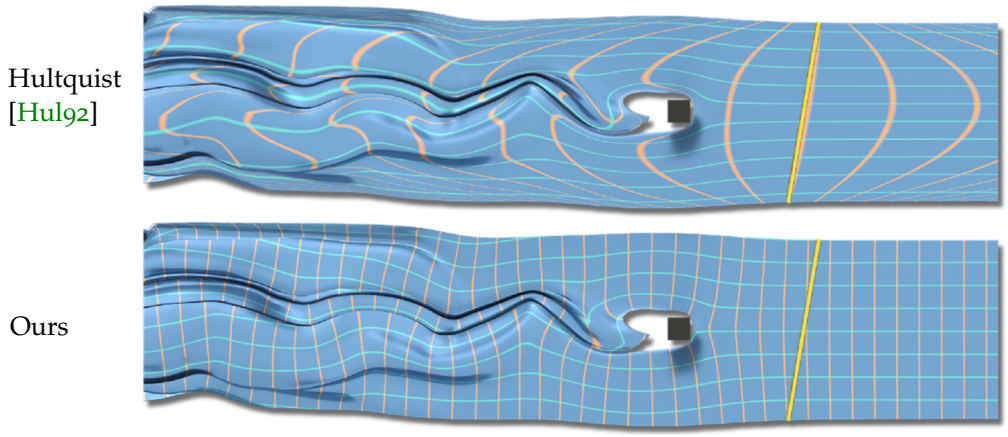
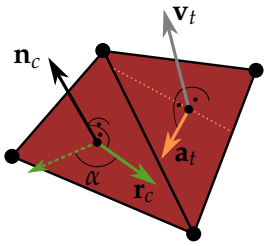


Figure 59: Stream Surface Parametrizations. The exact unnormalized time line/stream line parametrization obtained by front line-based integrators (top) exhibits more distortion than our Poisson-based tangential parametrization (bottom).

ORTHOGONAL PARAMETRIZATION. One application of orthogonally aligned surfaces for flow analysis is their applicability as general seed structures. Therefore, points on the interactively specified surfaces can also directly be used for seeding, e.g., *illuminated stream lines* [MLP*10] (see Figure 60).

To extract seed curves for surface integration we again rely on a parametrization. By construction, the vector field does not directly provide directions that are tangential to the surface and usable for parametrization guidance. To obtain these directions we perform a global surface-dependent rotation in the following way: for the normal $\mathbf{n}_c$ at the constrained vertex c as the reference, we compute a global rotation axis $\mathbf{r}_c$ by using one basis vector of the kernel of $\mathbf{n}_c$. The basis of the kernel can, e.g., be obtained by using a full QR-factorization $\mathbf{n}_c = \mathbf{Q}_c \mathbf{R}_c$ and taking the last two column vectors of $\mathbf{Q}_c$. Then the unnormalized guidance field is given by $\mathbf{a}_t = \mathbf{P}_t \left(\mathbf{R}(\mathbf{n}_c, \alpha)\, \mathbf{r}_c\right) \times \mathbf{v}_t$. Here, we have introduced one additional degree of freedom for the user in that $\mathbf{r}_c$ can additionally be rotated in the kernel by an angle α to align the resulting iso-contours differently in the orthogonal surface. Again, iso-contours are general curves as they are embedded in orthogonally aligned surfaces. Note that they have the tendency to span the surface in a straight way (see Figure 58 (left)).



CIRCULAR SEEDS. For certain flow phenomena straight seed curves may not be the desired type of seed structures. For example, recently circular seed curves were successfully used for illustrative flow visualization [HGH*10]. This type of curves can also be extracted in our computational framework. Note that circular curves can be regarded as iso-contours of a geodesic distance field centered at c. One way to compute geodesic distances is to use *fast marching methods* [KS98].

Instead, we use the recent method by Crane et al. [CWW13], who compute geodesics using a heat flow method, because it uses the same gradient-based operators we use in our work. We only sketch an outline of this method here. In essence, their method first performs a heat flow integration from a point heat source to obtain a heat distribution scalar field. Then a guidance vector field is obtained by normalizing the negative gradient of the heat distribution. The heat integration is performed by a single implicit backward Euler step by solving $(\mathbf{I} - \lambda \mathbf{L})\ \mathbf{h} = \mathbf{h}_0$ for the heat scalar field $\mathbf{h}$. Here, $\mathbf{h}_0$ is the initial heat distribution that is one at c and vanishes on all other vertices. As proposed by the authors, we use a scale-invariant time step of $\lambda = {}^5 A_{\mathcal{M}}/|\mathcal{T}|$, where $A_{\mathcal{M}}$ is the total surface area. Then the guidance vector field is obtained by normalizing the negative gradients $\mathbf{a} = -\mathbf{G}\,\mathbf{h}$ on each triangle and a final Poisson system is solved with these gradients for a distance field $d(\mathbf{x})$. We use iso-contours of $d(\mathbf{x})$ to extract circular seeds on orthogonally aligned surfaces. Figures 58 (right) shows an example of using this technique.

9.4 PARAMETRIZATION-BASED LIC-LIKE VISUALIZATION

The Poisson-based parametrization presented in the previous section can directly be used to compute dense LIC-like texture-based flow structure visualizations for stream and path surfaces [LHD*04]. For the unnormalized tangential case ($\mathbf{a}_t = \mathbf{v}_t$) the key observation is that the energy (66) approximates flow directions with gradients of $r(\mathbf{x})$ in least-squares sense. The scalar field $r(\mathbf{x})$ can therefore be interpreted as an approximation for the time coordinate of a *global* space-time parametrization of stream as well as path surfaces. Such space-time parametrizations are either hard to compute or are of poor quality (see Figure 59 (top)). For this reason, surface-based LIC techniques rely on chart-packing or screen-space techniques to overcome the absence of a high quality surface parametrization [LGSH06, LTWH08, HPW*12].

We use global tangential parametrizations to efficiently compute LIC-like visualizations. These special parametrizations allow to simply map precomputed seamless but structured textures onto the integral surface to visualize the flow behavior. One choice is a high contrast texture of colored noise that is anisotropically low-pass filtered in time direction and generates LIC-like flow structure patterns. Alternatively, an illustrative visualization of flow directions is obtained by using a seamless texture of flow-aligned arrows. Both textures are shown in Figure 63 (bottom). As texture structures are approximately aligned to flow directions due to the energy-minimizing parametrizations, the visualizations have a LIC-like character.

Flow directions are only approximated in least-squares sense and are not necessarily exactly interpolated everywhere by the minimizers of (66). However, in all our experiments we found that the gradients of $r(\mathbf{x})$ are generally very well

fitted to the flow and will only diverge in small regions of the surface. The time component of the energy e_p therefore vanishes almost everywhere on the surface, and the LIC-like visualization is not flow-aligned only in small localized regions. To remedy this limitation, we mask these regions by blending the texture with the average texture color at each pixel. The blending function $b(\gamma)$ depends on the local angle $\gamma(\mathbf{v}(\mathbf{x}), \nabla r(\mathbf{x}))$ between the flow and the gradient of the time component of the parametrization. A smooth cubic sigmoid-shaped blending polynomial that interpolates $b(0) = 0$ and $b(\pi/2) = 1$ turns out to be sufficient to hide the parts of the surface where $\nabla r(\mathbf{x})$ is not exactly aligned with $\mathbf{v}(\mathbf{x})$. The remaining structures of the visualization are aligned with the flow and their variation is also distributed approximately proportional to $\|\mathbf{v}\|$, since we use $\mathbf{a}_t = \mathbf{v}_t$ to fit the parametrization.

Our technique neither requires any flow integration nor costly texture advection and only amounts to optimizing a single global parametrization energy. This makes our texture-based method very efficient to compute as we only have to sample the vector field *once* to setup the resulting linear system (67). Additionally, our visualizations are frame-coherent and the texture map can also be used to animate the flow structures in steady stream surfaces: an animation of flow structures is obtained by simply offsetting the time-coordinate of the parametrization for each frame, resulting in a movement of the flow structure along approximated stream lines. Note that animation is only meaningful for stream and not for path surfaces.

9.5 RESULTS

We continue to provide further results of our approaches. Examples for the interactive work-flow and further results of all proposed techniques are demonstrated in the video accompanying the corresponding publication [MSRT13b] [2].

SEED SELECTION. Figure 60 shows integration results starting from seed curves on orthogonal surfaces. The orthogonal surfaces were all placed interactively. Their number of triangles ranges from $1.7 \cdot 10^3$ (STALLING2D) to $3.5 \cdot 10^3$ (CYLINDER). The STALLING2D flow is a conservative two-dimensional benchmark flow data set proposed by Stalling [Sta98]. We show multiple uniformly distributed stream surfaces on a highly curved orthogonal surface. Our approach for orthogonal surfaces does not suffer from "spiky" artifacts at fixed vertices that may appear using the APAP approach (see Figure 5 in [SRGT12]). A similar example is shown for the more complex CYLINDER flow [CSBI05] (see, e.g., [BFTW09, ELM*12, SGRT12]) where an orthogonal surface was placed near the flow vortex. Iso-contour variation reveals the narrow outflow region of the vortex through the orthogonal surface. Different levels of the tumbling

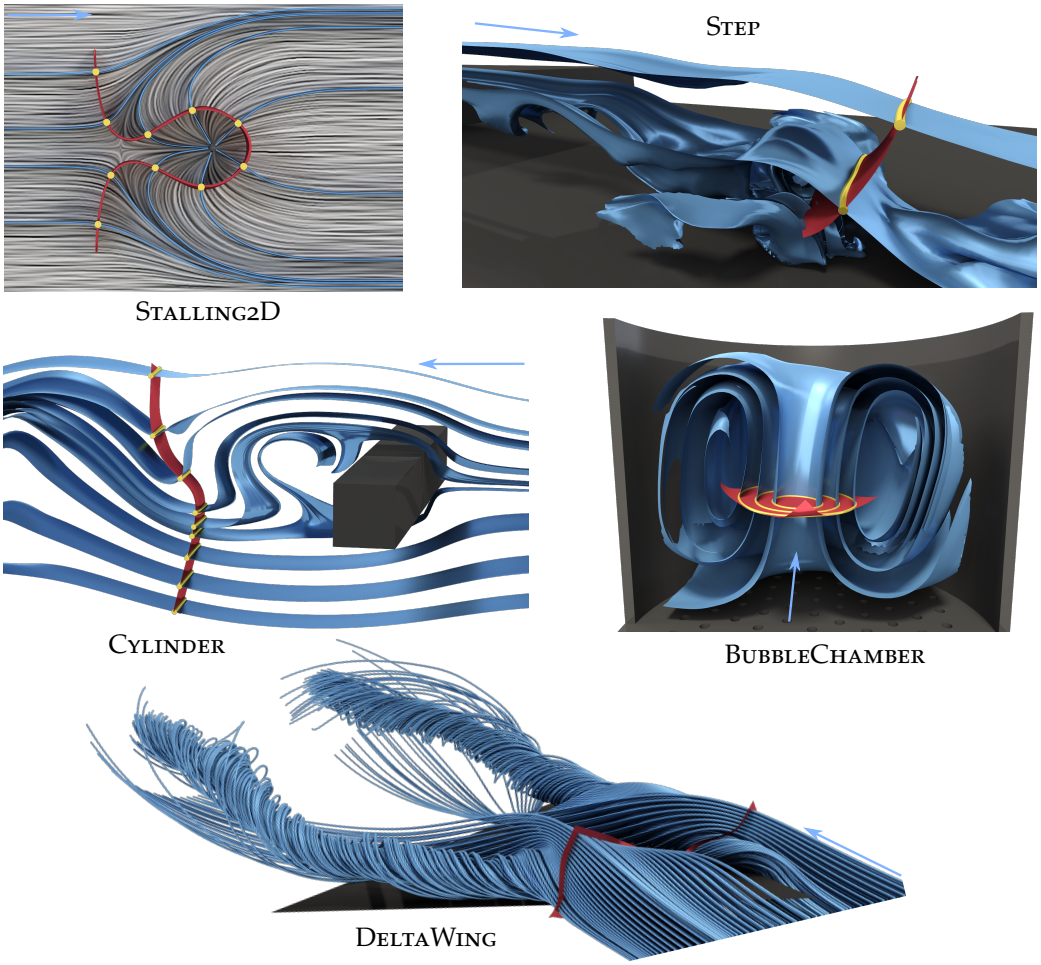


Figure 60: Interactive Seeding Results. The orthogonal surfaces (●) are positioned interactively by the user. Then stream surfaces and stream lines (●) are seeded from these orthogonal surfaces and integrated tangentially to the flow. Both orthogonal (Stalling2D, Cylinder, and Step) as well as circular parametrizations (BubbleChamber, cutaway view) were used to extract flow orthogonal seed curves.

flow in the measured BubbleChamber data set of a bioreactor can be visualized using circular seeding curves on an orthogonal surface located at the turnover point of the flow. The simulated DeltaWing data set of the flow at a triangle-shaped airplane is known to contain two dominant vortical structures [Dal83, GKT*08, BWF*10, HGH*10, SGRT12]. Flow-orthogonal surfaces can be used to seed integrated quantities like stream lines near the assumed locations of the vortices for their concrete visualization. Stream surfaces of different complexity can be selected as different iso-contours on an orthogonal surface in the Step data set of a flow behind a backward-facing step [KJ00]. Note that the shapes of all interactively deformed surfaces are fairly insensitive to tessellation

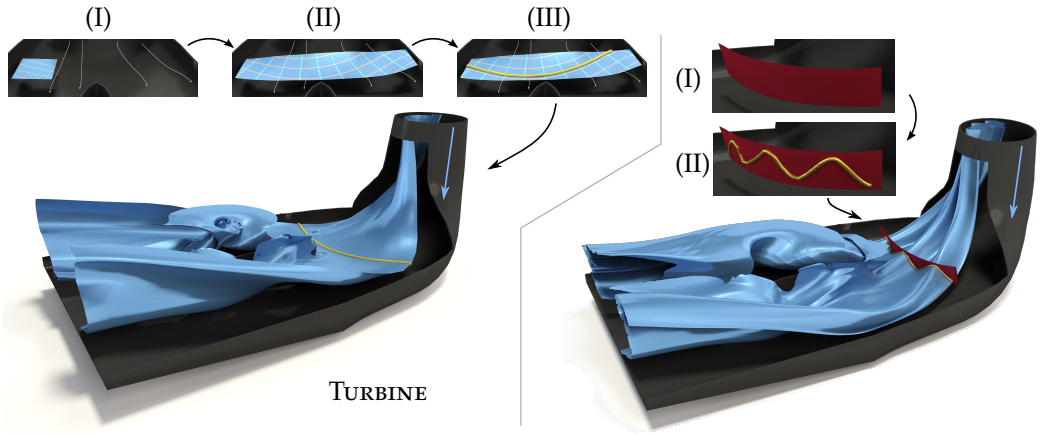


Figure 61: Seeding at the TURBINE. Left: a tangential surface is interactively placed into the inflow area of the flow (I). It is then grown *orthogonally* to the flow (II) and an iso-contour of a tangential parametrization yields a non-straight seed curve (III) for the integration of the final stream surface (top right). Right: onto an interactively placed orthogonal surface (I) an orthogonal seed curve is manually "drawn" (II) and used for stream surface integration.

quality, because the energy (63) is an integrated quantity and a discretization of a continuous energy.

Figure 61 (left) illustrates an effective work flow to compute seed curves in prescribed flow tangential surface patches at the example of a hydroelectric TURBINE [SGRT12]: a tangentially aligned surface is placed in the inflow region and subsequently grown *orthogonally* to the flow. Then a seed curve can be extracted from the tangential parametrization defining a stream surface that covers a large part of the domain and interpolates the initial patch. Note that extending a stream surface orthogonally to the flow is not possible with classic stream surface integrators, but poses no problem for our deformation-based approach. Figure 61 (right) shows another stream surface integrated from an orthogonal seed curve that is "drawn" by the user on an orthogonal surface, which was positioned interactively before.

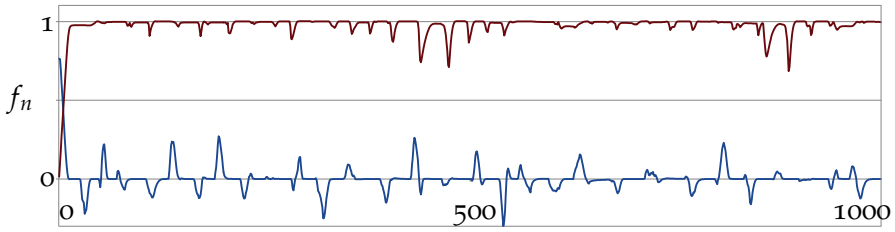
LIC-LIKE VISUALIZATION. Figure 63 shows LIC-like and illustrative visualizations that are based on our tangential parametrizations for both stream and path surfaces. The visualized structures capture flow directions in all non-masked regions and bifurcations are correctly represented by the parametrizations. We evaluate average angles (in degree) and their standard deviation $(\bar{\gamma}, \sigma_\gamma)$ between $\mathbf{v}$ and ∇r for the DELTAWING $(1.0°, 0.9)$, STEP $(2.2°, 2.4)$, and TURBINE $(2.7°, 5.7)$ data sets. These values are close to optimally aligned solutions. The larger standard deviation is caused by outlier regions, which are masked by blending. We note that for a few examples the parametrization may not be onto, i.e., it may contain self-intersections in the parameter domain. However, self intersections in the parametrization pose no problem in this application as

they only result in texels that are mapped to multiple parts of the surface. As the textures have no structure except flow alignment, which is maintained even at self-intersections, no visual artifacts or incorrect visualizations arise. Examples of frame-coherence / viewport-independence and steady flow animation are shown in the video of [MSRT13b] [2].

CONVERGENCE. The interactive deformations presented in Section 9.2 are designed to either maximize or minimize the total flux iteratively. We quantify the convergence of the deformation iteration within interactive user sessions. As flux depends on the vector field norm, which can significantly vary in the flow domain, we measure the discrete normalized flux

$$f_n = \frac{1}{A_{\mathcal{M}_k}} \sum_{t \in \mathcal{T}} A_t \, \mathbf{n}_t{}^{\mathrm{T}} \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$$

in each iteration k. The results are visualized in the next graphs:



User interaction / iteration k

The normalized flux f_n of each deformation iteration in an interactive session for tangentially (●) and orthogonally (●) aligned surfaces is shown. Translations, rotations as well as growing operations were performed by the user, which result in quickly optimized "flux spikes". In fact, we observe rapid decay for both surface types such that convergence is achieved after few iterations. Note that $f_n = 1$ for exactly orthogonal surfaces, and f_n vanishes for exactly tangential surfaces. Moreover, tangential surfaces always converge to exactly aligned surfaces, whereas orthogonal surfaces may not always reach an exact state. This is not surprising, since perfect orthogonality is generally not possible, except for, e.g., helicity-free flows [PCY09]. Nevertheless, the experiments indicate that our flow-orthogonal surfaces are generally close to exact perpendicularity.

PERFORMANCE. We solve linear systems using the CHOLMOD [CDHR08] library to perform sparse Cholesky factorization with fill-in reducing reordering. According to Crane et al. [CWW13], solving Poisson-type problems has, in theory, sub-quadratic complexity, but scales even better in practice [BBK05a]. All our examples were computed on an Intel Core i7-2600 3.4GHz Linux PC. The following table lists measured timings of our single-threaded deformation method:

$\lvert\mathcal{T}\rvert$	Preprocess	Iteration		IT/s	NConv
	Factor	GTransf	Solve		
$8 \cdot 10^2$	0.5	0.26	0.16	2,300	5
$5 \cdot 10^3$	2.3	1.5	0.25	570	9
$2 \cdot 10^4$	13.2	6.4	0.83	130	16
$8 \cdot 10^4$	932	24.2	4.5	35	19
$3.2 \cdot 10^5$	8,000	91.1	23.0	9	22

For differently sized meshes ($|\mathcal{T}|$) we summarize the required times to perform system factorization (Factor), gradient transformation (GTransf), and system solving (Solve) in *ms*. IT/s indicates performed number of iterations per second, and NConv denotes the average number of iterations for convergence. Using a mesh resolution of $|\mathcal{T}| \approx 5 \cdot 10^3$ turns out to be sufficient for all tested data sets. Moreover, significantly larger meshes can also be handled without problems. The timings indicate that interactive results can be guaranteed. A linear system factorization (Factor) has to be computed only once and in each iteration only efficient gradient transformations (GTransf) and system solving (Solve) by back-substitution need to be performed. Note that local gradient transformation computations require vector field sampling (see Section 9.2), which turns out to be costly for high mesh resolutions. However, since each transformation can be computed independently, this operation is a natural candidate for acceleration by, e.g., parallel vector field sampling and transformation computation on the GPU. The timings for tangential / orthogonal parametrization as well as for the LIC-like visualizations are very similar to the presented deformation timings, as almost identical systems are solved for Poisson-based parametrization.

9.6 DISCUSSION

RELATION TO INTEGRATION-BASED METHODS. Our method does not strive to replace classic front line-based stream surface integrators for tangential surfaces in terms of approximation quality: integrated surfaces are flow tangential by construction, and the approximation quality is high due to adaptive local mesh refinement, which we do not perform to guarantee interactivity. Rather, our approach applies Poisson-based methods to enhance the usability and effectiveness of the concept of surface integrators in multiple ways: approximate interactive deformation-based exploration and specification of interesting flow regions provides a "preview" for integration-based methods. Additionally, we solve the specification problem of general nontrivial seed curves using a parametrization-based extraction approach. Growing tangential surfaces orthogonally to the flow is another technique that is not possible with classic integrators. Additionally, orthogonal surfaces are naturally obtained by flux optimization,

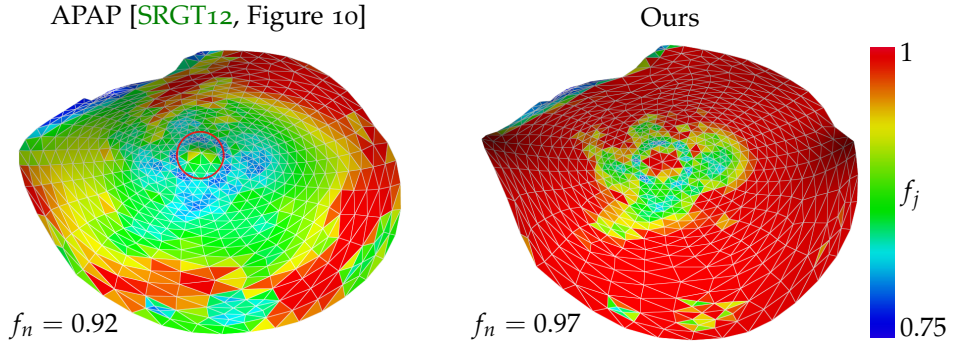


Figure 62: APAP Comparison. Starting from a converged integration-based APAP surface (left) our deformation converges to an even better aligned orthogonal surface (right). The color scale captures normalized $f_j \in [0.75, 1]$. The highlighted fixation artifact at the center vertex (•, left) is also removed by our deformation-based approach.

whereas only a few integration-based methods exist for these types of surfaces. Most recently, APAP surfaces were proposed for this problem.

COMPARISON TO APAP. In the case of flux maximization we search for the same class of surfaces as the APAP approach of Schulze et al. [SRGT12]: orthogonally aligned surfaces. In contrast to the integration-based APAP method, the vertices of our surfaces are not constrained to only move along (scaled) flow directions. Instead they move along general paths. This way we achieve higher flux rates compared to the APAP approach. Moreover, our method does not generate deformation artifacts at constrained vertices, whereas even with careful regularization these artifacts cannot be completely suppressed by the APAP approach. Figure 62 exemplifies both properties: we initialize our deformation method with a converged APAP surface and obtain an orthogonal surface of higher quality. Additionally, we only need to perform the expensive linear system factorization *once*, whereas APAP requires to solve a new linear system in *every* integration step.

SMOOTHED ENERGIES. The energy smoothness concept of Chapter 7 can directly be applied to the triangle-based energies considered in this chapter, too. In fact, the discussed systems of this chapter are *identical* to the Poisson-type problems of Section 7.4.2. Our results indicate that energy smoothing is not necessary in the first place and deformations converge in all relevant data sets. This is because target gradients are prescribed for *all* triangles in a smooth way due to the smoothness of the underlying vector field.

Still, we can construct pathological fields for which our iteration does not converge to a stable solution: consider the FOCUSSADDLE vector field $\mathbf{v}(\mathbf{x}) = (y - x,\, x - y,\, z)^{\mathrm{T}}$, which is not helicity-free. No exact orthogonal surfaces exist in this

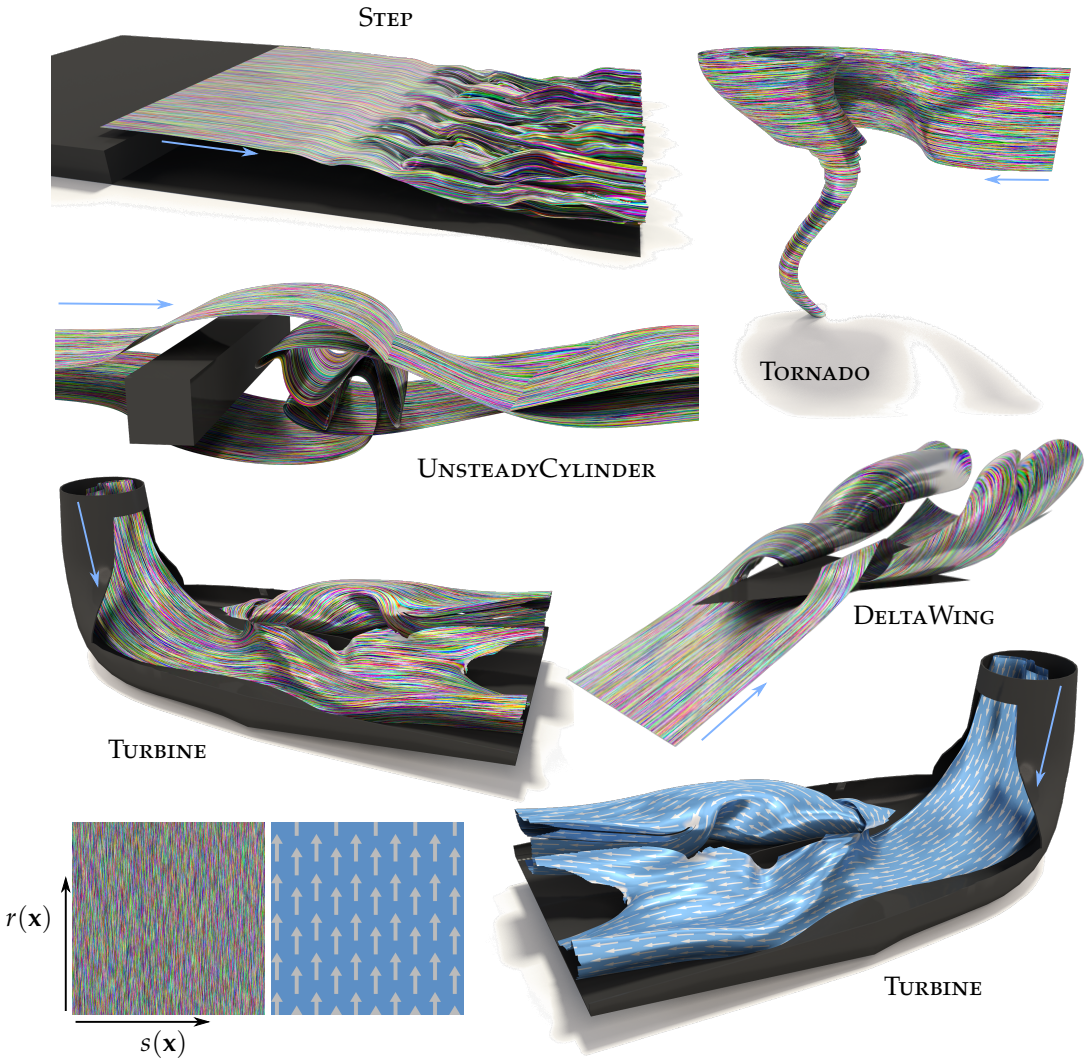


Figure 63: LIC-like Visualization Results. Precomputed textures of anisotropic noise and flow-aligned arrows (bottom) are mapped to the tangentially parametrized stream surfaces (Step, Turbine, DeltaWing, Tornado), and two path surfaces (UnsteadyCylinder). Resulting LIC-like and illustrative structures are flow-aligned everywhere except at locally masked regions (•).

field. Moreover, due to the swirling structure of the field, our optimization for orthogonal surfaces does not converge (in this special field!), because the local vector field before and after surface reconstruction differs in a way that results in non-converging rotational motion. Figure 64 (top) shows an example where we have fixed a center triangle with "hard" constraints using elimination to amplify deformation artifacts. Interestingly, the deformation converges instantly to an artifact-free and near-orthogonal surface with a smoothed Poisson energy that is discretized similar to (50) (Figure 64 (bottom) with $\beta > 0$). Hence, as energy smoothing incurs practically no extra costs due to the simple change of optimization norm, it can be beneficial to use it for the proposed deformations,

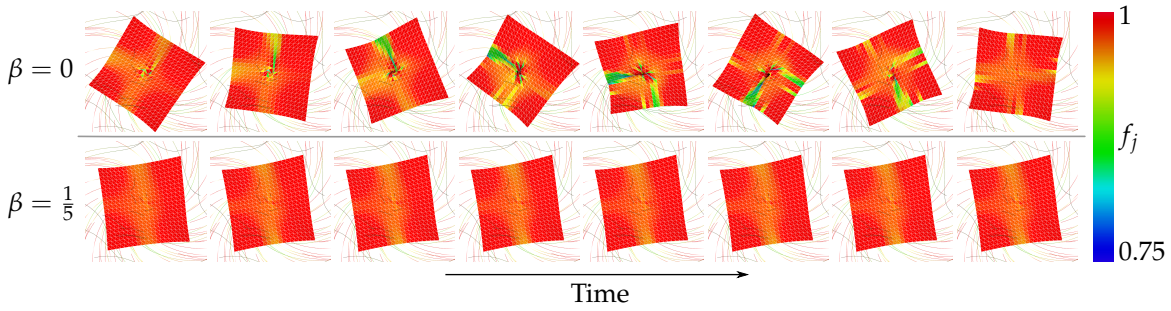


Figure 64: Smoothed Energies. Using the smoothed triangle energies of Chapter 7, the deformation for an orthogonally aligned surface instantly converges in the pathological FocusSaddle field ($\beta > 0$), which is not the case for the original energy ($\beta = 0$). Note that artifacts are intentionally amplified by using hard constraints on the fixed vertices of a center triangle.

too, although we observed rapid convergence in all other tested vector field even without energy smoothing.

LIMITATIONS. Our deformation method is based on local alignment of surface orientations to the flow. High curvature regions of the flow require tangential surfaces to change their normal rapidly. This is only possible for surfaces that are locally tessellated sufficiently high. However, since we perform no adaptive mesh refinement yet, the deformation iteration is unlikely to converge in these areas. An adaptive subdivision scheme (with costly refactorization) and criteria for its applications would therefore be required to also handle these regions. Note that this lack of refinement is not a severe problem for the current approach as meshes do not degenerate in these areas; the mesh is rather "pushed out" of high curvature areas due to folding penalization (exemplified in the video of [MSRT13b] [2]).

Essentially, the quality of LIC-like visualizations depends on the quality of the underlying parametrization. Although exact flow alignment cannot be guaranteed by the least-squares parametrization energy, the presented results show that the parametrizations are flow-aligned almost everywhere in data sets of practical relevance. Still, there exist regions where flow alignment may not exactly be met and which are masked in the visualization. Our parametrization trades exact flow alignment, which is hard to compute, for more efficient computation, which may not be exact everywhere. In contrast, screen-space LIC methods become unstable at silhouettes, and chart-packing approaches suffer from texture resolution and discontinuity artifacts. The explicit hierarchical handling of critical or hard-to-parameterize regions [LTWH08] and the analysis of better suited parametrization energies is, therefore, an interesting direction for further research.

9.7 SUMMARY

In this chapter, we introduced a number of interactive Poisson-based tools for surface-based visualizations of steady flow fields. We presented the idea of flux-optimizing deformations from which we obtain both flow-tangential and flow-orthogonal surfaces. We showed that interactive deformations and parametrizations are well-suited to support explorative analysis and seed specification to study flow phenomena. Our approach is effective thanks to a numerical factorization of the involved linear operators, and we demonstrated rapid convergence. Additionally, we introduced fast to compute LIC-like rendering techniques. Since these methods are all based on a small set of well-established differential operators, it is easy to integrate our approach into existing visualization packages.

Our deformation-based method may also be applied to different data, e.g., to solve ambiguities of DT-MRI fiber tracking [SS08]. Another promising direction is to respect additional application-dependent surface quality measures in the optimization. We believe that the proposed parametrizations are also beneficial for, e.g., applications like remeshing of integral surfaces and for rendering methods that depend on parametrizations with a low amount of distortion [HGH*10].

10

AUTOMATIC GLOBAL SELECTION OF STREAM SURFACES

In the previous chapter, we introduced a number of interactive flow exploration approaches. In this chapter, we complement these methods by presenting a technique that assists users in the completely *automatic* selection of stream surfaces. That is, we solve the following problem: given a $3d$ steady vector field $\mathbf{v}(\mathbf{x})$, we find a stream surface that describes the global flow behavior of $\mathbf{v}$ "best". Automatic characteristic surface selection methods are important for flow exploration, i.e., the detection of relevant flow features in unknown data sets. Additionally, unsupervised flow analysis applications benefit from automatically selected stream surfaces.

Automatic stream surface selection requires suitable quantitative and comparable criteria that evaluate the quality of stream surfaces relative to its defining flow. However, in the literature there has not been a formal description of what a "good" stream surface is. Hence, we start this chapter with the introduction of a new *quality measure* for stream surfaces. Our quality measure prefers surfaces where lines of curvature are aligned with stream lines on the surface as well as possible, i.e., the structure of the *shape* and the *flow* should be similar. This intrinsic quality measure is motivated by the fact that for such surfaces common line renderings of the geometry and the flow coincide and are therefore not interfering with each other. Additionally, we show that the quality measure is related to curvature-based surface fairing methods, which are well-known in geometry processing but were not yet applied to surface-based flow visualization problems. To make such a quality measure applicable, we show that it can be computed without the need to explicitly estimate curvature tensor fields on stream surfaces. Based on our quality measures, we provide an automatic algorithm for finding *globally* optimal stream surfaces. No user interaction is required by our approach. The utility of the quality measures and the extraction method is then demonstrated for a number different data sets.

10.1 BACKGROUND

A number of methods were proposed for the automatic exploration of flows using line and surfaces-based geometries.

AUTOMATIC LINE-BASED FLOW EXPLORATION. The problem of automatically finding good distributions of stream lines on $2d$ manifolds is well-researched. Turk and Banks [TB96] are first to propose an image-based algorithm, and Jobard and Lefer [JL97, JL00] propose direct methods to solve the stream line placement problem. Multiple improvements have been published and are discussed in the survey by McLoughlin et al. [MLP*10].

Stream line placement in $3d$ flow is a more complex problem. Several approaches have been proposed to find well-distributed seeds in data space as well as occlusion-aware methods that work in view space [MCHM10, GRT13] or apply clustering [CYY*11]. We again refer to the survey by McLoughlin et al. [MLP*10] for an overview on advanced stream line seeding in both $2d$ and $3d$ flows. Note that the problem of stream line selection is less complex than the stream surface selection problem, because the search space is significantly smaller: there is a unique stream line that passes through a domain point.

AUTOMATIC SURFACE-BASED FLOW EXPLORATION. Only few methods have been proposed for the more complex problem of automatic stream surface selection. Cai and Heng [CH97] present the first method that is based on implicit stream surfaces representations. The resulting surfaces can either be extracted as isosurfaces or rendered directly via volume rendering. However, the required stream function integral of the vector field can only be computed for curl-free flow. Theisel et al. [TWHS03], Weinkauf et al. [WTHS04], and Peikert et al. [PS09] automatically find seed curves from topological structures. Unfortunately, their methods may either extract too many or not enough stream surfaces. Recently, Edmunds et al. [EML*12] use isolines on domain boundaries as seed curves and propagate stream surfaces into surrounding space. Their method is limited to the existence non-vanishing flux on domain boundaries and has a domain filling behavior. In a follow-up work, a clustering approach of *local* flow properties is applied to find seed curve locations [ELM*12]. However, no surface-based measure is applied to evaluate the optimality of resulting stream surfaces. Since stream surfaces are *global* structures due to integration, our new approach is based on a global flow field analysis and optimization by measuring optimality of complete stream surfaces. In particular, our method is independent of vector field topology, or the existence of curl or outflow boundaries. Note that we are focusing on *one* representative and relevant stream surface, i.e., it is not the goal to densely cover the domain with surfaces that potentially hide each other. Furthermore, we search for a view-independent solution.

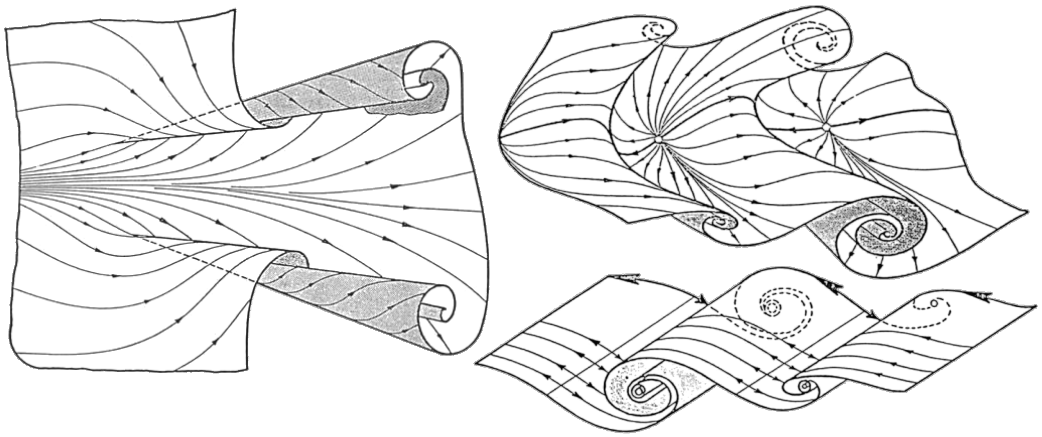


Figure 65: Classic Stream Surface Illustrations. Examples of hand-drawn flow illustrations using a *single* representative stream surface that visualize the global flow behavior. The images are courtesy of Uwe Dallmann [Dal83] and included in this thesis with the kind permission of the DLR Göttingen.

10.2 DESIDERATA

We continue to describe requirements and design decisions of our approach for relevant stream surface selection in more detail.

REQUIREMENT OF GLOBAL SURFACE-BASED APPROACH. All other automatic stream surface selection methods listed in the previous section have in common that they are *local* methods: stream surfaces are exclusively selected by the quality of seed curves, i.e., by considering $\mathbf{v}$ and its derivatives along the curves. Even if their seed curves perfectly fulfill such local criteria [ELM*12], nothing prevents the final stream surfaces from being integrated into areas where they are either less interesting or even lead to poor visualizations by hiding interesting structures. Instead, we argue that is is necessary to directly *evaluate* the quality of stream surfaces that are integrated from these seed curves. This requires a surface-based quality measure that is *global* by construction due to the domain-wide stream surface integration. We introduce such a surface-based quality measure in Section 10.4.

RESTRICTION TO SINGLE STREAM SURFACE. Common approaches for stream line selection focus on finding a set of stream lines that covers the domain in a dense and space-filling way. The situation for stream surfaces is different: even a single surface can cover the complete screen space. Moreover, it is known that even in low numbers stream surfaces tend to hide each other, leading to cluttered visualization. Also, having a look into classic literature where $3d$ flow illustrations are used reveals that most examples of $3d$ flows are only illustrated with a single carefully chosen stream surface [AS92, Dal83]. Some examples

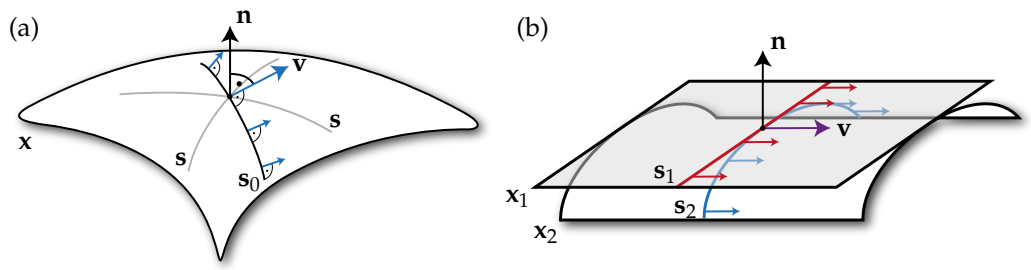


Figure 66: Stream Surface Parameterizations. (a) Different seed curves $\mathbf{s}$ describing the same stream surface. Among all stream surfaces there is a unique orthogonal-optimal one $\mathbf{s}_0$. (b) Different orthogonal-optimal seed curves produce different stream surfaces, here for the example of a constant $\mathbf{v}$.

of these hand-drawn illustrations of representative stream surfaces are shown in Figure 65. We want to find a *single* similarly relevant stream surface automatically. Note that this assumption fails for highly turbulent flows, for which surface-based approaches are generally considered unsuitable.

10.3 ON THE COMPLEXITY OF THE SEARCH SPACE

Before proposing a solution, we discuss the complexity of the problem of stream surface selection. In other words, we answer the question how many stream surfaces exist for a $3d$ vector field. Consider a point $\mathbf{q} \in \mathcal{D}$ and a normal $\mathbf{n}$ with $\mathbf{n}^\mathrm{T}\,\mathbf{v} = 0$. We analyze how many different stream surfaces exist that pass through $\mathbf{q}$ and are constrained to have the surface normal $\mathbf{n}$ there. Every stream surface can be described by a seed curve $\mathbf{s}(s)$ with

$$\mathbf{s}(0) = \mathbf{q}\,, \quad \dot{\mathbf{s}}(0)^\mathrm{T}\,\mathbf{n} = 0\,, \quad \dot{\mathbf{s}}(0) \times \mathbf{v}(\mathbf{q}) \neq \mathbf{0}\,. \tag{68}$$

Note that different seed curves that fulfill (68) describe the same stream surface. In fact, a seed curve $\mathbf{s}'(s) = \boldsymbol{\phi}^{\lambda(s)}(\mathbf{s}(s))$ (for any differentiable function $\lambda(s)$ with $\lambda(0) = 0$) gives the same stream surface in a different parametrization. Among all seed curves describing the same surface there is one unique representative: the seed curve fulfilling (68) and an additional orthogonality condition that is expressed by the ordinary differential equation (ODE)

$$\dot{\mathbf{s}}(s)^\mathrm{T}\,\mathbf{v}(\mathbf{s}(s)) = 0$$

for all s. We call such a seed curve an *orthogonal-optimal* seed curve $\mathbf{s}_0$. Figure 66 (a) illustrates the concept.

In order to find the number of different stream surfaces, we have to find the number of different orthogonal-optimal seed curves, i.e., curves that are the solution of the ODE with the initial value $\dot{\mathbf{s}}(0) = \mathbf{v}(\mathbf{q}) \times \mathbf{n}$. Note that unique solutions of this ODE are not well-defined: solving it as an initial value problem for a

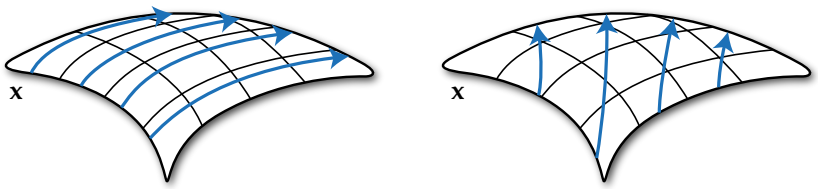


Figure 67: Stream Surface Quality. Left: a stream surface of *high* quality: the stream lines (•) are aligned with the lines of curvature (•). Right: a stream surface of *lower* quality where lines are not well-aligned.

certain s does not give a unique direction of continuation but a one-parametric family of directions. Thus, the set of different orthogonal-optimal seed curves is an infinite-dimensional infinite set. Figure 66 (b) showcases this property for a constant vector field in which two different stream surfaces that pass through the same point with the same normal are shown. From the number of different orthogonal-optimal seed curves it follows that the cardinality of the set of different stream surfaces through $\mathbf{q}$ with normal $\mathbf{n}$ is above the cardinality of $\mathbb{R}^n$ for any natural number n. This number of possible stream surfaces results in a very large search space and a high complexity of the problem of stream surface selection.

10.4 QUALITY MEASURES FOR STREAM SURFACES

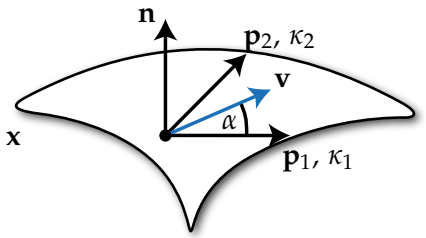
We continue to derive our quality measure for stream surfaces, which has a perceptual as well as a differential geometric interpretation.

PERCEPTUAL INTERPRETATION. Using *perceptual* arguments we investigate the question which intrinsic properties characterize a good stream surface. Their graphical representation is challenging because two pieces of information have to be simultaneously presented: the surface shape and the flow on the surface. The shape is usually represented by (non)-photorealistic rendering techniques [BWF*10]. To show the flow on the surface, most $2d$ steady flow visualization techniques can be adapted to stream surfaces: examples are image based techniques [LHD*04], stream lines [SLCZ09], or elementary techniques [PGL*12]. However, the perception of the shape and the flow are not independent – see for example the work by Laramee et al. [LHD*04], where image-based flow visualizations on the surface limit the perception of the shape. Hence, it is desirable to find stream surfaces where the representation of their shapes and the embedded flows yield similar structures and therefore do not interfere with each other.

In non-photorealistic rendering of shapes, standard techniques for surface representation are line rendering and hatching [DC90, SS02]. It is known that lines of curvature are good candidates for representing the surface shape [RKS00]. A

variety of user-studies show that lines of curvature improve the perceptibility of surfaces [Int97, GIHL00, KHSI04b, KHSI04a, SW04]. If these lines are stream lines at the same time, they can represent *both* shape and flow. Hence, the basic idea of our approach is to search for stream surfaces where the stream lines are aligned with the principal directions as good as possible. Figure 67 illustrates the setting. The differential geometric interpretation of this local criterion will motivate the global optimality of the surfaces minimizing this alignment measure. Also, our experiments indicate that this measure characterizes representative stream surfaces of a flow domain. Note that recently curvature approximations have already shown to be beneficial for integral surface rendering [HGH*10].

To formulate a local alignment error, consider a stream surface $\mathbf{x}(s,t)$ at a certain point, and let κ_1, κ_2 be the principal curvatures and $\mathbf{p}_1$, $\mathbf{p}_2$ the corresponding principal directions of $\mathbf{x}$ in its tangent space [BKP*10]. Furthermore, let α be the angle between $\mathbf{v}$ and one of the principal directions:



Then, we define the *local alignment error*

$$e_a := \cos(\alpha)\,\sin(\alpha)\,(\kappa_2 - \kappa_1)\,. \tag{69}$$

Note that e_a^2 neither depends on the choice of the principle direction nor on its orientation, and it vanishes if $\mathbf{v}$ is aligned with one of the principal directions. It also vanishes at umbilical points ($\kappa_1 = \kappa_2$) by construction, because alignment is not well-defined at these points.

DIFFERENTIAL GEOMETRIC INTERPRETATION. The error e_a has a related differential geometric interpretation providing an additional more global motivation for the measure: following Euler's theorem [Eul67] the *normal curvature* κ_n at $\mathbf{x}$ in the direction $\mathbf{v}$ is

$$\kappa_n = \kappa_1\,(\cos(\alpha)\,)^2 + \kappa_2\,(\sin(\alpha)\,)^2\,,$$

which gives $e_a = \frac{\mathrm{d}}{\mathrm{d}\alpha}\kappa_n$. Hence, e_a is the directional derivative of κ_n in flow direction. Surfaces that minimize e_a^2 therefore have low normal curvature variation in the direction of $\mathbf{v}$. The process of curvature variation minimization is generally termed *surface fairing*, which follows the *principle of simplest shape*: the surface should be free of any unnecessary details or oscillations (see, e.g., [MS92, WW92]). Therefore, minimizing e_a^2 globally yields stream surfaces that

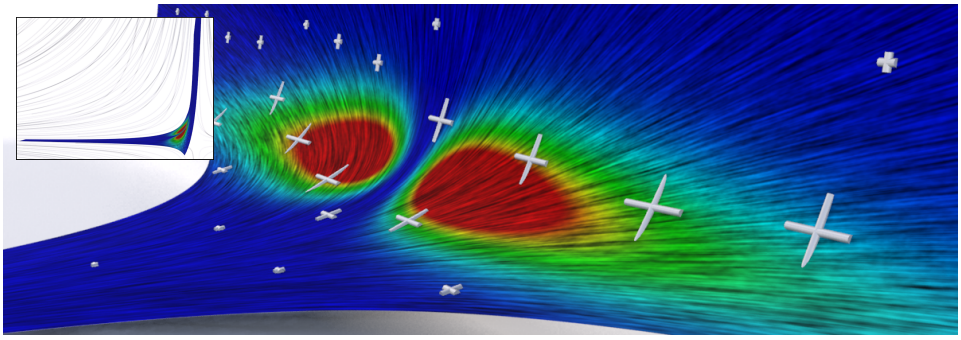


Figure 68: Stream Surface Alignment Quality Measure. Shown is a simple stream surface in a SADDLE vector field. Superimposed are principal curvature directions (•, scaled by $|\kappa_1 - \kappa_2|$) and the local vector field (LIC). On the surface the squared local alignment error e_a^2 is color coded (low error •, high error •), which depends on the alignment of the flow and *any* principal direction.

do not only capture local flow details but rather represent the global flow behavior.

The error e_a is not yet suitable for minimization because it requires a local estimation of the curvature tensor of $\mathbf{x}$. We consider the first and second order partial derivatives of the implicit parametrization of $\mathbf{x}$ from which the surface curvature can be computed:

$$\mathbf{x}_s = \dot{\mathbf{s}}\,, \quad \mathbf{x}_t = \mathbf{v}\,, \quad \mathbf{x}_{ss} = \ddot{\mathbf{s}}\,, \quad \mathbf{x}_{st} = \mathbf{J}\,\dot{\mathbf{s}}\,, \quad \text{and} \quad \mathbf{x}_{tt} = \mathbf{J}\,\mathbf{v}\,. \tag{70}$$

From (70) it is straightforward to compute κ_1, κ_2, $\mathbf{p}_1$, $\mathbf{p}_2$. Inserting these values into (69) gives the closed-form expression [1]

$$e_a = \frac{\mathbf{n}^{\mathrm{T}}\,\mathbf{J}\,(\mathbf{v}\times\mathbf{n})}{\|\mathbf{v}\|^2}\,. \tag{71}$$

Equation (71) shows a *remarkable* property: the local alignment error at $\mathbf{x}$ with normal $\mathbf{n}$ does not depend on the seed curve but only on $\mathbf{n}$, $\mathbf{v}$, and $\mathbf{J}$. In other words: all stream surfaces through a point $\mathbf{x}$ with normal $\mathbf{n}$ have the same local alignment error! See Figure 68 for an example of this local measure on a simple stream surface (Figure 73 shows the measure on more complex stream surfaces).

Based on e_a, we compute the *average squared alignment error* E_a by integrating e_a^2 over the stream surface:

$$E_a = \frac{1}{A}\int_{t_0}^{t_1}\int_{s_0}^{s_1} e_a^2\,\|\mathbf{x}_s\times\mathbf{x}_t\|\,\mathrm{d}s\,\mathrm{d}t \tag{72}$$

for the stream surface $\mathbf{x}(s,t)$ with $(s,t)\in[s_0,s_1]\times[t_0,t_1]$ and surface area A. The measure E_a is nonnegative and comparable for stream surfaces of different area.

1 The step from (69) to (71) is a straightforward computation for which we provide an accompanying Maple sheet corresponding to [MSRT13a] in the additional material folder `addmaterial/surfsel`.

The error E_a will be the target function for minimization. However, E_a has trivial minimizers, e.g., stream surfaces in laminar flows with almost vanishing Jacobian: in these areas, nearly planar stream surfaces minimize (72) because e_a vanishes. To exclude these trivial solutions, we expect a representative stream surface to have non-vanishing average normal curvature. Setting

$$E = \mathbf{x}_s^\mathrm{T}\,\mathbf{x}_s\,, \qquad F = \mathbf{x}_s^\mathrm{T}\,\mathbf{x}_t\,, \qquad G = \mathbf{x}_t^\mathrm{T}\,\mathbf{x}_t\,,$$
$$L = \mathbf{n}^\mathrm{T}\,\mathbf{x}_{ss}\,, \qquad M = \mathbf{n}^\mathrm{T}\,\mathbf{x}_{st}\,, \qquad N = \mathbf{n}^\mathrm{T}\,\mathbf{x}_{tt}\,,$$

the normal curvature is the ratio of first and second fundamental forms [BKP*10], i.e.,

$$\kappa_n = \frac{L\;\mathrm{d}s^2 + 2\,M\;\mathrm{d}s\,\mathrm{d}t + N\;\mathrm{d}t^2}{E\;\mathrm{d}s^2 + 2\,F\;\mathrm{d}s\,\mathrm{d}t + G\;\mathrm{d}t^2}\,. \tag{73}$$

Since the tangential direction $\mathbf{v}$ corresponds to the direction $(\mathrm{d}s, \mathrm{d}t) = (0, 1)$ in the parametrization, inserting (70) in (73) gives

$$\kappa_n = \frac{\mathbf{n}^\mathrm{T}\,\mathbf{J}\,\mathbf{v}}{\|\mathbf{v}\|^2}$$

as the closed-form expression of normal curvature using vector field quantities. To obtain a comparable measure we compute the *average squared normal curvature*

$$K_n = \frac{1}{A}\int_{t_0}^{t_1}\int_{s_0}^{s_1} \kappa_n^2\;\|\mathbf{x}_s \times \mathbf{x}_t\|\;\mathrm{d}s\,\mathrm{d}t \tag{74}$$

We will later minimize deviation of K_n from a prescribed K_0.

A third quality measure solves the ambiguity of the seed curve of a stream surface: we expect a good seed curve to be as-perpendicular-as-possible to the flow. We define

$$E_p = \frac{1}{\ell}\int_{s_0}^{s_1} \left(\frac{\mathbf{v}^\mathrm{T}\,\dot{\mathbf{s}}}{\|\mathbf{v}\|\,\|\dot{\mathbf{s}}\|}\right)^2 \|\dot{\mathbf{s}}\|\;\mathrm{d}s \tag{75}$$

as the average deviation from perpendicularity of $\mathbf{v}$ along $\mathbf{s}$, where ℓ is the arc length of $\mathbf{s}$. W.l.o.g. we assume $\mathbf{s}$ to be arc length parameterized, such that $\ell = s_1 - s_0$. Finally, we have to consider the area A of the stream surface as a fourth quality measure.

We aim to find a seed curve of the stream surface that minimizes these four quality measures. Direct numerical optimization of this problem is hardly feasible due to the search space complexity and the nonlinearity of involved terms. Instead, we develop a combinatorial algorithm that yields near-optimal stream surfaces w.r.t. our quality measures.

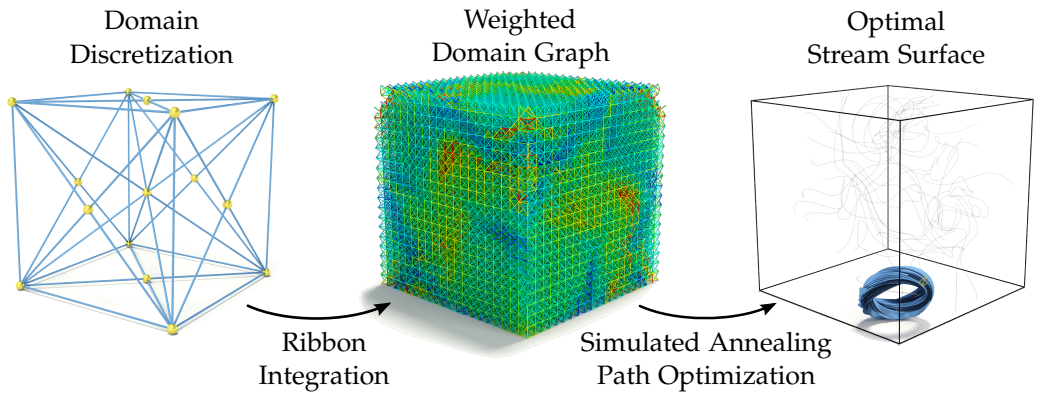


Figure 69: Algorithm Overview. We discretize the domain using a domain graph of refined cubic cells (left) and perform a stream ribbon integration for each edge to assign edge costs w_i by (76) (middle, low costs •, high costs •). A simple path of minimal costs computed using simulated annealing then yields a seed structure of the optimal stream surface (right).

10.5 AUTOMATIC SEED CURVE SELECTION

Using the stream surface quality measures we continue to present our stream surface selection strategy. The algorithm consists of the following major steps that are executed automatically for a user-provided flow domain and a set of parameters. See Figure 69 for an overview of the method. First, we create a spatial graph that densely covers the domain. Then, a stream ribbon integration is performed for each edge of this graph. The quality of each stream ribbon defines a cost value for each grid edge. We then perform a global optimization for paths of approximate minimal costs using a simulated annealing algorithm. In a final smoothing step on the resulting path, we obtain the seed curve for the integration of the resulting stream surface.

DOMAIN GRAPH. In order to perform a global optimization on the space of stream surfaces we need a suitable discretization of this space or equivalently of the space of seed curves. For this discretization we use the following *inclusion property for seed curves* of stream surface:

Given a seed curve $\mathbf{s}(s)$, $s_0 \leq s \leq s_1$, of a stream surface $\mathbf{x}$, this curve can be subdivided at a point $\mathbf{s}(s_s)$ into two curves $\mathbf{s}_a(s_a)$, $s_0 \leq s_a \leq s_s$ and $\mathbf{s}_b(s_b)$, $s_s \leq s_b \leq s_1$, such that $\mathbf{s} \equiv \mathbf{s}_a \cup \mathbf{s}_b$ holds. Likewise, if $\mathbf{s}_a$ and $\mathbf{s}_b$ are used as seed curves for new stream surfaces $\mathbf{x}_a$ and $\mathbf{x}_b$, then also $\mathbf{x} \equiv \mathbf{x}_a \cup \mathbf{x}_b$ holds because both $\mathbf{x}_a$ and $\mathbf{x}_b$ share the unique stream line starting at $\mathbf{s}(s_s)$. One can therefore always extend a stream surface with another stream surface if their corresponding seed curves connect in a common point.

This allows a domain discretization with short curve segments that join at common points. We use linear curve segments and the following subdivision scheme:

we first create a grid of uniform cubic cells that covers the whole domain. The edges of these cells are all axis-aligned. In order to provide more directional degrees of freedom, each cell is subdivided by inserting vertices at the six face centers and the cell center, which are connected by face diagonals and cell diagonals, respectively. Then each cell consists of 15 vertices and 44 edges, which are shared with neighboring cells. Figure 69 (left) shows our discretization scheme.

The sets of all vertices $\mathcal{V}$ and all edges $\mathcal{E} \subset \mathcal{V}^2$ define an undirected *domain graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices embedded in $\mathcal{D}$. Let $\mathcal{P}$ be the set of all *simple* paths in $\mathcal{G}$, i.e., paths with no vertex repetitions. We use simple paths to approximate the seed curves: any simple sequence of edges yields a piecewise linear seed structure candidate. Possible candidates are evaluated w.r.t. the quality of the stream surfaces that they define. The quality measure can be evaluated for each edge independently to define edge costs. Due to the inclusion property, the quality or equally the costs of a path results in a summation of associated edge costs.

EDGE COSTS. The cost $w_i > 0$ of each edge $i \in \mathcal{E}$ aggregates intrinsic properties of narrow stream ribbons $\mathbf{x}_i \subset \mathcal{D}$ (i.e., of stream surface with short seed curves). We calculate each $\mathbf{x}_i$ by performing a stream surface integration using each edge as the seed geometry. Integration is performed for a predefined maximum time range and is stopped at the domain boundary. There are no special requirements at this stage: we use Hultquist's algorithm [Hul92]; any other adaptive method for stream surface integration is applicable as well. A stream ribbon is integrated for each edge of the domain graph. This is the most time-consuming step of our method (see Section 10.6.4). However, this way edge costs are based on truly *nonlocal* features of the vector field.

There are two remarks on this step that are worth noting: as ribbons are integrated independently, we use a parallelized computation. Also, we observe that reliable edge costs generally do not require ribbons of high resolution. Therefore, a relatively coarse tessellation of the stream surface meshes is sufficient.

Edge costs are modeled as weighted combinations of the quality measures defined in Section 10.4, which are evaluated on each stream ribbon $\mathbf{x}_i$. Edge costs shall be minimal if a linear combination of quality measures is minimized.

We compute a discrete approximation of the surface integrals (72) and (74) by quadrature, which samples local values of e_a and κ_n at each triangle center and weights samples by triangle area. This yields values E_a^i and K_n^i for the ribbon $\mathbf{x}_i$ with surface area A^i. Similarly, E_p^i is an approximation to the line integral (75), which is also evaluated by quadrature along the domain graph edge.

The absolute values of these measures differ and cannot be compared between data sets. In order to obtain parameters that are independent of the data, a measure normalization is required that yields *relative* values. We normalize E_a^i,

K_n^i, E_p^i, and A^i to the range $[0,1]$ to obtain $\bar{E}_a^i$, $\bar{K}_n^i$, $\bar{E}_p^i$, and $\bar{A}^i$. For the curvature-based measures E_a^i and K_n^i we apply an additional log-transformation. This is due to the fact that curvatures are not bounded and can become very large. Moreover, they are not equally distributed, meaning that only in small regions large values appear, while large regions have rather small values for e_a and κ_n.

We model the final cost for an edge i of length ℓ_i by weighted quadratic contributions as

$$w_i = \ell_i \sum_{c \in \{a,n,p,A\}} \beta_c \left(F_c^i\right)^2 \tag{76}$$

with $F_a^i = \bar{E}_a^i$, $F_n^i = \bar{K}_n^i - \tau$, $F_p^i = \bar{E}_p^i$, and $F_A^i = 1 - \bar{A}^i$. Cost minima of w_i minimize the alignment error (72). Moreover, higher costs are assigned to edges that are aligned with the flow and whose stream ribbons have a smaller surface area. The weights β_x, $x \in \{a,n,p,A\}$ define the relative weighting of cost components. Generally, a good choice is an uniform weighting by $\beta_x = 1$: we use this setting unless stated otherwise for some experiments in Section 10.7. Then $\tau \in [0,1]$ remains the only parameter provided by the user: it steers the desired amount of average normal curvature in flow direction with $\tau = 1$ for higher and $\tau = 0$ for lower values. Essentially, τ captures the prescribed average squared normal curvature K_0 (see Section 10.4) after normalization. If an edge is aligned with the flow, i.e., $\bar{E}_p^i > \delta$ (we use $\delta = 0.8$), we ignore its weight and discard the edge from further processing by setting $w_i = \infty$. Figure 69 (middle) shows a graph with color-coded costs for each edge.

PATH COSTS. We define the total costs of a path $p_k \in \mathcal{P}$ as

$$c_\gamma(p_k) = (1-\gamma) \sum_{i \in \mathcal{E}_k} w_i + \gamma\, \kappa(p_k) \ .$$

$\mathcal{E}_k \subset \mathcal{E}$ is the set of edges of p_k. This is a linear blend between edge costs and a normalized measure of discrete polyline curvature κ. We include the additional path curvature term to penalize undesirable "space-filling" paths in areas of constant low edge costs (e.g., in laminar flow areas). The weight $0 \leq \gamma < 1$ is a user parameter that trades minimal edge costs versus straightest seed curves (see Section 10.6.1 for results of different choices for γ). A simple estimation of the (normalized) curvature of the polyline is sufficient, we use

$$\kappa(p_k) = \frac{1}{\pi} \sum_{e_i, e_{i+1} \in \mathcal{E}_k} \pi - \sphericalangle(e_i, e_{i+1}) \ .$$

Here, $\sphericalangle(e_i, e_{i+1})$ denotes the angle between two consecutive edges on p_k.

GLOBAL OPTIMIZATION PROBLEM. Our goal is to find a seed curve of the best stream surface w.r.t. the defined quality measure, i.e., most importantly, we prefer surfaces in which stream lines are aligned with lines of curvature. This requires searching for the globally best seed curve in the entire domain. We formulate this global optimization problem as a combinatorial problem: the space of seed curves is discretized as the space of simple paths in the domain graph, which covers the domain densely. The quality of a seed curve is evaluated as the sum of edge costs in a path. The prescribed arc length ℓ of seed curves is measured as the number of edges in a path, giving $n = \lfloor \ell / \ell^e_{\text{avg}} \rceil$ for an average graph edge length ℓ^e_{avg}, which depends on the domain dimension and graph resolution. Formally, we want to optimize

$$p^\star = \underset{p_k \in \mathcal{P}}{\operatorname{argmin}} \ \left\{ c_\gamma(p_k) \ \middle| \ |\mathcal{E}_k| = n \right\} \tag{77}$$

for an optimal simple path $p^\star$ of length n. This is a combinatorial assignment problem on a large finite search space $\mathcal{P}$. In the literature the dual problem is called the *Heavy Path Problem* [KBL12]. It is known to be NP-hard, i.e., it is practically infeasible to compute the *exact* solution for large n. Instead, we present an algorithm for computing an approximate solution in the following sections.

LOCAL MINIMAL PATHS. A key idea for a practical solution of (77) is to prune the search space. Instead of all simple paths of length n in $\mathcal{P}$, we consider only a subset $\mathcal{Q} \subset \mathcal{P}$ that only includes the simple minimal path starting in v of length n for each vertex $v \in \mathcal{V}$. This implies $|\mathcal{Q}| = |\mathcal{V}|$, where the paths in $\mathcal{Q}$ are determined by a *local* minimization. This pruning step is justified by the fact that any other paths $p \notin \mathcal{Q}$ do not even constitute local minima and are therefore not considered as candidates for a global minimum.

The locally minimal paths can be computed for each vertex using, e.g., Dynamic Programming, an algorithmic standard technique that was recently used for finding Heavy Paths [KBL12]. However, in practice this is only feasible for $n \leq 6$ due to the exponential combination complexity, which results from the high branching factor of our used domain graph (most vertices have 16 neighbors). We therefore approximate $\mathcal{Q}$ with a greedy iterative deepening depth-first search [Kor85], which restarts a depth-first search phase after each d steps from the current optimal solution. This is justified by the fact that edges of minimal costs are generally distributed along line structures in the graph – hence the depth-first search phase – due to the penalization of flow-aligned edges. We use $d = 5$ in all our experiments. This allows computing approximate local minimal paths of length $n > 15$ efficiently. We implemented a lazy evaluation of $\mathcal{Q}$, i.e., locally minimal paths are computed only on demand and then stored for subsequent evaluations.

10.5.1 Global Path Optimization by Simulated Annealing

Hard combinatorial problem similar to (77) are often encountered in many applications [HB10]. Different classes of algorithms exists that find approximate solutions, e.g., genetic algorithms [Gol89]. An often successfully used metaheuristic to approximately solve similar problems is the *Simulated Annealing* (SA) algorithm. It is a physically inspired algorithm that models the controlled slow cooling of heated materials. Slow cooling reduces defects in the crystal-like structure of the material, which can be interpreted as a form of internal energy that is optimized. The SA algorithm was introduced by Kirkpatrick et al. [KGV83] as a general heuristic global optimization tool for hard problems. For instance, SA was recently used in the visualization community by Sigg et al. to solve the NP-hard problem of generating optimal cutaway illustrations [SFCP12]. Further applications of SA can be found in the survey of Suman and Kumar [SK06]. We briefly describe the SA algorithm in the context of our problem and refer to the survey for more details on the method.

The SA algorithm introduces a system temperature $T > 0$ that serves as a control parameter. A candidate solution is iteratively improved (the amount of change depends on T), and T is slowly decreased following a temperature schedule.

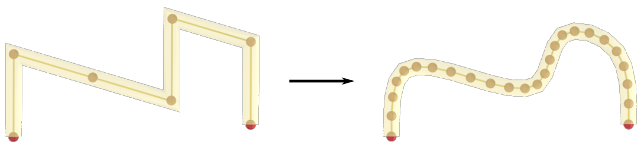
SINGLE OPTIMIZATION STEP. We use the SA algorithm for an iterative update of a candidate path that starts at the current vertex v_c, which is initialized randomly. A new candidate path is randomly sampled in the domain by selecting a new starting vertex v_n in the Euclidean neighborhood of v_c according to a normal distribution with mean $\mathbf{x}_c$ and variance $\sqrt{T}/2$, weighted by the length of the diagonal of the domain bounding box. Higher temperatures increase the probability for selecting distant candidate vertices, while at lower temperatures variations are more local. The probability that v_n is accepted for the next iteration depends on the cost difference $d = c_\gamma(v_c) - c_\gamma(v_n)$. Here, $c_\gamma(v)$ denotes the cost of the minimal simple path $p_v \in \mathcal{Q}$ that starts in vertex v. The probability is given by the Metropolis transition probability $P_{v_c \to v_n} = \min\{1, e^{d/T}\}$ proposed by Metropolis et al. [MRR*53] (we omit the Boltzmann constant of the original formulation). $P_{v_c \to v_n}$ guarantees that better solutions ($d > 0$) are always accepted. Likewise, worse solutions ($d < 0$) can also be accepted, as $P_{v_c \to v_n} > 0$ still holds, and the probability for these events is enlarged for higher temperatures T. This property enables SA to not get stuck in local minima but eventually leave them once entered.

TEMPERATURE SCHEDULE. The classic SA algorithm consists of a heating (annealing) and a cooling phase that schedule the temperature [KGV83]. We found the heating phase results in almost equal initial temperatures in all our experiments – we therefore usually skip heating and start the cooling phase with a

fixed initial temperature value T_0. In the cooling phase multiple optimization steps are performed at a fixed temperature, and we count the number of accepted steps m_a. If $m_a > m_s$ for a fixed value m_s, the temperature is lowered by multiplying it with a constant factor $0 < \lambda < 1$. The scaling of the temperature by λ is also performed if the number of total steps exceeds a maximum value $m_{max} > m_s$. This scheme results in an exponential temperature decay. We stop the iteration if m_{max} is not reached for three consecutive times and consider the solution to be converged to the optimal path $p^\star$. We used the constant factors $T_0 = 1/2$ and $\lambda = 0.9$ that performed well in all our experiments. We analyze choices for m_s and m_{max} in Section 10.6.3.

10.5.2 Seed Curve Generation

The path $p^\star$ represents a piecewise linear seed curve. We interpret this as an approximation to a smooth, C^2-continuous, seed curve $\mathbf{s}^\star$ that is obtained by smoothing $p^\star$ using an univariate subdivision scheme. Due to the close proximity of the spatial locations and tangential directions of $p^\star$ and $\mathbf{s}^\star$, we expect the resulting stream surfaces to have similar behavior.



We use a simple corner cutting scheme that yields C^2-continuous cubic splines in the limit (see, e.g., Sabin [Sab10]) with an additional endpoint interpolation rule. In each iteration, the scheme generates new points by the rules

$$\mathbf{y}_{2i}^{k+1} = \tfrac{1}{2}\mathbf{y}_i^k + \tfrac{1}{2}\mathbf{y}_{i+1}^k$$
$$\mathbf{y}_{2i+1}^{k+1} = \tfrac{1}{8}\mathbf{y}_i^k + \tfrac{6}{8}\mathbf{y}_{i+1}^k + \tfrac{1}{8}\mathbf{y}_{i+2}^k \,.$$

Typically, three subdivisions provide a sufficiently close approximation to the smooth curve. We obtain the resulting optimal stream surface $\mathbf{x}^\star$ by starting an integration from $\mathbf{s}^\star$.

10.6 RESULTS

We evaluate our method by a parameter description and present results of applying our approach to both analytical and real world vector fields. Additionally, we provide a quantitative analysis of the convergence behavior of the optimization and timings of our method.

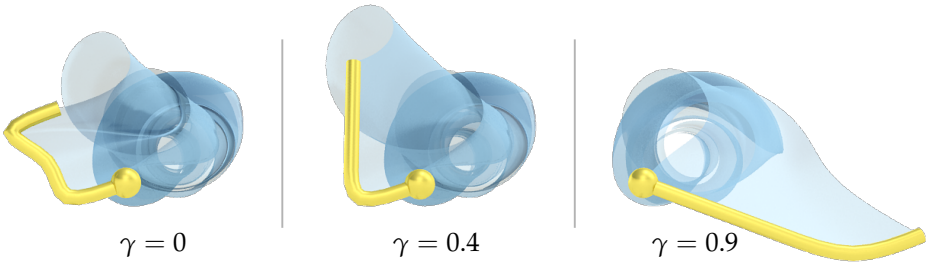


Figure 70: Path Curvature Parameter γ. The curves are generated from minimal paths (•) at the *same* highlighted vertex with different γ. Shown stream surfaces (•) are the *local* optima of c_γ given these curvature constraints in the DELTAWING flow.

10.6.1 Parameters

Our method requires specification of the domain graph resolution. It strongly influences the runtime of the method, as it prescribes the number of grid edges and therefore the number of stream ribbon integrations and the size of the search space of the optimization. However, we found that even relatively low resolutions yield sound results, e.g., all the analytical examples have a graph resolution of 20^3 cells. The highest resolution used in this work was a $20 \times 40 \times 15$ cells graph for the ACOUTLET data set shown in Figure 72. In general, we observe that a good default value for the resolution is given by the length ratio of vector field grid cell diagonal to domain graph cell diagonal (DR) around five (see the DR column of the table in Section 10.6.4). We evaluate the influence of different resolutions in Section 10.7.

Also, depending on the concrete vector field characteristics, the integration direction (i.e., forward only or forward and backward) and the maximum integration time should be specified. For vector fields with an inflow / outflow area (e.g., in the CYLINDER data set) we integrate in forward direction only until the integration reaches the boundary. In all other cases we integrate into both directions.

The number of edges n of the optimized paths dictates the total length of the optimized seed curve. It is related to the graph resolution, which specifies the range of edge lengths, and it influences the performance of the optimization, as local minimal path computations are cheaper for smaller n. We require the user to supply a value for n. Note that once the edge costs are computed, the less time-consuming path optimization step can be redone with a different n and γ, as required (see Section 10.6.4).

For different path curvature parameters γ we illustrate resulting minimal paths that start at the *same* vertex in Figure 70. All paths respect the local edge cost distribution (e.g., they still have the orthogonal-optimal property by lying in a plane orthogonal to the flow). Increasing values of γ come with increased

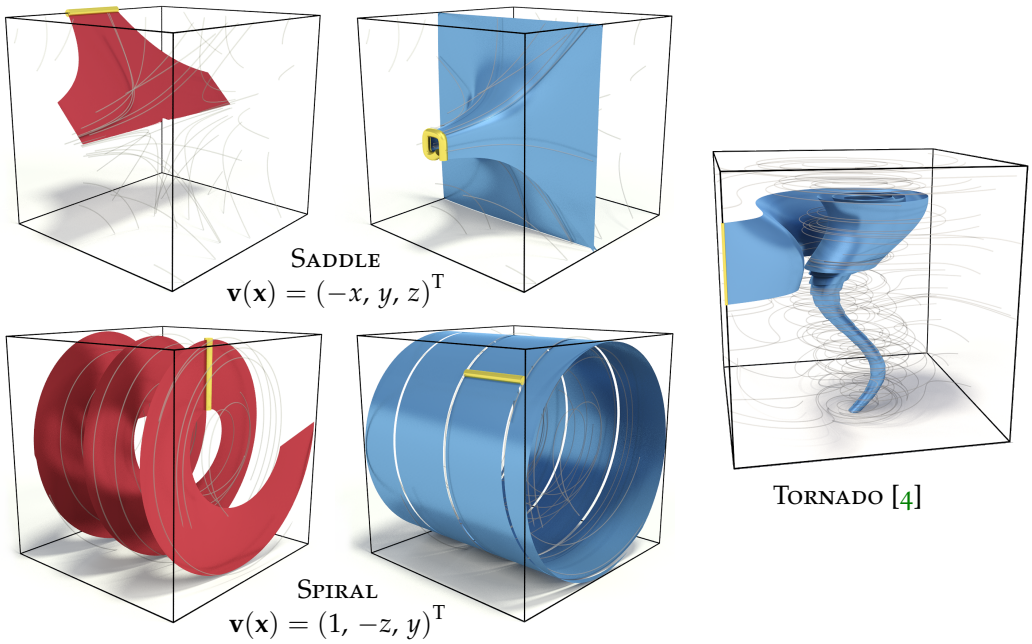


Figure 71: Automatic Selection Results for Synthetic Vector Fields. Alignment optimal stream surfaces selected by our method for $n = 7$ and integrated from the optimized seed curves (●) in the synthetic fields. Surfaces $\mathbf{x}^\star_{\tau=0}$ (●) minimize and surfaces $\mathbf{x}^\star_{\tau=1}$ (●) maximize mean normal curvature, respectively. The local alignment quality of the $\mathbf{x}^\star_{\tau=1}$ results is shown in Figure 73.

edge costs and therefore less optimal resulting stream surfaces. However, the minimal path in terms of path costs only might not always be the most desirable: for example, in areas of constant low edge costs the minimal path would be a space-filling path with high curvature, which we found to be a spurious result. In all our experiments we observed that using a value of $\gamma = 0.2$ avoids this issue and yields all results presented in this work.

10.6.2 Stream Surface Selection Results

We continue to present results of our approach and refer to video accompanying [MSRT13a] for more examples [2].

SYNTHETIC DATA SETS. Figure 71 shows a series of automatically selected stream surfaces in well-known simple analytic vector fields. These synthetic vector fields are well-suited to exemplify our quality concept of alignment (see Figures 68 and the closeups in Figure 73), combined with a variable prescribed mean normal curvature. To do so, we show results that minimize ($\tau = 0$) and maximize ($\tau = 1$) mean normal curvature while still optimizing the other quality

2 The video is located in the additional material folder `addmaterial/surfsel`.

measures (cf. Equation (76)). We call the resulting optimal surfaces $\mathbf{x}^\star_{\tau=0}$ and $\mathbf{x}^\star_{\tau=1}$. In the SADDLE field the optimal stream surface $\mathbf{x}^\star_{\tau=0}$ (with $\beta_n = 1$) is planar. In fact, it is *exactly* the same solution that is obtained for $\beta_n = 0$ in (76), i.e., if the mean normal curvature is ignored in the optimization. This confirms our theoretical proposition that the alignment measure is minimized by surfaces of vanishing curvature. Naturally, the absolute alignment measure E_a is higher for $\mathbf{x}^\star_{\tau=1}$. Still, the surface is the best-aligned solution given the additional mean normal curvature constraint.

Note that the optimized seed structures are not necessarily straight line segments only. We also note that these simple linear vector fields exhibit a high degree of symmetry and that the same optimal solutions can be generated by different seed structures. Moreover, a given solution might not be the unique globally optimal solution. Our algorithm selects one of the possible minimal solutions. As practical problems do in general not have such perfect symmetries, we expect to find unique solutions in these kinds of data sets.

REAL WORLD DATA SETS. We apply our selection method to a number of complex vector fields of different domains and varying characteristics. The resulting stream surfaces are shown in Figure 72. Again the optimal seed structures are general curves. The CYLINDER vector field represents the flow around a square cylinder [CSBI05]. It is a well-known and extensively studied phenomenon [BFTW09, ELM*12, SRGT12] (see also Figure 60). As we choose to integrate in forward direction only, both stream surfaces $\mathbf{x}^\star_{\tau=0}$ and $\mathbf{x}^\star_{\tau=1}$ are seeded at the boundary of the inflow area to maximize the area constraint. Additionally, both surfaces conform to the chosen τ parameter (low and high mean normal curvature) while still minimizing the alignment error. This property does also hold for the DELTAWING data set. It is a flow simulation around a triangle-shaped airplane (see [BWF*10, GKT*08, GTS*04, HGH*10, SRWS10, SRGT12] and Figures 60 and 63 for other approaches and results using a similar data set). Here, the mean normal curvature of $\mathbf{x}^\star_{\tau=0}$ vanishes. In contrast, the $\mathbf{x}^\star_{\tau=1}$ stream surface is well aligned with the two dominant vortex features. It is a single stream surface where both vortical parts are connected by laminar flow areas above and beneath the airplane. The ACOUTLET data set represents the flow in the outlet area of an air conditioning unit. It is used to predict the degeneration of filters in the dissipation grid layer (not shown in the rendering). With a resolution of $1.6 \cdot 10^7$ grid cells it is the largest data set we tested. Our method is able to select a curved seed structure of a stream surface with a high outflow rate. The outlet area of a hydroelectric turbine is simulated in the TURBINE data set where the flow is split at a bifurcation. Here, $\mathbf{x}^\star_{\tau=1}$ is a stream surface leaving the domain on both sides of the bifurcation. The ANEURYSM is a blood flow simulation at a human cerebral aneurysm, which is a weakness of the vessel wall that potentially leads to rupture and life-threatening bleeding. The selected stream surface $\mathbf{x}^\star_{\tau=1}$ is clinically relevant as it covers a large fraction of the volume of the

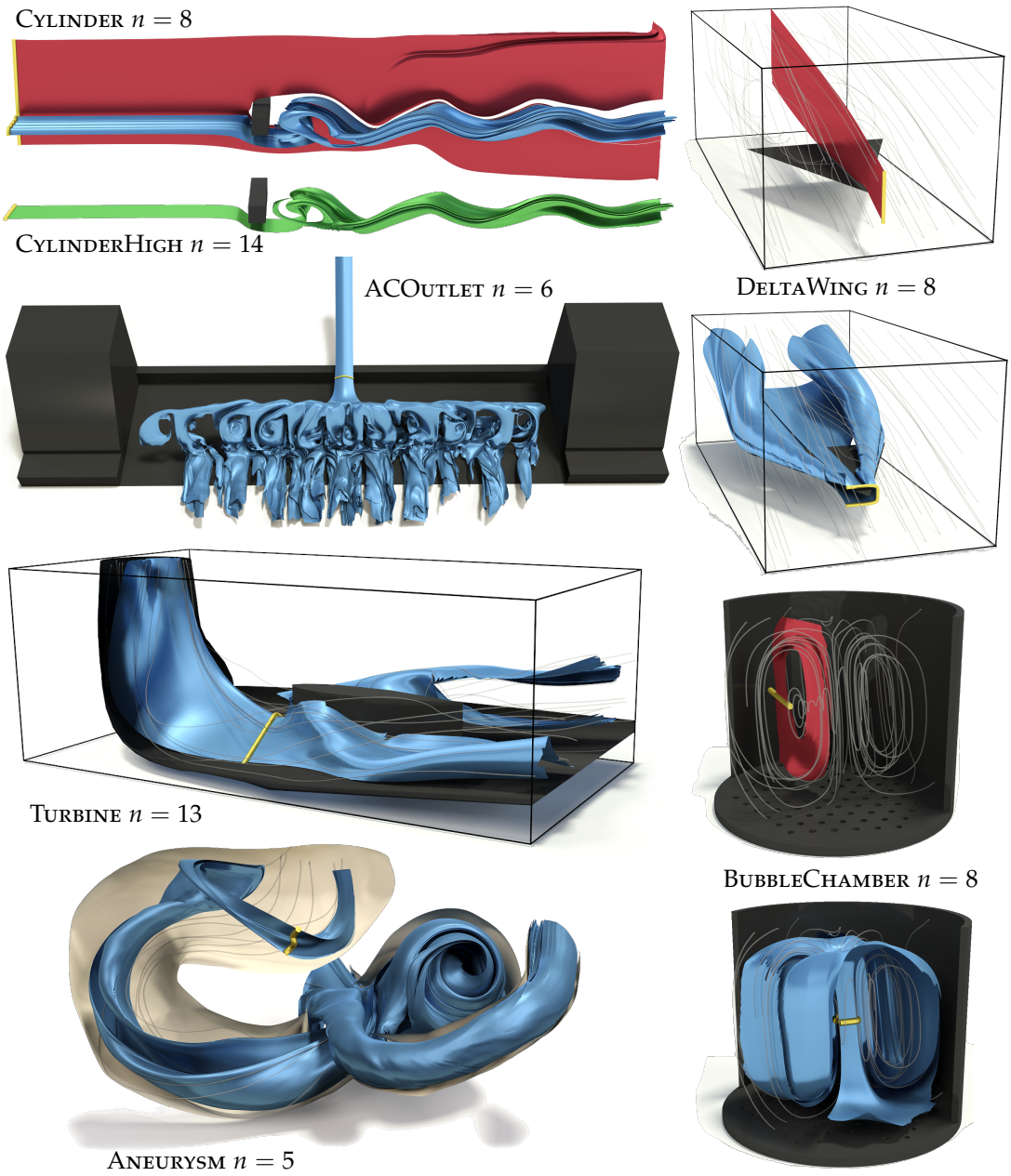


Figure 72: Automatic Selection Results for Simulated Vector Fields. Surfaces $\mathbf{x}^{\star}_{\tau=0}$ are colored •, surfaces $\mathbf{x}^{\star}_{\tau=1}$ are colored •, and the computed optimal seed curves are colored •. The • stream surface in CylinderHigh is $\mathbf{x}^{\star}_{\tau=1}$ extracted from a domain graph of higher resolution compared to Cylinder. The closeups in Figure 73 show the local alignment quality of the $\mathbf{x}^{\star}_{\tau=1}$ results.

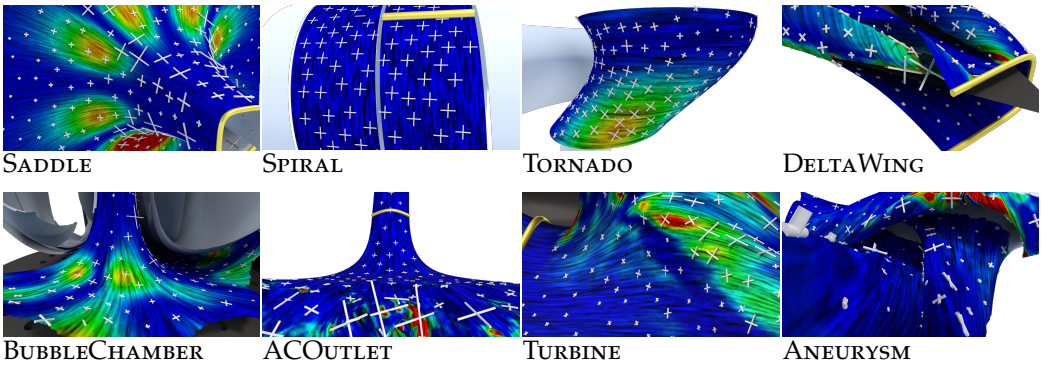


Figure 73: Alignment Measure. The closeups show the local alignment error e_a^2 (low error •, high error •), and scaled principal directions (cf. Figure 68) of the optimal $\mathbf{x}^\star_{\tau=1}$ solutions, which are shown in the Figures 71 and 72. Large surface regions are well-aligned to the flow.

aneurysm. The flow of the measured BUBBLECHAMBER data set of a bioreactor (see, e.g., [SWH05]) has different flow characteristics compared to the previous in / out-flow dominant examples. Nevertheless, the selected $\mathbf{x}^\star_{\tau=0}$ and $\mathbf{x}^\star_{\tau=1}$ are still similar to the previous examples concerning the planarity of $\mathbf{x}^\star_{\tau=0}$ and the more feature capturing property of $\mathbf{x}^\star_{\tau=1}$. The alignment measure visualizations in Figure 73 illustrate the local quality of the selected globally optimal solutions. In all examples the largest portion of the principal directions are well-aligned with the flow as prescribed by the alignment measure and optimized by our selection algorithm.

Our method optimizes for flow alignment of the shape of stream surfaces in combination with additional intrinsic stream surface properties. We expect our automatically selected globally optimal results computed this way to be relevant for the shown application areas. This is indicated by the fact that the stream surfaces we find are very close to the stream surfaces presented in other approaches. See, e.g., the related work on the DELTAWING data set, for which our automatically selected stream surface is very similar to the *manually* selected stream surfaces in [BWF*10, GKT*08, GTS*04, HGH*10, SWS09, SRWS10]. Additionally, the same stream surface is very close to the characteristic stream surface in a similar flow that is *hand-drawn* by Dallmann [Dal83] and shown in Figure 65 (left).

10.6.3 Simulated Annealing Convergence

For optimization we apply a heuristic search in form of the SA algorithm, which is a randomized method. It is therefore mandatory to analyze the quality and optimality of the results of this algorithm: we performed an evaluation run of our method on a synthetic and a simulated data set. Given a domain graph we find the ground truth optimal minimal path $p^\star$ using a naïve search and

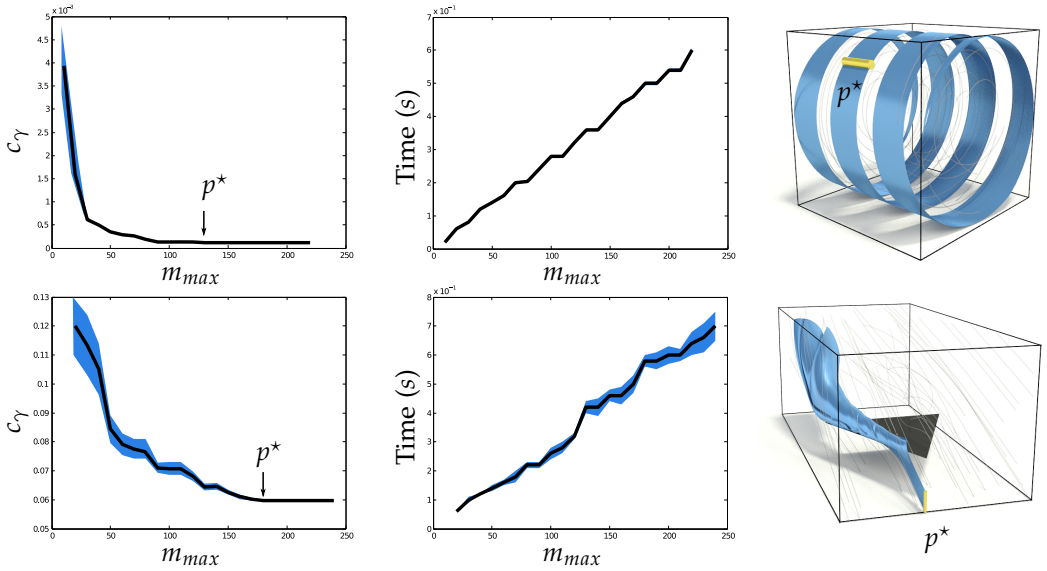


Figure 74: SA Convergence. For the SPIRAL (top) and DELTAWING (bottom) data sets we perform 100 SA optimizations per m_{max} and fixed $n = 4$. Graphs show the mean path costs and optimization times (•) and the 95% confidence interval (•). SA optimization converges to the globally optimal path $p^\star$ (shown right) for both data sets. Ground truth solution of $p^\star$ are found in 55s (SPIRAL) and 61s (DELTAWING), respectively, by a naïve search.

exact minimal path computations using full depth-first search. Due to the high branching factor of the graph this can only be done for a low number of edges – we use $n = 4$ in this experiment. As proposed by Sigg et al. [SFCP12], we use a fixed iteration number ratio of $m_{max}/m_s = 10$ and perform multiple SA optimizations per m_{max}. The results in Figure 74 show that the SA optimization does converge to a single cost minimum for high enough m_{max}. Moreover, the selected solutions are indeed global minima in the graph and are found in a fraction of the time required by naïve search. We use $m_{max} = 200$ in all given examples.

10.6.4 Timings

To evaluate the performance of our algorithm we measure the amount of time required for each step of our method on an Intel Core i7-2600 3.4GHz Linux PC with eight logical CPU cores and 16GB of main memory. The resulting timings for different data sets and different graph resolutions are given in the following table. For each data set the table shows grid cell to graph cell diagonal length ratio (DR) [3], the total number of graph edges, and the *total* computation time (in seconds) of each phase of our algorithm: stream ribbon integration (SR), edge

3 The DR value is not available for the ANEURYSM flow because the data is not sampled on a regular grid.

Data set	DR	$\lvert\mathcal{E}\rvert$	SR	EC	SA $n = 4$	SA $n = 8$	SA $n = 16$
TORNADO	2.0	$\approx 2 \cdot 10^5$	43	11	0.7	1.1	2.2
ACOUTLET	7.2	$\approx 2.5 \cdot 10^5$	308	80	1.6	2.3	3.1
ANEURYSM	na	$\approx 2.2 \cdot 10^5$	270	71	1.7	2.5	4.7
BUBBLECHAMBER	0.5	$\approx 1.1 \cdot 10^5$	60	17	0.9	2.8	6.3
CYLINDER	5.9	$\approx 7 \cdot 10^4$	179	45	0.7	1.6	3.1
CYLINDERHIGH	4.4	$\approx 3 \cdot 10^5$	406	103	1.9	2.8	3.3
DELTAWING	4.0	$\approx 1.4 \cdot 10^5$	170	44	0.6	1.0	2.1
TURBINE	5.7	$\approx 1.8 \cdot 10^5$	253	66	1.4	2.1	2.8

costs computation (EC), and simulated annealing optimization applied to three target edge numbers n (SA). Our method is not able to produce results instantaneously. However, as no user interaction is required, we consider our method to be an offline process for global data analysis. Still, all results in this work are computed within a few minutes by our method that has no previous knowledge of the given data set. Although we parallelize its computation, it is obvious that the full stream ribbon integration is the most time-consuming part of our method. It depends on both the graph resolution and data set characteristics. For example, integration in the simple TORNADO field ($2.5 \cdot 10^7$ triangles on all ribbons) can be done in a fraction of the time required for the CYLINDER ($1.1 \cdot 10^8$ triangles on all ribbons) data set, although the total number of ribbons is three times higher. Also, edge costs computations are proportional to both the number of stream ribbons and the number of total triangles, but in general they are cheaper than integration. The optimization using the SA algorithm turns out to be the fastest part of our algorithm. This is because the SA optimization reuses the results of previous computations in a highly condensed way in form of edge costs. Moreover, the whole graph is not necessarily visited in the optimization due to the stochastic nature of the algorithm. In the same way our algorithm can be further optimized: surface integration can be deferred by combining the SA optimization with edge cost computations and only needs to be performed when edge costs are evaluated for the first time. This lazy on demand surface integration then only has to be performed for a fraction of the whole domain.

10.7 DISCUSSION

We discretize the search space of seed structure candidates by edges of the domain graph. We will therefore only find solutions that are contained in the graph. We observe that increasing the domain graph resolution does only lead to locally finer solutions in the neighborhood of the coarser solutions. The global location of the optimum does not change as long as the domain is not undersampled.

Consider the CylinderHigh surface in Figure 72 as an example for an optimum with increased graph resolution of the Cylinder data set. Both surfaces have very similar position and shape, although the seed geometry changed slightly. However, the improvements in quality come at a linear expense of performance (see Section 10.6.4). We observed that the used graph resolutions are sufficient to find curves that are close to continuous seed structures of globally optimal stream surfaces. An explanation is that for each edge we obtain a *global* optimality estimation due to stream ribbon integration and stream ribbon evaluation. The change in global estimation for a higher graph resolution will only be large at the boundary of separating flow structures, such that in the majority of the domain the estimations stay similar. Therefore, the location of the optimal solution is also unlikely to change due to the higher domain graph resolution.

LIMITATIONS. We optimize for a single stream surface that strives to describe a data set best. This can also be seen as a limitation of our current approach, which is not optimizing for multiple distinct and "distant" stream surfaces *simultaneously*. A simple greedy adaption of our method is to apply the optimization iteratively and modify edge costs according to a distance measure of ribbons to already found optimal solutions. Still, our experience is that multiple stream surfaces quickly tend to occlude each other. Even with additional advanced rendering techniques the resulting visualizations become much harder to interpret compared to a single, globally optimal surface. We regard this problem as future research. Additionally, we note that our method will not always converge to a single distinct optimal stream surface in highly turbulent flows. However, it is well-known that surface-based approaches are not suitable to visualize these types of data sets.

10.8 SUMMARY

In this chapter, we presented a novel automatic approach for the selection of stream surfaces defined in $3d$ vector fields. Instead of using local flow properties for the selection, our method evaluates global surface-based quality measures on integrated stream surfaces. We showed that our new stream surface quality measure, which is based on flow alignment of principal directions, yields competitive results in a variety of data sets.

Until now we have only used intrinsic surface-based properties to define quality. Investigation of view- and application-dependent surface qualities is an interesting direction for future work, as our selection algorithm can handle different cost functions without modification. We would also like to extend our method to optimize the global solutions found in the discretized space locally in the neighborhood of this solution. Doing so would "polish" our solutions to optima in the continuous space.

11

CONCLUSIONS

In this thesis, we studied various aspects of the relation between geometric shapes and vector fields. Shape manipulation is a broad field in digital geometry processing, and in the first part of the thesis we identify specific problems of this field that we approach using vector field-based methods: different types of continuous deformations are proposed to manipulate geometric shapes of varying type and representation. In the flow visualization field, on the other hand, geometric shapes are considered that are defined by vector fields. In the second part of the thesis we propose interactive and automatic surface-based visualization techniques that are based on geometry processing concepts. Complementing the summaries on the individual contributions, we conclude this thesis with a short summary of both parts and a brief discussion of promising open research directions.

In the first part of this thesis, we applied kinematic vector fields for the computation of continuous shape manipulations in different contexts and applications. These include user-defined planar, volumetric, and surface deformations of both explicit and implicit shape representations, as well as vector field-based pose corrections. We showed that vector field-based approaches are well-suited to solve nonlinear problems that are generally considered hard to compute by single-step methods, e.g., near-isometric and volume-preserving shape deformations. In general, global deformation vector fields are efficient to compute as the solutions of linear systems or even as closed-form representations. Moreover, our results demonstrate that our vector field-based methods are on par or even outperform other nonlinear approaches in terms of deformation quality. The deformation quality is high as no search space reduction by, e.g., subspace or multiresolution techniques is performed. Also, no additional guiding structures like embedded skeletons or enclosing control cages are required by our continuous methods. In addition, they directly provide parameterized animations of shapes. In contrast to most other nonlinear deformation approaches, our vector field-based methods also guarantee certain desirable deformation properties by construction, e.g., volume-preservation or the prevention of local self-intersections. In conclusion, we contributed to the theory of vector field-based shape manipulation and presented a number of applications with a focus on digital geometry processing.

Certainly, the optimization of nonlinear measures stays a delicate problem, and most of our continuous methods focus on balancing accuracy and stability in favor of highest efficiency. We achieve accuracy and stability by using ODE integrators of high approximation order and adaptive step size control. In particular, this way we avoid iteratively computed solutions that are only locally minimal and which cannot be avoided by generic nonlinear solvers. Although we performed suitable accelerations in form of, e.g., GPU-based implementation wherever possible, integration-based methods tend to be computationally expensive. It is therefore an interesting challenge for further research to develop algorithms and concepts like specialized integrators and shape representations that accelerate vector field-based deformations without spoiling or derogating their quality. Beyond that, in the near future we also expect vector field-based deformations to become even more attractive with the advent of more efficient computational devices.

Additionally, we proposed the following general computational techniques that are independent of vector field-based shape manipulation and can also be applied to various other approaches of different fields:

- Our GPU-accelerated linear system setup in Chapters 3 and 4 can be used to speed up any cell-based energy operator assembly. A prominent example is stiffness matrix assembly of FEM deformation or fluid simulation methods. In fact, every energy discussed in Chapter 7 (and even the energy regularization itself) can be accelerated by a similar framework.
- Our backward Lagrangian integration scheme in Chapter 5 can be advantageous to solve advection problems for which the advecting flow is independent of the advected quantity.
- Our RBF center selection and partially GPU-based factorization update scheme in Chapter 6 can be used to improve both computation times and numerical stability of any mesh-less scattered data approximation using radial basis functions. The independence of the scheme of a specific function domain and data range make it applicable to a variety of different function reconstruction problems.
- In Chapter 7, we already demonstrated a wide range of geometry processing methods that can be enhanced by our energy regularization technique. Due to the generality and simplicity of the energy regularization concept, we expect it to be beneficial for various other variational problems.

In the second part of the thesis, we considered problems in which different types of geometric shapes are defined by flow vector fields. We applied concepts, which are well-known in geometry processing, to flow visualization tasks with a focus on surface-based visualizations. Contrary to, e.g., integral line-based methods, surface-based techniques are currently seldom used in practice. To facilitate the application of surface-based flow visualizations, we proposed two

diametral approaches: interactive specification and automatic selection methods. In contrast to integration-based methods, our interactive approach offers a new modality in that users are able to directly interact with complete flow-aligned surfaces. The proposed deformation-based method uses Poisson-type optimizations of generalized flux-based criteria that do not only yield flow-tangential but also flow-orthogonal surfaces. We showed that the latter are particularly well suited for seed structure definition. Illustrative surface-based visualizations are efficiently computed in the same framework. Still, time-consuming manual user interaction is often not desired or even impossible in automated flow analysis pipelines, and automatic surface selection methods are required. To solve this issue, we presented an automatic selection method for flow-relevant stream surfaces, which requires no user-interaction. Contrary to existing limited local approaches, our method is first to apply a global intrinsic surface-based quality criterion. Our method can be interpreted as selecting geometrically fair stream surfaces, whereas in geometry processing surfaces are optimized to be fair. For a number of different data sets we showed that our automatically selected results are very similar to relevant stream surfaces selected by flow visualization experts.

We demonstrated that our surface-based flow visualization approaches provide convincing results for a variety of data sets. Still, so far most of our proposed methods are designed to only handle steady $3d$ flows. It is an interesting direction for further research to extend the proposed concepts to more general flow types that might be more common in practice. Certainly, *unsteady* $3d$ flows are of major practical importance. Here, for automatic surface selection the major challenge is to shown that our stream surface quality measures are also applicable or extendible to this case. With acceptance of higher computational costs, our selection algorithm can certainly be applied to unsteady flows, although further performance enhancements are preferable. Recently, also *uncertain* flows received increased attention. This type of flow requires novel shape extraction approaches because integration is hardly possible or very expensive. We believe that our deformation-based surface extraction approach can be extended in this direction in a straightforward way by using weighted least-squares gradient fitting that respects local vector field uncertainties.

In summary, the approaches proposed in this thesis can be classified into the following important general aspects:

VECTOR FIELD-BASED METHODS.
We used kinematic vector fields as a general tool to solve a number of geometry processing problems in a continuous way. Examples include continuous deformations of explicit shapes (Chapters 3, 4, and 6) and implicit isosurfaces (Chapter 5). Certainly, every flow visualization technique is also vector field-based (Chapters 9 and 10).

GENERALIZATIONS.
We showed that the generalization of certain problem properties provides interesting new solutions and unified interpretations. Most notable, this includes a complete description of all possible vector field-based deformation types (Chapter 3), the computation of different types of characteristic surfaces by flux-optimization criteria (Chapter 9), as well as a unified enhancement of energy-based geometry processing approaches by energy regularization (Chapter 7).

GLOBAL VARIATIONAL OPTIMIZATION.
A variety of our solutions are modeled by global variational energy minimization (Chapter 7). Examples include vector field optimization in each integration step (Chapters 3 and 4) and iterative deformation optimization (Chapter 9).

DISCRETIZATIONS OF CONTINUOUS ENERGIES.
Our proposed energies are generally defined as integrated quantities on continuous domains. Suitable discretizations of these measures yield tessellation-insensitive results (Chapters 3, 4, 7, and 9) as well as computationally feasible search spaces (Chapter 10).

SMOOTHNESS.
We identify smoothness of energies (Chapters 3, 4, 7, and 9) and vector fields (Chapters 5 and 6) to be critical for a number of our approaches.

GPU-ACCELERATION.
We used GPU-based parallel computations whenever necessary and applicable (Chapters 3, 4, 5, and 6).

Part III

APPENDIX

A

DIFFERENTIAL OPERATORS OF LINEAR FUNCTIONS ON SIMPLICES

A d-dimensional simplex Ω_d ($d > 0$) is the convex hull of its $d+1$ vertices i in $\mathbb{R}^d$. In this work, we primarily study two- and three-dimensional simplices, i.e., triangles and tetraheda. In this section, we present differential operators to perform differentiation and integration of linear functions on simplices of any dimension. These operators have special cases for the low-dimensional simplices considered in this work. A linear function $\mathbf{f}_d \colon \Omega_d \to \mathbb{R}^m$ on Ω_d ($d > 0$) with m-dimensional range ($m > 0$) can be defined by the linear interpolation of coefficients $\mathbf{u}_i \in \mathbb{R}^m$ at its vertices with coordinates $\mathbf{x}_i \in \mathbb{R}^d$ as

$$\mathbf{f}_d(\mathbf{x}) = \sum_{i=0}^{d} \phi_i(\mathbf{x})\, \mathbf{u}_i \,, \tag{78}$$

where the basis functions (or *linear barycentric coordinates*) $\phi_i \colon \Omega_d \to \mathbb{R}$ are the unique linear functions on Ω_d that fulfill the Lagrange property $\phi_i(\mathbf{x}_j) = \delta^{ij}$ and the partition of unity property $\phi_d(\mathbf{x}) = 1 - \sum_{i=0}^{d-1} \phi_i(\mathbf{x})$. The union of multiple simplices to (pure) simplical complexes yields piecewise linear functions, e.g., the coordinate functions of triangular meshes. All operations on simplices presented in this section can be performed element-wise on simplical complexes. We continue to present differential operators to perform differentiation and integration of linear functions on simplices.

A.1 GRADIENT OPERATORS OF LINEAR FUNCTIONS ON SIMPLICES

The spatial gradient of the linear function is constant on each simplex. W.l.o.g. in this section we concentrate on scalar functions ($m = 1$), which generalize to higher-dimensional functions in a simple component-wise way. The function gradient on a simplex Ω_d is given by

$$\nabla f_d = \sum_{i=0}^{d} \nabla \phi_i \, u_i \,,$$

i.e., as the linear combination of (constant) basis function gradients $\nabla\phi_i \in \mathbb{R}^d$. Expressions for basis function gradients can be obtained explicitly for each simplex type (see, e.g., Botsch et al. [BKP*10, Section 3.3.3] for a derivation of basis function gradients on triangles).

Alternatively, we provide defining conditions that directly extend to any simplex dimensionality: we define basis function gradients $\nabla\phi_i$ on d-dimensional simplices generically as the solution of the linear system

$$\underbrace{\begin{bmatrix}\nabla\phi_0 & \cdots & \nabla\phi_d\end{bmatrix}}_{\mathbf{G}_d} = \begin{bmatrix}\mathbf{x}_1 - \mathbf{x}_0 & \cdots & \mathbf{x}_d - \mathbf{x}_0\end{bmatrix}^{-\mathrm{T}} \begin{bmatrix}-\mathbf{1}_d & \mathbf{I}_d\end{bmatrix}, \tag{79}$$

with a d-dimensional vector of ones $\mathbf{1}_d$ and identity matrix $\mathbf{I}_d$ on the right hand side. $\mathbf{G}_d$ is the $d \times (d+1)$ linear *gradient operator* that is of fundamental importance as it is used in many approaches throughout this work. It does only depend on the geometry of the simplex. The gradient operator performs the linear combination of basis function gradients for the function gradient:

$$\nabla f_d = \mathbf{G}_d \left(u_0, \ldots, u_d\right)^{\mathrm{T}}.$$

For $m > 1$, the *same* operator $\mathbf{G}_d$ is used to compute the m component-wise function gradients $\nabla\mathbf{f}_d^j$, $j = 1, \ldots, m$:

$$\begin{bmatrix}\nabla\mathbf{f}_d^1 & \cdots & \nabla\mathbf{f}_d^m\end{bmatrix} = \mathbf{G}_d \begin{bmatrix}\mathbf{u}_0 & \cdots & \mathbf{u}_d\end{bmatrix}^{\mathrm{T}}.$$

The system (79) for $\mathbf{G}_d$ can be derived by using a close relation of the gradient operator to deformation gradients. Specifically, let a linear deformation of a simplex be given by $\mathbf{u}(\mathbf{x}) = \mathbf{D}_d\,\mathbf{x} + \mathbf{t}_d$, $\mathbf{x} \in \Omega_d$ that maps vertex coordinates to new positions: $\mathbf{u}(\mathbf{x}_i) = \mathbf{x}_i'$. The constant translational part of the deformation $\mathbf{t}_d$ can be removed by performing pairwise subtraction giving $\mathbf{D}_d\left(\mathbf{x}_i - \mathbf{x}_j\right) = \mathbf{x}_i' - \mathbf{x}_j'$. This conditions corresponds to a property of the *deformation gradient* $\mathbf{D}_d$, which maps edge vectors of the simplex to edge vectors of the deformed simplex. As $\mathbf{D}_d$ is the gradient of the deformation $\mathbf{u}$, it can be computed component-wise by $\mathbf{G}_d$: $\mathbf{D}_d = \mathbf{G}_d \left[\mathbf{x}_0' \cdots \mathbf{x}_d'\right]^{\mathrm{T}}$. It is straightforward to obtain the system (79) for any simplex dimension d by combining these two properties using d pairwise subtractions.

For piecewise linear functions on (pure) simplical complexes defined by a set of $|\mathcal{T}|$ simplices and $|\mathcal{V}|$ vertices, all coefficients u_i can be stacked in a vector $\mathbf{u}$. With appropriate permutations a global $d\,|\mathcal{T}| \times |\mathcal{V}|$ gradient operator matrix $\mathbf{G}$ is assembled from the basis function gradients $\mathbf{G}_d$ on each simplex, such that $\mathbf{G}\,\mathbf{u}$ is the vector of stacked constant gradients on each simplex.

If simplices are embedded into spaces of higher dimension, e.g., triangle meshes representing surfaces in $3d$ with triangle normal $\mathbf{n}$, the basis function gradients can be constrained to the linear subspace spanned by the simplex:

$$\mathbf{G}_d = \begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_0 & \cdots & \mathbf{x}_d - \mathbf{x}_0 & \mathbf{n} \end{bmatrix}^{-\mathrm{T}} \begin{bmatrix} -\mathbf{1}_d & \mathbf{I}_d \\ 0 & \mathbf{0}_d{}^{\mathrm{T}} \end{bmatrix} .$$

Here, the gradient operator $\mathbf{G}_d$ is $(d+1) \times (d+1)$ with a kernel spanned by $\mathbf{n}$. Additional kernel dimensions might be added for, e.g., gradient computations on polylines embedded in $3d$.

The following important special case are derived from the generic description:

LINE SEGMENT ($d = 1$). Line segment in $1d$:

$$(x_1 - x_0)\ \mathbf{G}_1 = \begin{pmatrix} -1 & 1 \end{pmatrix}$$

Line segment embedded in $2d$ with normal $\mathbf{n}$:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^{\mathrm{T}} \\ \mathbf{n}^{\mathrm{T}} \end{bmatrix} \mathbf{G}_1 = \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix}$$

Line segment embedded in $3d$ with normal $\mathbf{n}$ and binormal $\mathbf{m}$:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^{\mathrm{T}} \\ \mathbf{n}^{\mathrm{T}} \\ \mathbf{m}^{\mathrm{T}} \end{bmatrix} \mathbf{G}_1 = \begin{pmatrix} -1 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Note that these systems always yield a line-based gradient operator of the form

$$\mathbf{G}_1 = \begin{bmatrix} \frac{\mathbf{x}_0 - \mathbf{x}_1}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2} & \frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2} \end{bmatrix}$$

in any dimension.

TRIANGLE ($d = 2$). Triangle in $2d$:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^{\mathrm{T}} \\ (\mathbf{x}_2 - \mathbf{x}_0)^{\mathrm{T}} \end{bmatrix} \mathbf{G}_2 = \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Triangle embedded in $3d$ with normal $\mathbf{n}$:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^{\mathrm{T}} \\ (\mathbf{x}_2 - \mathbf{x}_0)^{\mathrm{T}} \\ \mathbf{n}^{\mathrm{T}} \end{bmatrix} \mathbf{G}_2 = \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

(an equivalent system was derived by Botsch et al. [BSPG06] using trigonometric arguments, which do not directly extend to higher simplex dimensions).

TETRAHEDRON ($d = 3$). Tetrahedron in $3d$:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^\mathrm{T} \\ (\mathbf{x}_2 - \mathbf{x}_0)^\mathrm{T} \\ (\mathbf{x}_3 - \mathbf{x}_0)^\mathrm{T} \end{bmatrix} \mathbf{G}_3 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$$

A.2 INTEGRATION OPERATORS OF LINEAR FUNCTIONS ON SIMPLICES

In the context of least-squares optimization the integration of the squared norm of linear functions

$$\int_{\Omega_d} \|\mathbf{f}_d(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x}$$

on simplices Ω_d is of special interest. This integration can be described by a quadratic form in the function coefficients $\mathbf{u}_i$, see (78). We define the standard L^2-inner product of two functions over Ω_d as

$$\langle \phi_i, \phi_j \rangle_{\Omega_d} = \int_{\Omega_d} \phi_i(\mathbf{x}) \, \phi_j(\mathbf{x}) \, \mathrm{d}\mathbf{x} \, .$$

As $\|\mathbf{f}_d\|^2 = {\mathbf{f}_d}^\mathrm{T} \mathbf{f}_d = \sum_{i,j=0}^{d} \phi_i \, \phi_j \, {\mathbf{u}_i}^\mathrm{T} \mathbf{u}_j$ holds point-wise, due to linearity we have that the integral of squared norms is given by

$$\int_{\Omega_d} {\mathbf{f}_d}^\mathrm{T} \mathbf{f}_d \, \mathrm{d}\mathbf{x} = \sum_{i,j=0}^{d} \langle \phi_i, \phi_j \rangle \; {\mathbf{u}_i}^\mathrm{T} \mathbf{u}_j \, .$$

This integral can be written as the component-wise quadratic form $\mathbf{N}_{d,m}$

$$\int_{\Omega_d} {\mathbf{f}_d}^\mathrm{T} \mathbf{f}_d \, \mathrm{d}\mathbf{x} = \mathbf{u}^\mathrm{T} \underbrace{\mathbf{P}^\mathrm{T} (\mathbf{H}_d \otimes \mathbf{I}_m) \, \mathbf{P}}_{\mathbf{N}_{d,m}} \mathbf{u} = \|\mathbf{u}\|^2_{\mathbf{N}_{d,m}}$$

where $\mathbf{H}_d$ is a $(d+1) \times (d+1)$ matrix with $H_{d,ij} = \langle \phi_i, \phi_j \rangle$, $\mathbf{u} = \left({\mathbf{u}_0}^\mathrm{T}, \ldots, {\mathbf{u}_d}^\mathrm{T}\right)^\mathrm{T}$ is the $m\,(d+1)$-dimensional vector of "stacked" function coefficients, and $\mathbf{P}$ is the $m\,(d+1) \times m\,(d+1)$ permutation matrix that reorders the coefficients of $\mathbf{u}$ component-wise, i.e., the permutation is described by the mapping $\pi_i = \lfloor \frac{i}{d+1} \rfloor + (i \mod d+1)\, m$ for $i = 0, \ldots, m\,(d+1)$. We denote by $\mathbf{H}_d \otimes \mathbf{I}_m$ the Kronecker product of $\mathbf{H}_d$ with the m-dimensional identity matrix $\mathbf{I}_m$ that results in the component-wise application of $\mathbf{H}_d$. The matrix $\mathbf{N}_{d,m}$ is symmetric positive-definite and does only depend on the geometry of Ω_d and not on the coefficients $\mathbf{u}_i$. From hereon, we will omit the dimensional subscripts d and m when the context is unambiguous.

To determine the coefficients of $\mathbf{H}$ in practice, it is most convenient to first determine a constant quadratic form $\mathbf{N}^0$ for parametric unit simplices Ω^0 that are

aligned to unit axis. Then the integral on a concrete simplex can be reduced to the integration on unit simplices with an appropriate subsequent transformation.

On a d-dimensional unit simplex Ω^0 that has axis aligned edges and $\mathbf{x}_0$ coinciding with the origin, the inner products $\langle \phi_i, \phi_j \rangle$ have the particularly simple form

$$\begin{aligned}\langle \phi_i, \phi_j \rangle_d^0 &= \int_{\Omega^0} \phi_i(\mathbf{y})\, \phi_j(\mathbf{y})\, \mathrm{d}\mathbf{y} \\ &= \int_0^1 \int_0^{1-\alpha_1} \dots \int_0^{1-\sum_{k=1}^{d-1} \alpha_k} \alpha_i\, \alpha_j\, \mathrm{d}\alpha_0\, \mathrm{d}\alpha_1 \dots \mathrm{d}\alpha_{d-1}\end{aligned}$$

with $\alpha_n = 1 - \sum_{k=0}^{d-1} \alpha_k$. Each α_i corresponds to an evaluated basis function ϕ_i along the unit simplex.

For instance, on a triangular simplex ($d = 2$), the inner products evaluate to

$$\langle \phi_i, \phi_j \rangle_2^0 = \int_0^1 \int_0^{1-\alpha_1} \alpha_i\, \alpha_j\, \mathrm{d}\alpha_0\, \mathrm{d}\alpha_1 = \begin{cases} \frac{1}{12} & i = j \\ \frac{1}{24} & i \neq j \end{cases} .$$

In general, $\langle \phi_i, \phi_j \rangle_d^0 = \frac{1}{2} \langle \phi_i, \phi_i \rangle_d^0$ for $i \neq j$, and for any d the quadratic integrals $\langle \phi_i, \phi_i \rangle_d^0$ evaluate to [1]

$$\langle \phi_i, \phi_i \rangle_d^0 = \frac{1}{3} \prod_{i=1}^{d-1} \frac{1}{i+3} .$$

For the first three unit simplices the matrices $\mathbf{H}_d^0$ are therefore given by

$$\mathbf{H}_1^0 = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}, \quad \mathbf{H}_2^0 = \begin{pmatrix} \frac{1}{12} & \frac{1}{24} & \frac{1}{24} \\ \frac{1}{24} & \frac{1}{12} & \frac{1}{24} \\ \frac{1}{24} & \frac{1}{24} & \frac{1}{12} \end{pmatrix}, \quad \text{and} \quad \mathbf{H}_3^0 = \begin{pmatrix} \frac{1}{60} & \frac{1}{120} & \frac{1}{120} & \frac{1}{120} \\ \frac{1}{120} & \frac{1}{60} & \frac{1}{120} & \frac{1}{120} \\ \frac{1}{120} & \frac{1}{120} & \frac{1}{60} & \frac{1}{120} \\ \frac{1}{120} & \frac{1}{120} & \frac{1}{120} & \frac{1}{60} \end{pmatrix} .$$

For general, non-degenerate simplices Ω, let $\Phi\colon \Omega^0 \to \Omega$ be the differentiable affine map from the unit simplex to the concrete simplex. The transformation of the basis function integrals by Φ is determined by multi-dimensional integration by substitution:

$$\langle \phi_i, \phi_j \rangle = \int_{\Omega = \Phi(\Omega^0)} \phi_i(\mathbf{x})\, \phi_j(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_{\Omega^0} \phi_i(\mathbf{y})\, \phi_j(\mathbf{y})\, \left|\det(\nabla \Phi(\mathbf{y}))\right|\, \mathrm{d}\mathbf{y} .$$

As the gradient of an affine map Φ is constant, it follows that

$$\langle \phi_i, \phi_j \rangle = \left|\det(\nabla \Phi)\right|\, \langle \phi_i, \phi_j \rangle_d^0 .$$

1 Interestingly, $\left(\langle \phi_i, \phi_i \rangle_d^0\right)^{-1}$ is equal to the number of even permutations of n letters (see, e.g., http://oeis.org/A001710).

Using multi-dimensional integration by substitution once again, it can be shown that the determinant of the gradient $\nabla\Phi$ of this affine map is given by

$$\det(\nabla\Phi) = \frac{\operatorname{vol}(\Omega)}{\operatorname{vol}(\Omega^0)} = d!\,\operatorname{vol}(\Omega)\,,$$

with the unsigned volume $\operatorname{vol}(\Omega) = \int_\Omega 1\,\mathrm{d}\mathbf{x}$, and because the unit simplex volume is given by $\operatorname{vol}(\Omega^0) = \frac{1}{d!}$.

The final quadratic form $\mathbf{N}$ for the integration of the squared norm of a m-dimensional linear function on any simplex Ω of dimension d can therefore be written as

$$\begin{aligned}\mathbf{N} &= d!\,\operatorname{vol}(\Omega)\;\mathbf{P}^{\mathrm{T}}\left(\mathbf{H}_d^0 \otimes \mathbf{I}_m\right)\;\mathbf{P} \\ &= d!\,\operatorname{vol}(\Omega)\;\mathbf{N}^0\,.\end{aligned}$$

The integral operator of the squared norms of piecewise linear functions on a simplical complex is obtained by assembling the quadratic forms $\mathbf{N}$ of each simplex in an element-wise way into an operator $\mathbf{M}$ on the whole simplical complex.

For discretized surfaces ($n = 2$) and volumes ($n = 3$) this L^2-product operator is known as the *mass matrix* $\mathbf{M}$ in the Finite Element literature [Bra07]. Our formulation extends to any simplex dimension. Note that, to facilitate computations, the FEM mass matrix is commonly approximated by a uniform *lumped* diagonal mass matrix where the diagonal elements are $\frac{1}{d+1}$ of the combined volumes at each vertex. The lumped mass matrix is also frequently used by the geometric modeling community, e.g., for spectral-based shape analysis and deformations [ZvKD10, HSvTP10, HSTP11] and FEM-based geometric modeling [JTSZ10]. A more geometry-aware non-uniform discretization of a lumped diagonal mass matrix uses the Voronoi areas at each vertex instead of the uniform neighbor contribution [MDSB03, JBPS11]. In this work, we indicate which integration operator is applied in the context of the corresponding approach.

B

METRIC DEFORMATION ERRORS

A non-rigid single-step (i.e., not time-dependent) deformation $\mathbf{f}(\mathbf{x})\colon \mathcal{D}_d^0 \to \mathbb{R}^d$ (see Section 3.2.1) induces distortions of the initial d-dimensional shape $\mathcal{D}_d^0$. Distortions can be measured by different types of deformation errors that are characterized by the singular values σ_i, $i = 1, \ldots, d$ of the *deformation gradient* $\mathbf{D} = \nabla\mathbf{f}$. Let $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathrm{T}$ be the singular value decomposition of $\mathbf{D}$ with $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$. We assume that $\det(\mathbf{D}) > 0$, i.e., $\mathbf{f}$ is orientation-preserving. Hence, $\mathbf{U}$ and $\mathbf{V}$ represent rotations and no reflections, i.e., $\det(\mathbf{U}) = \det(\mathbf{V}) = 1$. We measure the L^2 deviation from prototype deformation gradients $\mathbf{M} = \mathbf{U}\mathbf{K}\mathbf{V}^\mathrm{T}$ that are factored using the same orthonormal matrices $\mathbf{U}$, $\mathbf{V}$, but by using different target singular values k_i: $\mathbf{K} = \mathrm{diag}(k_1, \ldots, k_d)$. This factorization is sound because it is known that the closest deformation gradient $\mathbf{M}$ to $\mathbf{D}$ of both isometric and conformal maps $\mathbf{f}$ can be factored this way (see, e.g., [LZX*08]).

Then a generic squared local deformation error that depends on the prototype singular values σ_i and is parameterized by the prototype singular values k_i is given by

$$\begin{aligned} e(\mathbf{x}) &= \|\mathbf{D} - \mathbf{M}(\mathbf{D})\|_F^2 = \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathrm{T} - \mathbf{U}\mathbf{K}\mathbf{V}^\mathrm{T}\|_F^2 \\ &= \mathrm{Tr}\Big(\mathbf{V}\mathbf{\Sigma}\mathbf{U}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathrm{T} - 2\,\mathbf{V}\mathbf{\Sigma}\mathbf{U}\mathbf{U}\mathbf{K}\mathbf{V}^\mathrm{T} + \mathbf{V}\mathbf{K}\mathbf{U}\mathbf{U}\mathbf{K}\mathbf{V}^\mathrm{T}\Big) \\ &= \mathrm{Tr}\Big(\mathbf{V}\left(\mathbf{\Sigma}^2 - 2\,\mathbf{\Sigma}\mathbf{K} + \mathbf{K}^2\right)\mathbf{V}^\mathrm{T}\Big) \\ &= \mathrm{Tr}\left(\mathbf{\Sigma}^2 - 2\,\mathbf{\Sigma}\mathbf{K} + \mathbf{K}^2\right) \\ &= \sum_i (\sigma_i - k_i)^2 \ . \end{aligned} \tag{80}$$

The total normalized deformation error of the whole map $\mathbf{f}$ is then given by

$$E = \frac{1}{\mathrm{vol}\left(\mathcal{D}_d^0\right)} \int_{\mathcal{D}_d^0} e(\mathbf{x})\,\mathrm{d}\mathbf{x} \ .$$

We use a normalization by the shape volume $\mathrm{vol}\left(\mathcal{D}_d^0\right)$ to obtain measures that are comparable between different shapes. For discretizations of the shape $\mathcal{D}_d^0$ by piecewise linear (pure) simplical complexes with n elements Ω_k, $k = 1, \ldots, n$,

the deformation gradients are constant on each simplex. Hence, the normalized integrated error simplifies to the weighted sum

$$E = \frac{1}{\operatorname{vol}(\mathcal{D}_d^0)} \sum_k \operatorname{vol}(\Omega_k)\ e(\Omega_k),$$

where $e(\Omega_k)$ is the constant error in every point of Ω_k, and $\operatorname{vol}(\Omega_k)$ is the simplex volume, e.g., triangle area or tetrahedron volume. This generic deformation error has the following special cases:

ISOMETRIC ERROR. The closest isometric deformation gradient $\mathbf{M}$ to $\mathbf{D}$ is given by $k_i = 1$, i.e., $\mathbf{M}$ has no scaling component and is a pure rotation. Therefore, the d-dimensional local isometric deformation error is given by

$$e_{isom} = \sum_i (\sigma_i - 1)^2 \ . \tag{81}$$

e_{isom} vanishes in any dimension for deformations that are (locally) rigid. This error has the important special cases

$$e_{isom}^{2d} = (\sigma_1 - 1)^2 + (\sigma_2 - 1)^2$$

and

$$e_{isom}^{3d} = (\sigma_1 - 1)^2 + (\sigma_2 - 1)^2 + (\sigma_3 - 1)^2$$

that define the *total isometric deformation errors* E_{isom}^{2d} and E_{isom}^{3d}, respectively.

CONFORMAL ERROR. The closest conformal deformation gradient $\mathbf{M}$ to $\mathbf{D}$ is given by $k_i = \frac{1}{d}\sum_j \sigma_j$, i.e., $\mathbf{M}$ only scales isotropically and is a similarity transformation. The d-dimensional local conformal deformation error can then be expressed in the following ways:

$$\begin{aligned} e_{conf} &= \sum_i \left(\sigma_i - \sum_j \frac{\sigma_j}{d} \right)^2 = \frac{1}{d} \sum_{(i,j)\in\mathcal{P}_d} (\sigma_i - \sigma_j)^2 \\ &= \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_d \end{pmatrix}^{\mathrm{T}} \underbrace{\begin{pmatrix} \frac{d-1}{d} & \frac{-1}{d} & \dots & \frac{-1}{d} \\ \frac{-1}{d} & \frac{d-1}{d} & \dots & \frac{-1}{d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-1}{d} & \frac{-1}{d} & \dots & \frac{d-1}{d} \end{pmatrix}}_{\mathbf{C}_d} \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_d \end{pmatrix} . \end{aligned} \tag{82}$$

Here, $\mathcal{P}_d$ is the set of tuples of pairwise permutations of $\{1, \dots, d\}$, e.g., $\mathcal{P}_3 = \{(1,2), (2,3), (3,1)\}$. Moreover, and in contrast to the isometric error, the conformal error can completely be described by the positive-semi-definite quadratic

forms $\mathbf{C}_d$. $\mathbf{C}_d$ has a one-dimensional kernel that is spanned by the constant functions with the basis $\mathbf{1}_d$. Hence, e_{conf} vanishes in any dimension for deformations that (locally) scale uniformly. Important special cases are then given by

$$e_{conf}^{2d} = \frac{1}{2}\left(\sigma_1 - \sigma_2\right)^2$$

and

$$e_{conf}^{3d} = \frac{1}{3}\left(\left(\sigma_1 - \sigma_2\right)^2 + \left(\sigma_2 - \sigma_3\right)^2 + \left(\sigma_3 - \sigma_1\right)^2\right)$$

that define the *total conformal deformation errors* E_{conf}^{2d} and E_{conf}^{3d}, respectively.

AUTHALIC ERROR. The authalic error that measures local volume variation cannot be written in the form of (80), because the singular value decomposition of the closest deformation gradient $\mathbf{M}$ to $\mathbf{D}$ that is authalic must not have the same orthonormal factors $\mathbf{U}$ and $\mathbf{V}$ of $\mathbf{D}$. Hence, we define the local squared authalic deformation error directly using the determinant of $\mathbf{D}$ by

$$e_{auth} = \left(\prod_i \sigma_i - 1\right)^2,$$

from which the *total authalic deformation errors* E_{auth}^{2d} and E_{auth}^{3d} follow.

Note that in the literature a number of alternative variants of these errors were introduced. For instance, Liu et al. [LZX*08] measure the same errors that we derived but restrict their analysis to the $2d$ case only. Also for $2d$ only, Hormann and Greiner [HG99] use the isometric measure $e_{isom}^{2d} = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}$, which is derived from the Frobenius norm-based condition number of $\mathbf{D}$. Their error is called *most isometric* and measures errors symmetrically around the origin $\sigma_1 = \sigma_2 = 1$. Interestingly, Solomon et al. [SBBG11a] use the most isometric error to measure conformality, i.e., $e_{conf}^{2d} = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}$. Additionally, they measure a symmetric version of area distortion $e_{auth}^{2d} = \sigma_1\sigma_2 + \frac{1}{\sigma_1\sigma_2}$. Lipman [Lip12] defines conformal distortion in $2d$ as $e_{conf}^{2d} = \frac{\sigma_1}{\sigma_2}$ with $\sigma_1 > \sigma_2$.

This variety of error measures indicates that there is no consensus on a single error definition, and the different definitions lead to various related error expressions. Throughout this work, we prefer the L^2 deviation of deformation gradients to closest prototype deformation gradients as described above. Although the absolute error values are not symmetric w.r.t. the origin $\sigma_i = 1$, we use these errors because they are precisely the terms that are minimized by a number of single-step deformation approaches. For example, as-rigid-as-possible (ARAP) deformations and least-squares conformal maps (LSCM) (see Chapter 7) *exactly* minimize the energies described by our error definitions of E_{isom} and E_{conf}.

ACRONYMS

AKVF Approximate Killing vector field

AMVF Approximate metric vector field

ACVF Approximate conformal vector field

AAVF Approximate authalic vector field

ASSVF Approximate skew-symmetric vector field

AKAP As-Killing-as-possible deformation

ARAP As-rigid-as-possible deformation

ASAP As-similar-as-possible deformation

LSCM Least-squares conformal map

BD-MAP Bounded distortion map

APAP As-perpendicular-as-possible surface

LIC Line integral convolution

ODE Ordinary differential equation

SVD Singular value decomposition

BIBLIOGRAPHY

[ACL00] Alexa M., Cohen-Or D., Levin D.: As-rigid-as-possible shape interpolation. In *Proc. SIGGRAPH* (2000), pp. 157–164. (Cited on page 13.)

[ACSTD07] Alliez P., Cohen-Steiner D., Tong Y., Desbrun M.: Voronoi-based variational reconstruction of unoriented point sets. In *Proc. SGP* (2007), pp. 39–48. (Cited on page 107.)

[ACWK04] Angelidis A., Cani M.-P., Wyvill G., King S.: Swirling-sweepers: Constant volume modeling. In *Proc. Pacific Graphics* (2004), pp. 10–15. (Cited on page 13.)

[AMR88] Abraham R., Marsden J., Ratiu T.: *Manifolds, Tensor Analysis, and Applications*. Springer, 1988. (Cited on page 142.)

[AS92] Abraham, Shaw: *Dynamics, the geometry of behavior*, 2nd ed. Addison-Wesley, 1992. (Cited on page 165.)

[AS05] Alexandrov O., Santosa F.: A topology-preserving level set method for shape optimization. *J. Comput. Phys. 204*, 1 (2005), 121–130. (Cited on page 16.)

[AW11] Alexa M., Wardetzky M.: Discrete laplacians on general polygonal meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 102:1–102:10. (Cited on page 106.)

[AWC04] Angelidis A., Wyvill G., Cani M.-P.: Sweepers: Swept user-defined tools for modeling by deformation. In *Proc. SMI* (2004), pp. 63–73. (Cited on page 13.)

[Bar84] Barr A. H.: Global and local deformations of solid primitives. In *Proc. SIGGRAPH* (1984), pp. 21–30. (Cited on page 13.)

[BAV*10] Bergou M., Audoly B., Vouga E., Wardetzky M., Grinspun E.: Discrete viscous threads. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 116:1–116:10. (Cited on page 14.)

[BBK05a] Botsch M., Bommes D., Kobbelt L.: Efficient linear system solvers for mesh processing. In *Proc. Mathematics of Surfaces* (2005), LNCS, pp. 62–83. (Cited on pages 37, 60, and 156.)

[BBK05b] Bronstein A. M., Bronstein M. M., Kimmel R.: Isometric embedding of facial surfaces into S^3. In *Proc. Scale-Space* (2005), pp. 622–631. (Cited on page 9.)

[BBSG09] Ben-Chen M., Butscher A., Solomon J., Guibas L.: On discrete killing vector fields and patterns on surfaces. *Comput. Graph. Forum (Proc. SGP) 29*, 5 (2009), 1701–1711. (Cited on page 15.)

[BDK98] Barequet G., Duncan C., Kumar S.: RSVP: a geometric toolkit for controlled repair of solid models. *IEEE TVCG 4*, 2 (1998), 162 –177. (Cited on page 86.)

[BDS*12] Bouaziz S., Deuss M., Schwartzburg Y., Weise T., Pauly M.: Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum (Proc. SGP) 31*, 5 (2012), 1657–1667. (Cited on page 13.)

[BFTW09] Bürger K., Ferstl F., Theisel H., Westermann R.: Interactive streak surface visualization on the gpu. *IEEE TVCG (Proc. Vis) 15*, 6 (2009), 1259–1266. (Cited on pages 137, 153, and 179.)

[BK03] Botsch M., Kobbelt L.: Multiresolution surface representation based on displacement volumes. *Comput. Graph. Forum (Proc. Eurographics) 22*, 3 (2003), 483–491. (Cited on page 87.)

[BK04] Botsch M., Kobbelt L.: An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (2004), 630–634. (Cited on page 12.)

[BK05a] Bischoff S., Kobbelt L.: Structure preserving CAD model repair. *Comput. Graph. Forum (Proc. Eurographics) 24*, 3 (2005), 527–536. (Cited on page 87.)

[BK05b] Botsch M., Kobbelt L.: Real-time shape editing using radial basis functions. *Comput. Graph. Forum (Proc. Eurographics) 24*, 3 (2005), 611–622. (Cited on pages 13, 87, and 94.)

[BKP*10] Botsch M., Kobbelt L., Pauly M., Alliez P., Levy B.: *Polygon Mesh Processing*. AK Peters, 2010. (Cited on pages 1, 110, 120, 168, 170, and 192.)

[BMF03] Bridson R., Marino S., Fedkiw R.: Simulation of clothing with folds and wrinkles. In *Proc. SCA* (2003), pp. 28–36. (Cited on page 14.)

[BMWG07] Bergou M., Mathur S., Wardetzky M., Grinspun E.: Tracks: toward directable thin shells. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 50:1–50:10. (Cited on page 14.)

[BO05] Boissonnat J.-D., Oudot S.: Provably good sampling and meshing of surfaces. *Graph. Models 67*, 5 (2005), 405–451. (Cited on pages 78 and 80.)

[Boh93] Bohn J. H.: *Automatic CAD-model repair*. PhD thesis, Rensselaer, Polytechnic Institute, 1993. (Cited on page 86.)

[BPGK06] Botsch M., Pauly M., Gross M., Kobbelt L.: PriMo: coupled prisms for intuitive surface modeling. In *Proc. SGP* (2006), pp. 11–20. (Cited on pages 10, 13, 56, and 61.)

[BPK05] Bischoff S., Pavic D., Kobbelt L.: Automatic restoration of polygon models. *ACM Trans. Graph. 24*, 4 (2005), 1332–1352. (Cited on page 87.)

[BPWG07] Botsch M., Pauly M., Wicke M., Gross M.: Adaptive space deformations based on rigid cells. *Comput. Graph. Forum (Proc. Eurographics) 26*, 3 (2007), 339–347. (Cited on page 13.)

[Bra07] Braess D.: *Finite Elements - Theory, Fast Solvers, and Applications in Solid Mechanics*, 3rd ed. Cambridge University Press, 2007. (Cited on pages 45 and 196.)

[BS08] Botsch M., Sorkine O.: On linear variational surface deformation methods. *IEEE TVCG 14*, 1 (2008), 213–230. (Cited on pages 9, 13, 61, 62, 69, 107, 115, 143, and 219.)

[BSK*13] Barton M., Shi L., Kilian M., Wallner J., Pottmann H.: Circular arc snakes and kinematic surface generation. *Comput. Graph. Forum (Proc. Eurographics) 32*, 2 (2013), 1–10. (Cited on page 15.)

[BSPG06] Botsch M., Sumner R., Pauly M., Gross M.: Deformation transfer for detail-preserving surface editing. In *Proc. VMV* (2006), pp. 357–364. (Cited on pages 12, 61, and 193.)

[BVGP09] Baran I., Vlasic D., Grinspun E., Popović J.: Semantic deformation transfer. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 36:1–36:6. (Cited on page 14.)

[BW08] Bachthaler S., Weiskopf D.: Animation of orthogonal texture patterns for vector field visualization. *IEEE TVCG 14*, 4 (2008), 741–755. (Cited on page 138.)

[BWF*10] Born S., Wiebel A., Friedrich J., Scheuermann G., Bartz D.: Illustrative stream surfaces. *TVCG (Proc. Vis) 16*, 6 (2010), 1329–1338. (Cited on pages 138, 154, 167, 179, and 181.)

[BWG09] Ben-Chen M., Weber O., Gotsman C.: Variational harmonic maps for space deformation. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 34:1–34:11. (Cited on pages 13 and 46.)

[BWK03] Baraff D., Witkin A., Kass M.: Untangling cloth. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 862–870. (Cited on page 87.)

[BWR*08] Bergou M., Wardetzky M., Robinson S., Audoly B., Grinspun E.: Discrete elastic rods. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008), 63:1–63:12. (Cited on page 14.)

[BWSK12] Bokeloh M., Wand M., Seidel H.-P., Koltun V.: An algebraic model for parameterized shape editing. *ACM Trans. Graph. 31*, 4 (2012), 78:1–78:10. (Cited on page 14.)

[BZK09] Bommes D., Zimmer H., Kobbelt L.: Mixed-integer quadrangulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 78–87. (Cited on

page 150.)

[CA06] CANI M.-P., ANGELIDIS A.: Towards virtual clay. In *ACM SIGGRAPH Courses* (2006), pp. 67–83. (Cited on pages 16 and 83.)

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *Proc. SIGGRAPH* (2001), pp. 67–76. (Cited on pages 87, 89, 93, 94, and 100.)

[CCI*07] CHEN M., CORREA C. D., ISLAM S., JONES M. W., SHEN P.-Y., SILVER D., WALTON S. J., WILLIS P. J.: Manipulating, deforming and animating sampled object representations. *Comput. Graph. Forum 26*, 4 (2007), 824–852. (Cited on page 15.)

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35*, 3 (2008), 22:1–22:14. (Cited on pages 37, 60, and 156.)

[CH97] CAI W., HENG P.-A.: Principal stream surfaces. In *Proc. Vis* (1997), pp. 75–81. (Cited on page 164.)

[CH12] CASHMAN T. J., HORMANN K.: A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Comput. Graph. Forum (Proc. Eurographics) 31*, 2 (2012), 735–744. (Cited on pages 14 and 48.)

[CK10] CAMPEN M., KOBBELT L.: Exact and robust (self-)intersections for polygonal meshes. *Comput. Graph. Forum (Proc. Eurographics) 29*, 2 (2010), 397–406. (Cited on pages 87 and 89.)

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proc. SIGGRAPH* (1993), pp. 263–270. (Cited on page 138.)

[CLL07] CHANG J. Y., LEE K. M., LEE S. U.: Multiview normal field integration using level set methods. In *Proc. CVPR* (2007), pp. 1–8. (Cited on page 138.)

[CMT*12] COROS S., MARTIN S., THOMASZEWSKI B., SCHUMACHER C., SUMNER R., GROSS M.: Deformable objects alive! *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012), 69:1–69:9. (Cited on page 14.)

[Coh09] COHEN-OR D.: Space deformations, surface deformations and the opportunities in-between. *J. Comput. Sci. Technol. 24*, 1 (2009), 2–5. (Cited on page 13.)

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 38:1–38:6. (Cited on pages 10, 13, and 116.)

[CSBI05] CAMARRI S., SALVETTI M.-V., BUFFONI M., IOLLO A.: Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *AIMETA XVII* (2005). (Cited on pages 153 and 179.)

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat. *ACM Trans. Graph.* (2013), (to appear). (Cited on pages 152 and 156.)

[CYY*11] CHEN C.-K., YAN S., YU H., MAX N., , MA K.-L.: An illustrative visualization framework for 3d vector fields. *Comput. Graph. Forum (Proc. PG) 30*, 7 (2011), 1941–1951. (Cited on page 164.)

[Dal83] DALLMANN U.: *Topological structures of three-dimensional flow separations*. Tech. rep., German Aerospace Center (DLR), 1983. (Cited on pages 154, 165, and 181.)

[Dav67] DAVIS H.: *Introduction to vector analysis*. Allyn and Bacon, Inc., 1967. (Cited on pages 74 and 75.)

[DBD*13] DENG B., BOUAZIZ S., DEUSS M., ZHANG J., SCHWARTZBURG Y., PAULY M.: Exploring local modifications for constrained meshes. *Comput. Graph. Forum (Proc. Eurographics) 32*, 2 (2013), 11–20. (Cited on page 15.)

[DC90] DOOLEY D., COHEN M. F.: Automatic illustration of 3d geometric models: Lines. In *Proc. i3D* (1990), pp. 77–82. (Cited on page 167.)

[dC92] DO CARMO M. P.: *Riemannian Geometry*. Birkhäuser Boston, 1992. (Cited on page 12.)

[DCG98] DESBRUN M., CANI-GASCUEL M.-P.: Active implicit surface for animation. In *Proc. GI* (1998), pp. 143–150. (Cited on page 16.)

[DG95] DESBRUN M., GASCUEL M.: Animating soft substances with implicit surfaces. In *Proc. SIGGRAPH* (1995), pp. 287–290. (Cited on page 16.)

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. *Comput. Graph. Forum (Proc. Eurographics) 21*, 3 (2002), 209–218. (Cited on page 120.)

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. SIGGRAPH* (1999), pp. 317–324. (Cited on page 15.)

[DSP06] DER K. G., SUMNER R. W., POPOVIĆ J.: Inverse kinematics for reduced deformable models. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (2006), 1174–1179. (Cited on page 14.)

[DZTS08] DYKEN C., ZIEGLER G., THEOBALT C., SEIDEL H.-P.: High-speed marching cubes using histogram pyramids. *Comput. Graph. Forum 27*, 8 (2008), 2028–2039. (Cited on page 77.)

[EB08] ENGLISH E., BRIDSON R.: Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008), 66:1–66:5. (Cited on page 14.)

[Efi57] EFIMOW N.: *Flaechenverbiegungen in Großen*. Akadmie-Verlag, 1957. (German). (Cited on pages 12 and 52.)

[ELC*12] EDMUNDS M., LARAMEE R. S., CHEN G., MAX N., ZHANG E., WARE C.: Surface-based flow visualization. *Comput. Graph. 36*, 8 (2012), 974–990. (Cited on page 137.)

[ELM*12] EDMUNDS M., LARAMEE R. S., MALKI R., MASTERS I., CROFT T. N., CHEN G., ZHANG E.: Automatic stream surface seeding: A feature-centered approach. *Comput. Graph. Forum (Proc. EuroVis) 31*, 3 (2012), 1095–1104. (Cited on pages 153, 164, 165, and 179.)

[EML*12] EDMUNDS M., MCLOUGHLIN T., LARAMEE R. S., CHEN G., ZHANG E., MAX N.: Advanced, automatic stream surface seeding and filtering. In *Proc. TPCG* (2012), pp. 53–60. (Cited on page 164.)

[EPT*07] ECKSTEIN I., PONS J.-P., TONG Y., KUO C.-C. J., DESBRUN M.: Generalized surface flows for mesh processing. In *Proc. SGP* (2007), pp. 183–192. (Cited on pages 15, 54, 67, 107, and 129.)

[Eul67] EULER L.: Recherches sur la courbure des surfaces. *Opera Omnia 2*, 1 (1767), 1 – 22. (French). (Cited on page 168.)

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. *Comput. Graph. Forum 30*, 8 (2011), 2246–2257. (Cited on pages 14, 47, 70, 85, 100, and 101.)

[FH05] Floater M. S., Hormann K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*. Springer, 2005, pp. 157–186. (Cited on page 19.)

[FKY08] Feng W.-W., Kim B.-U., Yu Y.: Real-time data-driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008), 91:1–91:9. (Cited on pages 14 and 85.)

[FPRJ00] Frisken S. F., Perry R. N., Rockwood A. P., Jones T. R.: Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proc. SIGGRAPH* (2000), pp. 249–254. (Cited on page 16.)

[FYK10] Feng W.-W., Yu Y., Kim B.-U.: A deformation transformer for real-time cloth animation. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 108:1–108:9. (Cited on pages 14 and 85.)

[GB08] Gain J., Bechmann D.: A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph. 27*, 4 (2008), 1–21. (Cited on page 13.)

[GBWT11] Günther T., Bürger K., Westermann R., Theisel H.: A view-dependent and inter-frame coherent visualization of integral lines using screen contribution. In *Proc. VMV* (2011), pp. 215–222. (Cited on page 138.)

[GHDS03] Grinspun E., Hirani A., Desbrun M., Schröder P.: Discrete shells. In *Proc. SCA* (2003), pp. 62–67. (Cited on page 14.)

[GHF*07] Goldenthal R., Harmon D., Fattal R., Bercovier M., Grinspun E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 49:1–49:7. (Cited on page 14.)

[Gib97] Gibson S. F. F.: 3D chainmail: A fast algorithm for deforming volumetric objects. In *Proc. SI3D* (1997), pp. 149–154. (Cited on page 16.)

[GIHL00] Girshick A., Interrante V., Haker S., Lemoine T.: Line direction matters: an argument for the use of principal directions in 3d line drawings. In *Proc. NPAR* (2000), pp. 43–52. (Cited on page 168.)

[GKT*08] Garth C., Krishnan H., Tricoche X., Bobach T., Joy K.: Generation of accurate integral surfaces in time-dependent vector fields. *IEEE TVCG 14*, 6 (2008), 1404–1411. (Cited on pages 137, 154, 179, and 181.)

[Gol89] Goldberg E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Wesley, 1989. (Cited on page 175.)

[GRT13] Günther T., Rössl C., Theisel H.: Opacity optimization for 3d line fields. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2013), (to appear). (Cited on pages 138 and 164.)

[GSMCO09] Gal R., Sorkine O., Mitra N. J., Cohen-Or D.: iWIRES: an analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 33:1–33:10. (Cited on page 14.)

[GTS*04] Garth C., Tricoche X., Salzbrunn T., Bobach T., Scheuermann G.: Surface techniques for vortex visualization. In *Proc. VisSym* (2004), pp. 155–164. (Cited on pages 137, 179, and 181.)

[GVL96] Golub G., Van Loan C.: *Matrix Computations*. Johns Hopkins University Press, 1996. (Cited on page 96.)

[GW01] Guskov I., Wood Z. J.: Topological noise removal. In *Proc. GI* (2001), pp. 19–26. (Cited on page 87.)

[GW06] Georgii J., Westermann R.: A multigrid framework for real-time simulation of deformable bodies. *Comput. Graph. 30*, 3 (2006), 408–415. (Cited on page 16.)

[Han10] HANSEN P. C.: *Discrete Inverse Problems: Insight and Algorithms*. SIAM, 2010. (Cited on pages 106, 108, and 130.)

[HAW07] HUANG Q.-X., ADAMS B., WAND M.: Bayesian surface reconstruction via iterative scan alignment to an optimized prototype. In *Proc. SGP* (2007), pp. 213–223. (Cited on page 107.)

[HB10] HENDRIX E., BOGLÁRKA G.-T.: *Introduction to Nonlinear and Global Optimization*. Springer, 2010. (Cited on page 175.)

[HF06] HORMANN K., FLOATER M. S.: Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph. 25*, 4 (2006), 1424–1441. (Cited on page 13.)

[HG99] HORMANN K., GREINER G.: MIPS: An efficient global parametrization method. In *Proc. Curves and Surfaces* (1999), pp. 153–162. (Cited on pages 12 and 199.)

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K.: IRIS: Illustrative rendering for integral surfaces. *IEEE TVCG (Proc. Vis) 16*, 6 (2010), 1319–1328. (Cited on pages 138, 151, 154, 161, 168, 179, and 181.)

[HJW87] HIGHAM N. J., JAMES I., WIIKINSON H.: Computing real square roots of a real matrix. *Linear Algebra Appl. 88*, 1 (1987), 405–430. (Cited on page 120.)

[HKRW06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D.: *Real-time Volume Graphics*. A.K. Peters, 2006. (Cited on pages 16 and 77.)

[HL93] HOSCHEK J., LASSER D.: *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, 1993. (Cited on page 107.)

[HMT*12] HAHN F., MARTIN S., THOMASZEWSKI B., SUMNER R., COROS S., GROSS M.: Rig-space physics. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012), 72:1–72:8. (Cited on page 14.)

[HPS08] HORMANN K., POLTHIER K., SHEFFER A.: Mesh parameterization: Theory and practice. In *ACM SIGGRAPH Asia Courses* (2008). (Cited on page 12.)

[HPW*12] HUANG J., PEI W., WEN C., CHEN G., CHEN W., BAO H.: Output-coherent image-space lic for surface flow visualization. In *Proc. PacificVis* (2012), pp. 137–144. (Cited on pages 138 and 152.)

[HRWW12] HEEREN B., RUMPF M., WARDETZKY M., WIRTH B.: Time-discrete geodesics in the space of shells. *Comput. Graph. Forum (Proc. SGP) 31*, 5 (2012), 1755–1764. (Cited on pages 15 and 48.)

[HSL*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S.-H., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (2006), 1126–1134. (Cited on page 13.)

[HSTP11] HILDEBRANDT K., SCHULZ C., TYCOWICZ C. V., POLTHIER K.: Interactive surface modeling using modal analysis. *ACM Trans. Graph. 30*, 5 (2011), 119:1–119:11. (Cited on pages 13 and 196.)

[HSvTP10] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Eigenmodes of surface energies for shape analysis. In *Proc. GMP* (2010), pp. 296–314. (Cited on pages 9 and 196.)

[HSvTP12] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Interactive spacetime control of deformable objects. *ACM Trans. Graph. 31*, 4 (2012), 71:1–71:8. (Cited on page 14.)

[Hul92] HULTQUIST J. P. M.: Constructing stream surfaces in steady 3d vector fields. In *Proc. Vis* (1992), pp. 171–178. (Cited on pages 137, 149, 150, 151, and 172.)

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

416. (Cited on page 66.)

[IMH05] Igarashi T., Moscovich T., Hughes J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 1134–1141. (Cited on pages 13, 31, 46, and 116.)

[Int97] Interrante V.: Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution. In *Proc. SIGGRAPH* (1997), pp. 109–116. (Cited on page 168.)

[JBK*12] Jacobson A., Baran I., Kavan L., Popović J., Sorkine O.: Fast automatic skinning transformations. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012), 77:1–77:10. (Cited on pages 13, 14, 70, and 121.)

[JBPS11] Jacobson A., Baran I., Popović J., Sorkine O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 78:1–78:8. (Cited on pages 13 and 196.)

[JL97] Jobard B., Lefer W.: Creating evenly-spaced streamlines of arbitrary density. In *Proc. VisSci* (1997), pp. 43–56. (Cited on page 164.)

[JL00] Jobard B., Lefer W.: Unsteady flow visualization by animating evenly-spaced streamlines. *Comput. Graph. Forum (Proc. Eurographics) 19*, 3 (2000), 31–39. (Cited on page 164.)

[JMD*07] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 71:1–71:10. (Cited on page 13.)

[JS11] Jacobson A., Sorkine O.: Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 30*, 6 (2011), 165:1–165:8. (Cited on page 14.)

[JSW05] Ju T., Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 561–566. (Cited on page 13.)

[JTSZ10] Jacobson A., Tosun E., Sorkine O., Zorin D.: Mixed finite elements for variational surface modeling. *Comput. Graph. Forum (Proc. SGP) 29*, 5 (2010), 1565–1574. (Cited on page 196.)

[Ju04] Ju T.: Robust repair of polygonal models. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (2004), 888–895. (Cited on page 87.)

[Ju09] Ju T.: Fixing geometric errors on polygonal models: A survey. *J. Comput. Sci. Technol. 24*, 1 (2009), 19–29. (Cited on pages 86 and 102.)

[JWS12] Jacobson A., Weinkauf T., Sorkine O.: Smooth shape-aware functions with controlled extrema. *Comput. Graph. Forum (Proc. SGP) 31*, 5 (2012), 1577–1586. (Cited on page 13.)

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *Proc. SGP* (2006), pp. 61–70. (Cited on page 142.)

[KBL12] Khabbaz M., Bhagat S., Lakshmanan L. V. S.: Finding heavy paths in graphs: A rank join approach. *CoRR 1112*, 1117 (2012), 1–16. (Cited on page 174.)

[KCATLF06] Kin-Chung Au O., Tai C.-L., Liu L., Fu H.: Dual laplacian editing for meshes. *IEEE TVCG 12*, 3 (2006), 386–395. (Cited on page 13.)

[KCvO08] Kavan L., Collins S., Žára J., O'Sullivan C.: Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph. 27*, 4 (2008), 105:1–105:23. (Cited on page 14.)

[KCVS98] Kobbelt L., Campagna S., Vorsatz J., Seidel H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proc. SIGGRAPH* (1998),

pp. 105–114. (Cited on page 12.)

[KFC*08] KILIAN M., FLÖRY S., CHEN Z., MITRA N. J., SHEFFER A., POTTMANN H.: Curved folding. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008), 75:1–75:9. (Cited on pages 9 and 55.)

[KFG09] KARNI Z., FREEDMAN D., GOTSMAN C.: Energy-based image deformation. *Comput. Graph. Forum (Proc. SGP) 28*, 5 (2009), 1257–1268. (Cited on page 13.)

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Trans. Graph. 27*, 2 (2008), 12:1–12:13. (Cited on pages 12, 14, 85, and 90.)

[KGBS11] KAVAN L., GERSZEWSKI D., BARGTEIL A. W., SLOAN P.-P.: Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 93:1–93:10. (Cited on page 107.)

[KGJ09] KRISHNAN H., GARTH C., JOY K.: Time and streak surfaces for flow visualization in large time-varying data sets. *IEEE TVCG (Proc. Vis) 15*, 6 (2009), 1267–1274. (Cited on page 137.)

[KGV83] KIRKPATRICK S., GELATT C. D., VECCHI M. P.: Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680. (Cited on page 175.)

[KHSI04a] KIM S., HAGH-SHENAS H., INTERRANTE V.: Conveying shape with texture: experimental investigations of texture's effects on shape categorization judgments. *IEEE TVCG 10*, 4 (2004), 471 –483. (Cited on page 168.)

[KHSI04b] KIM S., HAGH-SHENAS H., INTERRANTE V.: Conveying three-dimensional shape with texture. In *Proc. APGV* (2004), pp. 119–122. (Cited on page 168.)

[Kil88] KILLING W.: Die Zusammensetzung der stetigen endlichen Transformations-Gruppen. *Mathematische Annalen 31* (1888), 252–290. (German). (Cited on page 15.)

[KJ00] KALTENBACH H.-J., JANKE G.: Direct numerical simulation of flow separation behind a swept, rearward-facing step at $Re_H = 3000$. *POF 12*, 9 (2000), 2320–2337. (Cited on page 154.)

[KKKW05] KRÜGER J., KIPFER P., KONDRATIEVA P., WESTERMANN R.: A particle system for interactive visualization of 3D flows. *IEEE TVCG 11*, 6 (2005), 744–756. (Cited on page 138.)

[KMBG08] KAUFMANN P., MARTIN S., BOTSCH M., GROSS M.: Flexible simulation of deformable models using discontinuous galerkin fem. In *Proc. SCA* (2008), pp. 105–115. (Cited on page 14.)

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 64:1–64:8. (Cited on pages 9, 11, 15, 17, 35, 46, 48, 52, 54, 67, 70, and 85.)

[KOD*05] KRAUSKOPF B., OSINGA H. M., DOEDEL E. J., HENDERSON M. E., GUCKENHEIMER J., VLADIMIRSKY A., DELLNITZ M., JUNGE O.: A survey of methods for computing (un)stable manifolds of vector fields. *IJBC 15*, 3 (2005), 763–791. (Cited on page 138.)

[Kor85] KORF R. E.: Depth-first iterative-deepening: An optimal admissible tree search. *AI 27*, 1 (1985), 97–109. (Cited on page 174.)

[KS98] KIMMEL R., SETHIAN J.: Fast marching methods on triangulated domains. In *Proc. Nat. Acad. Sci.* (1998), pp. 8341–8435. (Cited on page 151.)

[KS12] KAVAN L., SORKINE O.: Elasticity-inspired deformers for character articulation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 196:1–196:8. (Cited on page 14.)

[LCF00] Lewis J. P., Cordner M., Fong N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH* (2000), pp. 165–172. (Cited on pages 14 and 85.)

[LGSH06] Laramee R. S., Garth C., Schneider J., Hauser H.: Texture advection on stream surfaces: A novel hybrid visualization applied to CFD simulation results. In *Proc. EuroVis* (2006), pp. 155–162. (Cited on pages 138 and 152.)

[LHD*04] Laramee R. S., Hauser H., Doleisch H., Vrolijk B., Post F. H., Weiskopf D.: The state of the art in flow visualization: Dense and texture-based techniques. *Comput. Graph. Forum 23*, 2 (2004), 203–221. (Cited on pages 138, 152, and 167.)

[LHZP05] Laramee R. S., Hauser H., Zhao L., Post F. H.: Topology-based flow visualization, the state of the art. In *Proc. TIV* (2005), pp. 1–19. (Cited on page 137.)

[Lip12] Lipman Y.: Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4 (2012), 108:1–108:13. (Cited on pages 10, 13, 31, 43, 116, 126, 127, 129, and 199.)

[LLC08] Lipman Y., Levin D., Cohen-Or D.: Green coordinates. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008), 78:1–78:10. (Cited on page 13.)

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 3 (2002), 362–371. (Cited on pages 120 and 142.)

[LSLCO05] Lipman Y., Sorkine O., Levin D., Cohen-Or D.: Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 479–487. (Cited on pages 12, 61, and 85.)

[LTWH08] Li G.-S., Tricoche X., Weiskopf D., Hansen C.: Flow charts: Visualization of vector fields on arbitrary surfaces. *IEEE TVCG 14*, 5 (2008), 1067–1080. (Cited on pages 138, 152, and 160.)

[LWP10] Li H., Weise T., Pauly M.: Example-based facial rigging. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 32:1–32:6. (Cited on page 14.)

[LZX*08] Liu L., Zhang L., Xu Y., Gotsman C., Gortler S. J.: A local/global approach to mesh parameterization. *Comput. Graph. Forum (Proc. SGP) 27*, 5 (2008), 1495–1504. (Cited on pages 10, 12, 13, 46, 47, 115, 116, 121, 123, 142, 147, 197, and 199.)

[MBWB02] Museth K., Breen D. E., Whitaker R. T., Barr A. H.: Level set surface editing operators. In *Proc. SIGGRAPH* (2002), pp. 330–338. (Cited on pages 15 and 83.)

[MCHM10] Marchesin S., Chen C.-K., Ho C., Ma K.-L.: View-dependent streamlines for 3d vector fields. *IEEE TVCG (Proc. Vis) 16*, 6 (2010), 1578 –1586. (Cited on page 164.)

[MDM*02] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B.: Stable real-time deformations. In *Proc. SCA* (2002), pp. 49–54. (Cited on pages 14 and 46.)

[MDSB03] Meyer M., Desbrun M., Schröder P., Barr A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer, 2003, pp. 35–57. (Cited on page 196.)

[Min01] Minka T. P.: *Old and New Matrix Algebra Useful for Statistics*. Tech. rep., MIT Media Lab, 2001. (Cited on page 19.)

[MIW13] Mattausch O., Igarashi T., Wimmer M.: Freeform shadow boundary editing. *Comput. Graph. Forum (Proc. Eurographics)* (2013), (to appear).

(Cited on page 14.)

[MJBF02] Milliron T., Jensen R. J., Barzel R., Finkelstein A.: A framework for geometric warps and deformations. *ACM Trans. Graph. 21*, 1 (2002), 20–51. (Cited on page 13.)

[MKB*08] Martin S., Kaufmann P., Botsch M., Wicke M., Gross M.: Polyhedral finite elements using harmonic basis functions. *Comput. Graph. Forum (Proc. SGP) 27*, 5 (2008), 1521–1529. (Cited on pages 14 and 87.)

[MKB*10] Martin S., Kaufmann P., Botsch M., Grinspun E., Gross M.: Unified simulation of elastic rods, shells, and solids. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 39:1–39:10. (Cited on page 14.)

[MLP*10] McLoughlin T., Laramee R. S., Peikert R., Post F. H., Chen M.: Over two decades of integration-based, geometric flow visualization. *Comput. Graph. Forum 29*, 6 (2010), 1807–1829. (Cited on pages 136, 137, 144, 145, 151, and 164.)

[MMTD07] Mullen P., McKenzie A., Tong Y., Desbrun M.: A variational approach to eulerian geometry processing. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 66–74. (Cited on page 16.)

[MN07] Magnus J., Neudecker H.: *Matrix differential calculus with applications in statistics and econometrics*, 3rd ed. Wiley, 2007. (Cited on page 29.)

[MQW01] McDonnell K. T., Qin H., Wlodarczyk R. A.: Virtual clay: a real-time sculpting system with haptic toolkits. In *Proc. I3D* (2001), pp. 179–190. (Cited on page 16.)

[MRR*53] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E.: Equation of state calculations by fast computing machines. *JCP 21*, 6 (1953), 1087–1092. (Cited on page 175.)

[MS92] Moreton H. P., Séquin C. H.: Functional optimization for fair surface design. *Proc. SIGGRAPH 26*, 2 (1992), 167–176. (Cited on page 168.)

[MS04] Mémoli F., Sapiro G.: Comparing point clouds. In *Proc. SGP* (2004), pp. 32–40. (Cited on page 9.)

[MS11] Manson J., Schaefer S.: Hierarchical deformation of locally rigid meshes. *Comput. Graph. Forum 30*, 8 (2011), 2387–2396. (Cited on page 13.)

[MTGG11] Martin S., Thomaszewski B., Grinspun E., Gross M.: Example-based elastic materials. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 72:1–72:8. (Cited on page 14.)

[MTLT88] Magnenat-Thalmann N., Laperrière R., Thalmann D.: Joint-dependent local deformations for hand animation and object grasping. In *Proc. GI* (1988), pp. 26–33. (Cited on page 85.)

[MTPS08] Mezger J., Thomaszewski B., Pabst S., Strasser W.: Interactive physically-based shape editing. In *Proc. SPM* (2008), pp. 79–89. (Cited on page 14.)

[MWCS13] Milliez A., Wand M., Cani M.-P., Seidel H.-P.: Mutable elastic models for sculpting structured shapes. *Comput. Graph. Forum (Proc. Eurographics) 32*, 2 (2013), 21–30. (Cited on page 13.)

[MWZ*13] Mitra N. J., Wand M., Zhang H., Cohen-Or D., Bokeloh M.: Structure-aware shape processing. In *Eurographics STARs*. 2013, p. (to appear). (Cited on page 14.)

[MYC*01] Morse B. S., Yoo T. S., Chen D. T., Rheingans P., Subramanian K. R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proc. SMI* (2001). (Cited on page 87.)

[MZS*11] McAdams A., Zhu Y., Selle A., Empey M., Tamstorf R., Teran J., Sifakis E.: Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 37:1–37:12. (Cited on pages 14 and 45.)

[NMK*06] Nealen A., Mueller M., Keiser R., Boxerman E., Carlson M.: Physically-based deformable models in computer graphics. *Comput. Graph. Forum 25*, 4 (2006), 809–836. (Cited on pages 14 and 54.)

[NSACO05] Nealen A., Sorkine O., Alexa M., Cohen-Or D.: A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 1142–1147. (Cited on page 12.)

[NT03] Nooruddin F., Turk G.: Simplification and repair of polygonal models using volumetric techniques. *IEEE TVCG 9*, 2 (2003), 191 – 205. (Cited on page 87.)

[OBS05] Ohtake Y., Belyaev A., Seidel H.-P.: 3D scattered data interpolation and approximation with multilevel compactly supported RBFs. *Graph. Models 67*, 3 (2005), 150–165. (Cited on page 87.)

[OBW*08] Orzan A., Bousseau A., Winnemöller H., Barla P., Thollot J., Salesin D.: Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3 (2008), 92:1–92:8. (Cited on page 142.)

[OF01] Osher S., Fedkiw R. P.: Level set methods: An overview and some recent results. *J. Comput. Phys. 169*, 2 (2001), 463–502. (Cited on pages 15 and 72.)

[OF02] Osher S. J., Fedkiw R. P.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002. (Cited on pages 72 and 74.)

[PCY09] Palmerius K. L., Cooper M., Ynnerman A.: Flow field visualization using vector field perpendicular surfaces. In *Proc. SCCG* (2009), pp. 27–34. (Cited on pages 138, 145, and 156.)

[PF01] Perry R. N., Frisken S. F.: Kizamu: A system for sculpting digital characters. In *Proc. SIGGRAPH* (2001), pp. 47–56. (Cited on pages 16 and 83.)

[PGB03] Pérez P., Gangnet M., Blake A.: Poisson image editing. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 313–318. (Cited on page 142.)

[PGL*12] Peng Z., Grundy E., Laramee R. S., Chen G., Croft N.: Mesh-driven vector field clustering and visualization: An image-based approach. *IEEE TVCG 18*, 2 (2012), 283–298. (Cited on page 167.)

[PJS06] Popa T., Julius D., Sheffer A.: Material-aware mesh deformations. In *Proc. SMI* (2006), pp. 141–152. (Cited on pages 14 and 85.)

[PP93] Pinkall U., Polthier K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1 (1993), 15–36. (Cited on pages 106 and 143.)

[PPF*11] Pobitzer A., Peikert R., Fuchs R., Schindler B., Kuhn A., Theisel H., Matkovic K., Hauser H.: The state of the art in topology-based visualization of unsteady flow. *Comput. Graph. Forum 30*, 6 (2011), 1789–1811. (Cited on page 137.)

[PS09] Peikert R., Sadlo F.: Topologically relevant stream surfaces for flow visualization. In *Proc. SCCG* (2009), pp. 43–50. (Cited on pages 137 and 164.)

[PTVF07] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007. (Cited on pages 37, 60, 77, and 97.)

[PVH*02] Post F. H., Vrolijk B., Hauser H., Laramee R. S., Doleisch H.: Feature extraction and visualisation of flow fields. In *Eurographics STARs*. 2002,

pp. 69–100. (Cited on page 137.)

[PVH*03] Post F., Vrolijk B., Hauser H., Laramee R., Doleisch H.: The state of the art in flow visualization: Feature extraction and tracking. *Comput. Graph. Forum 22*, 4 (2003), 775–792. (Cited on page 137.)

[PW89] Pentland A., Williams J.: Good vibrations: modal dynamics for graphics and animation. In *Proc. SIGGRAPH* (1989), pp. 215–222. (Cited on page 14.)

[RKS00] Rössl C., Kobbelt L., Seidel H.-P.: Line art rendering of triangulated surfaces using discrete lines of curvature. In *Proc. WSCG* (2000), pp. 168–175. (Cited on page 167.)

[RPP*09] Rosanwo O., Petz C., Prohaska S., Hotz I., Hege H.-C.: Dual streamline seeding. In *Proc. PacificVis* (2009), pp. 9–16. (Cited on page 138.)

[RSSG01] Rezk-Salama C., Scheuering M., Soza G., Greiner G.: Fast volumetric deformation on general purpose hardware. In *Proc. GH* (2001), pp. 17–24. (Cited on pages 16 and 83.)

[RTD*10] Ritschel T., Thormählen T., Dachsbacher C., Kautz J., Seidel H.-P.: Interactive on-surface signal deformation. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 36:1–36:8. (Cited on page 14.)

[RV11] Rossignac J., Vinacua A.: Steady affine motions and morphs. *ACM Trans. Graph. 30*, 5 (2011), 116:1–116:16. (Cited on page 14.)

[RWE08] Rössler F., Wolff T., Ertl T.: Direct GPU-based volume deformation. In *Proc. CURAC* (2008), pp. 65–68. (Cited on page 16.)

[SA07] Sorkine O., Alexa M.: As-rigid-as-possible surface modeling. In *Proc. SGP* (2007), pp. 109–116. (Cited on pages 13, 31, 46, 51, 68, 116, 118, 142, and 147.)

[Sab10] Sabin M.: *Analysis and Design of Univariate Subdivision Schemes*. Springer, 2010. (Cited on page 176.)

[SBBG11a] Solomon J., Ben-Chen M., Butscher A., Guibas L.: As-killing-as-possible vector fields for planar deformation. *Comput. Graph. Forum (Proc. SGP) 30*, 5 (2011), 1543–1552. (Cited on pages 9, 11, 15, 17, 20, 31, 33, 35, 40, 41, 46, 47, 52, 119, 124, 128, and 199.)

[SBBG11b] Solomon J., Ben-Chen M., Butscher A., Guibas L.: Discovery of intrinsic primitives on triangle meshes. *Comput. Graph. Forum (Proc. SGP) 30*, 2 (2011), 365–374. (Cited on page 48.)

[SBH*01] Scheuermann G., Bobach T., Hagen H., Mahrous K., Hamann B., Joy K. I., Kollmann W.: A tetrahedra-based stream surface algorithm. In *Proc. VIS* (2001), pp. 151–158. (Cited on page 137.)

[SBH07] Schulze F., Bühler K., Hadwiger M.: Interactive deformation and visualization of large volume datasets. In *Proc. GRAPP* (2007). (Cited on pages 16 and 83.)

[SCL*04] Sorkine O., Cohen-Or D., Lipman Y., Alexa M., Rössl C., Seidel H.-P.: Laplacian surface editing. In *Proc. SGP* (2004), pp. 175–184. (Cited on pages 12 and 61.)

[SDC09] Sýkora D., Dingliana J., Collins S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proc. NPAR* (2009), pp. 25–33. (Cited on page 13.)

[SFCP12] Sigg S., Fuchs R., Carnecky R., Peikert R.: Intelligent cutaway illustrations. In *Proc. PacificVis* (2012), pp. 185–192. (Cited on pages 175 and 182.)

[SGRT12] Schulze M., Germer T., Rössl C., Theisel H.: Stream surface parametrization by flow-orthogonal front lines. *Comput. Graph. Forum*

(Proc. SGP) 31, 5 (2012), 1725–1734. (Cited on pages 137, 149, 153, 154, and 155.)

[SH95] Stalling D., Hege H.-C.: Fast and resolution independent line integral convolution. In *Proc. SIGGRAPH* (1995), pp. 249–256. (Cited on page 138.)

[SHZO07] Sengupta S., Harris M., Zhang Y., Owens J. D.: Scan primitives for gpu computing. In *Proc. GH* (2007). (Cited on page 60.)

[SK06] Suman B., Kumar P.: A survey of simulated annealing as a tool for single and multiobjective optimization. *JORS 57*, 1 (2006), 1143–1160. (Cited on page 175.)

[SLCZ09] Spencer B., Laramee R. S., Chen G., Zhang E.: Evenly spaced streamlines for surfaces: An image-based approach. *Comput. Graph. Forum 28*, 6 (2009), 1618–1631. (Cited on page 167.)

[SM00] Shi J., Malik J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 8 (2000), 888–905. (Cited on page 66.)

[SMW06] Schaefer S., McPhail T., Warren J.: Image deformation using moving least squares. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (2006), 533–540. (Cited on pages 13 and 47.)

[SP86] Sederberg T. W., Parry S. R.: Free-form deformation of solid geometric models. In *Proc. SIGGRAPH* (1986), vol. 20, pp. 151–160. (Cited on page 13.)

[SP04] Sumner R. W., Popović J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (2004), 399–405. (Cited on pages 12, 90, 101, 142, and 144.)

[SRGT12] Schulze M., Rössl C., Germer T., Theisel H.: As-perpendicular-as-possible surfaces for flow visualization. In *Proc. PacificVis* (2012), pp. 153–160. (Cited on pages 138, 145, 146, 148, 153, 158, and 179.)

[SRWS10] Schneider D., Reich W., Wiebel A., Scheuermann G.: Topology aware stream surfaces. *Comput. Graph. Forum (Proc. EuroVis) 29*, 3 (2010), 1153–1161. (Cited on pages 137, 179, and 181.)

[SS02] Strothotte T., Schlechtweg S.: *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann, 2002. (Cited on page 167.)

[SS08] Schultz T., Seidel H.-P.: Estimating crossing fibers: A tensor decomposition approach. *IEEE TVCG (Proc. Vis) 14*, 6 (2008), 1635–1642. (Cited on page 161.)

[SSP07] Sumner R. W., Schmid J., Pauly M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007). (Cited on page 13.)

[Sta98] Stalling D.: *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, ZIB, 1998. (Cited on pages 137 and 153.)

[Sta99] Stam J.: Stable fluids. In *Proc. SIGGRAPH* (1999), pp. 121–128. (Cited on page 73.)

[STC*12] Schumacher C., Thomaszewski B., Coros S., Martin S., Sumner R., Gross M.: Efficient simulation of example-based materials. In *Proc. SCA* (2012), pp. 1–8. (Cited on page 14.)

[STWE07] Schafhitzel T., Tejada E., Weiskopf D., Ertl T.: Point-based stream surfaces and path surfaces. In *Proc. GI* (2007), pp. 289–296. (Cited on page 137.)

[SVWG12] Solomon J., Vouga E., Wardetzky M., Grinspun E.: Flexible developable surfaces. *Comput. Graph. Forum (Proc. SGP) 31*, 5 (2012), 1567–1576. (Cited on pages 9 and 55.)

[SW04] Sweet G., Ware C.: View direction, surface orientation and texture orientation for perception of surface shape. In *Proc. GI* (2004), pp. 97–106. (Cited on page 168.)

[SWH05] Sahner J., Weinkauf T., Hege H.-C.: Galilean invariant extraction and iconic representation of vortex core lines. In *Proc. EuroVis* (2005), pp. 151–160. (Cited on page 181.)

[SWS09] Schneider D., Wiebel A., Scheuermann G.: Smooth stream surfaces of fourth order precision. *Comput. Graph. Forum (Proc. EuroVis) 28*, 3 (2009), 871–878. (Cited on pages 137 and 181.)

[SWST12] Stöter T., Weinkauf T., Seidel H.-P., Theisel H.: Implicit integral surfaces. In *Proc. VMV* (2012), pp. 127–134. (Cited on page 137.)

[SZGP05] Sumner R. W., Zwicker M., Gotsman C., Popović J.: Mesh-based inverse kinematics. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 488–495. (Cited on pages 14 and 85.)

[SZT*07] Shi X., Zhou K., Tong Y., Desbrun M., Bao H., Guo B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 81:1–81:10. (Cited on page 13.)

[TB96] Turk G., Banks D.: Image-guided streamline placement. In *Proc. GRAPHITE* (1996), pp. 453–460. (Cited on page 164.)

[TWHS03] Theisel H., Weinkauf T., Hege H.-C., Seidel H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3d vector fields. In *Proc. Vis* (2003), pp. 225–232. (Cited on page 164.)

[TWHS05] Theisel H., Weinkauf T., Hege H.-C., Seidel H.-P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *IEEE TVCG 11*, 4 (2005), 383–394. (Cited on pages 86, 88, and 89.)

[Vax12] Vaxman A.: Modeling polyhedral meshes with affine maps. *Comput. Graph. Forum (Proc. SGP) 31*, 5 (2012), 1647–1656. (Cited on page 14.)

[vFTS06] von Funck W., Theisel H., Seidel H.-P.: Vector field based shape deformations. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (2006), 1118–1125. (Cited on pages 15, 74, 75, 83, and 100.)

[vFTS07a] von Funck W., Theisel H., Seidel H.-P.: Elastic secondary deformations by vector field integration. In *Proc. SGP* (2007), pp. 99–108. (Cited on page 15.)

[vFTS07b] von Funck W., Theisel H., Seidel H.-P.: Explicit control of vector field based shape deformations. In *Proc. Pacific Graphics* (2007), pp. 291–300. (Cited on pages 15, 36, 69, 75, and 83.)

[vFTS07c] von Funck W., Theisel H., Seidel H.-P.: Implicit boundary control of vector field based shape deformations. In *Proc. Mathematics of Surfaces* (2007), LNCS, pp. 154–165. (Cited on page 15.)

[vFTS08] von Funck W., Theisel H., Seidel H.-P.: Volume-preserving mesh skinning. In *Proc. VMV* (2008), pp. 407–414. (Cited on page 79.)

[vFWTS08] von Funck W., Weinkauf T., Theisel H., Seidel H.-P.: Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE TVCG (Proc. Vis) 14*, 6 (2008), 1396–1403. (Cited on page 137.)

[vW91] van Wijk J. J.: Spot noise texture synthesis for data visualization. In *Proc. SIGGRAPH* (1991), pp. 309–318. (Cited on page 138.)

[vW93] van Wijk J. J.: Implicit stream surfaces. In *Proc. Vis* (1993), pp. 245–252. (Cited on page 137.)

[WBCG09] Weber O., Ben-Chen M., Gotsman C.: Complex barycentric coordinates with applications to planar shape deformation. *Comput. Graph. Forum (Proc. Eurographics) 28*, 2 (2009), 587–597. (Cited on pages 13 and 47.)

[WBG07] Wicke M., Botsch M., Gross M.: A finite element method on convex polyhedra. *Comput. Graph. Forum (Proc. Eurographics) 26*, 3 (2007), 355–364. (Cited on page 14.)

[WBGH11] Weber O., Ben-Chen M., Gotsman C., Hormann K.: A complex view of barycentric mappings. *Comput. Graph. Forum (Proc. SGP) 30*, 5 (2011), 1533–1542. (Cited on pages 13 and 46.)

[WBRS09] Wirth B., Bar L., Rumpf M., Sapiro G.: Geodesics in shape space via variational time discretization. *LNCS 5681*, 1 (2009), 288–302. (Cited on page 15.)

[WDAH10] Winkler T., Drieseberg J., Alexa M., Hormann K.: Multi-scale geometry interpolation. *Comput. Graph. Forum (Proc. Eurographics) 29*, 2 (2010), 309–318. (Cited on pages 14, 70, 85, and 90.)

[Wen04] Wendland H.: *Scattered Data Approximation*. Cambridge University Press, 2004. (Cited on pages 87 and 92.)

[WHT12] Weinkauf T., Hege H.-C., Theisel H.: Advected tangent curves: A general scheme for characteristic curves of flow fields. *Comput. Graph. Forum (Proc. Eurographics) 31*, 2 (2012), 825–834. (Cited on pages 135 and 137.)

[WK88] Witkin A., Kass M.: Spacetime constraints. In *Proc. SIGGRAPH* (1988), pp. 159–168. (Cited on page 14.)

[WMKG07] Wardetzky M., Mathur S., Kaelberer F., Grinspun E.: Discrete laplace operators: No free lunch. In *Proc. SGP* (2007), pp. 33–37. (Cited on page 106.)

[WMZ12] Weber O., Myles A., Zorin D.: Computing extremal quasiconformal maps. *Comput. Graph. Forum (Proc. SGP) 31*, 5 (2012), 1679–1689. (Cited on pages 13 and 46.)

[WPP07] Wang R. Y., Pulli K., Popović J.: Real-time enveloping with rotational regression. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 73:1–73:10. (Cited on pages 14 and 85.)

[WR01] Westermann R., Rezk-Salama C.: Real-time volume deformation. *Comput. Graph. Forum (Proc. Eurographics) 20*, 3 (2001), 443—-451. (Cited on pages 16 and 83.)

[WSLG07] Weber O., Sorkine O., Lipman Y., Gotsman C.: Context-aware skeletal shape deformation. *Comput. Graph. Forum (Proc. Eurographics) 26*, 3 (2007), 265–273. (Cited on pages 14 and 85.)

[WTHS04] Weinkauf T., Theisel H., Hege H.-C., Seidel H.-P.: Boundary switch connectors for topological visualization of complex 3d vector fields. In *Proc. VisSym* (2004), pp. 183–192. (Cited on page 164.)

[WW92] Welch W., Witkin A.: Variational surface modeling. *Proc. SIGGRAPH 26*, 2 (1992), 157–166. (Cited on page 168.)

[WXXC08] Wang Y., Xu K., Xiong Y., Cheng Z.-Q.: 2d shape deformation based on rigid square matching. *Comput. Animat. Virtual Worlds 19*, 3 (2008), 411–420. (Cited on page 13.)

[XWY*09] Xu W., Wang J., Yin K., Zhou K., van de Panne M., Chen F., Guo B.: Joint-aware manipulation of deformable models. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 35:1–35:9. (Cited on page 14.)

[XZWB06] Xu D., Zhang H., Wang Q., Bao H.: Poisson shape interpolation. *Graph. Models 68*, 3 (2006), 268–281. (Cited on pages 85, 90, 102, 142, and 144.)

[XZY*07] Xu W., Zhou K., Yu Y., Tan Q., Peng Q., Guo B.: Gradient domain editing of deforming mesh sequences. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 84:1–84:10. (Cited on page 12.)

[YYPM11] Yang Y.-L., Yang Y.-J., Pottmann H., Mitra N. J.: Shape space exploration of constrained meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 6 (2011), 124:1–124:12. (Cited on page 15.)

[YZX*04] Yu Y., Zhou K., Xu D., Shi X., Bao H., Guo B., Shum H.-Y.: Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (2004), 644–651. (Cited on pages 12, 142, and 144.)

[ZG07] Zhu Y., Gortler S.: *3D Deformation Using Moving Least Squares*. Tech. rep., Harvard University, 2007. (Cited on page 13.)

[ZGHG10] Zhou K., Gong M., Huang X., Guo B.: Data-parallel octrees for surface reconstruction. *IEEE TVCG 17*, 5 (2010), 669–681. (Cited on page 142.)

[ZRKS05] Zayer R., Rössl C., Karni Z., Seidel H.-P.: Harmonic guidance for surface deformation. *Comput. Graph. Forum (Proc. Eurographics) 24*, 3 (2005), 601–609. (Cited on pages 12, 61, 142, and 144.)

[ZRS05] Zayer R., Rössl C., Seidel H.-P.: Discrete tensorial quasi-harmonic maps. In *Proc. SMI* (2005), pp. 278–287. (Cited on pages 142 and 147.)

[ZvKD10] Zhang H., van Kaick O., Dyer R.: Spectral mesh processing. *Comput. Graph. Forum 29*, 6 (2010), 1865–1894. (Cited on pages 65 and 196.)

ADDITIONAL REFERENCES

[1] ANN. ANN: A library for approximate nearest neighbor searching, 2010. Version 1.1.2 cs.umd.edu/~mount/ANN. (Cited on page 94.)

[2] ANSYS Inc. CFD-Post, 2012. Version 12.1 ansys.com. (Cited on page 136.)

[3] CGAL. Computational Geometry Algorithms Library, 2013. www.cgal.org. (Cited on pages 78 and 90.)

[4] Roger A. Crawfis. Tornado flow field, 1995. cse.ohio-state.edu/~crawfis/Data/Tornado. (Cited on page 178.)

[5] Jared Hoberock and Nathan Bell. NVIDIA Thrust: A parallel template library, 2012. Version 1.6.0, thrust.github.com. (Cited on pages 37, 60, and 97.)

[6] Intel. Math kernel library MKL, 2013. Version 11.0 software.intel.com/en-us/intel-mkl. (Cited on page 96.)

[7] NVIDIA. CUDA basic linear algebra subroutines (cuBLAS), 2013. Version 5.0 developer.nvidia.com/cublas. (Cited on page 96.)

[8] K. B. Petersen and M. S. Pedersen. The matrix cookbook. DTU, 2008. (Cited on page 95.)

[9] Pixologic. Zbrush, 2012. Version 4R5, pixologic.com/zbrush. (Cited on page 36.)

[10] VSG. Amira, 2013. vsg3d.com/amira. (Cited on page 219.)

DATA ACKNOWLEDGMENTS

The physical Otto-von-Guericke bust in Figure 1 is courtesy of Thorsten Grosch, the scanned mesh was optimized by Maik Schulze. The crocodile mummy of the same Figure was scanned by Rebecca Fahrig. We thank Mario Botsch and Olga Sorkine for providing benchmark deformation shapes in their survey [BS08]. The bonsai data set shown in Figure 36 is courtesy of Stefan Röttger. The lion, cat, and horse poses used in Chapter 6 were kindly provided by Robert Sumner. We thank Tino Weinkauf for resampling the CYLINDER data set used in the second part of this thesis. Gábor Janiga simulated the ANEURYSM flow. The DELTAWING and ACOUTLET data sets is courtesy of Markus Rütten. We thank Axel Seeger and Klaus Affold for providing the BUBBLECHAMBER data set. The TURBINE flow is part of the visualization system AMIRA [10]. The hand-drawn flow illustrations by Uwe Dallmann that are shown in Figure 65 were included with the kind permission of the DLR Göttingen.

DECLARATION

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, Dezember 2013

Janick Martinez Esturo