ANALYSIS-AWARE HIGHER ORDER SMOOTH THREE-DIMENSIONAL REPRESENTATIONS: CREATION, SIMULATION AND VISUALIZATION

by

Tobias Martin

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

May 2012

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Tobias Martin**

has been approved by the following supervisory committee members:

Elaine Cohen	, Chair	**12/9/11** Date Approved
Robert M. Kirby	, Member	**12/9/11** Date Approved
Richard F. Riesenfeld	, Member	**12/9/11** Date Approved
Charles Hansen	, Member	**12/9/11** Date Approved
Tom Lyche	, Member	**12/9/11** Date Approved

and by **Alan L. Davis** , Chair of

the Department of **School of Computing**

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

Volumetric parameterization is an emerging field in computer graphics, where volumetric representations that have a semi-regular tensor-product structure are desired in applications such as three-dimensional (3D) texture mapping and physically-based simulation. At the same time, volumetric parameterization is also needed in the Isogeometric Analysis (IA) paradigm, which uses the same parametric space for representing geometry, simulation attributes and solutions. One of the main advantages of the IA framework is that the user gets feedback directly as attributes of the NURBS model representation, which can represent geometry exactly, avoiding both the need to generate a finite element mesh and the need to reverse engineer the simulation results from the finite element mesh back into the model. Research in this area has largely been concerned with issues of the quality of the analysis and simulation results assuming the existence of a high quality volumetric NURBS model that is appropriate for simulation. However, there are currently no generally applicable approaches to generating such a model or visualizing the higher order smooth isosurfaces of the simulation attributes, either as a part of current Computer Aided Design or Reverse Engineering systems and methodologies. Furthermore, even though the mesh generation pipeline is circumvented in the concept of IA, the quality of the model still significantly influences the analysis result. This work presents a pipeline to create, analyze and visualize NURBS geometries. Based on the concept of analysis-aware modeling, this work focusses in particular on methodologies to decompose a volumetric domain into simpler pieces based on appropriate midstructures by respecting other relevant interior material attributes. The domain is decomposed such that a tensor-product style parameterization can be established on the subvolumes, where the parameterization matches along subvolume boundaries. The volumetric parameterization is optimized using gradient-based nonlinear optimization algorithms and datafitting methods are introduced to fit trivariate B-splines to the parameterized subvolumes with guaranteed order of accuracy. Then, a visualization method is proposed allowing to directly inspect isosurfaces of attributes, such as the results of analysis, embedded in the NURBS geometry. Finally, the various methodologies proposed in this work are demonstrated on complex representations arising in practice and research.

To Sumathi and my parents

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Elaine Cohen. It has been a real honor for me to be her Ph.D. student and academic son. I am very grateful and deeply appreciative of the fact that she was always patient with me and found time to discuss research. She also gave me advice and helped to solve problems related to my dissertation research, concrete feedback on the research papers I was working on, and also the freedom to pick new research problems. These things are not at all natural for a graduate student. Also, I am grateful that she funded me throughout my graduate studies (in part by the grant NSF IIS-1117997). Most importantly, the knowledge of geometry and math that she taught me opened doors to other research fields for me.

I would like to thank Mike Kirby for his support and patience to teach me everything I know about the finite element method, which is one of the fundamental building blocks of this work. I thank Richard F. Riesenfeld for his professional advice and wisdom, and for giving me the opportunity to speak in German. I am very glad that I had Charles Hansen in my Ph.D. committee, and I am thankful for his important feedback on several of my research projects and for the opportunity to be a teaching assistant for his computer graphics class. Finally, I am proud to have Tom Lyche on one of my major Ph.D. relevant publications, and for being a member of my Ph.D. committee. Gratitude also goes to Pushkar Joshi and Nathan Carr, who gave me the opportunity to intern at Adobe Systems. The project that we worked on is a part of this dissertation. Many thanks also to the staff of School of Computing, especially to Karen Feinauer and Ann Carlstrom who helped resolve a lot of issues related to my study.

I am very grateful that during my life as a graduate student, I found a very good friend in Mathias Schott. I believe that this friendship will last beyond my graduate studies. I am also very grateful that I was surrounded by Suraj Musuvathy and Guoning Chen. It was an honor to collaborate with them. I thank Pooyan Amini for being my gym buddy and introducing me to delicious Persian cuisine, and his brother Peiman Amini for the nice hikes. These friendships made the success of this work possible.

Words cannot express my gratitude to my parents, who supported me not only through graduate school, but supported me and made sacrifices for me my whole life. And for Sumathi, who was always there for me even though she was geographically far away. Her love, constant encouragement, and never ending patience with me let me succeed. Thank you Sumi and Mama and Papa, this work is dedicated to you.

CHAPTER 1

INTRODUCTION

The finite element method (FEM) [86] is a numerical technique to find approximate solutions to partial differential equations (PDE) on an object of interest. It is the *de facto* standard of many analysis packages which are part of Computer Aided Engineering (CAE) software. Given a Computer Aided Design (CAD) representation of the object of interest, in order to run analysis, the CAD model must be converted using a meshing technique into a CAE representation based, for instance, on hexahedral or tetrahedral elements. This conversion is often time consuming because generally, these CAD models are not watertight, nonmanifold or only represent the boundary of the object of interest and also often require manual work to repair the CAD representation due to these problems. See for instance Butlin et al. [26] for more discussion. A large amount of effort is expended in mesh generation [150] and optimization in also terms of mesh quality [108], a process which frequently takes much longer than the actual analysis. Once analysis has been performed, and the result of the analysis has been evaluated, the original CAD model typically must be modified and the procedure to generate a representation for FEM has to be iterated several times until a satisfactory solution is attained.

Proposing to use the same Non-Uniform Rational B-Splines (NURBS) [37] bases that are used to define the parametric geometry as the underlying representation for approximating the solutions to the PDE, Isogeometric Analysis (IA) introduced by Hughes et al. [87] claims to break this time-consuming cycle. In IA which is based on the Galerkin [86] method, simulation parameters, such as forces or material parameters are represented as attributes of the NURBS representation [87]. One of the main advantages of the IA framework is that the user gets feedback directly as attributes of the NURBS model representation, representing geometry exactly, thereby avoiding both the need to generate a finite element mesh and the need to reverse engineer the simulation results from the finite element mesh back into the model. Since IA works directly on the exact CAD representations, geometric refinement is unnecessary and an analyist can focuse only on refinement of the solution. However, as discussed below, one of the main challenges in IA is to create a volumetric shape representation suitable for analysis.

By following [87] and [36], the IA pipepline consists of three stages:

1: Modification of the Shape Representation (SR) depending on analysis needs.

2: Analysis on the SR.

3: Investigation of the solution on SR through error analysis and visualization. Go back to stage 1 or 2 as necessary.

Research in IA bridges aspects of approximation theory, modeling algorithms, simulation, and visualization. Much research in IA has largely been concerned with issues of the quality of the analysis and simulation results assuming the existence of a shape representation (e.g., trivariate NURBS or T-spline model) that is appropriate for simulation. While Lipton et al. [116] demonstrate that analysis still can be performed successfully on inferior NURBS representations, it will be shown in this work that by carefully creating representations, numerical stability and convergence rates can be significantly improved. There are currently no generally applicable approaches to generating such a representation or visualization methodologies suitable to visualize the solution on the higher-order NURBS representations.

One of the main foci of this work lies in proposing methodologies to create CAD representations that are suitable for analysis, so they, in turn, serve as CAE representations as well. We shall call such a representation a CAD/E representation. As discussed below, there are two paths to create a shape representation.

In the *active* modeling path, the CAD/E representation is created ab initio by a modeler using volumetric shape operations. This path is demonstrated, for instance, on a turbine blade in [4]. Already in the design process, tools and methodologies are necessary to produce models which do not require extensive manual repair, since CAD systems in general do not create representations suitable to transform into volumetric representations. An important problem is to assist the user in generating a full volume shape representation and related attribute representation early in the design process that is appropriate for IA, and to maintain it as the design evolves.

In the *post facto* modeling path, the representation is created from a point cloud, CT scan, triangle mesh boundaries, etc., depending on the application. The input as well can consist of multiple boundaries separating materials within the volume. Figure 1.1 illustrates the input for a femur and pelvis data set with inner and outer boundary represented by triangle meshes. Filling up the interior of the model, i.e., model completion, is a hard problem for surfaces, where the input is a set of boundary curves, and is even more difficult for volumes. As a first step, a volumetric parameterization is created from the input data. Then, given this parameterization, a NURBS representation is generated from it. Depending on the input, a global parameterization may not be possible, because of higher geometric complexity and topology. Therefore, decomposition strategies are needed to consistenly decompose the input object into a set of regions that are parameterized so that region parameterizations match along common boundaries. In the second step, given a parameterization of the object, data

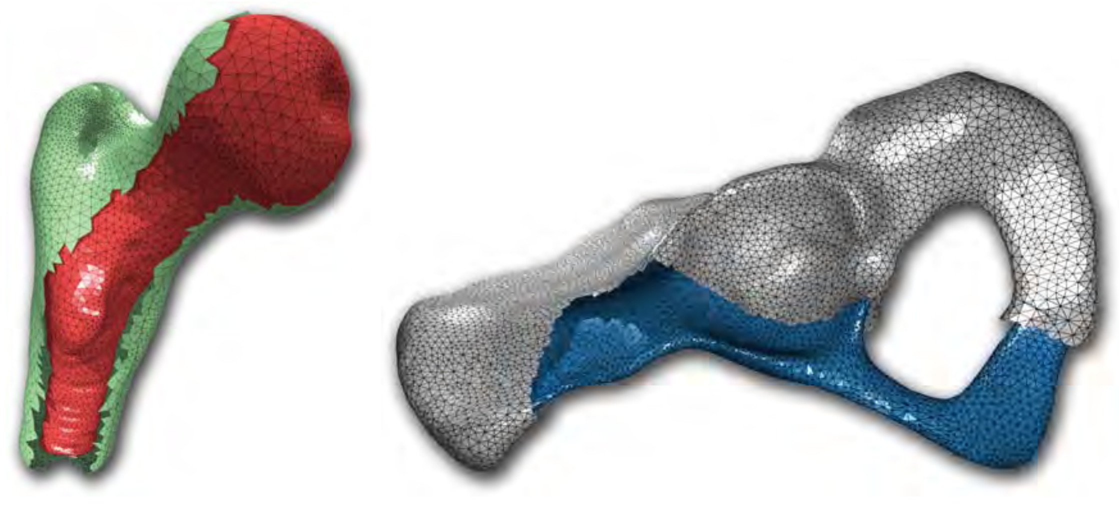


Figure 1.1. Input data. On the left: Triangle mesh boundaries of a femur bone; on the right: Triangle mesh boundaries of a pelvis bone. In both examples, the inner boundary separates cortical from trabecular bone material.

fitting techniques are required to create the final shape representation. For that, the parameterization must be sampled so that the resulting shape represention satisfy a specified error tolerance. Creating such a representation for IA is not an automatic approach and is therefore user-assisted. This is important because the user is aware of the parts of the object of interest that are important in simulation and require higher fidelity and so should be free of extraordinary points or degeneracies in the representation compared to other parts of the object. Therefore, various parameters can be specified by the user affecting the simulation result. For instance, given two models representing the model of interest, one may lead to a better rate of solution convergence relative to the other. Since there can be many representations for a single geometry, an appropriate representation should take into account the particular analysis being performed, as a study in [39] has demonstrated. Since in IA, the mesh generation pipeline is eliminated, issues affecting model quality must be considered. That is, model quality is a characterization of those representation properties of the model geometry that impact the model representing the object of interest. However, the longterm goal is to make more of the modeling steps automatic to reduce overall modeling time.

Once simulation has been applied to a trivariate representation of a physical domain, in general, mesh extraction techniques or direct visualization techniques are needed to inspect the result of the simulation. This is the third stage of the IA framework. Techniques either directly visualize the solution using, e.g., a ray-based method, or the isosurfaces are extracted and represented as triangle meshes that are rendered. The majority of finite element visualization software, however, displays only linear shape elements. For instance the Marching Cubes technique [119] is based on a uniform grid and only approximates isosurfaces linearly. Although some extensions have generalized it to higher order surfaces, a represention used in IA does not necessarily share the uniform grid structure. A straightforward method is to sample the higher-order geometry and solution representation and create a unstructured tetrahedral mesh and use Marching Tetrahedra [33] to extract isosurfaces. However, finding the correct sampling to extract a topological correct isosurface is a complex issue. Furthermore, it violates the IA concept inasmuch as the geometric representation is changed.

As mentioned above, other approaches consider a per-ray basis such as [155, 142] and are generally based on classic root finding techniques that use Jacobians or interval arithmetic [134]. NURBS involve an additional nonlinear map where classic methods are difficult to adopt and often fail for complex geometry. Research on direct NURBS visualization has focused on parametric surfaces and trivariate attribute NURBS defined over uniform spatial grids, but there is little work to visualize NURBS isosurfaces for attributes of general NURBS geometry.

1.1 Outline

Figure 1.2 gives an overview of the pipeline proposed in this work. Model completion, shape representation and analysis are the fundamental building blocks of this work. The Appendix gives an overview of the publications on which this dissertation is based.

1.1.1 Model Completion

As discussed above, there are two long term goals in geometric modeling: The ab initio modeling path where the user creates a CAD/E representation using volumetric shape operations and the *post facto* modeling path where input boundaries are given and a volumetric representation is created based on these boundaries. Both are formidable problems.

This work mostly focuses on the second goal, where in practice a closed triangle mesh in three space is given. This triangle mesh represents the boundary of the physical domain. Commonly, interior triangle meshes are given as well, separating different materials within the physical domain as seen in Figure 1.1. For IA, the goal is to create volume elements in the region enclosed by the multiple boundary triangle meshes. This process is referred to as model completion. The chapters outlined in Figure 1.2, where each one introduces a modeling methodology, are the core of this dissertation.

The choice of appropriate modeling methodology strongly depends on the complexity of the input boundaries and the answers to the following questions: What is the genus of the boundaries, what is the size of the features, are the interior boundaries contained within each other, and so forth.

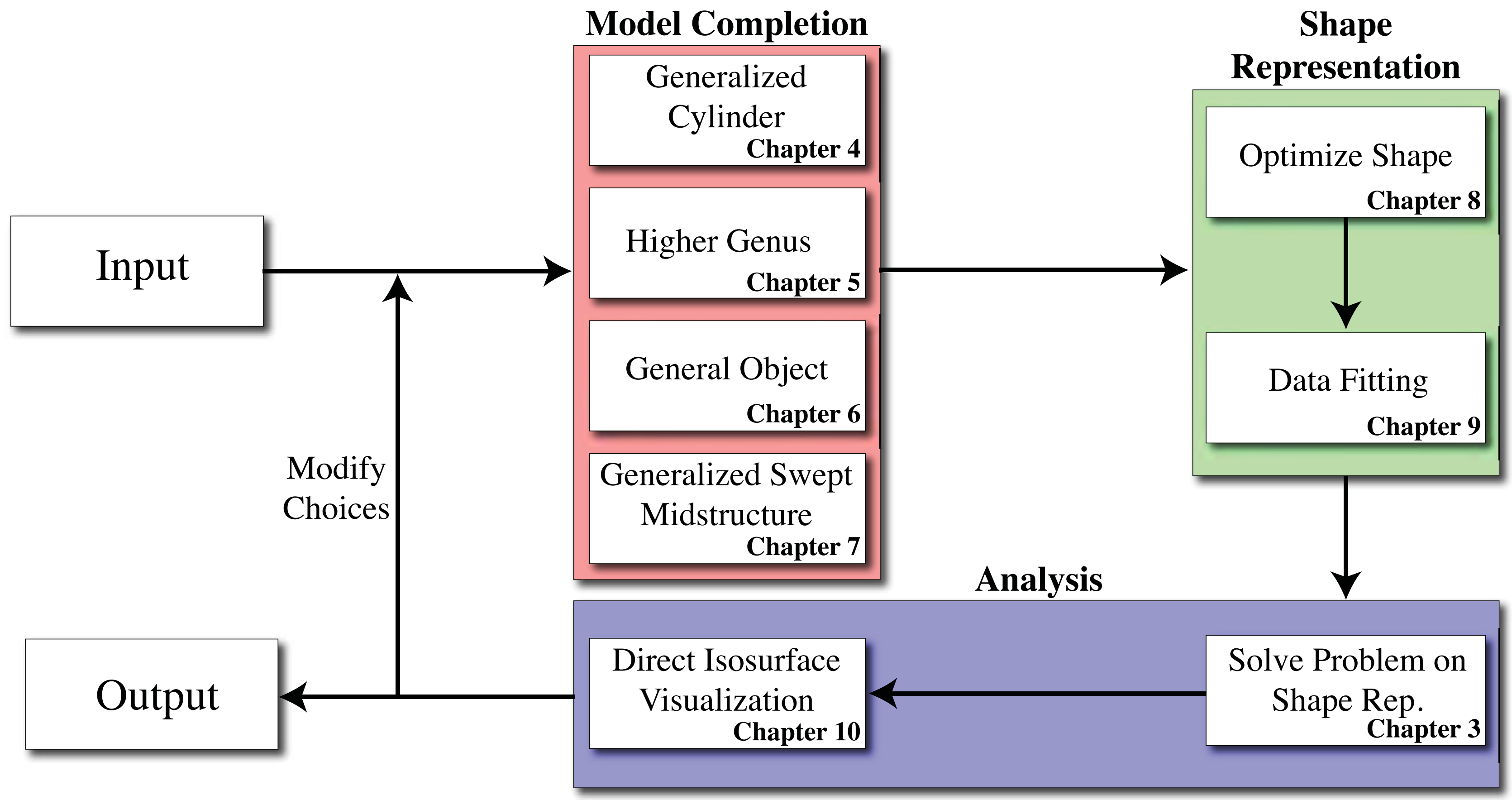


Figure 1.2. Overview of the pipeline introduced in this work. In the first step the volume of an input model is completed. Then, this parameterization is optimized and a higher order representation is generated from it. Simulation is applied to the volumetric shapre respresentation which is then visualized and inspected using direct isosurface visualization.

In the case where the object of interest exhibits genus-0 and cylindrical-like shape, Chapter 4 proposes a methodolgy to create a single and global volumetric parameterization of the object that could embody shape overhangs (see femur data set in Figure 1.1). This approach is often desired because decomposing of the object into subvolumes and subsequently recombining them is a complex process. Achieving continuity can be elusive, and decomposing in this manner often introduces extraordinary points.

The next level of complexity involves creating a volumetric parameterization for an object with higher genus and structural bifurcations, shapes for which a single parameterization is very difficult, if not impossible, to achieve. For instance, if the object is of genus-1, a single parameterization can be established by sweeping a surface along a closed curve. However, due to the complex shape, the resulting parameterization may contain severe distortion as evidenced in the pelvis data set in Figure 1.1 and detailed in Figure 1.3. Chapter 5 proposes a methodology for modeling more complex objects that can exhibit higher genus and contain multiple bifurcations by consistently decomposing the objects into subvolumes.

For the more general case, the input object could be any 2-manifold object, where the methods above are not applicable. In the general case it is very difficult to establish a volumetric parameterization consisting solely of NURBS elements. Therefore, Chapter 6

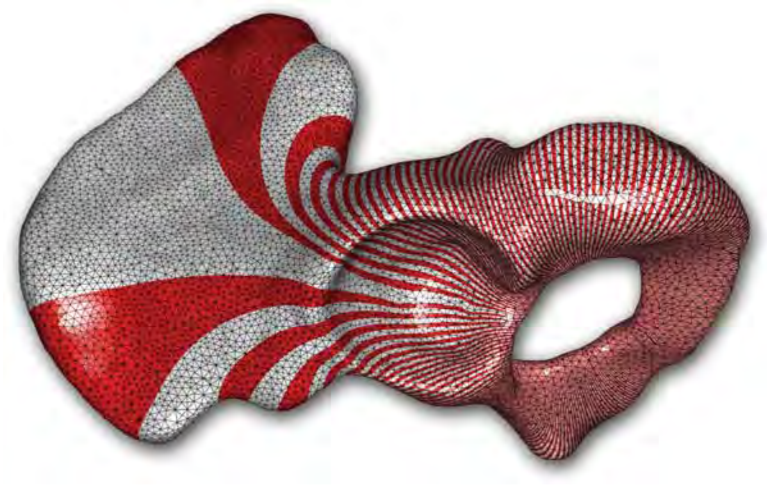


Figure 1.3. Sweeping can cause significant of parametric distortion, if the object is irregular.

introduces a mixed-element methodology that constructs a mixed-element volumetric representation of an arbitrary 2-manifold with high quality semistructured NURBS elements in selected regions and unstructured tetrahedral elements in the other regions.

The methodologies discussed in these chapters are each based on a midstructure representation. Since these midstructures are very specific to the respective modeling methods, they are introduced in the respective section with the corresponding modeling method. Chapter 7 introduces a more general framework to compute a midstructure of a given object. We demonstrate that it is suitable for hexahedral meshing and other applications like medial-based shape deformation.

1.1.2 Shape Representation

Once the model is completed, a higher order representation such as a trivariate B-spline is fit to an optimized version of the completed model. Chapters 8 and 9 discuss these two pipeline stages.

1.1.3 Analysis

Chapter 3 introduces the concept of analysis aware modeling by discussing the classic FEM pipeline and the IA pipeline. It will be demonstrated that different representations result in different convergence behavior. This is central to this work, as the user significanly influences the analysis, i.e., each choice made by the user during the modeling process results in a different analysis result.

1.1.4 Visualization of Analysis Result

Finally, once simulation has been applied to the representation, the solution is examined via isosurface visualization. Chapter 10 presents an approach which allows the user to directly inspect a solution without generating an intermediate representation of the volumetric object.

The following section introduces core concepts such as parameterization and harmonic functions used throughout the rest of this document. Chapter 2 reviews related work in the fields of parameterization, visualization, mesh generation and mesh quality.

1.2 Background

This chapter briefly presents core concepts required for this work.

1.2.1 Discrete Harmonic Functions

In this work harmonic functions are used to establish a volumetric parameterization over a domain Ω respecting inner material attributes. In general, a harmonic function is a function $u \in C^2(\Omega), u : \Omega \to \mathbb{R}$, with boundary $\partial\Omega$ satisfying Laplace's equation,

$$\nabla^2 u = 0, \tag{1.1}$$

where $\Omega \in \mathbb{R}^d$ and $\nabla^2 = \sum_i^d \frac{\partial^2}{\partial x_i^2}$ is the Laplace operator. In our case $d = 2, 3$, i.e., when $d = 2$ then $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, where $\partial\Omega$ is a polyline. When $d = 3$, then $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ where $\partial\Omega$ is a triangle mesh. u satisfies the maximum principle, i.e., it does not exhibit any local minima and maxima and is used in a variety of applications.

For instance in [91], harmonic functions are used for path-planning of complex robots in complex geometric models. The configuration space represents all degrees of freedom of the robot in the scene, where the location of the robot is fully described by a point. Harmonic functions are computed in this space, where paths along the gradient field of the harmonic function are computed, guiding the robot through the scene without colliding with obstacles. Figure 1.4 shows a cut-through of the harmonic function computed on a tesselated configuration space example. The figure also shows the corresponding paths.

The maximum principle makes harmonic functions suitable for tensor-products like parameterization as shown in [50, 196, 126]. In this work harmonic functions are utilized in order to fit a trivariate tensor product B-spline to a tetrahedral mesh generated from a set of triangulated isosurfaces. While in [143] harmonic functions are used for a robust Morse analysis [132] on triangle meshes, here they are used for geometric volume modeling. Chapter 4 uses harmonic vector fields, where the harmonic properties allow a consistent extraction of paths to construct a hexahedral representation (i.e., no self-intersection between adjacent paths) even when the boundary triangle mesh has overhangs (local concavities).

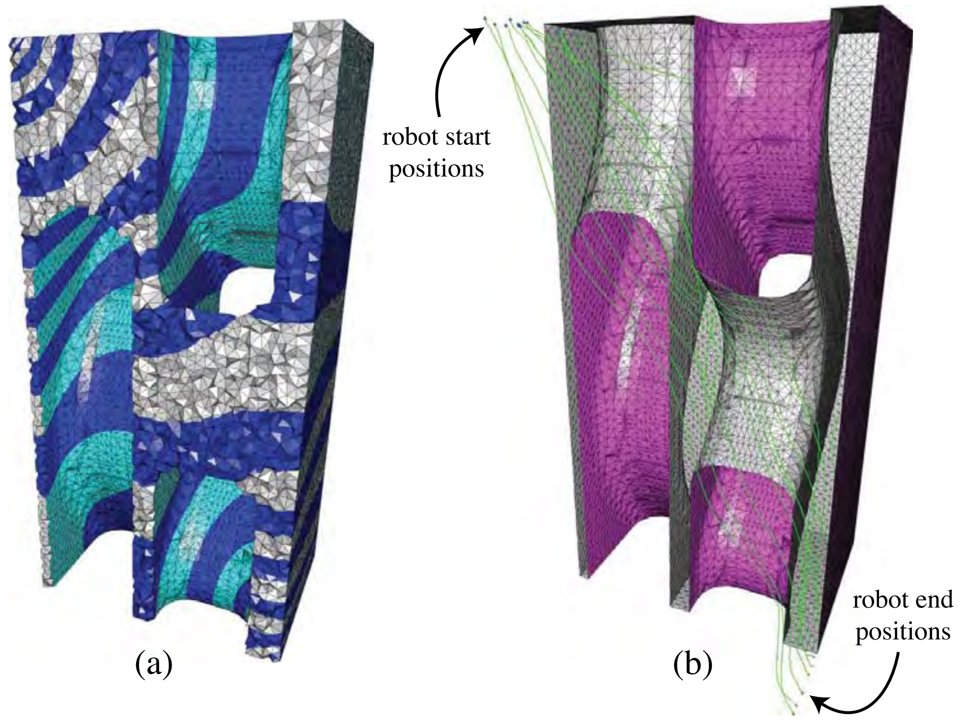


Figure 1.4. In (a), a harmonic function is computed through the volume of a configuration space. In (b), based on the gradient field of the harmonic function in (a), robot paths from start to corresponding end location are computed.

Herein, the finite element method (FEM) [86] is used to discretize Equation 1.1. The domain Ω is exactly represented with a tetrahedral mesh $(\mathcal{H}, \mathcal{T}, \mathcal{V}, \mathcal{C})$, where $\mathcal{H}$ is the set of tetrahedra. $\mathcal{T}$, the set of faces of the tetrahedra in $\mathcal{H}$. $\mathcal{V} \subset \mathbb{R}^3$, the set of vertices; and $\mathcal{C}$ specifies the connectivity of the mesh.

$\mathcal{V}$ can be decomposed into the sets $\mathcal{V}_B$ and $\mathcal{V}_I$, where $\mathcal{V}_B$ is the set of boundary vertices that lie on the Dirichlet boundaries that may correspond to the exterior triangle boundary or any interior triangular boundary. $\mathcal{V}_I$ is the set of vertices for which the solution is sought. The solution, then, is of the form,

$$u(x, y, z) = \sum_{v_k \in \mathcal{V}_I} \widehat{u}_k \phi_k(x, y, z) + \sum_{v_k \in \mathcal{V}_B} \widehat{u}_k \phi_k(x, y, z), \tag{1.2}$$

where $\phi_i(x, y, z)$ is the linear hat function associated with vertex v_i evaluating to 1 at v_i and to 0 at the other vertices defining the respective tetrahedron. The weak Galerkin's formulation is used to form the linear system $\mathbf{S}\vec{u} = \vec{f}$, where $\mathbf{S}$ is the stiffness matrix and $\vec{f}$ is a right-hand-side function. Being positive definite [86], S lends the linear system to being solved efficiently using the preconditioned conjugate gradient method [12].

The value of u at a point $p = (x, y, z)$ inside a tetrahedron is the linear combination $u(p) = \sum_{j=1}^{4} \hat{u}_j \phi_j(p)$, where $\hat{u}_j$ is associated with vertex v_j of the respective tetrahedron. Note, that j is a local index. The gradient field ∇u over Ω is piecewise constant, i.e., ∇u defined over a tetrahedron is the linear combination $\nabla u(p) = \sum_{i=1}^{4} \hat{u}_j \nabla \phi_j(p)$, where $\nabla \phi_j(p)$ is constant.

1.2.2 Parameterization

Given domain $\Omega \in \mathbb{R}^3$, a parameterization is a function

$$F : \Theta \to \Omega, \tag{1.3}$$

where F is a bijection and $\Theta \in \mathbb{R}^3$ is the reference domain. In the discrete case, for instance, two tetrahedral meshes $\mathcal{H}_\Theta$ and $\mathcal{H}$ are isomorphic if there is a correspondence between their vertices, edges, triangles and tetrahedra such that corresponding edges join corresponding vertices, corresponding triangles join corresponding vertices and edges and corresponding tetrahedra join corresponding vertices, edges and triangles.

As an example, the reference domain Θ of a trivariate B-spline is a single rectangular parallelepiped. On the other side, the domain of a polycube map [193] consists of many rectangular parallelepipeds, which is more similar to its corresponding physical domain Ω. Given such a parameterization, if Ω is to be represented by a higher-order trivariate B-spline, e.g., for purposes of applying physical analysis to Ω, its domain must be decomposed into a set of rectangular parallelopipeds where each one maps to some piece of Ω. Their images intersect only on parallelopiped boundaries. Since the collection of cubes is irregular, there can be extraordinary points on the surface so the T-spline generalization T-NURCCs [178] is more natural in this context, as it allows more complex reference domains.

In general, parameterization strategies search for ways to decompose Ω into subdomains, where each can be parameterized independently. This makes the strategy more flexible in order to represent more complex geometries with reduced parametric distortion. However, establishing matching parameterizations and continuity across the boundary surfaces of the subdomains is often more difficult to achieve. For instance, in the surface case, Ray et al. [165] propose a global parameterization method from input vector fields where the cuts of Ω (i.e., the topology of the base complex) emerge simultaneously from the global numerical optimization process. In this work, functions satisfying Laplace's Equation, i.e., harmonic functions as discussed above are used, to decompose Ω (if necessary) into subvolumes and establish a parameterization of each subvolume so that adjacent subvolumes have a matching parameterization where they connect while respecting interior material attributes in the parameterization.

1.2.3 Orthogonality

Two vector fields ∇u and ∇v are orthogonal, if the angle between two corresponding gradient vectors is $\pi/2$, i.e. $\langle \nabla u, \nabla v \rangle = 0$ everywhere on Ω. Orthogonality is important in the case of quadrilateral remeshing [196]. Two orthgonal vector fields result in rectangular quadrilaterals that behave better in numerical simulation than quadrilaterals that are more distored or close to being degenerate.

For instance, orthogonal vector fields can be established on a domain with right angle corners such as a cube. A 2D analog is shown in Figure 1.5. Let us call the sides of a discretized rectangle a, b, c and d. Side a and c are opposite and side b and c are opposite. In the following, $u = 0$ is assigned to a, $u = 1$ is assigned to c, $v = 0$ is assigned to b and $v = 1$ is assigned to d. Laplace's Equation is solved for u and v independently. The resulting vector fields ∇u and ∇v are orthogonal. Figure 1.6 shows orthogonality on more general domains in 2D. The left domain consists of right corners and therefore orthogonal vector fields can be established. Since the right domain in Figure 1.6 consists of a curved boundary, orthogonality cannot be established.

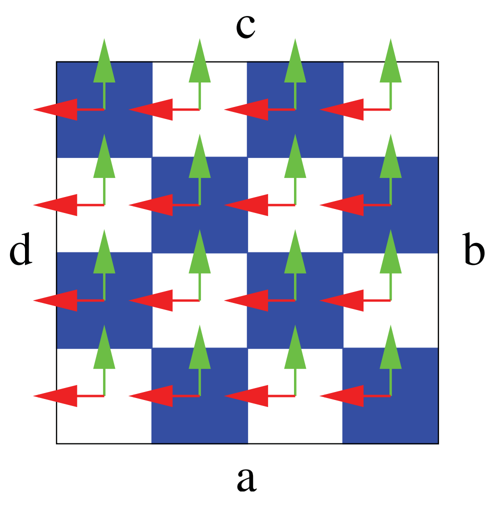


Figure 1.5. Orthogonal vector fields on a square domain.

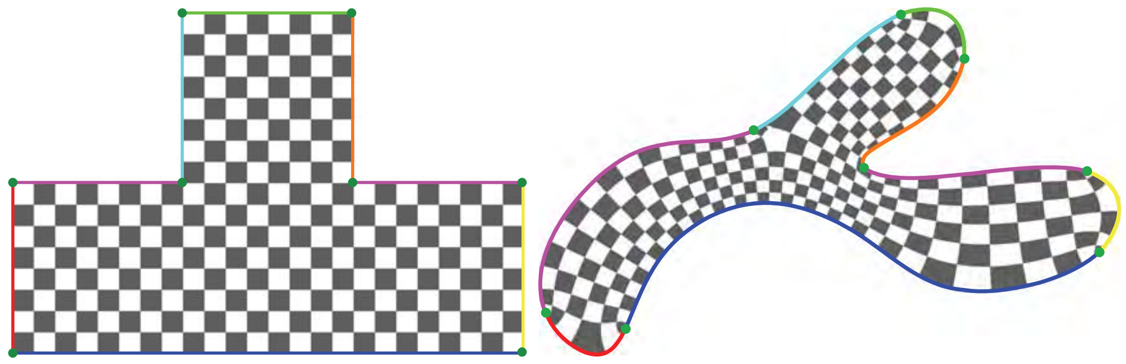


Figure 1.6. Parameterization on domains in 2D. On the left: Since the domain consists of right angle corners and straight lines, two orthogonal vector fields, resulting in a parameterization without any parametric distortion can be established. On the right: No orthogonal vector fields can be established, because the boundary of the domain is curved.

CHAPTER 2

RELATED WORK

This Section presents work that has been done in the fields of model parameterization (Section 2.1), isosurface visualization (Section 2.3), mesh generation and mesh quality (Section 2.2), relevant for this document.

2.1 Model Completion, Parameterization and Meshing

Parameterization is a hard problem for surfaces and even more so for volumes. In addition to use in modeling and remeshing, surface parameterization techniques have a wide variety of applications including texture mapping, detail transfer, fitting and morphing. For a more detailed description, please refer to the surveys [181, 61, 83]. Surface parameterizing techniques such as [117, 70, 196] deal with surface related issues and are not designed to be extended to model volumes. For instance, the authors in [6] motivate anisotropic remeshing and align mesh elements using the principal direction of curvature of the respective triangle mesh. Their approach yields a high quality quadrilateral mesh that has no relationship to the interior. If one were to offset the mesh in the normal direction, one would quickly get self intersections among the elements. Even if this can be avoided, eventually the hexahedral elements have degeneracies and eventually touch each other without proper alignment. Requiring parameterization of the interior makes this problem even more difficult since it is prone to self intersecting offsets and has to deal with skewed and twisted parameterizations.

The research on mesh generation is vast, and here, only the most relevant papers will be referenced. For fundamental algorithms and meshing methods, consult the survey [150]. In Chapter 6, a mixed element meshing methodology is proposed that is a combination of structured and unstructured mesh generation, based on a sampled midstructure of the object. The method is related to volumetric parameterization, unstructured meshing, midstructure methods, surface offsetting, and mixed element representations, all of which are reviewed briefly in the following.

Decomposing a domain of interest into a set of simpler subpatches involves "patch gluing" where a certain level of smoothness along the patch boundaries is desired. In [117], quadratic B-splines are generalized to fit arbitrary meshes creating triangular and rectangular surface patches. The lack of structure in the irregularities makes it clear that volumetric extensions do not immediately follow. Similarly, manifold splines [70] extend B-splines to surfaces of arbitrary topology, by modeling the domain of the surfaces with a manifold whose topology matches that of the polyhedral mesh. Then it embeds this domain into 2-space using a basis-function/control-point formulation. The domain of this technique is more complicated than the domain of a standard tensor product surface. As in [117], this approach also generates spline patches and glues them together by overlapping them, to get a "match" in the parameterization. The "glue" consists of mathematical operations such as control point constraints. In the case of a volume, patch boundaries are surfaces. Establishing smoothness and continuity between them is very difficult. In [144] the CubeCover algorithm is described which decomposes the volume enclosed by a triangle mesh into hexahedral elements. While the elements are of good quality, the structure depends on a user-defined metamesh. Therefore, in general, the output is an unstructured hexahedral mesh.

While being able to reconstruct some types of real world objects, the generalized cylinder(GC) [20] approaches and the approach by [90] are limited, because they require planar cross-sections. Note, representations that rely on planar cross-sections fail for objects with overhangs such as the femur in Figure 1.1. GC is a subclass of our method (Chapter 4) since we are able to model objects with overhangs as the femur in Figure 1.1. Furthermore, interpolation introduces oscillations on the B-spline, which is amplified when dealing with volumes, as in our case. And, for instance, in the method presented in Chapter 4, once a certain stage of the process is reached, the rest proceeds automatically without further user input.

Harmonic volumetric mappings between two solid objects with the same topology have been used in a variety of instances. Using related techniques, [202] and [115] are performing a 3D time-variant harmonic deformation from one volume to another volume with the same topology. In a diffeomorphic way, [202] applies this method to brain data.

Midstructures such as the medial axis and curve skeletons play an important role in mesh generation. For instance, Sheffer et al. [180] uses the embedded Voronoi graph of the object to decompose the object into simpler sweep-able subvolumes to automatically create a hexahedral mesh. Similarly, Armstrong et al. [10] shows that the medial axis of an object can be used to automatically identify features in mesh generation, dimensional reduction and detail removal. A computed simplified medial axis is then quadrangulated and extruded to construct a hex dominant mesh. Since the medial axis is part of the resulting mesh, it must have consistent topology and connectivity often requiring tedious manual efforts. The approach presented in Chapter 5 is also based on a midstructure. However, the midstructure can be represented by a set of points lying on it, significantly reducing the time to generate it. Furthermore, contrary to the approaches above, the method proposed here maintains the input surface representation in the resulting volumetric representation. In [170] a medial axis-based mesh generator is described. After the construction of a simplified medial axis, quad dominant meshes on the medial axis are generated and extruded to the boundary by advancing front schemes. In this case, advancing front schemes can cause inconsistency cases and for a more complex model, at least in part because it is difficult to control the front, e.g., the front may intersect itself resulting in degenerate elements. Furthermore, in addition to hexahedra, the resulting meshes contain prisms, pyramids and tetrahedrons.

In a similar approach Sheffer et al. [180] propose a method for automatic hexahedral meshing based on the embedded Voronoi graph, containing the full symbolic information of the Voronoi diagram and the medial axis of the object. It is used to decompose the object into sweepable subvolumes. Their approach is tested on CAD models with relatively simple medial axes. A more general model with a much more complex medial axis containing many small features (e.g., Figure 1.1) might result in many subvolumes thereby reducing the structured volume behaviour of the resulting mesh.

Polycube-Maps were originally proposed by Tarini et al. in [193] to remove seams in texture mapping by making the texture topology of the mesh compatible with the texture domain. Polycubes have been successfully used to construct manifold and polycube splines [73, 200] where the main challenge is to construct the polycube surface mesh for a given object. Wang et al. [201] propose a user-controllable framework where the user directly selects the corner points of the polycubes on the original 3D surface and based on that choice create the polycube maps applying discrete ricci flow. Polycubes have been suggested for volumetric parameterization [79, 69] but have not been presented for a variety of volume models including those that can contain interior material boundaries to which data fitting has to be applied to construct a representation for simulation.

Harmonic functions are holomorphic 1-forms as defined in Arbarello et al. [9] and were first introduced by Gu et al. [74] for use in surface parameterization. In this dissertation, harmonic functions are used in combination with appropriate midstructures to attain a parameterization that is consistent with respect to inner material boundaries in the parameterization. Harmonic functions have been shown useful in various applications, especially in the modeling and meshing community. Dong et al. [50] uses them to remesh triangle meshes into quadrilateral meshes with arbitrary topology where the harmonic functions are generated from user specified critical points on the triangle mesh. In [49], critical points are automatically determined by using Laplacian eigenfunctions defined on the surface. Tong et al. [196] automates the surface parameterization problem with discrete differential forms.

The aim of a volumetric parameterization method is to create as much interior structure as possible for a given input triangle mesh. The structure allows fitting smooth volumetric patches, e.g., trivariate B-splines to the geometry and simulation can be applied to the resulting volumetric representation. Volumetric parameterization is time consuming, especially because it can require significant manual user input. On the other hand, surface parameterization methods (e.g., see survey by [61] and the methods discussed above) are almost automatic and can produce well-shaped quadrilateral surface patches [49] and allow automatic B-spline fitting. A B-spline surface representation created this way can be used as input to the mixed element parameterization method introduced in Chapter 6, creating semistructured trivariate B-spline patches at the boundary of the object and unstructured tetrahedra in the inside. It requires minimal user input and so reduces modeling time significantly. The user can specify the depth of the trivariate boundary patches and has therefore the control of how much of the volume can be filled up with these higher-order elements. It will be demonstrated in the Chapter 6 that simulation scenarios exist where such representations yield stable and high-quality simulation results.

Mixed element methods, combining prisms and hexahedra with tetrahedra, have been used in various engineering scenarios such as aerospace applications [99], geophysics applications, computation fluid dynamics applications [89, 207], just to mention a few. Given a mixed element meshing approach, if the input surface is a quadrilateral mesh, pyramids are used to transition from boundary hexahedra to tetrahedra. If the input is a triangle mesh [96, 203], boundary prisms can be directly connected to interior tetrahedra. Another relevant mixed element approach is the H-morph [151]. Based on an advancing front scheme, an input tetrahedral mesh is iteratively converted into a hexahedral mesh. Tetrahedral elements which cannot be converted into hexahedral elements are connected to its adjacent hexahedral elements using pyramids.

In these cases, $C^{(0)}$ continuity between two element types is maintained. The approach presented in Chapter 6 does not enforce such continuity and in that respect is similar to, and motivated by approaches discussed in the introduction of Chapter 6.2.1, where the nodes defining the linear tetrahedra are linked to these higher-order elements. We show that the resulting representation yields convergence under mesh refinement. It will also be demonstrated that our mixed element approach has a similar convergence rate to a representation which uses only tetrahedral elements. However, as will be demonstrated in Chapter 6, the mixed element representation behaves more numerically stable in certain simulation scenarios.

In a similar vein to the approach presented in Chapter 6 is that by Wang et al. [96], presenting an approach to generate a mixed element mesh consisting of prismatic and tetrahedral elements. The mixed element representation is constructed from a triangulated surface representing the domain of interest. This triangulated surface is offset along a marching direction. A marching vector of a node is determined through local geometric constraints and a marching step size is chosen to reduce curvature of the previous marching surface. Similarly, [203] uses solutions to the Eikonal equation. Finally, tetrahedra are used to fill up the rest of the object. More recently, [76] first computes a distance field from an input triangle mesh. Based on a user specified thickness parameter, an isosurface of the distance field is extracted and a hexahedral shell mesh based on a polycube domain [193] and harmonic functions is constructed, i.e., the remaining volume in the object is void. Peng et al. [157] proposes a method to add volume textures to an input surface. The method offsets the input surface based on an ODE. The ODE is designed such that the limit offset surface is the medial axis of the input surface. Note, when the input surface contains sharp edges the medial axis touches the input surface. In this case volumetric element tend to have low quality and does not result in a thick shell as desired in our motivation.

In the approach presented in Chapter 6, we offset inward a higher-order surface representation of the object, to construct thick higher-order trivariate NURBS elements at the boundary and fill up the interior of the object with linear tetrahedra. We make use of harmonic functions in combination with a midstructure representation to define an offset function for the exterior surface. The use of harmonic functions has several advantages. Firstly, they are flexible as they allow the user to specify any appropriate midstructure, e.g., a point-sampled simplified medial axis or a 1D curve skeleton of the object or an isosurface as used in [76]. Secondly, marching directions and step sizes are implicitly determined and guarantee not to introduce degenerate elements due to the maximum principle of harmonic functions making them easier to control. Thirdly, the use of harmonic functions allow offsetting the input surface deeper into the interior of the object of interest it represents. These advantages make the offsetting approach more robust while also simplifying implementation. Note that offset in this context means to offset based on a harmonic gradient field directions.

2.2 Model and Mesh Quality

Mesh quality characterizes the geometric shape of the elements in a mesh representing some physical domain. Mesh generation methods (e.g., [186]) producing mesh representations for traditional finite elements focus on quality measures for finite element meshes. These measures were originally derived from mathematics that characterizes the approximating properties and power of $\mathcal{U}_F$ to a solution. Those properties depend on $\mathcal{C}$, the geometric mapping F, the particular partial differential equation that must be solved, and the boundary conditions. Those same conditions are embedded in model quality issues of a CAD model suitable for analysis, being the key for *analysis aware modeling* for IA, however, they are manifested differently. Basic mathematical principles used to define mesh quality for finite element analysis are reviewed in the following.

The typical map F for finite element analysis is a collection of mappings from the ideal element (triangle, tetrahedron or square, cube) to each element of the mesh (Figure 2.1), and $\mathcal{S}$ is typically the space of linear functions or bi- (tri-) linear functions over the ideal element. If F_i is the mapping from the ideal element to the $i-$th element, the space $\mathcal{U} = \bigoplus_i \mathcal{U}_{F_i}$ forms the approximating space, usually further constrained so that the approximating function space has only C^0 elements. Since each F_i is usually affine, $\mathcal{U}$ is a collection of local affine linear or bilinear maps.

Discretization error and stiffness matrix conditioning are two variables in mesh quality discussions and are therefore important issues for a successful analysis in the finite element method. The discretization error is the difference between the approximation computed by the finite element method and the true solution, in this case $u(x,y) - \widehat{u}(x,y)$, measured in an appropriate norm. Stiffness matrix conditioning affects the time and accuracy of analysis and depends on the shape of the elements. A mesh which results in a small condition number better performs using an iterative solver. However, as discussed in [182], this does not imply that the discretization error is small. For instance, for triangles it is known that small angles can cause poor conditioning. However, depending on the PDE, these triangles cause a smaller initial discretization error.

In traditional finite elements, it is well acknowledged that the initial mesh upon

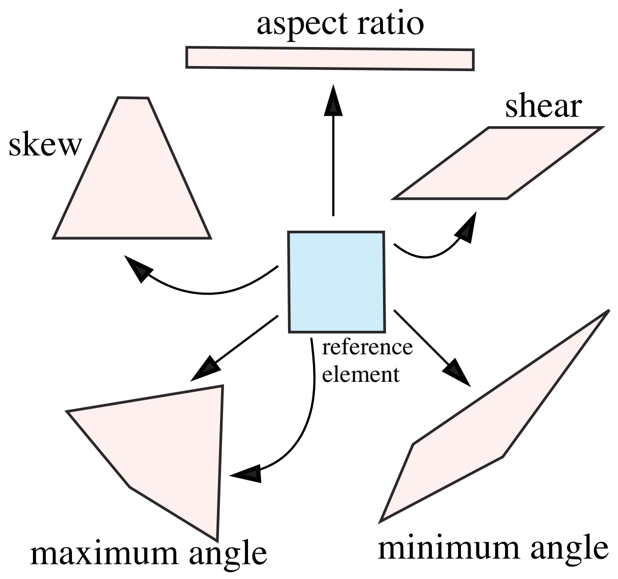


Figure 2.1. Examples for mesh quality metrics in traditional finite elements.

one runs one's simulation and refinement algorithms greatly impact the quality of the results. In the best case, the quality of the mesh impacts the constants that exist in the asymptotic error estimates and determine the level of refinement at which the asymptotic behavior begins. In the worst case, the mesh quality under adaptive refinement is a successively worsening ill-conditioned system which renders the computations useless for engineering practice. Thus, mesh quality is an important issue for finite element methods, when generated meshes will be used in engineering practice for analysis. Because of this reliance of analysis in practice on mesh quality, there is a large body of literature addressing the subject. In particular, the reader is referred to [154]. Given a mesh, the question has to be raised whether its quality is appropriate so that successful analysis can be applied to it. As in the case of B-Splines, h-refinement or knot insertion [37], respectively, do not improve the initial mesh quality. In order to improve the quality of the initial mesh, tremendous effort has been devoted to generate high quality meshes used in finite elements. Generally, schemes are used which relocate the vertex (or node) positions without altering the connectivity of the mesh.

Shewchuk in [182] gives bounds on the extreme eigenvalues for Poisson's equation. For instance, λ^t_{max}, which is the maximum eigenvalue of a local stiffness matrix for triangle t, can be bounded by

$$\frac{l_1^2 + l_2^2 + l_3^2}{8\,A} \leq \lambda_{max}^t \leq \frac{l_1^2 + l_2^2 + l_3^2}{4\,A}, \tag{2.1}$$

where A is the area of t, and l_i is the length of edge i of t. Note that the maximum eigenvalue of the stiffness matrix can be bounded in terms of the maximum λ_{max}^t and the maximum number of elements meeting at a single vertex [64]. This allows one to look at the elements t to make a conclusion about the global stiffness matrix conditioning.

Therefore, local smoothing techniques also called node-movement strategies are usually used to improve a mesh which relocates each vertex according some objective function to improve the mesh quality in the neighborhood of that vertex. The most commonly used smoothing technique is Laplacian smoothing [59] which moves the current vertex to the geometric center of its incident vertices. However, Laplacian smoothing does guarantee improvement in element quality. This approach is improved by using an objective function which moves the vertex based on a quality metric such as sign of the volume, aspect ratio, angle between adjacent edges (best 90 degrees and worst 180 degrees in case of quadrilateral meshes), stretching orientation, Oddy metric [147] and quality measures for tetrahedral meshes [156]. See Figure 2.1 which shows some of these quality metrics for quadrilaterals. These measures are used to describe the quality of an element. The reader is referred to [107] where these metrics are discussed and applied to different input meshes.

Quality metrics like orientation, volume, shape, length ratio and skew are embedded in the element's Jacobian matrix J, which is part of an affine map which maps a point from the reference element to its corresponding physical element (also see Figure 2.1), and can be accessed by applying the QR factorization of J. In the case of triangles, the reference element is the triangle defined by the vertices $(0, 0)$, $(1, 0)$ and $(0, 1)$). Then, linear maps like the Frobenius norm are used to convert J into a scalar which is then used to design a objective function optimizing certain quality metrics. The reader is referred to the work of Knupp [108] who develops a mathematical theory of geometric quality metrics applied to unstructured meshes. If a choice is made on the objective function, an optimization process is applied to minimize the objective function and therefore mesh is improved. Newton search [107] for instance is an efficient way to minimize the objective function.

Note that these method are proposed for a trilinear simulation and geometry basis. In the case of B-splines, it will be demonstrated that these quality measure are not sufficient and new and more appropriate metrics should be developed to describe the quality of the mapping F. Also, due to the tensor-product nature, modeling with NURBS has limitations. Sweeping [37] however, is a powerful technique to model a wide class of objects. To generate a swept volume, a NURBS surface is swept along a NURBS curve. Often, given a "sweepable" physical domain Ω, there are usually more ways for the sweep. Different choices result in degeneracies or more skewed elements at regions where this might be not desirable. As we will see on an 2D example in the results section, different choices affect the discretization error. They also affect the stability of linear systems and therefore eigenvalue problems.

2.3 Isosurface Visualization

Visualization techniques are used in numerous engineering fields–including medical imaging, geosciences, and mechanical engineering–to generate a two-dimensional view of a three-dimensional scalar or vector data set. Additionally, they can visualize simulation results (e.g., generated with the finite element method). Consequently, the development of such visualization algorithms has received much attention in the research community. Techniques usually fall into three groups: (1) direct volume rendering, (2) isosurface mesh extraction followed by isosurface mesh rendering, and (3) direct rendering of isosurfaces.

Techniques in category (1) typically involve significant computation, especially when dealing with arbitrary geometric topologies represented by high-order basis functions such as NURBS. In ray-based direct volume rendering methods (see [111, 127]), it is necessary to integrate each ray through the volume using sufficiently many integration steps. Each integration step requires an expensive root-solving due to the nonlinear mapping. Hua et al. [84] presented an algorithm to directly render attribute fields of tetrahedral-based trivariate simplex splines by integrating densities along the path of each ray corresponding to a pixel. In the case of uniform grid data sets, accumulating slices aligned along the viewing direction (see [205]) is efficient and commonly used in practice, even though ray-based techniques offer a range of optimizations (e.g., empty space skipping).

Methods in category (2) assume a regular grid of data and extract isosurfaces using Marching Cubes (MC) [119], resulting in a piecewise planar approximation of the isosurface. After isosurface mesh extraction, the faces of the isosurface mesh are rendered. Marching Tetrahedra (MT) [33] is applied to both structured and unstructured tetrahedra-based grids. In both MC and MT, the corners of a hexahedral or tetrahedral element, respectively, are used to determine if the isosurface passes through the respective element. Then, the intersections between the element's edges and the isosurface are determined to create piecewise linear facets approximating the isosurface. Although these approaches are efficient and therefore widely used in practice, they approximate the isosurface by piecewise linear facets within an element with some ambiguity, and therefore do not guarantee topological correctness. As an example, Figure 2.2 shows a discretized domain with 300 000 linear tetrahedra. As will be shown in Chapter 10, this domain can be represented with a single triquintic NURBS element. As seen in Figure 2.2a, the respective isosurface extracted with MT has ambiguities in the topology, resulting from data that are known only at the corners of the elements and hence can miss isosurface features. Furthermore, the time to construct the respective

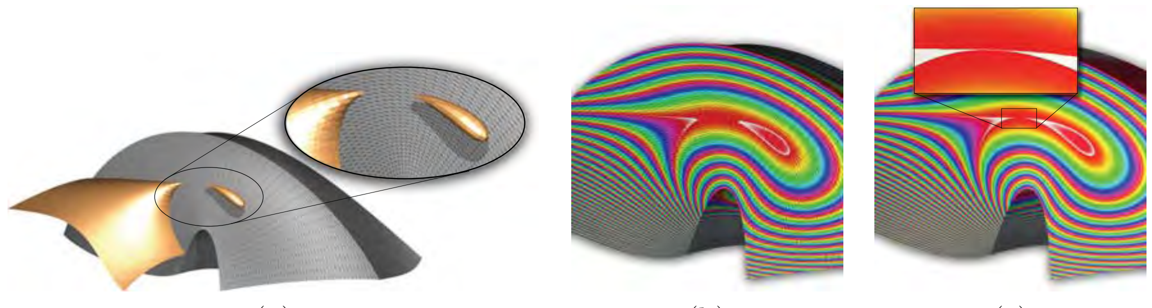

(a) (b) (c)

Figure 2.2. Discretization of domain from Figure 10.1c with 300k tetrahedra and application of marching tetrahedra (using ParaView). (a) isosurface; (b) scalar field on tetrahedra; (c) our approach on single triquintic NURBS patch mesh representation can be computationally laborious. Schreiner et al. [173] propose an advancing-front method for constructing manifold isosurfaces with well-shaped triangles (Figure 2.3), although it has some difficulties when the front meets itself (the stitching problem). Meyer et al. [130] propose a particle system on high-order finite element mesh (arbitrary geometric topology), which applies surface reconstruction on the particles to construct the isosurface mesh; however, the visualization produced is not a water-tight surface. When the data are known only at the corners of a hexahedral mesh, our method constructs an approximation by filtering the data with a high-order approximating or interpolating trivariate B-spline filter (see [121]). The filter can be trilinear (only $C^{(0)}$), tricubic ($C^{(2)}$), or higher degree, as required by the user. Then, an isosurface of the high-order approximation is directly rendered with pixel accuracy.

In category (3), the isosurface is rendered directly, i.e., for every pixel on the image plane, its corresponding point on the isosurface is determined (Figure 2.3, left). Once the point on the isosurface for a given pixel is known, the pixel can be shaded using the gradient as the normal for the given point. Another motivation to visualize specific isosurfaces is to color-code information, such as material density, to get a better

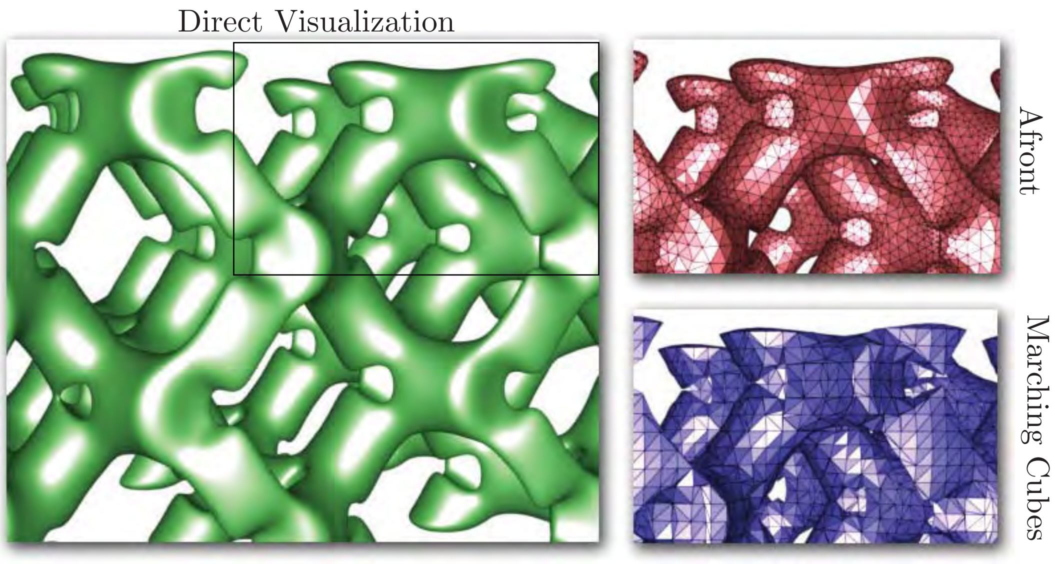


Figure 2.3. Isosurface from silicium data set (volvis.org), isovalue of 130 using Marching Cubes (using ParaView), Afront (ρ = 0.3) and Direct visualization with our proposed method.

understanding through which materials the isosurface passes. Knoll et al. [106] use a trilinear reconstruction filter on a structured grid and a ray-based octree approach to render isosurfaces and achieve interactive frame rates. Nelson et al. [142] propose a ray-based isosurface-rendering algorithm for high-order finite elements using classic root-finding methods, but do not consider element curvature (i.e., the multiple entry and exit problem). Kloetzli et al. [104] construct a set of structured Bézier tetrahedra from a uniform grid to approximate any reconstruction filter with arbitrary footprint. Given this reconstruction, generated from gridded input data (e.g., medical or simulation data), they directly visualize isosurfaces using the ray/isosurface intersection method presented by Loop [118].

The method proposed in Chapter 10 is most closely related to class (3) approaches, i.e., our proposed method directly visualizes an isosurface from a trivariate NURBS of arbitrary geometric complexity. However, instead of following only a ray-based scheme, our approach computes the intersection between a ray frustum and the isosurface. Furthermore, it is often desired to visualize the geometry represented by the NURBS. While approaches similar to the work in [2] can be used to render the object-surface geometry, our approach can be used to simultaneously visualize both the geometry represented by the NURBS and the visualization of isosurfaces of the attribute representation in a robust way. Intersecting a ray frustum with an object in the scene is related to the approaches that propose cone-tracing given in the work [7] and beam-tracing (see [80]) for more efficient anti-aliasing, soft shadows, and reflections. However, both of those techniques deal only with polygonal objects. For isosurfaces of algebraic functions, the dissertation [47] presents interval approaches to create intersection tests in the ray-tracing of implicit surfaces. In particular, it shows a ray sampling-based method to exploit the coherence of rays to accelerate the process of ray-tracing implicit surfaces, which can also be used for anti-aliasing isosurface silhouettes.

CHAPTER 3

ANALYSIS-AWARE MODELING

IA [87] has been proposed as a methodology for bridging the gap between Computer Aided Design (CAD) and Finite Element Analysis (FEA). Although both the traditional and isogeometric pipelines rely upon the same conceptualization to solid model steps, they drastically differ in how they bring the solid model both to and through the analysis process. The IA process circumvents many of the meshing pitfalls experienced by the traditional pipeline by working directly within the approximation spaces used by the model representation. This chapter demonstrates that in a similar way to how mesh quality is used in traditional FEA to help characterize the impact of the mesh on analysis, an analogous concept of *model quality* exists within IA. The consequence of these observations is the need for a new area within modeling – *analysis-aware modeling* – in which model properties and parameters are selected to facilitate IA.

The concept of IA was first introduced by Hughes et al. in [87] as a means of bridging the gap between Computer Aided Engineering (CAE), including Finite Element Analysis (FEA), and Computer Aided Design (CAD). Those familiar with the application of FEA to CAD-based models are well-aware of the complications and frustrations which arise when one attempts to take a "solid model" (a term which we will define in the context of the modeling community in Section 3.3) as produced by a typical commercially available CAD system, generate a surface tessellation and corresponding volumetric representation in terms of meshing elements (e.g., triangles and quadrilaterals on the surfaces and tetahedra and hexahedra in the volume), and run an analysis. On this "preprocessing" side prior to the actually analysis step, a large amount of effort is expended in mesh generation and optimization (in terms of mesh quality), sometimes to the point of consuming more time than what is taken by the actual analysis step. Once an analysis is run, solution refinement often requires mesh adaptation or in worst case regeneration, both of which require consultation with the original CAD-model. IA claims to break this common but insidious cycle by choosing an alternative route from the solid geometric model to analysis. In IA, one works with the functions used to generate the model directly by using the function space used for model generation as the approximating space in which field solutions are built (hence the name *iso*-geometric).

By circumventing many of the pitfalls that one encounters during the mesh generation process by working directly with the solid model, IA effectively eliminates the geometric error component of the analysis pipeline. Geometric refinement is no longer necessary; the analyst can focus attention solely on solution refinement. It is our thesis that although circumventing the mesh generation pipeline implies that one no longer needs to consider *mesh quality*, there are still issues of *model quality* that must be considered. In a similar way to how mesh quality is a geometric means of assessing the impact of a mesh on the function space which it induces in the classic finite element process, *model quality* is a characterization of those properties of the representation of the model geometry that impact the representation space (or trial space) used to approximate the fields of interest. The consequence of these observations is the need for a new area within modeling – *analysis-aware modeling* – in which model properties and parameters facilitate IA.

3.1 Nomenclature

In this section we set up the environment for considering IA in the context of linear second-order partial differential equations (PDEs) with zero Dirichlet boundary conditions. We note that there exists a straightforward extension to nonzero boundary conditions and also to Neumann boundary conditions. The issues we raise are quite general and will arise in using the isogeometric concept in solving many types of partial differential equations: Some examples we will consider and upon which we will comment in the examples section will have nonzero boundary conditions and/or more complex partial differential operators (such as those found in the modeling of linear elasticity).

We do, however, set up here the nomenclature to illustrate these specific problems in an arbitrary number of space dimensions.

Although some of these terms may appear obvious to either those familiar with geometric modeling or those familiar with engineering analysis, we believe it is important for both the geometric modeling and finite element analysis communities to be overtly explicit during this time of confluence of ideas.

Let $\Omega \subset \mathbb{R}^s$ with $s \in \mathbb{N}$ be a bounded domain with boundary $\partial\Omega$. Ω is the physical domain, often called the *world space* or *physical space*.

Using the notation $D_j = D_j^1$ to denote the partial derivative with respect to the j-th variable, define

$$D^\alpha := D_1^{\alpha_1} \cdots D_s^{\alpha_s}, \tag{3.1}$$

a mixed partial derivative of total order $|\alpha| = \alpha_1 + \cdots + \alpha_s$. Then the column vector

$$\nabla f = [D_1 f, \ldots, D_s f]^T$$

denotes the gradient of f. Further, let

$$H^1 = H^1(\Omega) := \{f : \Omega \to \mathbb{R} : D^\alpha f \in L_2(\Omega),\ |\alpha| \leq 1\} \tag{3.2}$$

$$V = H_0^1 = H_0^1(\Omega) := \{f \in H^1(\Omega) : f = 0 \text{ on } \partial\Omega\}, \tag{3.3}$$

denote the Sobolev spaces of functions with values and first order partial derivatives in $L_2 = L_2(\Omega)$.

Let $\{\phi_i\}_{i=1}^n \subset H^1(\Omega)$ be linearly independent functions in $H^1(\Omega)$. Moreover we define

$$J := \{j \in \{1, \ldots, n\} : \phi_j \in V\} \text{ and } |J| = \#\{j : j \in J\},$$

that is, the set of indices of those ϕ_j that vanish on $\partial\Omega$. If $s = 1$ then typically $J = \{2, \ldots, n-1\}$. We define the space

$$V_h^q := \{\sum_{j \in J} \boldsymbol{c}_j \phi_j : \boldsymbol{c}_j \in \mathbb{R}^q,\ j \in J,\ q \in \mathbb{N}\}, \tag{3.4}$$

and note that $V_h = V_h^1$ is a subspace of V. The index h is a flag indicating finite dimensionality and is often a measure of element diameter. The space V_h^q forms the space in which the approximation to the solution of the differential equation is made.

Suppose $\{\psi_j\}_{j=1}^n$ is a set of real-valued linearly independent functions on a partition of the unit cube $\Theta = [0,1]^s$ in $\mathbb{R}^s$, and the functions ϕ_j are given as

$$\phi_j(\boldsymbol{x}) = \psi_j \circ \boldsymbol{F}^{-1}(\boldsymbol{x}),\ j = 1, \ldots, n, \tag{3.5}$$

where $\boldsymbol{F} = (F_1, \ldots, F_s) : \Theta \to \Omega$ is a bijection. Figure 3.1 illustrates a modification between the shape of a ψ and its corresponding ϕ induced by $\boldsymbol{F}^{-1}$.

Moreover, we assume that

$$\begin{gathered} \boldsymbol{F}(\Theta^o) \subset \Omega^o \\ \sigma := \boldsymbol{F}|_{\partial\Theta} : \partial\Theta \to \partial\Omega, \end{gathered} \tag{3.6}$$

i.e., $\boldsymbol{F}$ maps interior to interior and boundary to boundary. If we use the same functions ψ_j to define both the ϕ_j's and $\boldsymbol{F}$

$$\boldsymbol{F} = \sum_{j=1}^n \boldsymbol{\gamma}_j \psi_j,\ \text{for some } \boldsymbol{\gamma}_j \in \mathbb{R}^s,\ j = 1, \ldots, n, \tag{3.7}$$

then the approach to solving the partial differential equation is called *isogeometric*.

Typically analysis will be carried out on models whose definition is piecewise over multiple hypercubes $\Theta = \{\Theta^i = [0,1]^s : i = 1, \ldots, K\}$ for some finite $K \in \mathbb{N}$, and $\boldsymbol{F}$ is defined piecewise in terms of the mappings from each Θ^i such that, for all i, j, $i \neq j$,

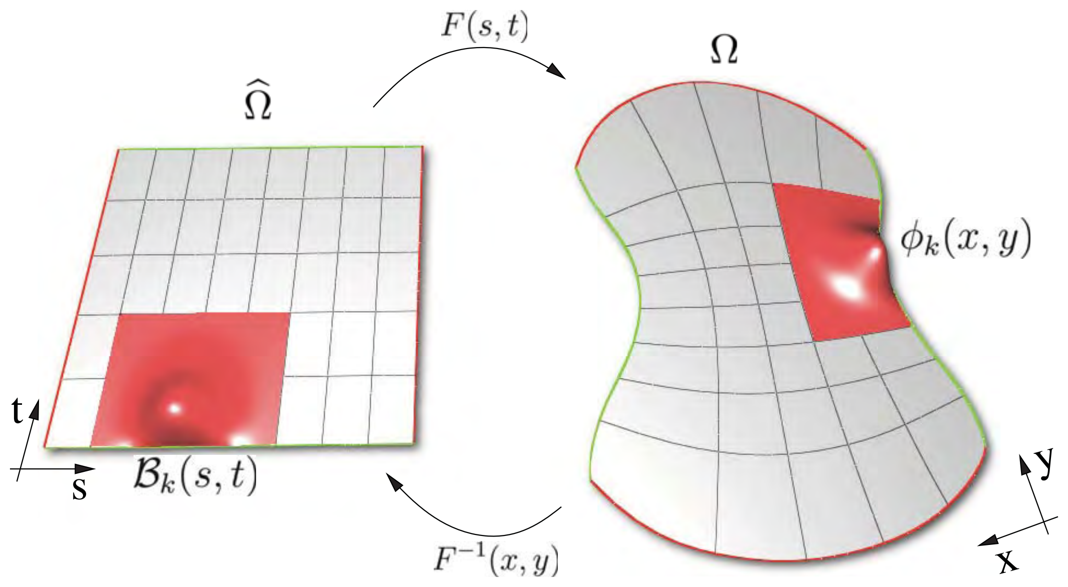


Figure 3.1. F maps a point from the reference domain Θ to the physical domain Ω. Correspondingly, F^{-1} maps a point from Ω back to Θ. With F^{-1}, it is possible to define basis functions on Ω as compositions of F^{-1} with basis function defined on Θ.

$$\boldsymbol{F}(\partial\Theta^i)\bigcap\boldsymbol{F}(\partial\Theta^j) \subset \partial\boldsymbol{F}(\Theta^i)\bigcap\partial\boldsymbol{F}(\Theta^j).$$

Further $\boldsymbol{F}$ must be continuous on every nonempty intersection. Θ is called the *parametric domain* or the *reference space*.

CAD systems typically create representations of the model that define mappings σ from $\partial\Theta$ to $\partial\Omega$, and can be written as a collection of mappings $\sigma_i : [0,1]^{s-1} \to \mathbb{R}^s, i = 1, 2 \ldots, 2s$ that agree on their shared boundaries. When $s = 3$, this representation is sometimes suitable for performing isogeometric shell analysis, but in order to perform a full volumetric analysis, the model must be *completed*. That is, the representation must be extended to completely define the interior so that V_h^q is defined. We will discuss this further in what follows as this is often a nontrivial process.

Typically, ψ's are tensor product B-splines, Non-Uniform Rational B-Splines, rectangular subdivision surfaces or T-splines.

3.2 Outline

This chapter is structured as follows. In Section 3.3 we present the modeling to analysis pipeline. We briefly describe the process of going from model conceptualization to the *solid* model, and then distinguish between the classic route of surface and volumetric mesh generation as done for classic finite element analysis from the boundary representation and volumetric model generation as done for IA. In Section 3.4 we provide the mathematical and algorithmic descriptions of the isogeometric methodology employed in this work. In Section 3.5 we take a step back so as to appreciate the parametric modeling of geometry from the perspective of, and in the language of, isogeometric analysis. By doing so, we hope to demonstrate that in general there is not "a model" (a single ideal model) on which one does IA, but rather that designers are presented with a collection of modeling choices – some of which may inadvertently impact analysis. In Section 3.6 we present one-, two- and three-dimensional examples comparing different model completions and demonstrating the impact of model completion on quality of the solutions one obtains from an IA. We conclude in Section 3.7 with a summary of this work, some conclusions that we can draw and proposals of future work.

3.3 The Modeling to Analysis Pipeline

In this section, we review the classic model to mesh to analysis pipeline as appreciated by most FEA researchers, and then provide the corresponding modification to the pipeline as introduced by IA. Note that we pay particular attention to the use of nomenclature in this section, as the confluence of concepts from two fields (modeling and analysis) has led to misconceptions in both fields as to what is being discussed. We will use Figure 3.2 as our visual guide through this process.

3.3.1 Conceptualization to *Solid* Model

The stages of the pipeline from conceptualization to *solid* model are denoted by the left half of Figure 3.2. The designer has in mind a *concept* or *ideal* of what is to be designed, and uses a CAD modeling system to construct a collection of surfaces that are meant to represent the outer boundary of the object of interest. Note that the modeler is not working with three-dimensional manifold representations, but rather is working

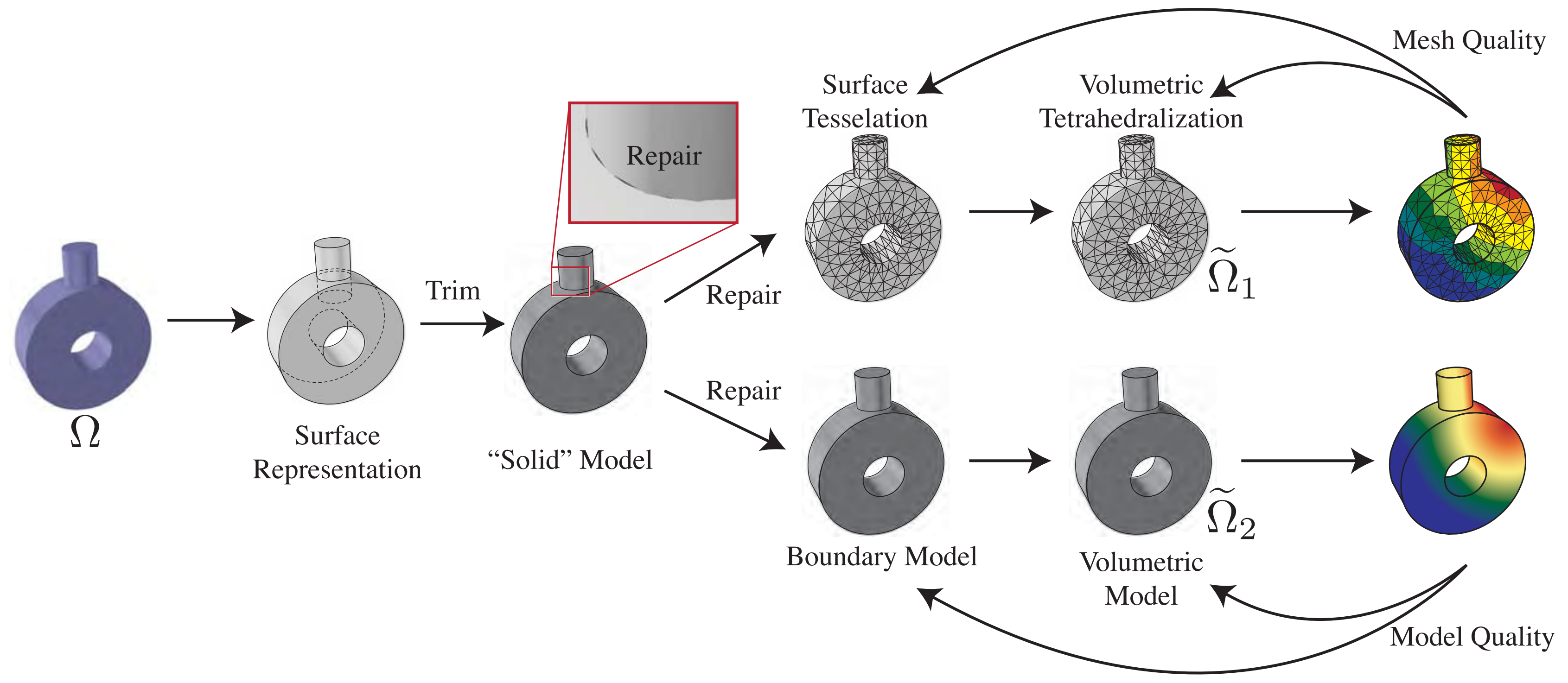


Figure 3.2. Diagram presenting the classic concept-to-mesh pipeline (top branch) and the concept-to-model pipeline (bottom branch). A detailed discussion of the diagram is presented in the text.

with surface subregions that are intended to bound the object of interest. These surfaces are often constructed one-by-one without regard for how they will connect, intersect or overlap with other pieces. The modeler then uses the CAD system to accomplish what is referred to as *trimming*, an attempt to connect (or stitch) the surfaces together to form a *watertight* model, that is, one that clearly delineates $\mathbb{R}$ into three regions, inside the model, outside the model, and on the boundary of the model. Within the shape modeling community, the term *solid model* is used to characterize such a representation. When the solid model is represented using pieces of bounding 2-manifolds, the representation is called a *boundary representation* or *b-rep.* Whereas the surface representation (pretrimming) does not necessarily faithfully represent the geometric and topological properties (such as being a water-tight surface) on the conceptual object, the newly formed solid model should. At the conclusion of this process, a solid model is output from the CAD system. Although called a solid, it is a collection of pieces of surfaces and connectivity information that are intended to bound a three-dimensional object and that is intended to be watertight.

Unfortunately the intersections between sculptured surface pieces that define the curves along which the pieces should be trimmed and stitched together cannot be exactly represented in the parameter spaces of the defining surfaces, but rather are defined implicitly. Hence, while CAD systems have different approaches to explicitly representing these curves, the trimmed surfaces and resulting b-rep models are all approximated along the trimming edges. Frequently it may be necessary to *repair* the model to make it suitable for later processes such as analysis or fabrication.

3.3.2 Traditional Meshing Pipeline Leading to Analysis

The traditional meshing pipeline leading to analysis is denoted by the upper branch of the right half of Figure 3.2. In the figure, we have purposefully placed quotations around solid to draw the reader's attention to two things. First, as previously mentioned, the solid model is not solid (in the sense of the term as used by analysts), but rather denotes the boundary of the object. Secondly, the solid model as produced by CAD systems is not always truly water-tight, but possibly only visually water-tight.

This issue has been the bane of many surface tessellation efforts which have devised schemes under the assumption that the solid model formed a mathematically water-tight (i.e., *closed in the topological sense*) representation. In going from the solid model to a surface tessellation, it is often necessary to invoke a repairing procedure. We mark the repair process as being the point of deviation between the traditional pipeline and IA as many of the repair procedures used in traditional mesh generation assume that the target representation is a piecewise linear tessellation. The result of the repair and surface generation process is a tessellation of boundary of the object of interest. This tessellation is an approximation to the *true* geometry (in this case, the CAD-model), where approximation decisions have been made both in terms of how repairs are done and in how finely the tessellation captures the features of the original model. If a three-dimensional analysis is desired, the next step is to generate a volumetric tessellation, normally by filling in the volume with elements of the appropriate type (hexahedra or tetrahedra) for the analysis of interest.

In the classic finite element procedure, one generates a tessellation which approximates the true geometry, and then uses this tessellation to induce a function space in which approximations will be made. For instance, in classic linear finite elements over triangles, the triangular tessellation induces a piecewise linear (in total degree) space which is C^0 continuous along the edges of the triangles. As is well known by finite element practitioners, given two tessellations, both of which faithfully represent the geometry, one can get drastically different solutions due to the properties (or richness) of the approximating space that is induced.

A natural feedback loop developed between analysts and mesh generation experts concerning the impact of meshes on solution quality. These metrics have commonly become known as *mesh quality metrics*. That is, they are geometric considerations (normally involving things like ratios of angles of elements, aspect ratios of edges, *etc.*) which help guide the development of meshes appropriate for analysis. Although these metrics have deep foundations within approximation theory, they are often abstracted away so that only geometric qualities of the mesh are discussed. We remind the reader, however, that maximizing mesh quality is, in its essence, an attempt to positively shape the approximating function space induced by the mesh.

3.3.3 Isogeometric Pipeline Leading to Analysis

While IA is still a young field, the authors hypothesize that isogeometric pipeline leading to analysis is denoted by the lower branch of the right half of Figure 3.2. As in the case of the traditional pipeline, repair is needed to ensure that the model being used for analysis meets the required topological constraints (such as closure) of the problem. In the case of IA, however, this repair process must be done keeping in mind the original and target representations. A starting point for isogeometric discussions in line with the finite element approaches is the boundary model, which should be a geometrically and topologically correct model of the bounding surfaces of the object. If a three-dimensional analysis is desired, volumetric representations must be generated prior to the analysis. The approximating space generated during an IA is dependent upon the boundary model (in 2D) or volumetric model (in 3D) that is used. Just as in the case of classic mesh generation, two different volumetric models generated from the same boundary model will create two different approximating spaces. Analogous to mesh quality impacting analysis, *model quality* impacts IA.

A different starting point for IA is that consideration during the shape (usually boundary) modeling process should be given to create a representation that lends itself to IA. There are frequently many modeling operations that lead to different representations for either the exactly same or closely related boundary shapes. Some of those representations are better suited to analysis than others, and within those groups, some are better suited to certain types of analysis than others. Similar issues have been recognized in created representations for models suitable for the multitude of computer-aided manufacturing processes and techniques. However, progress has been made in developing CAD systems that develop representations that, while suitable for design and display, are *fabrication-aware*, thus enabling a smoother, faster transition between design and fabrication.

Analysis-aware modeling in the context of IA may prove to be a key step towards that progression for design, engineering analysis, and simulation. Towards that end, this chapter raises several important issues through a combination of analysis and demonstration in which the interaction between representation and analysis can either enhance or make the product evolution process difficult. Until such time as these issues have been quantified and embedded in analysis-aware modeling systems, the human modeler must be mindful of them.

3.4 Mathematical Formulation

In this section, we first review the basic mathematical representational building blocks on which IA as well as many CAD and geometric modeling systems represent geometry. An overview of NURBS (Non-Uniform Rational B-Splines) can be found in [37]. All computational algorithms are presented there, so in the following section we discuss definitions of B-spline and NURBS functions and their combinations to define parametric mappings of global geometry. Note that this discussion provides the mathematical building blocks of modeling, but does not address how these building blocks are assembled as part of the modeling process. We will delve into the mind of the modeler in a subsequent section (Section 3.5).

3.4.1 The Framework

Let $\Omega \subset \mathbb{R}^s$ with $s \in \mathbb{N}$ be a bounded domain with boundary $\partial\Omega$. The symbols D^α, H^1, H_0^1, and ∇ are defined as in Equations (3.1), (3.2), and (3.3), respectively. These Sobolev spaces have a norm given by

$$\|u\|_1^2 = \int_\Omega \left(u(x)^2 + \nabla u(x)^T \nabla u(x)\right) dx.$$

Let $a : H_0^1 \times H_0^1 \to \mathbb{R}$ denote the bilinear form

$$a(u, v) := \int_\Omega \nabla u(x)^T \nabla v(x) dx, \tag{3.8}$$

This bilinear form is positive definite on H_0^1, and H_0^1 is a Hilbert space with inner product $a(u, v)$ and associated norm $\|u\| = \sqrt{a(u, u)}$. We let

$$(u, v) := \int_\Omega u(x)v(x)dx, \quad u, v \in L_2(\Omega),$$

be the usual L_2 inner product. For a general review of Sobolev spaces we refer the reader to [175].

Discussions in this chapter will mostly be based on problems that arise in the relationship between geometry and analysis models. Most studies are focused on two prototypical mathematical model problems that arise in analysis, the Poisson problem and a corresponding eigenvalue problem, respectively.

$$-\nabla^2 u = f \text{ on } \Omega, \qquad u = 0 \text{ on } \partial\Omega, \tag{3.9}$$

$$-\nabla^2 u = \lambda u \text{ on } \Omega, \quad u = 0 \text{ on } \partial\Omega. \tag{3.10}$$

Given $f \in L_2(\Omega)$ the weak form of (3.9) is to find $u \in V := H_0^1(\Omega)$ such that

$$a(u, v) = (f, v), \quad v \in V \quad . \tag{3.11}$$

It is well known that (3.11) has a unique solution u, see [32].

The weak form of (3.10) is to find $\lambda \in \mathbb{R}$ and a nonzero $u \in V$ such that

$$a(u, v) = \lambda(u, v), \quad v \in V \quad . \tag{3.12}$$

Since this involves finding the eigenvalues and eigenfunctions of a compact, symmetric operator in the Hilbert space $(V, a(\cdot, \cdot))$ there exists an increasing sequence of strictly positive eigenvalues

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_k \leq \cdots$$

with $\lim \lambda_k = \infty$ and associated eigenfunctions u_k, which can be orthogonalized so that

$$a(u_j, u_k) = \lambda_k \delta_{j,k}, \quad j, k \geq 1.$$

Moreover, the eigenfunctions form a complete system, *i.e.,*the set of all linear combinations is dense both in V and $L_2(\Omega)$, again see [32].

For V_h^q defined as in Equation 3.4, the *Galerkin method* for (3.11) consists in finding $u_h \in V_h^q$ such that

$$a(u_h, v_h) = (f, v_h), \; v_h \in V_h.$$

Writing $u_h = \sum_{j \in J} c_j \phi_j$, we obtain a linear system for the unknown coefficients $\boldsymbol{c}$.

$$\boldsymbol{S}\boldsymbol{c} = \boldsymbol{f}, \quad \boldsymbol{S} = \big(a(\phi_i, \phi_j)\big)_{i,j \in J}, \quad \boldsymbol{f} = \big((f, \phi_i)\big)_{i \in J}. \tag{3.13}$$

Since the ϕ's are linearly independent and vanish on $\partial\Omega$ the *stiffness matrix* $\boldsymbol{S}$ is a symmetric positive definite matrix and (3.13) has a unique solution that is amendable to iterative methods like conjugate gradient [12].

The *Rayleigh-Ritz method* for (3.12) consists of finding λ and a nonzero $u_h \in V_h^q$ such that

$$a(u_h, v_h) = \lambda(u_h, v_h), \; v_h \in V_h.$$

Writing $u_h = \sum_{j \in J} c_j \phi_j$ as before we obtain a generalized eigenvalue problem

$$\boldsymbol{S}\boldsymbol{c} = \lambda \boldsymbol{M}\boldsymbol{c}, \quad \boldsymbol{S} = \big(a(\phi_i, \phi_j)\big)_{i,j \in J}, \quad \boldsymbol{M} = \big((\phi_i, \phi_j)\big)_{i,j \in J}, \tag{3.14}$$

where both $\boldsymbol{S}$ and the *mass matrix* $\boldsymbol{M}$ are positive definite.

Thus the eigenvalues λ_{kh} of (3.14) that approximate the exact eigenvalues of (3.12) are positive

$$0 < \lambda_{1h} \leq \cdots \leq \lambda_{|J|h}$$

and the eigenfunctions u_h can be chosen to be orthogonalized so that

$$a(u_{jh}, u_{kh}) = \lambda_{kh}\delta_{j,k}.$$

Moreover $\lim_{h\to 0} \lambda_{kh} = \lambda_k$ for $1 \leq k \leq |J|$, provided $\lim_{h\to 0} \inf_{v_h \in V_h} \|u_k - v_h\| = 0$ for $1 \leq k \leq |J|$.

3.4.2 Definition of Isogeometric Finite Element Analysis

Suppose the basis functions $\phi_j(x)$ are given as in Equation (3.5). Moreover, we assume that (3.6) holds, i.e., $\boldsymbol{F}$ maps interior to interior and boundary to boundary, and that $\boldsymbol{F}$ is defined as in Equation (3.7). Then the Galerkin and Rayleigh-Ritz methods for (3.11) or (3.12) are called *isogeometric*.

The elements s_{ij} of the stiffness matrix $\boldsymbol{S}$ can be expressed in terms of the gradients of the ψ_j basis functions. Let

$$\boldsymbol{J} = \boldsymbol{J_F} := \begin{bmatrix} D_1F_1 & \cdots & D_sF_1 \\ \vdots & & \vdots \\ D_1F_s & \cdots & D_sF_s \end{bmatrix} = \begin{bmatrix} \nabla F_1^T \\ \vdots \\ \nabla F_s^T \end{bmatrix} \tag{3.15}$$

be the Jacobian of $\boldsymbol{F}$. Note that the elements of $\boldsymbol{J}$ are functions defined on Θ. We assume that $\boldsymbol{J}(t)$ is nonsingular for all $t \in \Theta$. Then

$$s_{ij} = \int_\Omega \nabla\phi_i(x)^T \nabla\phi_j(x)dx = \int_\Theta \nabla\psi_i(t)^T \boldsymbol{N}(t) \nabla\psi_j(t)dt, \quad i, j = 1, \ldots, n, \tag{3.16}$$

where

$$\boldsymbol{N} = |\det(\boldsymbol{J})|\boldsymbol{J}^{-T}\boldsymbol{J}^{-1}. \tag{3.17}$$

Note that $\boldsymbol{N}(t)$ is positive definite for all $t \in \Theta$. Explicitly, for $s = 1$,

$$N(t) = \frac{1}{|F'(t)|},$$

and for $s = 2$

$$N = \frac{1}{|\det(\boldsymbol{J})|}\begin{bmatrix} \|\nabla F_2\|_2^2 & -\nabla F_1^T F_2 \\ -\nabla F_1^T F_2 & \|\nabla F_1\|_2^2 \end{bmatrix}.$$

If

$$\boldsymbol{K} := |\det(\boldsymbol{J})|^{-1/2}\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \quad \boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_s),\ \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I},$$

with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_s > 0$ is the singular value decomposition of $\boldsymbol{K}$ then

$$\boldsymbol{N} = \boldsymbol{U}\boldsymbol{\Sigma}^{-2}\boldsymbol{U}^T$$

is the spectral decomposition of $\boldsymbol{N}$. Since $\boldsymbol{N}$ is positive definite the eigenvalues of $\boldsymbol{N}$ are the inverse square of the singular values of $\boldsymbol{K}$ and the orthonormal eigenvectors of $\boldsymbol{N}$ are the right singular vectors of $\boldsymbol{K}$.

3.4.3 B-splines

For integers $n \geq 1$ and $d \geq 0$ let $\boldsymbol{\tau} = \{\tau_i\}$ be a nondecreasing finite sequence of real numbers. We refer to $\boldsymbol{\tau}$ as a *knot vector* and its components as *knots*. On $\boldsymbol{\tau}$ we can recursively define degree d B-splines $B_{j,d} = B_{j,d,\boldsymbol{\tau}} : \mathbb{R} \to \mathbb{R}$ by

$$B_{j,d}(t) = \frac{t - \tau_j}{\tau_{j+d} - \tau_j}B_{j,d-1}(t) + \frac{\tau_{j+d+1} - t}{\tau_{j+d+1} - \tau_{j+1}}B_{j+1,d-1}(t)\ , \quad t \in \mathbb{R}, \tag{3.18}$$

starting with

$$B_{j,0}(t) = \begin{cases} 1, & \text{if } \tau_j \leq t < \tau_{j+1}, \\ 0, & \text{otherwise.} \end{cases}$$

Here we use the convention that terms with zero denominator are defined to be zero. We let $\mathbb{B}_{d,\boldsymbol{\tau}} = \{B_{j,d,\boldsymbol{\tau}}\}_j$.

A B-spline $B_{j,d}$ of degree d has the following properties:

1. It depends only on knots $\tau_j, \ldots, \tau_{j+d+1}$ and is identically zero if $\tau_{j+d+1} = \tau_j$.

2. For $t \in (\tau_j, \tau_{j+d+1})$, $0 < B_{j,d}(t) \leq 1$ and $B_{j,d}(t) = 0$ otherwise. The interval $[\tau_j, \tau_{j+d+1}]$ is called the *support* of $B_{j,d}$.

3. Its derivative is

$$DB_{j,d}(t) = d \left(\frac{B_{j,d-1}(t)}{\tau_{j+d} - \tau_j} - \frac{B_{j+1,d-1}(t)}{\tau_{j+d+1} - \tau_{j+1}} \right),$$

   again with the convention that terms with 0 denominator are set to 0.

4. If m of the $\tau_j, \ldots, \tau_{j+d_1}$ are equal to one value z, then $D^r B_{j,d}$ is continuous at z for $r = 0, \ldots, d - m$ and $D^{d-m+1} B_{j,d}$ is discontinuous at z.

5. Its integral is

$$\int_{\tau_j}^{\tau_{j+d+1}} B_{j,d}(t)dt = \frac{\tau_{j+d+1} - \tau_j}{d+1}.$$

6. It is affine invariant, i.e., for $u, v, t \in \mathbb{R}$ $B_{j,d,u\boldsymbol{\tau}+v}(ut+v) = B_{j,d,\boldsymbol{\tau}}(t)$, where $u\boldsymbol{\tau}+v := (u\tau_j + v)_j$.

Now suppose n, d are integers with $0 < d < n$. We say that $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{n+d+1}$ is a $(d+1)$ *extended* knot vector on an interval $[a, b]$ if

$$a = \tau_{d+1} < \tau_{d+2}, \quad \tau_n < \tau_{n+1} = b, \quad \tau_{i+d+1} > \tau_i, \quad i = 1, \ldots, n.$$

It is $(d{+}1)$-*regular* or $(d{+}1)$-*open* if in addition $\tau_1 = a$ and $\tau_{n+d+1} = b$; it is $(d{+}1)$-*regular uniform* or $(d+1)$-*open uniform* if $\tau_{i+1} - \tau_i = h$ for $i = d+1, \ldots, n$ and $h > 0$. The knot vector is *uniform* if $\tau_{i+1} - \tau_i = h > 0$ for $i = 1, \ldots, n+d$.

On the knot vector $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{n+d+1}$ we can define n B-splines of degree d. The linear space of all linear combinations of B-splines is the *spline space* defined by

$$\mathbb{S}^q_{d,\boldsymbol{\tau}} := \Big\{ \sum_{j=1}^{n} \boldsymbol{c}_j B_{j,d} \mid \boldsymbol{c}_j \in \mathbb{R}^q \text{ for } 1 \le j \le n \Big\}, \quad d \ge 0, \quad q \ge 1.$$

An element $f = \sum_{j=1}^n c_j B_{j,d}$ of $\mathbb{S}_{d,\boldsymbol{\tau}} = \mathbb{S}^1_{d,\boldsymbol{\tau}}$ is called a *spline function* if $q = 1$ or just a *spline* of degree d with knots $\boldsymbol{\tau}$, and $(c_j)_{j=1}^n$ are called the *B-spline coefficients* of f. For $q > 1$ the combination $\boldsymbol{f} = \sum_{j=1}^n \boldsymbol{c}_j B_{j,d}$ is a *spline curve.*

Suppose $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{n+d+1}$ is a $(d{+}1)$-open knot vector on $[a,b]$. A spline $f : [a,b] \to \mathbb{R}$ is by definition continuous from the right. We define $f(b)$ by taking limits from the left. Let $\boldsymbol{f} = \sum_{j=1}^n \boldsymbol{c}_j B_{j,d}$. Then the following properties hold:

1. B-splines $(B_{j,d})_{j=1}^n$ are *linearly independent* on $[a,b]$ and therefore a basis for $\mathbb{S}^q_{d,\boldsymbol{\tau}}$.

2. *Partition of unity*: $\sum_{j=1}^n B_{j,d} \equiv 1, \quad t \in [a,b]$.

3. *Convex hull property*: $\boldsymbol{f}(t)$ lies in the convex hull of $[\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n]$.

4. *Smoothness*: If z occurs m times in $\boldsymbol{\tau}$ then f has continuous derivatives of order $0, \ldots, d-m$ at z.

5. *Locality*: If t is in the interval $[\tau_k, \tau_{k+1})$ for some k in the range $d+1 \le k \le n$ then

$$f(t) = \sum_{j=k-d}^{k} c_j B_{j,d}(t), \tag{3.19}$$

6. *Affine invariance*: If $u\boldsymbol{\tau} + v := (u\tau_j + v)_{j=1}^{n+d+1}$ then

$$\sum_{j=1}^{n} c_j B_{j,d,u\boldsymbol{\tau}+v}(ut+v) = \sum_{j=1}^{n} c_j B_{j,d,\boldsymbol{\tau}}(t), \quad t \in [a,b],\ u,v \in \mathbb{R}. \tag{3.20}$$

7. *Nodal representation*:

$$t = \sum_{j=1}^{n} \tau_{j,d}^{*} B_{j,d}(t), \quad \tau_{j,d}^{*} = \frac{\tau_{j+1} + \cdots + \tau_{j+d}}{d}, \quad t \in [a,b].$$

8. *Derivative of a spline*:

$$Df(t) = d\sum_{j=2}^{n} \frac{c_j - c_{j-1}}{\tau_{j+d} - \tau_j} B_{j,d-1}(t), \quad t \in [a,b],$$

   where terms with 0 valued denominator are set to 0.

9. *Integral of a spline*:

$$\int_{\tau_1}^{\tau_{n+d+1}} f(t)dt = \sum_{j=1}^{n} \frac{\tau_{j+d+1} - \tau_j}{d+1} c_j. \tag{3.21}$$

10. If $z = \tau_{j+1} = \cdots = \tau_{j+d} < \tau_{j+d+1}$ for $1 \le j \le n$ then $f(z) = c_j$.

11. *Marsden's identity*:

$$(s-t)^d = \sum_{j=1}^{n} \prod_{i=j+1}^{j+d} (s - \tau_i) B_{j,d}(t), \quad t \in [a,b], \quad s \in \mathbb{R}. \tag{3.22}$$

3.4.4 Knot Insertion and Degree Raising (h- and p-refinement)

Suppose $\boldsymbol{\tau}$ is a knot vector. The distinct elements in $\boldsymbol{\tau}$ are called *break points.* We define the *multiplicity* of z in $\boldsymbol{\tau}$ as

$$\mu_{\boldsymbol{\tau}}(z) = \#\{\tau_j \in \boldsymbol{\tau} : \tau_j = z\}, \quad z \in \mathbb{R}.$$

Notice that $\mu_{\boldsymbol{\tau}}(z) = 0$ if z is not equal to one of the knots in $\boldsymbol{\tau}$. For $k \geq 0$ we define the knot vector $\boldsymbol{\tau}^{(k)}$ to have the same break points as $\boldsymbol{\tau}$, and

$$\mu_{\boldsymbol{\tau}^{(k)}}(\xi) = \mu_{\boldsymbol{\tau}}(\xi) + k \text{ for all } \xi \in \boldsymbol{\tau}.$$

Thus we increase the multiplicity of each break point in $\boldsymbol{\tau}$ by k.

If $\boldsymbol{t}$ is another knot vector then we say that $\boldsymbol{\tau} \subset \boldsymbol{t}$ if each break point ξ in $\boldsymbol{\tau}$ is also a break point in $\boldsymbol{t}$ and $\mu_{\boldsymbol{\tau}}(\xi) \leq \mu_{\boldsymbol{t}}(\xi)$.

Let d, e be integers, $0 \leq d \leq e$, let $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ be $(d+1)$ extended on $[a, b]$ and let $\boldsymbol{t} = (t_i)_{i=1}^{m+e+1}$ be an $(e+1)$ extended knot vector on the same interval $[a, b]$. If $\boldsymbol{\tau}^{(e-d)} \subset \boldsymbol{t}$ then $\mathbb{S}_{d,\boldsymbol{\tau}} \subset \mathbb{S}_{e,\boldsymbol{t}}$, and there is a matrix $\boldsymbol{A} \in \mathbb{R}^{m,n}$ transforming the B-splines in $\mathbb{S}_{d,\boldsymbol{\tau}}$ into the B-splines in $\mathbb{S}_{e,\boldsymbol{t}}$. Thus

$$B_{j,d,\boldsymbol{\tau}} = \sum_{i=1}^{m} a_{ij} B_{i,e,\boldsymbol{t}}, \quad j = 1, \ldots, n, \text{ or } \boldsymbol{B}_{d,\boldsymbol{\tau}}^T = \boldsymbol{B}_{e,\boldsymbol{t}}^T \boldsymbol{A},$$

where $\boldsymbol{B}_{d,\boldsymbol{\tau}}^T = [B_{1,d,\boldsymbol{\tau}}, \ldots, B_{n,d,\boldsymbol{\tau}}]$ and $\boldsymbol{B}_{e,\boldsymbol{t}}^T = [B_{1,e,\boldsymbol{t}}, \ldots, B_{m,e,\boldsymbol{t}}]$ are row vectors.

If $f = \sum_{j=1}^{n} c_j B_{j,d,\boldsymbol{\tau}}$ then $f = \sum_{i=1}^{m} b_i B_{i,e,\boldsymbol{t}}$, where

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{c}, \quad \boldsymbol{c} = [c_1, \ldots, c_n]^T, \quad \boldsymbol{b} = [b_1 \ldots, b_m]^T. \tag{3.23}$$

The case where $e = d$ is called *knot insertion* and corresponds to h-refinement in the finite element literature. The situation where $e > d$ and $\boldsymbol{t} = \boldsymbol{\tau}^{(e-d)}$ is called *degree* *raising* or *degree elevation* and corresponds to what is commonly known as p-refinement or p-enrichment [175, 45, 98]. In the general case where $\boldsymbol{\tau}^{(e-d)}$ is a proper subset of $\boldsymbol{t}$ both knot insertion and degree raising occur. When this transformation is carried out with degree raising followed by knot insertion, Hughes [87] introduced the term k-refinement to the isogeometric literature. Although it is possible to do the transformation in opposite order, i.e., a knot insertion followed by a degree raising, as observed in [87] in their discussion of k-refinement, this ordering leads to more coefficients and less smooth functions.

There are two algorithms for knot insertion. In Boehm's algorithm one knot at a time is inserted. In particular, if z is inserted in $\boldsymbol{\tau}$ say between τ_k and τ_{k+1} so that $\tau_k \leq z < \tau_{k+1}$, then we obtain (3.23) with

$$b_i = \begin{cases} c_i & i = 1, \ldots, k-d, \\ \frac{z-\tau_i}{\tau_{i+d}-\tau_i} c_i + \frac{\tau_{i+d}-z}{\tau_{i+d}-\tau_i} c_{i-1}, & i = k-d+1, \ldots, k, \\ c_{i-1} & i = k+1, \ldots, n+1. \end{cases} \tag{3.24}$$

Alternatively, using the Oslo Algorithms [34] we can insert all knots simultaneously and compute the elements of $\boldsymbol{A}$ row by row. Suppose t_i is located between τ_k and τ_{k+1}, i. e., $\tau_k \leq t_i < \tau_{k+1}$, then for row i,

$$\alpha_{j,r}(i) = \frac{t_{i+r} - \tau_j}{\tau_{j+r} - \tau_j} \alpha_{j,r-1}(i) + \frac{\tau_{j+r+1} - t_{i+r}}{\tau_{j+r+1} - \tau_{j+1}} \alpha_{j+1,r-1}(i), \quad j = k-r+1, \ldots, k, \; r = 1, \ldots, d, \tag{3.25}$$

starting with $\alpha_{j,k} = \delta_{j,k}$. Then we obtain Equation (3.23) with $a_{i,j} = \alpha_{j,d}(i)$ for $j = k-d, \ldots, k$ and $a_{i,j} = 0$ for other values of j.

The recurrence relation in (3.25) bears a strong resemblance to the one for B-splines given in (3.18). Since the numbers $\alpha_{j,d}(i)$ also have rather similar properties to $B_{j,d}(t)$, they are called *discrete B-splines*. For example

$$\alpha_{j,d}(i) \geq 0, \; j = 1, \ldots, n, \quad \sum_{j=1}^{n} \alpha_{j,d}(i) = 1, \; i = 1, \ldots, m.$$

For degree raising we also compute the transformation matrix $\boldsymbol{A}$ row by row [35]. Suppose as for knot insertion that $\tau_k \le t_i < \tau_{k+1}$ and set $\Lambda_{j,0,r}(i) = \delta_{j,k}$ for $0 \le r \le e$, $\Lambda_{j,\ell,r}(i) = 0$ for all j if $0 \le r < \ell$, and $\Lambda_{j,\ell,r}(i) = 0$ for all ℓ, r if $j < k - \ell$ or $j > k$. If we compute

$$\Lambda_{j,\ell,r}(i) = \frac{\ell}{r}\left(\frac{t_{i+r}-\tau_j}{\tau_{j+r}-\tau_j}\Lambda_{j,\ell-1,r-1}(i) + \frac{\tau_{j+r+1}-t_{i+r}}{\tau_{j+r+1}-\tau_{j+1}}\Lambda_{j+1,\ell-1,r-1}(i)\right) + \frac{r-\ell}{r}\Lambda_{j,\ell,r-1}(i), \tag{3.26}$$

for $\ell = 1,\dots,d$, $r = \ell,\dots,e$ and $j = k-\ell,\dots,k$ then $a_{i,j} = \Lambda_{j,d,e}(i)$ for $j = k-d,\dots,k$ and 0 for other values of j. Again terms with 0 denominator are set to 0. For $e = d$ we only need to compute $\Lambda_{j,\ell,\ell}(i)$ in (3.26) and we see that $\Lambda_{j,\ell,\ell}(i) = \alpha_{j,\ell}(i)$ for all j, r. It is shown in [135] that $\boldsymbol{A}$ is a nonnegative stochastic matrix:

$$\Lambda_{j,d,e}(i) \ge 0,\ j = 1,\dots,n, \quad \sum_{j=1}^{n}\Lambda_{j,d,e}(i) = 1,\ i = 1,\dots,m, \quad 0 \le d \le e.$$

An algorithm that is a literal implementation of (3.26) has complexity $O(de^2m)$; however, it is possible to derive faster algorithms for this kind of conversion. The main advantages are that it is quite stable and simple to implement.

Alternatively, degree raising can be carried out by converting each segment to Bernstein form, performing degree raising on the Bernstein form, and then converting back to spline form.

3.4.5 NURBS

Suppose $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{n+d+1}$ is a $(d+1)$-regular knot vector on $[a,b]$. Given positive numbers $\boldsymbol{w} = \{w_i\}_{i=1}^{n}$, we define the associated NURBS-basis of degree d by

$$R_{j,d}(t) := \frac{w_j B_{j,d}(t)}{\sum_{i=1}^{n} w_i B_{i,d}(t)}, \quad t \in [a,b], \quad j = 1,\dots,n, \tag{3.27}$$

where $B_{i,d}$ is the B-spline of degree d with knots $\tau_i,\dots,\tau_{i+d+1}$. Given $\boldsymbol{c}_j \in \mathbb{R}^q$ the sum

$$\boldsymbol{f} = \sum_{j=1}^{n} \boldsymbol{c}_j R_{j,d} \tag{3.28}$$

is called a *NURBS function* if $q = 1$ and a *NURBS curve* if $q > 1$. We have $R_{i,d} = B_{i,d}$ when $w_i = 1$ for all i. NURBS curves retains many of the desirable properties of splines curves. Moreover,

1. NURBS can represent conic sections exactly.
2. $R_{i,d}$ has the same local support and smoothness properties as $B_{i,d}$.
3. NURBS basis functions are nonnegative and form a partition of unity, hence the convex hull property holds.
4. $\{R_{1,d}, \ldots, R_{n,d}\}$ is linearly independent on $[a, b]$.
5. A NURBS curve is affine invariant.

The exact properties of these functions depend on $\boldsymbol{w}$ as well as the knot vector $\boldsymbol{\tau}$ and degree.

3.4.6 Tensor Product Splines

Using multi-index notation, an s-variate tensor product B-spline has the form

$$B_{\boldsymbol{j},\boldsymbol{d},\boldsymbol{T}}(\boldsymbol{t}) = \prod_{i=1}^{s} B_{j_i,d_i,\boldsymbol{\tau}_i}(t_i), \text{ where } B_{j_i,d_i,\boldsymbol{\tau}_i} \in \mathbb{B}_{d_i,\boldsymbol{\tau}_i},$$

where $\boldsymbol{d} = (d_1, \ldots, d_s)$, $\boldsymbol{T} = (\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_s)$, and $\boldsymbol{j} = (j_1, \ldots, j_s)$. Define $\mathbb{B}_{\boldsymbol{d},\boldsymbol{T}}$ to be the set of all possible such s-variate combinations. The $s-$variate tensor product spline space is defined by

$$\mathbb{S}^q_{\boldsymbol{d},\boldsymbol{T}} := \Big\{ \sum_{1 \le \boldsymbol{j} \le \boldsymbol{n}} \boldsymbol{c_j} B_{\boldsymbol{j},\boldsymbol{d},\boldsymbol{T}} \mid \boldsymbol{c_j} \in \mathbb{R}^q \text{ over all } B_{\boldsymbol{j},\boldsymbol{d},\boldsymbol{T}} \in \mathbb{B}_{\boldsymbol{d},\boldsymbol{T}} \Big\}, \quad d \ge 0, \quad q \ge 1.$$

The definition for the s-variate rational is extended analogously.

Let $F \in \mathbb{S}^s_{\boldsymbol{d},\boldsymbol{T}}$, and fix the i-th coordinate to be an element of the knot vector in that dimension. The $(s-1)$ free variables in Θ form an $(s-1)$-dimensional unit cube, $\ell_{i,j}(\boldsymbol{t}) = (t_1, \ldots, t_{i-1}, \tau^i_j, t_{i+1}, \ldots, t_s)$ for $\boldsymbol{t} = (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_s) \in [0,1]^{s-1}$, and $F(\ell_{i,j}(\boldsymbol{t}))$ is called a *generalized knot-line.*

3.4.7 NURBS Elements

Define $\theta_{\boldsymbol{i}}$ to be an s-dimensional rectangular parallelepiped (Cartesian product):

$$\theta_{\boldsymbol{i}} = [\boldsymbol{T}_{\boldsymbol{i}}, \boldsymbol{T}_{\boldsymbol{i}+1}] := [\tau^1_{i_1}, \tau^1_{i_1+1}] \times \cdots \times [\tau^s_{i_s}, \tau^s_{i_s+1}].$$

Each nonzero function in $\mathbb{S}_{\boldsymbol{d},\boldsymbol{T}}$ is a single multivariate polynomial over the interior of $\theta_{\boldsymbol{i}}$, and $\bigcup_{\boldsymbol{i}} \theta_{\boldsymbol{i}} = \Theta$. Then $\Omega_{\boldsymbol{i}} = F(\theta_{\boldsymbol{i}})$ is an *element* or a *patch* in the physical space. The generalized knot-lines form the boundaries between the patches.

Sometimes a measure of behavior of the system as a whole is gauged by the behaviors of the collection of *localized* stiffness matrices, one for each element. Denote by $\boldsymbol{S}^{\boldsymbol{i}}$ the matrix formed by integrating over a just the $\boldsymbol{i}$-th element. That is

$$a^{\boldsymbol{i}}(u,v) := \int_{\Omega_{\boldsymbol{i}}} \nabla u(x)^T \nabla v(x) dx$$

is used in computing the elements of $\boldsymbol{S}^{\boldsymbol{i}}$. If $s = 3$ and $\boldsymbol{d} = (3,3,3)$, then there are 64 $\phi_{\boldsymbol{j}}$ that are nonzero over $\Omega_{\boldsymbol{i}}$, and $\boldsymbol{S}^{\boldsymbol{i}}$ is (only) a 64×64 matrix.

Define the d-extended rectangular parallelepiped

$$\theta_{\boldsymbol{i},\boldsymbol{d}} := [\boldsymbol{T}_{\boldsymbol{i}-\boldsymbol{d}+1}, \boldsymbol{T}_{\boldsymbol{i}+\boldsymbol{d}}].$$

The significance of this set is that the value of a polynomial spline on θ_i only depends on the knots in $\theta_{\boldsymbol{i},\boldsymbol{d}}$.

3.5 Parametric Representation of Geometry

In this section we now focus on portraying a modeler's view of defining parametric representations of geometry, what is often called *modeling* in the CAD world. In general, the creator of the shape model is not the person who performs the analysis. Although many systems have analysis modules, the subsystem to create the shape is focused solely on shape. Furthermore, it would miss the point to create a system devoted exclusively to *design for analysis*, because the created design must be for shape, for analysis, for manufacturing and fabrication, for assembly analysis, versioning, and more. Hence *analysis-aware* modeling that exploits the currently available modeling flexibility of existing systems should be aimed at supporting the designer to make intelligent decisions about modeling that result in models F that are better able to support analysis while preserving the capability of supporting the other important facets of the production process. For the most part, we present the discussion in the context of dimensional models to illustrate the points in the studies of Section 3.6.

While the analyst begins the process with a shape, the designer works towards a shape *representation* that meets the design specifications as the end goal. Hence the process for attaining the modeling goal varies with the design discipline, the individual designer, and the CAD system environment. For certain types of design, e.g., feature-lines establish key characteristics of shape. The subsequent surface must be generated around those features. Surfaces are assembled into models along surface edge curves, matching them carefully. In another style of modeling four boundary curve elements define the essence of a surface whose interior representation must be conformally generated. Networks of such regions form the model. Still other styles of design create reference curves that define surfaces through operations such as surface of revolution, extrusion, sweep, and the like. Another style employs named standard feature objects like hole, boss, fillet, *etc.*, to describe shape, from which the designer or CAD system can generate the surfaces. Prevalent practice in design engineering delineates planar regions by bounding curves. The CAD process requires that a tensor product surface description is then created for computational purposes, typically a difficult process unless a nontensor product representation is added to the general representation. Typically, the model will be bounded by many surfaces, so a volume model will not be able to be represented as the mapping of a single cube.

It is important to point out that current CAD modeling typically focuses on constructing of the boundaries to define an object. Although it is clearly the case that the boundary representations alone are sufficient for some types of analysis like shell analysis, they are not sufficient for all types of analysis. In particular, it is important to appreciate that modeling systems have nurtured a modeling mindset focused on generating surface representations, not on full volumetric representations. However, it is necessary to create a fully specified volumetric representation F so that the space V_h can be defined and used in the analysis. Creating F from its boundaries is called *model completion*. To be suitable for a full volume analysis, but unnecessary for most design and fabrication requirements, the interior of the bounded region should have a representation as well, i.e., it should be a volumetric model.

However, issues of modeling volumes or completing a boundary model to a full volume model have not been the subject of broad research focus other than a few scattered efforts [28, 152, 128, 88]. We first examine some of the challenges of model completion.

3.5.1 Completion

There has been significant modeling research on the issue of completing a surface given boundaries, for both boundary curves in 2-D (our case) and 3-D. The generic problem is formidable, especially for a nonconvex bounded region. Indeed, for complicated planar regions, and even more so for 3-D curved boundaries, generating representations for smooth completions is still an area of research. Using a tensor product form requires the existence of four bounding curves, as described below. Complicated regions do not naturally lend themselves to this form (see Figure 3.3), in much the same way that complicated volumes cannot be straightforwardly represented as a single mapping of the unit cube. So it is necessary to decompose the model into multiple regions, each one of which is the mapping of a cube. The process for attaining the decomposition is not well defined. Thus, it will be better understood if the modeling process can incorporate, without undue effort on the part of the designer, intrinsically volumetric

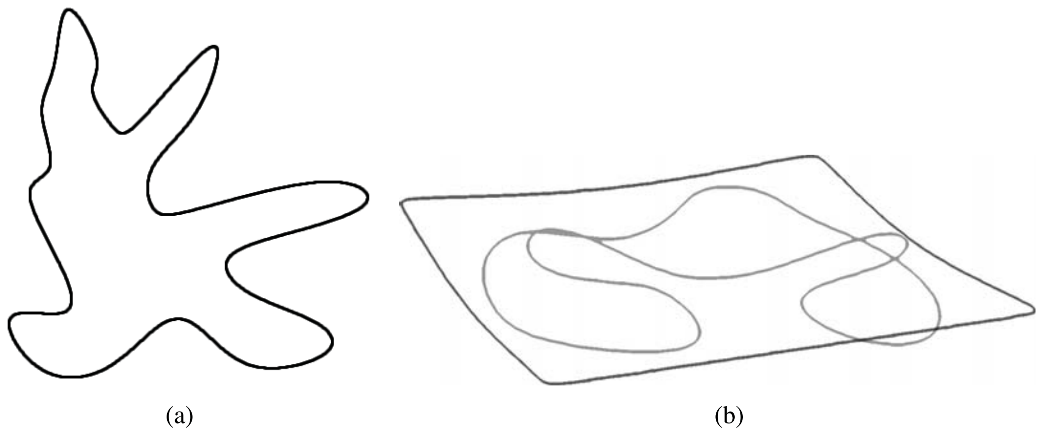


Figure 3.3. Boundary curves. (a) in 2D; (b) in 3D.

design operators. We believe simpler cases will illustrate the issues that arise in creating representations F that defines Ω and the reference space when starting with a boundary representation.

Putting aside the details of how to effect this, assume that Ω has a theoretical decomposition, and the current concern is creating a single mapping from Θ with whatever partition is necessary. That is, there are $2s$ boundary faces, and opposite boundary faces share the same degree and knot vectors. The studies that we present are characterized by $\Theta, \Omega \subset \mathbb{R}^s$, $s = 1, 2, 3$. As remarked earlier, $\partial\Theta$ has $2s$ bounding faces, each an $s-1$ manifold. The mappings from each face in $\partial\Theta$ to $\partial\Omega$ are designated by the coordinate held constant over the face. Hence, the face labeled $i = 2(j-1)+(\ell)$, $j = 1, \ldots, s$ and $\ell = 1, 2$ corresponds to the face that holds the $j-$th coordinate constant to value $\ell - 1$, and $\sigma_i = F|_{(\partial\Theta)_i}$. Considered separately, $\sigma_i : [0,1]^{s-1} \to \Omega$.

CAD models are generally represented only in terms of the boundary, that is, as a collection of mappings $\mathcal{A} = \{\alpha_p\}_p$, $\alpha_p : [0,1]^2 \to \mathbb{R}^3$, that have not be created with any considerations for analysis. Simply designing and representing the model can be a major challenge. Resulting representations have the characteristic that:

- $\bigcup_p \alpha_p([0,1]^2)$ form a closed region of space $\partial\Omega$;
- Two surface pieces can meet only along a boundary curve, which is either identical or entirely disjoint;

- Arbitrarily many surfaces pieces can define a boundary; and
- Arbitrarily many surface pieces can meet at a point.

3.5.2 Representing a Line Segment

A line segment may be considered the parametric completion of its boundary, namely, the two endpoints. Consider points $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. Viewed as a B-spline curve, the linear parameterization of the line segment joining them is $\gamma(t) = \boldsymbol{P}_1 B_{1,1}(t) + \boldsymbol{P}_2 B_{2,1}(t)$, where the corresponding knot vector is $\boldsymbol{\tau} = [0, 0, 1, 1]$. Using the degree raising algorithms (p-refinement) this can be represented as a higher order curve γ_d. Since $\gamma_d(t) = \gamma(t)$ for all t, the curve exhibits constant velocity. Using knot insertion to refine the higher degree curve, perhaps nonuniformly, we obtain a curve $\tilde{\gamma}_d(t)$, that is still the same curve, but written in a different representation.

It is possible to write the same line with different, seemingly rather arbitrary, nonlinear parameterizations. Now, we create several representations for later use in Section 3.6.

Let $\boldsymbol{P}_1 = (0, 0)$ and $\boldsymbol{P}_2 = (1, 0)$. We can just consider the mapping from $[0, 1] \rightarrow [0, 1]$, since the second coordinate is 0. Let $d = 3$, and consider two different knot vectors to complete the interior of the interval. Let $\boldsymbol{\tau}_1$, be the open uniform knot vector,

$$\boldsymbol{\tau}_1 = [0, 0, 0, 0, h, 2h, \ldots, (n-4)h, 1, 1, 1, 1], \quad h = 1/(n-3) ,$$

and let for $0 < a < b < 1/2$, $n > 8$, and $\delta = (1 - 2b)/(n - 8)$,

$$\boldsymbol{\tau}_2 = [0, 0, 0, 0, a, b, \delta, 2\delta, ..., (n-9)\delta, 1-b, 1-a, 1, 1, 1, 1], \tag{3.29}$$

where the value of δ is chosen so that $b, \delta, 2\delta, \ldots, (n-9)\delta, 1-b$ is a uniform partition of $[b, 1-b]$. By the Nodal Representation Property (Section 3.4.3),

$$x = \mathcal{I}_k(x) := \sum_{j=1}^{n} \tau_{k,j}^{*} B_{j,3,\boldsymbol{\tau}_k}(x), \tag{3.30}$$

where $\tau_{k,j}^{*} = (\tau_{k,j+1} + \tau_{k,j+2} + \tau_{k,j+3})/3$, $j = 1, \dots, n$, so that,

$$\boldsymbol{\tau}_1^{*} = (\tau_{1,j}^{*})_{j=1}^{n} = [0, \frac{1}{3}h, h, 2h, \dots, 1-h, 1-\frac{1}{3}h, 1], \quad h = 1/(n-3).$$

Define a nonlinear parameterization of the unit interval with uniformly spaced coefficients given by

$$U_k(x) = \sum_{i=1}^{n} \frac{i-1}{n-1} B_{i,3,\boldsymbol{\tau}_k}(x), \quad k = 1, 2. \tag{3.31}$$

Notice that $\mathcal{I}_k$ is the identity and U_k stretches the two knot intervals near both endpoints, $k = 1, 2$. This process can be extended to higher degree, in which case the $d-1$ knot intervals near both endpoints are stretched.

Applying the derivative formula reveals that on the first two knot-intervals and the last two knot-intervals the derivative changes quadratically, but on the rest of the interior knot-intervals, it is constant. Hence U_k is linear on all but the two boundary knot intervals near the ends. An additional application of the derivative formula reveals that the second derivative is negative on the first two intervals and positive on the last two intervals, so the curve is concave on the first two intervals, and convex on the last two intervals. This mapping effects a stretch of the two knot intervals at each end and preserves it as constant in the middle (see Figure 3.4).

We explore the effects of control polygon degeneracy on the knot intervals by creating $\boldsymbol{c}_1$, corresponding to map M_1, to have a cluster of two identical sequential control points ($c_{1,n/2} = c_{1,n/2+1}$), and $\boldsymbol{c}_2$, corresponding to M_2, to have one cluster of three identical sequential control points (at the corresponding center of the control point range), both are defined over a uniform open knot vector that yields n basis functions, where n is even for M_1 and odd (1 more) for M_2, and for which the remaining control points are uniformly spaced. So for $n = 10$,

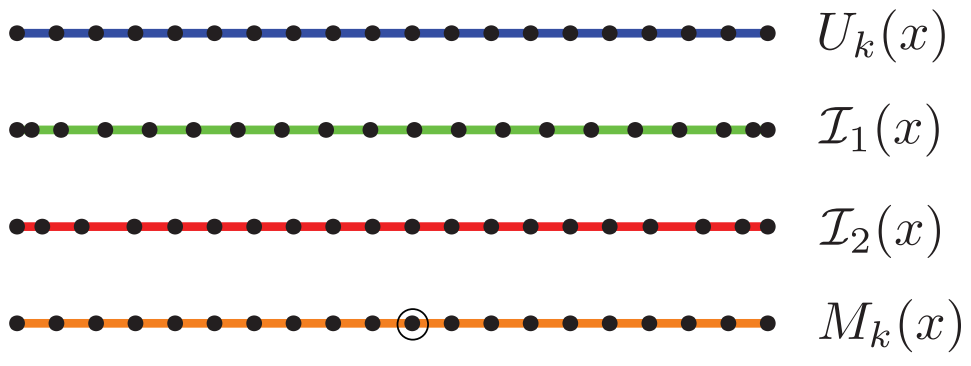


Figure 3.4. Coefficients of the geometry maps. The top image portrays the evenly spaced coefficients of $U_k(x)$, $k = 1, 2$. The second and third rows are coefficients of $\mathcal{I}_k(x)$, the identity, for $k = 1, 2$. They depend on the knot vectors. The fourth row shows the coefficients of M_1 are mostly the same as the coefficients of U_1, except for the double control point.

$$\text{Ordered Control points for } \boldsymbol{c}_1 = [0, 1/8, 1/4, 3/8, 1/2, 1/2, 5/8, 3/4, 7/8, 1] \tag{3.32}$$

$$\text{Ordered Control points for } \boldsymbol{c}_2 = [0, 1/8, 1/4, 3/8, 1/2, 1/2, 1/2, 5/8, 3/4, 7/8, 1]. \tag{3.33}$$

In Section 3.6.1 we investigate the interactions between the knot vectors and the mappings $\mathcal{I}$ and U that act as the map from reference space to physical space in the case of longitudinal vibrations along a string. Cottrell et al. [39] studied this problem for $\mathcal{I}_1$ and U_1. We investigate more general cases and consider how interactions of mappings and knot vectors change the V_h and affect the eigenstructures.

3.5.3 Completing Surface Regions Bounded by Curves

Given a curvilinear rectangular mesh of curves, there has been significant work on techniques to complete the representations to an implied smooth surface, including early work by Coons [38] and Gordon [67] in representing shape objects. However, when a single boundary has no straightforward decomposition into four boundary curves, most of these methods cannot be applied directly. Rather, the user must decompose the boundary into pieces amenable to patch fitting, and then work to guarantee that the patches join smoothly. Guaranteeing that the interior boundaries are identical can be a challenge when bounding curves are nonlinear.

Research in [128] for the planar case and [88] for the nonplanar case seem to generate reasonable surfaces and parameterizations for modeling, but they have not been tested for suitability in analysis. Figure 3.5 shows completions of the bounding curves in Figure 3.3 formed using these methods, respectively.

First we investigate some specific representations of simple geometries for curves and their surface completions that are used in our studies in Section 3.6.

3.5.4 Representing a Circular Arc

Used to represent part of a boundary, circular arcs appear ubiquitously in mechanical design. Suppose it is necessary to represent an arc of β radians taken from a circle of radius r. We follow the usual approach in [37] to create a quadratic NURBS template in the $x - y$ plane that can be affinely mapped to any position. The arc is represented initially as a quadratic rational B-spline with knot vector $\boldsymbol{\tau} = [0, 0, 0, 1, 1, 1]$. The representation will have one knot span. As shown in [37], the arc can be easily represented by the same Euclidean control points with many different functions w as long as $w_1 w_3 / w_2^2 = \sec^2(\beta/2)$. Commonly, $w_1 = w_3 = 1$ and $w_2 = \cos(\beta/2)$, so the arc

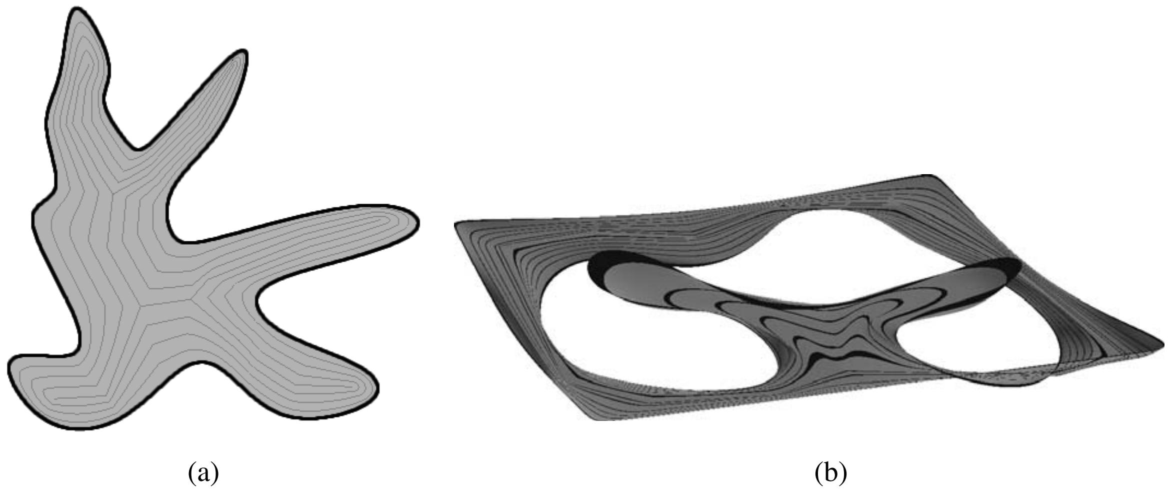


Figure 3.5. Filled boundary curves. (a) in 2D; (b) in 3D

is represented as

$$\boldsymbol{A}(t) = \sum_{i=1}^{3} \boldsymbol{P}_i R_{j,2}(t) \qquad w(t) = B_{1,2}(t) + \cos(\beta/2) B_{2,2}(t) + B_{3,2}(t),$$

where $\boldsymbol{P}_1 = r(1,0), \boldsymbol{P}_2 = r(1, \tan(\beta/2)), \boldsymbol{P}_3 = r(\cos\beta, \sin\beta)$ and the $R_{j,2}$ are defined as in Section 3.4.5. When $\beta = \pi/2$, and $r = 1$, $\boldsymbol{P}_1 = (1,0)$, $\boldsymbol{P}_2 = (1,1)$, $\boldsymbol{P}_3 = (0,1)$ and $w_2 = \sqrt{2}/2$. If $\beta = \pi$, $w_2 = 0$ which can lead to computational consistency problems. If $\beta > \pi$, then $w_2 < 0$, leading to yet other computational problems, such as potential zeros in the denominator. Thus a commonly used constraint is that $\beta < \pi$.

A full circle can be represented as three rotated instances of a $2\pi/3$ arc, giving rise to the NURBS representation

$$\boldsymbol{C}_3(t) = \sum_{i=1}^{7} \boldsymbol{P}_{3,i} R_{i,2,\boldsymbol{\tau}_3}(t) \quad \text{defined with } w_3(t) = \sum_{i=1}^{7} w_{3,i} B_{i,2,\boldsymbol{\tau}_3}(t), \quad t \in [0,1],$$

where

$$\begin{aligned} \boldsymbol{\tau}_3 &= [0,0,0,1/3,1/3,2/3,2/3,1,1,1], \\ \mathcal{P}_3 &= r[(-\sqrt{3},-1),(0,-1),(\sqrt{3},-1),(\sqrt{3}/2,1/2),(0,2),(-\sqrt{3}/2,1/2),(-\sqrt{3},-1)], \end{aligned}$$

and

$$\boldsymbol{w}_3 = [1,1/2,1,1/2,1,1/2,1,1].$$

The control points are successive vertices and midpoints, respectively, of the sides of an equilateral triangle that inscribes a circle of radius r having the origin as center.

Alternatively we can represent the circle as four rotated images of a quarter circle. This gives the NURBS curve

$$\boldsymbol{C}_4(t) = \sum_{i=1}^{9} P_{4,i} R_{i,2,\boldsymbol{\tau}_4}(t), \quad \text{defined with } w_4(t) = \sum_{i=1}^{9} w_{4,i} B_{i,2,\boldsymbol{\tau}_4}(t), \quad t \in [0,1],$$

where

$$\begin{aligned}
\boldsymbol{\tau}_4 &= [0,0,0,1/4,1/4,1/2,1/2,3/4,3/4,1,1,1],\\
\mathcal{P}_4 &= \frac{r}{2}[(-1,-1),(0,-1),(1,-1),(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1)],\\
\boldsymbol{w}_4 &= [1,1/\sqrt{2},1,1/\sqrt{2},1,1/\sqrt{2},1,1/\sqrt{2},1].
\end{aligned}$$

For this configuration defining the circle in terms of an arc in each of the quadrants, the control points are the successively alternating vertices and midpoints of the sides of a square with that inscribed circle of radius r centered at the origin. The reader is referred to both [159] and [21], which discuss various ways to represent circles in fuller detail. These representations only guarantee C^0 across the boundaries of each of the arcs.

Both 3-arc and 4-arc representations are rational quadratic. A third rational representation of a circle is given by two semicircular arcs with cubic representations joined with C^1 smoothness. This representation, called $\boldsymbol{C}_2$ uses only six control points [159], is shown in Figure 3.6(b), and has configuration

$$\text{knot vector:} \quad \boldsymbol{\tau} = [0,0,0,0,1/2,1/2,1,1,1,1], \tag{3.34}$$

$$\text{weights:} \quad \boldsymbol{w} = [9,1,1/3,1/3,1,9] \tag{3.35}$$

$$\text{control coefficients:} \quad \mathcal{P}_2 = \{(1,0),(1,2),(-1,2),(-1,-2),(1,-2),(1,0)\}. \tag{3.36}$$

In the figure, although the radii are drawn with uniformly spaced parameter values, notice the nonuniformity of the disc parameterization. That means that uniform h-refinement can well lead to nonuniformly sized analysis elements. That may be desirable for some tangent analysis problems. If it is not desirable then parameterizing the radii nonuniformly and creating curvilinear radii (to generate elements of more equal size) may be an appropriate completion when given $\boldsymbol{C}_2$ as a boundary representation.

Above we have used subscripts to reflect the number of distinct rational arc pieces used to represent a complete circle.

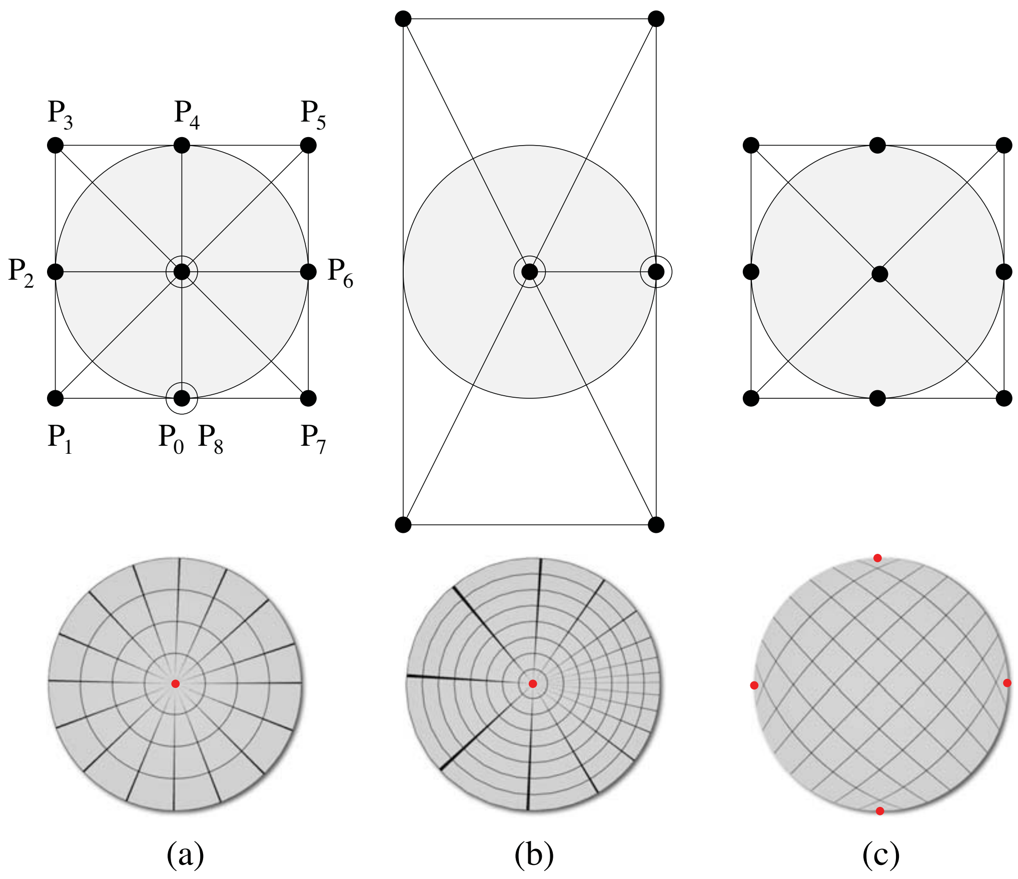


Figure 3.6. Three ways to represent a disc of radius one. Column (a) is mapped with D_1, column (b) with D_2, and column (c) with D_3. Circles around black control points mean that more than one control point sits on the black control point location. Degeneracies are marked with red points. These representations are used to solve the drumhead problem.

3.5.5 Solid Discs from Circular Boundaries

The disc can be represented in many ways using NURBS. In this section we discuss three different representations that are all exact but differ in construction, degeneracies and smoothness. There are two widely used representations for completing the disc. The first, used in [209, 17], is a polar type of parameterization of the disc induced by using $\boldsymbol{C}_4$ as one boundary. Select a point in the interior of the circle $\mathcal{O}$. Usually the center of the circle is selected for symmetry reasons, but another point could serve. Now select a representation for the unit interval. It could be the identity on any knot vector or any degree, it could be linear, quadratic, or it could be a cubic, like U_k. A tensor product representation is generated by selecting as the Euclidean part of the control points the scaled and rotated version of one of the representations of the line and one of the above representations of a circle. Translate, rotate and scale multiple instances of the line representation so there is one starting at $P_{4,j}$ and ending at $\mathcal{O}$ for each j. Name the rotated scaled instance of the i-th coefficient of the line from $P_{4,j}$ to the origin be $D_{1,i,j}$ with $w_{1,i,j} = w_{4,j}$. Call this representation D_1.

This representation creates an orthogonal parameterization of the unit circle whose isoparametric lines are either circles or radii. See Figure 3.6(a). The mesh is shown in the upper figure; isoparametric lines are drawn in the lower one. In this figure, the radial parameterization is linear. The lines drawn are showing parameterization, but not indicated analysis elements. An analogous solid representation can be generated from $\boldsymbol{C}_3$.

The resulting tensor product representation degenerates one whole boundary curve to a single point, so $J_{D_1}(\mathcal{O}) = 0$. The rate at which it goes to zero can be affected by modifying nearby control points so that the radii are not parameterized linearly, as in U_k. Thus the effect on $\boldsymbol{F} = D_1$ and its power to represent the solution space is adjustable without affecting the boundary geometry. It will affect the element shapes. The discussion in Section 3.6.4 concerns the impact on the induced function space and the ensuring impact on analysis results. Call D_1 the mapping that embodies $\boldsymbol{C}_4$ as one boundary in representing disc, places the origin as its opposite, and represents the radii linearly.

$\boldsymbol{C}_2$ is also a polar type. The initial NURBS representation for the disc has two rational cubic semicircles. Since there is one interior double knot, it is C^1 at the join. (Figure 3.6 (b)). The rest of the surface is generated using the same polar approach ad for D_1. However, the result is not as uniform an angular representation as for the first case. Call this mapping D_2.

Note that for both mappings, an annulus could easily be modeled by choosing the circle representation for the smaller radius as the inner boundary and using a line representation that terminates at the control points of the smaller circle instead of the origin. The w's that define the rational space remain the same.

The third modeling choice is fundamentally different from the first two inasmuch as it uses opposite arcs as the matched boundaries of the tensor product representation.

$$\begin{aligned}
\sigma_1(v) &= \boldsymbol{A}(v) \\
\sigma_2(v) &= R_\pi(\boldsymbol{A}(-v)); \\
\sigma_3(u) &= R_{\pi/2}(\boldsymbol{A}(u)); \\
\sigma_4(u) &= R_{3\pi/2}(\boldsymbol{A}(-u)),
\end{aligned}$$

where $\boldsymbol{A}$ is the $\pi/2$ radian circular arc, $R_\theta(\boldsymbol{P})$ means to rotate $\boldsymbol{P}$ through an angle θ. Nine control points determine the tensor product rational quadratic surface, eight of which are specified by the boundary control points. The remaining is chosen to be $\mathcal{O}$ and the associated w set to $1/2$. Call this mapping D_3, as shown in Figure 3.6(c). With respect to the number of basis functions, this is the most concise representation of a disc. In this case nine control points are needed as compared to 18 and 12 required for the first two representations, respectively. Furthermore, note that this mapping exhibits no interior degeneracy, but there are four locations on the boundaries at which the boundary curves meet at which the Jacobian vanishes. The rate at which the Jacobian goes to zero can be adjusted by modifying the initial NURBS representations of the boundary curves, and adjusting the rate at which the Jacobians go to 0 by modifying the locations of the control points on the interior, particularly those that result from h- or k-refinement. The boundary geometry is unperturbed by these modifications, but V_h changes, because $\boldsymbol{F}$ is different, even though they are all in a single $\mathbb{S}_2$. The various completions are not affine transformations of each other. Note that neither $\boldsymbol{C}_2$ nor $\boldsymbol{C}_3$ are suitable for use with this disc representation, but are quite reasonable for most other computational uses encountered in CAD. Again, see the discussion in Section 3.6.4 concerning the impact on the induced function space and the corresponding impact on analysis.

3.5.6 Volumetric Models such as Solid Cylinders and Solid Tori

CAD systems generate multiple boundary models from the three circle representations above, cylinders (without the top and bottom surface), tori, and other types of extruded and swept surfaces. However, although CAD systems do not generate volumetric models, volumetric models of those shapes can be generated from the disc surface completions above. A variant of the disc of revolution has been used to generate geometry of vascular structures and to create the trial space for IA of blood through those structures[209]. It was used to generate geometry for optical lenses and carry the varying index of refraction volume attribute for computing the optical behavior of those lenses [127]. A sweep surface is defined as

$$\sigma(u,v) = A(u) + M_u(S(v))$$

where S is the cross section curve, A is a spline curve along which S is swept, and M_u is a transformation incorporating rotation and nonuniform scaling of $S(v)$ as a function of u. Unfortunately this representation has self-intersections wherever the radius of curvature of A is less than the first intersection of the curve normal of A with $\sigma(u,v)$. Generalized cylinders and tori also fall into this category. If the boundary has no self intersections, this can be made into a volumetric sweep by using the surface completion $S(v,w)$ of $S(v)$. If there are any self intersections, then this method is unsuitable.

Generalizations of this representation include allowing S to also be a function of u, and allowing S to be nonplanar. Both of those generalizations were combined in [125] to create a modeling technique for generalized cylinder-like objects with overhang regions. Such shapes cannot be modeled as a single NURBS patch if the cross section surfaces in the sweep are required to be planar. Because of the complexities of the boundary and some constraints on the isoparametric contours, there are no straightforward techniques for modeling some of them as images of multiple 3-cubes. For example, Figure 3.7(left) cannot be modeled as a single sweep with planar cross sections. 3.6.11. A partition of the data into multiple Θ domains will create mappings F that also split material properties

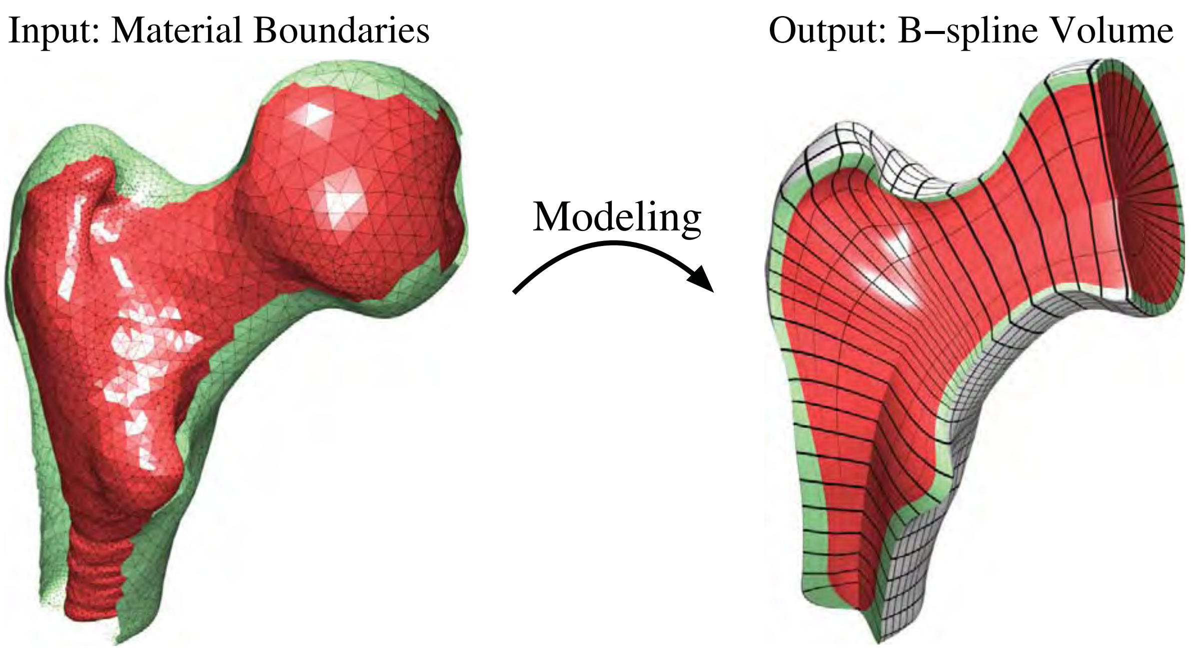


Figure 3.7. Input (left) and output (right) of the modeling methodology proposed in Chapter 4.

(bone type), and Young's modulus (for linear elasticity) in unnatural ways, not along isoparametric surfaces of the resulting mappings. Thus, it was deemed appropriate to allow some distortion in the parameterization, while still maintaining a quality model for analysis. An example of this approach is studied in Section 3.6.11.

3.6 Studies

In this section we examine studies that demonstrate how different modeling choices, in fact, can easily lead to different simulation results. The first mathematical model problem (Section 3.6.1) is the study of the eigenstructure of a system under different completion representations. We use a structural vibration analysis problem in both 1-D and 2-D. Then in Section 3.6.5, the Poisson equation is solved on different physical domains in 2D, where each domain is exactly represented with different choices of geometric model. In both sections h-refinement is applied and convergence rates are compared. Finally, in Section 3.6.11 we present a 3D study of the linear elastic deformation of a complex geometric model of a human femur.

3.6.1 Vibrations

The natural frequencies of a vibrating string are typically modeled by Equation 3.10. Finding the natural frequencies can correspondingly be transformed to solving the system in Equation 3.14. While the nonaffine mapping F affects the eigenstructure, so does the choice of underlying B-spline space $\mathbb{S}$.

3.6.2 Longitudinal Vibrations of a 1-D Elastic Rod

Consider the eigenvalue problem (3.10) for $s = 1$ and $\Omega = [0, 1]$

$$-u''(t) = \lambda u(t), \quad t \in (0,1), \quad u(0) = u(1) = 0. \tag{3.37}$$

The exact eigenvalues and eigenfunctions for this problem are

$$\lambda_k = k^2\pi^2, \quad u_k(t) = \sin(k\pi t), \quad k = 1, 2, 3, \ldots. \tag{3.38}$$

The eigenfunctions u_k are orthogonal both with respect to the usual L_2 inner product and the energy inner product $a(u, v) = \int_0^1 u'(t)v'(t)dt$.

Cottrell et al. [39] solved this problem numerically using an isogeometric Rayleigh-Ritz method. It was demonstrated that with uniform knots one can get rid of outliers using a nonlinear mapping F. We demonstrate here that, with a nonuniform knot vector, a linear F also has no outliers. Also, we show that control mesh degeneracies in the interior of Ω have a negative effect on the eigenstructure.

We solve (3.37) numerically by the isogeometric Rayleigh-Ritz method (3.14) using four different spaces V_h generated by different bases $(\phi_i = \psi_i \circ F^{-1})_{i=1}^n$.

We use $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$, as defined in Section 3.5.2 as knot vectors, providing uniform open and nonuniform open knots with larger reference space elements near the ends. The mappings to physical space are the identity, $\mathcal{I}_k$, and the uniformly spaced coefficients, $\mathcal{U}_k$, over each knot vector, $k = 1, 2$. Then $V_{\ell,\mathbb{S}_k}$ is the physical space of approximating functions space for the k-th knot vector, where $\ell \in \{\mathcal{I}, U\}$. While $V_{\mathcal{I},\mathbb{S}_k} = \mathbb{S}_k$ is the spline space for the k-th knot vector, $V_{U,\mathbb{S}_k} = span\{\phi = \psi(U_k^{-1}) : \psi \in \mathbb{S}_k\}$ is not a spline space.

As we have shown in Section 3.5.2, both U_1 and U_2 are increasing, concave on the first $d-1$ intervals, convex on the last $d-1$, and linear in between. Thus, U stretches the intervals near the boundaries and shrinks the interior ones with a constant scaling. The same behavior will be observed for other degrees, as long as n is sufficiently large, which occurs when we consider the discrete normalized spectrum. The choices of a and b are the stretch factor at the ends. If they are chosen too large, then the slope of the interior line segment becomes small. The consequence is that the slope of U_k^{-1} is large in that region. Since the values of J_F and $J_{F^{-1}}$ affect both the stiffness and mass matrices, they can adversely affect the eigenstructure. However, an optimal location will depend on the number of interior knots as well. This study was run with several different nonuniform knot vectors, although only one is shown below. Note that $\mathcal{I}_k' \equiv 1$, but U_k is more interesting in its behavior.

The *normalized discrete spectrum* is $\boldsymbol{\eta} = [\eta_0, \eta_1, \ldots, \eta_{N-1}]$ where η_k is the ratio between the eigenvalue $\sqrt{\lambda_{k,h}}$ and its corresponding exact solution, (3.38), i.e.,

$$\eta_k = \sqrt{\frac{\lambda_{k,h}}{\lambda_k}}. \tag{3.39}$$

Designed to have knots at prescribed distances from the endpoints of both sides, $\boldsymbol{\tau}_2$ has its remaining knots evenly-spaced across the rest of the interior interval. We demonstrate that the identity map with this basis creates $V_h = \mathbb{S}_{d,\boldsymbol{\tau}_2}$ with no optical branches in the normalized discrete spectrum. It becomes clear that the mapping F, the space $\mathbb{S}_{d,\boldsymbol{\tau}_2}$ and the particular PDE being solved all interact to affect the appropriation properties of the resulting V_h and the rates at which computed solutions converge towards the true solution.

Examine the ϕ-basis functions defined on Ω in Figure 3.8 of the four mappings U_k and $\mathcal{I}_k$ where Figure 3.4 shows their coefficients. By stretching the end elements, U_k stretches the basis functions at the boundaries so they extend farther into the interval

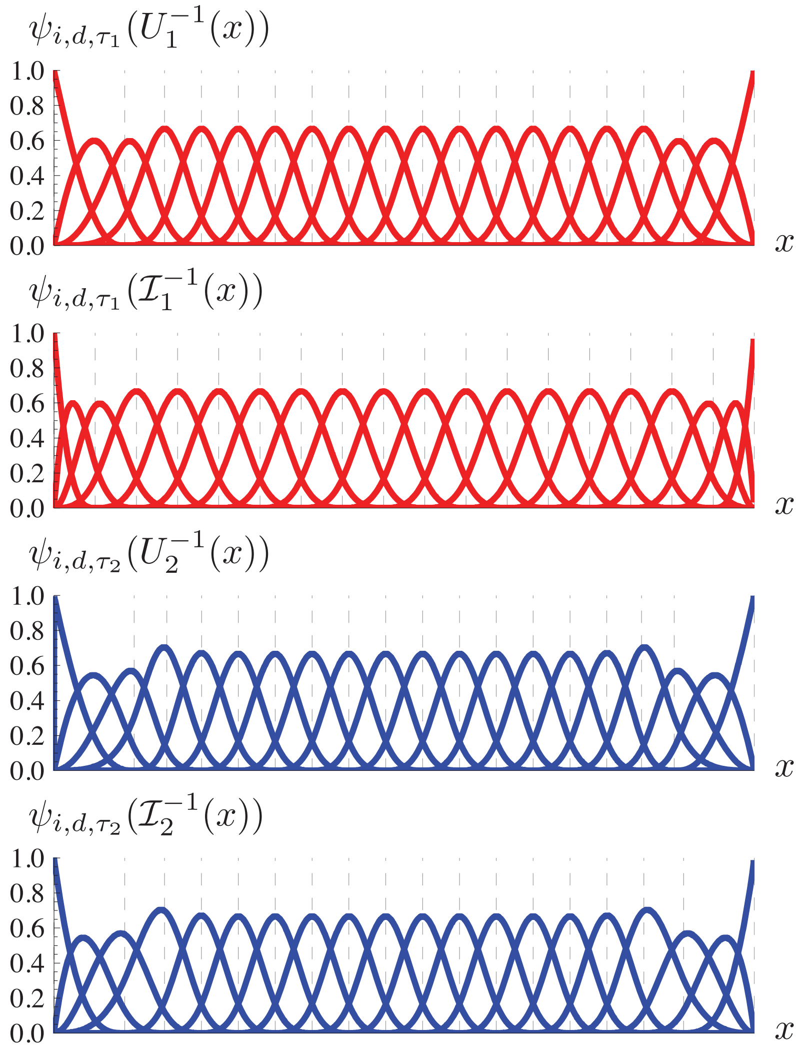


Figure 3.8. Given the knot vectors $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$, the first column shows $\psi_{i,d,\boldsymbol{\tau}_k}(U_k^{-1}(x))$, and the second shows $\psi_{i,d,\boldsymbol{\tau}_k}(\mathcal{I}_k^{-1}(x))$ $(k = 1, 2)$. Note that $\psi_{i,d,\tau_k}(\mathcal{I}_k^{-1}(x))$ are not stretched, but setting a and b appropriately causes the end functions to exhibit wider support and more resemble $\psi_{i,d,\boldsymbol{\tau}_k}(U_k)$, $k = 1, 2$

and have a more rounded shape, compared to $\mathcal{I}_k$ that maintain the uniform knot spacing on Ω from Θ.

By referring to Figure 3.9 and Figure 3.10, given the uniform open knot vector $\boldsymbol{\tau}_1$, as observed in [39], for the identity on the uniform open knot vector $\mathcal{I}_1$, there are outliers of η towards the end of the spectrum that become worse with increasing degree. The U_1 spectrum does not produce outliers, and with growing degree η gets flatter. This is

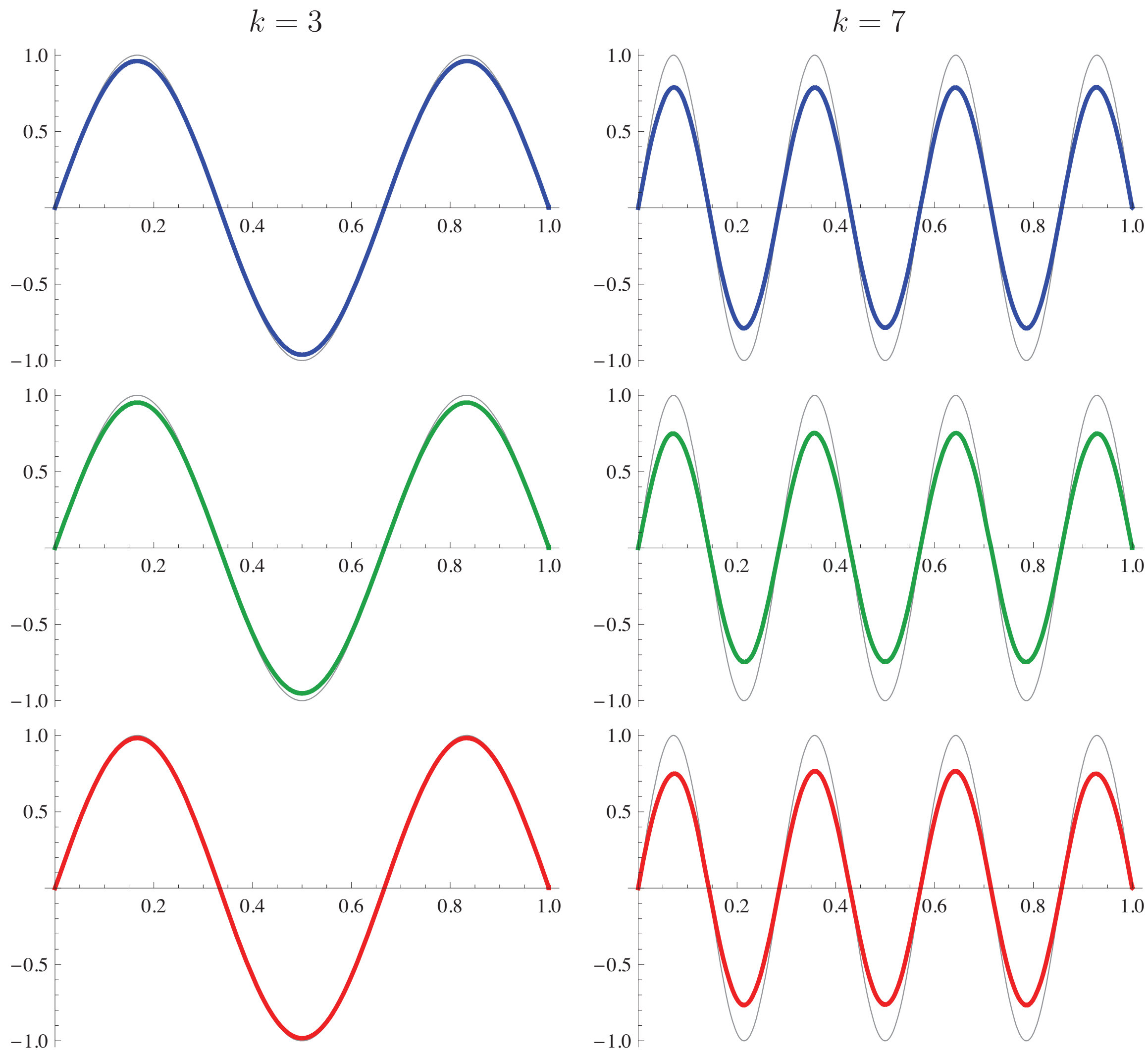
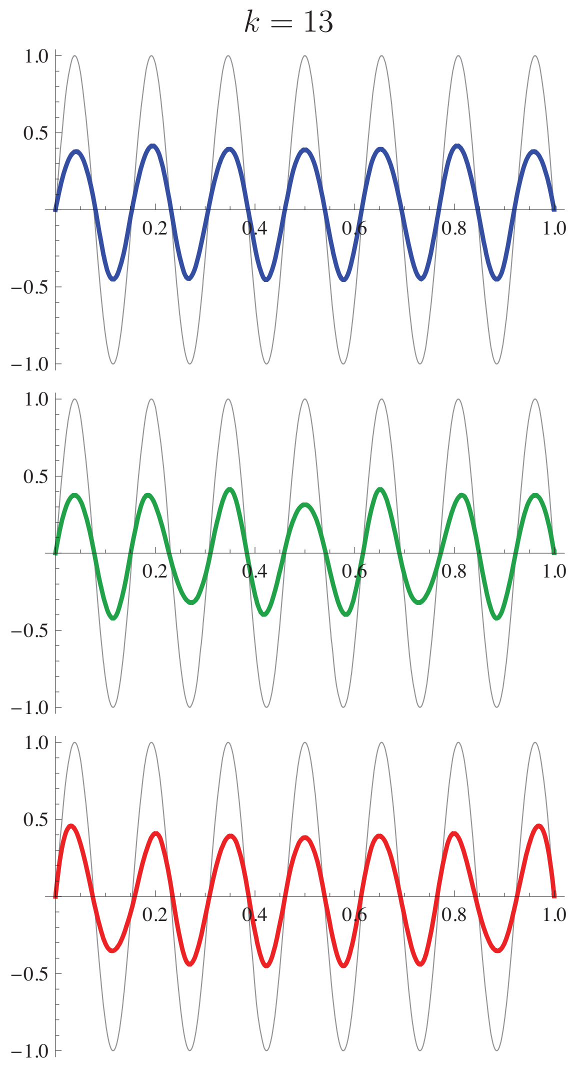


Figure 3.9. Approximated eigenfunctions (exact solution in gray) for the modes $k = 3, 7$ using the nonuniform mapping U_1 (first row), and the identity mappings $\mathcal{I}_1$ (second row), $\mathcal{I}_2$ (third row).



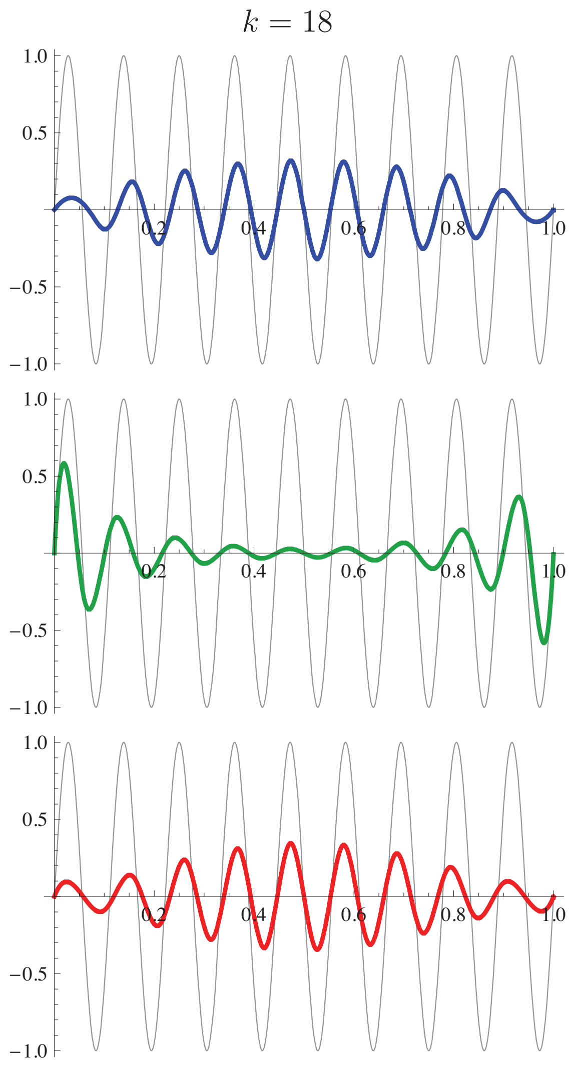


Figure 3.10. Approximated eigenfunctions (exact solution in gray) for the modes $k = 13, 18$ using the nonuniform mapping U_1 (first row), and the identity mappings $\mathcal{I}_1$ (second row), $\mathcal{I}_2$ (third row).

shown in Figure 3.11 (a). However, with the choice of $\boldsymbol{\tau}_2$, the linear map $\mathcal{I}_2$ performs better than U_1 in Figure 3.11 (a)) where outliers are completely removed and η is flatter than using U_1. Note, that the η for U_2 is not as flat as with $\mathcal{I}_2$; the corresponding η are not as flat as with U_1. It is useful to also look at the eigenvectors. The eigenvectors allow us to generate the approximations to the eigenfunctions. For $k = 18$, using either U_1 or $\mathcal{I}_2$ approximates the true eigenfunction better than $\mathcal{I}_1$ does.

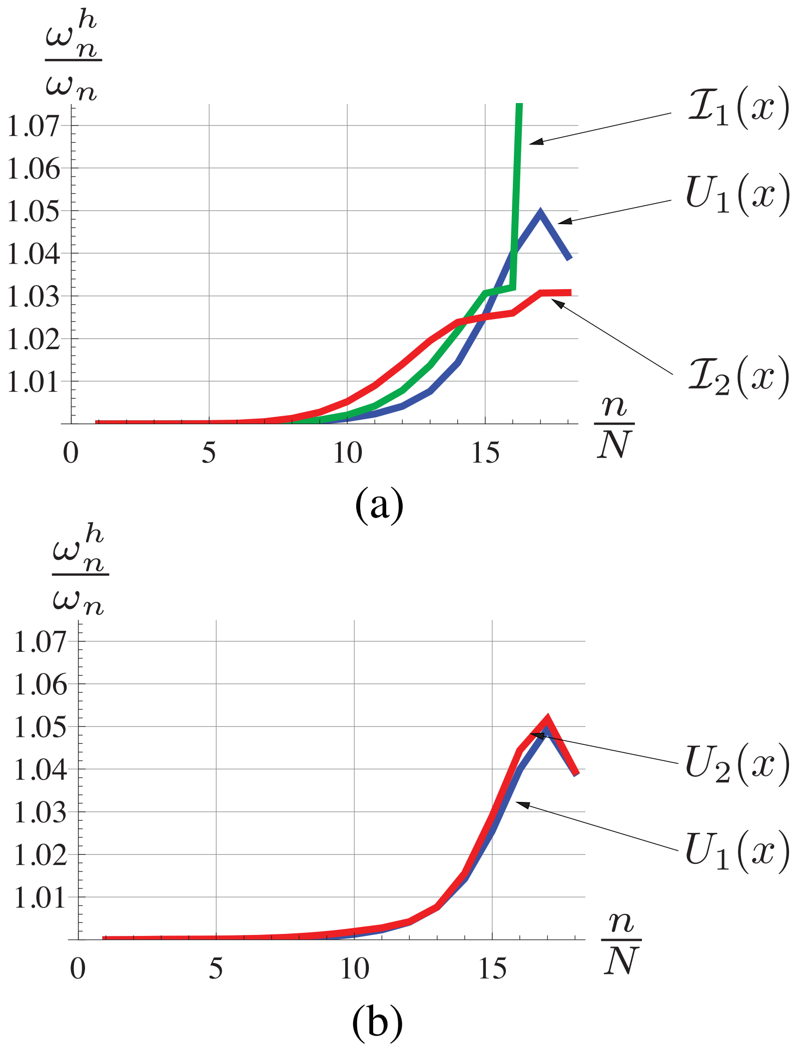


Figure 3.11. Two cubic identity mappings constructed from different knot vectors. The particular choice of nonuniform knot vector yields a flatter normalized discrete spectra for the vibrating rod.

This example illustrates that the Jacobian of a mapping is not the sole factor governing numerical quality. Having the identity matrix as its Jacobian, the identity mapping on $\boldsymbol{\tau}_1$ produces equally-sized elements. A standard isoparametric mesh quality metric [182], would judge its quality as optimal. This study, however, demonstrates that nonuniform knot vectors perform better, producing, when knots are chosen as above, wider elements at the domain's boundary, and still maintain a constant Jacobian. In the same way that the FEA community has had to rethink (or expand their thinking) concerning mesh quality in the face of anisotropic mesh refinement usage [158, 18, 198], modeling for IA will need to carefully consider the both the geometric and induced function space impact of uniformity and nonuniformity within the representation. In particular, element quality is not the only factor in deciding whether an analysis will succeed, it is necessary to understand the mapping $F : \theta \to \Omega$ as well.

Although it is not widely recognized in practice, we have demonstrated that knot vector choice is important. Although initial knot vectors may be decided during design and not up to the analyst, the methodology of mesh refinement is under the purview of the analyst. Uniform h-refinement may not be an optimal strategy. In higher dimensions the choice of knot vectors is more complicated. Given the tensor product nature of a NURBS, inserting a knot at a certain location might simultaneously produce more favorable results in one region and a less favorable one in others. The use of T-Splines [178] could be the solution to that problem.

3.6.3 Influence of Control Mesh Degeneracies on Normalized Discrete Spectra

In this section we examine how degeneracies in the control mesh affect normalized discrete spectra. During modeling, it can happen that the resulting control mesh of a model contains control points that coincide. In the following, we examine mappings M_1 and M_2 that are cubic mappings with uniformly spaced control points except for a single instance of identical repeated control points occurring about midway along the control point list, a configuration that creates a mesh degeneracy. M_1 has a double control point and M_2 has a triple control point. To illustrate this degeneracy, the reader is referred to Figure 3.12 which shows the basis functions $\psi_{i,\boldsymbol{\tau}_1}(M_1^{-1}(x))$ and $\psi_{i,\boldsymbol{\tau}_1}(M_2^{-1}(x))$.

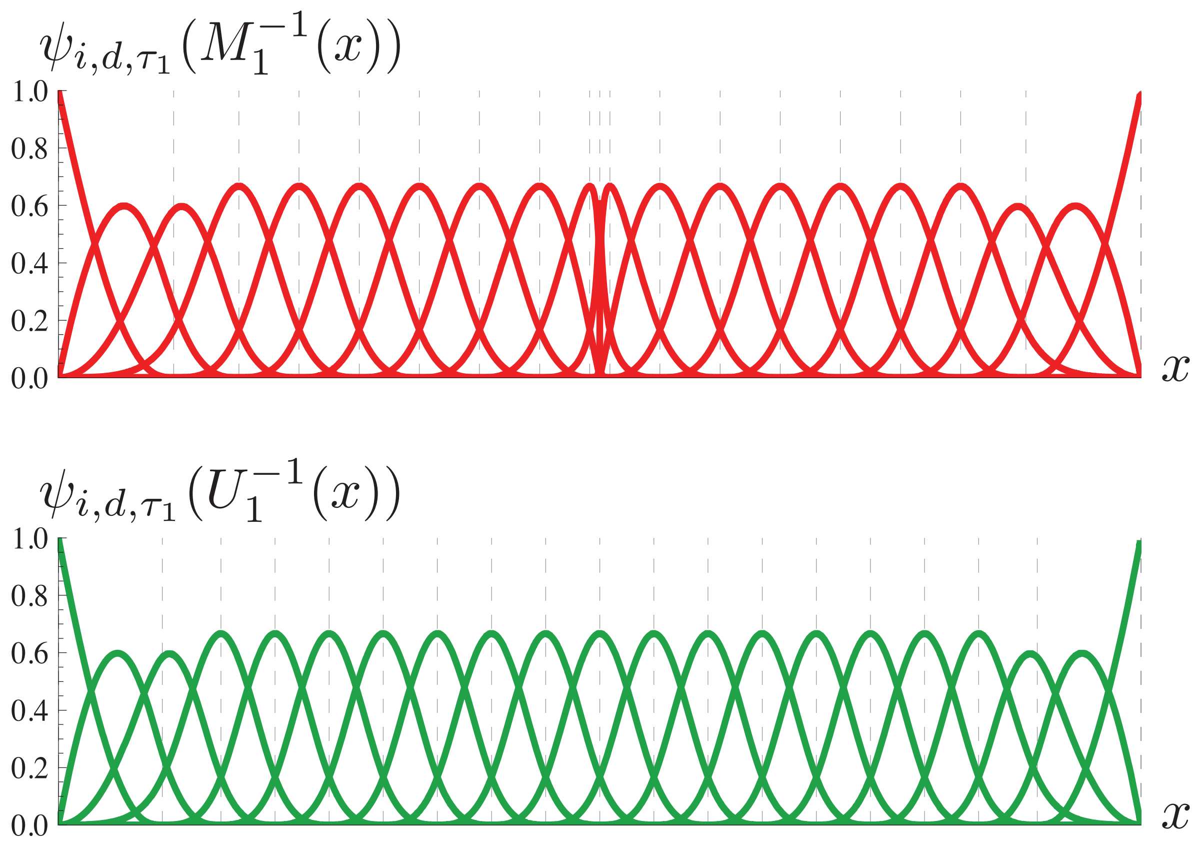
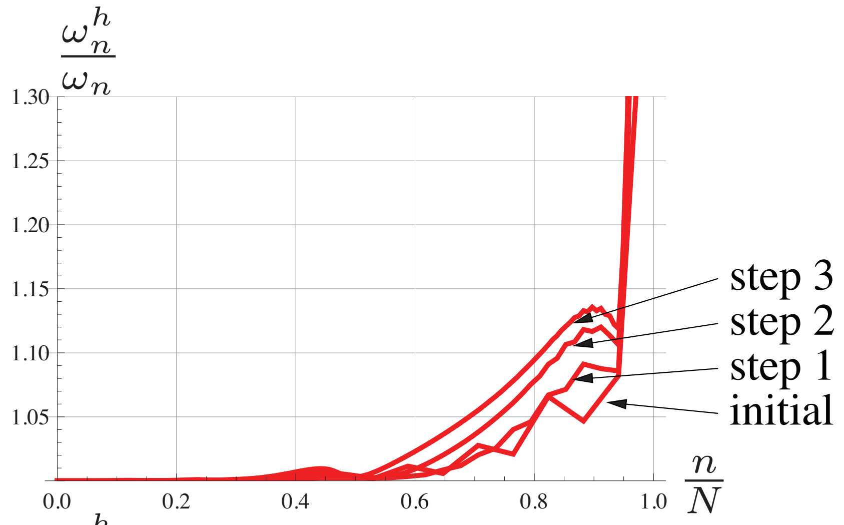


Figure 3.12. The cubic basis functions defined on $\Omega = [0, 1]$ for $V_{M_1,\mathbb{S}_1}$ and $V_{U_1,\mathbb{S}_1}$, respectively. The control points for the M_1 are uniformly spaced, except the control point in the middle is duplicated. $\psi_{i,d,\boldsymbol{\tau}_1}(U_1^{-1}(x))$ are shown for comparison (bottom).

We apply h- and k-refinements to the spaces $V_{M_1,\mathbb{S}_1}$ and $V_{U_1,\mathbb{S}_1}$. This results in spaces $V_{h/(2^m),M_1,\mathbb{S}_1}$ and $V_{h/(2^m),U_1,\mathbb{S}_1}$, respectively, for m steps of h-refinement, and $V_{h/(2^{p-3}),M_1,\mathbb{S}_1,p-3}$ and $V_{h/(2^{p-3}),U_1,\mathbb{S}_1,p-3}$ when the k-refinement raises the degree to p, respectively. Note that the h-refining takes place after each step of the degree elevation, and halves all knot interval spacing. The corresponding normalized discrete spectra are computed. Figure 3.13 shows the results for h-refinement and Figure 3.14 shows the k-refined versions of the two models. The outlier problem is not ameliorated by refinement of either type. Although the normalized discrete spectrum becomes slightly worse under h-refinement, it becomes flatter under k-refinement–except for the outliers.



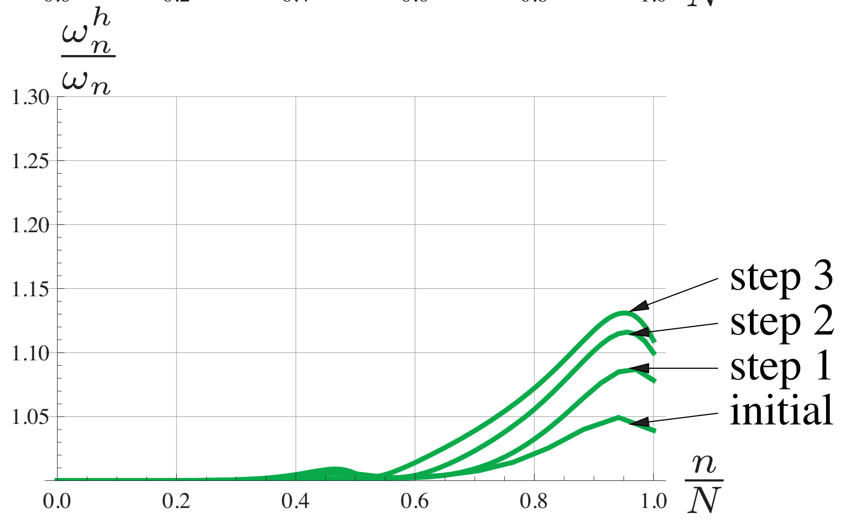


Figure 3.13. h-refinement: For both mappings (degenerate and nondegenerate), the normalized discrete spectra are slightly worse under uniform refinement. The uniformly refined space generated by M_1 always has outliers.

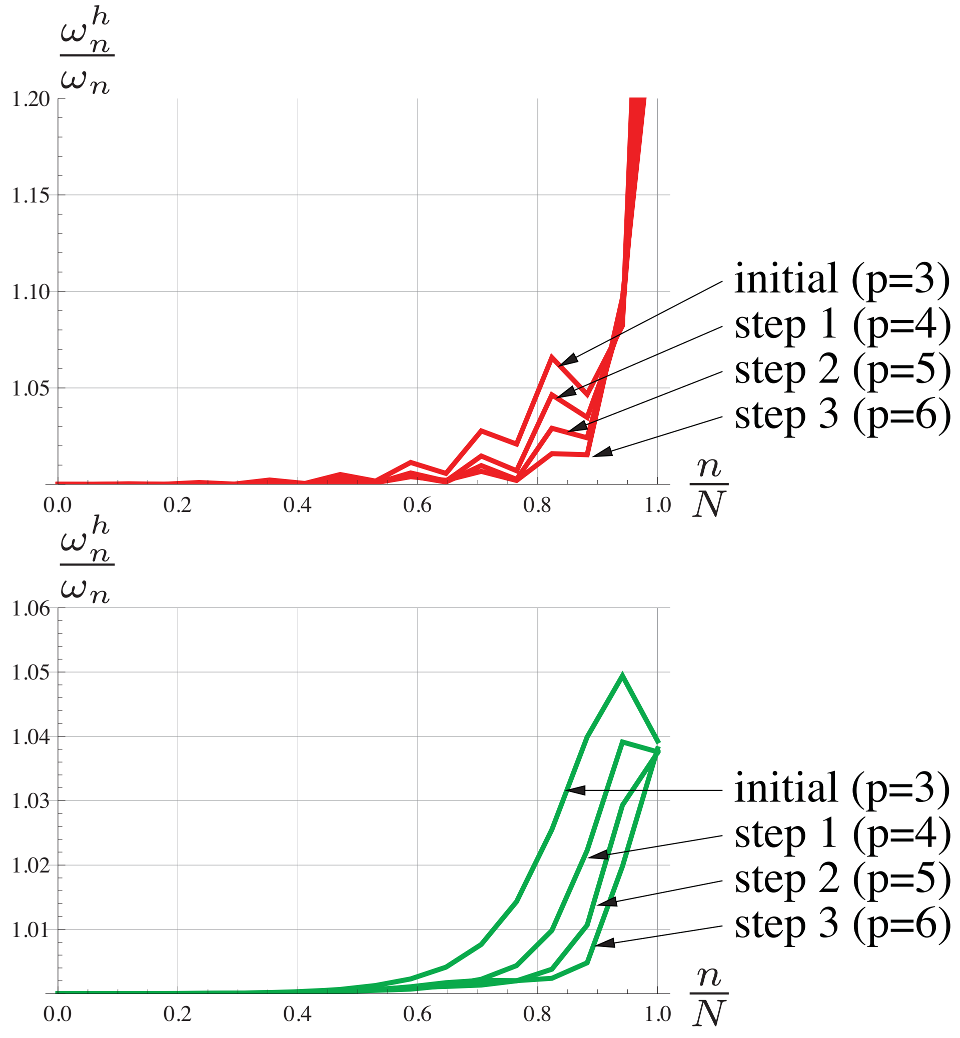


Figure 3.14. k-refinement: For both mappings (degenerate and nondegenerate), the normalized discrete spectra become flatter with elevated degree. However, the outlier exists throughout uniformly refining for the spaces based on M_1.

Investigations of M_1 and its inverse M_1^{-1} (Figure 3.15) indicate possible reasons. The uniform spacing of the coefficients causes the mappings to stretch the boundary elements, creating a mapping that is concave for smaller values of t and convex for values of t close to 1. A multiple control point causes the parameterization to be nonlinear near the nodes corresponding to the multiple control point, pulling the density of the mapping towards that node. Hence the slope of M_1 becomes small near 1/2, and so the slope of M_1^{-1} gets large. So, in this small region the behavior of the Jacobian affects the conditioning of the stiffness matrix to cause larger maximum eigenvalues of the stiffness matrix and the outliers. An embedded 0 tangent direction and inflection point in M_2 (see Figure 3.16), cause M_2^{-1} to have a singularity at the corresponding value. The stiffness matrix conditioning is accordingly worse.

Since the maps M_i are unchanged in both h- and k-refinement, those modes isolate the effects to fewer, smaller elements, but alleviate this problem only to a limited

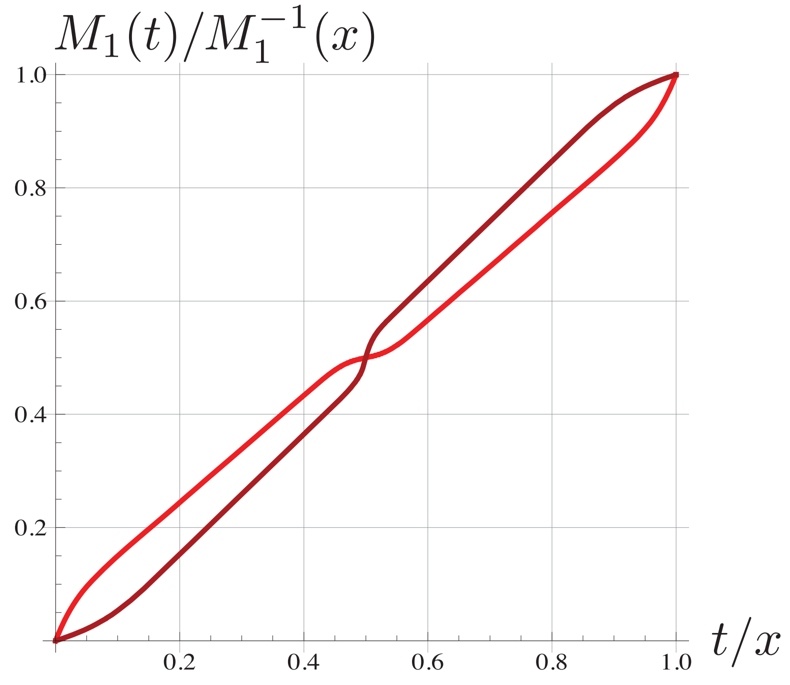
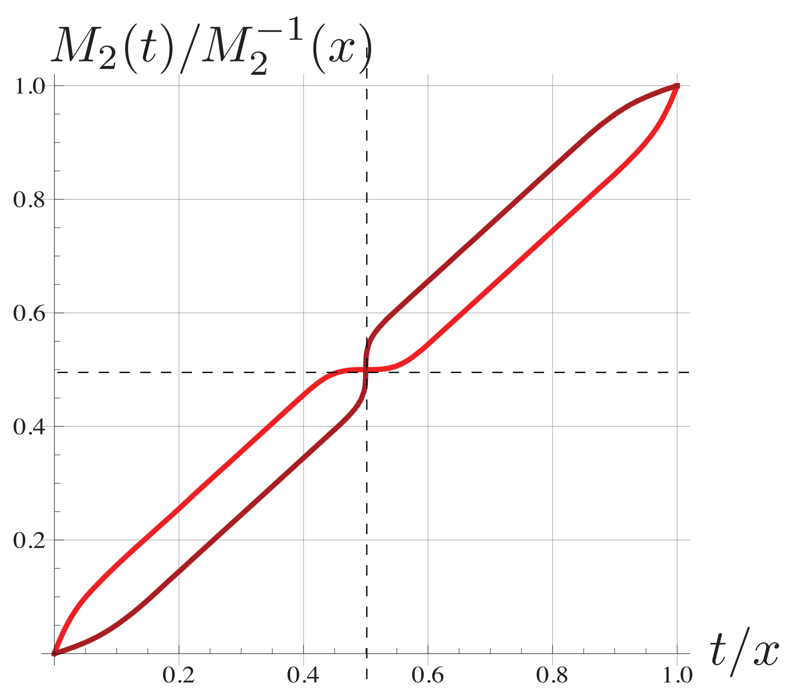


Figure 3.15. Degenerate mapping (red) with control point 1/2 duplicated and corresponding inverse mapping (darker red). The inverse has a high slope and therefore large first derivative at $t = 1/2$ causing a negative impact on the stiffness matrix conditioning (results in larger eigenvalues).



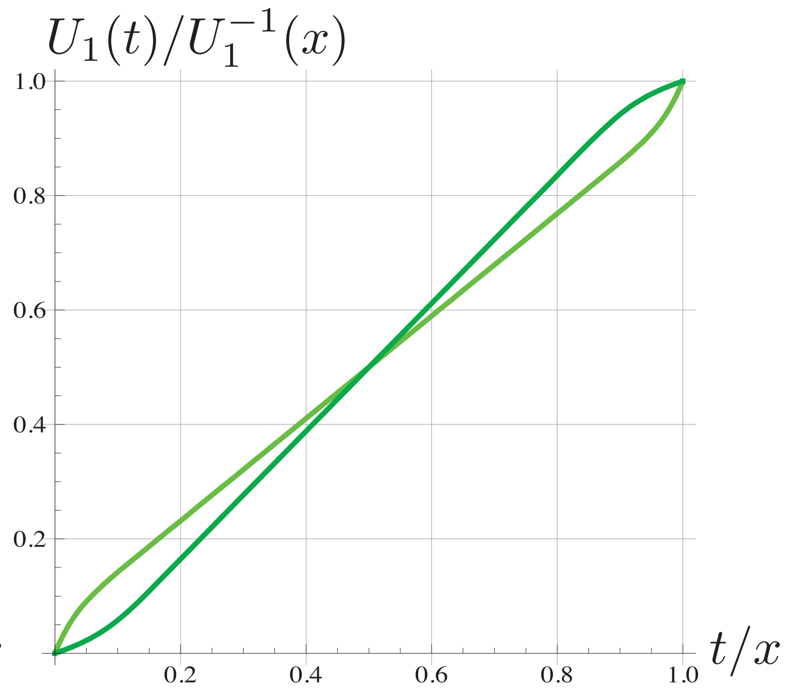


Figure 3.16. A triple control point embeds a horizontal tangent at $t = 1/2$ for M_2. Hence M_2^{-1} is singular at $t = 1/2$. However $t = 1/2$ is a knot so the singularity occurs only at an element boundary. The comparison with U_1 (green) where slope of the inverse (dark green) is constant away from the boundary elements.

extent, as we have seen in the above experiments. Generalized eigenvalue problems seem unsuitable candidates for use with models having control mesh degeneracies. Other analyses that are not sensitive in the same way may be suitable for use with models whose representations have multiple control points.

3.6.4 Drumhead Problem

In this section we solve the 2D version of the generalized eigenvalue problem given in Equation (3.10) on the disk centered at the origin with radius one and generate the normalized discrete spectrum. We use the three different disk representations developed as distinct surface completions to the circular boundary in Section 3.5.3.

The natural frequencies for the drumhead problem on the disk are the zeros of a Bessel function [25]. Using both quadratic and cubic degrees to investigate the effects of the single degree change, this study generates the spectra for D_1 and D_3. Thus, let $D_{i,j}$ indicate the i-th representation in Figure 3.6, and j indicate the degree.

Figure 3.17 shows normalized spectra for the mappings $D_{i,j}$, all of which have undergone h-refinement using a uniform knot vector. It can be seen that $D_{3,j}$ produces a

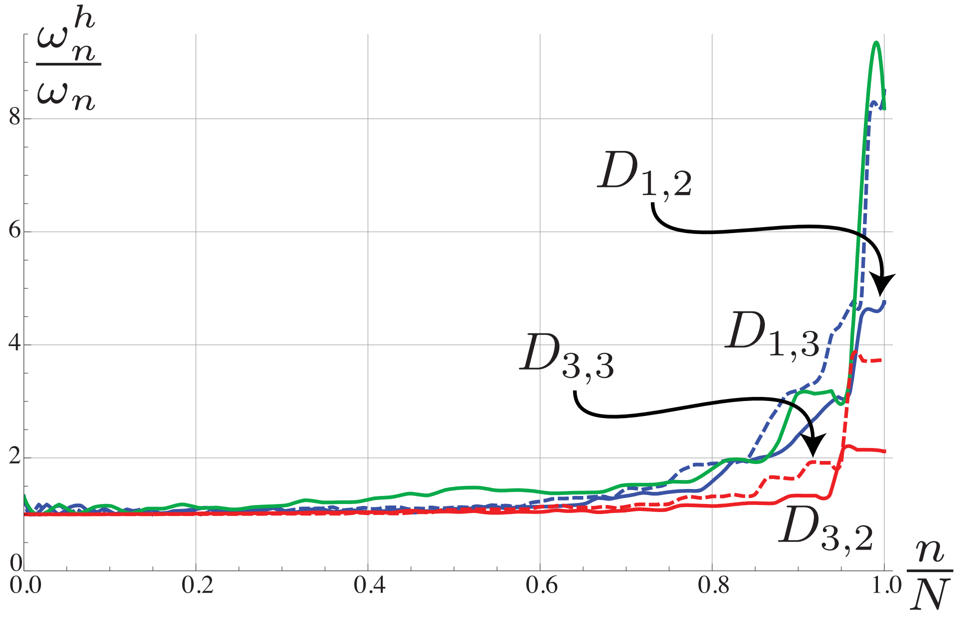


Figure 3.17. Normalized discrete spectra for different representations of a disc.

much flatter curve than the mappings $D_{1,j}$ and $D_{2,j}$. It also can be seen that an elevated degree has a negative impact on the result. The mapping $D_{2,3}$ performs the poorest. We speculate that this is related to nonuniform parameterization, because the elements that result under uniform h-refinement do not have uniform size and cannot represent the uniform spectral behavior as well. By referring to Figure 3.18 the spectrum for the mapping $D_{3,2}$ is computed using a nonuniform h-refinement. The refinement process creates the elements for refined knot vector $\boldsymbol{\tau}_2$ from Section 3.5.2 in both parametric directions. The effect is to have larger elements near the circular boundary. The result is a flatter spectrum which has a maximum ratio of about 1.5, compared to approximately 2.2 obtained through uniform refinement.

3.6.5 Poisson Equation on 2D Domains

In this section we solve the Poisson equation (3.9) over four domains $\Omega_i \in \mathbb{R}^2, i = 1, 2, 3, 4$ Ω_1, the unit square; Ω_2, the square with $4\times$ magnification; Ω_3, the unit disc

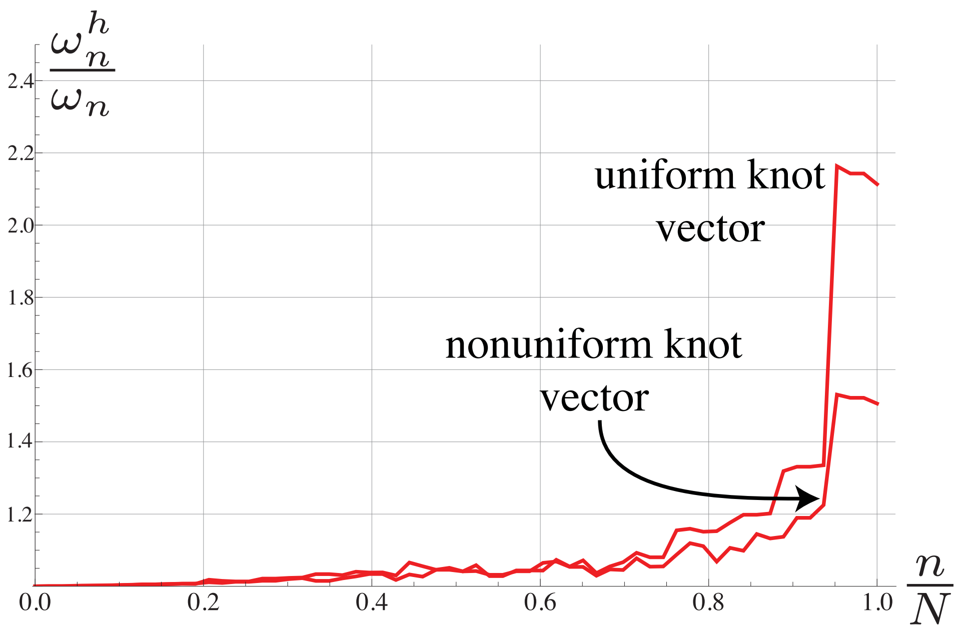


Figure 3.18. The disc model $D_{3,2}$ with a nonuniform knot vector, where refining knots are chosen as to create $\boldsymbol{\tau}_2$ (Section 3.5.2). It yields a flatter discrete spectrum than the normalized spectrum showed in Figure 3.17.

and Ω_4, the quarter annulus having inner radius one and outer radius of two. Although every domain is represented with an exact NURBS model, the completions from the boundary representations to the interiors differ. We consider relative aspects of mesh quality. We find a sequence of approximations u_h of the unknown solution u, such that $\lim_{h\to 0} ||u - u_h|| = 0$. Thus, h controls the approximation quality and is related to the knot spacing, element size, and V_h.

We represent Ω_1 with two bicubic models $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, each with degree $\boldsymbol{d} = (3,3)$ and knot vectors $\boldsymbol{T} = (\boldsymbol{\tau}_1, \boldsymbol{\tau}_1)$, where $\boldsymbol{\tau}_1$ the uniform open knot vector defined in Section 3.5.2. The coefficients for $\boldsymbol{S}_1$ are taken from the mapping U_1 and are uniformly spaced in each parametric direction. $\boldsymbol{S}_2$ uses coefficients from the identity map $\mathcal{I}$, and uses nodal representation to form the identity map on the unit square. Specifically, we define

$$\boldsymbol{S}_1(\boldsymbol{t}) = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} \left(\frac{i_1 - 1}{n_1 - 1}, \frac{i_2 - 1}{n_2 - 1}\right) B_{\boldsymbol{i},\boldsymbol{d},\boldsymbol{T}}(\boldsymbol{t}) \qquad \boldsymbol{S}_2(\boldsymbol{t}) = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} \left(\tau^*_{1,i_1}, \tau^*_{1,i_2}\right) B_{\boldsymbol{i},\boldsymbol{d},\boldsymbol{T}}(\boldsymbol{t}). \quad (3.40)$$

As in the 1-D examples, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ differ in the choice of control points which results in the identity map or linear map $\boldsymbol{S}_2$, i.e., $\boldsymbol{S}_1(\boldsymbol{t}) = (\boldsymbol{t})$ and in the nonlinear map $\boldsymbol{S}_2$. Figure 3.19 shows the elements for $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ for $n = m = 10$.

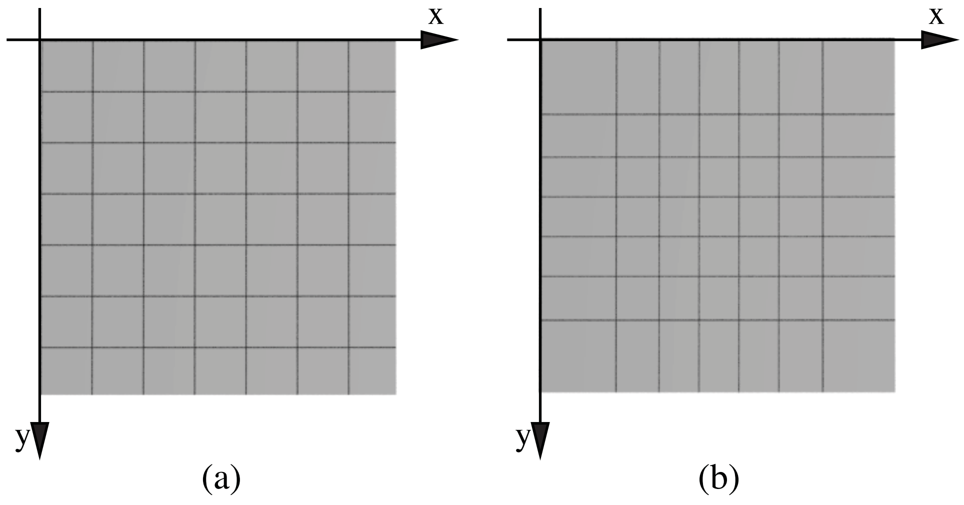


Figure 3.19. Both NURBS models represent Ω exactly. $\boldsymbol{S}_2$ on the left is the identity map; the control points in $\boldsymbol{S}_1$ on the right are uniformly spaced.

Here the region Ω_2 is represented by the models $\boldsymbol{S}_3$ and $\boldsymbol{S}_4$, which are scaled and translated versions of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively.

Then, Ω_3 is represented with the three different disc models D_1, D_2 and D_3, as in Section 3.5.3.

Finally, Ω_4 is represented using three different completions, $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$, shown in Figure 3.20. All mappings are cubic in both parametric directions and use the same open knot vectors with interior knot multiplicity of two, i.e., all mappings are C^1. We use $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ to consider the differences between model quality criteria for meshes and how they apply to isogeometric models. They are generated from $\boldsymbol{A}_1$. The interior control points for $\boldsymbol{A}_2$ are slightly perturbed from those used in $\boldsymbol{A}_1$, creating analysis elements with wiggly boundaries. This situation could potentially arise from data fitting algorithms or noisy data. Interior control points for $\boldsymbol{A}_3$ are chosen so that the knot line curves are orthogonal where they cross. We still create perturbations so that element boundaries have some wiggles. Since one quality measure for isogeometric FEA meshes is that the element boundaries are orthogonal where they meet, this example sets up a similar scenario, but in the context of boundaries with wiggly sides.

In these studies we investigate Poisson's equations whose solutions are smooth analytical functions $u_i : \Omega_j \to \mathbb{R}$, where $i, j = 1, 2, 3, 4$ and defined as,

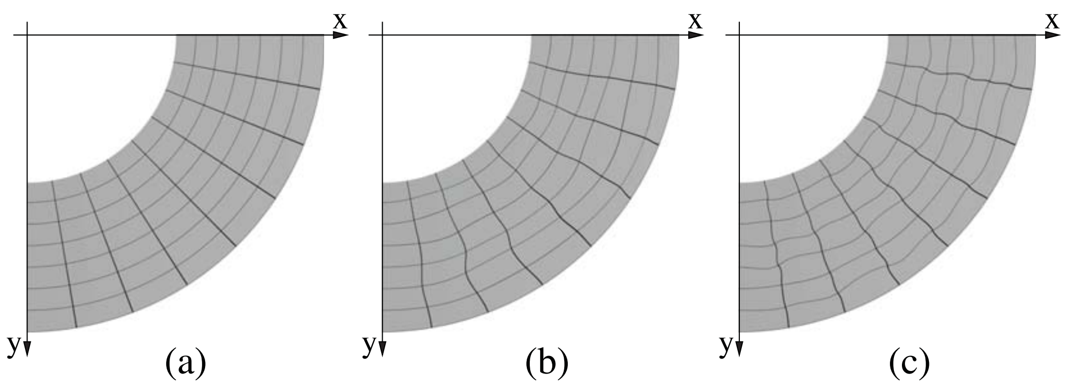


Figure 3.20. Three exact representations of the boundary of a quarter annulus with an inner radius of one and an outer radius of two with different completions. In (b), control points in the interior are slightly perturbed. In (c), element boundaries are orthogonal where they cross.

$$u_1(x,y) = \sin(2\,\pi\,x)\,\sin(2\,\pi\,y), \tag{3.41}$$

$$u_2(x,y) = \frac{25\,(-\frac{1}{e^{100}} + e^{-12.5\,(x^2+y^2)})}{2\,\pi}, \tag{3.42}$$

$$u_3(x,y) = J(4, J_{zero}(4,2)\,r(x,y))\,\sin(4\,\theta(x,y)), \tag{3.43}$$

$$u_4(x,y) = \sin(2\,\pi\,r(x,y))\,\sin(2\,\pi\,b(x,y)), \tag{3.44}$$

where

$$r(x,y) = \sqrt{x^2+y^2} - 1 \qquad b(x,y) = \frac{2\,\arccos(x/\sqrt{x^2+y^2})}{\pi} \tag{3.45}$$

and, $\theta(x,y)$ defines the angle between the vector (x,y) and the Cartesian coordinate axes. In other words $r(x,y)$ and $\theta(x,y)$ convert the Cartesian coordinate (x,y) into the polar coordinate $(r(x,y), \theta(x,y))$. Furthermore, $J(n,z)$ is the nth Bessel function of the first kind at $z \in \mathbb{R}$, and $J_{zero}(n,m)$ is the mth zero of the nth Bessel function of the first kind. u_3, selected to be the $(4,2)$-th mode of vibration of a drumhead [25], evaluates to zero on the boundary of Ω_3.

In experimenting we formulated five test cases involving solving the Poisson equation (3.11). They are shown in Table 3.1.

Cases 1, 2, 4, and 5 use $u_i|_{\partial\Omega} \equiv 0$. Nonzero Dirichlet boundary conditions for Case 3 are chosen by a nodal interpolation of u_1.

We use uniform h-refinement, placing a new knot in the middle of every parametric knot span. The elements for building the approximate solution then are split in turn. The shape of the new elements is not an exact half split of the old because the mappings

Table 3.1. Test cases for Poisson equation.

case	physical space	ideal solution	mapping
1	Ω_1	u_1	$\boldsymbol{S}_1$, $\boldsymbol{S}_2$
2	Ω_2	u_2	$\boldsymbol{S}_3$, $\boldsymbol{S}_4$
3	Ω_3	u_1	D_1, D_2, D_3
4	Ω_3	u_3	D_1, D_2, D_3
5	Ω_4	u_4	A_1, A_2, A_3

are not affine. Suppose there are m_ℓ control points after ℓ levels of h-refinement. The isogeometric solution after ℓ h-refinements to u_i is denoted $\widehat{u}^{\ell}_{j,i}$ for test case j result. The error is computed as

$$\epsilon_{\infty,j,i,\ell} = \sup_{\boldsymbol{x}\in\Omega}|\widehat{u}^{\ell}_{j,i}(\boldsymbol{x}) - u_i(\boldsymbol{x})| \ . \tag{3.46}$$

A piecewise linear convergence curve is defined by the points $(\sqrt{m_\ell}, \epsilon_{\infty,j,i,\ell})$. In the following discussion, *good* or *poor* convergence means that the negative slope of the convergence curve is higher or lower, respectively, compared to the convergence of a different mapping. While the analysis solutions converge, as predicted, under h refinement, these studies are aimed at determining how many levels of refinement are necessary before the asymptotic behaviors dominate. If the same quality result can be obtained on a simpler mesh, that situation is always desirable. We discuss each test in more detail. We refer the reader to Figures 3.21 through 3.23 which show log-log plots of the convergence curves for the five performed tests discussed in the following. For purposes of illustration, next to each log-log plot the exact solution u_i on the respective domain Ω_n is shown.

3.6.6 Test 1

Initially, both mappings have almost the same error. During the initial levels of refinement the linear map $\boldsymbol{S}_2$ has a better convergence rate than the nonlinear map $\boldsymbol{S}_1$. Then, both mappings converge with the same rate, i.e., the convergence curves become parallel. The error curve of $\boldsymbol{S}_2$ is below the curve of $\boldsymbol{S}_1$ by the respective offset, which means that the error of $\boldsymbol{S}_2$ is about an order of 10 smaller than the corresponding error of $\boldsymbol{S}_1$. So the convergence behavior of the Poisson problem is better under the identity map, differing from the better convergence behavior for $\boldsymbol{S}_1$ for the eigenvalue problem.

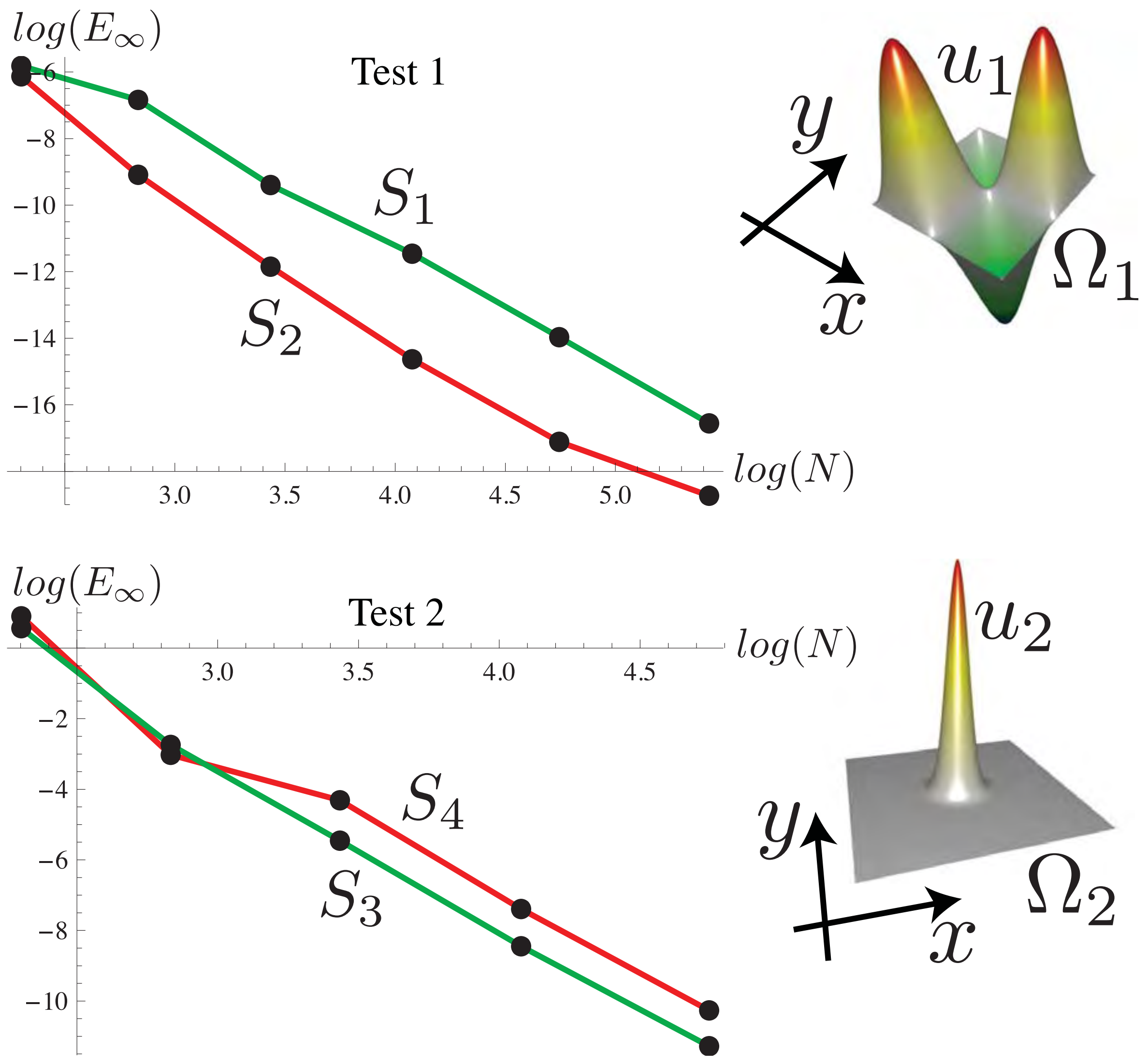


Figure 3.21. Poisson test 1 and 2. The plots on the left show the error curves for different mappings, on which the exact solution on the respective domain is shown on the right.

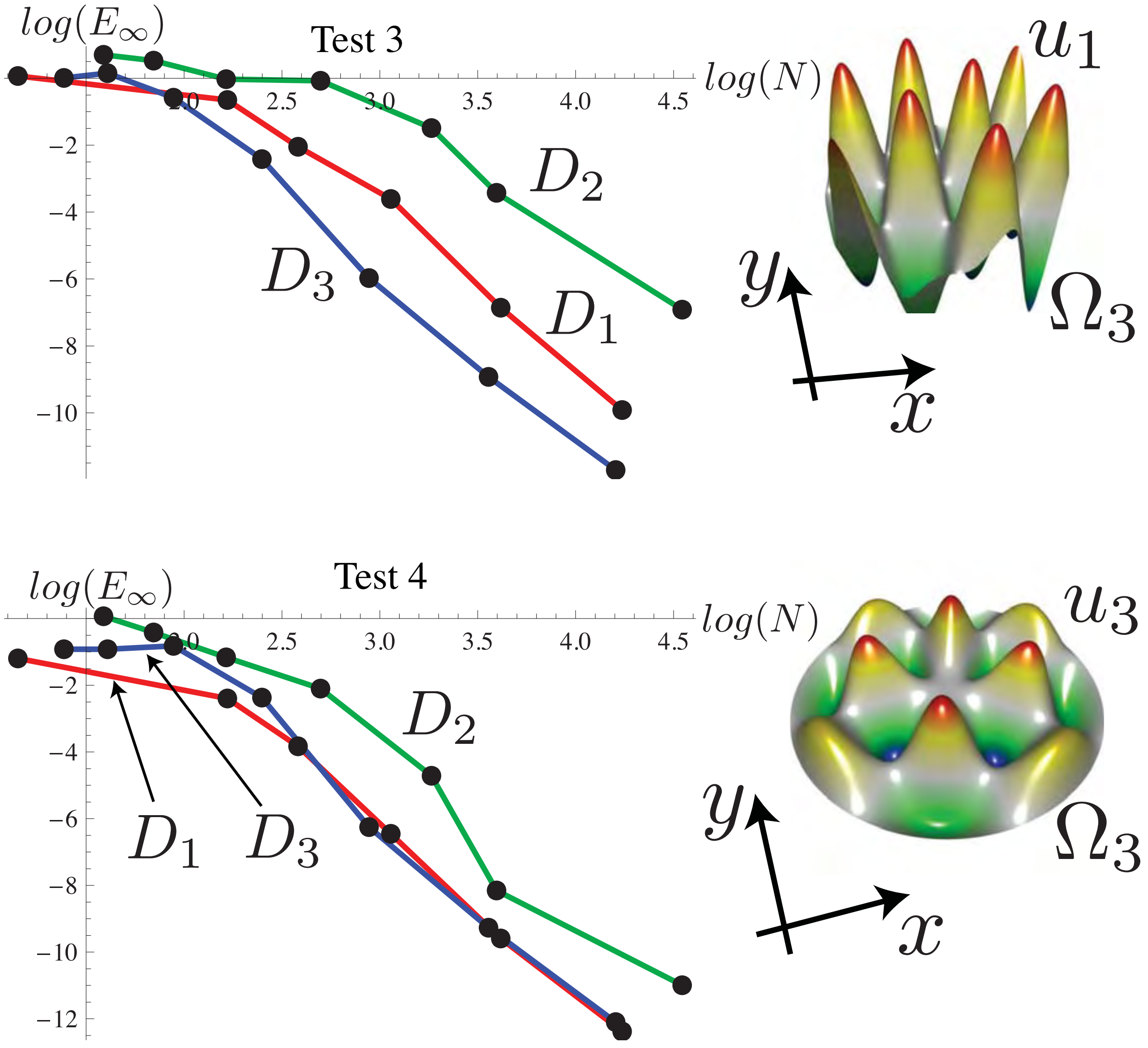


Figure 3.22. Poisson test 3 and 4. The plots on the left show the error curves for different mappings, on which the exact solution on the respective domain is shown on the right.

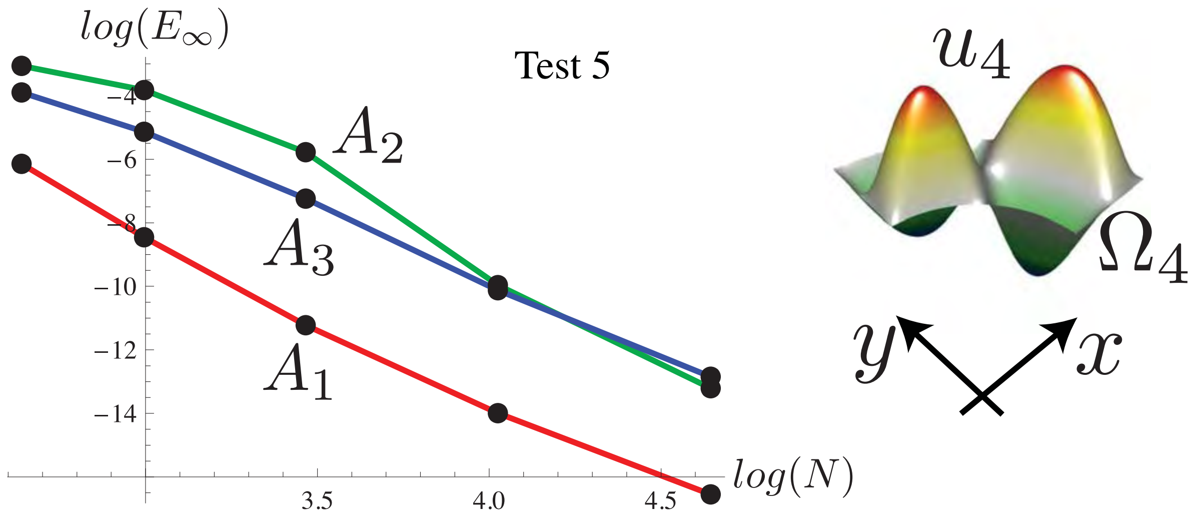


Figure 3.23. Poisson test 5. The plots on the left show the error curves for different mappings, on which the exact solution on the respective domain is shown on the right.

3.6.7 Test 2

A similar observation can be made for this test. Initially, in the second refinement step, both mappings have the same error. Then, in the subsequent refinement step, the error for the linear map $\boldsymbol{S}_4$ decreases more slowly. However, in further refinement steps, both error curves have the same slope. For this problem, contrary to Test 1, the nonlinear map $\boldsymbol{S}_3$ performs better. This is understandable inasmuch as the approximating space has greater basis function density near the center of the regions.

3.6.8 Test 3

During the first refinement steps, all three NURBS models converge rather slowly, although D_1 exhibits the least error. After about the fourth refinement step, the error for D_3 drops the most, and drops the least for D_2. Thereafter, all mappings converge at the same rate, where the error of D_1 is about an order of ten larger than D_3 and D_2 is about four orders of ten larger than D_3. This case has nonzero boundary conditions. Given the same number of functions to approximate u across all three mappings, D_3 will have a higher percentage that are nonzero on the boundary and so can be used to better approximate the true solution on the boundary. Each time both parameters are uniformly refined, D_1 only doubles the number of nonzero basis function on the boundary. But D_3 quadruples that number. So, for D_1 it would be better to h-refine more in the polar direction than in the radial direction to better approximate the boundary.

3.6.9 Test 4

The error curves look similar to those in Test 3. Initially, all mappings converge slowly; however, the curves of D_1 and D_3 start to overlap, while D_2 lags a bit to reach the same rate of convergence.

3.6.10 Test 5

In the final test, the initial errors for all mappings are different, with A_1 having the smallest error. Except for an initial slower convergence with A_2, all three mappings soon converge at the same rate, where the offsets between A_1 and A_2, and between A_1 and A_3 are considerable. In this case, the wiggles contributed to the problems for A_2 and A_3, the better behavior near the element boundary corners for A_3 alleviates some of the problems. So, although not necessary for convergence, the smoother sided elements of A_1 appear to help convergence.

Note that models $\boldsymbol{S}_1$ and $\boldsymbol{S}_3$ have the same stretching properties. The same holds for $\boldsymbol{S}_2$ and $\boldsymbol{S}_4$, where no stretching occurs due to the linear mapping properties. However, in Test 1, the linear map $\boldsymbol{S}_2$ performs better than the nonlinear mapping. In Test 2 the nonlinear map $\boldsymbol{S}_1$ performs better than $\boldsymbol{S}_2$. The reason for that lies in the different properties of the known solutions u_1 and u_2 as shown in Figures 3.21 and 3.22. u_2 is everywhere zero but has a peak around the origin. $\boldsymbol{S}_3$ performs better because more elements are pulled into the center of the domain (see Figure 3.19 (left)). Similar to the 1D case in Section 3.6.2, having the determinant of the Jacobian have value near 1 does not necessarily mean that a mapping performs better than a mapping which involves distortion, at least for issues not related to conditioning of the stiffness matrix. In the case of Test 1, however, the stretching leads to the neglect of other important regions, and therefore $\boldsymbol{S}_2$ performs better, because it more uniformly distributes the elements over the domain, a characteristic that treats the known function u_1 more favorably.

In the first three tests all mappings have roughly the same initial error, i.e., the convergence curves emerge from roughly the same starting position. This is not the case for Tests 4 and 5. In Test 4, D_1 has the smallest initial error, in Test 5, A_1 has the smallest error. Even though at a certain refinement stage, all mappings have the same asymptotic error estimation, the initial error shows that if a model is created with care, refinement steps can be saved, attaining better analysis with fewer elements. In case of Test 5, with the given true solution, wiggles or other perturbations have a negative impact on the initial error. Note that wiggles and other perturbations which were artificially introduced in our test models are common modeling artifacts and often unintentionally occur in practice during data-fitting.

3.6.11 Linear Elastic Deformation of a Volumetric Model

As a final example we examine the isotropic linear elastic deformation of a human femur. The femur is modeled with the methodology proposed in [125] and introduced in Section 3.5.6. The modeling input is an exterior and interior boundary triangle mesh, where the volume between the exterior and interior represents the cortical bone and the volume of the interior boundary represents the trabecular bone. From that, a single B-spline volume with proven approximation power is created. The model is C^2 but has a degeneracy along the cylindrical axis. The placement of the axis is user-guided, chosen depending on the model and simulation parameters. Such a model is difficult to decompose in multiple patches for the following reasons. Patch boundaries may not be planar, and gluing them with certain continuity is difficult. Furthermore, since the object consists of different materials it is not clear how to represent it with multiple patches. Creating a single B-spline volume introduces more deformation in the geometry, but it avoids both problems, i.e., it does not involve any patch gluing, and respecting material attributes in the parameterization can be achieved without tremendous effort. Another innoative aspect of this model is that the angular parameterization is periodic. Unlike all previous modeling studies done for IA, the resulting representation takes advantage of the characteristics of periodic B-splines to ensure that the model is C^2 everywhere. Even more important, it means that V_h has only C^2 functions except at the natural bone boundary and the skeletal axis.

Both parts, i.e., the cortical part and trabecular part, use a Poisson ratio of 0.9. A Young's modulus of 17×10^9 (N/m^2) is applied to the more solid cortical part; 100×10^6 (N/m^2) is applied to the trabecular part. The boundary between the cortical and trabecular materials is an isoparametric surface of the trivariate B-spline representation.The bottom of the femur is held rigid in the x, y and z directions. Load is distributed over the apex of the femur in the direction of the base of the femur. Figure 3.24 portrays the femur where magnitudes of the displacement are visualized.

Figure 3.24 also shows h-refined versions r_i of the femur. r_0 is the initial representation which is a data-reduced version from the original representation. r_1 and r_0 are created by uniform h-refinement in u, where r_1 is created from r_0 and r_2 is created from r_1. r_3 and r_4 (not shown) are created by uniform h-refinement in v and w, respectively. Linear elasticity is applied to each of the representations with the same material parameters and boundary conditions resulting in the five solution representations s_i. s_i is compared to s_{i+1} by applying the respective refinement to s_i. The error $\boldsymbol{e}_i$ is the maximum L2-norm of the difference of the coefficients between s_i and s_{i+1}: $\boldsymbol{e} = \{1 \times 10^{-4}, 2 \times 10^{-5}, 3.8 \times 10^{-6}, 2 \times 10^{-7}\}$.

3.6.12 Result Summary

In Section 3.6.2 we discussed different knot vector and mesh choices for the longitudinal vibrations of a 1-D rod. It was shown that nonuniform knot vectors can improve the results and avoid outliers. Then, we examined the 2-D version of this problem, the drumhead problem, completing the disk with different representations of the boundary. While none of the completions would have any effect on the design of the shape, the completion had strong effects on the normalized discrete spectrum.

Then, in Section 3.6.5 the Poisson equation was solved on different domains for different smooth known functions, where the domains were exactly represented with alternative choices of more or less common NURBS models. As shown in [16], sufficient h-refinement eventually converges to these cases where there is a starting NURBS volumetric model that meets certain criteria on the knot spacing and the Jacobian

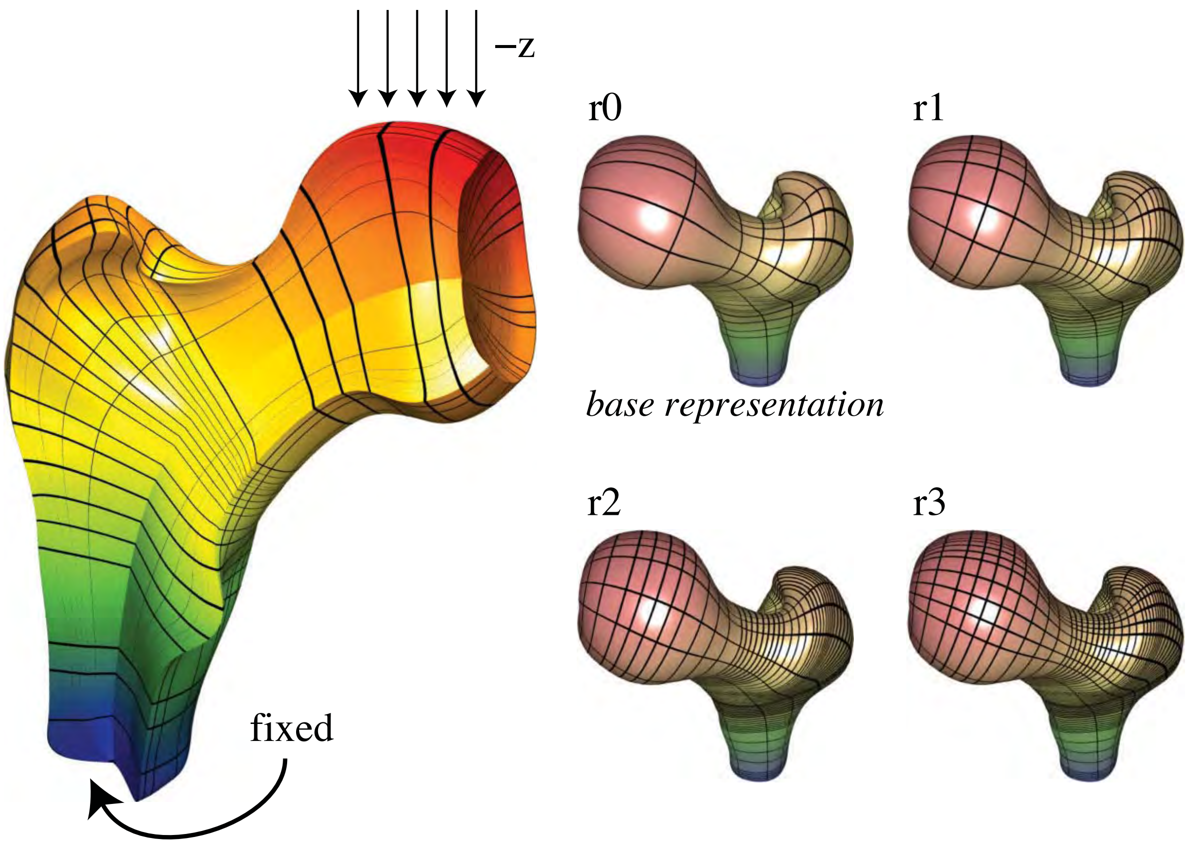


Figure 3.24. Linear elastic deformation of a femur.

of the mapping. That is, no matter what regular mapping was initially chosen to represent a domain Ω, eventually the error decreases at the same rate. However, the refinement level at which a mapping reaches the asymptotic behavior and the resulting offset compared to other models, initially chosen by the modeler, allows us to make judgements about the quality of the model which could be observed in the above test cases. By a more careful model design, a design appropriate for analysis, computation time and efforts to refine a model can be saved.

We conclude that boundary modeling and boundary completion techniques can enhance the speed of convergence and quality of the results.

3.7 Summary and Conclusions

IA has demonstrated itself as a paradigm that may actually successfully bridge CAD modeling and FEA analysis. With analysis tools able to act natively upon the same mathematical building blocks employed in the modeling community, there exists a realistic chance that a seamless pipeline based on a shared representation might be in the future. It is important though to appreciate that the constraints under which these two communities currently work, and will continue to work, are sometimes complementary, often different, and occasionally competing. In this chapter, we have attempted to provide insight into the modeler's perspective on the process of model design and construction, and to demonstrate and highlight explicitly that choices that arise within the modeling process may have consequences downstream the line when analysis is performed on an isogeometric model. We detail several of the outstanding issues and considerations within modeling and at the interface of modeling and analysis. These fundamental problems must be explored and addressed as the area of IA moves forward. We advocate a new area of research – *analysis-aware modeling* – by which modelers become cognizant of how their modeling choices impact the quality of analysis, and hence can incorporate this knowledge into the balancing act of design considerations and constraints that the modeler is already juggling. Several modeling methodologies are proposed in the subsequent chapters. IA is a superb vehicle to promote the marriage of CAD and FEA as it represents *modeling-aware analysis*. We hope to have demonstrated that there is a correspondingly important and symmetric need for the modeling community to reciprocate in developing analysis-aware modeling. We conclude by emphasizing that both can only be done through the continual interaction and dialog between the two communities.

CHAPTER 4

CYLINDRICAL-LIKE OBJECTS WITH SHAPE OVERHANGS

This chapter presents a methodology based on discrete volumetric harmonic functions to parameterize a volumetric model in a way that it can be used to fit a single trivariate B-spline to data so that simulation attributes can also be modeled. The resulting model representation is suitable for IA [87]. Input data consist of both a closed triangle mesh representing the exterior geometric shape of the object and interior triangle meshes that can represent material attributes or other interior features. The trivariate B-spline geometric and attribute representations are generated from the resulting parameterization, creating trivariate B-spline material property representations over the same parameterization in a way that is related to [127] but is suitable for application to a much larger family of shapes and attributes. The technique constructs a B-spline representation with guaranteed quality of approximation to the original data. Then we focus attention on a model of simulation interest, a femur, consisting of hard outer cortical bone and inner trabecular bone. The femur is a reasonably complex object to model with a single trivariate B-spline since the shape overhangs make it impossible to model by sweeping planar slices (Figure 4.1). The representation is used in an elastostatic IA, demonstrating its ability to suitably represent objects for IA.

A frequently occurring problem is to convert 3D data, for instance image data acquired through a CT-scan, to a representation on which physical simulation can be applied as well as for shape representation. Grids or meshes, based on primitives like triangles, tetrahedra, quadrilaterals and hexahedra are frequently used representations for both geometry and analysis purposes.

Mesh generation software like [186] generates an unstructured tetrahedral mesh from

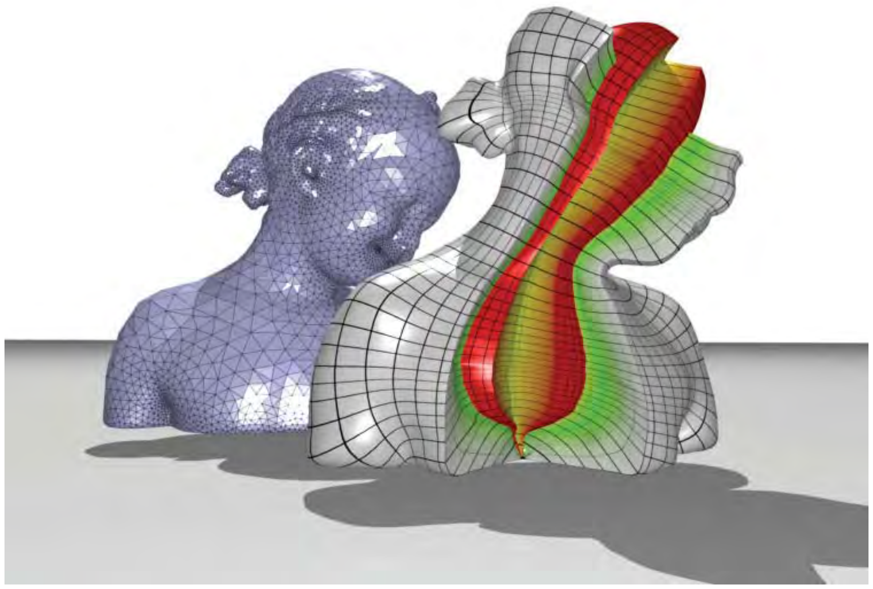


Figure 4.1. Bimba input. (a) triangle mesh of Bimba statue; (b) corresponding trivariate B-spline, where interior represents material information used in simulation.

given input triangle meshes. Unstructured grids modeling techniques [84] improve the modeling and rendering of multidimensional, physical attributes of volumetric objects. However, unstructured grid techniques have drawbacks and certain types of simulation solvers [40] have a preference for structured grids. Creating a structured quadrilateral surface representation and an integrated structured hexahedral internal volume representation from unstructured data is a problem that has undergone significant research. Though topologically limited, structured grids have advantages– especially with growing mesh sizes. For instance, simulation like linear elasticity, multiresolution algorithms like wavelet decomposition or multiresolution editing can be efficiently applied to them. Such structured hexahedral meshes are highly prized in many types of finite element simulations, and generally still require significant manual interaction.

Generating a structured hexahedral grid, parameterizing the volume, and generating a suitable trivariate B-spline model from unstructured geometry and attributes representing a generalized cylinder is the main focus of this chapter.

The contributions in this chapter include

- a framework to model a single trivariate B-spline representation from an exterior boundary, and possibly interior boundaries that have the same genus as the exterior boundary. The boundaries are triangle surfaces, representing geometry or material information, possibly generated from image data.

- a technique to create a trivariate B-spline that has a consistent parameterization across given isosurfaces.

- demonstration of the framework on real unstructured data, a femur obtained through a CT-scan and apply stress simulation to it (see Figure 4.2). A femur consists of a cortical bone, with high densities, and an interior part consisting of a porous, trabecular bone. The transition between cortical and trabecular part is smooth, making trivariate B-splines a candidate to model such a scenario.

4.1 Preliminaries and Notation

In this work we define volumetric harmonic functions over an input triangular boundary or tetrahedral mesh and generate a volumetric parameterization of the model. Then, a trivariate B-spline is fit to the data with parameters that measure error. The following sections briefly recall B-spline definitions and properties of harmonic functions and ways to solve them over triangle and tetrahedral meshes.

4.1.1 Tensor-Product B-splines

A B-spline volume, or a trivariate tensor-product B-spline volume is a mapping $V : [0,1]^3 \to \mathbb{P}^3$ that can be formulated as

$$V(u,v,w) = \sum_{i_0,i_1,i_2=0}^{n_0,n_1,n_2} P_{i_0,i_1,i_2} B_{i_0,p_0}(u) B_{i_1,p_1}(v) B_{i_2,p_2}(w).$$

where the $P_{i_0,i_1,i_2} \in \mathbb{R}^3$ are the control points of the $(n_0+1)\times(n_1+1)\times(n_2+1)$ control mesh, having basis functions B_{i_j,p_j} (defined in [37]) of degree p_j with knot vectors $T^j = \{t_i^j\}_{i=0}^{n_j+p_j}$ for $j = 0,1,2$.

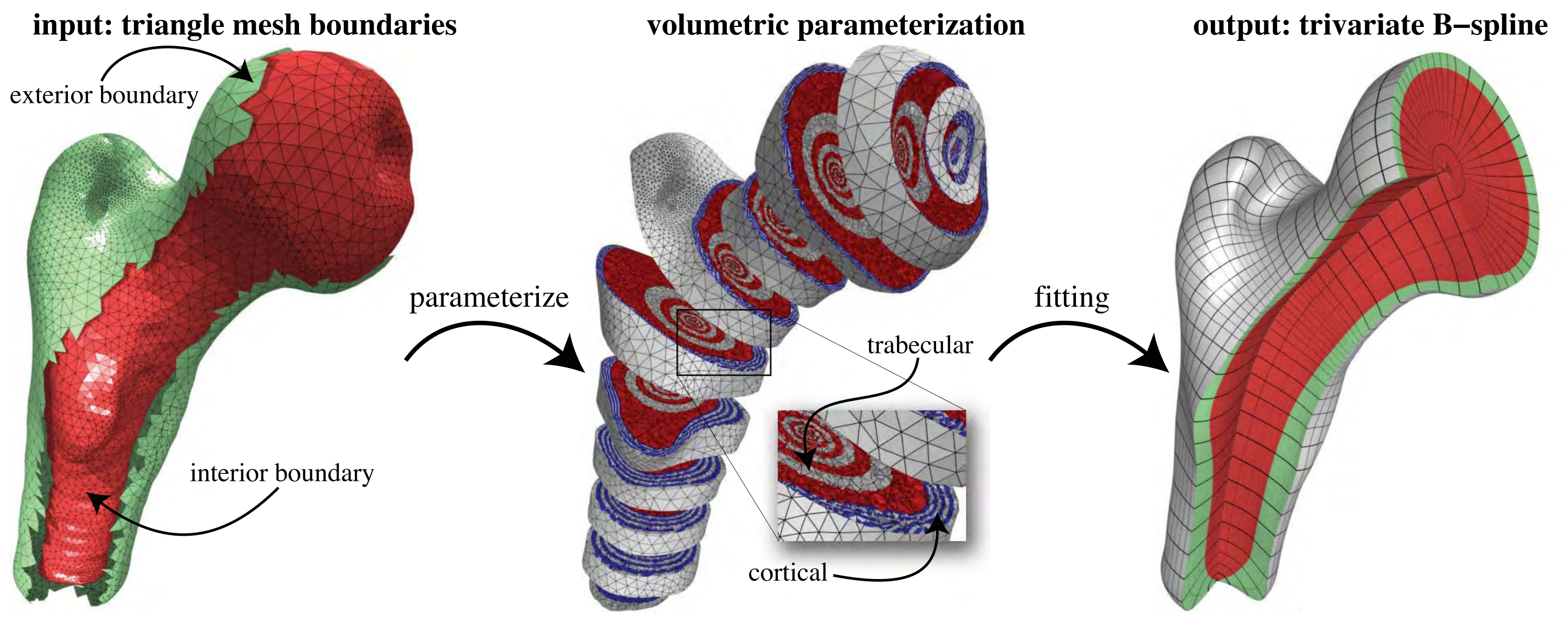

input: triangle mesh boundaries
volumetric parameterization
output: trivariate B−spline
exterior boundary
interior boundary
parameterize
fitting
trabecular
cortical

Figure 4.2. A femur consists of two materials: The outer solid part, or cortical bone, represented by the volume between the input triangle meshes; and the inner soft part, or trabecular bone, represented by the volume of interior triangle mesh (red). These volumes are parameterized (middle) and a single trivariate tensor product B-spline is fitted against it (right), respecting the input triangle meshes in its parameterization. This makes it easier to specify respective material properties. Black isolines represent knotlines in the trivariate parameterization.

4.1.2 Discrete Harmonic Functions

Given a domain $\Omega \in \mathbb{R}^n$, where in our case $n = 2$ and $n = 3$, as discussed in Chapter 1 a harmonic function is a function $\mathbf{u} \in C^2(\Omega)$, $\mathbf{u} : \Omega \to \mathbb{R}$, satisfying Laplace's equation, as defined in Equation 1.1.

Harmonic functions satisfy the maximum principle, namely they have no local minima/maxima and can therefore be used as Morse functions [132, 143]. Also, this property makes them suitable to create a tensor-product style parameterization, as done in [196] for surfaces. In this chapter harmonic functions are utilized in order to fit a trivariate tensor product B-spline to a tetrahedral mesh generated from a set of triangulated isosurfaces.

In this chapter, we describe a tetrahedral mesh by the tuple $(\mathcal{H}, \mathcal{T}, \mathcal{V}, \mathcal{C})$ over the domain Ω. $\mathcal{H}$ is the set of tetrahedra and $\mathcal{T}$ is the set of faces of the tetrahedra in $\mathcal{H}$. $\mathcal{V}$ is the set of vertices, $\nu = (x_\nu, y_\nu, z_\nu) \in \mathcal{V} \subset \mathbb{R}^3$ of the tetrahedra in $\mathcal{H}$, and $\mathcal{C}$ specifies the connectivity of the mesh (the adjacency of vertices, edges, triangular faces and tetrahedra). Furthermore, $\mathcal{T}_B$ is the subset of $\mathcal{T}$ whose elements are faces of exactly one tetrahedron. The elements of $\mathcal{T}_B$ form the original exterior triangle mesh for the object. $\mathcal{V}_B \subset \mathcal{V}$ is the set of vertices defining the triangles in $\mathcal{T}_B$.

Solving equations for any but the simplest geometries requires a numerical approximation. We use mean-value coordinates [60] to solve Equation 1.1 on $\mathcal{T}_B$. Refer to [143] which discusses in more detail how to set up the appropriate linear system. The Finite Element Method (FEM) [86] is used to solve Equation 1.1 on $\mathcal{H}$. The set $\mathcal{V}$ is decomposed into two disjoint sets, $\mathcal{V}_C$ and $\mathcal{V}_I$, representing vertices that lie on the Dirichlet boundary (and hence denote positions at which the potential $\mathbf{u}$ is known) and vertices for which the solution is sought, respectively.

Then, in the case of finite elements, solutions are of the form:

$$\mathbf{u}(x, y, z) = \sum_{\nu_k \in \mathcal{V}_I} \widehat{\mathbf{u}}_k \phi_k(x, y, z) + \sum_{\nu_k \in \mathcal{V}_C} \widehat{\mathbf{u}}_k \phi_k(x, y, z),$$

where the sums denote the weighted degrees of freedom of the unknown vertices, and the Dirichlet boundary condition of the solution, respectively. $\phi_i(x, y, z)$ are the linear hat functions, which are 1 at ν_i and 0 at ν_i's adjacent vertices. Using the weak Galerkin formulation [86] yields a linear system of the form $\mathbf{S}\vec{\mathbf{u}} = \vec{f}$, consisting of stiffness matrix $\mathbf{S}$ and a right-hand-side function $\vec{f}$. Because the stiffness matrix is positive definite [86], the solution of the linear system is amenable to iterative methods such as the preconditioned conjugate gradient method [12].

Every point inside the tetrahedral mesh volume either lies on the boundary or inside a tetrahedron and the point's "$\hat{\mathbf{u}}$-value" is a linear combination of the vertices of the tetrahedron in which it lies. Given a tetrahedron defined by four vertices ν_{j_i}, $i = 1, 2, 3, 4$ and the corresponding basis functions ϕ_{j_i}, the $\hat{\mathbf{u}}$-value of a point ν inside a the tetrahedron, is the linear combination $\hat{\mathbf{u}}(\nu) = \sum_{i=1}^{4} \hat{\mathbf{u}}_{j_i}\phi_{j_i}(\nu)$, where the $\hat{\mathbf{u}}_i$'s are the respective harmonic coefficients of the tetrahedron's defining vertices.

The gradient $\nabla\hat{\mathbf{u}}$ over a tetrahedron is the linear combination

$$\nabla\hat{\mathbf{u}}(x, y, z) = \sum_{i=1}^{4} \hat{\mathbf{u}}_{j_i}\nabla\phi_{j_i}(x, y, z),$$

where $\nabla\phi_{j_i}(x, y, z) = \left(\frac{\partial\phi_{j_i}(x,y,z)}{\partial\nu_x}, \frac{\partial\phi_{j_i}(x,y,z)}{\partial\nu_y}, \frac{\partial\phi_{j_i}(x,y,z)}{\partial\nu_z}\right)$.

Note, that $\nabla\hat{\mathbf{u}}$ is constant over a tetrahedron and its boundary so it changes piecewise constantly over the tetrahedral mesh. In the following, $\mathbf{u}_\Omega$ means that the harmonic function $\mathbf{u}$ is defined over domain Ω, where Ω is $\mathcal{H}$ or $\mathcal{T}_B$.

4.2 Framework Overview

This section gives a high level overview of our proposed modeling framework. Our framework takes as input a tetrahedral mesh $\mathcal{H}$ containing, if given, interior triangle meshes such as the trabecular bone triangle mesh illustrated in Figure 4.2. Given that, the following framework steps describe the generation of the trivariate B-spline.

1: The user makes an initial choice of two critical points. These are used to establish a surface parameterization in two variables defined by orthogonal harmonic functions $\mathbf{u}_{\mathcal{T}_B}$ and $\mathbf{v}_{\mathcal{T}_B}$ (Section 4.3).

2: A structured quadrilateral mesh is generated using the surface parameterization calculated in the previous step (Section 4.3.1).

3: In this phase we move to working with the complete tetrahedral mesh. Two harmonic functions are calculated over $\mathcal{H}$ (Section 4.4):

   – $\mathbf{u}_{\mathcal{H}}$ is determined by solving Equation 1.1 with $\mathbf{u}_{\mathcal{T}_B}$ as the Dirichlet boundary condition.

   – $\mathbf{w}$ is a harmonic function orthogonal to $\mathbf{u}_{\mathcal{H}}$, having a harmonic value of 0 on $\mathcal{T}_B$ and 1 on an interior skeleton generated using $\nabla \mathbf{u}_{\mathcal{H}}$. Interior boundaries have a value between 0 and +1.

4: Isoparametric paths with constant $\mathbf{u}$-parameter value are extracted using $\nabla \mathbf{w}$. They start at vertices defining the quadrilateral mesh from step 2 and end at the skeleton. These paths are used to generate a structured hexahedral mesh which is a remesh of $\mathcal{H}$ (Sections 4.5, 4.5.1 and 4.5.2).

The intermediate structured meshes are constructed so that they faithfully approximate the input data. A trivariate B-spline is generated from the hexahedral mesh generated in step 4, by using an iterative fitting approach which avoids surface undulations in the resulting B-spline. This approach is presented in Chapter 9. The resulting B-spline can therefore have a high resolution. Additional postprocessing steps include data reduction techniques to reduce complexity and to generate B-spline volumes of different resolutions.

4.3 Modeling the Exterior

In this section a parameterization $\mathfrak{X}_2$ in two variables $\mathbf{u}$ and $\mathbf{v}$ defined over $\mathcal{T}_B$ is established. The choice of $\mathfrak{X}_2$ requires the user to choose two appropriate vertices ν_{min} and ν_{max} from the set of vertices in $\mathcal{V}_B$. Then, $\nabla^2 \mathbf{u}_{\mathcal{T}_B} = 0$ is solved with $\mathcal{V}_C = \{\nu_{min}, \nu_{max}\}$ as the Dirichlet boundary, where we set $\mathbf{u}_{\mathcal{T}_B}(\nu_{min}) = 0$ and $\mathbf{u}_{\mathcal{T}_B}(\nu_{max}) = 1$.

The choice of these two critical vertices depends on the model and on the simulation. As pointed out by [50], critical vertices affect the quality of the parameterization which in our case also affects the trivariate B-spline we are fitting. Since the user might be aware of which regions require higher fidelity and lower distortion in later simulations, the user can select a pair of critical vertices to yield an appropriate parameterization.

Since $\mathbf{u}_{\mathcal{T}_B}$ is defined only on $\mathcal{T}_B$ it can be computed rapidly which allows the user to modify it if unsatisfied with the result.

Once the user is satisfied with $\mathbf{u}_{\mathcal{T}_B}$, the harmonic function $\mathbf{v}_{\mathcal{T}_B}$ is computed so that $\nabla\mathbf{u}_{\mathcal{T}_B}$ and $\nabla\mathbf{v}_{\mathcal{T}_B}$ are nearly orthogonal. In order to calculate $\mathbf{v}_{\mathcal{T}_B}$, two seed points s_0 and s_1 on $\mathcal{T}_B$ are chosen. The first seed s_0 can be chosen arbitrarily. Given s_0, $\nabla\mathbf{u}_{\mathcal{T}_B}$ is used to extract an isoparametric path as in [50]. The path is circular, i.e., it starts and ends at s_0, and it has length l. s_1 is chosen on that path, so the path length between s_0 and s_1 is $l/2$. Note, that $\mathbf{u}_{\mathcal{T}_B}$ and $\mathbf{v}_{\mathcal{T}_B}$ are holomorphic 1-forms as defined in [9] and used in [74] to compute global conformal parameterizations.

Starting from s_0 two paths are created p_0^+ and p_0^- by following $\nabla\mathbf{u}_{\mathcal{T}_B}$ and $-\nabla\mathbf{u}_{\mathcal{T}_B}$, respectively. They end at the edges of triangles that has ν_{max}/ν_{min}, respectively, as one of its vertices (as shown in Figure 4.3). Merging p_0^+ and p_0^- yields p_0. Vertices are inserted into the mesh where p_0 intersects edges. Call $\mathcal{V}_{min}$ the set of these vertices. The same procedure is applied to determine p_1 passing through s_1. Vertices are inserted into the mesh where p_1 intersects edges. These vertices define $\mathcal{V}_{max}$. Note that $\mathcal{V}_{min} \cap \mathcal{V}_{max} = \emptyset$, and since, as a property of harmonic functions, if there exists only one minimum

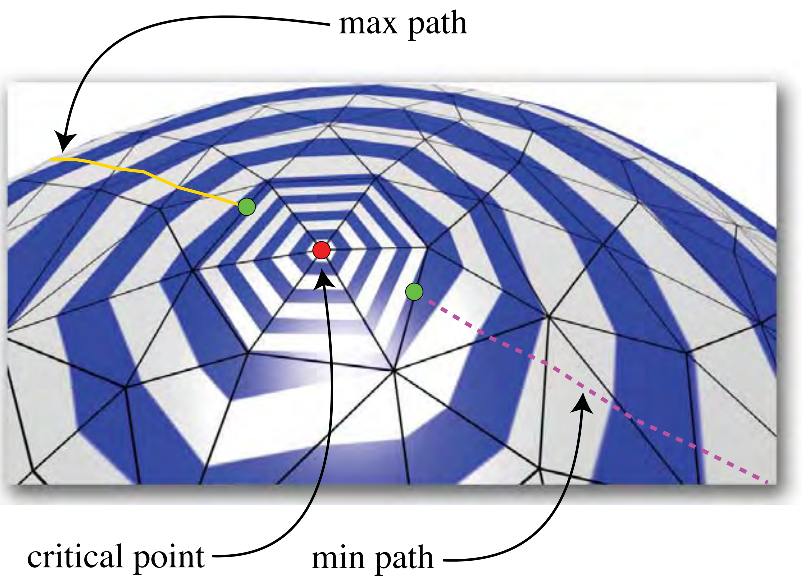


Figure 4.3. Critical paths end at the edge of a triangle, where one of its vertices is ν_{min} or ν_{max}.

(ν_{min}) and one maximum (ν_{max}), no saddle points can exist [143]. Then the mesh is retriangulated with the new vertex set.

Next, $\nabla^2 \mathbf{v}_{\mathcal{T}_B} = 0$ is solved with $\mathcal{V}_C = \mathcal{V}_{min} \cup \mathcal{V}_{max}$ as the Dirichlet boundary, where we set $\mathbf{v}_{\mathcal{T}_B}(\nu)_{\forall \nu \in \mathcal{V}_{min}} = 0$ and $\mathbf{v}_{\mathcal{T}_B}(\nu)_{\forall \nu \in \mathcal{V}_{max}} = 1$. Since the critical paths p_0 and p_1 do not reach the extremal points ν_{min} and ν_{max} (see Figure 4.3), $\mathbf{u}_{\mathcal{T}_B}$ and $\mathbf{v}_{\mathcal{T}_B}$ are not appropriately defined inside the ring of triangles around ν_{min} and ν_{max}. Let $\mathbf{u}_{start}$ be the largest $\mathbf{u}$-value of the vertices defining the ring of ν_{max}, and let $\mathbf{u}_{end}$ be the smallest $\mathbf{u}$-value of the vertices defining the ring of ν_{min}. Now, given $\mathbf{u}^{-1}_{\mathcal{T}_B}$ and $\mathbf{v}^{-1}_{\mathcal{T}_B}$, the inverse harmonic function $\mathcal{X}_2$ is constructed which maps a parametric value in the domain $[\mathbf{u}_{start}, \mathbf{u}_{end}] \times [0, 1]$ onto $\mathcal{T}_B$, i.e., $\mathfrak{X}_2 : [\mathbf{u}_{start}, \mathbf{u}_{end}] \times [0, 1] \rightarrow \mathcal{T}_B$.

$\mathcal{X}_2$ is not bijective yet as Figure 4.4 illustrates. It shows a closed isoparametric line in $\mathbf{u}_{\mathcal{T}_B}$, i.e., a closed piecewise polyline where each of its vertices has the same $\mathbf{u}$-value. The paths p_0 and p_1 divide the exterior surface into two regions I and II. Let $\alpha(\nu)$ be the part of the harmonic $\mathbf{v}$-mapping which maps a vertex ν in region I onto $[0, 1]$. The corresponding function for region II is called $\beta(\nu)$. In order to make $\mathcal{X}_2$ bijective we define a single harmonic $\mathbf{v}$-mapping

$$\gamma(\nu) = \begin{cases} \alpha(\nu)/2 & , \nu \in I \\ 1 - \beta(\nu)/2 & , \nu \in II \end{cases}$$

Figure 4.5 illustrates these transformations. At this stage, every $\nu \in \mathcal{V}_B$ has a $\mathbf{u}$- and $\mathbf{v}$-parameter value. Note that $\mathbf{v}$ is periodic so $0 \equiv 1$.

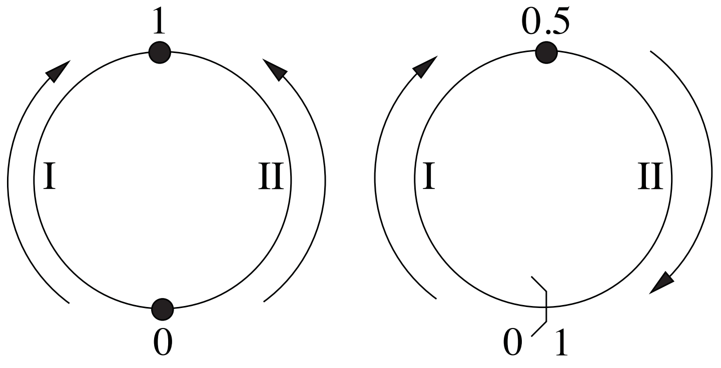


Figure 4.4. $\mathcal{X}_2$ (left) is not bijective.

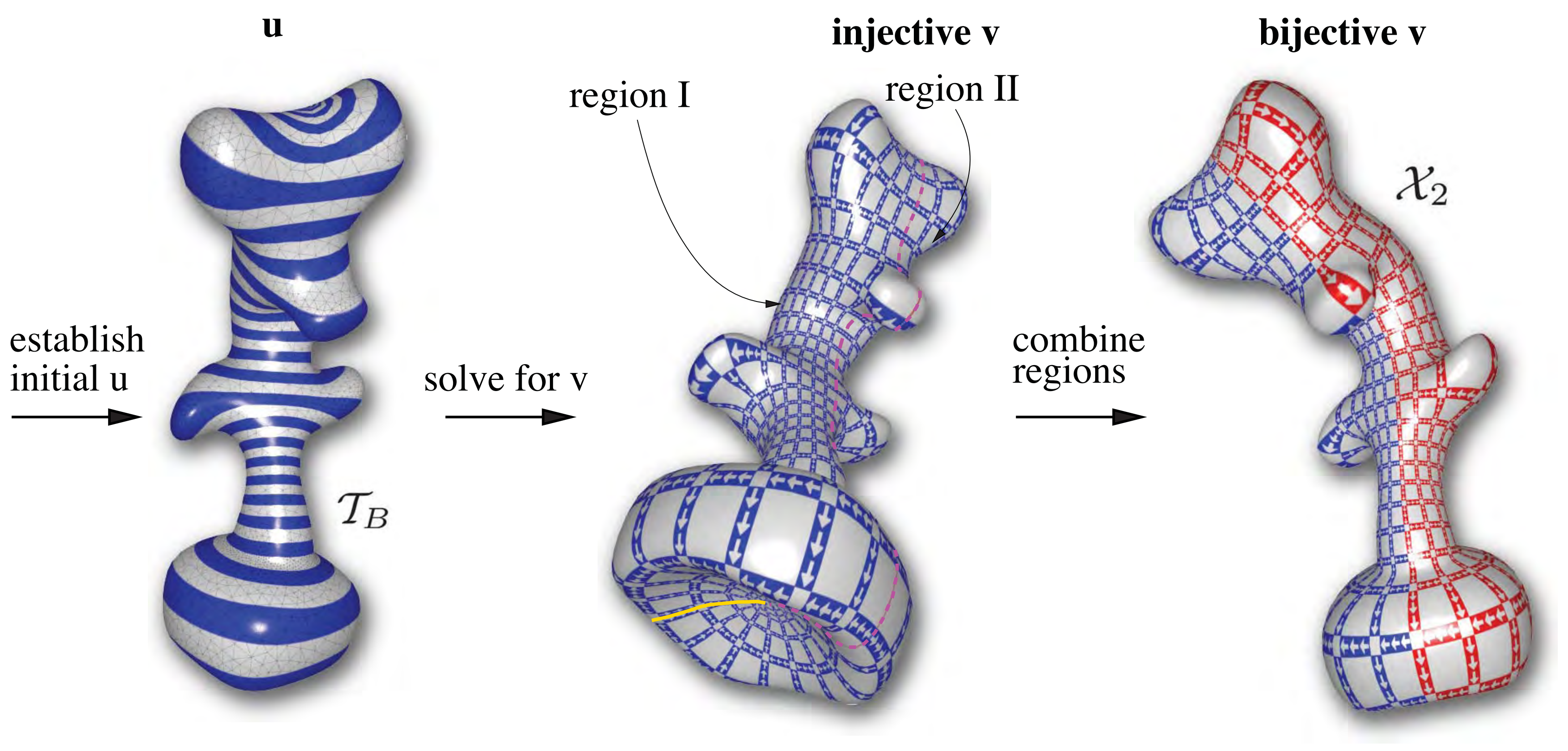


Figure 4.5. Establishing the surface parameterization $\mathcal{X}_2$. On the left: the harmonic function $\mathbf{u}_{\mathcal{T}_B}$ defined by two critical points is established over $\mathcal{T}_B$; middle: Based on $\mathbf{u}_{\mathcal{T}_B}$, the orthogonal harmonic function $\mathbf{v}_{\mathcal{T}_B}$ is calculated. At this stage $\mathbf{u}_{\mathcal{T}_B}$ and $\mathbf{v}_{\mathcal{T}_B}$ define an injective transformation; on the right: Scaling and translation yields the parameterization $\mathcal{X}_2$.

A region whose corners consist of right-angles can be parameterized so that the resulting gradient fields are orthogonal [196]. However, in our case, $\mathbf{u}_{\mathcal{T}_B}$ degenerates to points (ν_{min} and ν_{max}), implying that $\nabla\mathbf{u}_{\mathcal{T}_B}$ and $\nabla\mathbf{v}_{\mathcal{T}_B}$ are not orthogonal near ν_{min} and ν_{max}. This means that a quadrangulation in this area is of poorer quality. Note that ν_{min} and ν_{max} were chosen in areas which are not important in the proposed simulation. Furthermore, in case of the femur, 98% of the angles between the gradient vectors in $\nabla\mathbf{u}_{\mathcal{T}_B}$ and $\nabla\mathbf{v}_{\mathcal{T}_B}$ lie in the range $[\pi/2 - 0.17, \pi/2 + 0.13]$.

4.3.1 u- and v-section Extraction

Similar to [82], our goal is to extract a set of **u**- and **v**-parameter values so that the corresponding isoparametric curves on the model define a structured quadrilateral mesh which represents the exterior of the tetrahedral mesh faithfully.

Let $\widehat{\mathcal{T}_B}$ be the exterior triangle mesh inversely mapped into the parameter space as illustrated in Figure 4.6 (left). We seek to find a set $U = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{n_0}\}$ of **u**-values and a set $V = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{n_1}\}$ of **v**-values so that the collection of images of the grid form an error bounded grid to the model. The isocurve at a fixed $\mathbf{u}_i \in U$ corresponds to the line $l_i(\mathbf{v}) = (\mathbf{u}_i, \mathbf{v})$ in parameter space, where $\mathbf{v} \in [0, 1]$, and, the isocurve at a fixed

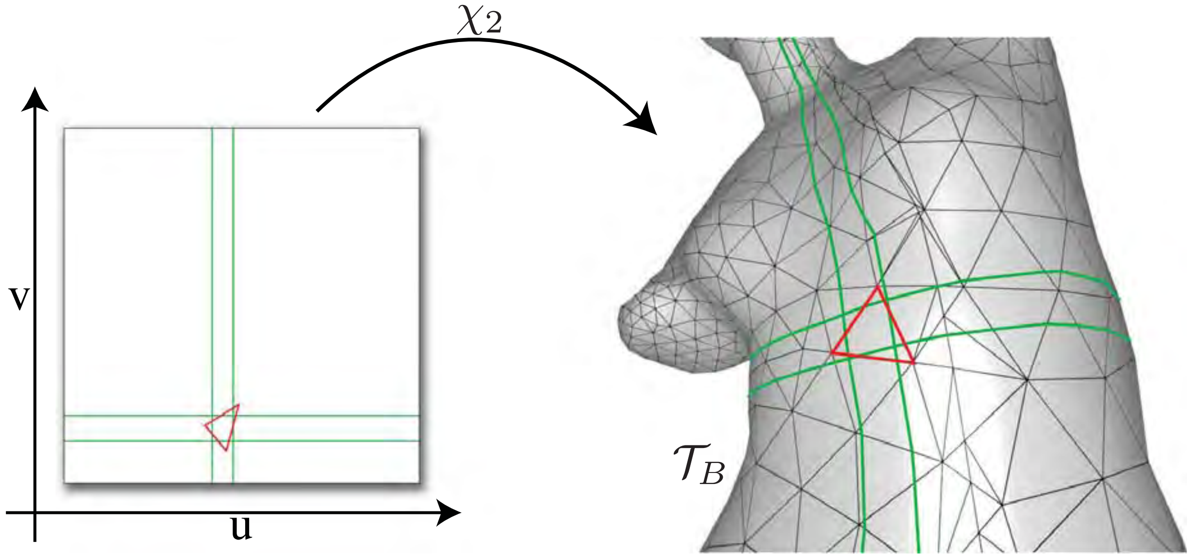


Figure 4.6. $\mathfrak{X}$ maps a vertical line at $\mathbf{u}_0$ in parameter space onto a closed isoparametric line on $\mathcal{T}_B$. Accordingly, $\mathfrak{X}$ maps a horizontal line at $\mathbf{v}_0$ onto an isoparametric which starts at ν_{min} and ends at ν_{max}.

$\mathbf{v}_j \in V$ corresponds to the line $\widehat{l_j}(\mathbf{u}) = (\mathbf{u}, \mathbf{v}_j)$ in parameter space, where $\mathbf{u} \in [0,1]$. $l_i(\mathbf{v})$ and $\widehat{l_j}(\mathbf{u})$ are orthogonal. Note, that $\mathfrak{X}_2$ maps $l_i(\mathbf{v})$ and $\widehat{l_j}(\mathbf{u})$ to isocurves on $\mathcal{T}_B$. The intersections of the lines $l_i(\mathbf{v})$ and $\widehat{l_j}(\mathbf{u})$, i.e., the parameter pairs $(\mathbf{u}_i, \mathbf{v}_j)$ define a structured grid with rectangular grid cells over the parametric domain and hence a quadrilateral mesh over $\mathcal{T}_B$. This quadrilateral mesh is a remesh of $\mathcal{T}_B$.

Let $\mathcal{E}$ be the set of edges defining the triangles in $\widehat{\mathcal{T}_B}$. U and V are chosen so that every edge in $\mathcal{E}$ is intersected by at least one $l_i(\mathbf{v})$ and one $\widehat{l_j}(\mathbf{u})$, as shown in Figure 4.6 for one triangle. U and V are calculated independently from each other. An edge $e \in \mathcal{E}$ is defined by two points in parametric space $(\mathbf{u}_e, \mathbf{v}_e)$ and $(\mathbf{u}'_e, \mathbf{v}'_e)$. Based on $\mathcal{E}$, we define $\mathcal{S}_\mathbf{u}$ to be the set of intervals defined as the collection of intervals $(\mathbf{u}_e, \mathbf{u}'_e)$ such that $(\mathbf{u}_e, \mathbf{v}_e)$ and $(\mathbf{u}'_e, \mathbf{v}'_e)$ are the endpoints defining an edge $e \in \mathcal{E}$. We employ the interval structure for stabbing queries [52], that takes a set of intervals (in our case $\mathcal{S}_\mathbf{u}$) and constructs an interval tree $\mathcal{I}_\mathbf{u}$ in $O(n \log n)$, where n is the number of intervals in $\mathcal{S}_\mathbf{u}$. Every node in $\mathcal{I}_\mathbf{u}$ includes an interval location $\mathbf{u} \in [0,1]$. $\mathcal{I}_\mathbf{u}$ covers every interval $\rightarrow$ edge $\rightarrow$ triangle in $\mathcal{T}_B$. The $\mathbf{u}$-values of the nodes in the tree define the set U and the vertical stabbing lines $l_i(\mathbf{v})$.

V is defined analogously, with the difference that $\mathcal{S}_\mathbf{v}$ consists of intervals defined by the segments $(\mathbf{v}_e, \mathbf{v}'_e)$ for which $(\mathbf{u}_e, \mathbf{v}_e)$ and $(\mathbf{u}'_e, \mathbf{v}'_e)$ are the endpoints of an edge $e \in \mathcal{E}$. Then, V consists of the $\mathbf{v}$-values defining the nodes in $\mathcal{I}_\mathbf{v}$ and the horizontal stabbing lines $\widehat{l_j}(\mathbf{u})$.

The algorithm to determine U and V guarantees that in a rectangle defined by the points $p_0 = (\mathbf{u}_i, \mathbf{v}_j)$, $p_1 = (\mathbf{u}_{i+1}, \mathbf{v}_j)$, $p_2 = (\mathbf{u}_{i+1}, \mathbf{v}_{j+1})$ and $p_3 = (\mathbf{u}_i, \mathbf{v}_{j+1})$, where $\mathbf{u}_i, \mathbf{u}_{i+1} \in U$ and $\mathbf{v}_j, \mathbf{v}_{j+1} \in V$, there is either the preimage of at most one vertex of a triangle (Case 1) or none (Case 2). These two cases are illustrated in Figure 4.7.

We show this is true by contradiction. That is, assume that there are two vertices in the same rectangle. Since we require that the input mesh is a 2-manifold, there has to be a path defined by triangle edges from one vertex to the other. However, due to the interval tree property that every interval is cut by at least one stabbing line, at least one isoline with fixed $\mathbf{u}$-value and one isoline with fixed $\mathbf{v}$-value intersects an edge. Therefore, the two vertices must be separated.

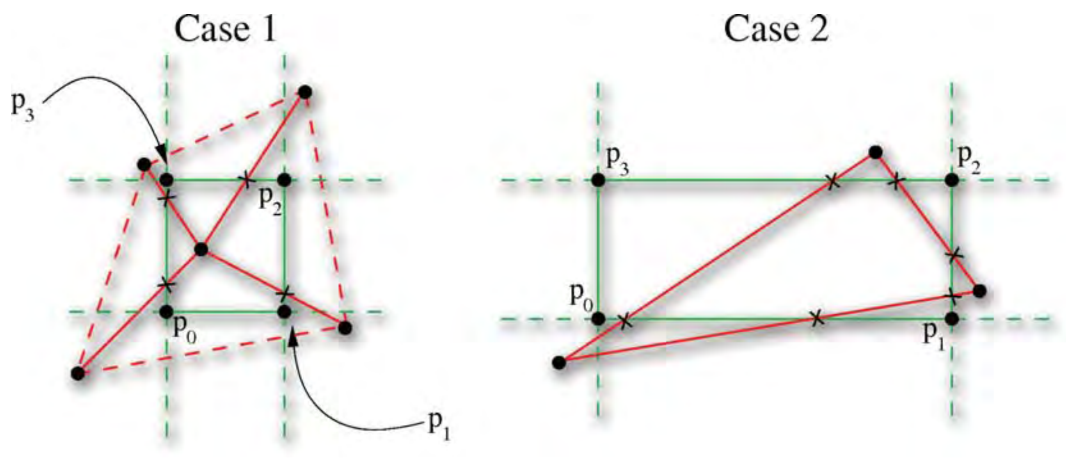


Figure 4.7. Either there is one vertex in the rectangle defined by the points p_0, p_1, p_2 and p_3, or none. Crosses mark edge intersections.

Now we want to ensure that the quadrilateral grid that we are deriving is within error tolerance. Let us consider the rectangle $\widehat{R}_{i,j}$ defined by the points $(\mathbf{u}_i, \mathbf{v}_j)$ and $(\mathbf{u}_{i+1}, \mathbf{v}_{j+1})$ (as in Figure 4.7). The vertices of its corresponding bilinear surface $R_{i,j}$ on $\mathcal{T}_B$ are $\mathfrak{X}_2(\mathbf{u}_i, \mathbf{v}_j)$, $\mathfrak{X}_2(\mathbf{u}_{i+1}, \mathbf{v}_j)$, $\mathfrak{X}_2(\mathbf{u}_{i+1}, \mathbf{v}_{j+1})$ and $\mathfrak{X}_2(\mathbf{u}_i, \mathbf{v}_{j+1})$. We measure how far $R_{i,j}$ is away from the triangle mesh. We look at this measurement for the two above cases separately.

For case 1, let $(\mathbf{u}^*, \mathbf{v}^*)$ be the parameter value of the vertex lying in $\widehat{R}_{i,j}$. Consider one of the triangles associated with that point, each edge of $\widehat{R}_{i,j}$ maybe intersected by either zero, one, or two of the triangle's edges. If intersections exist, we transform them with $\mathfrak{X}_2$ onto $\mathcal{T}_B$ and measure how far they are away from $R_{i,j}$. Furthermore, the distance between $\mathfrak{X}_2(\mathbf{u}^*, \mathbf{v}^*)$ and $R_{i,j}$ is determined. Given a user-defined ϵ, if the maximum of all these distances is smaller than $\epsilon/2$, we have sufficient accuracy, if not, then we insert a new $\mathbf{u}$-slice between $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ and a new $\mathbf{v}$-slice between $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ and reexamine the newly created rectangles. Case 2 is handled similarly to Case 1 without the projection of the interior point.

Depending on the resolution of $\mathcal{T}_B$, the sets U and V may have more parameter values than necessary. For instance, if $\mathcal{T}_B$ is a densely triangulated cylinder, most of the parameter values in U are not necessary. To some extent, more isolines are needed around features. On the other hand, isolines might also be needed in areas on which force due to boundary conditions is applied. These regions could have no shape features at all. After the B-spline volume is modeled using our framework, refinement and data reduction techniques [120]are applied to yield trivariate approximations with different resolutions. However, it still can be helpful to remesh the input triangle meshes first with a feature aware triangulator such as Afront [174] which generates meshes having more triangles in regions with higher curvature and fewer triangles in regions with very low curvature. In this case, more knots are placed in regions with higher curvature regions and fewer knots are places in more flat regions. This can reduce the resolution of the initial mesh.

Given an input triangle mesh, an upper bound on the error can be determined. Since there is a guarantee that every edge is intersected by at least one isoline with fixed **u**- and one isoline with fixed **v**-value, the maximum error can be computed in the following way: Given $\mathcal{T}_B$, we consider the ring of a vertex $\nu \in \mathcal{V}_B$, where the ring is the set of all adjacent vertices of ν being elements in $\mathcal{V}_B$. We construct a bounding box where one of its axes are coincident with the normal of ν. The height of the bounding box side coincident with the normal of ν is the error for that ring. We compute such a bounding box for every vertex on the exterior. The maximum height will be the maximum error.

4.4 Modeling the Interior

Once the exterior parameterization is determined, the tetrahedral mesh ($\mathcal{H}$) is parameterized. Using FEM, $\nabla^2 \mathbf{u}_{\mathcal{H}} = 0$ is solved, where $\mathcal{V}_B$ with its respective **u**-values is used as the Dirichlet boundary condition. Now, all elements in $\mathcal{V}$ have a **u**-parameter value. In the surface case, fixing a **u**-value gives a line in parameter space and a closed isocurve on the surface. In the volume case, fixing a **u**-value gives a plane in parameter space and a surface called an isosheet in the volume. The boundary of an isosheet for a fixed $\mathbf{u}_0$ is the isocurve on the surface at $\mathbf{u}_0$.

Now, for each boundary slice at $\mathbf{u}_{i_0}$, it is necessary to extract its corresponding isosheet. First however, a skeleton is created to serve as isosheet center for all isosheets. Then, a third function $\mathbf{w}$ is created whose gradient field $\nabla \mathbf{w}$ points to the skeleton. $\nabla \mathbf{w}$ is used to trace a path starting at $\mathfrak{p}_{i_0,j} = \mathfrak{X}_2(\mathbf{u}_{i_0}, \mathbf{v}_j)$ and ending at the skeleton on the sheet, for $j = 0, \ldots, n_1$ (see right). $\nabla\mathbf{w}$ is constructed to be tangent to the isosheet at a given point, so $\nabla\mathbf{u}_{\mathcal{H}}$ and $\nabla\mathbf{w}$ are orthogonal. This guarantees that every point on the extracted **w**-path has the same **u**-value.

In [199] a method is proposed, to construct skeleton curves from an unorganized collection of scattered points lying on a surface which can have a "tree-like" structure. They calculate a geodesic graph over the point set. Using that graph, they extract level sets, closed and piecewise linear. The centroids of all the level sets form the skeleton. When level sets are not convex the centroid may lie outside the objects. Furthermore, the skeleton may have loops if the centroid of a given level set a lies above the centroid of the level set b lying above a. Similarly, [110] explicitly establishes a scalar function, similar to a harmonic function, over a triangle mesh. By choosing a source vertex, for every vertex on the triangle mesh, shortest distances are calculated which establish a parameterization in one parameter. The skeleton is calculated as in [199] and therefore cannot guarantee whether it lies within the triangle mesh.

The skeleton is created by tracing two paths which start at a user-specified seed using $+\nabla\mathbf{u}_{\mathcal{H}}$ and $-\nabla\mathbf{u}_{\mathcal{H}}$ and end at ν_{min} and ν_{max}, respectively. Merging these two paths yields the skeleton. By the definition of $\nabla\mathbf{u}_{\mathcal{H}}$, the skeleton can have no loops. The skeleton has the properties of a Reeb Graph [184], in that its end vertices correspond to ν_{min} and ν_{max}. While the Reeb graphs in [184] are defined over a surface, our Reeb graph, i.e., the skeleton, is defined over the volume. Because of the way $\nabla\mathbf{w}$ is built, a sheet is orthogonal to the skeleton, which is also a property of GC. The orthogonal property of the skeleton and $\nabla\mathbf{w}$ is also a desirable property to attain a good B-spline fit. The skeleton can be computed in interactive time, and the user has flexibility in choosing the seed. In general, the seed should be placed such that the resulting skeleton lies in the "center" of the innermost isosurface, like the axis of a cylinder.

Just solving $\nabla^2\mathbf{w} = 0$ with the respective boundary conditions does not guarantee orthogonality of $\nabla\mathbf{u}_{\mathcal{H}}$ and $\nabla\mathbf{w}$, and if $\nabla\mathbf{u}_{\mathcal{H}}$ and $\nabla\mathbf{w}$ are not orthogonal, there is no reason that a path will have the same **u**-value throughout. This implies the **w**-parameter will need further adjustment to guarantee a well behaved parameterization and so adjacent isosheets do not overlap. In order to enforce orthogonality $\nabla\mathbf{w}$ is constructed in the following two steps: (1) The points defining the skeleton are inserted into the tetrahedral mesh and a new mesh is formed. Then, $\nabla^2\widehat{\mathbf{w}} = 0$ is solved over the tetrahedral mesh, subject to Dirichlet boundary conditions defined by the set $\mathcal{V}_C = \mathcal{V}_B \cup \mathcal{V}_{T_1} \cup \ldots \cup \mathcal{V}_{T_k} \cup \mathcal{V}_S$. $\mathcal{V}_B$ consists of the boundary vertices where $\widehat{\mathbf{w}}(\nu)_{\forall\nu\in\mathcal{V}_B} = 0$, $\mathcal{V}_S$ consists of the vertices defining the skeleton where $\widehat{\mathbf{w}}(\nu)_{\forall\nu\in\mathcal{V}_S} = 1$, $\mathcal{V}_{T_i}$ is the set of vertices defining the ith of k isosurfaces where $\widehat{\mathbf{w}}(\nu)_{\forall\nu\in\mathcal{V}_{T_i}} = i/(k+1)$. In the case of the femur and in Figure 4.8, there is one isosurface, namely the surface separating the trabecular and cortical bone. In this case $\mathcal{V}_C = \mathcal{V}_B \cup \mathcal{V}_{T_1} \cup \mathcal{V}_S$, where $\widehat{\mathbf{w}}(\nu)_{\forall\nu\in\mathcal{V}_{T_1}} = 1/2$. Then in step (2), for every tetrahedron, we project its $\nabla\widehat{\mathbf{w}}$ gradient vector onto the plane whose normal is the corresponding $\nabla\mathbf{u}_{\mathcal{H}}$, to form $\nabla\mathbf{w}$.

4.5 Tracing w-paths

Flow line extraction over a closed surface triangle mesh is described in [50]. In our case, we extract flow lines throughout a volume. A flow line, or a **w**-path will start on $\mathcal{T}_B$, where $\mathbf{w} = 0$ and traverses through $\mathcal{H}$ until it reaches the skeleton on which

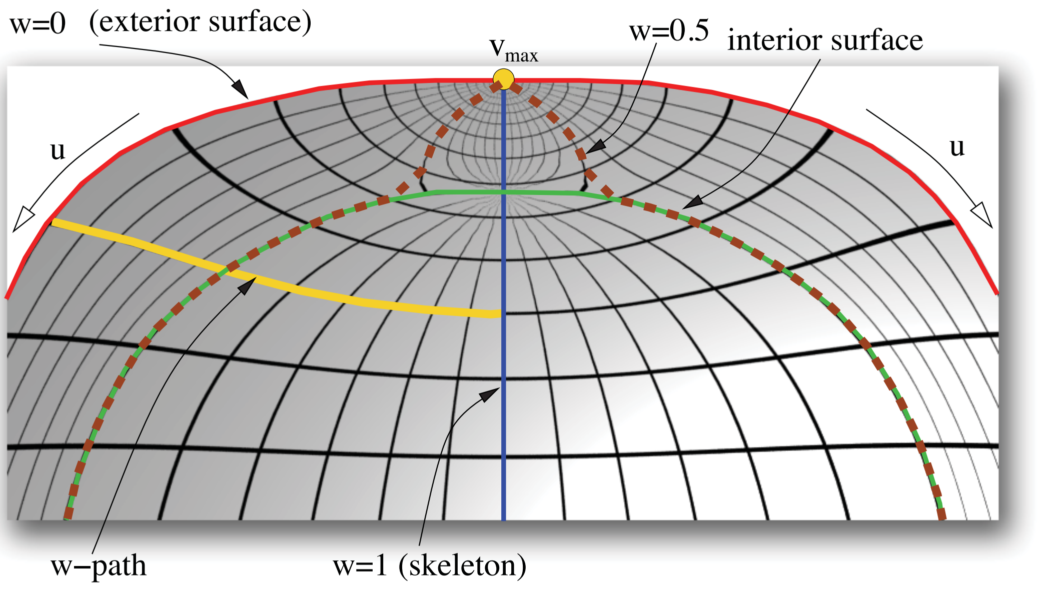


Figure 4.8. A cross section of an object with an exterior boundary and an interior isosurface representing geometry or attribute data. The skeleton and boundaries were used to establish $\nabla\mathbf{w}$. Isolines visualize the **uw**-scalar field used to trace **w**-paths from the exterior to the skeleton.

$\mathbf{w} = 1$. The resulting $\mathbf{w}$-path is a piecewise linear curve where every segment belongs in a tetrahedron. The two ends of the segment lie on faces of the respective tetrahedron and is coincident with $\nabla\mathbf{w}$. Since $\nabla\mathbf{u}_{\mathcal{H}}$ and $\nabla\mathbf{w}$ are orthogonal, every point on such a segment has a constant $\mathbf{u}$-value, and therefore, the $\mathbf{w}$-path has a constant $\mathbf{u}$-value.

During $\mathbf{w}$-path traversal, in the regular case, the endpoint q of the $\mathbf{w}$-path will lie on a face of a tetrahedron. The next traversal point is determined by constructing a ray $\vec{r}$ with origin at q with $\nabla\mathbf{w}_{\mathcal{H}}$ of the adjacent tetrahedron as direction. $\vec{r}$ is then intersected with the faces of the adjacent tetrahedron, except the triangle on which q lies, to find the next q. Let p be the intersection between $\vec{r}$ and triangle t. t is a face of two tetrahedra, the current and the next tetrahedron. The line segment $\overline{qp}$ is added to the current $\mathbf{w}$-path, and p becomes q.

During the $\mathbf{w}$-path traversal, several pathological cases can arise. One is when the intersection point p lies on an edge e of the current tetrahedron. Since the edge is part of two triangles, an ambiguity exists as to which face should be chosen. Instead we consider all tetrahedra that have e as an edge. For each of these tetrahedra we construct a ray having its origin at p with $\nabla\mathbf{w}$ of the tetrahedron as its direction. If there is an intersection between a tetrahedron's ray and one of its faces, then we choose that face of the respective tetrahedron as the next triangle. Analogously, at the other degeneracy, when p lies at a vertex of the tetrahedron, we examine every tetrahedron that coincides with this vertex. We choose the tetrahedron in which we can move furthest in $\nabla\mathbf{w}$ direction.

Another degenerate case arises when $\vec{r}$ does not intersect with any triangle, edge or vertex of the current tetrahedron. This implies that $\vec{r}$ points outward from the tetrahedron. When this occurs, we construct a plane through q orthogonal to $\nabla\mathbf{u}_{\mathcal{H}}$ of the current tetrahedron. Every point on that plane in the tetrahedron has the same $\mathbf{u}$-value. We intersect the plane with the edges of the triangle in which q is located. In general position, there are two intersections. We choose that intersection which has a bigger $\mathbf{w}$-value as next point on the $\mathbf{w}$-path, because it lies closer to the skeleton. Since the intersection point is a point on an edge or a vertex, the first special case is applied.

4.5.1 w-path Extraction

In Section 4.3.1, we discussed how the Cartesian product of the sets U and V spans over the $\mathbf{uv}$-domain. $\mathfrak{X}_2$ maps the grid point $(\mathbf{u}_i, \mathbf{v}_j)$ to the point $\mathfrak{p}_{i,j}$ in $\mathcal{T}_B$. The points $\mathfrak{p}_{i,j}$ are used as starting points to trace $\mathbf{w}$-paths, as described above in Section 4.5. Now, $\mathcal{X}_3 : [\mathbf{u}_{start}, \mathbf{u}_{end}] \times [0,1] \times [0,1] \to \mathcal{H}$ is a parameterization in three variables $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, where $\mathfrak{X}_3(\mathbf{u}, \mathbf{v}, 0) \equiv \mathfrak{X}_2(\mathbf{u}, \mathbf{v})$, $\mathfrak{X}_3(\mathbf{u}, \mathbf{v}, 1)$ defines the skeleton, and $\mathfrak{X}_3(\mathbf{u}, \mathbf{v}, i/(k+1))$ defines the ith isosurface.

In this section, we want to find a set $W = \{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{n_2}\}$ where $\mathbf{w}_0 = 0$ and $\mathbf{w}_{n_2} = 1$, which contains n_2 parameter values. The Cartesian product $U \times V \times W$ defines a structured grid on $[\mathbf{u}_{start}, \mathbf{u}_{end}] \times [0,1] \times [0,1]$ and a structured hexahedral mesh with points $\mathfrak{p}_{i,j,k} = \mathfrak{X}_3(u_i, v_j, w_k)$ in $\mathcal{H}$ with degeneracies only along the skeletal axis. Note, that $\mathfrak{p}_{i,j,k}$ refers to the kth point on the jth $\mathbf{w}$-path on isosheet i, i.e., by fixing $\mathbf{u}_{i_0}$, the points $\mathfrak{X}_3(\mathbf{u}_{i_0}, \mathbf{v}_j, \mathbf{w}_k)$ lie and approximate isosheet i_0 and connect to a structured quadrilateral mesh called $\mathfrak{S}_{i_0}$.

Let $\mathfrak{h}_{i_0,j_0} : [0,1] \to \mathcal{H}$ be the j_0th $\mathbf{w}$-path on $\mathfrak{S}_{i_0}$, defined by the points $\mathfrak{p}_{i_0,j_0,k}$, where $k = 0, \ldots, n_2$. Depending on the choice of $\mathbf{w}$-values in W, points on $\mathfrak{h}_{i_0,j_0}$ may have different $\mathbf{u}$-values. This leads to a modification of $\mathbf{u}$ on the interior parameterization. This is allowed as long the $\mathbf{u}$-value of these points is smaller (bigger) than the $\mathbf{u}$-value of the upper (lower) adjacent isosheet. Otherwise, isosheets might intersect.

Originally, when a $\mathbf{w}$-path is extracted as discussed above, all parameter values in $[0,1]$ map to points whose $\mathbf{u}$-values are the same. The reason for that is that the line segments defining the initial $\mathbf{w}$-path all lie in tetrahedra and coincide with $\nabla \mathbf{w}_{\mathcal{H}}$ of the respective tetrahedron. Furthermore, every extracted $\mathbf{w}$-path is defined initially by different sets of $\mathbf{w}$-values. In order to determine W, we have to make sure that the quadrilateral sheets $\mathfrak{S}_i$ do not intersect.

Let $W_{i,j} = \{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{n_{2i,j}}\}$, where $\mathbf{w}_0 = 0 < \mathbf{w}_1 < \ldots < \mathbf{w}_{n_{2i,j}} = 1$, be the sorted set of $\mathbf{w}$-parameter values for $\mathfrak{h}_{i_0,j_0}$, consisting of $n_{2i,j} + 1$ points. $n_{2i,j}$ depends on the number of tetrahedra the path is travelling through. If the path is close to ν_{min} or ν_{max}, the path is probably shorter than a path towards to the middle of the object. A valid and common W could be found by calculating the union of all $W_{i,j}$, where then W would contain an unnecessarily large number of $\mathbf{w}$-values. Therefore as a first step we simplify every $W_{i,j}$ by removing unnecessary $\mathbf{w}$-values from it. Let $\mathbf{u}_i$ be the $\mathbf{u}$-value of the current slice. We scan $W_{i,j}$ and remove an element $\mathbf{w}_k$ when the $\mathbf{u}$-value of the point $(P_{k-1} + P_{k+1})/2$ is in the range $[(\mathbf{u}_i - \mathbf{u}_{i-1})/2, (\mathbf{u}_i + \mathbf{u}_{i+1})/2]$, where P_k is the position in the tetrahedral mesh where $\mathbf{w}_k$ lies. This implies that sheet i does not intersect with one of its adjacent sheets. The P_k's leading to the smallest differences are removed first. This is applied iteratively until no further points can be removed from the path.

Then, we merge the simplified sets $W_{i,j}$ to get W. After merging, elements in W may be very close to each other. We therefore remove elements in W, such that every pair of parameter values in W has at least a distance (in parameter space) of ϵ between each other. Furthermore, the parametric $\mathbf{w}$-values for the isosurfaces are added to W, too, such as $0.5 \in W$, where $\mathcal{X}_3(\mathbf{u}, \mathbf{v}, 0.5)$ represents the inner boundary in Figure 4.2.

4.5.2 w-path Smoothing

Since there is only an exterior $\mathbf{v}$-parameterization, points lying on a $\mathbf{w}$-path have a constant $\mathbf{u}$-value but not a constant $\mathbf{v}$-value. This results in path wiggling as shown in Figure 4.9. Path wiggling means that parts of a given path may lie closer to its adjacent path on the left than to its adjacent path on the right.

Laplacian smoothing [63] is an efficient way to smooth a mesh and remove irregularities. As pointed out in [63], applying it to a hexahedral mesh can lead to inconsistencies, like "tangling" of hexahedra. This especially happens in regions with overhangs, where in our case, Laplacian smoothing would change the $\mathbf{u}$- and $\mathbf{w}$-value of a point, leading to overlapping sheets. However, Laplacian smoothing is computationally efficient and we adapt it for our case in the following way.

Let $\nabla \mathbf{v}_{\mathcal{H}}$ be the cross product field between $\nabla \mathbf{u}_{\mathcal{H}}$ and $\nabla \mathbf{w}$, i.e., $\nabla \mathbf{v}_{\mathcal{H}} = \nabla \mathbf{u}_{\mathcal{H}} \times \nabla \mathbf{w}_{\mathcal{H}}$. Since $\mathbf{v}_j$ is not constant along the $\mathbf{w}$-path, during mesh smoothing, we restrict the location $\mathfrak{p}_{i,j,k}$ to change only along $\nabla \mathbf{v}_{\mathcal{H}}$. Let $Project(\mathfrak{p}, d)$ be a function defined over $\mathcal{H}$ which determines a point $\mathfrak{p}'$ along $\nabla \mathbf{v}_{\mathcal{H}}$ with distance d from $\mathfrak{p}$. $\mathfrak{p}$ and $\mathfrak{p}'$ both lie on a piecewise linear $\mathbf{v}$-path, and the $\mathbf{v}$-path section defined by $\mathfrak{p}$ and $\mathfrak{p}'$ has a length of d. Since $\nabla \mathbf{u}_{\mathcal{H}}$, $\nabla \mathbf{v}_{\mathcal{H}}$ and $\nabla \mathbf{w}$ are orthogonal vector fields, $\mathfrak{p}$ and $\mathfrak{p}'$ have the same

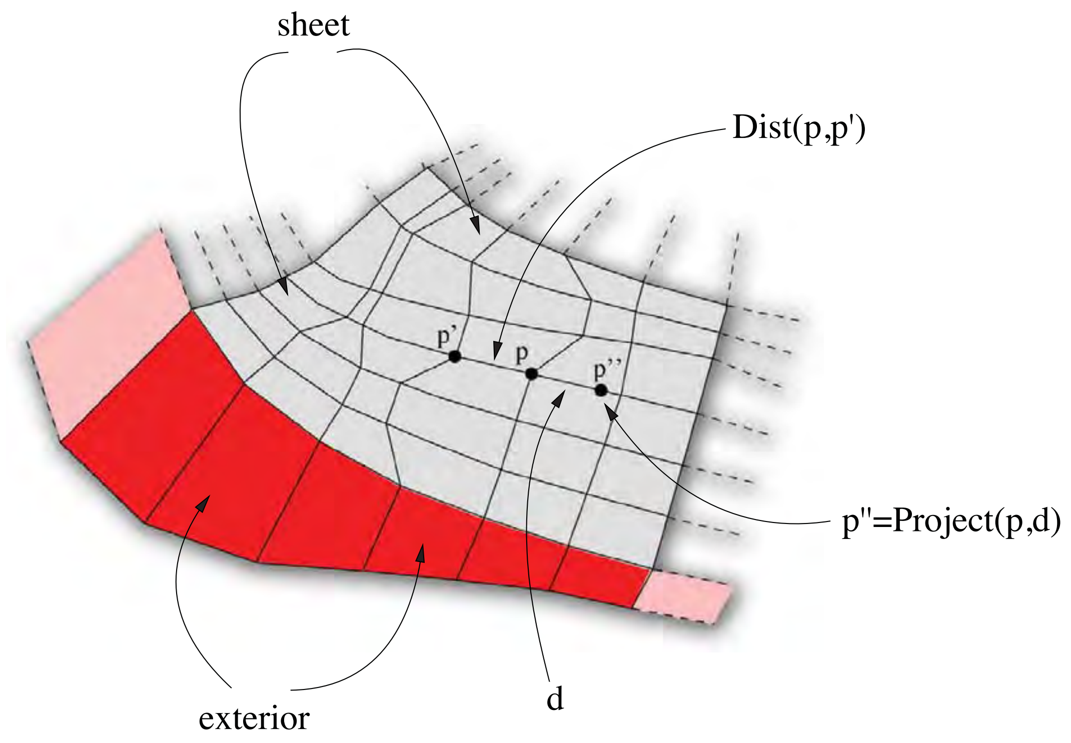


Figure 4.9. Due to the linear property of $\nabla\mathbf{u}$ and $\nabla\mathbf{v}$ and special cases during $\mathbf{w}$-path extraction as discussed in section 4.5, adjacent paths may collapse.

$\mathbf{u}$- and $\mathbf{w}$-value. Furthermore, let $Dist(\mathfrak{p}, \mathfrak{q})$ be a function that computes the length of the $\mathbf{v}$-path section defined by $\mathfrak{p}$ and $\mathfrak{q}$, requiring that $\mathfrak{p}$ and $\mathfrak{q}$ have the same $\mathbf{u}$- and $\mathbf{w}$-value. Now, the position of the mesh point $\mathfrak{p}_{i,j,k}$ is updated by

$$\mathfrak{p}'_{i,j,k} = Project(\mathfrak{p}_{i,j,k}, \frac{1}{2}(Dist(\mathfrak{p}_{i,j,k}, \mathfrak{p}_{i,j-1,k}) + Dist(\mathfrak{p}_{i,j,k}, \mathfrak{p}_{i,j+1,k}))).$$

After this procedure is applied to every $\mathfrak{p}_{i,j,k}$ where $i > 0$ and $i < n_0$, the old positions $\mathfrak{p}_{i,j,k}$ are overwritten with $\mathfrak{p}'_{i,j,k}$. By repeating this procedure the mesh vertices move so that for a given $\mathfrak{p}_{i,j,k}$, the ratio $Dist(\mathfrak{p}_{i,j,k}, \mathfrak{p}_{i,j+1,k})/Dist(\mathfrak{p}_{i,j,k}, \mathfrak{p}_{i,j-1,k})$ moves closer to 1, by maintaining a constant $\mathbf{u}$- and $\mathbf{w}$-value. Therefore, this approach avoids isosheet intersection. The procedure terminates when $\max|Dist(\mathfrak{p}_{i,j,k}, \mathfrak{p}_{i,j+1,k})| < \epsilon$, where ϵ is user-defined.

4.6 Possible Extensions

So far we have assumed that the user chooses two min/max points as the first step in our modeling framework. As discussed above, those two critical points are the end points of a skeleton line through the model. This works well when the lengths of the **w**-paths of a given slice are similar. If isosheets are circular and the skeleton goes through the center, the lengths of the **w**-paths are equal to the radius of the isosheet. By fixing $\mathbf{u}_0$ and $\mathbf{w}_0$ the quadrilateral $\mathfrak{q}_i$ defined by the points $\mathfrak{X}_3(\mathbf{u}_0, \mathbf{v}_0, \mathbf{w}_0)$ and $\mathfrak{X}_3(\mathbf{u}_0, \mathbf{v}_i + \epsilon, \mathbf{w}_0 + \epsilon)$ for a given small ϵ has the same area for any $\mathbf{v}_i$.

However, input models exist, where **w**-path lengths of a given isosheet are different. Refer to Figure 4.10 for a simple example, where the user chose ν_{min} and ν_{max}. For a constant **u**-value we extracted the corresponding isosheet. Isosheets in that model have a rectangular shape. For such a shape a skeletonal line is not appropriate: Isoparametric lines in **v** towards the exterior are rectangular, but change into circles when approaching the skeleton. This results in distortion: For a given isosheet we can find the shortest and the longest **w**-path. The quadrilaterals $\mathfrak{q}_i$ do not have the same area; they are bigger around the longest path, compared to its areas around the shortest path.

Numerical applications such as finite elements desire more uniform element sizes. By modifying the choice of the skeleton the resulting B-spline can be improved. Instead of choosing a single vertex as a critical point, the user chooses a critical path as in Figure 4.11. The resulting skeleton is a surface. In that case, critical paths suit the rectangular shape better than critical points. The **w**-paths of a sheet have about the same length,

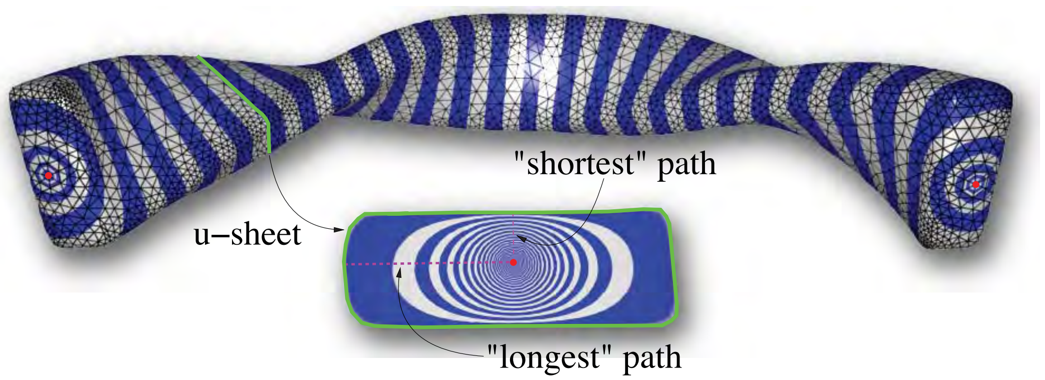


Figure 4.10. **w**-parameterizations using min/max points.

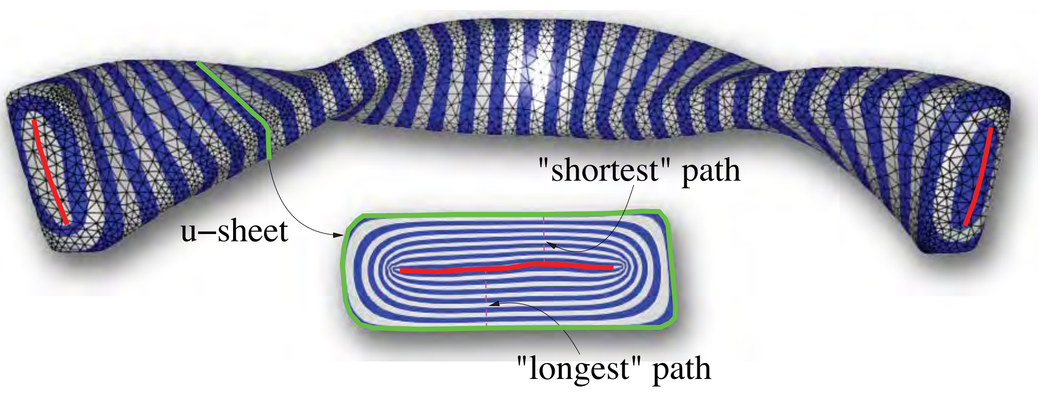


Figure 4.11. **w**-parameterizations using min/max paths.

resulting in less stretching and therefore more uniform quadrilaterals. This approach is investigated in detail in the following chapter.

For a given input mesh, its medial axis and our choice of the skeleton are related. Selecting the medial axis as the skeleton leads to difficulties since it may have small branches which would require splitting the object up into parts which have to be glued together. This would make modeling very difficult. The skeleton we pick is a simplified version of the medial axis. It uses this representation's ability to deal with overhangs and localized features in that simplification. The skeleton we choose can be regarded as approximation of the medial axis. For instance, in Figure 4.11, our choice of min/max paths yields a skeleton which is a surface similar to the medial axis of the object.

4.7 Experiments

Except for initial user-required choice of the critical points ν_{min}, ν_{max} and the initial seed to determine the skeleton, our modeling framework runs fully automatically. $\mathbf{u}_{\mathcal{T}_B}$ takes a few seconds to compute, allowing the user (being aware of simulation parameters) to try out different initial parameterizations. After that first step $\mathbf{u}_{\mathcal{H}}$ and $\mathbf{w}$ are computed, paths are extracted and the trivariate B-spline is generated from that. We implemented the proposed framework on a 16 node cluster, reducing the modeling time for the femur in Figure 4.2 to about 30 minutes.

We applied IA [87] in form of Linear elasticity [86] (see Figure 4.12) to a data reduced version of the resulting B-spline volume.

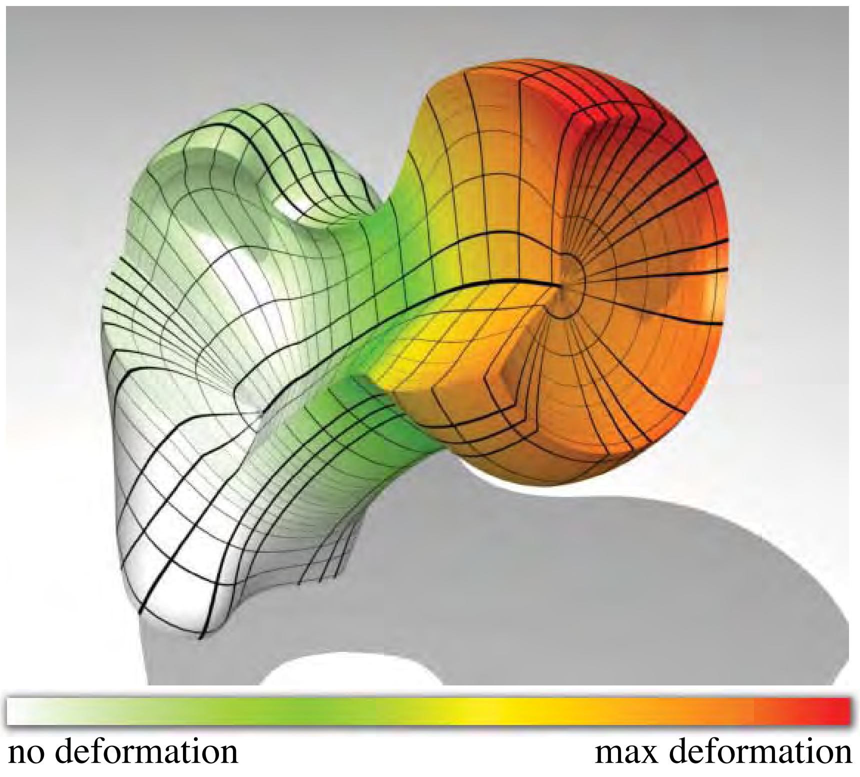


Figure 4.12. Isogeometric analysis: Elastostatics applied to the data reduced trivariate B-spline representation of the femur. Load is applied to the head of the femur.

Isocurves of a harmonic function defined on a smooth surface converge to circles when approaching a critical vertex. In case that the harmonic values are linearly interpolated across a triangle mesh, isocurves are nonplanar "n-gons," where n depends on the number of triangles the respective isocurve crosses. When approaching a critical point, isocurves are defined only by a few vertices as can be seen in Figure 4.3. In order to improve the parameterization in the regions near the critical points, additional vertices are inserted in the respective regions and retriangulation is applied in these areas. Similarly, inserting additional vertices in the region around the skeleton can help to improve the volumetric parameterization and therefore the quality of the initial structured hexahedral mesh.

4.8 Conclusion

In this chapter, a framework to model a single trivariate B-spline from input genus-0 triangle meshes was proposed. The final B-spline was computed using a novel iterative approximation approach which will be presented in Chapter 9, avoiding oscillations observed in B-spline interpolation. We guarantee that the slices defining the B-spline do not overlap and only have degeneracies only along the skeleton. Linear elasticity was applied to the resulting B-spline to demonstrate its practical use. Harmonic functions in three parameters are used to establish an initial parameterization suitable for tensor-product B-splines. This allows to model objects with overhangs such as the femur shown in Figure 4.2. However, modeling B-splines from triangle meshes with a higher genus or bifurcations requires an extension of our framework. This is subject of the next chapters of this work.

CHAPTER 5

OBJECTS WITH HIGHER GENUS AND BIFURCATIONS

The methodology described in this chapter generalizes the method proposed in the previous chapter, i.e., it presents a methodology to create higher order parametric trivariate representations such as B-splines or T-splines, from closed triangle meshes with higher genus or bifurcations. The input can consist of multiple interior boundaries which represent inner object material attributes. Fundamental to our approach is the use of a midsurface in combination with harmonic functions to decompose the object into a small number of trivariate tensor-product patches that respect material attributes. The methodology is applicable to thin solid models which we extend using the flexibility of harmonic functions and demonstrate our technique, among other objects, on a genus-1 pelvis data set containing an interior triangle mesh separating the cortical part of the bone from the trabecular part. Finally, a B-spline representation is generated from the parameterization.

Generating volumetric models suitable for simulation is an increasingly important task in geometric modeling. In practice a closed triangle mesh in 3-D space is given which represents the boundary of a physical domain. Commonly, interior triangle meshes are given that separate different materials within the physical domain. For simulation, the goal is to create volume elements in the region enclosed by the multiple boundary triangle meshes. This process is referred to as model completion. In practice, two choices are usually given: Filling up the volume using a unstructured grid representation based on tetrahedral or hexahedral elements or using a structured grid representation.

Unstructured tetrahedra meshes are often used in practice because they are fast, almost fully automatic and work on any geometric topology. However, these representations have drawbacks. For instance, multiresolution algorithms such as refinement on an unstructured representation are more difficult than on a structured grid. Furthermore, if higher-order elements are used in simulation, cross element continuity greater than C^0 is difficult to achieve. Also, some simulation methods such as linear elasticity give better results on hexahedral meshes than on tetrahedral meshes with the same degree of complexity.

Structured grids do not share these problems, i.e., refinement can be easily applied. Furthermore, the user can specify higher orders of continuity across elements more easily, making it easier to achieve higher continuity on a structured representation compared to an unstructured representation. Therefore, structured representations are desired for many types of finite element simulations such as IA [87]. However, structured grids also have limitations, in that they require more user interaction to create them and also depend more strongly on topology. This chapter presents a methodology to create structured representations for a class of objects as discussed below.

This chapter makes the following contributions:

- A method to establish a volumetric parameterization for objects represented with triangle meshes with higher genus and bifurcations based on a midsurface, suitable for both B-spline and T-spline fitting. The volumetric parameterization method is a hybrid technique inheriting positive aspects of both polar- and polycube- style parameterizations.

- An isoparametric methodology to parameterize the volume that respects interior boundaries of material attributes.

We assume that interior material boundaries are of similar geometric complexity as the exterior triangle mesh boundary and that the interior material boundaries are nested within each other. A demonstration of the approach on diverse objects such as a genus-1 pelvis and a genus-1 propeller is given. The pelvis data set consists of an interior triangle mesh to separate the interior soft (cortical) material from the hard (trabecular) boundary layer material. The parameterization is used to fit a trivariate

B-spline to the bone for a smooth transition from the cortical to the trabecular part of the bone.

Chapter 4 proposed a methodology to create trivariate representations of generalized cylinders which also allows representation of interior materials attributes. However, the approach works only on genus-0 objects and is suitable only for objects with smaller shape overhangs such as local bifurcations. The main reason for these limitations is that only one trivariate B-spline is used to create the trivariate representation to avoid the necessity of gluing patches. Similarly, Aigner et al. [4] apply this concept to a class of engineering objects, sweeping a parameterized planar surface along multiple guiding curves. The sweep in these approaches correspond to a single skeletal line. Given a topologically more complex object, a single skeletal line is unsuitable. First, it ignores the genus of the object and it ignores branches that can result in parametric distortions.

In the following discussion the reader is referred to Figure 5.1. There are two well understood ways to parameterize a disc. The first is a polar parameterization with high-quality elements close to the boundary, but with a degeneracy at the center. This type of parameterization is used in generalized cylinders [20] and was further generalized in [126]. The second way to parameterize a disc is to choose four corners at the boundary of the disc, and decompose the boundary into 4 iso-parametric segments. The interior elements have high quality, i.e., they are close to having right angles at the vertices, but the elements closer to the corners are more skewed. This may affect the quality of the physical simulation in those corner regions. Polycubes [193, 200, 201] are generalizations of this kind of parameterization.

In this chapter, we propose a blend of these two parameterization types, i.e., near to the boundary the parameterization is polar-like but in the interior, it has the advantages of polycube maps. In this way, there are no parametric or geometric degeneracies, i.e., each element is a geometric quadrilateral. Also, no corners are specified at the boundary and hence the quadrilaterals closer to the boundary have nearly right angles.

Our method reduces the dimensionality of the parameterization problem by parameterizing a simplified manifold base surface lying in the interior of the object using 2D parameterization techniques and using that to parameterize the volume. The size of

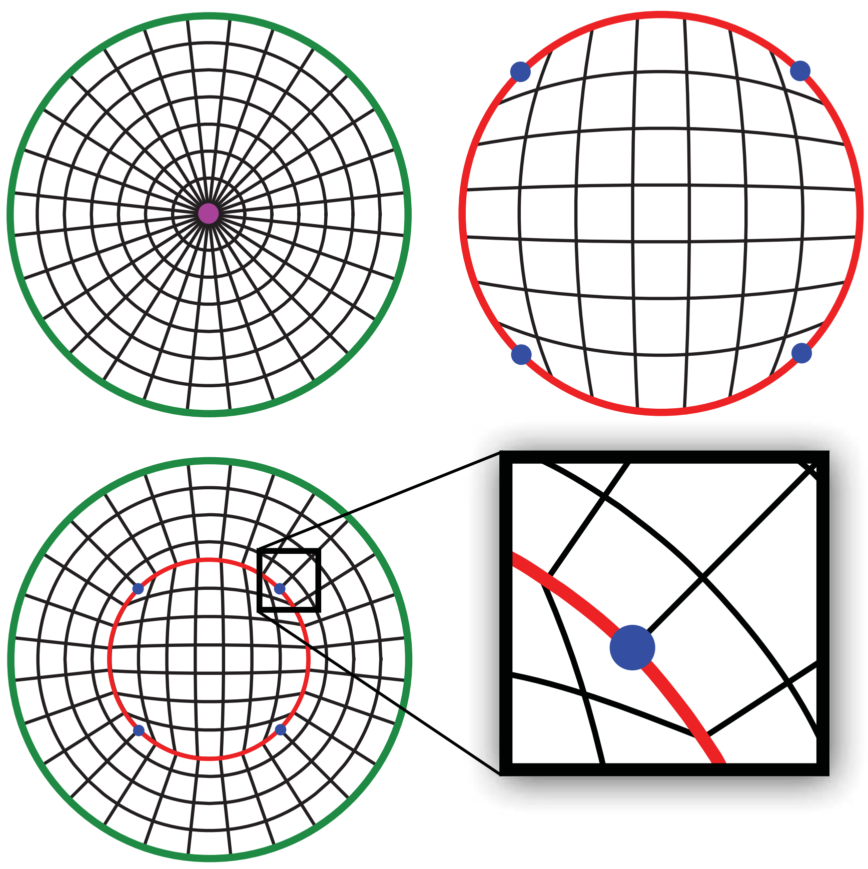


Figure 5.1. Parameterization of a disc. Top left: Polar-like with one degenerate point (magenta); Top right: Polycube-like with four corner points (blue); Bottom: Blend of the two parameterizations.

the base surface, derived from a midsurface, determines which of the above discussed parameterization methods are favored and can be controlled by the user according to requirements in simulation.

Midsurfaces, decomposing an object into two pieces, are common occurrences in modeling for simulation. Considerable efforts have been invested to find them on CAD models (e.g., [133]) consisting mostly of flat faces. Finding a midsurface for a more general object is an unsolved problem. In this chapter we make an effort to find midsurfaces suitable for volumetric parameterization. Musuvathy et al. [139] present a medial axis computation algorithm on 3D regions bounded by B-spline surfaces.

The choice of the midsurface and its subsequent parameterization affects the volumetric parameterization of the input object. While this reduces the modeling time significantly, the class of objects that can be parameterized, such as a hand, a pelvis or a genus-3 torus as shown in Figure 5.2, is extended. While these objects are not general solid models, we also show that our method can be applied to more general models such as a genus-1 propeller as shown in Figure 5.3.

An overview of our proposed framework is given in Section 5.2. Section 5.3 is concerned with the construction and parameterization of the midsurface used in Section 5.4 to decompose the object of interest. Section 5.5 discusses the construction of the volumetric parameterization of the object of interest. This chapter is concluded with results, discussion of limitations and extensions (Section 5.6 and 5.7).

5.1 Parameterization Strategies

The following paragraphs present an approach based on a midsurface, decomposing the object of interest into two regions that is used to create a volumetric parameterization from an exterior boundary that respects interior material boundaries in the parameterization. For better illustration of the proposed technique, we show two

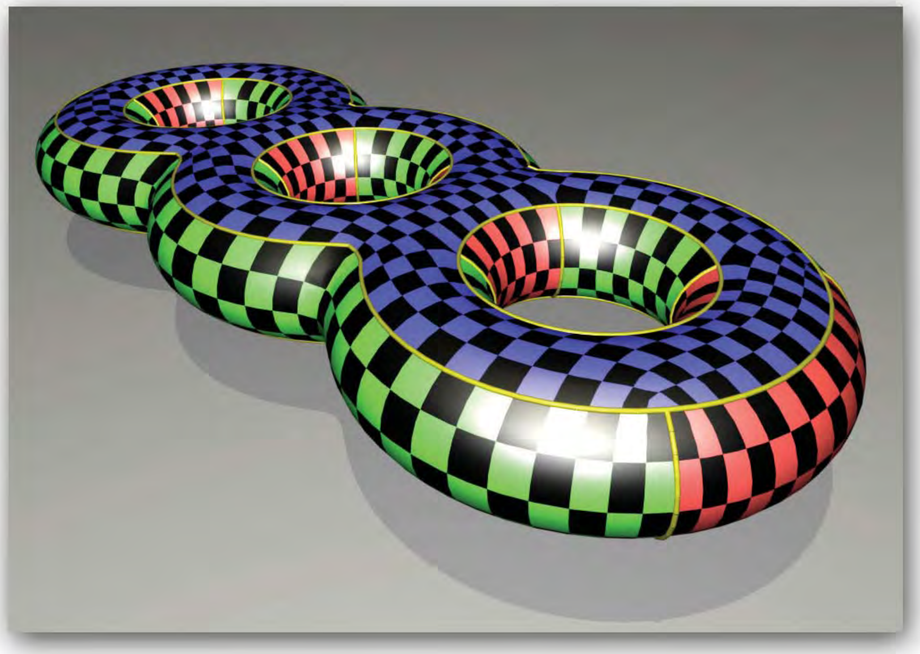


Figure 5.2. Parameterization of a genus-3 torus.

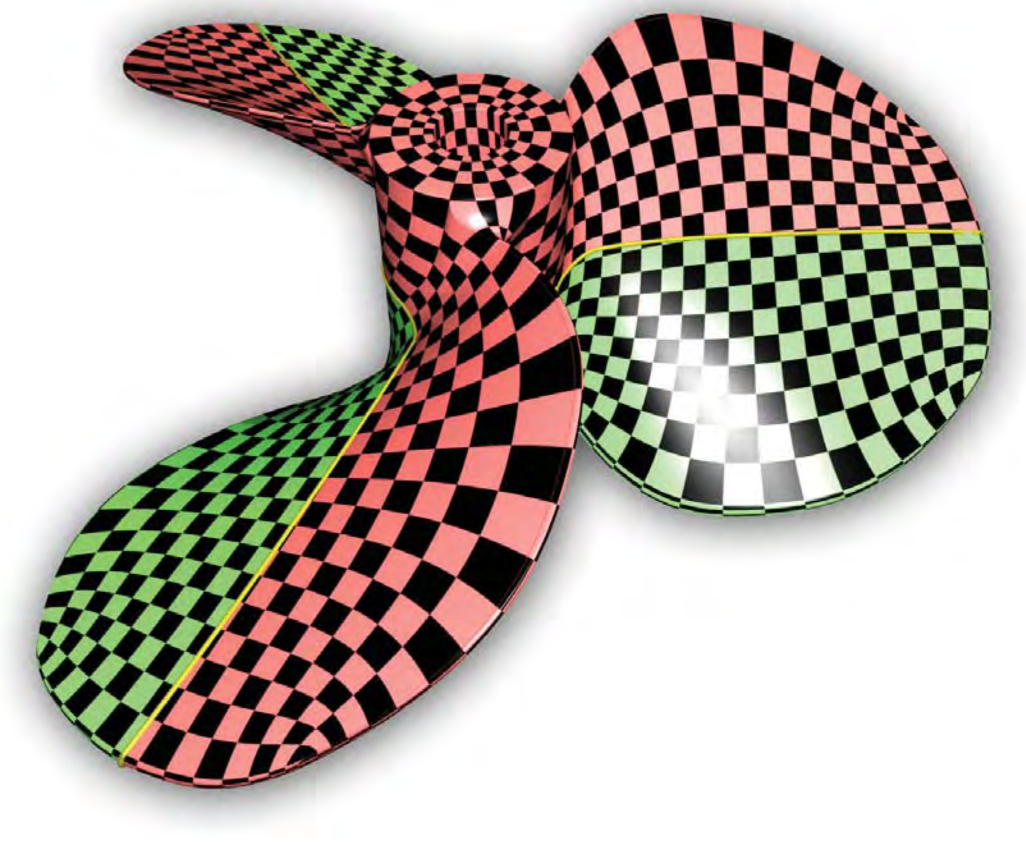


Figure 5.3. Parameterization of a CAD propeller of genus-1 with a single midsurface (midsurface boundary in yellow).

parameterization strategies on a 2D domain that has neither holes nor bifurcations.

Figure 5.4 illustrates the approach. The input consists of one closed polyline (piecewise linear line segments), defining the physical domain. The inner boundary is specified by the user as the interface between polar and square parameterization. The inner boundary has similar geometric complexity related to the outer boundary. The domain is discretized with triangles and the inner boundary is embedded in the triangulation. The goal is to parameterize this domain where, similar to the motivation presented in Chapter 4, interior boundaries are isoparametric. A midcurve that lies inside the innermost area, i.e., the area enclosed by the inner boundary, is constructed based on the outer polyline. Then, paths are traced from the corners of the midcurve to the outer polyline. This configuration decomposes the space into different regions (see Figure 5.4).

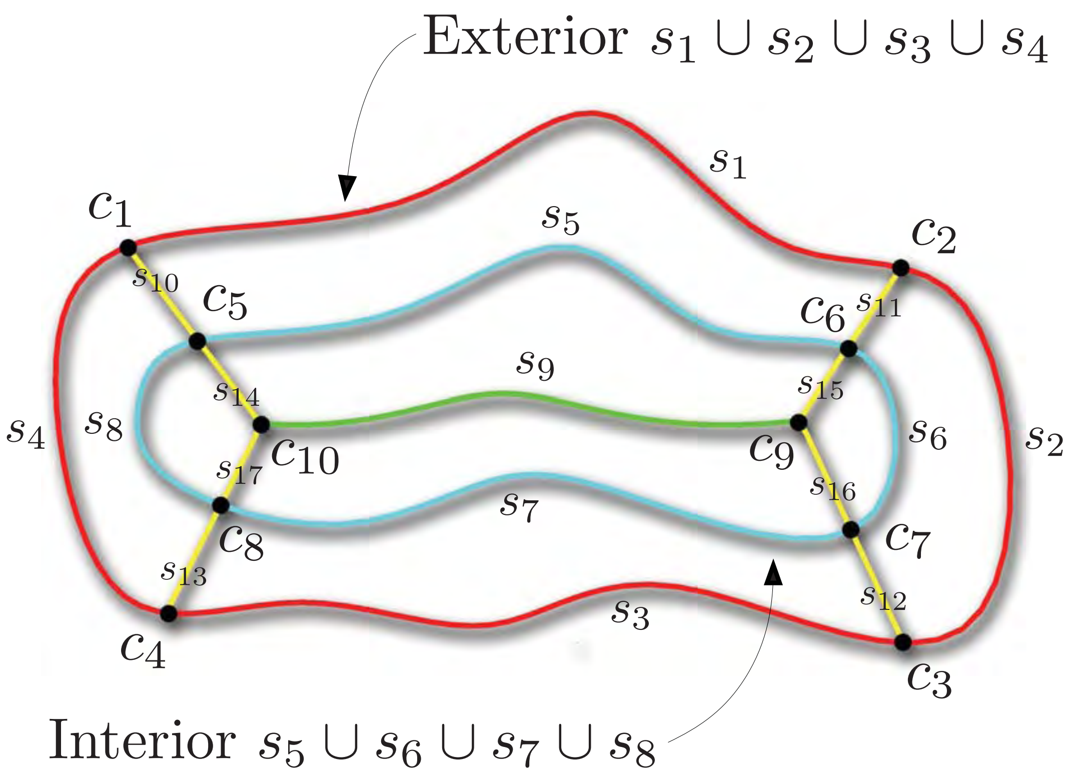


Figure 5.4. 2D input consisting of an exterior domain boundary. The inner boundary, which is the interface between the polar and square parameterization, is specified by the user. Boundaries are decomposed to divide the domain into subregions.

A region is enclosed in the general case by four boundaries. Harmonic functions are used to parameterize each region so that the parameterization of a boundary between two adjacent regions match.

In the following, α_i, $i = 1, 2, 3$ refer to the three regions as labeled in Figure 5.5a, and β_i for $i = 1, \ldots, 5$ refer to the five regions as labeled in Figure 5.5b. Note that $\Omega = \alpha_1 \cup \alpha_2 \cup \alpha_3 = \beta_1 \cup \ldots \cup \beta_5$, where Ω refers to the whole domain. Furthermore, s_i (Figure 5.4) are open polylines represented with vertices from an augmented triangle mesh representing Ω. The notation

$$\{s_i\}_i^n \leftarrow u = x$$

means that the scalar $x \in \mathbb{R}$ is assigned to the parametric direction u of the vertices of the collection of segments $s_1, s_2, \ldots, s_n$ as part of a Dirichlet boundary condition.

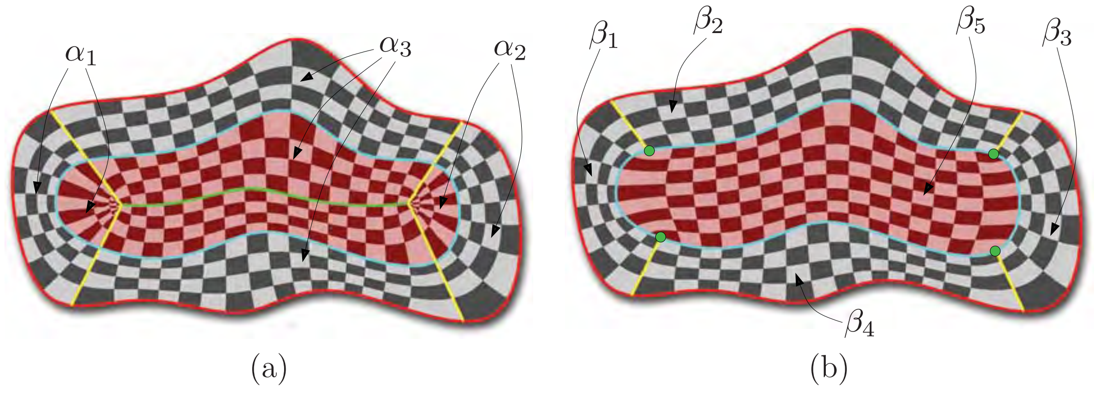


Figure 5.5. Parameterization strategies on 2D input.

5.1.1 Strategy 1

Laplace's Equation 1.1 is solved over Ω for the parametric v-direction with the following Dirichlet boundary: $\{s_1, s_2, s_3, s_4\} \leftarrow v = 1$ where $s_1 \cup s_2 \cup s_3 \cup s_4$ is the outer boundary; $\{s_5, s_6, s_7, s_8\} \leftarrow v = 0.5$ where $s_5 \cup s_6 \cup s_7 \cup s_8$ is the inner boundary. Finally, $\{s_9\} \leftarrow v = 0$, where s_9 is a midcurve of the outer boundary. For the parametric u-direction, Equation 1.1 is solved for the regions α_1, α_2 and α_3 independently (see Figure 5.5). Boundary conditions for α_1 are: $\{s_{10}, s_{14}\} \leftarrow u = 0$ and $\{s_{13}, s_{17}\} \leftarrow u = 1$. Boundary conditions are set up for α_2 analogously. Equation 1.1 is solved for α_3 by assigning $u = 0$ to $s_{10} \cup s_{14} \cup s_{13} \cup s_{17}$, and $u = 1$ to $s_{11} \cup s_{15} \cup s_{16} \cup s_{12}$.

5.1.2 Strategy 2

The parametric v-direction is constructed in two steps. Equation 1.1 is solved over the region $\beta_1 \cup \beta_2 \cup \beta_3 \cup \beta_4$ (see Figure 5.5) with the boundary condition $\{s_1, s_2, s_3, s_4\} \leftarrow v = 1$ and $\{s_5, s_6, s_7, s_8\} \leftarrow v = 0.5$. Then, Equation 1.1 is solved over region β_5 with boundary condition $\{s_5\} \leftarrow v = 0$ and $\{s_7\} \leftarrow v = 1$. The u-direction is constructed analogously: For region β_1, assign $\{s_{10}\} \leftarrow u = 0$ and $\{s_{13}\} \leftarrow u = 1$. For region β_2, $\{s_{10}\} \leftarrow u = 0$ and $\{s_{11}\} \leftarrow u = 1$. Boundary conditions are assigned accordingly to region β_3 and β_4. For the inner region β_5, $\{s_8\} \leftarrow u = 0$ and $\{s_6\} \leftarrow u = 1$.

Both strategies are based on the same decomposition of Ω, where both have advantages and disadvantages. In the first strategy, since α_1 and α_3 are enclosed by only three segments each, the resulting parameterization has a degeneracy at the corner c_9

and c_{10} of the midcurve. The second strategy avoids this degeneracy, since all regions have four boundary segments. However, the resulting parameterization contains four extraordinary points in the interior (Figure 5.5b). Establishing a smooth function is therefore difficult to accomplish. In 3D, next to extraordinary points there are also extraordinary edges having other than 4 subvolumes attached to it. A smooth representation around these edges is difficult to achieve. T-NURCCs [177], a generalization of T-splines have been proposed for surfaces for dealing with this problem.

5.2 Framework Overview

This section gives an overview of our methodology to parameterize domains in $\mathbb{R}^3$. Section 5.1 discussed two parameterization strategies in 2D. Based on a 2D input domain boundary, the enclosed volume is decomposed into regions by introducing additional segments s_i using a midsurface. While in 2D, these segments are open polylines, in 3D these segments correspond to triangulated surfaces called $\mathcal{S}_i$, the faces.

Our framework takes n closed input triangle meshes $\mathcal{T}_i$ for $i = 1, \ldots, n$, where $\mathcal{T}_{i+1}$ is nested within $\mathcal{T}_i$. $\mathcal{T}_1$ represents the boundary of the physical domain Ω and interior triangle meshes $\mathcal{T}_i$ for $i = 2, \ldots, n$ separate materials. The following framework steps describe the generation of a set of trivariate tensor-products which represent Ω and respect the $\mathcal{T}_i$.

1: Construct a base surface $\mathcal{M}$ with respect to $\mathcal{T}_1$, so that $\mathcal{M}$ lies within $\mathcal{T}_n$. Then, create a tetrahedral mesh $\mathcal{H}$ from $\mathcal{T}_1$ which embeds all the interior $\mathcal{T}_i$ and $\mathcal{M}$ as tetrahedral faces.

2: Create a harmonic scalar field w by solving Laplace's equation (1.1) so that interior material boundaries are respected, i.e., ∇w is orthogonal to $\mathcal{T}_i$.

3: Given ∇w, sweep segments from the boundary of $\mathcal{M}$ to form surfaces that decompose the volume enclosed by $\mathcal{T}_1$ into subvolumes.

4: Establish u- and v-harmonic scalar fields on subvolumes to parameterize $\mathcal{H}$. The w scalar field from Step 2 is used as the third parametric direction in the volumetric parameterization strategy.

5: Depending on the parameterization strategy, fit trivariate B-splines, T-splines or T-NURCCs, respectively, to the parameterized subvolumes.

The trivariate representation is constructed so that the boundaries closely approximate the input data. Due to the tensor-product nature of the representation, the resolution of the trivariate grids can be high. Therefore, as a postprocessing step, data reduction [120] is applied to the final trivariate representation.

5.3 Midsurface Construction

Chapter 2 discusses work that has been done to compute the medial axis of an object in 3D. In general, for a given triangle mesh, these approaches construct an approximate medial axis. For volumetric modeling however, often these medial axes are not suitable as they contain features like local fins or other nonmanifold topology. The reader is referred to Figure 5.6 which shows a medial axis generated from the Olivier hand data set using the tight cocone software [46]. Often, a time-consuming postprocessing step involving manual removal and change in medial axis topology is necessary to clean up the medial axis for it to be suitable for volumetric parameterization.

Often in modeling, a simplified medial axis is sufficient when it at least captures the topology of the object. For instance, the approach proposed in [126] is based on

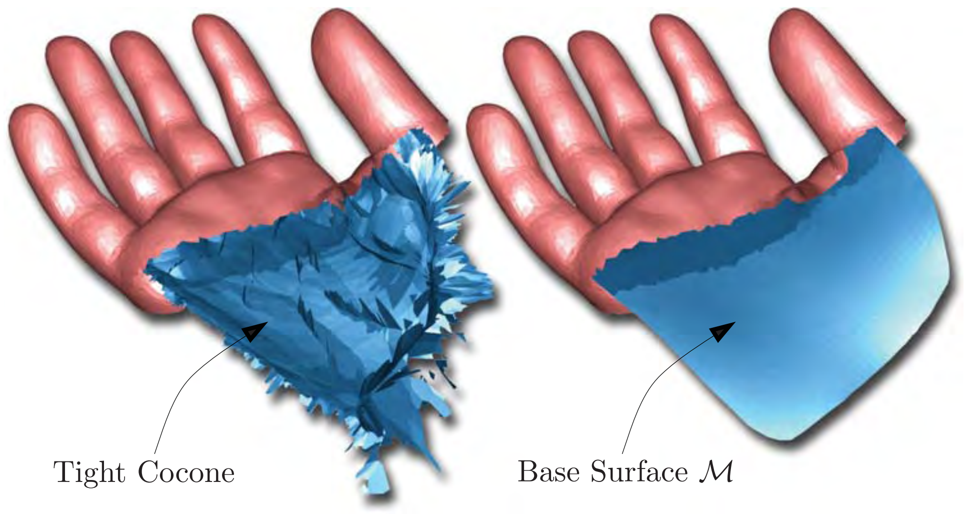


Figure 5.6. Simplified medial axis' of Olivier hand data set (left: tight cocone [46], right: our approach).

a skeleton line of the object. Harmonic functions were used as an aid to consistently fill up the interior, especially useful if the object has local overhangs far away from the medial axis.

In this section we construct a manifold midsurface $\mathcal{M}'$ (i.e., a single sheet) that has its boundary on $\mathcal{T}_1$ and decomposes each $\mathcal{T}_i$ into two volumes. $\mathcal{M}'$ is constructed to have no fins or other nonmanifold geometry. A base surface $\mathcal{M}$ is computed by trimming $\mathcal{M}'$. We call it a base surface, because its size and parameterization affects the resulting volumetric parameterization. With this approach to use a base surface for parameterization, the approach given in [126] is generalized.

Constructing $\mathcal{M}$ has four steps: (1) The midsurface sheet $\mathcal{M}'$ is extracted, thus decomposing each T_i into two regions (Section 5.3.1). (2) A trimmed $\mathcal{M}'$, $\mathcal{M}'_{trim}$, is computed using a scalar field defined on $\mathcal{M}'$ and parameterized (Section 5.3.2). (3) The medial axis of the parametric domain of $\mathcal{M}'_{trim}$ is formed (Section 5.3.3). (4) Points of the 2D medial axis of the parameter domain of $\mathcal{M}'_{trim}$ are inserted into $\mathcal{M}'_{trim}$ in order to construct the base surface $\mathcal{M}$ (Section 5.3.4).

5.3.1 Construction of $\mathcal{M}'$

Given $\mathcal{T}_1$ with genus g, the midsurface $\mathcal{M}'$ is constructed by defining a set $\mathbb{S} = \{\delta_1, \ldots, \delta_{g+1}\}$ each of whose elements is a closed path on $\mathcal{T}_1$. Together they decompose $\mathcal{T}_i$ into two regions. Path δ_j on $\mathcal{T}_1$ is a piecewise linear function defined by a sequence of vertices from $\mathcal{T}_1$ and corresponds to a boundary of $\mathcal{T}_1$'s midsurface $\mathcal{M}'$. We present the construction of $\mathcal{M}'$ by referring to Figure 5.7 for illustration. -1 is assigned to the first region (green) and $+1$ is assigned to the second region (red). This Dirichlet boundary condition is used to solve Equation 1.1 over Ω. Let f be the corresponding solution. Then, the isosurface $\mathcal{M}'$ at isovalue $f(x, y, z) = 0$ is extracted using marching tetrahedra [33]. Given the properties of Laplace's Equation (1.1), $\mathcal{M}'$ is guaranteed to lie within $\mathcal{T}_1$. Several approaches can be used to construct the paths defining the boundary of $\mathcal{M}'$, including, for example, (1) Extraction of salient ridge lines from $\mathcal{T}_1$, (2) Connection of critical points using Morse analysis. Both approaches require user assistance and are discussed below.

The set of ridge lines for a given object defines the complexity of its medial axis.

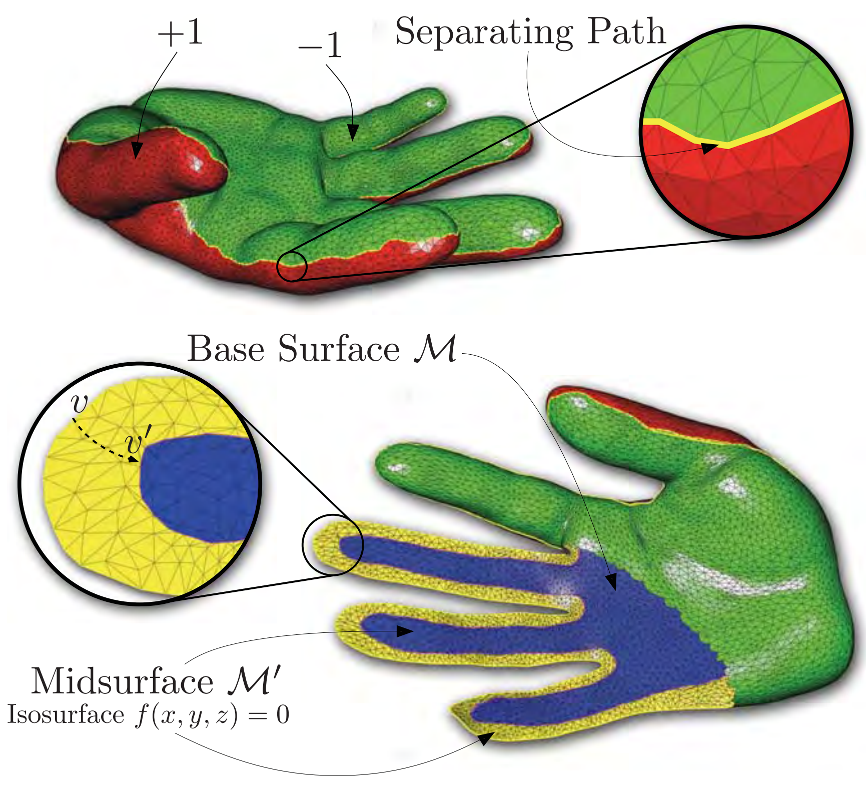

+1
−1
Separating Path
Base Surface $\mathcal{M}$
v
v'
Midsurface $\mathcal{M}'$
Isosurface $f(x, y, z) = 0$

Figure 5.7. Olivier hand model ($n = 1$): midsurface $\mathcal{M}'$ (yellow and blue area) and base surface (blue area). Since $n = 1$, $\mathcal{M}' = \mathcal{M}'_{trim}$.

However, only a few salient ridge lines characterize the global structure and topology of the object and medial axis. Ridge extraction techniques such as [148] can be used to extract ridges from $\mathcal{T}_1$. However, it is difficult to extract closed ridge lines on discrete data, since ridges require higher-order information. A user process is often necessary to close them in order to decompose $\mathcal{T}_1$ into two regions. The paths to decompose $\mathcal{T}_1$ can refer to closed ridges or crests requiring $\mathcal{T}_1$ to be doubly curved. For instance, if $\mathcal{T}_1$ is an ellipsoid, $\mathcal{M}'$ is a solid ellipse with boundary along the major equator of $\mathcal{T}_1$. In this case, the boundary of $\mathcal{M}'$ is equal to one of the ridge lines of $\mathcal{T}_1$.

In our framework, we follow the method by Ni et al. [143] to extract the topological structure of $\mathcal{T}_1$, that is, the user specifies critical points, i.e., minimum points and maximum points on $\mathcal{T}_1$ to construct a harmonic scalar field on $\mathcal{T}_1$. Discrete Morse analysis is used to find saddle points on $\mathcal{T}_1$. For every saddle, the gradient field of the harmonic field is used to create paths connecting the critical points, i.e., connecting saddles with saddles, saddles with minima and saddles with maxima. Paths reaching the same minima or maxima are removed. The bifurcation emanating at every saddle has a path which ends at either a maximum or a minimum or at a different saddle. Given such a path, a point on the opposite side (in case there is no path defined on that side already) is chosen automatically to create a new path which ends at a minimum or a maximum. The remaining paths can be joined at the critical points to create a single closed path to decompose $\mathcal{T}_1$ into two regions.

However, this approach sometimes fails to determine a set of desirable closed loops since determining the paths in $\mathbb{S}$ to decompose $\mathcal{T}_1$ into two pieces suitable for volumetric parameterization is a difficult and unsolved problem in general. Suitable in this context means that the final volumetric parameterization contains as little parametric distortion as possible. An alternative is to have the user just draw boundary curves onto $\mathcal{T}_1$ using a technique such as [15], as was for instance done to parameterize a propeller from a triangulated CAD representation (Figure 5.3).

Given a set of valid paths $\mathbb{S}$, the resulting scalar field f defined on $\mathcal{H}$, as computed above, can be used to formulate a quality measure on the choice of $\mathbb{S}$ in the following way. For every vertex v_k on $\mathcal{T}_1$, trace a path $\mathbf{h}_k$ through $\mathcal{H}$ using $+\nabla f$ or $-\nabla f$ depending on which side of $\mathcal{T}_1$ v_k lies. Let $\mathbb{H}$ be the set containing these paths. A good choice of $\mathbb{S}$ results in $\mathbb{H}$ with a small variance of its paths. A higher variance means that there are both shorter and longer paths directly affecting the parametric distortion of the end result.

Given a choice of paths $\mathbb{S}$, f and the resulting $\mathbb{H}$ can be computed relatively efficiently due to the linear basis used to compute f and the paths in $\mathbb{H}$ which change piecewise constantly over $\mathcal{H}$. This is a useful tool for the user to make a judgment on the choice of $\mathbb{S}$. It is clear that there are objects which cannot be modelled with a single midsurface, i.e., the approach works only on some classes of models.

5.3.2 Parameterization of Trimmed $\mathcal{M}'$

In this work we assume that $\mathcal{T}_{i+1}$ is contained within $\mathcal{T}_i$ for $i = 1, \ldots, n-1$. Furthermore, we require that $\mathcal{M}$ is contained within the innermost triangle mesh boundary $\mathcal{T}_n$. Therefore, we compute $\mathcal{M}'_{trim} = \mathcal{M}' \cap \mathcal{T}_n$, i.e., the boundaries of $\mathcal{M}'_{trim}$

lie on $\mathcal{T}_n$. Note if $n = 1$, $\mathcal{M}'_{trim} = \mathcal{M}'$. $\mathcal{M}'_{trim}$ is parameterized by adopting the 2D analogue version of polycubes given in [193] to a flattened version of $\mathcal{M}'$ using a flattening method such as presented in [179]. Similarly as in [201], the user picks corners on the boundaries of $\mathcal{M}'$ to decompose them into isoparametric sides acting as Dirichlet boundary conditions to solve the 2D version of Equation 1.1 in u and v. Then, the boundary of $\mathcal{M}'_{trim}$ is decomposed into a set of segments s_j where the two boundary vertices of s_j are 2D polycube corners. Each segment is isoparametric in u or v. As in the 3D case [201], the 2D polycube representation for $\mathcal{M}'_{trim}$ is parameterized in u and v. An example of the parameterization of $\mathcal{M}'_{trim}$ is given in Figure 5.8.

5.3.3 Construction of 2D Medial Axis on $\mathcal{M}'_{trim}$

Given the parameterized sheet $\mathcal{M}'_{trim}$ the medial axis of its 2D parameter domain boundary is computed. Since the domain has axis aligned boundaries this is quite straightforward. Those edges of the medial axis that extend to the boundary of the

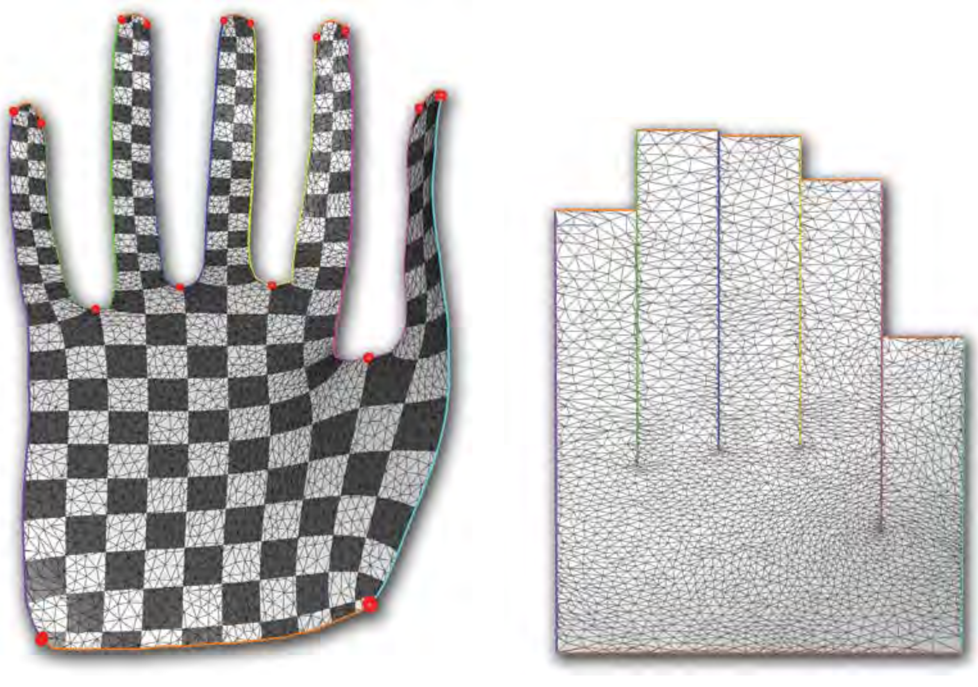


Figure 5.8. Polycube like parameterization of midsurface $\mathcal{M}'$ on hand model. (a) parameterization; (b) corresponding parameteric domain

parametric domain are removed leaving only the medial sheet curves.

5.3.4 Construction of $\mathcal{M}$

Then, the image of the medial axis computed in Section 5.3.3 in $\mathcal{M}'_{trim}$ is inserted into the mesh and used to construct a harmonic scalar field β evaluating to 1 at medial axis image points on $\mathcal{M}'_{trim}$ and 0 at the boundary of $\mathcal{M}'_{trim}$. The user is given the flexibility to specify an isovalue m_0 where the parts on $\mathcal{M}'_{trim}$ with $\beta(x, y, z) < m_0$ are removed to create the final $\mathcal{M}$. Figure 5.9 illustrates different choices of m_0 affecting the w-scalar field computed in Section 5.4 and hence the element shapes. Finally, $\nabla\beta$ is used to move the corners defined on the boundary of $\mathcal{M}'_{trim}$ to the boundary of $\mathcal{M}$. Figure 5.10 illustrates these steps.

5.4 Decomposition of Volume

The following presents the method for decomposing the volume enclosed by $\mathcal{T}_1$ into subvolumes, one subvolume on the positive side of $\mathcal{M}$, one on the negative side of $\mathcal{M}$, and a set of crest volumes around the g boundaries of $\mathcal{M}$ in Ω with genus g. Ω is decomposed using a surface $\mathcal{S}$ consisting of g disjoint surface pieces. $\mathcal{S}$ is the surface

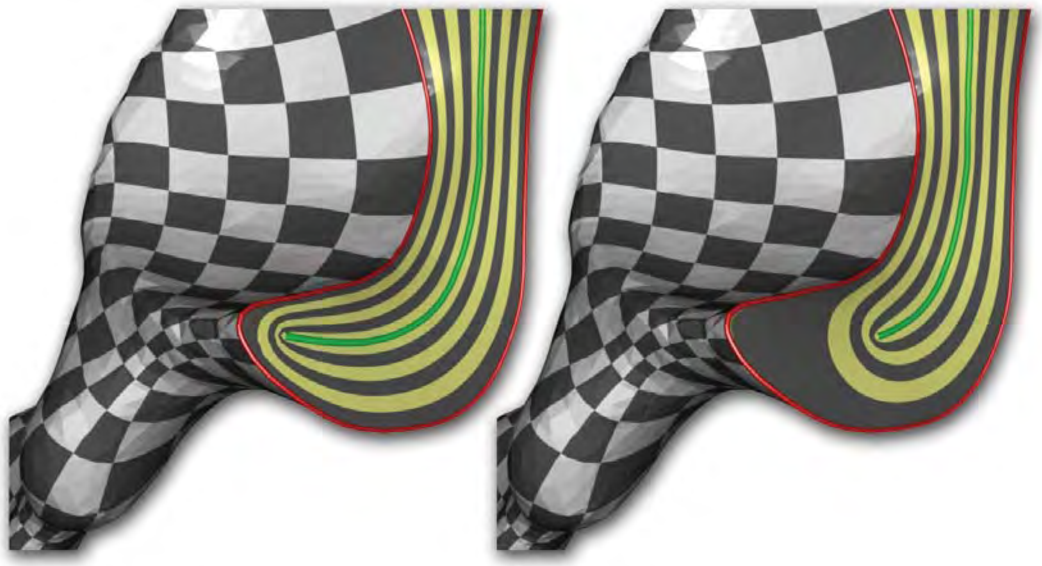


Figure 5.9. A trim value m_0 closer to 1 (right) can introduce parametric distortion, but favors the periodic parameterization. A smaller trim values favors the polycube parameterization.

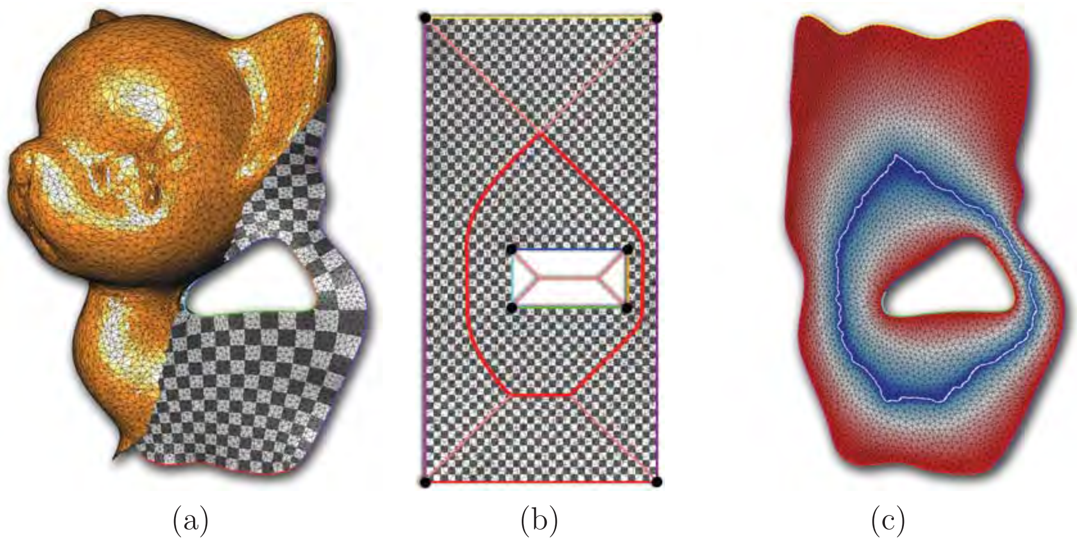


Figure 5.10. Parameterization of midsurface $\mathcal{M}'$ of kitten model. (a) Partial view of parameterized midsurface $\mathcal{M}'$; (b) Corresponding parameter domain; (c) Scalar field f is used to trim $\mathcal{M}'$.

shared by these subvolumes. The construction of $\mathcal{S}$ and hence the decomposition of Ω involves the construction of interior faces $\mathcal{S}^+$ and $\mathcal{S}^-$, where $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$. $\mathcal{S}^+$ and $\mathcal{S}^-$ are computed by following a harmonic gradient field for each of the g boundaries of $\mathcal{M}$ until $\mathcal{T}_1$ is reached, passing through the intermediate triangle meshes $\mathcal{T}_2, \ldots, \mathcal{T}_n$ along the way. Figure 5.4 shows the 2D equivalent of this decomposition.

Since the base surface $\mathcal{M}$ respects only the topological structure of Ω, unlike the true medial axis, the generation of these faces is challenging because in some regions, depending on the distortion, a face has to move more than in regions where $\mathcal{M}$ is closer to $\mathcal{T}_1$. In the following we assume that $\mathcal{T}_i$ and $\mathcal{M}$ have a feature aware and regular triangulation. Then, $\mathcal{S}^+$ and $\mathcal{S}^-$ are merged to form $\mathcal{S}$ and then decomposed into faces that relate to the corners of $\mathcal{M}$. The construction of $\mathcal{S}^+$ and $\mathcal{S}^-$ and the resulting decomposition of $\mathcal{S}$ is discussed in Section 5.4.1 and Section 5.4.2.

5.4.1 Construction of Faces $\mathcal{S}^+$ and $\mathcal{S}^-$

$\mathcal{S}^+$ and $\mathcal{S}^-$ are robustly constructed by solving Laplace's equation (1.1) over $\mathcal{H}$ by constructing a harmonic scalar field w. $\mathcal{H}$ contains $\mathcal{M}$ and all the $\mathcal{T}_i$ as submeshes with the Dirichlet boundary conditions that $w = 0$ is assigned to T_1 and $w = 1$ is assigned to $\mathcal{M}$. Furthermore, $w = (i-1)/n$ is assigned to the interior boundary $\mathcal{T}_i$. Given a point p on $\mathcal{M}$ and its normal $\vec{n}$, ∇w can be used to trace two paths emanating from $\mathcal{M}$ and ending on $\mathcal{T}_1$. The start positions of these paths are $p + \epsilon\,\vec{n}$ and $p - \epsilon\,\vec{n}$, respectively, where ϵ is a small number and $\vec{n}$ is the normal of p. The harmonic nature of w guarantees that these paths do not contain loops or end at local minima within Ω.

Let $\mathbb{P} = \{p_1, p_2, \ldots, p_n\}$ be the vertices of a piecewise linear boundary of $\mathcal{M}$ where p_i are the boundary points. From $\mathbb{P}$, two additional polylines $\mathbb{P}^+ = \{p_1^+, p_2^+, \ldots, p_n^+\}$ and $\mathbb{P}^- = \{p_1^-, p_2^-, \ldots, p_n^-\}$ are constructed as discussed next. $\nabla w(p_i) = \vec{b}_i$, where $p_i \in \mathbb{P}$ and $\vec{n}_i$ is the normal at p_i. $\vec{t}_i$ is the boundary tangent on $\mathcal{M}$ at p_i and $\vec{b}_i$ is the corresponding binormal (see Figure 5.11).

Given p_i, we find p_i^+ and p_i^- (not shown in Figure 5.11) so that the angle between $\nabla w(p_i^+)$ and $\vec{b}_i$ is $\approx \theta$ and between $\nabla w(p_i^-)$ and $\vec{b}_i$ is $\approx -\theta$. The user choice of θ affects

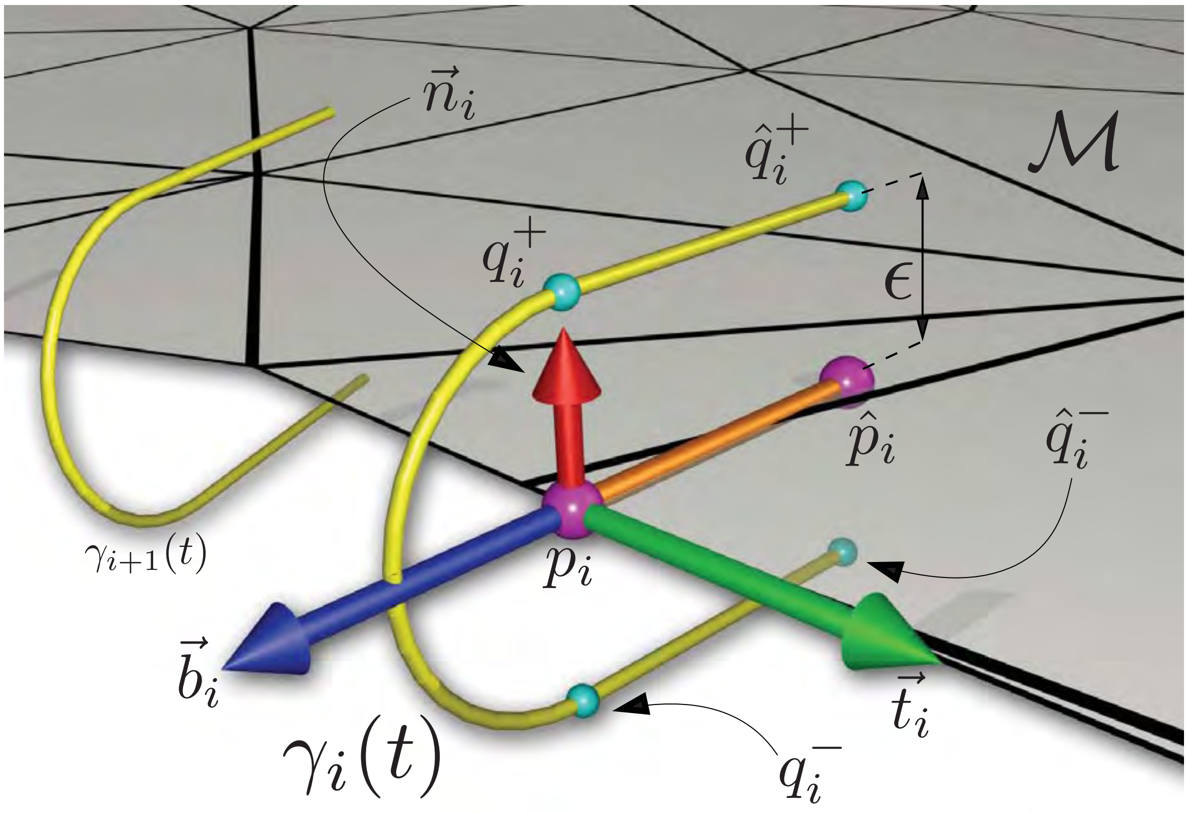


Figure 5.11. Construction of $\mathbb{P}^+ = \{p_1^+, \ldots, p_n^+\}$ and corresponding $\mathbb{P}^- = \{p_1^+, \ldots, p_n^+\}$. p_i^+ and p_i^-, respectively, are evaluations of its corresponding $\gamma_i(t)$ and not shown in this figure.

the crest region, i.e., how close the final $\mathcal{S}^+$ and $\mathcal{S}^-$ are to each other. For the models used in this chapter $\theta = \pi/4$ is a good choice. Based on that, p_i^+ and p_i^- are determined as discussed next.

As shown in Figure 5.11, $\hat{\mathbb{P}} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n\}$, where $\hat{p}_i = p_i - \epsilon\, \vec{b}_i$ for all $p_i \in \mathbb{P}$. A curve $\gamma_i : [0,1] \to \mathbb{R}^3$ is defined by

$$\gamma_i(t) = \begin{cases} (1 - 3\,t)\, \hat{q}_i^+ + 3\,t\, q_i^+, & 0 \le t < \frac{1}{3} \\ M\, c(t) + p_i, & \frac{1}{3} \le t < \frac{2}{3} \\ (3 - 3\,t)\, q_i^- + (3\,t - 2)\, \hat{q}_i^-, & \frac{2}{3} \le t < 1, \end{cases} \tag{5.1}$$

where $c(t) = h\,(\sin((3\,t - 3)\,\pi), 0, \cos((3\,t - 3)\,\pi))$ and M is the 3×3 rotation matrix defined by $\vec{n}_i$, $\vec{b}_i$ and $\vec{t}_i$. Furthermore, $\hat{q}_i^+ = \hat{p}_i + \epsilon\, \vec{n}_i$, $q_i^+ = p_i + \epsilon\, \vec{n}_i$ and $\hat{q}_i^- = \hat{p}_i - \epsilon\, \vec{n}_i$, $q_i^- = p_i - \epsilon\, \vec{n}_i$.

Point $p_i^+ = \gamma_i(\bar{t})$, where $\bar{t} \in [0, 1/2]$, is determined such that the angle between $\gamma_i(\bar{t})$ and $\nabla w(\gamma_i(\bar{t}))$ is close to θ. p_i^- is determined analogously on the interval $[1/2, 1]$.

By referring to Figure 5.12, surface $\mathcal{S}^+$ is constructed incrementally: A surface mesh is generated where the polylines $\mathbb{P}_0^+ := \mathbb{P}$ and $\mathbb{P}_1^+ := \mathbb{P}^+$ define quadrilateral elements having one edge at the boundary of $\mathcal{M}$. Row $\mathbb{P}_k^+$ is determined by computing, for every point $p^+ \in \mathbb{P}_{k-1}^+$, a line segment using ∇w starting at p^+ and ending at a face of the

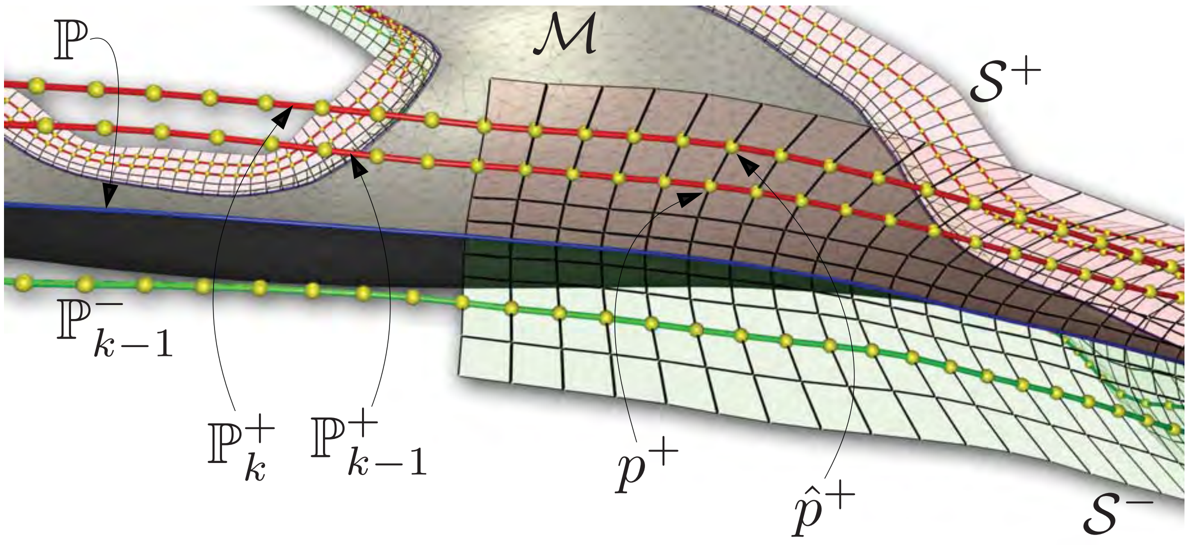


Figure 5.12. Construction of $\mathcal{S}^+$ and $\mathcal{S}^-$.

tetrahedron in which p^+ lies (pathological cases are robustly handled as in Chapter 4). The shortest line segment with length d determines the amount $\mathbb{P}_k^+$ travels through $\mathcal{H}$, i.e., $|\hat{p}^+ - p^+| = d$ where $\hat{p}^+$ is the new point in $\mathbb{P}_k^+$ corresponding to p^+ and lying on the line segment corresponding to p^+.

Once a new row has been computed it is triangulated with the previous row. Furthermore, if the distance between two adjacent points in $\mathbb{P}_k^+$ gets twice as large as the corresponding points in $\mathbb{P}_0^+$, then a new column is inserted between them emanating at row k. Accordingly, columns are removed when the distance of these points is less than half the distance of the corresponding points in $\mathbb{P}_0^+$. $\mathcal{S}^-$ is analogously constructed using $\mathbb{P}$ and $\mathbb{P}^-$.

5.4.2 Construction of Faces $\mathcal{S}_j$

Once $\mathcal{S}^+$ (red surface in Figure 5.13) and $\mathcal{S}^-$ (green surface in Figure 5.13) have been computed they are merged into the single face $\mathcal{S}$ for each loop on the boundary of $\mathcal{M}$. After this process has been applied to all boundaries of $\mathcal{M}$, the surfaces connecting the boundaries of $\mathcal{M}$ with $\mathcal{T}_1$ decompose the volume enclosed by $\mathcal{T}_1$ into three subvolumes: 1) the volume $\mathcal{H}^1$ enclosed by all $\mathcal{S}^+$, $\mathcal{M}$ and $\mathcal{T}_1$, 2) the volume $\mathcal{H}^2$ enclosed by all $\mathcal{S}^-$, $\mathcal{M}$ and $\mathcal{T}_1$, and 3) the crest volume $\mathcal{H}^3$ from the volume which remains, i.e., $\mathcal{H}^3 = \overline{\mathcal{H} \setminus (\mathcal{H}^1 \cup \mathcal{H}^2)}$ ($\overline{\mathcal{A}}$ is the closure of A). Note that $\mathcal{H}^3$ consists of $g+1$ disjoint pieces (g is the genus of $\mathcal{T}_i$).

The crest volumes are further divided: For every corner vertex c_j defined on the boundary of $\mathcal{M}$ (see Section 5.3.2), a corner surface $\mathcal{C}_j$ (yellow surfaces in Figure 5.13) is computed. $\mathcal{C}_j$ connects c_j to $\mathcal{T}_1$ by passing through all the interior $\mathcal{T}_i$. One boundary curve of $\mathcal{C}_j$ lies on $\mathcal{S}^+$, the other on $\mathcal{S}^-$. $\mathcal{C}_j$ is computed analogously to $\mathcal{S}^+$ and $\mathcal{S}^-$, by using scalar field w to propagate a polyline through the volume starting at c_j till $\mathcal{T}_1$ is reached. Given the corners surfaces $\mathcal{C}_j$, $\mathcal{S}$ can be decomposed into faces $\mathcal{S}_i$, where the face $\mathcal{S}_i$ corresponds to the segment i of $\mathcal{M}$ as classified in Section 5.3.2. Furthermore, $\mathcal{S}_i$ and the two corner surfaces at its boundary enclose a surface on $\mathcal{T}_1$ (Figure 5.13). These surfaces on $\mathcal{T}_1$ and the faces $\mathcal{S}_i$ are used to establish a parameterization over Ω as discussed in the following Section.

The automatic approach proposed above to construct $\mathcal{S}^+$ and $\mathcal{S}^-$ is consistent, i.e.,

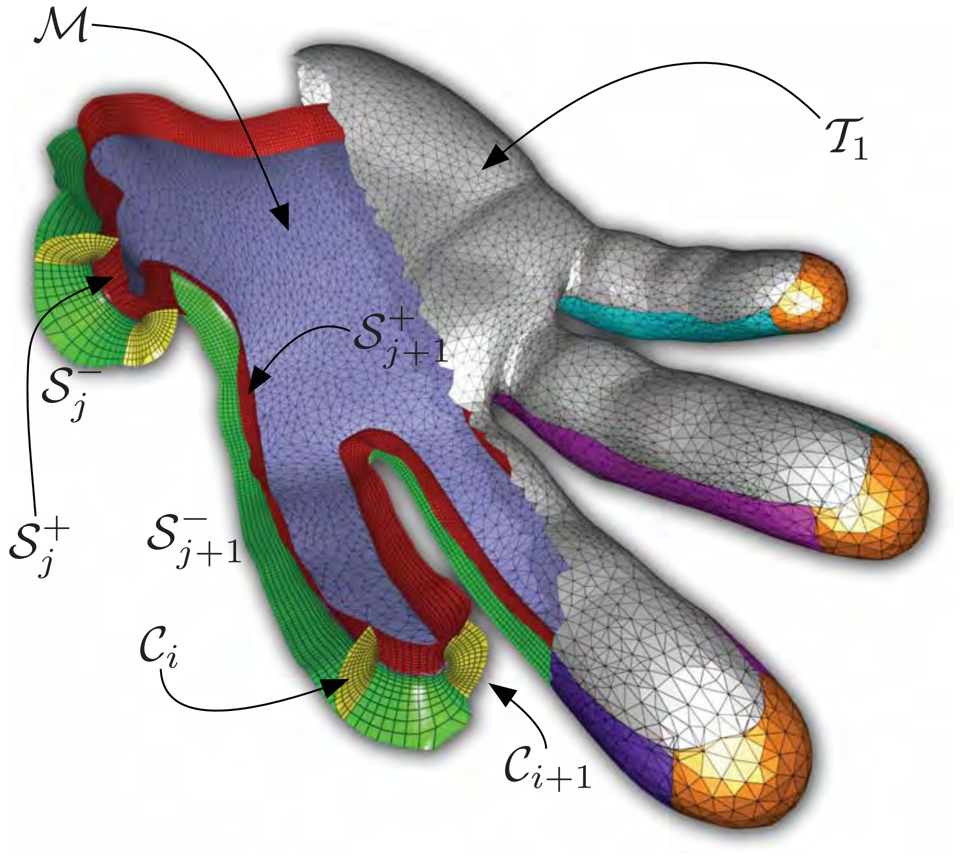


Figure 5.13. Decomposition of Olivier hand data set.

∇w flows from $\mathcal{M}$ to $\mathcal{T}_1$ without any local sinks as guaranteed by the choice of boundary conditions used to compute w, and it also guarantees that $\mathcal{S}^+$ and $\mathcal{S}^-$ reach the exterior $\mathcal{T}_1$ orthogonally, passing orthogonally through the interior boundaries $\mathcal{T}_i$ $(i = 2, \ldots, n)$. This means that the surfaces shared by two adjacent subvolumes are orthogonal to the boundaries $\mathcal{T}_i$, resulting in more orthogonal parameterizations in these regions. This approach makes establishing continuity between two adjacent subvolumes easier, as discussed below.

5.5 Volumetric Parameterization

In this section the method to volumetrically parameterize the domain Ω represented with $\mathcal{H}$ is presented. The 2D parameterization strategies from Section 5.1 are extended to 3D. While in 2D, the boundaries were defined by polylines referred to as segments, in 3D, boundaries are represented with faces $\mathcal{S}_i$ as discussed above. Furthermore, as in 2D, the interior material boundaries $\mathcal{T}_i, i = 2, \ldots, n$ are respected in the parameterization. Both strategies make use of the w-scalar field used to extract the faces $\mathcal{S}$.

5.5.1 Strategy 1

This strategy uses the parametric direction w computed in the previous Section and requires only computing u- and v- scalar fields over $\mathcal{H}$. As discussed above, every boundary segment s_i of $\mathcal{M}$ (see Figure 5.13) corresponds to a face $\mathcal{S}_i$ and is constant value of either u or v. $\mathcal{T}_1$, $\mathcal{S}_i$ and the two corner faces adjacent to $\mathcal{S}_i$ together form the boundary of either a crest subvolume $\mathcal{H}_i^{ucrest}$ or $\mathcal{H}_i^{vcrest}$. Let

$$\mathcal{H}^{ucrest} = \bigcup_{i=1}^{m} \mathcal{H}_i^{ucrest}, \tag{5.2}$$

where m is the number of faces $\mathcal{S}_i$. $\mathcal{H}^{vcrest}$ is defined analogously.

Next, the scalar fields for u and v are constructed. They are constructed analogously, so we present only the construction for u. For all edge segments s_i of $\mathcal{M}$ with constant u-value u_i, assign u_i to its respective $\mathcal{S}_i$ embedding s_i. Solve Equation 1.1 on $\mathcal{H}^u$, where $\mathcal{H}^u = \overline{\mathcal{H} \setminus \mathcal{H}^{ucrest}}$.

Now, the subvolume $\overline{\mathcal{H} \setminus (\mathcal{H}^{ucrest} \cup \mathcal{H}^{vcrest})}$, i.e., $\mathcal{H}$ without the crest subvolumes, has a parameterization. $\mathcal{H}^{ucrest}$ has a parametric v-scalar field, and $\mathcal{H}^{vcrest}$ has a corresponding u-scalar field. Similar to the 2D case, the remaining scalar field is constructed by assigning a constant parameter value to $\mathcal{S}^+$ and a constant parameter value to $\mathcal{S}^-$ followed by solving Laplace's Equation as above. This step can be seen as consistently sweeping $\mathcal{S}^+$ to $\mathcal{S}^-$ through the crest region of $\mathcal{H}$, and since the first row of $\mathcal{S}^+$ and $\mathcal{S}^-$ are the same, i.e., the first row lies on the boundary of $\mathcal{M}$, the resulting parameterization has a degeneracy along $\mathcal{M}$.

5.5.2 Strategy 2

This strategy results in a parameterization with no degeneracies, but with several internal extraordinary points. A value w_0 is picked on the w- scalar field that is less than the w-value of $\mathcal{T}_n$ and the respective isosurface, called $\mathcal{T}_{n+1}$, at w_0 is extracted.

The parameterization for the volume enclosed by $\mathcal{T}_1$ and $\mathcal{T}_{n+1}$ follows strategy 1 while the parameterization for the volume enclosed by $\mathcal{T}_{n+1}$ is done differently. The choice of w_0 often depends on simulation requirements, i.e., its choice affects how much the resulting volumetric parameterization is polar like versus polycube like. As in the 2D case (Figure 5.5b), in this strategy, $\mathcal{S}^+$ and $\mathcal{S}^-$ emanate from $\mathcal{T}_{n+1}$ rather than from $\mathcal{M}$, by trimming those parts of $\mathcal{S}^+$ and $\mathcal{S}^-$ lying within the innermost triangle mesh boundary $\mathcal{T}_{n+1}$ in $\mathcal{H}$ away. This implies that $\mathcal{M}$ is not part of the resulting representation.

As discussed above, every isoparametric segment s_j of $\mathcal{M}$ corresponds to a face $\mathcal{S}_j$ and two corner surfaces corresponding to its endpoints. Lets call this configuration $\hat{\mathcal{S}}_j$. Let $\mathcal{T}_{j,n+1}$ be the part of $\mathcal{T}_{n+1}$ inside the crest region $\hat{\mathcal{S}}_j$. Figure 5.13 shows $\mathcal{T}_{j,n+1}$ in color for the case $j = 1$. The union of all $\mathcal{T}_{j,n+1}$ refers to the crest region $\mathcal{T}_{n+1}^{crest}$ of $\mathcal{T}_{n+1}$. Furthermore, let $\{\mathcal{T}_{n+1}^+, \mathcal{T}_{n+1}^-\} = \mathcal{T}_{n+1} \setminus \mathcal{T}_{n+1}^{crest}$ where $\{\mathcal{T}_{n+1}^+$ refers to the sub triangle mesh of $\mathcal{T}_{n+1}$ on the side where $\mathcal{S}^+$ passes through $\mathcal{T}_{n+1}$. $\mathcal{T}_{n+1}^-$ is the sub triangle mesh of $\mathcal{T}_{n+1}$ on the side where the collection of $\mathcal{S}^-$ pass through $\mathcal{T}_{n+1}$. The w-scalar field over $\mathcal{T}_{n+1}$ is computed by assigning w_0 to $\mathcal{T}_{n+1}^-$ and w_1 to $\mathcal{T}_{n+1}^+$ where $w_0 < w_1$ followed by solving Equation 1.1.

Then, depending whether segment s_j of $\mathcal{M}$ is isoparametric in u or in v, the respective isovalue is assigned to the vertices defining $\mathcal{T}_{j,n+1}$. The u- and v- scalar field can be computed for the volume enclosed by $\mathcal{T}_{n+1}$ by solving Equation 1.1 with these boundary conditions. Figure 5.14 shows a part of the volumetric parameterization for the hand model using this parameterization strategy.

5.6 Modeling Examples

In the following paragraphs we demonstrate our proposed methodology by establishing a volumetric parameterization on a genus-1 pelvis data set consisting of an exterior triangle mesh boundary and an interior material boundary as illustrated in Figure 1.1 on the right) using the first parameterization strategy as discussed above. Parameterized subvolumes adjacent to each other are glued with $C^{(0)}$ continuity.

Given $\mathcal{M}$, several parameterization choices are possible. Specific corner selections

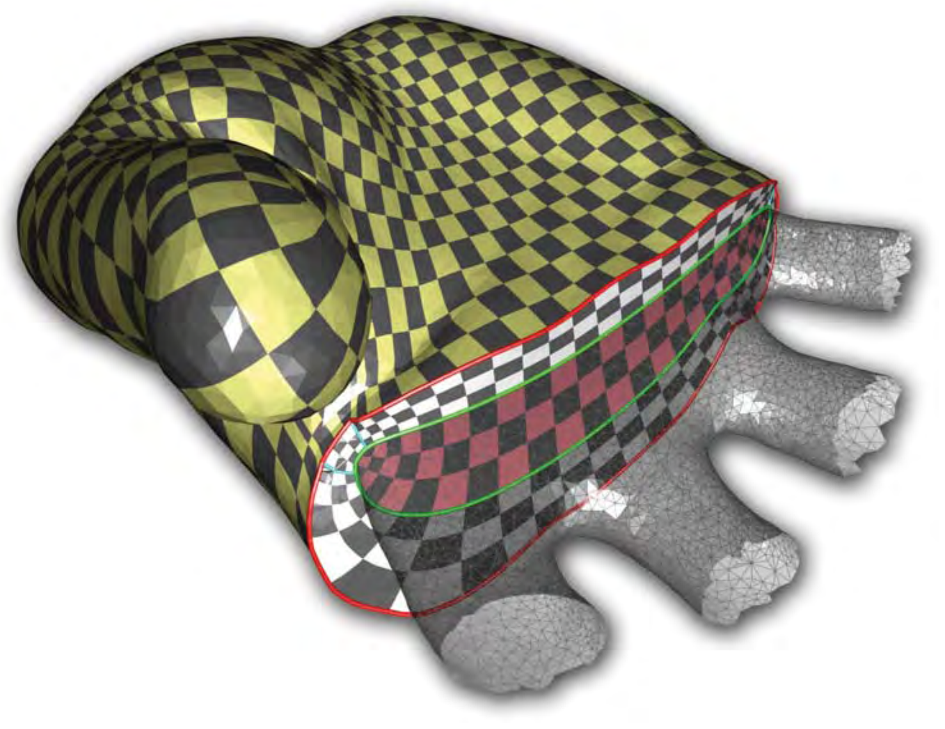


Figure 5.14. Second volumetric parameterization strategy for Olivier hand data set. (Parametric cut in u, showing a part of the interior material boundary.)

may result in a volumetric parameterization of $\mathcal{H}$, such that the geometry of $\mathcal{H}$ is followed more naturally. The reader is referred to Figure 5.15 showing two different u-scalar fields on a schematic genus-1 midsurface similar to $\mathcal{M}$ of the pelvis. A parameterization of $\mathcal{M}$ on the right in Figure 5.15 is chosen so that the handle region of $\mathcal{M}$ is similar to a sweep kind of parameterization, where the bottom part of $\mathcal{M}$ is similar to a polycube kind of parameterization. The resulting scalar field follows the geometry of the midsurface more naturally compared to the scalar field on the left of Figure 5.15 which is polycube like.

Midsurfaces have been applied mainly to thin solids. We demonstrate our technique on a triangulated CAD data set of a propeller (Figure 5.3) where the user defined the boundary of $\mathcal{M}$ manually. A usual midsurface would be more complex, having a sheet for the hub and fins for each of the blades. Instead, the chosen midsurface slices the propeller across the hub resulting in a small midsurface. With a minimal user interaction to choose the size of $\mathcal{M}$ and corners on it, the object could be parameterized

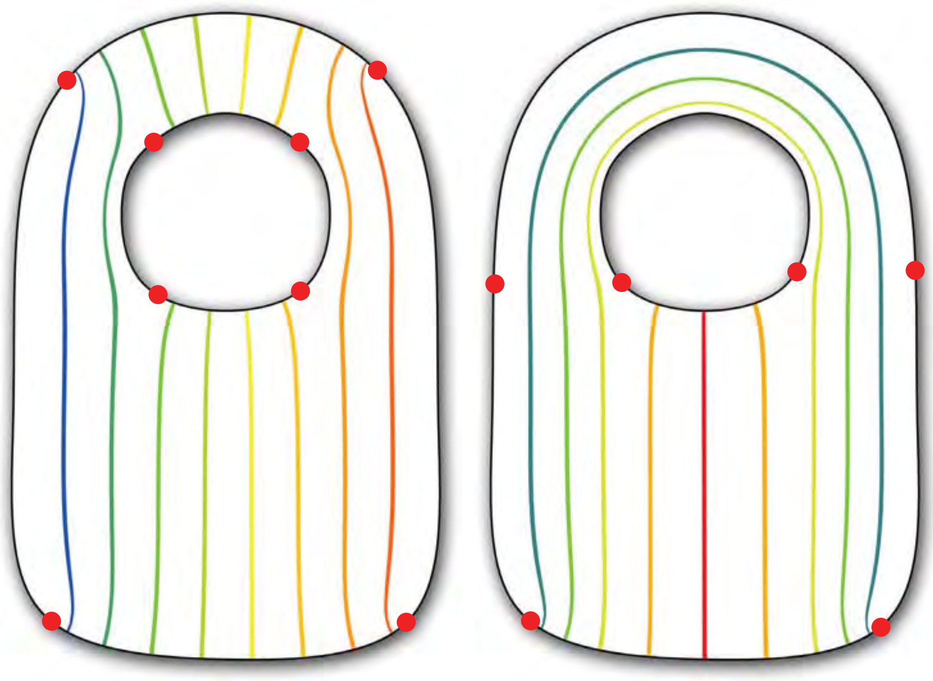


Figure 5.15. Different choices of corners (red), resulting in two different scalar fields on a midsurface.

efficiently. This example shows that the proposed midsurface based parameterization technique can model more than thin solids (Figure 5.16).

5.7 Conclusion

In this chapter we proposed a methodology to create volumetric parameterizations of triangle meshes with interior material boundaries. The algorithm presents a generalization of the method proposed in [126] and two parameterization strategies suitable to fit trivariate tensor-product B-splines or T-splines to the respective volumetric parameterization. Note that a B-spline representation can be converted into a T-spline representation. Once converted, local refinement on the T-spline can be used to increase the accuracy of the fit.

The volumetric parameterization is based on a midsurface, constructed as part of the algorithm and does not require the time consuming clean-up procedures that are often required when simplifying a medial axis. The use of harmonic functions allows the use of a relatively simple midsurface for more complex geometry such as the pelvis

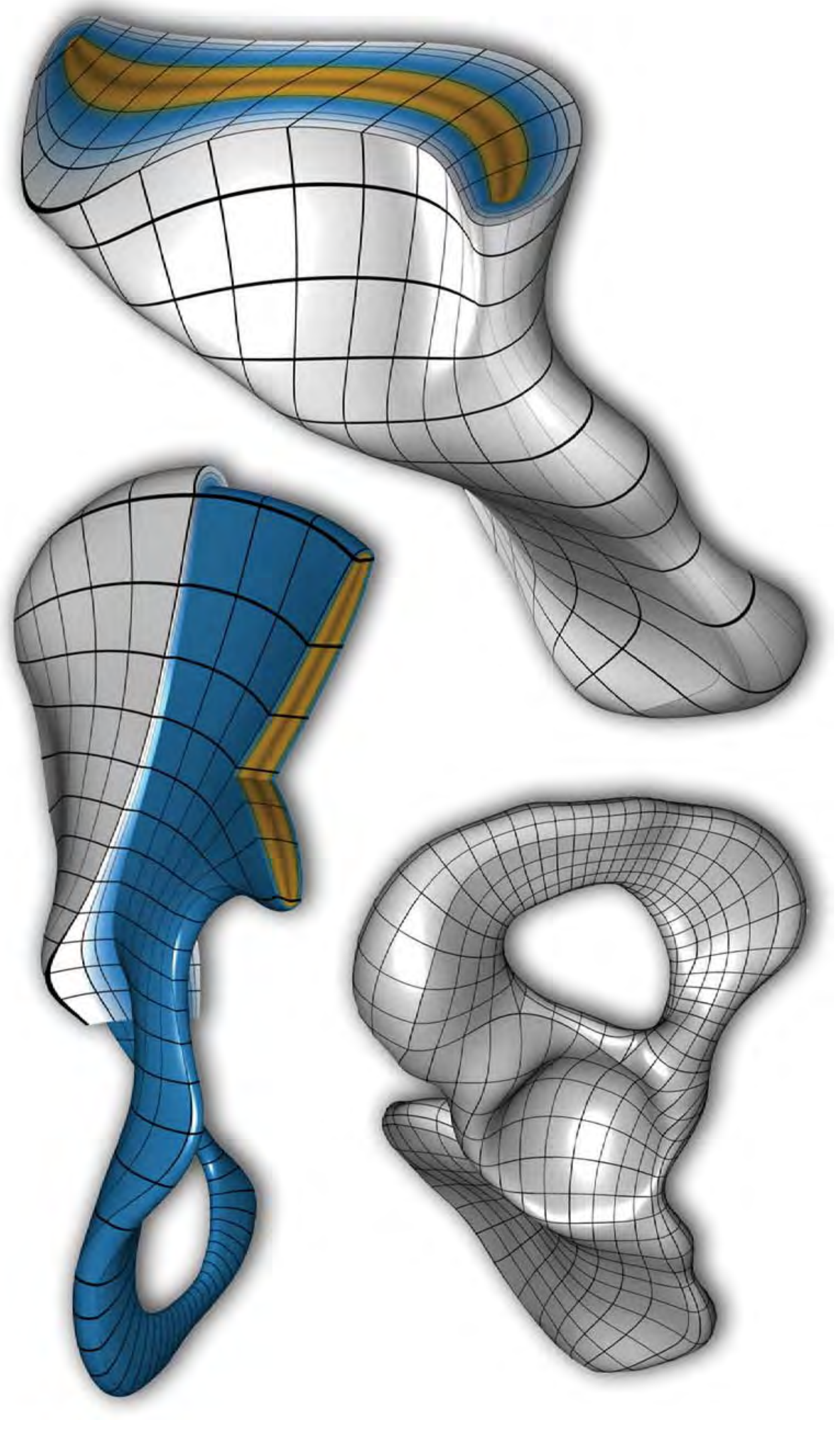


Figure 5.16. Parameterized pelvis.

model or the propeller in Figure 5.3 to guarantee a consistent parameterization resulting in a relative few number of volumetric subpatches. The harmonic nature of the parameterization guarantees that adjacent subvolumes are orthogonal to the scalar field respecting interior material boundaries. While the algorithm requires initial user input to specify corners on the midsurface, the rest of the algorithm proceeds automatically.

This has advantages, for instance, in that the user has control over where corner vertices should be placed, which is often important in simulation where the critical regions on the domain of interest should be free of corners and degeneracies. Also, a good corner selection can result in a more appropriate parameterization where its gradient field follows the geometry more naturally, as was shown on the pelvis data set. However, placing corners can be more challenging for the user on more complex input models. While in the current approach the user gets aid for the corner selection, we are investigating how this initial step could be further automated through a more in-depth analysis of the geometry.

A further generalization should not require interior material boundaries be contained within each other and also the case where the interior material boundaries are unrelated and geometrically more complex than the exterior surface. Lastly, for a more general approach, to avoid distortions in the parameterization, the definition of the midsurface has to be further generalized to include multiple sheets and fins.

CHAPTER 6

MORE GENERAL OBJECTS

In this chapter we present a methodology to create a volumetric representation from a 2-manifold without boundaries represented with untrimmed NURBS surfaces. A trivariate NURBS representation is difficult to construct from such a representation, especially when the input surface patches were created with only a boundary representation as the goal. These kinds of inputs arise in numerous existing geometric modeling scenarios such as models from CAD systems, subdivision surfaces, quadrilateral meshing and data-fitting. Our approach, nearly automatic and only requiring minimal user input, creates a mixed element representation using trivariate NURBS elements at the boundary and unstructured tetrahedral elements in the interior of the object. The original boundary representation of the input model is maintained in the final representation, allowing the volumetric representation to be used in both computer graphics simulations and IA applications. We demonstrate that the mixed element representation yields convergence under model refinement, and that it can be used efficiently in elastic body simulation. For that, we adapt rotational elements previously proposed for linear tetrahedra, to higher-order trivariate NURBS elements.

Objects represented with B-spline and NURBS surfaces are widely used in computer graphics and engineering applications. For instance, a subdivision surface is frequently converted into a set of NURBS surfaces for rendering purposes, or a mechanical part is modeled or reverse-engineered using NURBS patches of higher-order polynomial degrees. If one wants to apply simulation to the volume enclosed by the object, a volumetric representation must first be generated. Generally, the input surfaces are tesselated and a tetrahedral or hexahedral mesh is generated which is then used for simulation. However, due to the tesselation procedure, the higher-order representation is not part of the simulation representation and the tesselation can only approximate the input surface.

To model geometry and simulations simultaneously, IA has been proposed [87]. It applies physical analysis directly to the model representation. This means that the smooth basis functions used to model the smooth geometry are also used as basis for simulation, where good simulation results can be achieved even when the representation contains elements with inferior quality as shown in [40]. This is due to the use of the smooth and higher-order basis. After simulation, the user gets feedback as attributes of the model representation, avoiding the need to generate a finite element mesh and the need to reverse-engineer from the finite element mesh.

However, creating volumetric representations with higher-order polynomial degree and smoothness properties is a difficult problem and is the subject of significant research, especially when it is necessary to maintain the original parameterization. Since the majority of modeling tools are based on surfaces only, tools that create volumetric models rarely exist. Creating such volumetric modeling tools require the definition of new modeling mechanisms, resulting in significant development efforts. Furthermore, users will need to invest time to learn these new volumetric modeling tools and modeling paradigms.

Another way to create a volumetric representation is to create it from NURBS surfaces representing the boundary of the model using existing tools. However, depending on the complexity of the input model, creating a trivariate NURBS representation from it is an unsolved problem. A first approach might offset the boundary representation of the model as a normal moving front into its interior, but then one must deal with the correspondence problem and self-intersecting elements, when the front meets itself. Creating a representation that can be used in simulation, i.e., one that is free of degenerate elements, usually requires manual effort and therefore increases modeling and meshing time significantly. Note that in this case it is desired that the NURBS surface representation is maintained in the volumetric output, making hexahedral meshing strategies such as plastering, whisker-weaving or grid-based methods (see [150]) or more recent methods, e.g., the method discussed in [112] difficult to adopt.

In this chapter we make the following contributions:

- Presentation of a new mixed element modeling approach to create a volumetric representation from an input object represented with NURBS surfaces. By referring to Figure 6.1, the approach, mostly automatic, creates semistructured trivariate NURBS elements at the input surface boundary and fills the remaining interior volume with unstructured linear tetrahedra. The different element types are unified with a collocating approach.
- A convergence study is performed on a mixed element representation to demonstrate its potential in a simulation environment based on the finite element method.
- Finally, an adaptation of the previously proposed warped linear tetrahedra stiffness approach [137, 57] is proposed for trivariate NURBS elements to allow the mixed element representation to be used in interactive simulation environments. A demonstration of deformable body simulation is performed on the mixed element representation.

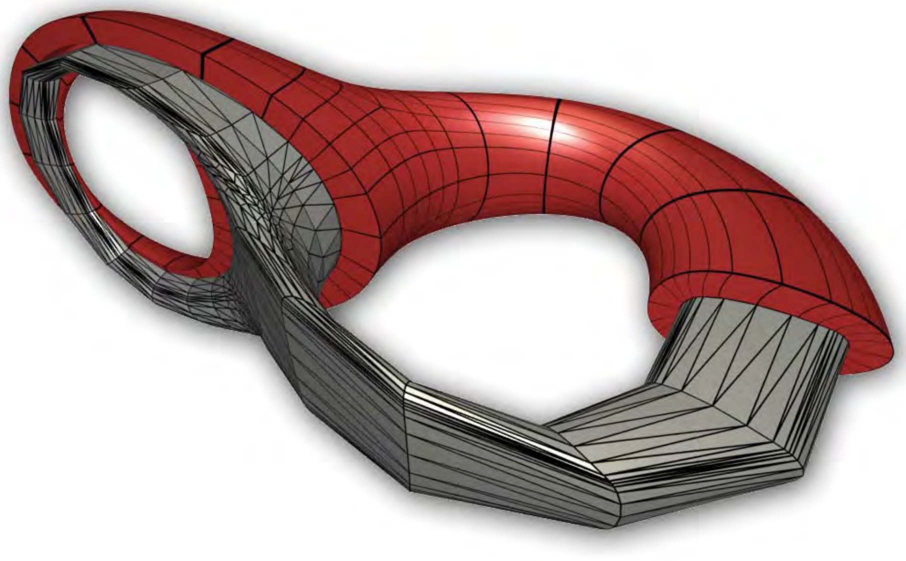


Figure 6.1. Mixed element representation: Tricubic NURBS elements (red) at the boundary. Linear tetrahedra (gray) in the interior of the object. The surface patches are constructed from a Catmull-Clark subdivision object.

The approach is based on harmonic functions [143] and a sampled midstructure such as a point-sampled medial axis of the object. This allows offsetting the input model into its interior without creating degenerate elements to produce a volumetric semi-structured mesh near the boundary of the object. The proposed strategy significantly reduces the time needed to create a volumetric model and avoids the correspondence problem by matching the interior boundary NURBS elements with linear tetrahedra.

NURBS elements require more computational effort during simulation, e.g., numerical integration and evaluation of simulation quantities due to higher-order polynomial degree. These quantities (e.g., mass or material properties) can be efficiently evaluated on linear tetrahedra due to its lower degree. However as discussed above, simulation applied to structured NURBS elements can produce simulation results of higher quality because of smoothness across elements (e.g., see studies in [87, 40]). Figure 6.2 illustrates that the proposed representation can be useful in applications such as animating elasticity, where high quality NURBS elements on the domain's boundary prevent element degeneracies. Linear elements, prone to flipping in nonlinear simulation scenarios, are placed away from the critical regions, and the overall impression of the animation visually appears smoother as well.

The mixed element representation is introduced in Section 6.1. Section 6.3 performs a convergence study in 2D by verifying convergence of the mixed element representation under refinement and also describes the warped stiffness approach adopted to higher-order NURBS elements. The chapter discusses results in Section 6.4 and concludes in Section 6.5.

6.1 Volumetric Representation

This section describes our proposed modeling pipeline. Let the domain of interest be the input $\mathcal{S}$, a 2-manifold without boundaries. $\mathcal{S}$ consists of a collection of coefficients $c_l \in \mathbb{R}^3$ and associated weights $h_l \in \mathbb{R}$ and a set of parametric patches s_k, defined as

$$s_k(u,v) := \frac{\sum_{i=0}^{p} \sum_{j=1}^{q} h_{i,j}\, c_{i,j}\, B_{i,p,\tau_u}(u)\, B_{j,q,\tau_v}(v)}{\sum_{i=0}^{p} \sum_{j=1}^{q} h_{i,j}\, B_{i,p,\tau_u}(u)\, B_{j,q,\tau_v}(v)}, \tag{6.1}$$

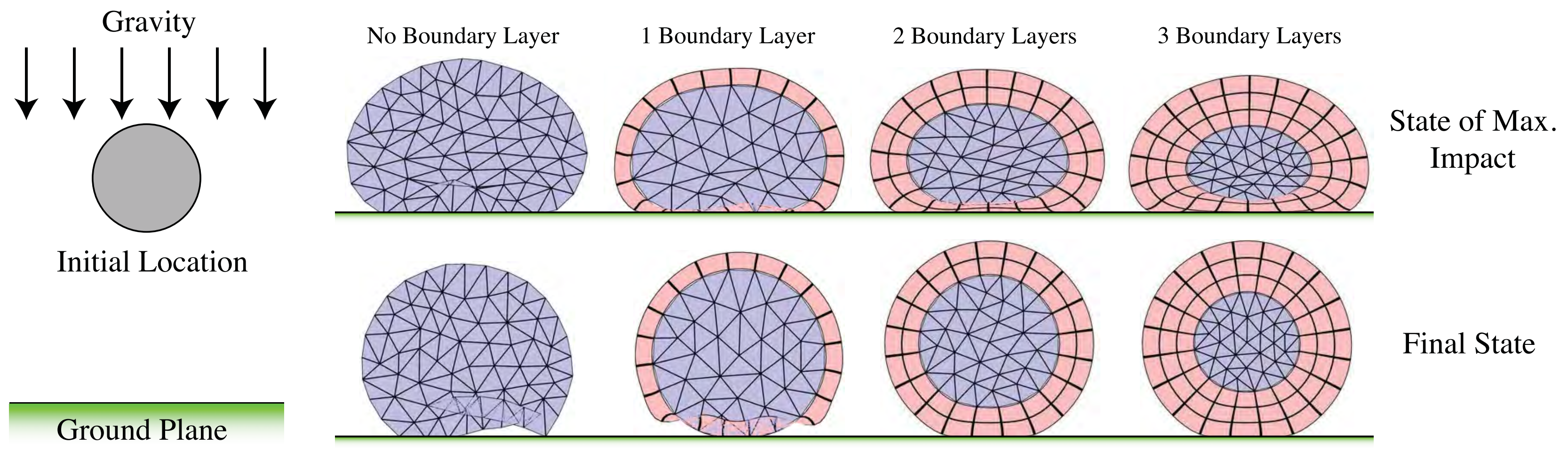


Figure 6.2. Effect of $C^{(2)}$ elements at domain boundary: The configuration of this 2D elasticity example, based on a Hookean material using Chauchy-Green strain, is illustrated on the left. The top row shows the state of maximal impact, i.e., the state where the elastic potential energy reaches its maximum, when the shape hits the ground plane. The bottom row shows the final state of the object. By increasing the number of boundary layers and its thicknesses, the corresponding final states do not exhibit flipped elements. Note, due to the gravitational force the final shape is not circular.

where $B_{i,p,\tau_u}(u)$ and $B_{j,q,\tau_v}(v)$ are B-spline basis functions as defined in [37]. p and q are the degrees and τ_u and τ_v the local knot vectors of patch $s_k(u,v)$. The coefficients $c_{i,j}$ with corresponding scalar weights $h_{i,j}$ define a control mesh of dimension $(p+1)\times(q+1)$. Note that the indices (i,j) in Equation 6.1 are local to the patch $s_k(u,v)$, i.e., there exists a mapping to a global index l.

By referring to Figure 6.3, $\mathcal{S}$ and an associated sampled midstructure (generated by using one of several methods, e.g. [46]) are used to construct a volumetric representation with the following steps:

1: Triangulate input and create unstructured tetrahedral mesh containing the point-sampled midstructure.

2: Construct a harmonic function on the tetrahedral mesh (Figure 6.3b).

3: The harmonic function is used to offset the input surface into the interior, where the user has control to specify the thickness of the resulting semistructured volumetric representation (Figure 6.3c).

4: The limit surface (see green bounding B-spline surfaces in Figure 6.3c) having the same mesh layout as the input is triangulated (see Figure 6.3d) and its interior is filled with unstructured tetrahedral elements (Figure 6.3d).

The general definition of $\mathcal{S}$ and its patches $s_k(u,v)$ allow a wide range of input representations. For instance, if the input is an unstructured quadrilateral mesh, then $s_k(u,v)$ defines a bi-linear patch with $p=q=1$ and $\tau_u=\tau_v=\{0,0,1,1\}$ and $h_l=1$. A more general input may use different weights and a mix of floating and open knot vectors (see [37]), allowing higher continuity among adjacent patches and the representation of rational geometries.

The following sections discuss these pipeline steps in more detail, i.e., Step 1 and 2 are discussed in Section 6.1.1, Step 3 is discussed in Section 6.1.2. Step 4 is discussed in 6.2. The mixed element representation is finally defined in Section 6.2.1.

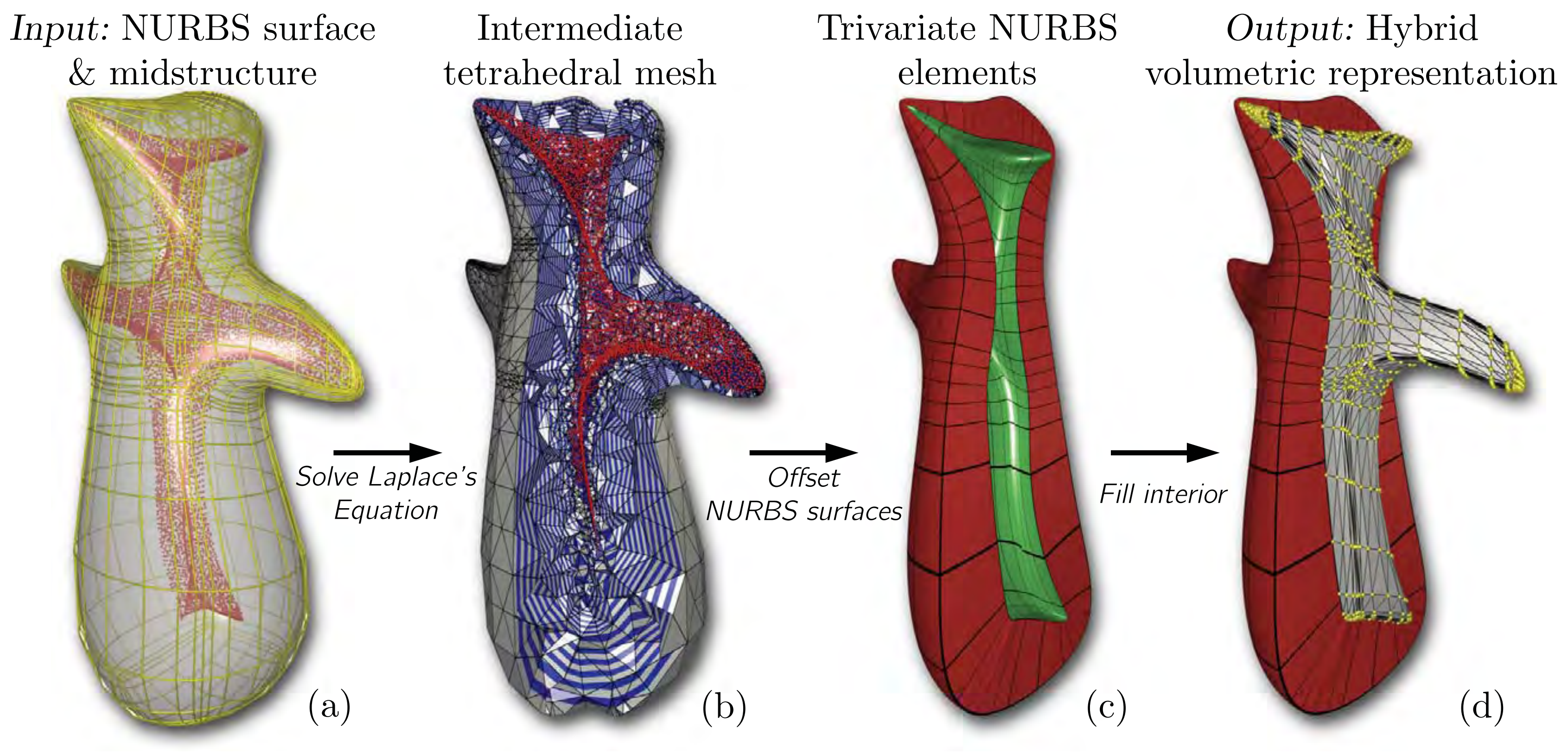

Input: NURBS surface & midstructure
Intermediate tetrahedral mesh
Trivariate NURBS elements
Output: Hybrid volumetric representation
Solve Laplace's Equation
Offset NURBS surfaces
Fill interior
(a)
(b)
(c)
(d)

Figure 6.3. The input are NURBS surface patches, in this case generated from a Catmull-Clark subdivision model. A sampled midstructure such as a simplified medial axis is used to generate trivariate NURBS patches at the boundary of the volumetric representation. The remaining interior is filled up with unstructured tetrahedral elements. (Elements in (b), (c) and (d) are culled to visualize the interior.)

6.1.1 Harmonic Mapping

Given an input surface $\mathcal{S}$, the first step in our framework consists of triangulating the set of coefficients c_l, by triangulating the regular local control mesh $c_{i,j}$ for each patch $s_k(u, v)$ as defined in Equation 6.1. From this triangle mesh and a sampled midstructure of $\mathcal{S}$, an unstructured tetrahedral mesh $\mathcal{H}$ is constructed (e.g., by using [186]).

Let $\mathcal{V}$ be the set of vertices in $\mathcal{H}$. Let $\mathcal{V}_E$ be the set of vertices on the boundary of $\mathcal{H}$ and $\mathcal{V}_I$ be the set of vertices defining the midstructure of $\mathcal{S}$ lying in the interior of Note, $\mathcal{V}_E$ and $\mathcal{V}_I$ are subsets of $\mathcal{V}$ and also that $\mathcal{V}_E$ may contains vertices in addition to the coefficients c_l, depending on quality control of the tetrahedral mesh generator.

Given $\mathcal{H}$, $\mathcal{V}_E$ and $\mathcal{V}_I$, a harmonic function on $\mathcal{H}$ is constructed. A harmonic function is a scalar function $w \in C^2(\Omega), w : \mathcal{H} \to \mathbb{R}$, satisfying Laplace's Equation 1.1.

w satisfies the maximum principle, i.e., it does not exhibit any local minima and maxima and has been shown useful in the domain of meshing and volumetric parameterization. See for instance [50, 126, 196, 76].

The finite element method [86] is used to discretize Equation 1.1. $\mathcal{V}_e$ and $\mathcal{V}_I$ are the sets of vertices on which the solution is known (Dirichlet boundary), where in this context, the vertices in $\mathcal{V}_E$ are set to 0 and the vertices in $\mathcal{V}_I$ are set to 1. With this setup $w(x, y, z)$ evaluates to 0 at the boundary of $\mathcal{H}$ and to 1 at vertices defining the midstructure (see Figure 6.3b).

A solution has the form

$$w(x, y, z) = \sum_{\mathbf{v}_k \in \mathcal{V}} \hat{w}_k \, \phi_k(x, y, z), \tag{6.2}$$

where $\phi_k(x, y, z)$ are linear hat functions [86] associated with the respective vertex in $\mathcal{H}$. Galerkin's formulation [86] is used to set up a linear system which is solved to assign a harmonic value with every vertex in $\mathcal{V}$. The gradient field ∇w over $\mathcal{H}$ is piece-wise constant, flowing towards the sampled midstructure.

6.1.2 Offset Input Surface

The harmonic function $w(x, y, z)$ is used to offset the vertices c_l defining $\mathcal{S}$ into the interior of $\mathcal{S}$. In a first step, the user is given the ability to choose a parameter value

$w_0 \in [0,1]$, allowing the user to control the volume enclosed by $\mathcal{S}$ and the isosurface $w(x,y,z) = w_0$ (see Figure 6.3c). Once this choice has been made, the remaining pipeline stages proceed automatically. Note that since the midstructure representation is sampled, the isosurface at w_0 could be of higher genus than the input $\mathcal{S}$, especially when w_0 is close to 1. The proposed method requires that the isosurface at w_0 and the input have the same genus, i.e., the user must choose a w_0 that satisfies this condition.

For every c_l defining the input surface $\mathcal{S}$ a path $g_l(\omega)$, $g_l : \mathbb{R} \to \mathbb{R}^3$ is constructed emanating from c_l by following ∇w and terminating at point c_l', where $w(c_l') = w_0$, i.e., $g_l(0) = c_l$ and $g_l(w_0) = c_l'$. Note that $g_l(\omega)$ is parameterized through the harmonic field $w(x,y,z)$. Let $\mathcal{S}'$ be the surface defined with the coefficients c_l' (see green surface in Figure 6.3c), with corresponding surface patches $s_k'(u,v)$.

Given the paths $g_l(\omega)$, surface patches $s_k(u,v)$ can be swept into the interior of $\mathcal{S}$. Assume there exists a sorted set $W = \{\omega_0 = 0, \ldots, \omega_n = w_0\}$, consisting of n scalars, where $\omega_i \in [0, w_0]$. Given W, for each $s_k(u,v)$ a volumetric NURBS patch $v_k(u,v,w)$, defined as

$$v_l(u,v,w) := \sum_{i,j,k=0}^{p,q,n} w_{i,j,k}\, g_{i,j}(\omega_k)\, \mathcal{B}_{i,j,k}(u,v,w), \tag{6.3}$$

can be constructed, where

$$\mathcal{B}_{i,j,k} := B_{i,p,\tau_u}(u)\, B_{j,q,\tau_v}(v)\, B_{k,r,\tau}(w), \tag{6.4}$$

and where $v_l(u,v,w)$ defines a triple sum. r defines the degree in w and τ is the knot vector in the domain $[0,1]$. The weights $w_{i,j,k}$ in the interior of $\mathcal{V}$ are set to 1. W is determined through an optimization procedure trying to determine the values ω_i such that the volumetric elements in $v_l(u,v,w)$ have equal volume.

Due to the maximum principle of w, paths $g_l(\omega)$ are consistent and therefore self-intersecting elements, as they often arise by offsetting a surface along its normal field, generally do not occur. Furthermore, due to the properties of Laplace's Equation 1.1, $g_l(\omega)$ and the resulting $v_l(u,v,w)$ are orthogonal to the input surface $\mathcal{S}$.

Let $\mathcal{V}$ be the collection of volumetric patches $v_l(u, v, w)$. Note that the outer boundary of $\mathcal{V}$ and the input surface $\mathcal{S}$ are the same. The inner boundary of $\mathcal{V}$, i.e., $\mathcal{S}'$, (see red surface in Figure 6.4) is geometrically similar to the isosurface $w(x, y, z) = w_0$.

6.2 Tetrahedralize the Interior

For the following discussion let $\mathcal{I}$ be a collection of tuples $(c'_l, t_l, s'_k(u, v))$, where $t_l = (u^*_l, v^*_l)$ is the node location [37] corresponding to c'_l, and $s'_k(u, v)$ is the surface patch whose parametric domain contains t_l. Given such a tuple, let $x_l = s'_k(t_l)$. x_l is equivalent to a first-order projection of c'_l onto $s_k(u, v)$ [37] and can be computed efficiently. Figure 6.5 illustrates this setup for a single NURBS surface and its corresponding triangle mesh.

The final step in our proposed framework consists of filling up the interior of $\mathcal{S}'$ with unstructured tetrahedra. Similarly as in the first step of our framework (Section 6.1.1), $\mathcal{S}'$ is triangulated. However, instead of the coefficients c'_l, the locations x_l are used for

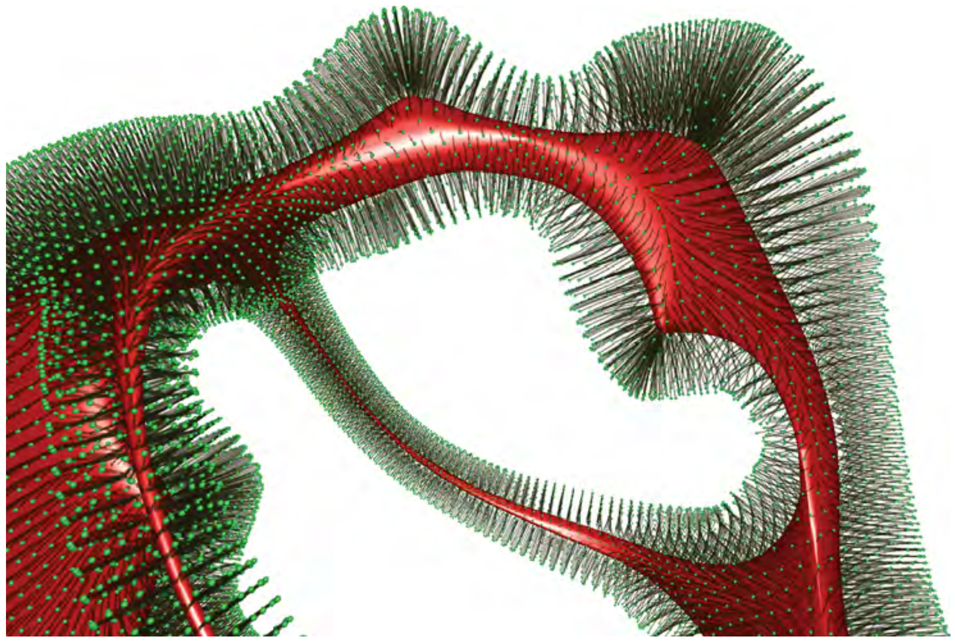


Figure 6.4. Paths following ∇w (gray) emanate at c_l (green points) and terminate on isosurface $w(x, y, z) = w_0$ which is approximated with $\mathcal{S}'$ (red). The choice of the midstructure allows creation of elements in thinner tubular regions.

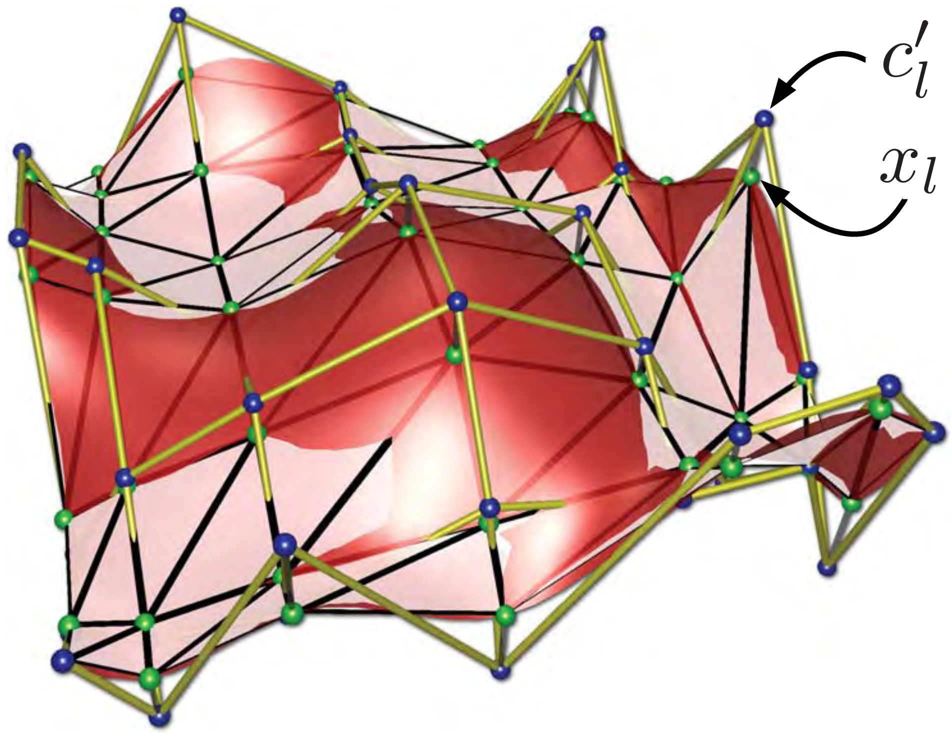


Figure 6.5. Illustration of $C^{(2)}$ NURBS surface and corresponding $C^{(0)}$ triangle mesh. c_l' (blue) for the NURBS surface and x_l (green) for the triangle mesh boundary, represent the same degree of freedom. Due to geometric concavities, the representations are overlapping.

the triangulation (see Figure 6.5). The interior of the resulting triangle mesh is filled up with unstructured tetrahedra to construct the tetrahedral mesh $\mathcal{T}$ (Figure 6.3d).

6.2.1 Mixed Element Representation

After completion of these framework steps, the proposed mixed element volumetric representation is defined by the tuple

$$\mathcal{H} = (\mathcal{V}, \mathcal{T}, \mathcal{I}). \tag{6.5}$$

Even though c_l' and its corresponding x_l in $\mathcal{I}$ are at different locations (Figure 6.6), in our simulation framework discussed below, they represent the same degree of freedom. $\mathcal{I}$ is the interface between $\mathcal{V}$ and $\mathcal{T}$.

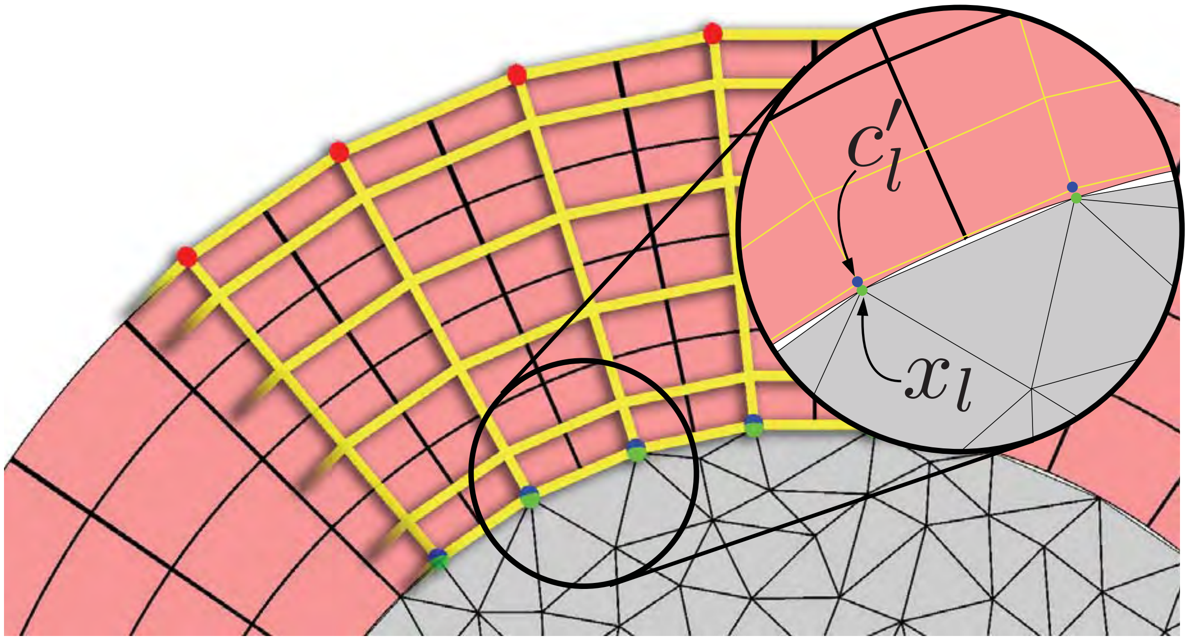


Figure 6.6. Mixed element representation in 2D. Quadratic NURBS elements at the boundary and linear triangles in the interior. Quadratic NURBS elements were chosen at the boundary to demonstrate the worst-case scenario of the alignment with triangles in the interior. As discussed in the text, convergence is achieved even in this more general scenario. Using cubic NURBS elements, the triangles are more closely aligned with the B-spline element boundaries (see Figure 6.2).

The connection of two different representations of interfaces, in our case $\mathcal{V}$ and $\mathcal{T}$, has been examined in the context of mechanics modeling, in the areas of contact and fluid structure interaction. For instance, the reader is referred to [162, 58, 75, 206, 27, 102]. Our approach is a collocation approach, in contrast to a mortar approach, where $\mathcal{V}$ and $\mathcal{T}$ are forced to match at specified collocation points, i.e., the points in $\mathcal{I}$. Given this setup, it is clear that the two interfaces will not match because of the difference in representations (see Figure 6.5 and 6.6). The interface will have gaps or overlaps depending on whether the relationship is concave or convex. In our case, we are interested in volumetric representations, so although overlapping volumetric representations are not the desired goal, we can be confident that the two interfaces (and hence the volumetric representations) converge to a common representation. This is due to B-spline properties [37], where under successive refinement, the control mesh of $\mathcal{S}'$ converges to $\mathcal{S}'$ and hence the gaps and overlaps between the boundary of $\mathcal{T}$ and

$\mathcal{S}'$ get smaller.

This hybrid choice has been made for two reasons. While continuity could be more easily achieved in the 2D case, the proposed method avoids patching higher-order tetrahedral elements to quadrilateral surface patches in 3D. This choice also simplifies model subdivision and generation of $\mathcal{T}$. Secondly, in the subsequent sections it will be demonstrated that this representation behaves stably in the simulation environment: Convergence on a 2D problem is achieved (Section 6.3.1) and dynamic physically-based animation is efficiently applied to the 3D case (Section 6.3.2).

Given a static or dynamic problem, discretized using the finite element method [86], in the general case, there exists a solution function $\alpha : \Omega \rightarrow \mathbb{R}^d$. For instance in linear elasticity $d = 3$ where α describes a displacement field defined over Ω, or in heat conduction where α represents a temperature scalar field defined over Ω, i.e., $d = 1$. In our framework Ω is represented with $\mathcal{H}$.

Given the discontinuous nature of $\mathcal{H}$, once α has been computed, $\alpha(s'_k(t_l)) \neq \alpha(x_l)$ each of a given tuple $(c'_l, t_l, s'_k(u,v)) \in \mathcal{I}$. $\alpha(x_l)$ is the evaluation of α at tetrahedra that have x_l as one of its vertices. Therefore, before simulation proceeds, $\hat{\alpha}_l = \alpha(s'_k(t_l))$ is computed, where $\hat{\alpha}_l$ is the coefficient of α corresponding to the tetrahedron where one of its vertices is x_l. This enforces that $\alpha(s'_k(t_l)) = \alpha(x_l)$. In the subsequent sections we demonstrate that with the proposed collocation-based approximation, convergence can be achieved and efficient physically based animation can be performed to Ω.

6.2.2 Robustness and Practical Considerations

$\mathcal{S}'$ tends to have self-intersections if the input surface contains highly stretched elements, e.g., see Figure 6.1. Therefore, before the mixed element representation generation framework is executed, the input surface is appropriately refined, such that the elements have similar size and shape. While the element count of the input surface is increasing, the resulting triangle mesh corresponding to $\mathcal{S}'$ has a better quality and is more suitable to generate a unstructured tetrahedral mesh.

Furthermore, while paths computed from a harmonic field are guaranteed to be free of intersections, paths traced on a discretized field can overlap. The resulting higher-order elements comprising the thick shell can therefore have self-intersections which should be avoided. In a similar fashion to the method proposed in Chapter 9, paths are traced simulataneously in small steps, where in each step the front defined by the path endpoints are smoothed using a Laplacian Smoothing scheme such that the endpoints remain on the corresponding isosurface.

6.3 Simulation Studies

In this section we examine the proposed mixed element representation as discussed in Section 6.2.1 in two simulation scenarios. In the first study, we show that the solution computed on such a representation converges under refinement using finite elements. In the second study, we demonstrate that deformable body simulation based on finite elements can be efficiently applied to it.

6.3.1 Convergence Study in 2D

Here, we examine a study in 2D to show that our proposed mixed element representation as discussed in Section 6.2.1 produces comparable simulation results with respect to convergence rates under h-refinement [86] than the equivalent representation which uses only triangles. Note that h-refinement increases the degrees of freedom at each refinement step.

Assume we are given the following smooth analytical function $g : \Omega \to \mathbb{R}$,

$$g(x, y) := J(4, J_0(4, 2)\, r(x, y)) \sin(4\, \theta(x, y)), \tag{6.6}$$

where $r(x, y) := \sqrt{x^2 + y^2}$, and $\theta(x, y)$ defines the angle between vector (x, y) and the Cartesian coordinate axes, i.e., $r(x, y)$ and $\theta(x, y)$ convert (x, y) into polar coordinates. Ω in this study represents a disk centered at the origin with a radius of 1 with boundary $\partial\Omega$. Furthermore, $J(n, z)$ is the nth Bessel function of the first kind at $z \in \mathbb{R}$, and $J_0(n, m)$ is the mth zero of the nth Bessel function of the first kind. Since $g(x, y) = 0$ at $\partial\Omega$, the Dirichlet boundary condition at $\partial\Omega$ is set to zero. In the following experiment, let $f(x, y) := \nabla^2 g$.

In this study we investigate Poisson's equation $-\nabla^2 \widetilde{g} = f$, solved using Galerkin's method on two disk representations: (1) the disk is represented with NURBS elements at the boundary and triangles in the interior using the mixed element framework as discussed in Section 6.2.1; (2) the disk is is represented with triangles only. Both representations are refined three times where the error $|g(x, y) - \widetilde{g}(x, y)|_2$ is computed for each refinement step. The corresponding log-log diagram is shown in Figure 6.7. The figure also shows the disk representations at an intermediate refinement step and the corresponding solution along the z-axis of the respective disk.

The convergence of a triangle representation to the true solution is h^{p+1}, where p is the order on element and h its radius. For instance, quadratic NURBS elements therefore have a cubic convergence rate [16], while linear tetrahedral elements converge quadratically. However, as the experiment in this section shows, under h-refinement, the slope for both representations is approximately -2 indicating quadratic convergence even due to the presence of higher-order NURBS elements. Our mixed element representation has two inherent approximation errors: (1) Geometric approximation error of the interior representation and (2) approximation error of the field. Both geometric subdivision and linear triangle elements have a quadratic convergence rate. Therefore, the overall convergence rate is limited to be quadratic. However, convergence rates are not as crucial in applications such as computer graphics, where stability and high quality simulation results are of greater importance. Such a scenario is examined in the following.

6.3.2 Warped Stiffness

The previous section examined the numerical performance of the proposed mixed element framework in the 2D case on a static problem. In this section we show that the mixed element representation is also useful in the context of deformable body simulation in 3D, where the equation of motion

$$M\,\ddot{\mathbf{x}} + C\,\dot{\mathbf{x}} + K\,(\mathbf{x} - \mathbf{x_0}) = f_{ext} \tag{6.7}$$

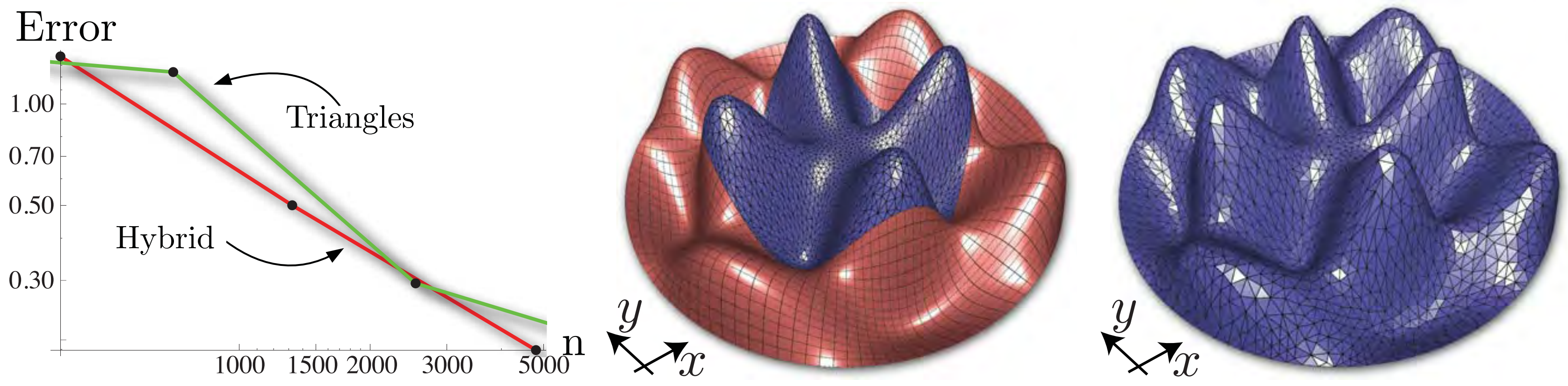


Figure 6.7. Convergence behavior of hybrid representation. Left: convergence plot comparing the mixed element representation with the representation which only uses triangles. For the error computation, we chose the l_2 norm between the approximated and true solution. n is the degree of freedom for the refinement steps. Solutions along z-axis on mixed element representation (middle) and triangle only representation (right), during one h-refinement step.

is solved on the domain of interest. The coordinate vector $\mathbf{x}$ is a function of time, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the first and second derivatives of $\mathbf{x}$ in respect to time. x_0 is the rest state (state of equilibrium) at $t = 0$, M is the mass matrix, C the damping matrix, and f_{ext} is the external force (see [86]). K is the stiffness matrix representing the second derivative of Cauchy's linear strain (see [141]), which is often chosen in practice, because K must be computed only once at the beginning of the simulation, resulting in efficient time-step computations and a larger time-step choice due to implicit integration. The choice of a linear model leads to even more computational speedups with NURBS elements, i.e., the local stiffness matrix K_e for a NURBS element is much larger in comparison to that for a tetrahedral element due to the higher number of degrees of freedom and its higher polynomial degree, which also requires numerical integration to compute the entries in K_e.

The linear strain model, however, is only valid close to the equilibrium configuration and therefore is not suitable for large rotational deformations, due to the lack of the nonlinear terms in the linear strain model. The warped stiffness approach proposed by [57, 137] is an efficient and robust method to avoid these distortions. In this approach, K and the resulting force vectors acting on the object (Equation 6.7) are updated every time step in the rotated coordinate frame for each element. This is equivalent to applying the rotation of an element at the current time-step to its stiffness matrix at $t = 0$, i.e., by referring to [137], at the current time-step, the local stiffness matrix for a given element is $K'_e = R_e\, K_e\, R_e^{-1}$. Furthermore, $f = K'\,\mathbf{x} + f'_0$, where K' is the global stiffness matrix constructed from the local stiffness matrices K'_e. f'_0 is constructed from $f'_{0e} = R_e\, f_{0e}$, where f_{0e} is the element's force vector acting at the element at $t = 0$.

Given a tetrahedron, the transformation from its rest state to its current state can be described with an affine mapping, i.e., the rotational part of the transformation can be determined uniquely. This is not the case for a nonlinear transformation of a NURBS element, i.e., the rotation of a point in the rest state to its corresponding point in the current state may vary across the element. For a given NURBS element, we approximate its rotation R_e by extracting the rotational part of the Jacobian $J = J_p \cdot J_m^{-1}$ evaluated at the center of the element. J_m is the Jacobian of the parametric mapping of the element in the rest state, J_p is the Jacobian of the parametric mapping of the element at its state at the current time-step.

Our experiments indicate that this approximation is sufficient and produces plausible deformations, i.e., false object scalings do not visibly occur.

6.4 Results

We applied the proposed approach to the following closed input objects, represented with the bicubic patches: B-spline surfaces computed from Catmull-Clark subdivision surfaces shown in Figure 6.1 and Figure 6.3; Bézier surfaces with $G^{(1)}$ continuity computed from quadrilateral meshes computed using the approach by [165] shown in Figure 6.8(a), (b), (c); NURBS surfaces approximating a mechanical part with $C^{(2)}$ continuity in the interior and $C^{(0)}$ at adjacent surface patches (Figure 6.8(d)). In all cases, a cubic knot vector is used for the w-direction. In the current framework, the midstructure has been computed using the tight co-cone approach [46], where for some models, the resulting medial axis has been manually simplified. For instance, for the fandisk model (Figure 6.8(d)), sheets corresponding to sharp features were manually removed. Although manual medial axis simplification is a time-consuming procedure, our approach only requires a sampling of a midstructure and does not require proper topological connectivity, so the simplification was performed rapidly and did not exceed ten minutes for each of the input models. We applied laplacian smoothing and edge collapses to produce triangles of similar size and area. The corresponding vertex set was used as sampling for the intermediate tetrahedral mesh (Figure 6.3(b)).

The only remaining user input to the subsequent pipeline stages was the choice of thickness of the outer B-spline volumetric region as discussed in Section 6.1.1. The reason for this is that a user might want to specify a different thickness depending on the application. For instance, a thicker NURBS volume would result in more high-order trivariate patches with wider basis function support, resulting in slightly slower simulation, but higher simulation quality as illustrated in Figure 6.2 in a 2D example. However, for prototyping purposes, a user might choose thinner volumes to compute simulation results more quickly.

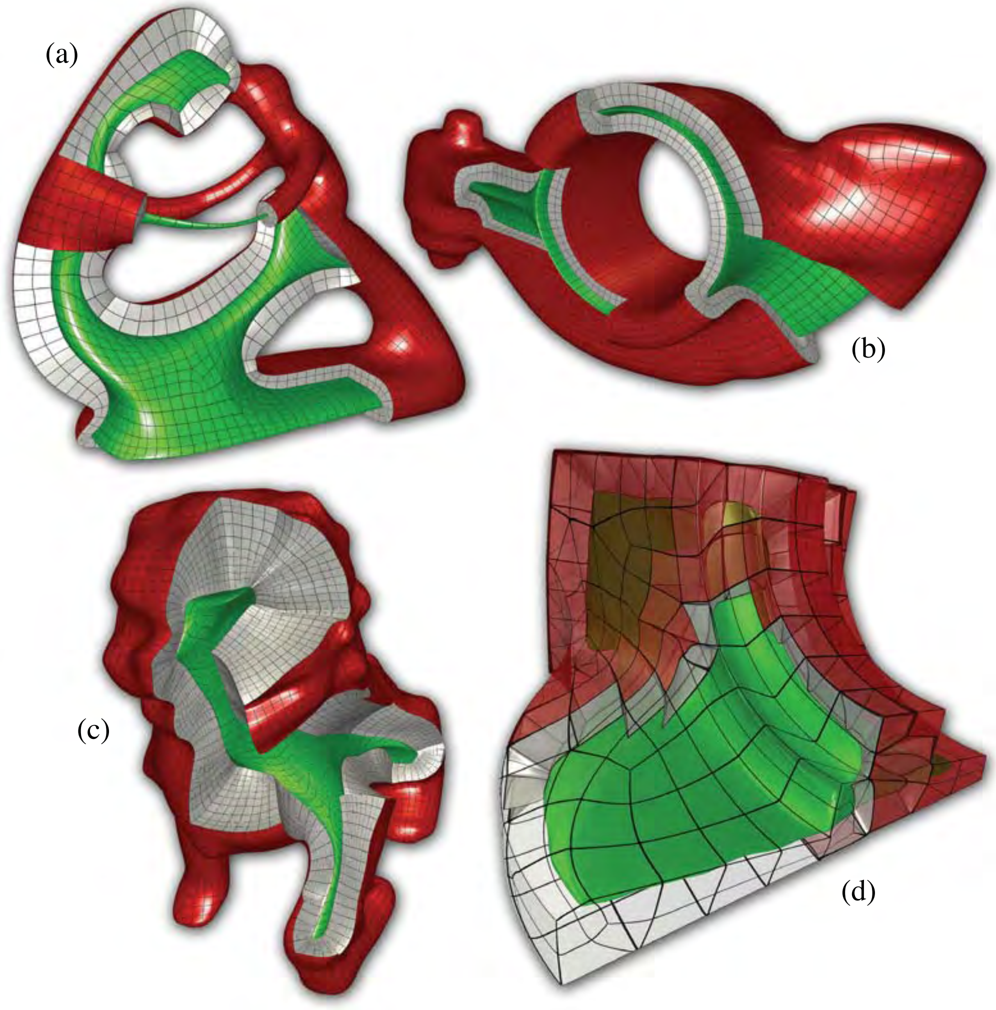


Figure 6.8. Methodology applied to four objects. Elements are culled to better visualize the filled interior of the objects. Object boundaries and interior boundaries are shown in red and green, respectively. The input for (a), (b) and (c) are bicubic Bézier patches. The input for (d) are bicubic NURBS surfaces. The wire-frame shows $C^{(0)}$ continuities.

The remaining computational step consisting of constructing Laplace's matrix, tracing paths and creating volumetric elements, can be neglected as these operations can be parallelized and computed efficiently. Timings were less than five minutes for each of the test models. As discussed in Section 6.1.2, after the user performed all required input, the sampling procedure in [126] is adapted for our context that resamples the paths to create roughly equal B-spline volumes by avoiding degenerate elements. Note that this procedure only re-samples the paths $g_l(\omega)$ computed from the input points and ∇w as discussed in Section 6.1.2 and does not require any other recomputation.

6.5 Conclusion

In this chapter we proposed a framework to create a volumetric representation from an input surface represented with untrimmed NURBS surfaces. The approach is based on harmonic functions which are used to offset the input surface until a user-specified offset distance is reached. The remaining interior volume is filled with unstructured tetrahedra, and a collocation approach is proposed to address the correspondence problem. The boundary surface of the volumetric representation and the input surface are the same, i.e., smoothness and geometry of the input surface are maintained in the volumetric representation. The use of harmonic functions in this context allows a flexible choice of an appropriate midstructure: A simplified medial axis was used for the examples in this chapter, but other representations such as one-dimensional curve skeletons and midsurfaces can be specified as well. Hence, this methodology generalizes approaches such as the methodologies presented in Chapter 4 and Chapter 5. If desired, the major part of the volume can be filled up with high-order volumetric elements.

We have shown that the method can be applied to NURBS surfaces from various inputs, which means that creating the volumetric representation is independent from creating the input surfaces. This is in contrast to volumetric parameterization methods. Therefore, the proposed pipeline can be added to existing quadrilateral mesh, surface modeling, and data-fitting pipelines. We plan to extend the current framework to allow further input representations such as T-splines [178] which allow modeling more complex objects.

A limitation of the approach is the geometric discontinuity between the high-order boundary in the interior and the boundary of the interior tetrahedral mesh, where the convergence rate of the tetrahedral mesh is the limiting factor. While we have demonstrated that the resulting mixed element volumetric representation is stable, can be used in simulation scenarios and is motivated by discontinuous mixed element simulation representations from other areas, in future work we plan to extend our frame-work to work to a wider range of element types, for instance higher-order tetrahedral elements. Note that a fully continuous representation is more difficult to achieve, mainly because of the smooth input surface representations which can have an arbitrary choice of continuity and smoothness properties. Therefore, we also plan to investigate how the current setup performs in comparison to a discontinuous Galerkin approach, such as a mortar method [162], to further evaluate the performance of our proposed simulation framework.

In conclusion, we see our approach as especially useful in applications which require that the input surface parameterization has to be maintained in the volumetric representation, such as in IA. But the approach is also useful in applications where the time to generate volumetric inputs is very limited, requiring only little user input, which is generally desired and even required in various computer graphics applications.

CHAPTER 7

GENERALIZED SWEPT MIDSTRUCTURE

In this chapter we introduce a novel midstructure called the generalized swept midstructure (GSM) of a closed polygonal shape, and a framework to compute it. The GSM contains both curve and surface elements and has consistent sheet-by-sheet topology, versus triangle-by-triangle topology produced by other midstructure methods. To obtain this structure, a harmonic function, defined on the volume that is enclosed by the surface, is used to decompose the volume into a set of slices. A technique for computing the 1D midstructures of these slices is introduced. The midstructures of adjacent slices are then iteratively matched through a boundary similarity computation and triangulated to form the GSM. This structure respects the topology of the input surface model is a hybrid midstructure representation. The construction and topology of the GSM allows for local and global simplification, used in further applications such as parameterization, volumetric mesh generation and medical applications.

7.1 Introduction

Many applications in the field of computer graphics and visualization require interior midstructures of three-dimensional objects that represent their form or shape with lower dimensional entities. One-dimensional curve skeletons [11], and the 3D medial axis [187], are such examples that have been used for mesh generation, animation, registration, and segmentation applications. Curve skeletons faithfully represent an object in tubular regions. For more general geometry, a medial axis is preferred since it consists of surface sheets [187] and they better capture the shape than curve skeletons.

However, a medial axis is very sensitive to small changes in shape and it produces nearly degenerate polygons in tubular regions. Our quest for a new type of midstructure is motivated by three reasons. (1) In some situations, it is desirable for a user to design a skeleton. Since a user might be aware of which regions require higher fidelity elements and a reduced number of extraordinary points in later simulations, the user can design a midstructure to yield the appropriate mesh layout. This is similar to the motivation of the approach presented in [126] and [124] and is difficult to achieve with medial axis computation algorithms. (2) The topological structure and connectivity information of midstructure components is often required for the aforementioned applications. However, existing techniques for medial axis computation of polygonal models (e.g., tight co-cone [46], powercrust [8], discrete scale axis (DSA) [131]) do not classify surface sheets of the medial axis as illustrated in the magnified view in Figure 7.1. Given a polygonal

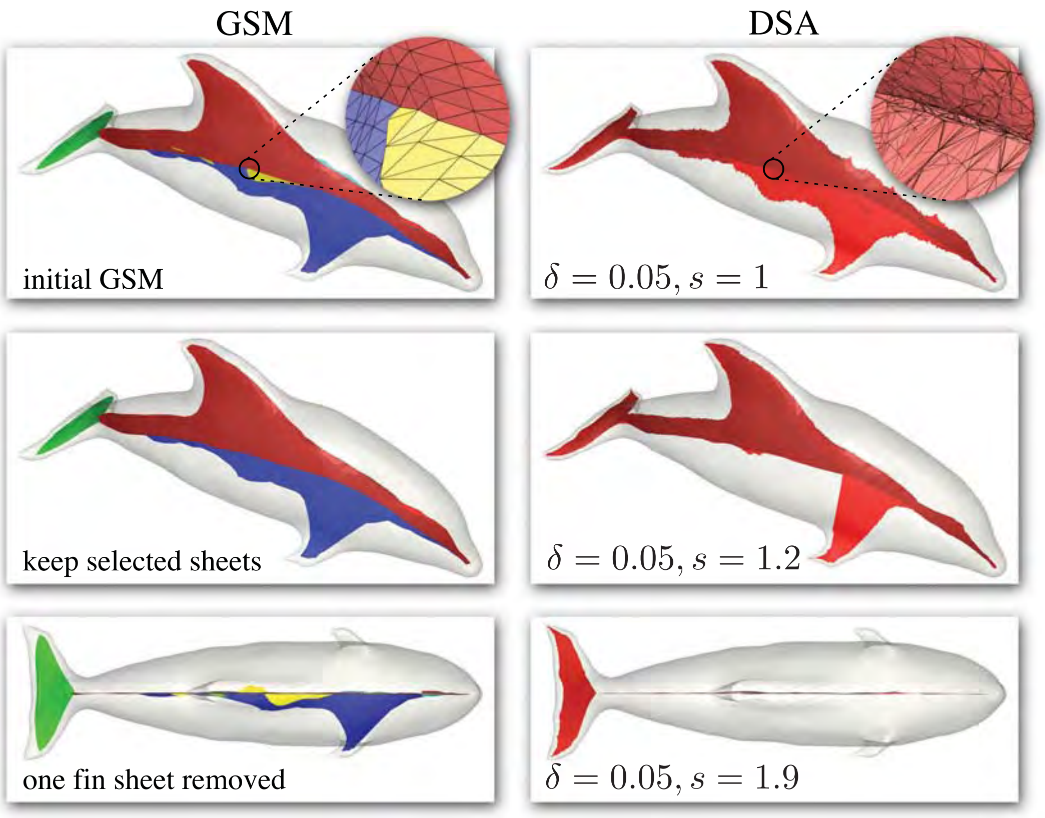


Figure 7.1. Sheet based simplification using GSM (left) vs. global simplification using DSA (right). Magnified regions compare triangulations in a region where surfaces meet. Only the GSM shows 3 distinct sheets (red, blue, yellow) while the DSA shows similar polygons. The yellow GSM sheet (top left) is removed (mid left GSM). To simplify DSA in this region requires removal of additional structures in other regions (mid right DSA). Bottom row illustrates removal of a large blue sheet on one side of the GSM, whereas simplification of the DSA results in removal of polygons from both sides.

medial axis representation, identifying surface sheets requires postprocessing that often includes topological fixing and establishing connectivity information. This is a tedious process as it requires extensive human input. (3) Neither curve skeletons nor 3D medial axes have an embedded parameterization (see Figure 7.2). Such a parameterization can be useful for 3D cross field design [144, 85].

This chapter introduces the *Generalized Swept Midstructure* (GSM) (Figure 7.3) with the three properties, discussed in the following.

7.1.1 First Property

The GSM is a hybrid midstructure, consisting of both curve and surface elements, that is suitable for an input shape consisting of both general and tubular regions. Examples are presented in Section 7.7.

7.1.2 Second Property

Analogously to [79] for curve skeletons or [143] in computing fair morse functions to extract the topological structure of a surface mesh, the user determines a sweeping

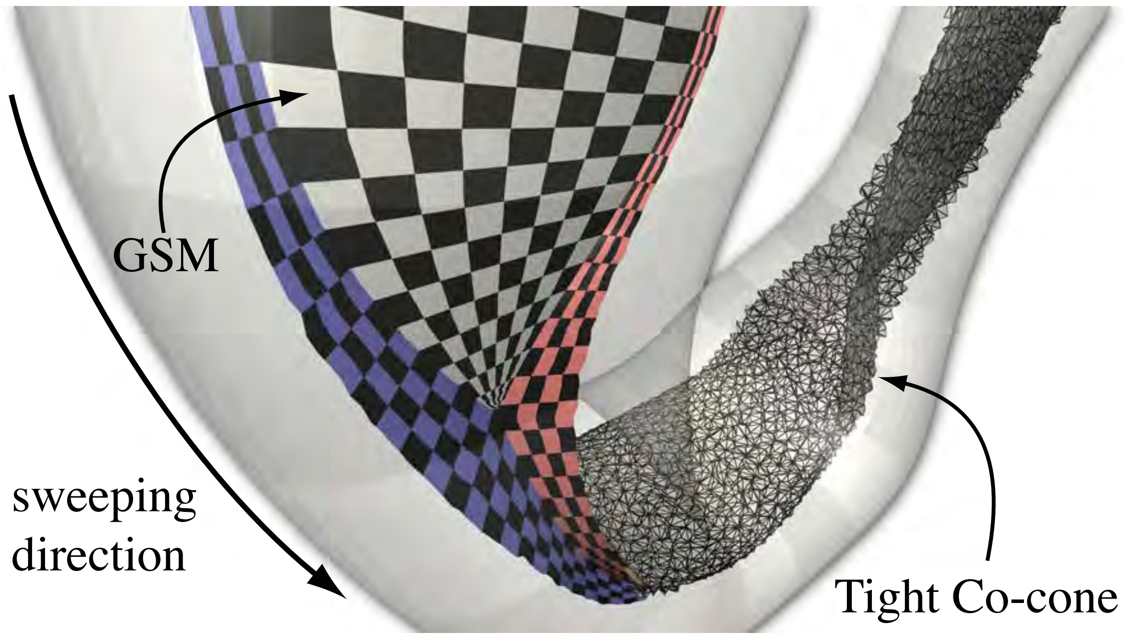


Figure 7.2. The tight co-cone is intrinsic to the object, consisting of accurate but complex medial topology. A GSM identifies sheets (blue, red and white) and because of its swept generation, it has a natural parameterization.

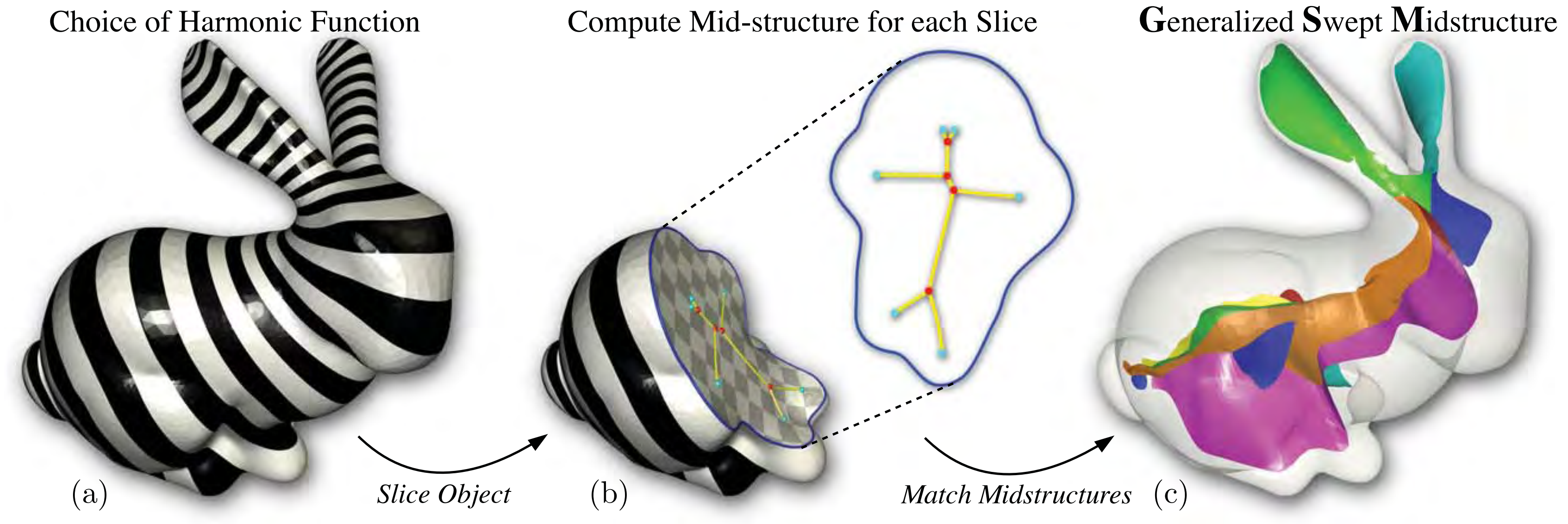

Choice of Harmonic Function
Compute Mid-structure for each Slice
Generalized Swept Midstructure
(a)
Slice Object
(b)
Match Midstructures
(c)

Figure 7.3. Pipeline for the Generalized Swept Midstructure (GSM): (a) User choice of harmonic function. (b) Object is decomposed into (curved) slices, and a medial axis is computed for each slice. (c) Slices are iteratively matched into a GSM (colored surface sheets).

strategy by choosing a harmonic function that conforms naturally to an object's shape. Apart from this specification, the construction of the GSM is automatic. From this harmonic function a 3D harmonic function is computed. In this work, midstructures (i.e., simplified medial axes) of level sets of the 3D harmonic function are swept across the input object (Figure 7.3 (a) and (b)) to generate a GSM. The level sets are typically nonplanar regions. This generality reduces the number of connected components and therefore enables the approach to be used for a wide variety of objects. A theoretical framework for a swept medial axis was proposed in [42]. The goal of that work was to present theoretical results on geometric properties of the object in terms of medial axes of the planar slices. In our work, we present practical algorithms for an implementation of a generalization of that framework and demonstrate several results.

7.1.3 Third Property

The swept nature of a GSM suggests a parameterization strategy. Namely, one parameter is assigned as the harmonic function value and the second parameter is obtained by assigning parameter values to the midstructure of each slice. Figure 7.2 shows an example. Furthermore, the GSM construction approach automatically classifies the various sheets of the midstructure, computed by tracking transitions of curve segments of the midstructures of the level sets. Given the sheet-by-sheet topology of the GSM (see colored sheets in Figure 7.3 c), the user can select sheets of the GSM that are to be preserved or removed in a simplification procedure (see Figure 7.1). A simplified midstructure can be used in subsequent applications, such as generation of 3D cross fields or volumetric parameterization.

The contributions of this chapter include:

- Introduction of the GSM, a novel midstructure based on sweeps of midstructures of nonplanar level sets, and a pipeline to compute it (Section 7.3).
- A novel planar medial axis computation algorithm, from which the midstructure is computed (Section 7.5).

- A matching algorithm for consecutive midstructures to create a GSM with consistent topology (Section 7.6).

7.2 Background

There has been vast research on midstructures related to the proposed GSM. 1D curve skeletons and 3D medial axes are special types of midstructures. Algorithms for computing them are reviewed in the surveys [187, 19].

In a similar fashion to the GSM construction, level set diagrams [110] are constructed by connecting barycenters of isocontours of a scalar function defined on a surface. Curve skeletons are extracted by improving Reeb graphs of harmonic functions in [79]. Mesh contraction using constrained Laplacian smoothing is used to construct curve skeletons in [11].

Exact arithmetic is used to compute medial axes of polyhedra in [41]. Approximations of the medial axis of polygonal meshes are computed using distance fields in [62] and Voronoi diagrams in [190]. Algorithms for computing medial axes from point-sampled surfaces based on Voronoi graphs are [8, 46, 30]. The discrete scale axis [131] is a variant of the medial axis that computes connected polygons of medial surfaces corresponding to dominant shape features at a user specified simplification scale.

Voronoi based medial axis computation algorithms are computationally efficient. However, since there is no sheet topology information, there is no explicit relationship between medial axis regions and object shape features. Furthermore, in contrast to a GSM, global methods do not suggest a strategy to parameterize the resulting medial axis, at least in part because sheet structure is undetermined. With an explicit sweeping direction, the GSM identifies sheets, and has a natural parameterization that can be used for later applications as discussed in Section 7.8. Figure 7.2 shows a comparison between the tight co-cone and GSM.

A hybrid structure is derived via topological analysis of the 3D medial axis of an object in [68] and is used to annotate tubular and more general regions of the object. However, the derived structure is susceptible to problems associated with medial axis computation. Thinning algorithms such as those presented in [93] are used to derive skeletons of objects represented as subregions of volumetric grid data. The derived skeletons consist of discrete voxels without topology. The topology must be inferred in a post process, and is susceptible to errors stemming from sampling density and object orientation within the grid. Our proposed approach automatically generates curve and surface sheet skeletons in appropriate areas with consistent topology at the transition regions and sheet topology in surface regions.

7.3 The GSM

The generalized swept midstructure (GSM) is a midstructure obtained by joining midstructures of nonplanar slices of a polygonal representation of a closed 3D object. The GSM, a connected structure lying in the interior of the object, is a generalization of the swept medial axis as proposed in [42]. The GSM consists of triangulated surfaces and curves represented as polylines. The GSM is invariant under rigid body transformations and scaling.

7.3.1 Computational Pipeline Overview

This section provides an overview of our methodology to construct a midstructure for a closed surface triangle mesh. Let $(\mathcal{T}, \mathcal{V}_T, \mathcal{C}_T)$ define the bounding triangle mesh, where $\mathcal{T}$ is the set of triangles, $\mathcal{V}_T$ is the set of vertices, and $\mathcal{C}_T$ is the connectivity of the mesh. Based on $\mathcal{T}$, a volumetric representation $\Omega \subset \mathbb{R}^3$ is constructed, represented as an unstructured tetrahedral mesh, denoted by $(\mathcal{H}, \mathcal{V}_H, \mathcal{C}_H)$, where $\mathcal{H} \subset \mathbb{R}^3$ is the set of tetrahedra, $\mathcal{V}_H$ the set of vertices defining the tetrahedra, and $\mathcal{C}_H$ the connectivity of the tetrahedral mesh. $\mathcal{H}$ is constructed using a tetrahedral meshing method, e.g., [186], and has $\mathcal{T}$ as its boundary.

The following steps describe the construction of GSM from the input shape $\mathcal{T}$, as shown in Figure 7.3:

1: Compute a harmonic function $u(x, y, z)$ on $\mathcal{H}$ (Section 7.4).

2: Decompose $\mathcal{H}$ into a sequence of nonplanar slices L_i (L_i are level sets of $u(x, y, z)$) (Section 7.4.1).

3: Extract a simplified 2D midstructure for each L_i (Section 7.5).

4: Starting from the first slice, iteratively construct the midstructure by matching the midstructures of two adjacent slices until the last slice is reached (Section 7.6).

7.4 Harmonic Functions

The GSM framework can be used if the dataset already contains a slicing strategy (e.g., segmented data from a volumetric scan). In this case we proceed to step 2 in the GSM construction pipeline. Otherwise, we compute a harmonic function.

A harmonic function is a function $u \in C^2(\mathcal{H}), u : \mathcal{H} \to \mathbb{R}$, satisfying Laplace's Equation 1.1.

Galerkin's formulation [86] is used to discretize Equation 1.1. $\mathcal{V}_H$ can be decomposed into the set $\mathcal{V}_B$ for which the solution is known (Dirichlet boundary) and the set $\mathcal{V}_I$, for which a solution is sought. A solution has the form

$$u(x, y, z) = \sum_{\mathbf{v} \in \mathcal{V}} \hat{u}_k \, \phi_k(x, y, z), \tag{7.1}$$

where $\phi_k(x, y, z)$ are linear hat functions associated with vertex $v_k \in \mathcal{V}$. The gradient field ∇u over $\mathcal{H}$ is therefore piecewise constant.

In our framework, many methods can be used to create the slicing strategy, such as [49], but we chose approaches similar to [143, 50, 79], where the user determines the points in the set $\mathcal{V}_B$. The user therefore has control over $u(x, y, z)$ and the resulting sweeping strategy. Figure 7.4 illustrates two harmonic functions on the genus-1 kitten model. While $u(x, y, z)$ in Figure 7.4a has two saddles, $u(x, y, z)$ in Figure 7.4b follows a torus-like sweep. Both are valid. Figure 7.4 also shows the corresponding GSMs for these two distinct choices.

7.4.1 Decomposition of $\mathcal{H}$

Given the harmonic function $u(x, y, z)$, a slice L_i (Figure 7.3b), at value $u_i \in \mathbb{R}$ is the level set satisfying $u(x, y, z) = u_i$. L_i is extracted using marching tetrahedra [33].

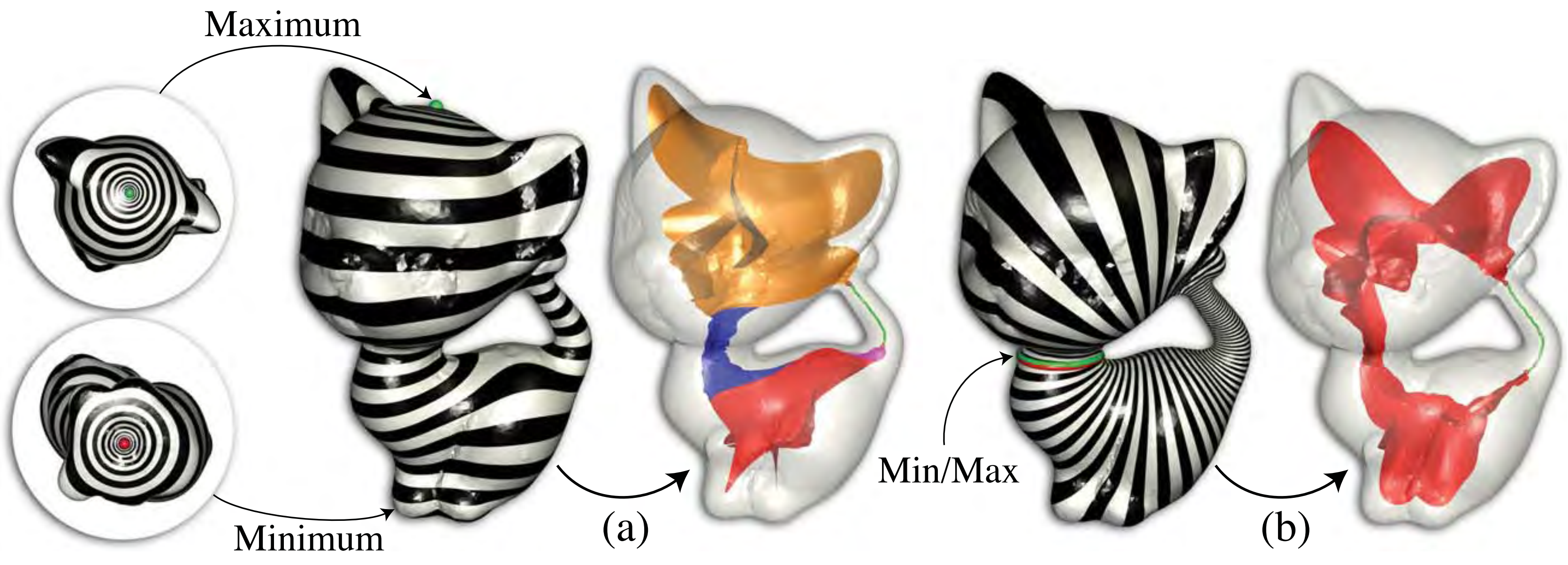


Figure 7.4. Two different harmonic functions on kitten model result in different GSMs.

Depending on the choice of $\mathcal{V}_B$ and resulting saddle points [143], L_i can consist of multiple disjoint nonplanar 2-manifolds represented with triangle meshes with boundaries.

Once the user specifies the harmonic function $u(x, y, z)$, which determines the cutting strategy, the object is decomposed into a set of slices L_i such that every triangle in $\mathcal{T}$ is intersected by at least one slice which captures the global features in $\mathcal{T}$. For each vertex p_i^k of L_i, a path can be constructed from p_i^k to a new point p_j^k, the *projection* of p_i^k on level set L_j, by following $\nabla u(x, y, z)$. Let $l_{i,j}^k$ be the length of this path.

Then, given this set of slices, let $\epsilon_i = \max_{\forall k}\{l_{i,i+1}^k\}$ be the distance between slice i and $i+1$. Due to distortions of $u(x, y, z)$ and triangulation of $\mathcal{T}$, ϵ_i is not constant across the slices. To achieve a cutting of the object such that ϵ_i varies slowly, the input surface can be remeshed into a triangle mesh whose triangles have approximately the same size and shape. These parameters can be chosen by the user to maintain a specific feature size. Such a triangulation can be computed using, for instance, Afront[173] which creates a triangulation with smaller triangles in regions with higher curvature and larger triangles in regions with lower curvature. Section 7.7 presents an example that shows GSMs of different versions of an input object.

Each component of L_i is flattened using the CGAL [1] implementation of the LSCM [113]. The boundary of the flattened L_i is approximated with a periodic B-spline curve using the method proposed in [126]. A medial axis with topological structure is computed for the planar region enclosed by this curve, using a novel technique presented in Section 7.5. This medial axis is simplified, yielding the midstructure which is mapped onto the respective component of L_i and incrementally matched with that of an adjacent slice to construct the GSM (Section 7.6).

7.5 Computing Midstructure of Slices

In order to construct reliable GSMs, we require midstructures to consist of smooth curves and smoothly changing geometry and topology between adjacent slices. Several techniques for computing the medial axis of a planar region from piecewise smooth [163, 3] or discrete boundary representations [187, 19] exist. However, such approaches introduce artifacts, due to the nature of the representation, and human interaction is required to remove them to compute a suitable midstructure. In a dissertation by Musuvathy [138] a method is presented to automatically and accurately compute the medial axis with topology of the parametric B-spline curve that approximates the boundary of the flattened level set L_i. A suitable midstructure is then computed by simplifying the medial axis based on its topology.

The medial axis of a planar region enclosed by a bounding curve γ is the locus of centers of maximally inscribed circles that are tangent to two points on γ, with the limit points of the locus [66]. The contact points of each maximal circle with the boundary curve are called *foot points* for the corresponding medial axis point. A limit point is either an *end point* or a *junction point* at which the maximally inscribed circle has one or three foot points, respectively. Three medial curve segments meet at a junction point. Figure 7.5 shows an example of the medial axis of a planar region computed using the method presented in a dissertation by Musuvathy [138]. This method also computes foot points and distance to the boundary for each medial axis point, thereby giving the complete medial axis transform.

7.5.1 Medial Axis Simplification

The midstructure is constructed by simplifying the computed medial axis. Leaf segments are deleted and internal segments are merged. Medial segments incident at distance critical points are merged. These operations are performed when the respective segment length is smaller than the slicing distance (ϵ_i). Note that this procedure may result in more than three incident curves at a junction point, but does not add complexity to the matching algorithm presented in the next section.

If all segments of the medial axis are smaller than ϵ_i, the medial axis is contracted to the centroid of the region. This situation occurs when the boundary is nearly circular and therefore the medial axis consists of small segments near the center of the region. These contractions result in 1D curve segments in the GSM. We will denote a topological graph of the midstructure of the level set L_i as $M_i = G(N_i, E_i)$, where N_i is the set of the end points and junction points, and E_i is the set of edges that connect these points. These edges correspond to the curved segments in the original midstructures that are densely sampled for later GSM representation. p_i^j is used to denote the j^{th} node in N_i,

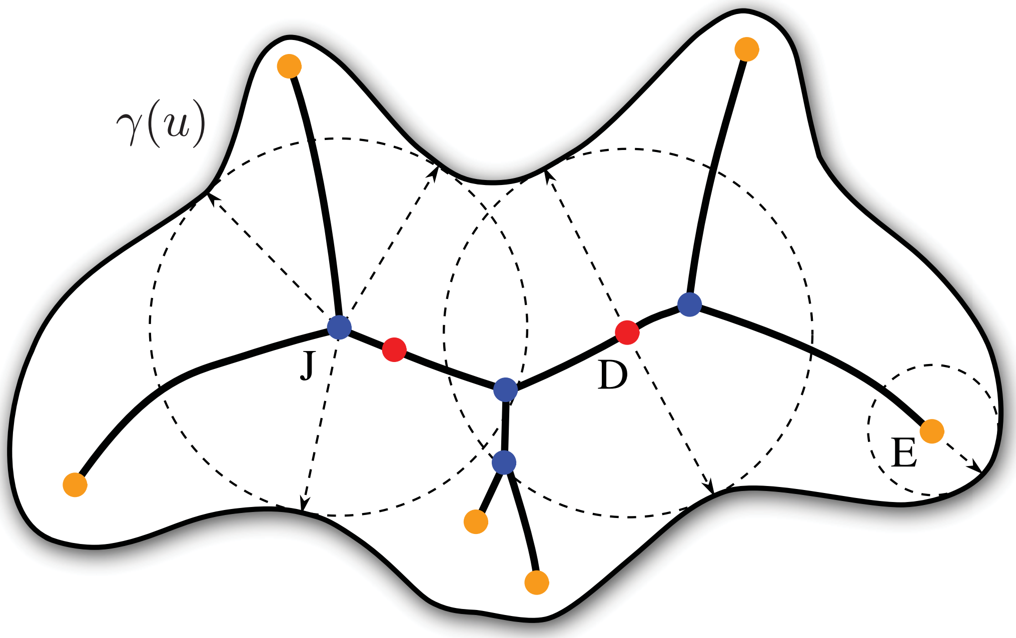


Figure 7.5. Medial axis of a region bounded by a B-spline curve. Along with their maximal circles, end points (E) are shown in orange, junction points (J) in blue, and distance critical points (D) in red. The arrows point to the foot points.

and $(p_i^j, p_i^k) \in E_i$ represents the edge between those nodes.

7.6 Matching Successive 2D Midstructure

After computing midstructures for each level set (Section 7.5), given the two successive midstructures represented as two graphs (Section 7.5), we match the edges of the graphs. A triangulation is used to connect these matching pairs based on the samples along the original midstructure (see Figure 7.6). A number of existing graph matching techniques can be applied to accomplish this step [65, 188, 103]. However, these methods typically deal with more general graph matching problems without knowing the relation between the two graphs that are matched. Thus, their algorithms are usually complicated and computationally expensive. In contrast, in the present problem, one graph is evolved from the other through a small change and hence the generic transitions between the two graphs are well-defined [66]. Therefore, a simpler matching technique can be devised by finding the correspondences along the section boundary curves from

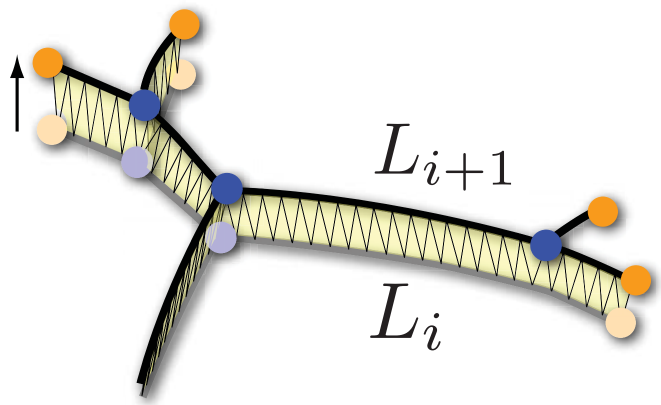


Figure 7.6. A triangulation is used to connect these matching pairs based on the samples along the original midstructures.

which the midstructures are computed (Section 7.5).

7.6.1 Topological Changes

For smoothly changing geometry of the boundary, there are only two generic transitions of the midstructures [66]: **Leaf creation/annihilation (***Type 1***)** and **Flip configuration (***Type 2***)**. *Type 1* corresponds to the creation (or destruction) of a feature (e.g., a protrusion) on the boundary. To illustrate *Type 2*, consider the junction points, p_i^j and p_i^k in M_i and p_{i+1}^l and p_{i+1}^r in M_{i+1}, respectively. Each pair is connected by an edge. In the continuous case, edge (p_i^j, p_i^k) will first collapse into a single node before growing to edge (p_{i+1}^l, p_{i+1}^r). However, the discrete cutting will likely not capture the degenerate point as shown in Figure 7.7.

During matching, we assume at most one topoligical change on an edge (including its two end points) of the graph when evolving from one level set to the next. If this assumption is not satisfied, additional level sets between the original pair must be added until it is satisfied. In the event this cannot be attained, other cases will be investigated in a future work.

7.6.2 Matching

Let $\{cp_i\}$ be the set of foot points of M_i. We match two graphs M_i and M_{i+1} according to the distance of $\{cp_i\}$ and $\{cp_{i+1}\}$ on ∂L_{i+1}. We first project $\{cp_i\}$ (on

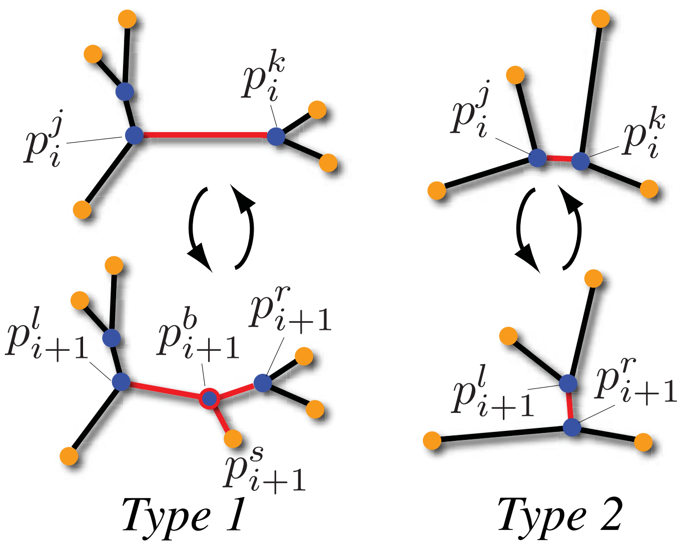


Figure 7.7. The discrete cutting will likely not capture the degenerate point.

∂L_i) to ∂L_{i+1} as discussed in Section 7.4.1. On the boundary of the level set L_{i+1}, the distance between any two foot points cp and cp' is defined as the shortest arc-length between them along ∂L_{i+1}, $(\widehat{cp, cp'})$. Assume that all foot points are sorted along ∂L_{i+1} (either clockwise or counter clockwise). Given a foot point cp_i^j of M_i, there are exactly two points cp_{i+1}^l and cp_{i+1}^r from M_{i+1} that enclose cp_i^j along the 1D boundary ∂L_{i+1}. Therefore, finding the closest point to a given foot point can be done in constant time. Two end points p_i^j, p_{i+1}^r (Figure 7.8, bottom left) from the two graphs are called *close* if their foot points are the closest pair on ∂L_{i+1}. We then pair them in the matching, denoted as $p_i^j \leftrightarrow p_{i+1}^r$. Two junction points p_i^g, p_{i+1}^b (Figure 7.8, middle left) from the two graphs are called *close* if the foot points of p_i^g are directly next to the ones of p_{i+1}^b pairwisely on ∂L_{i+1} or their leaf nodes are all close to each other.

Given the above distance and similarity metric, our matching algorithm can be described as follows (Figure 7.8). **1.** Match two closest end points. **2.** Match two closest junction points. **3.** Match two edges if their end points are matched pairwise. **4.** Handle topological changes and match remaining edges. Handling topological changes proceeds as follows. 1) for *Type 1*, a junction point p_{i+1}^{l+1} is introduced (or removed) if a new branch edge $(p_{i+1}^{l+1}, p_{i+1}^s)$ is growing out from (or collapsing onto) an existing edge (p_i^k, p_i^{k+1}) that is split to two edges $(p_{i+1}^l, p_{i+1}^{l+1})$ and $(p_{i+1}^{l+1}, p_{i+1}^{l+2})$. We then match

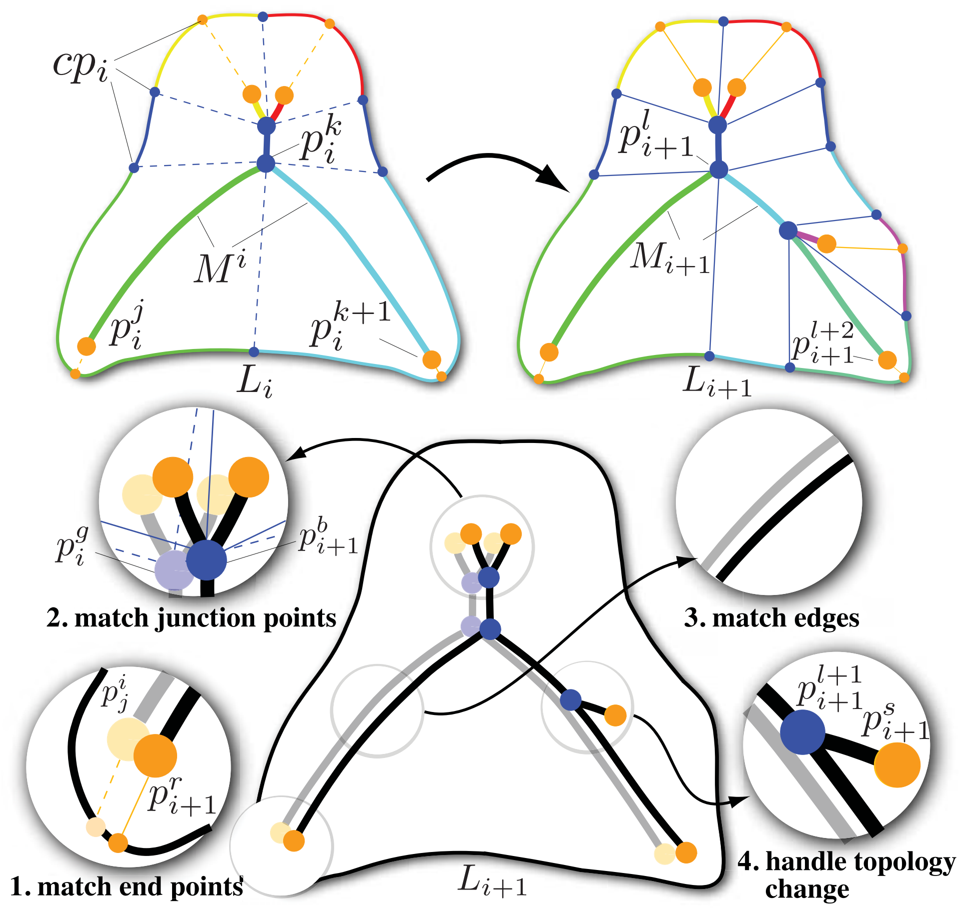


Figure 7.8. Illustration of matching algorithm. The top row shows two consecutive slices, L_i and L_{i+1} and their midstructures, M_i and M_{i+1}. The foot points of the end points (orange dots) and junction points (blue dots) are highlighted on the boundaries. Each point in the midstructure and its foot points are linked through straight lines. The bottom figures illustrates the matching steps 1–4. Note that all the foot points in level L_i have been projected to level L_{i+1}. For illustration purpose, we overlap the midstructure M_i (skeleton with light colors) with M_{i+1} (skeleton with dark colors).

$(p_i^k, p_i^{k+1}) \leftrightarrow \big((p_{i+1}^l, p_{i+1}^{l+1}), (p_{i+1}^{l+1}, p_{i+1}^{l+2})\big)$. Note that if the new branch edge is growing from an existing junction point, we do nothing. 2) for *Type 2*, there are two unmatched junction points for each graph, e.g., p_i^j and p_i^{j+1} at M_i, p_{i+1}^r and p_{i+1}^{r+1} at M_{i+1}. They are connected by an edge in their corresponding graph. In the meantime, all their connecting end points are matched pairwisely (see the four end points in the illustrative example of *Type 2* above). It is this configuration that allows us to identify *Type 2* topological change. To handle that, we insert four matching pairs: $p_i^j \leftrightarrow p_{i+1}^r$, $p_i^j \leftrightarrow p_{i+1}^{r+1}$,

$p_i^{j+1} \leftrightarrow p_{i+1}^r$, and $p_i^{j+1} \leftrightarrow p_{i+1}^{r+1}$. Note that if a skeleton graph contains only a single node, everything in the successive graph will be mapped to this node. This guarantees the continuous transition between 1D curve and 2D surface structures of a GSM.

7.6.3 Handling Bifurcations

The aforementioned matching framework works well for level sets with one connected component. It is not sufficient for the case where the number of connected components of the level sets changes at the saddle points of the harmonic function (e.g., the splitting and merging of level sets), for instance, at the basis of the ears of the bunny. We extend the framework to handle the matching at bifurcations as follows.

Let C_i and C_{i+1} be the number of connected components in L_i and L_{i+1}, respectively. Assume $C_i < C_{i+1}$ (i.e., splitting). We project cp_i^j onto ∂L_{i+1} (If $C_i > C_{i+1}$ (i.e., merging), we project cq_{i+1}^l onto ∂L_i). Each projected foot point $cp_{i+1}^{j'}$ is assigned a component index after projection. All the foot points of one node p_i^j are in the same component after projection because of the properties of the harmonic function. We extract subgraphs from M_i based on the assigned component indices. These subgraphs are matched with the corresponding components of M_{i+1} using the same algorithm described in Section 7.6.2.

By referring to Figure 7.9, let e_i represent the edge $(p_i^j, p_i^{j+1}) \in E_i$ of M_i. It splits

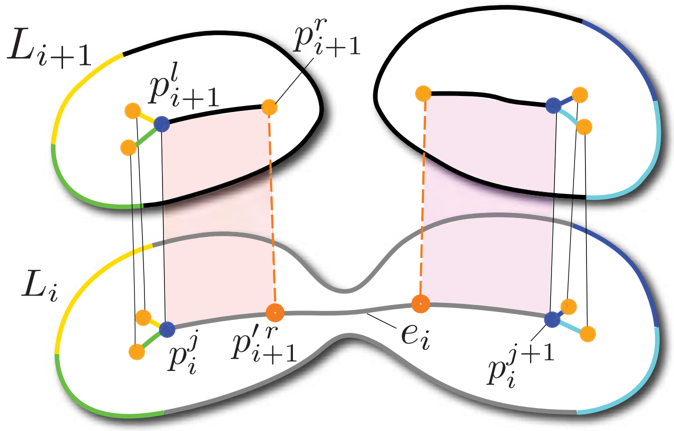


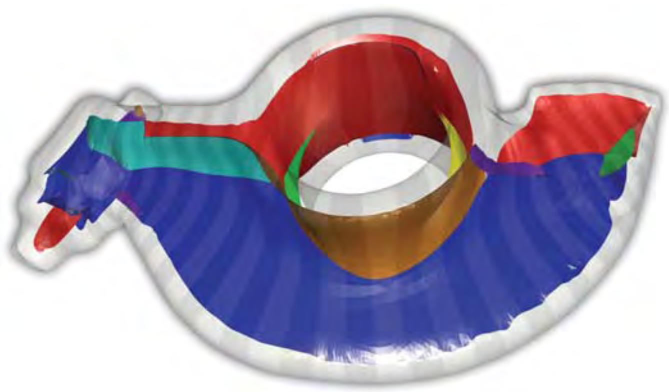
Figure 7.9. Handling of GSM bifurcations.

into two edges in M_{i+1}. Assume that p_i^j is matched to p_{i+1}^l. We examine the edges adjacent to p_{i+1}^l in M_{i+1}, and find out the one whose other end node p_{i+1}^r has not been matched and has the smallest Euclidean distance to e_i. We then project p_{i+1}^r onto e_i at $p_i^{r'}$ and construct a partial matching between $(p_i^j, p_i^{r'})$ and (p_{i+1}^l, p_{i+1}^r). We process p_i^{j+1} similarly.

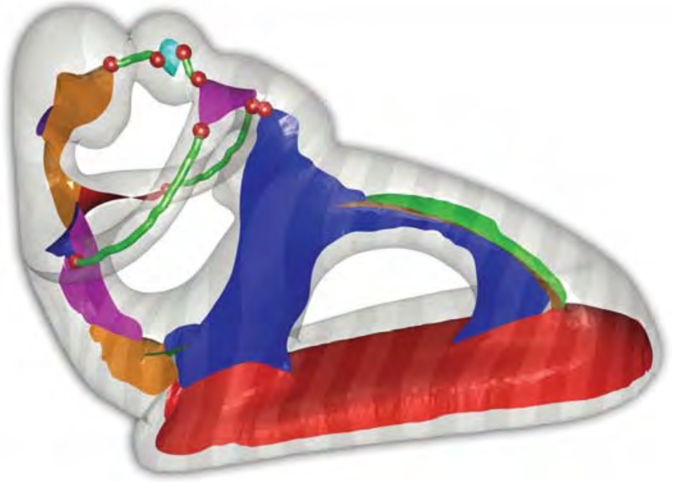
7.7 Results and Discussion

Figure 7.10 shows results of our framework for a number of graphics, medical, and CAD models. The iterative construction of the GSM allows us to track topological changes of the midstructures of level sets along the user desired cutting orientation. Different color sheets in Figure 7.10 represent the evolution of their individual feature components of the midstructure (the edges of the simplified medial graph). Figure 7.11 presents a comparison with 1D curve skeletons [11] and discrete scale axes [131] for a model with tubular and more general regions (see the middle and right columns). Both the curve skeletons and discrete scale axes are computed using the programs provided by the authors of those papers. This comparison shows that the hybrid structure of the GSM captures the tubular and more general regions of each object as curve and surface elements, while the other two approaches contain only either of the two. The topology of the GSM enables smoothing of sheet boundaries.

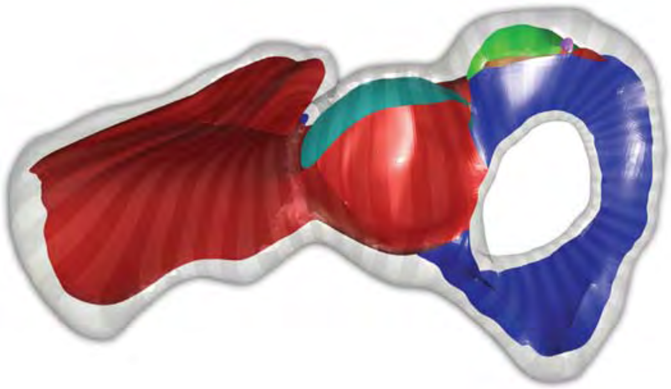
The user interaction to compute the harmonic function for the models presented in this chapter did not exceed 5 minutes. The remaining pipeline steps to compute the resulting GSM proceeded automatically. To extract one slice and compute its corresponding midstructure takes about 1 minute in our current implementation. Since this computation can be performed independently per slice, our framework can leverage multicore computer architectures. We implemented the GSM pipeline on an interlinked Intel Xeon X730 Processor comprised of 32 cores, where GSM computation for the examples shown in this chapter did not exceed 20 minutes. In comparison, the representations constructed by global algorithms such as the discrete scale axis [131], the tight co-cone [46] or the skeleton computed through mesh contraction [11] took less than two minutes for the triangle meshes used in this chapter. However, the GSM



400, $\epsilon = [0.006, 0.135]$

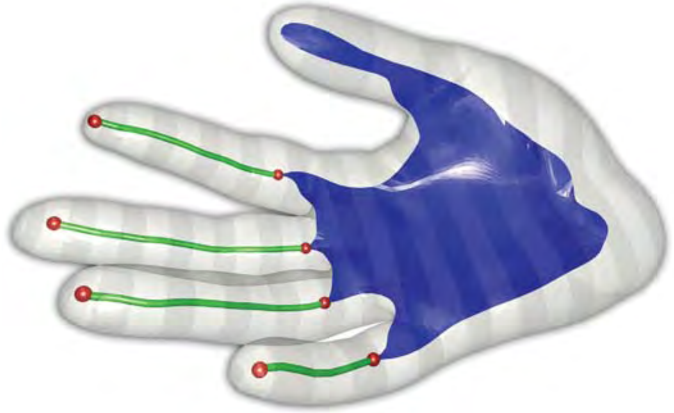


350, $\epsilon = [0.004, 0.099]$

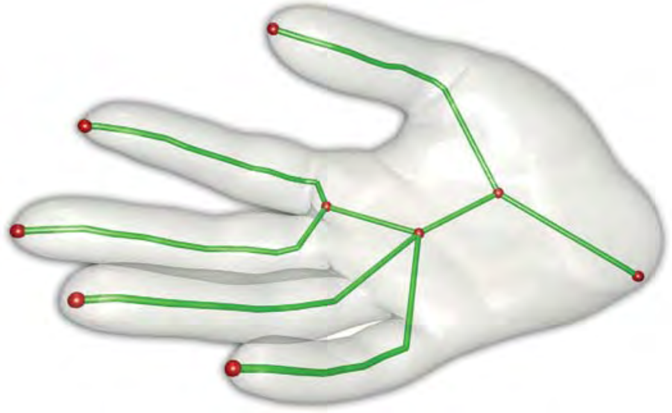


250, $\epsilon = [0.032, 0.151]$

Figure 7.10. GSMs for rockerarm, fertility and pelvis models. Different GSM sheets are shown in different colors. The sweeping strategy is shown for each model on its boundary. For each GSM, the number of slices and the minimum ϵ and maximum ϵ are provided below the respective model.



74, $\epsilon = [0.148, 0.986]$



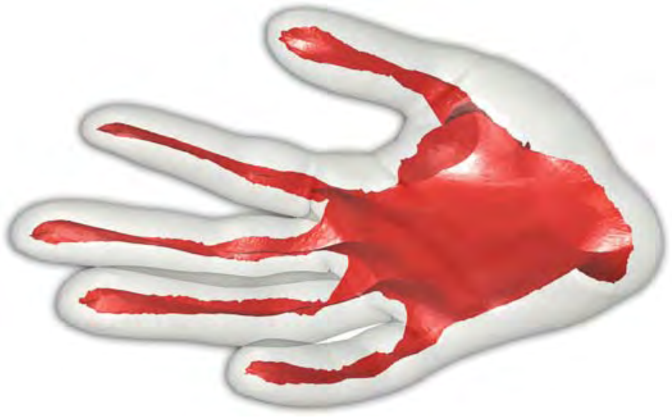


Figure 7.11. Comparison of midstructures. Top: GSM; middle: 1D curve skeleton; right: Discrete Scale Axis. Different GSM sheets are shown in different colors. Furthermore, the sweeping strategy is shown on its boundary. The number of slices and the minimum ϵ and maximum ϵ for the GSM are provided below the respective model. For the curve skeleton, Laplacian constraint scale and positional constraint weight are 2 and 1, respectively. For discrete scale axis, $\delta = 0.01$ and $s = 1.1$.

automatically derives the toplogical structure and classifies sheets, whereas given a medial axis computed from existing techniques, significant additional time is required for sheet classification and other postprocessing.

Figure 7.12 shows an example of an object represented with two different triangulations. The input object in Figure 7.12 (bottom) has a coarser and more irregular triangulation than Figure 7.12 (top). It can be seen that sharper features lead to distortions of the harmonic function, resulting in larger slicing densities in these regions, e.g., tips of the dolphin fins in Figure 7.12. The GSM of the coarser mesh still captures the global shape features represented in the GSM of the object with finer mesh. Since

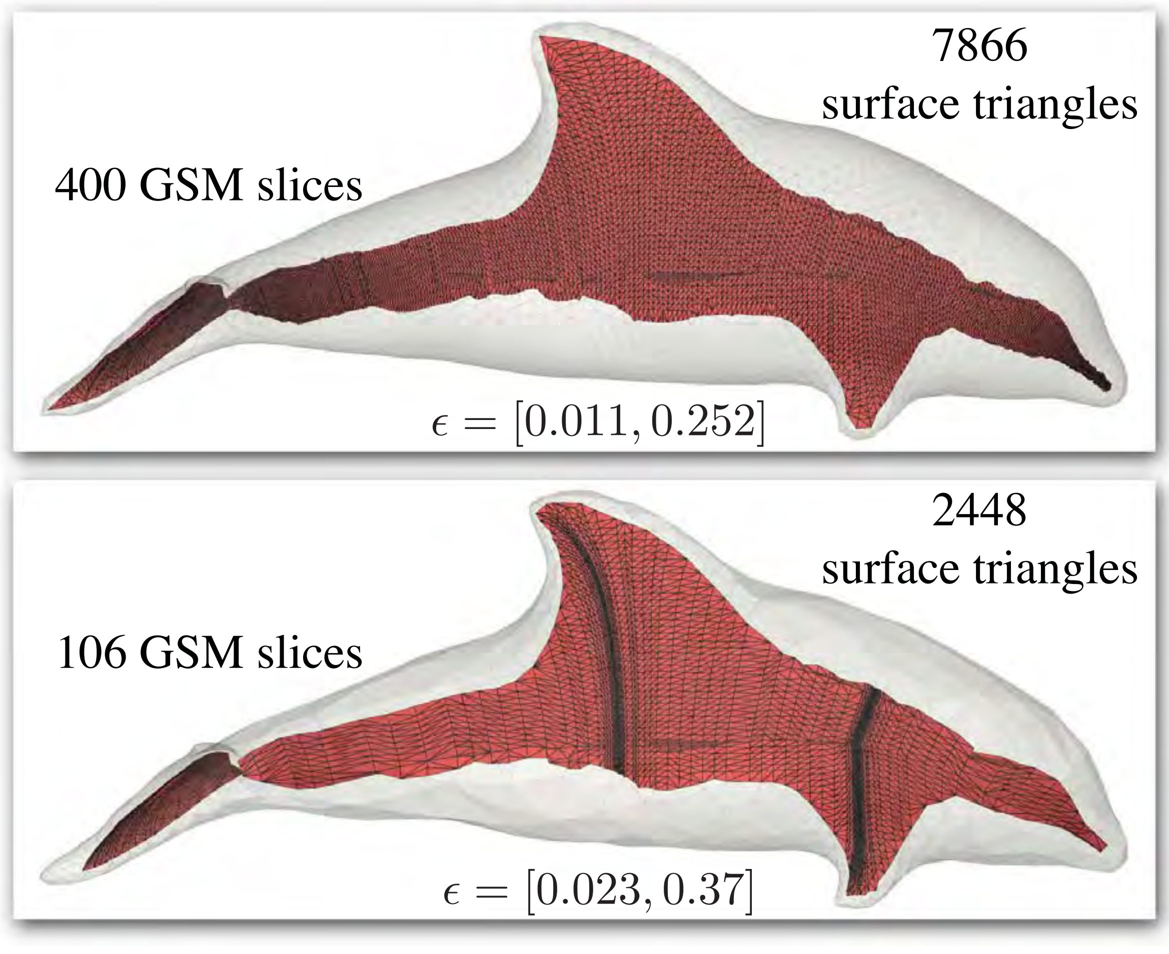


Figure 7.12. GSMs computed on a uniform vs. coarse feature-aware triangle mesh. Both use one minimum and one maximum. Curved slices created from the harmonic functions are swept from tail to nose and capture overhang regions consistently.

the number of slices for the coarser mesh is a quarter of the finer one, the computation time for its GSM is roughly four times faster.

7.7.1 Limitations

The GSM pipeline requires the user to specify critical points to compute a harmonic function. An appropriate choice of these points could be challenging for models with more complex geometry and topology. The extremal points of a 1D curve skeleton are given as hints to the user to recommend critical points. Note that the resulting GSM has then a visually similar structure to a medial axis. Another limitation is that slices have to be of genus-0 (i.e., no inner boundaries), which is due to the proposed medial axis computation algorithm for the slices requiring a closed input curve. In addition, sharp features in the input object may not be preserved if the cutting misses the features. Furthermore, the current graph matching cannot handle complex configurations of topological changes. Finally, the current computation is relatively slow due to the slow B-spline root solving. We plan to address these issues in future work.

7.8 Applications

In this section we highlight two potential applications for which a GSM can be useful. In the first application, a GSM could be used to generate a semistructured hexahedral mesh (Figure 7.13 (a)) by decomposing the input object into simpler subvolumes, where subvolumes correspond to sheets in the GSM. Then, each subvolume is parameterized. Furthermore, the natural parameterization of the GSM could potentially be used for 3D cross field design which is used to generate a hexahedral mesh using a method such as [144]. The proposed GSM pipeline could help in the following way. A desired cutting strategy could be chosen by the user, where the resulting GSM could be used to align hexahedral elements along the chosen sweeping direction.

The second application lets the user deform the object based on the GSM. The consistent topology of the GSM has the potential to produce higher quality deformations compared to other medial and skeleton based shape deformations. Figure 7.13 (b) shows an example of the model in Figure 7.11. The fingers could be deformed using skeleton-based deformation, while the palm could be deformed by editing the surface

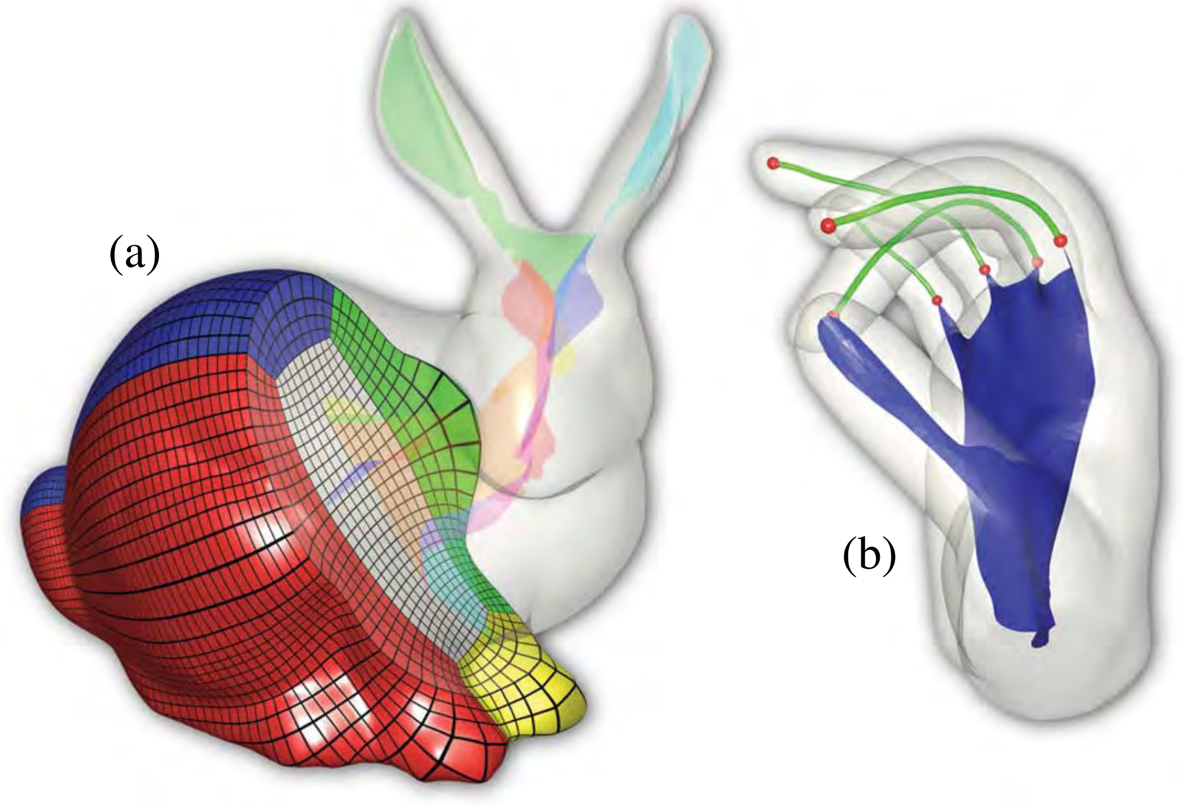


Figure 7.13. Applications of the GSM. (a) Cut through a hexahedral mesh, where the mesh layout was determined by the GSM for the respective model; (b) Deformation based on GSM.

sheet of the GSM through Laplacian mesh editing.

7.9 Conclusions

This chapter presents a new hybrid midstructure called the *Generalized Swept Midstructure* (GSM), containing curve and surface elements with consistent topology. A pipeline to incrementally construct the GSM of polygonal objects is presented that uses a novel planar midstructure computation algorithm in conjunction with an algorithm to match two similar 2D midstructures. The result is a connected structure that is close to the 3D medial axis and respects the topology of the input surface. The sweeping strategy is determined by a user who selects a small set of critical points to define a harmonic function that naturally conforms to an object's shape. The GSM is then incrementally constructed by sweeping midstructures of level sets of the harmonic function.

The structure of a GSM is user controlled via the choice of a sweeping strategy and is therefore flexible to adapt appropriately for specific applications. This is not the case for existing skeleton and 3D medial axis algorithms that determine an intrinsic midstructure. Curve skeletons are more suitable for tube-shaped objects and 3D medial axes are more suitable for objects with more general regions. The hybrid structure of the GSM enables it to be applied for objects consisting of both region types. Existing hybrid skeletonization approaches first compute approximations of 3D medial axes that are then analyzed to differentiate tubular from nontubular regions. However, those approaches are susceptible to topological issues with the 3D medial axis approximation. We have demonstrated potential GSM applications, such as hexahedral meshing and GSM-based shape deformation.

CHAPTER 8

NONLINEAR OPTIMIZATION

In this chapter we present a method for accelerating the convergence of gradient-based nonlinear optimization algorithms. We start with the theory of the Sobolev gradient, which we analyze from a signal processing viewpoint. By varying the order of the Laplacian used in defining the Sobolev gradient, we can effectively filter the gradient and retain only components at certain scales. We use this idea to adaptively change the scale of features being optimized in order to arrive at a solution that is optimal across multiple scales. This is in contrast to traditional descent-based methods, for which the rate of convergence often stalls early once the high frequency components have been optimized. Our method is conceptually similar to multigrid in that it can be used to smooth errors at multiple scales in a problem, but we do not require a hierarchy of representations during the optimization process. We demonstrate how to integrate our method into multiple nonlinear optimization algorithms, and we show a variety of optimization results in variational shape modeling, parameterization, and physical simulation. Being able to optimize shape is a problem of fundamental importance to computer graphics. Several graphics tasks can be posed as shape optimization problems, where an input shape is optimized by iteratively modifying its parameters so that an energy associated with the shape is minimized. The choice of the energy depends on the application at hand. Frequently occurring examples include the Willmore energy for variational shape design [204], an angle- and area-preserving energy for surface parameterization [171], and a strain-based energy for statics problems [195]. In all these cases, the shape optimization problem boils down to a numerical optimization of the appropriate energy, subject to proper boundary conditions.

In many cases of practical importance, including the above examples, finding an optimal solution amounts to solving a set of nonlinear equations. Linear approximations (e.g., [23]) do not always produce the expected result [140], and the minimization of nonlinear energies is often necessary. The numerical optimization of such nonlinear energies can be a lengthy process, usually performed iteratively and off-line for starting shapes that are not already close to optimal. Typically, after the first few iterations, the rate of energy decrease is significantly reduced, and the progress towards the minimum is slow.

The reduction in the rate of energy decrease during the optimization process is not necessarily a sign that the shape is close to optimal. More often, this is an indication that the energy being minimized is ill-conditioned. While minimizing an ill-conditioned energy, the rapid initial decrease is due to the optimization of shape features of high spatial frequency, which requires just a few iterations and results in a large reduction of the energy. In other words, the shape becomes *locally* close to optimal. The subsequent progress towards the desired minimum is slow because features of low spatial frequency must be optimized, which require more *global* changes to the shape. This phenomenon is commonly observed in denoising applications, where the high frequency noise can be rapidly removed in just a few iterations, while the low frequency features take a long time to smooth out [194]. The bottom row of Figure 8.1 shows that after the high frequency details of the dragon model quickly become smooth, the progress towards the global minimum becomes negligible.

8.1 Contributions

Ill-conditioned optimization processes can be sped up by preconditioning, one form of which is to modify the search direction at each step of the optimization process in order to find a direction that allows larger steps towards the minimum. In this chapter, we study the Sobolev preconditioner [97]. In particular, we present

- an analysis that shows the connection between the Sobolev gradient and filter design, leading to an intuitive explanation of the behavior of the Sobolev preconditioner in the context of optimization;

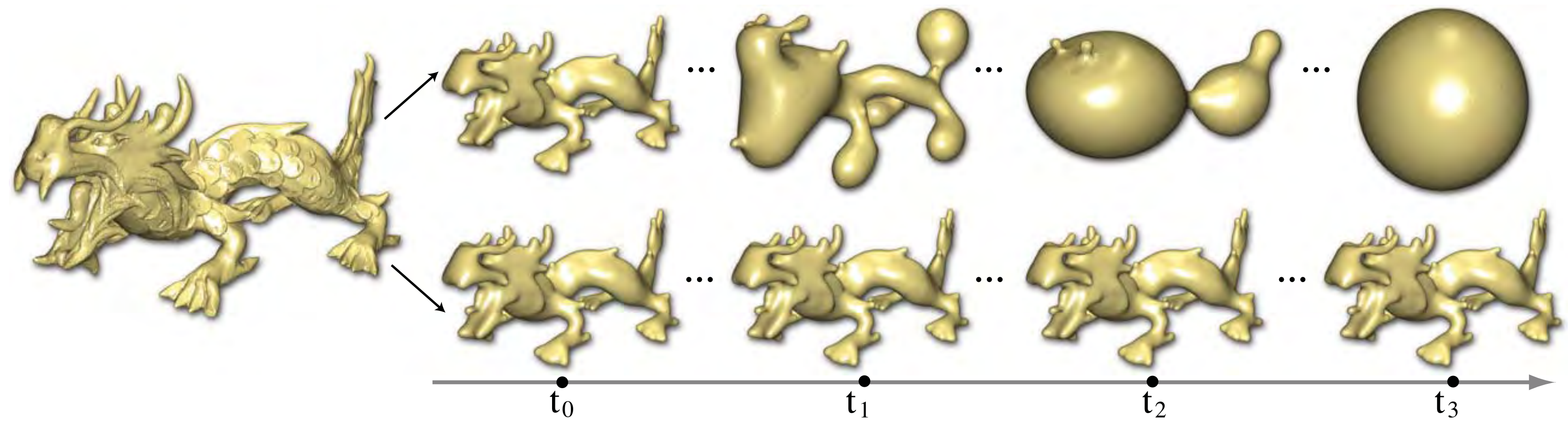


Figure 8.1. A nonlinear, hinge-based bending energy [71] is optimized using gradient descent for a genus-0 triangle mesh. Depending on the minimization state, our algorithm (top row) chooses what feature size to minimize. High frequency features are rapidly removed in the initial optimization steps, and lower frequency features are removed in the later optimization steps. By contrast, a nonpreconditioned optimization (shown in the lower row) produces very little shape change as evident in snapshots taken at similar computational times.

- a novel, multiscale optimization algorithm that uses the Sobolev preconditioner in a general nonlinear optimization pipeline suitable for graphics tasks;
- a novel algorithm for filtering the forces in a physical simulation, which allows larger and more stable timesteps; and
- a comparison that shows the performance benefits of incorporating the Sobolev preconditioner in standard optimization algorithms, such as gradient descent, the nonlinear conjugate gradient method, and the limited memory BFGS (L-BFGS) algorithm [146].

8.2 Overview

Consider the gradient descent optimization algorithm, in which the gradient of the energy is used as the search direction along which to move towards the minimum. At a high level, the Sobolev preconditioner smooths the standard gradient by removing high frequency components, which allows the optimizer to take steps that change the lower frequency, or more global, components of the shape. We extend this to a multiscale setting, tuning the frequency response of the preconditioner at each step of the optimization in order to accelerate the overall convergence of the algorithm. This preconditioner can be easily incorporated into an existing implementation of an optimization algorithm by simply applying the Sobolev preconditioner to the search direction at each iteration and leaving the rest of the optimizer unchanged.

We use the Sobolev preconditioner in the contexts of variational shape modeling, parameterization, and elasticity simulation. The results indicate that even for cases where traditional methods tend to stall and do not produce optimal results, we demonstrate that we are able to achieve far more optimal shapes in reasonable runtimes.

8.3 Sobolev Preconditioning Background

Sobolev preconditioning for elliptic partial differential equations as defined in [97] is a well-known approach with a substantial amount of study [97]. In this section we will focus on previous works that use the Sobolev gradient for scientific computing tasks commonly found in computer graphics.

Renka and Neuberger proposed the use of the Sobolev gradient for curve and surface smoothing [168, 167]. Charpiat et al. [29] and Sundaramoorthi et al. [191] used the Sobolev gradient for speeding up curve flows used for active contours in a more general framework that also considered other inner products. Eckstein et al. [51] solved the surface flow equivalent of the previous two papers. These authors all describe how they get significant performance benefits in their flows by using the H^1 Sobolev gradient instead of the standard L^2 gradient. Inspired by these works, we extend this idea to show how higher-order Sobolev gradients can be used as components in preconditioning filters.

The multigrid method [24] is a conceptually closely related, and commonly used, technique for solving ill-conditioned optimization problems. It has successfully been used in a wide variety of contexts, such as finite element methods [169], parameterization [5], and mesh deformation [183]. The multigrid method requires a hierarchical representation for the shape. At each level of the hierarchy, an iterative smoother is used to remove the highest frequency errors. Since the scale of features removed depends on what frequencies can be represented by the discretization, one can use a hierarchy that spans from coarse to fine discretizations to reduce errors across multiple scales. While multiresolution preconditioners are effective, representing the input shape hierarchically for general representations can be challenging [72].

In Section 8.6, we demonstrate several applications of our method. We include a discussion of work related to the applications in that section.

8.4 Mathematical Framework

In this section, we formally introduce the relationship between the gradient of an energy and the inner product on the space of shape deformations. We start with the L^2 gradient, and we show how other gradients can be constructed. For one particular form of this construction, we provide an interpretation of this operation as a filter applied to the L^2 gradient (see Figure 8.2). In Section 8.5, we exploit this filtering viewpoint to create a shape optimization algorithm in the spirit of a multigrid approach.

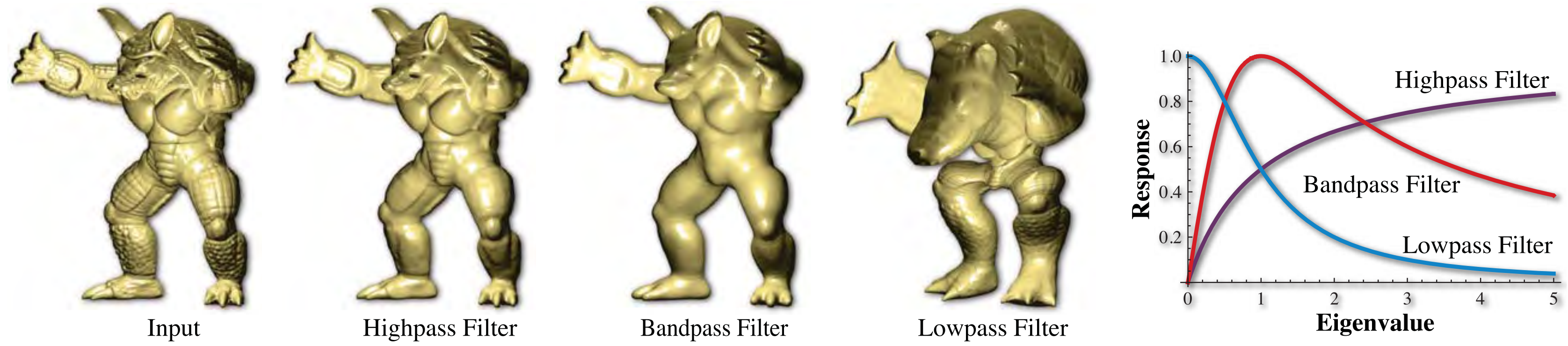


Figure 8.2. Each of the filters shown on the right can be used to select which eigenmodes of the Laplacian to keep in the gradient during the optimization process. The figures on the left, in which we applied the various filters to the L^2 gradient of the Willmore energy, show that the eigenmodes correspond to surface features at different scales.

8.4.1 Shape and Tangent Space

Let Ω be the space of possible shapes for the problem of interest, and let $\mathbf{x} \in \Omega$ be a point in this space. The tangent space, $T_{\mathbf{x}}\Omega$, contains all tangent vectors to Ω at the point $\mathbf{x}$. Given a sufficiently smooth path, $\boldsymbol{\gamma} : \mathbb{R} \to \Omega$, such that $\boldsymbol{\gamma}(t) = \mathbf{x}$, the tangent space represents the space of all possible velocities $\boldsymbol{\gamma}'(t)$ that the path could have at $\mathbf{x}$.

We discretize Ω using a piecewise polynomial basis. In the discrete setting, a point in shape space is given by $\mathbf{x} = \sum_{i=1}^{n} x_i \phi_i$, where $\{\phi_i\}_{i=0}^{n}$ is the set of basis functions and $\{x_i\}_{i=0}^{n}$ is the set of coefficients that defines the shape. For example, one possible discretization of a surface embedded in $\mathbb{R}^3$ is a triangle mesh, for which $\{x_i\}_{i=0}^{n}$ is the set of vertex positions. We can write any path in the discrete setting as $\boldsymbol{\gamma}(t) = \sum_{i=1}^{n} x_i(t)\phi_i$ with corresponding tangent vector given by $\boldsymbol{\gamma}'(t) = \sum_{i=1}^{n} x_i'(t)\phi_i$. As in the continuous setting, the tangent space at a point is the space that contains all possible velocities that a path could have at that point.

8.4.2 Inner Product on Tangent Space

We can equip the tangent space with an inner product, $\langle \mathbf{u}, \mathbf{v} \rangle$ for $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}}\Omega$. The canonical L^2 inner product is given by $\langle \mathbf{u}, \mathbf{v} \rangle_{L^2} = \int \mathbf{u} \cdot \mathbf{v}$.

In the discrete setting, the L^2 inner product for two tangent vectors $\mathbf{u}$ and $\mathbf{v}$ can be written in matrix form as

$$\begin{aligned} \langle \mathbf{u}, \mathbf{v} \rangle_{L^2} &= \langle \sum_{i=1}^{n} u_i \phi_i, \sum_{j=1}^{n} v_i \phi_i \rangle_{L^2} = \sum_{i=1}^{n} \sum_{j=1}^{n} u_i v_j \langle \phi_i, \phi_j \rangle_{L^2} \\ &= \mathbf{u}^T \mathbf{M} \mathbf{v} \ , \end{aligned}$$

where in the second line, $\mathbf{u}$ and $\mathbf{v}$ are the vectors of coefficients $\{v_i\}_{i=1}^{n}$ and $\{u_i\}_{i=1}^{n}$, and $\mathbf{M}$ is commonly called the "mass matrix" whose entries are defined by $\mathbf{M}_{ij} = \langle \phi_i, \phi_j \rangle_{L^2}$.

As noted by Eckstein et al. [51], we can define alternate inner products using

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{A}} = \langle \mathcal{A}\mathbf{u}, \mathbf{v} \rangle_{L^2} = \langle \mathbf{u}, \mathcal{A}\mathbf{v} \rangle_{L^2} \ , \tag{8.1}$$

where $\mathcal{A} : T_{\mathbf{x}}\Omega \to T_{\mathbf{x}}\Omega$ is a self-adjoint, positive definite linear operator.

The choice of inner product is key and, as we will show, has a large impact on the performance of descent-based algorithms. Our goal in Section 8.5 will be to tailor $\mathcal{A}$ to the task of optimizing the performance of these algorithms.

8.4.3 Gradient Vector Field

Given a smooth energy, $E : \Omega \to \mathbb{R}$, that assigns a real value to a given shape, the gradient of E at $\mathbf{x}$ is defined as the vector field, $\mathbf{g}(\mathbf{x}) \in T_{\mathbf{x}}\Omega$, that satisfies

$$\langle \mathbf{g}(\mathbf{x}), \mathbf{v} \rangle = \lim_{\epsilon \to 0} \frac{E(\mathbf{x} + \epsilon \mathbf{v}) - E(\mathbf{x})}{\epsilon}$$

for all $\mathbf{v} \in T_{\mathbf{x}}\Omega$ (see, e.g., [48]). In other words, the gradient is the vector field whose inner product with $\mathbf{v}$ yields the rate of change of the energy along $\mathbf{v}$. Since the gradient is defined with respect to an inner product, each inner product leads to a different vector field corresponding to the gradient. We can relate the $\mathcal{A}$ gradient to the L^2 gradient by combining $\langle \mathbf{g}_{\mathcal{A}}(\mathbf{x}), \mathbf{v} \rangle_{\mathcal{A}} = \langle \mathcal{A}\mathcal{A}^{-1}\mathbf{g}_{L^2}(\mathbf{x}), \mathbf{v} \rangle_{L^2}$ with (8.1) to obtain

$$\mathbf{g}_{\mathcal{A}}(\mathbf{x}) = \mathcal{A}^{-1}\mathbf{g}_{L^2}(\mathbf{x}) \ . \tag{8.2}$$

Therefore, if we have an expression for the L^2 gradient of the energy, we can solve for the $\mathcal{A}$ gradient using this equation.

In the discrete setting, the energy E depends on the coefficients $\{x_i\}_{i=1}^n$, which we write using an abuse of notation as $E(\mathbf{x})$. We can evaluate the rate of change of E at $\mathbf{x}$ in the direction of $\mathbf{v}$ as

$$\sum_{i=1}^{n} \frac{\partial E(\mathbf{x})}{\partial x_i} v_i = \frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{v} = \langle \mathbf{g}_{L^2}(\mathbf{x}), \mathbf{v} \rangle_{L^2} = \left(\mathbf{g}_{L^2}(\mathbf{x})\right)^T \mathbf{M}\mathbf{v},$$

which, using the symmetry of $\mathbf{M}$ and the fact that this equation must hold for arbitrary $\mathbf{v}$, leads to

$$\mathbf{g}_{L^2}(\mathbf{x}) = \mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \ .$$

From (8.2), we obtain that for a linear operator $\mathcal{A}$,

$$\mathbf{g}_{\mathcal{A}}(\mathbf{x}) = \mathcal{A}^{-1}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \tag{8.3}$$

8.4.4 Frequency Response of Gradient Filter

Motivated by the work of Kim and Rossignac [100] on constructing filters for mesh processing, we choose $\mathcal{A} = \mathbf{B}^{-1}\mathbf{A}$, where $\mathbf{B}$ and $\mathbf{A}$ are both linear combinations of powers of the Laplace-Beltrami operator, Δ, given by $\mathbf{A} = \sum_{i=0}^{p}(-1)^i a_i \Delta^i$ and $\mathbf{B} = \sum_{i=0}^{q}(-1)^i b_i \Delta^i$. Here, $\{a_i\}_{i=0}^{p}$ and $\{b_i\}_{i=0}^{q}$ are the sets of scalar coefficients that define each linear combination, and $\Delta^0 = \mathbf{I}$ is the identity operator. Note that using this definition for $\mathcal{A}$, the resulting inner product is no longer symmetric. However, for the purposes of optimization, this is a valid choice because the resulting gradient can still be used as a descent direction.

In order to construct the $\mathcal{A}$ gradient in the discrete setting, we require a discrete version of the Laplace-Beltrami operator, denoted by $-\mathbf{L}$. For example, we use the well-known cotan operator [160] for optimization problems involving triangle meshes. In general, we define the discrete Laplace-Beltrami operator as $-\mathbf{L} = -\mathbf{M}^{-1}\mathbf{K}$, where $\mathbf{M}$ is the mass matrix defined as before by $\mathbf{M}_{ij} = \langle \phi_i, \phi_j \rangle$ and $\mathbf{K}_{ij} = \langle \nabla\phi_i, \nabla\phi_j \rangle_{L^2}$ is the stiffness matrix.

Given a discrete Laplace-Beltrami operator, we can write

$$\mathbf{A} = \sum_{i=0}^{p} a_i \mathbf{L}^i \qquad \text{and} \qquad \mathbf{B} = \sum_{i=0}^{q} b_i \mathbf{L}^i \tag{8.4}$$

and use (8.3) to obtain

$$\mathbf{A}\mathbf{g}_{\mathcal{A}} = \mathbf{B}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \ . \tag{8.5}$$

Kim and Rossignac [100] show that, under some reasonable assumptions, the eigendecompositon of $\mathbf{L}$ can be written as

$$\mathbf{L} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \ ,$$

where $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements, $\lambda_i = \mathbf{\Lambda}_{ii}$, are the eigenvalues of $\mathbf{L}$, and $\mathbf{Q}$ is a matrix whose columns are the eigenvectors of $\mathbf{L}$. Using this decomposition, we can write (8.5) as

$$\left(\sum_{i=0}^{p} a_i \mathbf{\Lambda}^i\right) \mathbf{Q}^{-1}\mathbf{g}_{\mathcal{A}} = \left(\sum_{i=0}^{q} b_i \mathbf{\Lambda}^i\right) \mathbf{Q}^{-1}\mathbf{g}_{L^2} \ .$$

Each row of this equation corresponds to an eigenmode of $\mathbf{L}$, which we can use to examine what happens to the corresponding eigenmode of $\mathbf{L}$ that is present in $\mathbf{g}_{L^2}$ when we compute $\mathbf{g}_{\mathcal{A}}$. We rewrite row j as

$$\left(\mathbf{Q}^{-1}\mathbf{g}_{\mathcal{A}}\right)_j = h(\lambda_j)\left(\mathbf{Q}^{-1}\mathbf{g}_{L^2}\right)_j \ ,$$

where $h(\lambda)$ is the transfer function given by

$$h(\lambda) = \frac{\sum_{i=0}^{q} b_i \lambda^i}{\sum_{i=0}^{p} a_i \lambda^i} \ . \tag{8.6}$$

This transfer function encodes the amount to which the eigenmode with eigenvalue λ is amplified or attenuated when computing $\mathbf{g}_{\mathcal{A}}$ from $\mathbf{g}_{L^2}$. Thus, by varying the coefficients a_i and b_i, we can control the frequency response of $\mathcal{A}^{-1}$, which we exploit in the following section when designing our optimization procedure.

8.5 Algorithm

The goal of shape optimization is to find a path from a given initial shape $\mathbf{x}_0 = \boldsymbol{\gamma}(0)$ to $\mathbf{x}_{\min} = \lim_{t\to\infty} \boldsymbol{\gamma}(t)$ such that $E(\mathbf{x}_{\min})$ is a minimum. Perhaps the simplest such algorithm is gradient descent, which chooses the velocity of the path to be aligned with the gradient of the energy, so that $\boldsymbol{\gamma}'(t) = -\mathbf{g}(\mathbf{x})$. A variant of this is the nonlinear conjugate gradient method, which uses information from previous search directions to guide the optimization towards a solution. The Newton-type methods, such as L-BFGS [146], compute a second-order approximation to the energy at each step to perform the optimization. Regardless of the particulars of each algorithm, most descent-based methods can be described as iteratively computing a search direction and performing a line search that approximately minimizes the energy along this direction. We apply our filtering to this descent direction.

Given a tuple $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A}$ and $\mathbf{B}$ are defined as in the previous section, Algorithm 1 shows how we modify the nonlinear conjugate gradient method to minimize $E(\mathbf{x})$. Traditionally, $\mathbf{A} = \mathbf{B} = \mathbf{I}$ so that $\mathbf{g}_0$ and $\mathbf{g}_1$ correspond to the L^2 gradient. In this case, the algorithm very quickly minimizes the high frequencies features in the shape. However, lower frequency features are minimized only very slowly, and the algorithm tends to stall as the convergence rate slows down.

Let $\mathbb{F} = \{(\mathbf{A}_k, \mathbf{B}_k)\}_{k=1}^m$ be a set consisting of m tuples, with each element of $\mathbb{F}$ corresponding to a different choice for the coefficients $\{a_i\}_{i=0}^p$ and $\{b_i\}_{i=0}^q$ in (8.4). Each tuple in $\mathbb{F}$ also corresponds to a different transfer function $h_k(\lambda)$. We set $\mathbf{A}_0 = \mathbf{B}_0 = \mathbf{I}$. Our algorithm is designed such that each one of these transfer functions attenuates a different frequency range. For the transfer function in (8.6), Kim and Rossignac [100]

Algorithm 1 : MinimizeCG($(\mathbf{A}, \mathbf{B})$, $\mathbf{x}$)

```
$\mathbf{g}_0 \leftarrow \mathbf{A}^{-1}\mathbf{B}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$
$\mathbf{s}_0 \leftarrow -\mathbf{g}_0$
for $i = 1 \to$ max iterations do
    $\alpha \leftarrow LineSearch(\mathbf{x}, \mathbf{s}_0)$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\, \mathbf{s}_0$
    $\mathbf{g}_1 \leftarrow \mathbf{A}^{-1}\mathbf{B}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$
    $\beta \leftarrow \langle \mathbf{g}_1, \mathbf{g}_1 \rangle / \langle \mathbf{g}_0, \mathbf{g}_0 \rangle$
    $\mathbf{s}_0 \leftarrow \beta\, \mathbf{s}_0 - \mathbf{g}_1$
    $\mathbf{g}_0 \leftarrow \mathbf{g}_1$
end for
return  $\mathbf{x}$
```

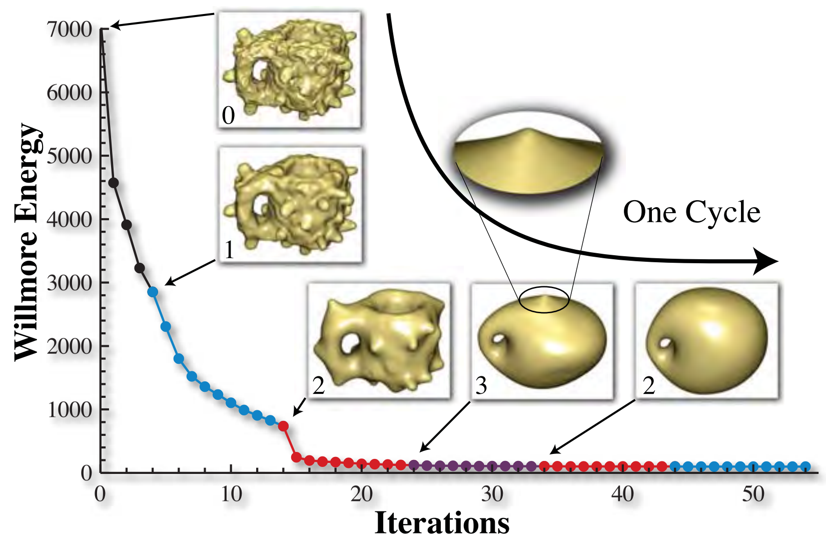
provide a method for computing coefficients based on an intuitive set of parameters available to the user. In our case, we have found that the algorithm works well with only three filters ($m = 3$), shown in Figure 8.2, which can be viewed as a highpass, bandpass, and lowpass filter. The coefficients are given by $\{a_0 = 1, a_1 = 1, b_1 = 1\}$, $\{a_0 = 1, a_2 = 1, b_1 = 2\}$, and $\{a_0 = 1, a_2 = 1, b_0 = 1\}$ for the highpass, bandpass, and lowpass filters, with all other coefficients equal to zero. The form of the transfer function in (8.6) is general enough so that more sophisticated filters could be designed if more levels are required for the optimization.

Algorithm 2 summarizes the procedure we use to iteratively optimize the shape at multiple scales. We start by optimizing the high frequency components with $k = 0$, and then optimize lower and lower frequency components by increasing k until the maximum is reached. While increasing k will optimize more global features, it may introduce higher frequency features during the minimization process (see inset in Figure 8.3–left). Thus, one cycle of Algorithm 2 involves first increasing k and then decreasing k as we move up and down the hierarchy of frequencies during optimization to ensure that the shape is optimized at all scales, similar to the multigrid V-cycle. The algorithm terminates when the magnitude of changes to the shape is below a given threshold or when the maximum number of cycles has been reached.

Figure 8.3–left shows the convergence plot for minimizing the Willmore energy of a genus-1 object. Interestingly, the minimum of the energy is not unique, and the resulting shape is not necessarily a symmetric torus. Note that as we increase k, even though the energy drops only very slowly, the shape still undergoes large, global deformations.

Algorithm 2 : MinimizeMultiScale(x)

```
for $j = 1 \rightarrow$ number of cycles do
    for $k = 0 \rightarrow m$ do
        $\mathbf{x} \leftarrow \mathbf{MinimizeCG}\big((\mathbf{A}_k, \mathbf{B}_k), \mathbf{x}\big)$
    end for
    for $k = m - 1 \rightarrow 0$ do
        $\mathbf{x} \leftarrow \mathbf{MinimizeCG}\big((\mathbf{A}_k, \mathbf{B}_k), \mathbf{x}\big)$
    end for
end for
return x
```



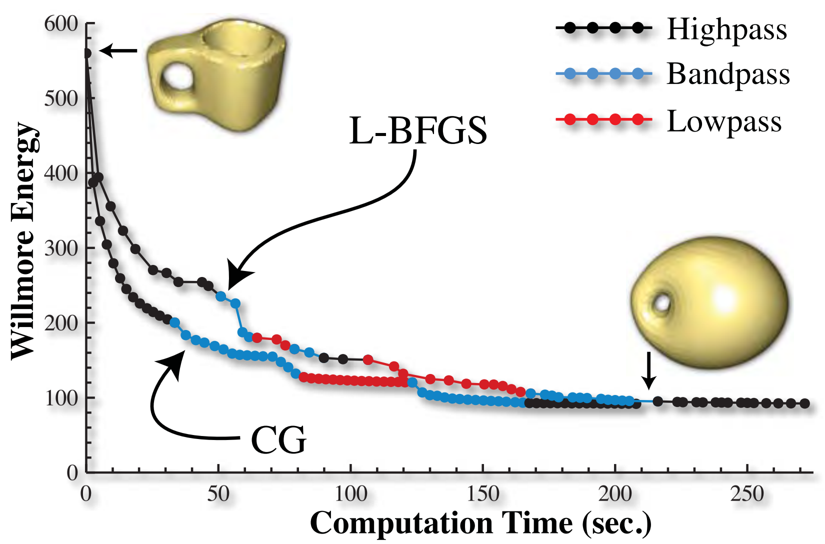


Figure 8.3. Convergence of multiscale algorithm. Left: One cycle ($k = 0, 1, 2, 3, 2, 1$), where k is the filter index, to minimize bending energy on the input mesh (top left). The index is increased when the magnitude of changes to the shape falls below a given threshold in order to minimize more global features and then decreased to remove small-scale features introduced by the lower pass filters. Right: Multiscale minimizers based on our implementation of the nonlinear conjugate gradient method and the L-BFGS method exhibit similar convergence rates. While L-BFGS requires fewer iterations, it requires more time per iteration.

The call to Algorithm 1 in Algorithm 2 can be replaced with other nonlinear optimization algorithms such as gradient descent or L-BFGS. Figure 8.3–right shows a comparison between our implementation of the nonlinear conjugate gradient method, shown in Algorithm 1, and the L-BFGS algorithm, for which we use a library that can be found online [149]. While L-BFGS requires fewer steps to reach the minimum, each step requires more computation. The result for gradient descent looks very similar to these plots and has been omitted for clarity.

Our implementation of the nonlinear conjugate gradient method uses Brent's line-search method [161], which requires an upper bound, ϵ, on the stepsize. If Brent's method uses a stepsize close to ϵ, we increase the stepsize to $2\,\epsilon$. Conversely, if the maximum number of iterations is reached, it is called again with an upper bound of $\epsilon/2$. A starting value of $\epsilon = 0.1$ is used for all examples in this chapter. The sparse matrix implementation of the Boost library is used to store $\{\mathbf{A}_k, \mathbf{B}_k\}_{k=1}^{m}$. We use PARDISO [172] to solve the linear system in (8.4) that defines the preconditioned gradient. Note that we could use an iterative solver, such as the conjugate gradient method, instead of a direct solver, which would avoid having to compute and store higher-order Laplacians.

8.6 Applications

The following sections show how the algorithm above speeds up nonlinear problems found in typical computer graphics applications.

8.6.1 Variational Modeling

The variational modeling of surfaces is the process of constructing aesthetically pleasing smooth shapes by optimizing some curvature-based energy. The inputs are generally user-specified positional and tangential constraints.

A commonly used surface energy is the second-order bending energy that computes the area integral of the square of the mean curvature of the surface, $E = \int H^2 dA$, also known as the Willmore energy [204], where H is the mean curvature. Given appropriate boundary conditions, variants of this formulation (e.g., $\int({\kappa_1}^2 + {\kappa_2}^2)dA$, where κ_i are the principal curvatures, and $\int H^2 dA + \int K dA$ where K is the Gaussian curvature) differ only by constants that depend on topological type and are equivalent in terms of optimization. In this chapter, we use the dihedral-angle-based discrete operator [71] for approximating the Willmore energy for triangle meshes (Figure 8.1).

Minimizing the Willmore energy has the effect that during the minimization process, the surface gets more and more spherical (see Figures 8.1, 8.3, and 8.4). Often however, this behavior is undesirable when the user does not want the shape to bulge out (see Figure 8.5).

If the bulgy results using the Willmore energy are undesirable, a different energy has to be defined to prevent this behavior. Mehlum and Tarrou [129] proposed to minimize the squared variation in normal curvature integrated over all directions in the tangent plane:

$$\frac{1}{\pi}\int\left(\int_0^\pi \kappa_n'(\theta)^2 d\theta\right) dA \tag{8.7}$$

This is a third-order energy since it requires taking the derivative of the normal curvature, where the resulting expression contains third-order partial derivatives. Third-order energies tend to be even more ill-conditioned than second-order energies such as the Willmore energy, which can make their optimization prohibitively expensive using traditional nonlinear minimization algorithms. Other examples where higher-order energies are desired are discussed in [92].

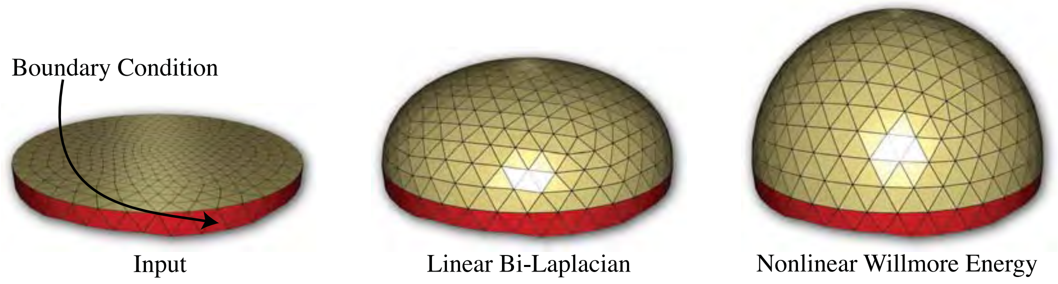


Figure 8.4. The user-specified tangent constraints (red triangles) are intended to force the input shape (left) to inflate into a hemisphere. A linear method, such as solving the bi-Laplacian in x, y and z, is not able to reconstruct the hemisphere (middle). With our framework, the nonlinear Willmore energy is minimized in a few steps to create the desired shape (right).

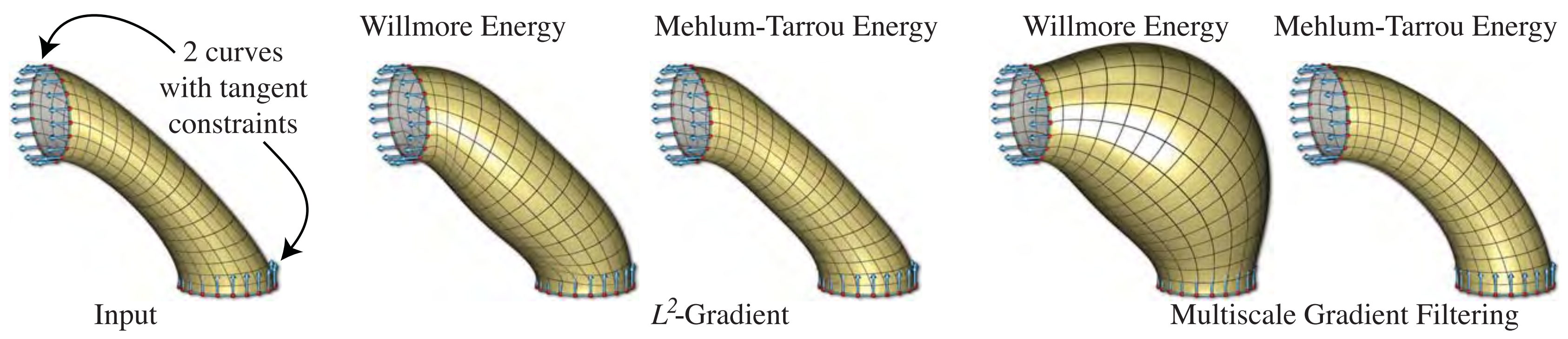


Figure 8.5. Starting from a C^2-continuous B-spline surface generated from two input curves (left), we minimize the second-order Willmore energy and the third-order Mehlum-Tarrou energy subject to positional and tangential constraints. A traditional descent-based algorithm using the L^2 gradient stalls after a few iterations (middle), whereas the Multiscale Gradient Filtering approach converges to a more optimal solution for both energies (right).

Figure 8.5 shows a comparison between the Willmore energy and the Mehlum-Tarrou energy as defined in (8.7) using our approach. The input to the optimization is a C^2-continuous nonuniform bicubic B-spline that is fit through two B-spline curves with tangential constraints. The energy and gradient for this surface representation is straightforward to compute using the equations in [129]. In order to achieve the optimal configuration, the shape needs to move globally into a curved cylinder. The figure also shows a comparison to the state of the surfaces minimized without preconditioning the gradients, demonstrating that our method yields a more optimal shape than the traditional approach in the same computational time.

8.6.2 Parameterization

Computing high quality surface parameterizations for use in applications such as texture mapping is a well studied topic (see, e.g., [61]). The classic aim is to find a suitable embedding from $\mathbb{R}^3 \to \mathbb{R}^2$ that minimizes some form of measured distortion. Numerous proposals have been made for defining distortion metrics which produce pleasing results. Perhaps the most popular has been conformal mapping, which attempts to preserve angular distortion. Such conformal solutions are efficient to compute, requiring a linear solve, but unfortunately do a poor job of preserving area (see Figure 8.6 for a comparison).

When considering preserving both area and angle during flattening one must turn to nonlinear metrics. A family of such metrics have been proposed based on the singular values of the 3×2 Jacobian matrix J_i that maps each triangle T_i from 3D into the 2D parametric domain[171, 189, 208]. A perfect isometric mapping will have singular values equal to 1. Large and small singular values imply stretching and shrinking respectively.

For our experiments we use the L^2 *texture stretch* metric from[171]. Given a triangle T_i from mesh M, its root-mean-square stretch error can be defined as follows: $E(T_i) = \sqrt{(\tau + \gamma)/2}$, where τ and γ are the singular values of the triangle's Jacobian matrix J_i. The entire parametrization can be computed by minimizing the following expression over the mesh:

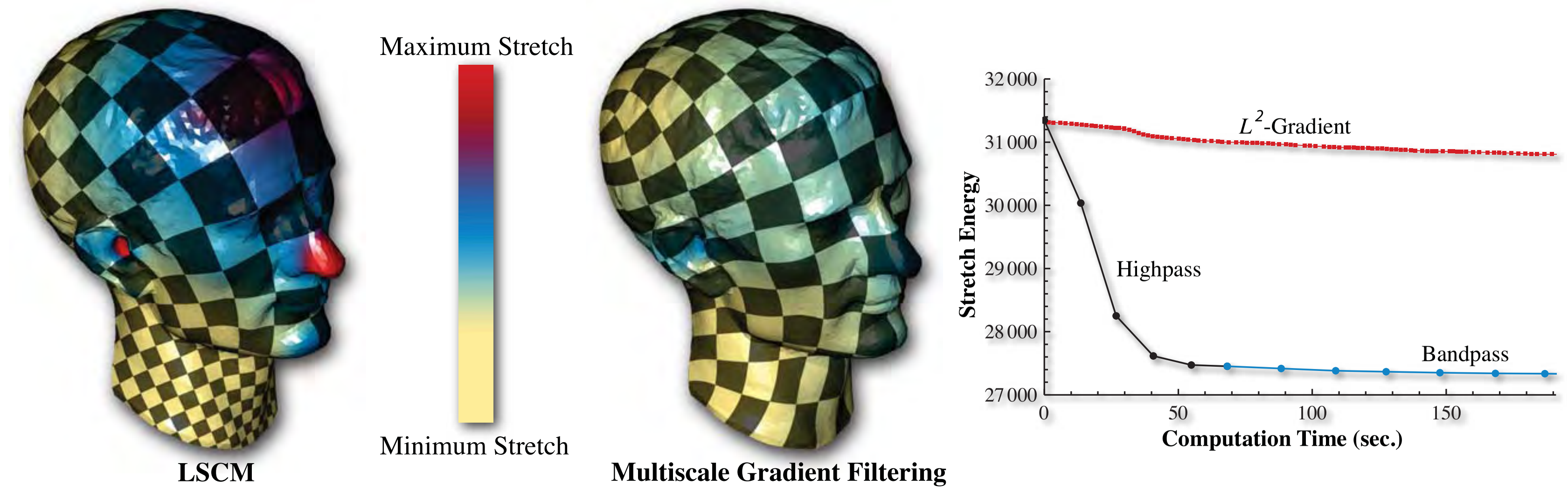


Figure 8.6. Starting from the LSCM parameterization, the Multiscale Gradient Filtering approach is used to minimize texture stretch. By varying the filter, a minimum is achieved much faster than by just minimizing the energy using the L^2 gradient.

$$\sqrt{\sum_{T_i \in M} E(T_i)^2 A(T_i) / \sum_{T_i \in M} A(T_i)} \tag{8.8}$$

where $A(T_i)$ is the surface area of the triangle in 3D. When a triangle's area in 2D approaches zero (e.g., becomes degenerate), its stretch error $E(T_i)$ goes to infinity. Thus, relatively small perturbations of the parameterization can result in large changes in expression (8.8). For this reason the nonlinear system is quite stiff, producing a set of gradients during optimization that may greatly vary in magnitude. The effect of this is a system that is challenging to solve efficiently.

We applied our gradient preconditioning method to solve this problem. Figure 8.6 shows the results of our method for minimizing texture stretch. We start with an initial parametrization using a Least Squares Conformal Map [114], shown on the left. We then run our optimization procedure on the nonlinear energy to obtain the results shown in the middle, demonstrating that we obtain a solution with much less distortion. On the right, we compare the energy behavior to no preconditioning, demonstrating that we obtain the optimal shape with far less computation.

In Chapter 4, 5 and 6, modeling methodolgies are discussed which create a volumetric parameterization from an input triangle mesh. In these methodologies the object of interest is decomposed into simpler subvolumes, where each volume is parameterized by solving Laplace's Equation. Since for each parameteric direction, Laplace's Equation is solved independently, there is no guarantee that the vector fields of the resulting scalar fields are orthogonal. The left column of Figure 8.7 shows a 2D and 3D illustration of this observation. The optimization framework in this chapter is used to improve these parameterizations in the following.

Hormann et al. [81] proposed the MIPS parameterization method which constructs a global parameterization of a triangulated surface over a planar region with minimal distortion. In particular, Hormann et al. define an energy functional, examining the mapping between a triangle in parameter space to its corresponding triangle in world space. Since this mapping can be described with a single linear map, Hormann et al. [81] use the two singular values of the respective transformation matrix, and define

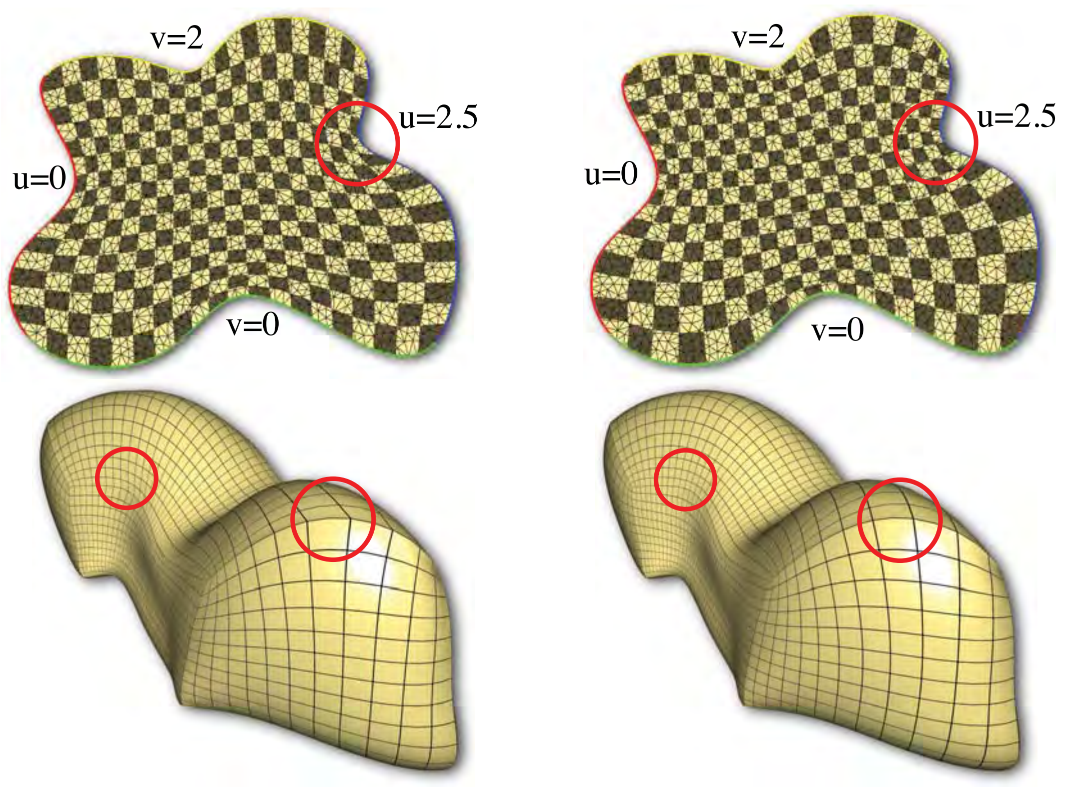


Figure 8.7. Parameterizations in 2D (top row) and 3D (bottom row). The left parameterization in each row shows regions (circles) where the vector fields of the respective parameterization is not orthogonal. Resulting elements are skewed. The right parameterization of each row shows the improved parameterization.

the energy of a triangle to be the ratio of the singular values. If the singular values are equal it implies that the element is at most uniformly deformed (unifom scaling). While Hormann et al. [81] only showed examples for the 2D case, a similar energy functional can be described for the 3D case, where the mapping is a 3x3 matrix corresponding to singular values σ_1, σ_2 and σ_3. Namely, a suitable 3D energy functional is $E = \sigma_1/\sigma_2 + \sigma_1/\sigma_3 + \sigma_2/\sigma_3$, where $\sigma_1 \geq \sigma_2 \geq \sigma_3$. Figure 8.7 shows intial results to improve parameterizations based on this energy. In contrast to the method by Hormann et al. [81], the rectangular shape of the parametric domain cannot be changed during the optimization, but vertices can be moved along the parameteric boundaries. The shape of the parameteric domains has to be maintained, so that a B-spline surface or volume, respectively, can be fit to the optimized parameterization, as discussed in Chapter 9. Furthermore, the energy functional in the MIPS method does not create orthogonal elements close to the boundary. Therefore, in our implementation, a different energy functional is used for triangles or tetrahedra at the boundary to improve orthogonality for the regions close to the boundary.

8.6.3 Nonlinear Elasticity

To this point, we have considered typical shape optimization problems, where the user inputs a shape that is then optimized with respect to a given energy using the algorithm discussed in Section 8.5. In these applications, the user is generally only interested in the end state and not in the path taken to get there. However, we demonstrate that our proposed framework can also be applied to the simulation of the dynamics of elastic objects, where the user is indeed interested in the path.

We begin with Hooke's law for continuum mechanics (see, e.g., [122]), which states that the stress, $\boldsymbol{\sigma}$, for an isotropic elastic material is linearly related to the strain, $\boldsymbol{\epsilon}$, by

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda \mathrm{Tr}\left(\boldsymbol{\epsilon}\right)\mathbf{I} \ ,$$

where λ and μ are the Lamé first and second parameters that determine the elastic response of the material, $\mathrm{Tr}\left(\boldsymbol{\epsilon}\right) = \sum_k \boldsymbol{\epsilon}_{kk}$ is the trace of the strain tensor, and $\mathbf{I}$ is the identity. The finite strain tensor encodes the deformation of the material from its undeformed configuration at $\mathbf{x}_0$ to its deformed configuration $\mathbf{x}$ and is defined as $\boldsymbol{\epsilon} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$, where $\mathbf{F}$ is the deformation gradient, which maps tangent vectors from the undeformed to the deformed configuration. The elastic potential energy, a function of $\mathbf{x}$, is defined as

$$U\left(\mathbf{x}\right) = \frac{1}{2}\int \mathrm{Tr}\left(\boldsymbol{\sigma}\boldsymbol{\epsilon}\right) dA \ , \tag{8.9}$$

where the integration is over the undeformed configuration. Figure 8.8 shows an example where $U(\mathbf{x})$ is minimized on an hexahedral mesh using a trilinear Bézier basis. Similar

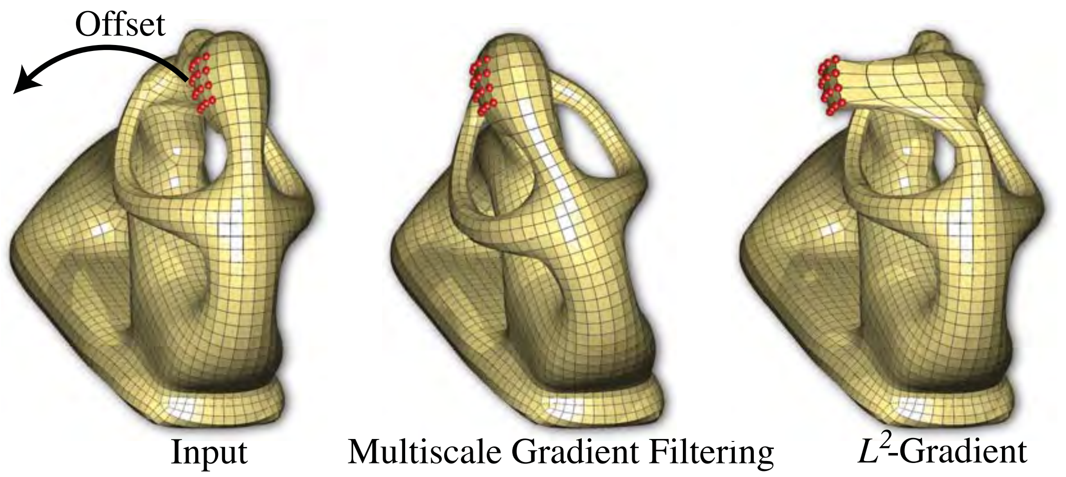


Figure 8.8. We minimize elastic potential energy based on nonlinear strain on an unstructured hexahedral mesh with a trilinear Bézier basis. Multiscale Gradient Filtering quickly propagates the boundary conditions through the shape, whereas a minimizer based on the L^2 gradient stalls.

to the previous cases, the preconditioned gradients help to propagate the deformations to the rest of the mesh much faster.

The equations of motion for the elastic material with Rayleigh damping are

$$\mathbf{M}\,\ddot{\mathbf{x}} + \mathbf{C}\,\dot{\mathbf{x}} - \mathbf{f}_{\text{int}}\,(\mathbf{x}) = \mathbf{f}_{\text{ext}}\,(\mathbf{x}, \dot{\mathbf{x}})\,, \tag{8.10}$$

where $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the first and second derivative of $\mathbf{x}$ with respect to time, $\mathbf{M}$ is the physical mass matrix, $\mathbf{C}$ is the Rayleigh damping matrix, $\mathbf{f}_{\text{ext}}$ is a vector of external forces such as gravity, and $\mathbf{f}_{\text{int}}\,(\mathbf{x}) = -\,\partial U(\mathbf{x})/\partial \mathbf{x}$ is a vector of internal forces that depends on the elastic potential energy. The equations of motion define a coupled system of ordinary differential equations that must be integrated in time to solve for the path of the object. For stiff materials, implicit integrators are often used because they offer superior stability over explicit methods [14], resulting in a costly system of nonlinear equations that must be solved at each timestep. Furthermore, since the equations are nonlinear, even implicit integrators are not unconditionally stable, resulting in a restriction on the maximum allowable timestep size [78].

Various strategies exist for reducing the amount of computation needed to integrate the equations of motion. Since many solid materials, such as metal and wood, usually undergo small deformations, the infinitesimal strain tensor, given by $\boldsymbol{\epsilon} = \frac{1}{2}\left(\mathbf{F}^T + \mathbf{F}\right) - \mathbf{I}$, can be used instead of the finite strain tensor in the definition of the elastic potential energy. In the case of large deformation, the naïve use of the infinitesimal strain tensor results in undesirable distortions and ghost forces, so corotational approaches [136, 137], which factor out rotations from the motion on a per-element or per-vertex basis, are often employed. Both for the small displacement case and the corotational approaches, replacing the finite strain tensor with the infinitesimal strain tensor and using an implicit integrator results in having to solve only a linear rather than a nonlinear system per timestep, saving a significant amount of computation. In the case of small deformations, the resulting system is also unconditionally stable, which is very attractive since it allows large timesteps to be used.

We propose an alternative for allowing one to take larger timesteps while integrating the equations of motion based on our framework. We modify Equation 8.10 to filter the internal forces, so the modified equations are given by

$$\mathbf{M}\,\ddot{\mathbf{x}} + \mathbf{C}\,\dot{\mathbf{x}} - \mathcal{A}^{-1}\mathbf{f}_{\text{int}}(\mathbf{x}) = \mathbf{f}_{\text{ext}}(\mathbf{x}, \dot{\mathbf{x}}) \ , \tag{8.11}$$

where $\mathcal{A}^{-1}$ is designed to remove the high frequency components from the internal forces. By filtering the forces, the physics of the system is changed; however, for applications where the timestep size must remain fixed, we can choose $\mathcal{A}$ to filter those frequencies that cannot be resolved, allowing a larger timestep. As shown in Figure 8.9, this approach does not exhibit any of the distortions apparent when using the small deformation assumption and, like the corotational approach, allows for large timesteps, even if an explicit integrator is used. We believe that this approach can also be used in conjunction with a corotational method, allowing larger timesteps within that context and for implicit integrators in general. In addition, this approach is not limited to the linear constitutive equation given by Hooke's law but can be applied to filter out undesirable frequencies from any forces.

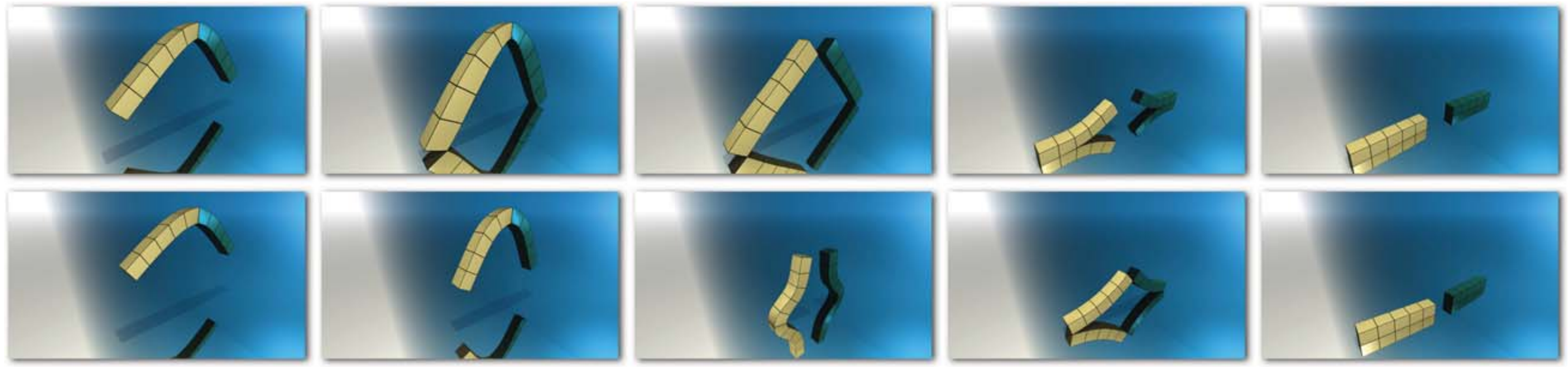


Figure 8.9. Two animation sequences of an elongated bar. Top row: The elongated bar undergoes deformations based on the infinitesimal strain tensor. The lack of nonlinear terms causes large distortions and ghost forces in the shape. Bottom row: By filtering high frequencies from the forces based on the finite strain tensor, a larger timestep and nonlinear deformation behavior can be achieved.

8.7 Conclusions and Discussion

In this chapter, we provide a general framework for nonlinear shape optimization. We show applications of our method for optimizing ill-conditioned, nonlinear energies in a variety of different contexts, in each case demonstrating an improvement in the quality of the results and a reduction in the time required to compute a solution (see Table 8.1). We also provide an extension of our method to the case of simulating the dynamics of a physical system, which enables the use of larger timesteps during the course of the simulation. We believe that for many cases in which practitioners resort to approximations, sacrificing quality for speed, our framework can be used to obtain superior results at comparable runtimes.

Our method is conceptually similar to a multigrid method in that it can be used to smooth errors at multiple scales. However, there is an important limitation of our method: a coarser level in a multigrid approach usually requires less computation, whereas in our framework, smoothing errors at larger scales increases the requisite

Table 8.1. Performance of the various optimizations of Willmore (W.), Mehlum-Tarrou (M.T.) and Elastic Potential (E.P.) energy, peformend on triangles (T.), B-spline surfaces (B.S.) and B-spline volumes (B.V.). Timings were taken on a quadcore machine. Datasets denoted with $^{(*)}$ are shown in the supplementary video. In each case, optimization is stopped when shape reaches an acceptable state.

Dataset	E.	Rep.	# of El.	Basis	Time
Dragon	W.	T.	153K	$C^{(0)}$	4h
Disk	W.	T.	720	$C^{(0)}$	1m
Cup 1	W.	T.	6K	$C^{(0)}$	3m
Cup 2	W.	T.	26K	$C^{(0)}$	6m
Caesar	Stretch	T.	16K	$C^{(0)}$	1m
Cylinder	W.	B.S.	320	$C^{(2)}$	5m
Cylinder	M. T.	B.S.	320	$C^{(2)}$	30m
Fertility	E. P.	B.V.	3.5K	$C^{(0)}$	1h
Bunny$^{(*)}$	W.	T.	10K	$C^{(0)}$	3m
Buddha$^{(*)}$	W.	T.	30K	$C^{(0)}$	6m
Teardrop$^{(*)}$	W.	T.	3.8K	$C^{(0)}$	5m
Pin$^{(*)}$	W.	T.	3.3K	$C^{(0)}$	5m

computation because of the higher-order Laplacians involved. On the other hand, the implementation of our method does not require a hierarchy of representations during the optimization process and can therefore be easily integrated into existing optimization systems. Therefore, we believe that this framework provides a valuable benchmark that can be used to evaluate the practicality of nonlinear formulations for many problems.

CHAPTER 9

DATA FITTING

The modeling methodologies presented in Chapter 4, 5, and 6 propose strategies to complete the model and produce a single or a collection of structured hexahedral meshes. In the following let us focus on a single $(n_0+1)\times(n_1+1)\times(n_2+1)$ structured hexahedral mesh with vertices $\mathfrak{p}_{i,j,k}$, which is periodic along the first direction. It will be demonstrated later that the method proposed in this chapter also works on a nonperiodic representation. The hexahedral mesh, guaranteed to be consistent (e.g., it is free of self-intersections), has the same tensor-product nature as a trivariate B-spline. Given this representation, a trivariate B-spline has to be generated which approximates or even interpolates the hexahedral mesh. One of the first decisions to make is to choose between an interpolation or an approximation scheme. The criteria include generating a consistent representation, avoiding volumetric oscillations resulting in degeneracies in the trivariate B-spline representation.

The first aspect would imply an interpolation scheme: Since the points of the hexahedral mesh lie on the resulting B-spline, the error on these points is zero. However, interpolation can cause oscillations and there are no guarantees that the B-spline is consistent. Figure 9.1 shows this effect on a real dataset, where fitted curves are used to generate tubes by sweeping a disk along the curve using the Frenet frame. The following proposed approximation method yields more regular curves, resulting in more regular tubes. Since the initial hexahedral mesh can have a very high resolution, solving a global interpolation problem requires additional extensive computation time. Furthermore, the input triangle meshes were eventually acquired through segmentation of volumetric image data, they approximate the original data already, especially after

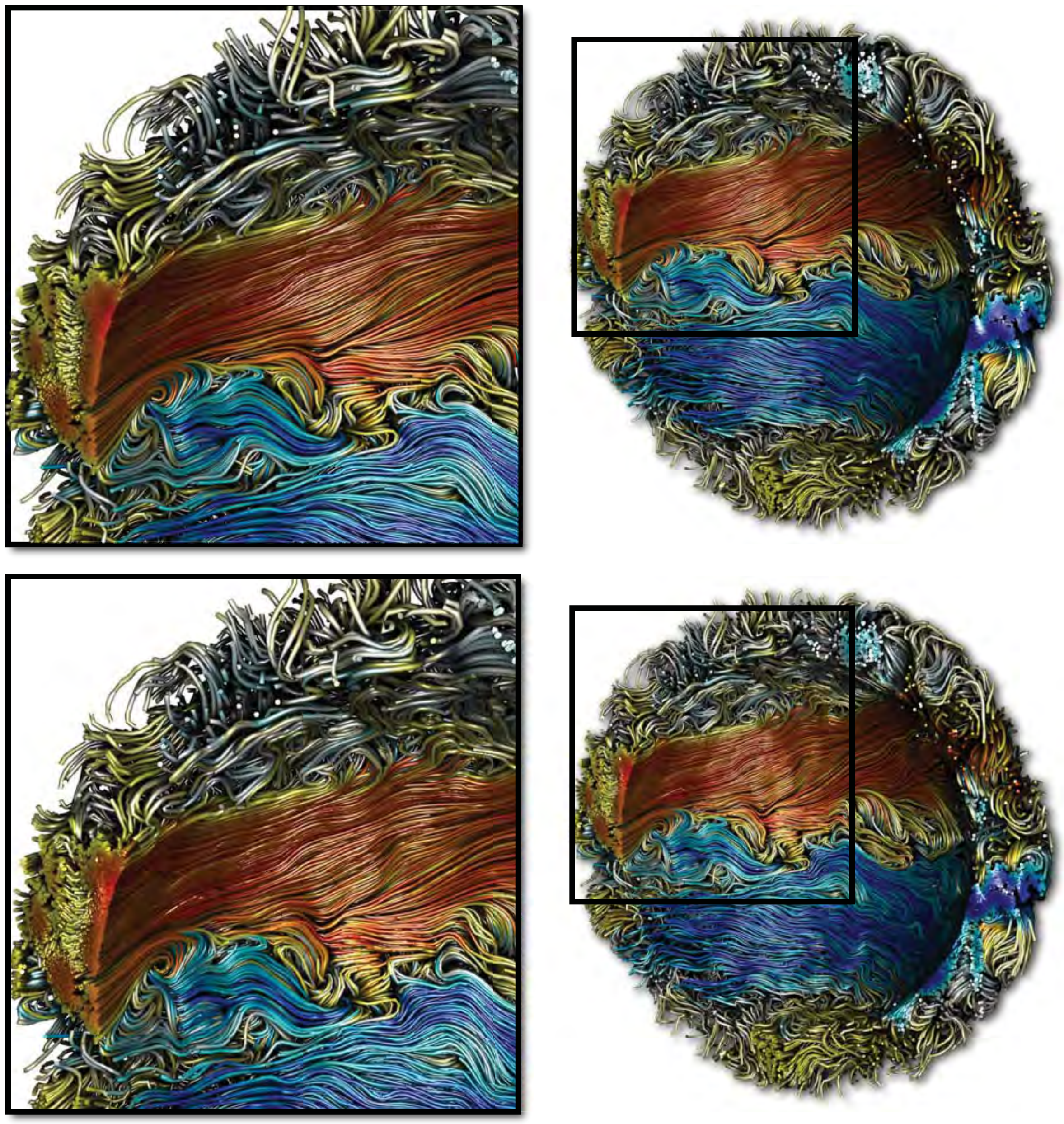


Figure 9.1. The approximation method proposed in this work constructs more regular curves without introducing additional features. The Frenet frames of the resulting tubes are more well-behaved (top). In comparison, the tubes computed based on a classic interpolation scheme (bottom) makes the tubes look more noisy and irregular.

a triangle remesh. Interpolation of such an approximation would not necessarily make sense. Therefore, the second aspect implies an approximation scheme that also avoids wrinkles in the final mesh. Then, it has to be decided, what approximation error is appropriate. This depends on the hexahedral mesh. Sheets which are bent need an adequate number of control points so that the intersection among adjacent sheets is avoided. The choice of an appropriate number of control points is difficult to determine.

We therefore adopt an approach which is a hybrid of both, maximizing advantages of both and minimizing their drawbacks. We allow the user to control how close the B-spline is to the approximating points of the hexahedral mesh. A consistent B-spline with as few oscillations as possible is desirable. Our solution is to develop an approximation iteratively. Figure 9.2 illustrates the proposed iterative approximation behavior on a curve example.

For the volumetric case, the hexahedral mesh is chosen as the initial control mesh. This guarantees that the B-Spline volume lies inside the control volume and that no further features are introduced to the B-spline volume. Furthermore, we set degrees in the three directions $p_0 = 3$, $p_1 = 3$ and $p_2 = 1$, and use a uniform open knot vector in $\mathbf{u}$ and $\mathbf{w}$, and a uniform periodic knot vector in $\mathbf{v}$.

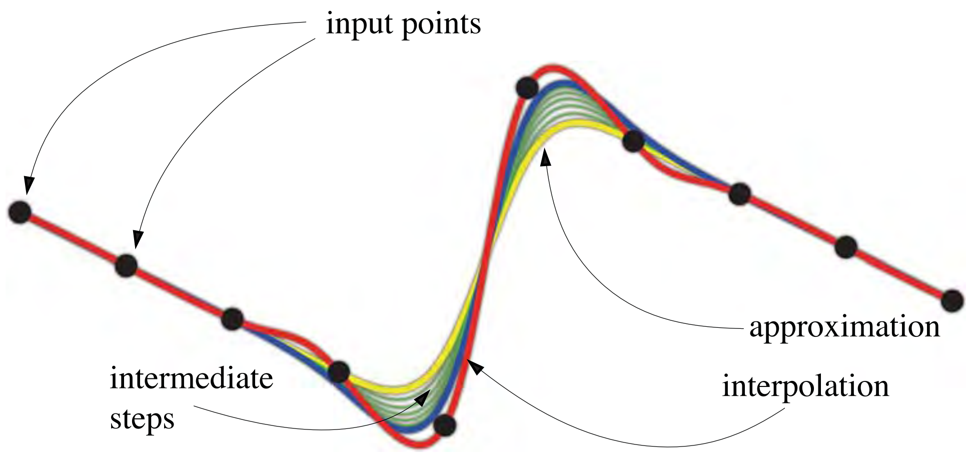


Figure 9.2. Iterative improvement of intput curve (yellow) to the final curve, approximating the input points (black). Intermediate curves are drawn in green. The red curve interpolates the input points.

9.1 Preliminaries

For a fixed k_0, let P^c_{i,j,k_0} be the cth control mesh in an iterative relaxation procedure, where $S^c_{k_0}(u,v)$ is the B-spline surface at iteration c with control mesh P^c_{i,j,k_0}. At $c = 0$, set $P^0_{i,j,k_0} := \mathfrak{p}_{i,j,k_0}$. In the cth iteration, where $c > 0$, we update

$$P^{c+1}_{i,j,k_0} = P^c_{i,j,k_0} + \lambda\Delta^{[c]}, \tag{9.1}$$

where $\Delta^{[c]}$ is a direction vector and is chosen such that $S^c_{k_0}(u,v)$ grows towards $\mathfrak{p}_{i,j,k_0}$. $\lambda \in (0,1)$ is a user-defined scalar, in our case $\lambda = 0.5$.

$\Delta^{[c]}$ is defined in terms of $\mathfrak{p}_{i,j,k_0}$ and $S^c_{k_0}(u^*,v^*)$ corresponding to the control point P^c_{i,j,k_0}. u^* and v^* can be determined by projecting the control point P^c_{i,j,k_0} onto $S^c_{k_0}$. A first-order approximation to this projection is to evaluate $S^c_{k_0}$ at the appropriate node [37], i.e., $u^*_i = \sum_{\mu=1}^{3} t^0_{i+\mu}/3$ and $v^*_j = \sum_{\mu=1}^{3} t^1_{j+\mu}/3$. Since T^1 is uniform periodic, $v^*_j = t^v_{j+p^1-1}$, where v^* in that case is also exact and corresponds to the jth control point. This is not true for the uniform open knot vectors T^0. Either $t^u_{i+2} \le u^*_i \le t^u_{i+3}$ (Case 1) or $t^u_{i+1} \le u^*_i \le t^u_{i+2}$ (Case 2), therefore $S^c_{k_0}(u^*_i, v^*_i)$ lies only near P^c_{i,j,k_0}. If Case 1 applies, then let $i_0 = i-1$, otherwise for Case 2, let $i_0 = i-2$. Then, $S^c_{k_0}(u^*_i, v^*_j) = \sum_{k=0}^{p} (B_{i_0+k,p_0}(u^*_i) C_{i_0+k,j})$, where $C_{k,j} = (P^c_{k,j-1,k_0} + P^c_{k,j+1,k_0})/6 + (2P^c_{k,j,k_0})/3$. Note that, $B_{j-1,q}(v^*_j) = B_{j+1,q}(v^*_j) = 1/6$ and $B_{j,q}(v^*_j) = 2/3$.

In order to define $\Delta^{[c]}$, we ask how to change the current control point P^c_{i,j,k_0} such that $S^{c+1}_{k_0}(u^*_i, v^*_j)$ moves closer to $\mathfrak{p}_{i,j,k_0}$. To answer this, we set

$$\overline{S^c_{k_0}(u^*_i, v^*_j)} = S^c_{k_0}(u^*_i, v^*_j) - \frac{2B_{i,p}(u^*_i)}{3} P^c_{i,j,k_0},$$

and rewrite

$$\mathfrak{p}_{i,j,k_0} = \overline{S^c_{k_0}(u^*_i, v^*_j)} + \frac{2B_{i,p}(u^*_i)}{3}(P^c_{i,j,k_0} + \Delta^{[c]}) \tag{9.2}$$

$$\Delta^{[c]} = \frac{3}{2B_{i,p}(u^*_i)}(\mathfrak{p}_{i,j,k_0} - S^c_{k_0}(u^*_i, v^*_j)). \tag{9.3}$$

The iteration stops when $\epsilon_{max} < \epsilon$, where $\epsilon_{max} = \max ||\mathfrak{p}_{i,j,k} - S^c_{k_0}(u^*_i, v^*_j)||_2$ for every $S^c_{k_0}$. $||.||_2$ is the second vector norm and ϵ is user-defined. The resulting surfaces $S^c_{k_0}$ define the final trivariate B-spline control mesh.

The user choice of λ affects quality and running time of our proposed approximation method. Choosing a λ closer to one reduces the number of iterations but lowers the quality of the final solution. A λ closer to zero requires more iterations but results in a higher quality mesh. Please refer to Figure 9.3 which shows the results for $\lambda = 0.1$ and $\lambda = 0.8$. For $\lambda = 0.1$, 12 iterations were required; for $\lambda = 0.8$ the algorithm terminated after three iterations. In both cases, $\epsilon = 0.01$. For $\lambda = 0.8$, it can be observed that the resulting surface contains unpleasant wrinkles, as they typically appear in interpolation schemes.

9.2 Convergence

The proposed method converges, when at every step the magnitude of $\Delta_i^{[c]}$ gets smaller, and in the limit

$$\lim_{c\to\infty} ||\Delta_i^{[c]}||_2 = 0.$$

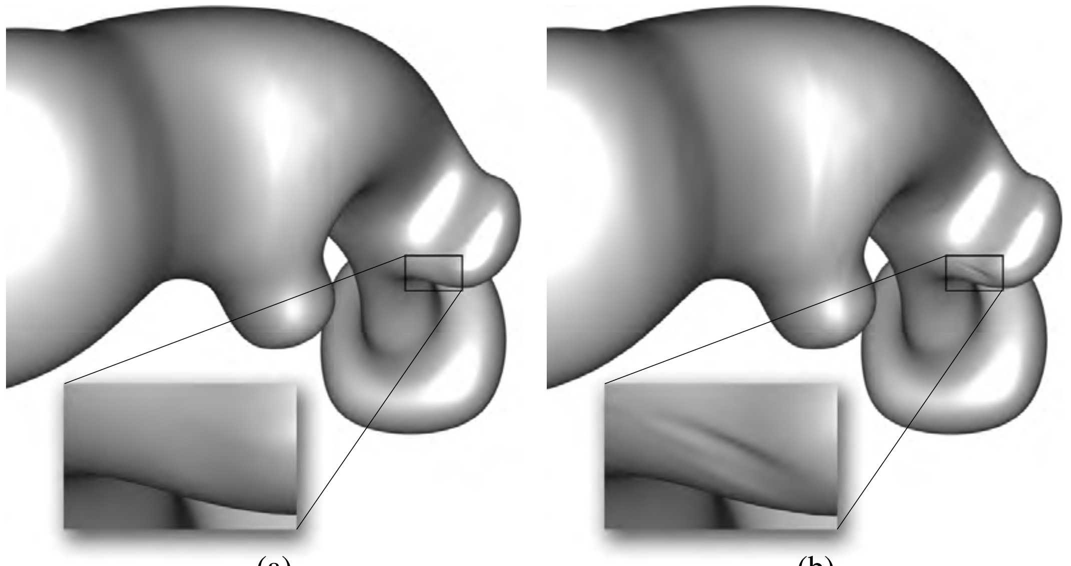


Figure 9.3. Different choices of λ achieve different qualities of approximations. For (a) $\lambda = 0.1$ and for (b) $\lambda = 0.8$ was used. In both cases $\epsilon = 0.01$.

Let us consider the 2D case with a uniform periodic knot vector T. The points $\mathfrak{p}_i$, where $i = 0, \dots, n-1$, define a closed polyline. As above, the initial vertices which define the control polygon are $P_i^{[0]} := \mathfrak{p}_i$. We want to iteratively move the B-spline curve defined by $\{P_i^{[c]}\}$ and T closer to the initial data points $\{\mathfrak{p}_i\}$, where

$$P_i^{[c+1]} = P_i^{[c]} + \lambda \Delta_i^{[c]}.$$

Due to the periodic and uniform knot vector T,

$$\begin{aligned} \mathfrak{p}_i &= \frac{1}{6}\left(P_{i-1}^{[c]} + P_{i+1}^{[c]}\right) + \frac{2}{3}\left(P_i^{[c+1]}\right) \\ &= \frac{1}{6}\left(P_{i-1}^{[c]} + P_{i+1}^{[c]}\right) + \frac{2}{3}\left(P_i^{[c]} + \Delta_i^{[c]}\right). \end{aligned}$$

Solving for $\Delta_i^{[c]}$ yields,

$$\Delta_i^{[c]} = \frac{3}{2}\left(\mathfrak{p}_i - \left(\frac{1}{6}\left(P_{i-1}^{[c]} + P_{i+1}^{[c]}\right) + \frac{2}{3}P_i^{[c]}\right)\right). \tag{9.4}$$

In matrix notation, Equation 9.4 can be rewritten as

$$\vec{\Delta}^{[c]} = \frac{3}{2}\left(\mathfrak{p} - \mathbf{C} \cdot \vec{P}^{[c]}\right), \tag{9.5}$$

where "$\cdot$" is the matrix-vector product. $\vec{\Delta^{[c]}}$ and $\vec{P^{[c]}}$ are vectors with n components, where the ith component is $\Delta_i^{[c]}$ and $P_i^{[c]}$, respectively, and $\mathbf{C}$ is a $n \times n$ circulant matrix [43], where row i is a circular shift of i components of the n-component row vector $[\frac{2}{3}, \frac{1}{6}, 0, \dots, 0, \frac{1}{6}]$, in short

$$\mathbf{C} = circ\left(\frac{2}{3}, \frac{1}{6}, 0, \dots, 0, \frac{1}{6}\right).$$

Note, that if $c = 0$, then

$$\vec{\Delta^{[0]}} = \frac{3}{2}\left(\vec{\mathfrak{p}} - \mathbf{C}\cdot \vec{P^{[0]}}\right) = \frac{3}{2}\left(\vec{\mathfrak{p}} - \mathbf{C}\cdot \vec{\mathfrak{p}}\right), = \frac{3}{2}\left(\mathbf{I} - \mathbf{C}\right)\cdot \vec{\mathfrak{p}},$$

where $\mathbf{I}$ is the identity matrix.

If $c = 1$, then

$$\vec{\Delta^{[1]}} = \frac{3}{2}\left(\vec{\mathfrak{p}} - \mathbf{C}\cdot \vec{P^{[1]}}\right) = \left(I - \frac{3}{2}\lambda\,\mathbf{C}\right)\cdot \vec{\Delta^{[0]}}.$$

From that, induction is used to show that

$$\vec{\Delta^{[c+1]}} = \left(\mathbf{I} - \frac{3}{2}\lambda\,\mathbf{C}\right)\cdot \vec{\Delta^{[c]}} = \mathbf{A}^{c+1}\cdot \frac{3}{2}\left(\mathbf{I} - \mathbf{C}\right)\vec{\mathfrak{p}}, \tag{9.6}$$

where

$$\mathbf{A} = \mathbf{I} - \frac{3}{2}\lambda\,\mathbf{C} = circ\left(1-\lambda, -\frac{\lambda}{4}, 0, \ldots, 0, -\frac{\lambda}{4}\right).$$

The magnitude of $\Delta^{[c]}$ converges against 0, implying that our fitting procedure converges, if

$$\lim_{c\to\infty} \mathbf{A}^c = \mathbf{Z},$$

where $\mathbf{Z}$ is the zero matrix. This implies, according to [43], that the eigenvalues of $\mathbf{A}$ are, $|\lambda_r| < 1$, $r = 0, 1, \ldots, n-1$. Since $\mathbf{A}$ is a circulant matrix, it is diagonalizable by $\mathbf{A} = \mathbf{F}^* \cdot \mathbf{\Lambda} \cdot \mathbf{F}$, where $\mathbf{\Lambda}$ is a diagonal matrix, whose diagonal elements are the eigenvalues of $\mathbf{A}$, and $\mathbf{F}^*$ is the Fourier matrix with entries given by

$$\mathbf{F}^*_{jk} = \frac{1}{\sqrt{n}}\cdot e^{\frac{2\pi ijk}{n}}.$$

$\mathbf{F}^*$ is the conjugate transpose of $\mathbf{F}$. Due to the fact, that $\mathbf{A}$ is a circulant matrix, its eigenvalue vector $\vec{v}$ can be computed by $\sqrt{n} \cdot \mathbf{F}^* \cdot \vec{v}_{\mathbf{A}} = \vec{v}$, where

$$\vec{v}_{\mathbf{A}} = [1-\lambda, -\lambda/4, 0, \ldots, 0, -\lambda/4].$$

Applying that to our case, we get

$$\lambda_r = (1-\lambda) - \frac{\lambda}{4}\left(\cos r\frac{2\pi}{n} + \cos r(n-1)\frac{2\pi}{n}\right),$$

so $1 - \frac{3}{2}\lambda \leq \lambda_r \leq 1 - \frac{\lambda}{2}$. Hence, in every step the magnitude of $\Delta_i^{[c]}$ decreases which results in the convergence of our proposed data fitting approach for uniform periodic B-spline curves.

In the case when T is uniform open, equation 9.6 can be rewritten by $\vec{\Delta^{[c+1]}} = \mathbf{A}^{c+1} \cdot \mathbf{S} \cdot (\mathbf{I} - \mathbf{C}) \cdot \vec{\mathfrak{p}}$, where $\mathbf{A} = (\mathbf{I} - \lambda\,\mathbf{S} \cdot \mathbf{C})$. $\mathbf{S}$ is a diagonal matrix where the diagonal elements are defined by

$$\mathbf{S}_{ii} = \frac{1}{B_{i,p}(u_i^*)}$$

and $\mathbf{C}_{ij} = B_{j,p}(u_i^*)$ which is not circulant. Therefore, the bound on the eigenvalues given above does not apply, due to the end conditions. However, we conducted experiments with different values for λ, and the maximum eigenvalue is always less than one and stays the same, independent of the problem size n, indicating convergence. For $\lambda = 1/2$, the eigenvalues of $\mathbf{A}$ range from 0.15 to 0.85.

In the surface case, the two curve methods are interleaved as is done for tensor product nodal interpolation. It is guaranteed to converge given the curve method properties.

9.3 Simplification

The resulting trivariate B-spline tends to have a high resolution. Therefore, as a postprocessing step we apply a data reduction algorithm [120] to the B-spline representation and iteratively decide how to reduce complexity on the surface or on the attribute data in the interior, by minimizing error.

CHAPTER 10

DIRECT ISOSURFACE VISUALIZATION

In this chapter, we present a novel isosurface visualization technique that guarantees the accuarate visualization of isosurfaces with complex attribute data defined on (un-)structured (curvi-)linear hexahedral grids. Isosurfaces of high-order hexahedral-based finite element solutions on both uniform grids (including MRI and CT scans) and more complex geometry representing a domain of interest can be rendered using our algorithm. Additionally, our technique can be used to directly visualize solutions and attributes in IA, an area based on trivariate high-order NURBS (Non-Uniform Rational B-splines) geometry and attribute representations for the analysis. Furthermore, our technique can be used to visualize isosurfaces of algebraic functions. Our approach combines subdivision and numerical root-finding to form a robust and efficient isosurface visualization algorithm that does not miss surface features, while finding all intersections between a view frustum and desired isosurfaces. This allows the use of view-independent transparency in the rendering process. We demonstrate our technique through a straightforward CPU implementation on both complex-structured and complex-unstructured geometry with high-order simulation solutions, isosurfaces of medical data sets, and isosurfaces of algebraic functions.

The demand for isosurface visualization techniques arises in many fields within science and engineering. For example, it may be necessary to visualize isosurfaces of data from CT or MRI scans on structured grids or numerical simulation solutions generated over approximated geometric representations, such as deformed curvilinear high-order (un-)structured grids representing an object of interest. In this context, high-order means that polynomials with degree > 1 are used as the basis to represent either the geometry or the solution of a Partial Differential Equation (PDE). High-order data is the set of coefficients for these solutions.

Given one of these representations, a visualization technique such as the Marching Cube technique [119], direct isosurface visualization [155], or surface reconstruction applied to a sampling of the isosurface, is frequently used to extract the isosurface. However, given high-order data representations, we seek visualization algorithms that act natively on different representations of the data with quantifiable error.

In this chapter, we present a novel and robust ray frustum-based direct isosurface visualization algorithm. The method is exact to pixel accuracy, a guarantee which is formally shown, and it can be applied to complex attribute data embedded in complex geometry. In particular, the method can be applied to the following representations:

1. Structured hexahedral (hex) geometry grids with discrete data (e.g., CT or MRI scans). The proposed method filters the discrete data with a interpolating or approximating high-order B-spline filter [121] to create a high-order representation of the function that was sampled by the grid.

2. Structured hex-based representations with high-order attribute data, where the geometry can be represented using trilinear or higher order basis.

3. Structured and unstructured hex meshes, each of which element's shape may be deformed by a mapping (curvilinear shape elements) and with simulation data (higher polynomial order).

4. Algebraic functions. The representation is exact.

We demonstrate that our method is up to three times faster and requires fewer subdivisions and therefore less memory than related techniques on related problems.

An added motivation to this work is the fact that trivariate NURBS [37] have been proposed for use in Isogeometric Analysis (IA) [87] to represent both geometry and simulation solutions ([87, 40, 209]). Simulation parameters are specified through attribute data, and the analysis result is represented in a trivariate NURBS representation linked to the shape representation. This is the first algorithm that can produce accurate visualizations of isogeometric analysis results.

With degree > 1 in each parametric direction and varying Jacobians (i.e., nonlinear mappings), trivariate NURBS that represent an object of interest (see Figure 10.1) have no closed-form inverse. Existing visualization methods designed to work efficiently on regular spatial grids have not been extended to work robustly and efficiently and preserve smoothness on these complex and high-order geometries. Furthermore, standard approaches for direct visualization are ray-based and assume single entry and exit points of a ray with an element. That hypothesis is no longer true for curvilinear elements. Hence, those approaches are difficult to extend to arbitrary complex geometry with curvilinear elements. Note that finding the complete collection of entry and exit points into curvilinear elements is a nontrivial task.

In practice, representations of more complex geometry on which numerical simulation techniques are applied often contain geometric degeneracies resulting from either mesh generation or the data-fitting process. For instance, poorly-shaped elements can lead to a Jacobian with a determinant close to zero, which presents challenges during simulations. In addition, and more importantly for this chapter, it presents a challenge in visualizing isosurfaces of the high-order simulation solution. Thus, there is a need for isosurface visualization techniques that deal robustly with both degenerate and near-degenerate geometry.

After discussing mathematical framework in Section 10.1, we define our mathematical formulation by stating the visualization problem in Section 10.2, which is solved in Section 10.3. Implementation details are given in Section 10.4, and sections 10.5) and 10.6) analyse the results of our technique, followed by a conclusion.

10.1 Background

The following sections introduce the mathematical framework for the mathematical formulation in this chapter.

10.1.1 Trivariate NURBS

A trivariate tensor product NURBS mapping is a parametric map $\mathcal{V}\colon [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2] \to \Omega \subset \mathbb{R}^3$ of degree $\mathbf{d} = (d_1, d_2, d_3)$ with knot vectors $\tau = (\tau_1, \tau_2, \tau_3)$, defined as

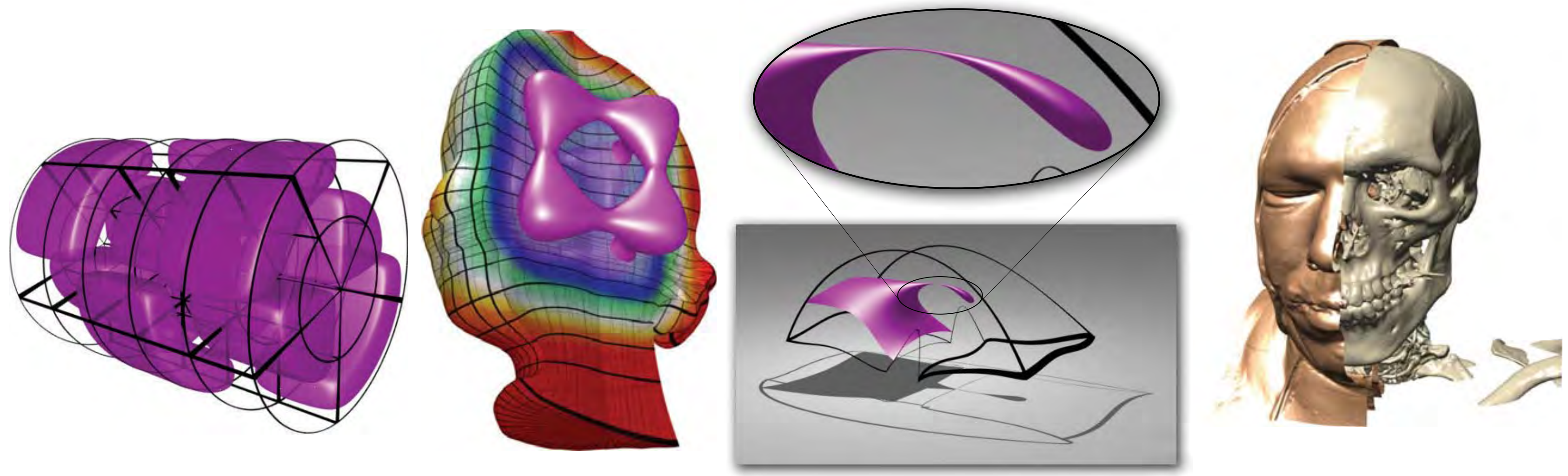

Isosurfaces of Rational Geometry
(a)
Isosurfaces of Data Fitted Geometry
(b)
Isosurfaces of Algebraic Functions
(c)
Isosurfaces of CT/MRI Scans
(d)

Figure 10.1. Our method applied to four representative isosurface visualizations. (a) Vibration modes of a solid structure; (b) Solution to Poisson Equation; (c) Teardrop under nonlinear deformation; (d) Two Isosurfaces of the visible human data set

$$\mathcal{V}(\mathbf{u}) := \frac{\sum_{\mathbf{i}=1}^{\mathbf{n}} w_{\mathbf{i}}\, \mathbf{c}_{\mathbf{i}}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})}{\sum_{\mathbf{i}=1}^{\mathbf{n}} w_{\mathbf{i}}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})} \tag{10.1}$$

$$= \left(\frac{x(\mathbf{u})}{w(\mathbf{u})}, \frac{y(\mathbf{u})}{w(\mathbf{u})}, \frac{z(\mathbf{u})}{w(\mathbf{u})}\right), \tag{10.2}$$

where $\mathbf{c}_{\mathbf{i}} \in \mathbb{R}^3$ are the control points with associated weights $w_{\mathbf{i}}$ of the $n_1 \times n_2 \times n_3$ control grid, $\mathbf{i} = (i_1, i_2, i_3)$ is a multi-index, and $\mathbf{u} = (u_1, u_2, u_3)$ is a trivariate parameter value. Every coefficient $\mathbf{c}_{\mathbf{i}}$ has an associated trivariate B-spline basis function $\mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u}) = \prod_{j=1}^{3} B_{i_j,d_j,\tau_j}(u_j)$.

$B_{i_j,d_j,\tau_j}(u_j)$ are linearly independent piecewise polynomials of degree d_j with knot vector $\tau_j = \{t_k^j\}_{k=1}^{n_j+d_j}$. They have local support and are $C^{(d_i-1)}$. Furthermore, $\sum_{\mathbf{i}=1}^{\mathbf{n}} \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u}) = 1$ (see [37]). Figure 10.2 illustrates these definitions for the 1D case.

$\mathbf{c}_{\mathbf{i}} \in \mathbb{R}^3$, $\mathcal{V}(\mathbf{u})$ describes the physical geometry and is referred to as the *geometric mapping*. Suppose an attribute $\mathcal{A}(\mathbf{u})$ is related to $\mathcal{V}(\mathbf{u})$ where the attribute function $\mathcal{A}\colon [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2] \to \mathbb{R}^{(k)}$ can be formulated as

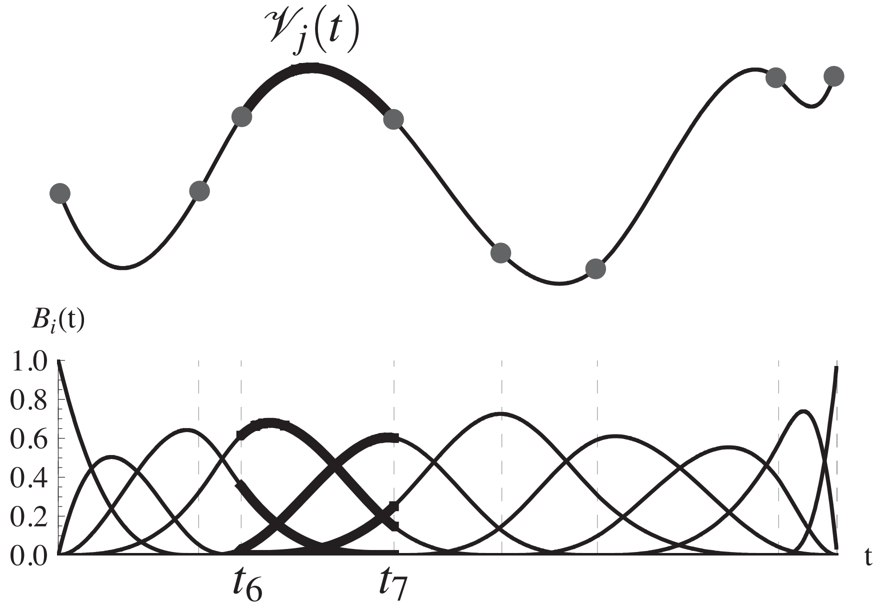


Figure 10.2. Cubic NURBS curve with nonuniform knot vector and open end conditions.

$$\mathcal{A}(\mathbf{u}) := \frac{\sum_{\mathbf{i}=1}^{\mathbf{n}} w_{\mathbf{i}}\, a_{\mathbf{i}}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})}{\sum_{\mathbf{i}=1}^{\mathbf{n}} w_{\mathbf{i}}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})} \tag{10.3}$$

$$= \frac{a(\mathbf{u})}{w(\mathbf{u})}. \tag{10.4}$$

where $\mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})$ is defined as above.

Let $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ and $\mathcal{A}_{\mathbf{i}}(\mathbf{u})$ refer to the geometry and attribute mapping of the **i**th knot span, $\mathbf{i} = (i_1, i_2, i_3)$, called a "patch," i.e., its parametric domain is $[t^1_{i_1}, t^1_{i_1+1}) \times [t^2_{i_2}, t^2_{i_2+1}) \times [t^3_{i_3}, t^3_{i_3+1})$, where

$$\mathcal{V}_i(\mathbf{u}) := \left(\frac{x_{\mathbf{i}}(\mathbf{u})}{w_{\mathbf{i}}(\mathbf{u})}, \frac{y_{\mathbf{i}}(\mathbf{u})}{w_{\mathbf{i}}(\mathbf{u})}, \frac{z_{\mathbf{i}}(\mathbf{u})}{w_{\mathbf{i}}(\mathbf{u})} \right), \mathcal{A}_i(\mathbf{u}) := \frac{a_{\mathbf{i}}(\mathbf{u})}{w_{\mathbf{i}}(\mathbf{u})}. \tag{10.5}$$

For the purpose of clarity, we consider only scalar attributes, although this approach works equally well for vector attributes. $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ and $\mathcal{A}_{\mathbf{i}}(\mathbf{u})$ are each a single trivariate tensor product polynomial (or rational), and $\mathbb{G} := \{(\mathcal{V}_{\mathbf{i}}(\mathbf{u}), \mathcal{A}_{\mathbf{i}}(\mathbf{u}))\}_{\mathbf{i}}^{\mathbf{n}-\mathbf{d}}$ is the set of geometry and attribute patches, respectively. Note that each geometry patch $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ has a corresponding attribute patch $\mathcal{A}_{\mathbf{i}}(\mathbf{u})$. Furthermore, in case Ω cannot be represented using a single mapping $\mathcal{V}(\mathbf{u})$, then Ω is represented as a collection of the mappings $\mathcal{V}(\mathbf{u})$ and $\mathcal{A}(\mathbf{u})$.

Figure 10.3 illustrates these definitions with a single NURBS surface representing $\Omega \in \mathbb{R}^2$.

10.1.2 Classical Problem Statement

Let $\Omega \in \mathbb{R}^3$ be the domain of interest and $g(x, y, z)$ where $g : \Omega \to \mathbb{R}$ is an attribute function. In isosurface visualization, the user specifies an isovalue $\hat{a}$ at which to inspect the implicit isosurface of $g(x, y, z) - \hat{a} = 0$. By referring to Figure 10.3 (showing the 2D scenario), in ray-based visualization techniques, the ray, passing through the center of a pixel, is represented as $\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}$, where $\mathbf{o}$ is the origin of the ray (location of the eye) in $\mathbb{R}^3$, $\mathbf{d}$ the direction of the ray, and $t \in \mathbb{R}$ the ray parameter. One wants to find the set of t-values that satisfy $f(t) = 0$, where $f(t) = g(\mathbf{r}(t)) - \hat{a}$.

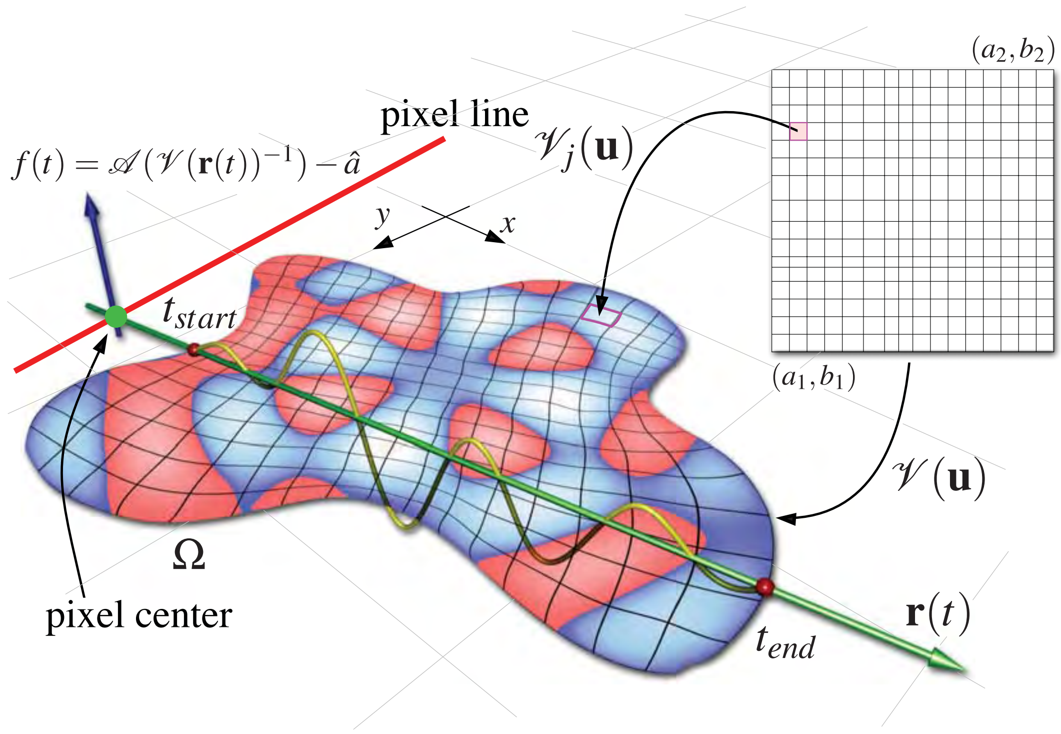

(a_2, b_2)
pixel line
$f(t) = \mathscr{A}(\mathscr{V}(\mathbf{r}(t))^{-1}) - \hat{a}$
$\mathscr{V}_j(\mathbf{u})$
y
x
t_{start}
(a_1, b_1)
$\mathscr{V}(\mathbf{u})$
Ω
pixel center
$\mathbf{r}(t)$
t_{end}

Figure 10.3. 2D analogy: Ray passing through a bivariate NURBS surface with color-coded attribute field $\mathcal{A}(\mathbf{u})$ intersecting isocontour at roots of $f(t)$, where the red points refer to entry and exit points with the surface.

When Ω represents a uniform scalar grid, efficient and interactive methods exist to directly visualize isosurfaces, including a GPU approach to visualize trivariate splines with respect to tetrahedral partitions that transform each patch to its Bernstein-Bézier form [95]. Earlier, a direct rendering paradigm of trivariate B-spline functions for large data sets with interactive rates was presented in the work by [164], where the rendering is conducted from a fixed viewpoint in two phases suitable for sculpting operations. Entezari et al. [56] derive piecewise linear and piecewise cubic box spline reconstruction filters for data sampled on the body-centered cubic lattice. Given such a representation, they directly visualize isosurfaces. Similarly, Kim et al. [101] introduce a box spline approach on the face-centered cubic (FCC) lattice and propose a reconstruction algorithm that can interpolate or approximate the underlying function based on the FCC and directly visualize isosurfaces.

In the case where $g(x, y, z)$ describes an algebraic function in $\mathbb{R}^3$, Blinn [22] uses a hybrid combination of univariate Newton-Raphson iteration and regular falsi. More recently, Reimers et al. [166] developed an algorithm to visualize algebraic surfaces of high degree, using a polynomial form that yields interactive frame rates on the GPU. Toledo et al. [44] present GPU approaches to visualize algebraic surfaces on the GPU. Interval analysis ([134]) has been adopted by Hart [77] and recently by Knoll et al. [105] to visualize isosurfaces of algebraic functions as well.

In the following discussion, let $\mathcal{V}(\mathbf{u})$ represent a general domain of interest Ω together with an attribute field $\mathcal{A}(\mathbf{u})$. In this case, Ω is not a cube which has undergone no or at most an affine transformation. Therefore, $g(x, y, z) := \mathcal{A}(\mathcal{V}^{-1}(x, y, z))$. $\mathcal{V}^{-1}(x, y, z)$ is the inverse of a nonidentity and nonaffine mapping, i.e., it cannot be represented in closed-form and in order to evaluate the corresponding $f(t)$, the inverse of $\mathcal{V}^{-1}(x, y, z)$ has to be computed using a root-solving method. Because of this, it is not clear how these methods can be extended to work with the nonlinear, nonpolynomial mapping $\mathcal{V}^{-1}(x, y, z)$. Computing all the roots along $\mathbf{r}(t)$ with those methods would involve re-application of the respective visualization algorithm, making extensions of such approaches computationally intractable.

Before any root-solving takes place, the set $\mathbb{I} \subset \mathbb{G}$ is computed where the geometric sub patches $\mathcal{V}_{\mathbf{i}}(\mathbf{u}) \in \mathbb{I}$ might get intersected by $\mathbf{r}(t)$ and contain the respective isosurface. Finding the roots of $f(t)$ is equivalent to finding the roots of $f_{\mathbf{i}}(t)$ of the geometry patches $\mathcal{V}_{\mathbf{i}}(\mathbf{u}) \in \mathbb{I}$, where

$$f_{\mathbf{i}}(t) := \mathcal{A}_{\mathbf{i}}(\mathcal{V}_{\mathbf{i}}^{-1}(\mathbf{r}(t))) - \hat{a} = 0. \tag{10.6}$$

Solving Equation 10.6 requires finding the range of values of t where $f_{\mathbf{i}}(t)$ is defined, i.e., the t-values which correspond to the entry and exit points of $\mathbf{r}(t)$ into $\mathcal{V}_{\mathbf{i}}^{-1}(\mathbf{r}(t))$. Depending on the geometric complexity of Ω, this range can consist of multiple disjoint intervals where each interval is defined by an entry and exit point of the ray with $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$.

One way to compute these intervals is to use the Bézier clipping method proposed in the work [145] on the six sides of the elements in $\mathbb{I}$, implying that the elements in $\mathbb{I}$ have to be turned into Bézier patches using knot insertion (see [37]). While Bézier clipping is an elegant way to visualize Bézier surfaces, it has problems at silhouette pixels. A discussion of its problems and proposed solutions can be found in [53]. Once these pairs of entry and exit points are computed, a numerical root-solving technique, such as the Newton-Raphson method or bisection method, is applied to $f_{\mathbf{i}}(t)$ for each pair. The limitations of these classic methods are well-known. That is, Newton's method requires an initial starting value close to the root and depends on $f'_{\mathbf{i}}(t)$, so it fails at degeneracies and where the derivative is close to zero. Krawczyk [109] presents a Newton-Raphson algorithm that uses interval arithmetic for the initial guess. Toth [197] applies this method to render parametric surfaces. However, since Newton's method needs the derivative of $f_{\mathbf{i}}(t)$, it can fail at the edges of $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ as discussed in Abert [2], leading to the well-known *black pixel* artifacts at the patch boundaries, as shown in Figure 10.4. The bisection method is more robust but converges only linearly. The main problem with the bisection method is that the signs of $f_{\mathbf{i}}(t)$ at the entry and exit points must be different, a requirement which often cannot be fulfilled. In summary, an approach which attempts to solve Equation 10.6 can fail when finding the entry and exit points, or finding the inverse $\mathcal{V}_{\mathbf{i}}^{-1}(x, y, z)$, or finding the roots of $f_{\mathbf{i}}(t)$ fails. Furthermore, there

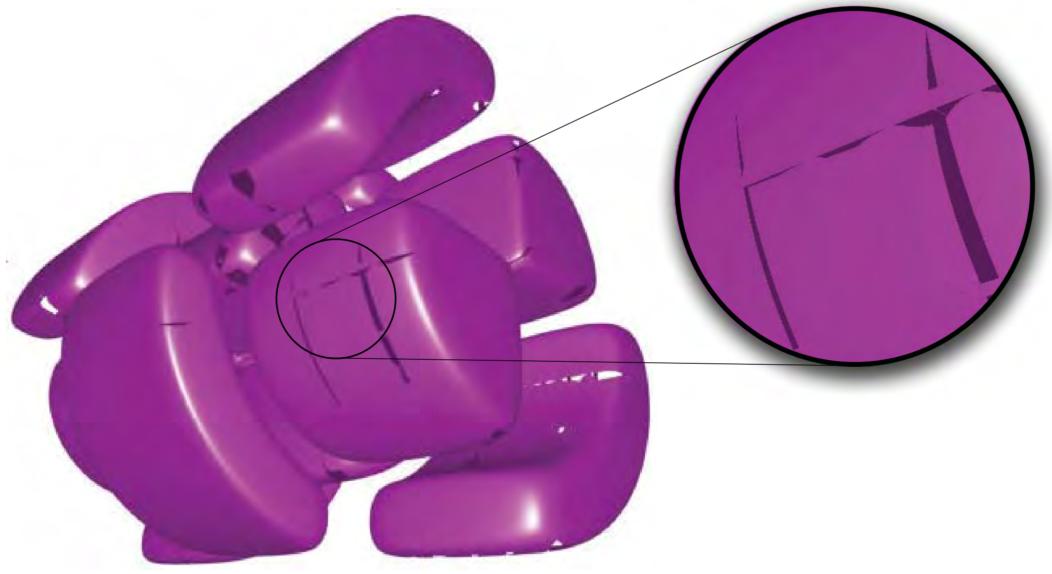


Figure 10.4. Piecewise trivariate cubic Bézier patches results in *black pixel* artifacts, due to degenerate derivative at the Bézier patch edges.

is no guarantee of determining all intersections between the isosurface and the area corresponding to the pixel, i.e., it may only determine the intersections at the ray itself.

Another standard approach to intersect a ray $\mathbf{r}(t)$ with an isosurface, as defined in the work by [185], is to solve the system of four equations and four unknowns:

$$\begin{pmatrix} r_x(t) \\ r_y(t) \\ r_z(t) \\ \mathcal{A}(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} x(\mathbf{u}) \\ y(\mathbf{u}) \\ z(\mathbf{u}) \\ \hat{a} \end{pmatrix},$$

where $r_x(t)$, $r_y(t)$ and $r_z(t)$ are the x-, y- and z- coordinates of $\mathbf{r}(t)$, respectively. Such a nonlinear system can be solved using the general geometric constraint-solving approach proposed by Elber et al. [55] that uses subdivision and higher dimensional Axis-Aligned Bounding Box (AABB) tests to find a solution where $\mathbf{r}(t)$ and $\mathcal{V}(\mathbf{u})$ are piecewise polynomial or piecewise rational. Elber et al. applied their approach to bisectors, ray-traps, sweep envelopes, and regions accessible during 5-axis machining, but not to rendering isosurfaces. However, as we propose here, pixel-exact isosurface visualization requires further augmentation of the algorithm.

In the following approach, we develop a formulation for a guaranteed determination of all intersections between a ray frustum and an isosurface. The proposed method computes the set of roots simultaneously, avoiding any computation of intervals on which $f_{\mathbf{i}}(t)$ is defined.

10.2 Mathematical Formulation

In this section, we develop the mathematical formulation that is used to intersect a ray frustum (Figure 10.5) with the implicit isosurface $\mathcal{A}(\mathbf{u}) - \tilde{a} = 0$ embedded within $\hat{\mathcal{V}}(\mathbf{u})$, which can represent arbitrary geometry. $\tilde{a}$ is the scalar value for which the isosurface will be visualized.

In the following, we assume the coefficients $\mathbf{c}_{\mathbf{i}}$ and the corresponding weights $w_{\mathbf{i}}$, as defined in Section 10.1.1, are in eye space, i.e., the camera frustum sits at the origin, pointing down the negative z-axis. Let $\mathbf{P}$ be the 4×4 projection matrix defining the camera frustum, where

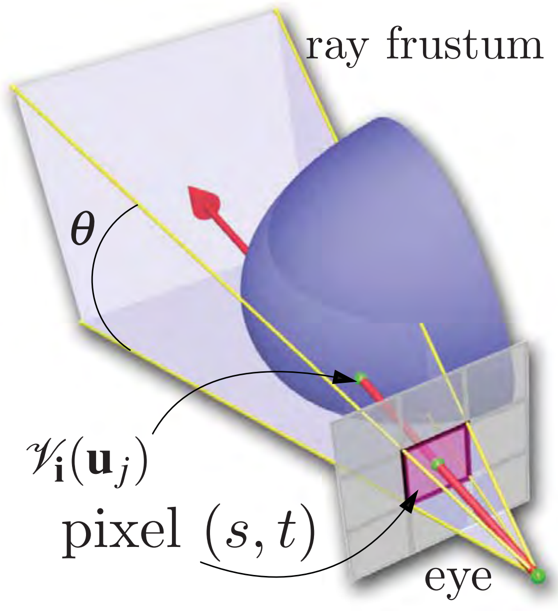


Figure 10.5. Ray Frustum/Isosurface Intersection for pixel (s,t) shaded in magenta with adjacent pixels shaded in grey.

$$\mathbf{P} = \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2\,far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix}. \tag{10.7}$$

In this case, **P** defines a frustum with a near plane of *near* units away from the eye with a size of $[-1,1]\times[-1,1]$, and a far plane of far away from the eye, where $near < far$. Furthermore, **P** projects along the z-axis.

P transforms the frustum and all geometry from eye space into perspective space, i.e., the frustum is transformed into the unit cube $[-1,1]^3$ and every ray frustum in eye space is transformed into a ray box in perspective space. Coefficients $\mathbf{c_i}$ and weights $w_\mathbf{i}$ are transformed into perspective space by

$$(\hat{w}_\mathbf{i}\,\hat{x}_\mathbf{i}, \hat{w}_\mathbf{i}\,\hat{y}_\mathbf{i}, \hat{w}_\mathbf{i}\,\hat{z}_\mathbf{i}, \hat{w}_\mathbf{i})^T = \mathbf{P}\circ(w_\mathbf{i}\,x_\mathbf{i}, w_\mathbf{i}\,y_\mathbf{i}, w_\mathbf{i}\,z_\mathbf{i}, w_\mathbf{i})^T, \tag{10.8}$$

where $\hat{c}_\mathbf{i} = (\hat{x}_\mathbf{i}, \hat{y}_\mathbf{i}, \hat{z}_\mathbf{i})$ and

$$\begin{pmatrix} \hat{x}_{\mathbf{i}} \\ \hat{y}_{\mathbf{i}} \\ \hat{z}_{\mathbf{i}} \\ \hat{w}_{\mathbf{i}} \end{pmatrix} = \begin{pmatrix} (near * x_{\mathbf{i}})/z_{\mathbf{i}} \\ -(near * y_{\mathbf{i}})/z_{\mathbf{i}} \\ \frac{(2*far*near+(far+near)\, z_{\mathbf{i}})}{(far-near)*z_{\mathbf{i}}} \\ -w_{\mathbf{i}}\, z_{\mathbf{i}} \end{pmatrix}. \tag{10.9}$$

From that,

$$\begin{aligned} \hat{\mathcal{V}}(\mathbf{u}) &:= \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} \hat{w}_{\mathbf{i}}\, \hat{\mathbf{c}}_{\mathbf{i}}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})}{\sum_{\mathbf{i=1}}^{\mathbf{n}} \hat{w}_{\mathbf{i}}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})} && (10.10) \\ &= \left(\frac{\hat{x}(\mathbf{u})}{\hat{w}(\mathbf{u})}, \frac{\hat{y}(\mathbf{u})}{\hat{w}(\mathbf{u})}, \frac{\hat{z}(\mathbf{u})}{\hat{w}(\mathbf{u})} \right) && (10.11) \end{aligned}$$

is $\mathcal{V}(\mathbf{u})$ in perspective space. Furthermore, let $\hat{\mathbf{x}} = (\hat{x}, \hat{y}, \hat{z})$ be a point in perspective space. Although the transformed ray frustum, mapped from eye space to perspective space is a rectangular parallelepiped, we still call it a *ray frustum* to evoke its shape in eye space.

Given a ray frustum constructed from ray $\mathbf{r}(t)$ as shown in Figure 10.5, there are three types of intersections between a ray frustum and the isosurface: 1) The isosurface intersects the four planes of the ray frustum and the isosurface's normals point either towards or away from the eye over the whole frustum and $\mathbf{r}(t)$ passes through the isosurface; 2) $\mathbf{r}(t)$ passes through the isosurface but the ray frustum contains an isosurface silhouette; 3) Same as case 2) but the $\mathbf{r}(t)$ does not pass through the isosurface. Figure 10.6 illustrates these three intersection types.

In types 1) and 2), $\mathbf{r}(t)$ intersects the isosurface and can be detected with ray-isosurface intersection. Type 3 requires a different approach. Note that there are cases for which sampling approaches such as pixel subdivision will fail.

First, we present how to detect type 1 and type 2 cases and then discuss how to detect type 3. For an image with resolution $h \times h$ pixels where h is the number of pixels per row and column, we follow the development of Kajiya [94] to detect type 1 and 2 as:

$$x - b_s = 0 \text{ and } y - b_t = 0 \text{ with } b_k = 2(k/h) - 1 + k/(2h), \tag{10.12}$$

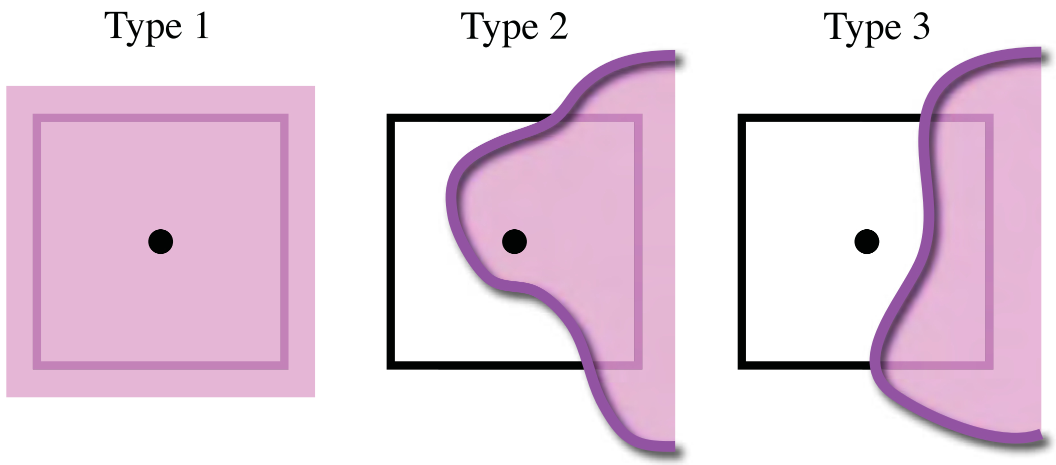


Figure 10.6. Three ray frustum/isosurface intersection types: 1) Ray frustum and corresponding pixel is fully covered; 2) isosurface silhouette intersects ray frustum with ray intersecting isosurface; 3) Same as 2) but ray does not intersect isosurface.

which are two orthogonal planes in perspective space corresponding to pixel at (s, t) whose intersection define a ray $\mathbf{r}(t)$ aligned with the unit cube.

Given pixel (s, t),

$$\begin{pmatrix} \hat{\alpha}(\mathbf{u}) \\ \hat{\beta}(\mathbf{u}) \\ \hat{\gamma}(\mathbf{u}) \end{pmatrix} := \frac{1}{\hat{w}(\mathbf{u})} \begin{pmatrix} \hat{x}(\mathbf{u}) \\ \hat{y}(\mathbf{u}) \\ a(\mathbf{u}) \end{pmatrix} - \begin{pmatrix} b_s \\ b_t \\ \tilde{a} \end{pmatrix} \tag{10.13}$$

rational B-splines. Note, $a(\mathbf{u})$ is defined in Equation 10.4.

The following constraints must be satisfied for a ray/isosurface intersection:

$$\begin{pmatrix} |\hat{\alpha}(\mathbf{u})| \\ |\hat{\beta}(\mathbf{u})| \\ |\hat{\gamma}(\mathbf{u})| \end{pmatrix} < \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \end{pmatrix} \tag{10.14}$$

i.e., given a solution $\mathbf{u}$, the corresponding $\hat{\mathcal{V}}(\mathbf{u})$ must lie along the ray and on the isosurface within tolerance of $\epsilon = 1/(2\,h)$. This ensures that a solution lies within a pixel. Multiplying Equation 10.14 by $\hat{w}(\mathbf{u})$,

$$\begin{pmatrix} |\alpha(\mathbf{u})| \\ |\beta(\mathbf{u})| \\ |\gamma(\mathbf{u})| \end{pmatrix} < \hat{w}(\mathbf{u}) \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \end{pmatrix} \tag{10.15}$$

where $\alpha_{\mathbf{i}} = \hat{x}_{\mathbf{i}} - \hat{w}_{\mathbf{i}}\, b_s$, $\beta_{\mathbf{i}} = \hat{y}_{\mathbf{i}} - \hat{w}_{\mathbf{i}}\, b_t$ and $\gamma_{\mathbf{i}} = a_{\mathbf{i}} - \hat{w}_{\mathbf{i}}\, \tilde{a}$ and $(\alpha(\mathbf{u}), \beta(\mathbf{u}), \gamma(\mathbf{u})) := \sum_{\mathbf{i}=1}^{\mathbf{n}} (\alpha_{\mathbf{i}}, \beta_{\mathbf{i}}, \gamma_{\mathbf{i}})\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})$.

Equation 10.15 is not sufficient to detect every isosurface/ray frustum intersection. If an isosurface silhouette lies within the ray frustum but does not get intersected by $\mathbf{r}(t)$ (type 3), then there is no $\mathbf{u}$ that satisfies Equation 10.15, even though some part of the isosurface (silhouette) lies within the ray frustum. Let

$$\nu(\mathbf{u}) := J_{\hat{\mathbf{x}}}(\mathbf{u}) \cdot \nabla_{\mathbf{u}} \mathcal{A}(\mathbf{u}) = \nabla_{\hat{\mathbf{x}}} \mathcal{A}(\mathbf{u}) \tag{10.16}$$

be the gradient in normal direction of the isosurface at $\mathbf{u}$ in perspective space, where $J_{\hat{\mathbf{x}}}(\mathbf{u})$ is the Jacobian at $\mathbf{u}$ in perspective space, then

$$\begin{aligned} \hat{\delta}(\mathbf{u}) &:= \nu(\mathbf{u})_{\hat{z}} && (10.17) \\ \hat{\eta}(\mathbf{u}) &:= \Big(\Big(\frac{\hat{x}(\mathbf{u})}{\hat{w}(\mathbf{u})}, \frac{\hat{y}(\mathbf{u})}{\hat{w}(\mathbf{u})}, 0 \Big) \times (\nu(\mathbf{u})_x, \nu(\mathbf{u})_y, 0) \Big)_{\hat{z}}, && (10.18) \end{aligned}$$

are rational B-splines, where $\nu(\mathbf{u})_{\hat{z}}$ is the B-spline representing the $\hat{z}$-component of $\nu(\mathbf{u})$.

With ϵ defined as above, a point $\hat{\mathcal{V}}(\mathbf{u})$ on the isosurface silhouette must satisfy

$$\begin{pmatrix} |\hat{\delta}(\mathbf{u})| \\ |\hat{\eta}(\mathbf{u})| \\ |\hat{\gamma}(\mathbf{u})| \end{pmatrix} < \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \end{pmatrix}, \tag{10.19}$$

i.e., it must lie on the isosurface ($\hat{\gamma}(\mathbf{u}) < \epsilon$), the z-component of the gradient is 0 ($\hat{\delta}(\mathbf{u}) < \epsilon$), and the isosurface is orthogonal to the ray $\mathbf{r}(t)$ from the center of the pixel ($\hat{\eta}(\mathbf{u}) < \epsilon$), i.e., the z-component of the cross-product between the point and the normal of the isosurface must be zero. Similarly by multiplying Equation 10.19 by $\hat{w}(\mathbf{u})$,

$$\begin{pmatrix} |\delta(\mathbf{u})| \\ |\eta(\mathbf{u})| \\ |\gamma(\mathbf{u})| \end{pmatrix} < w(\mathbf{u}) \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \end{pmatrix}, \tag{10.20}$$

where $\delta(\mathbf{u})$ and $\eta(\mathbf{u})$ are defined in terms of the B-spline basis $\mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u})$ and where coefficients $\delta_{\mathbf{i}}$ and $\eta_{\mathbf{i}}$ can be computed using Bézier [54] or B-spline [31] multiplication.

Define

$$\mathcal{S}_I := \{\mathbf{u} : (\alpha(\mathbf{u}), \beta(\mathbf{u}), \gamma(\mathbf{u})) = (0, 0, 0)\}. \tag{10.21}$$

Then, $\mathcal{S}_I$ is the set of $\mathbf{u}$ satisfying Equation 10.15. $\mathcal{S}_I$ is the set of values where $\mathbf{r}(t)$ intersects the isosurface and is computed such that the set of points $\mathcal{V}(\mathcal{S}_I)$ on the isosurface lie inside the ray frustum corresponding to $\mathbf{r}(t)$ (type 1 and 2). Define $\mathcal{S}_S$ to be the set of $\mathbf{u}$ where $\mathcal{V}(\mathcal{S}_S)$ does not get intersected by $\mathbf{r}(t)$ but a part of an isosurface lies within the ray frustum at $\mathbf{r}(t)$ and that corresponds to a silhouette satisfying the second constraint in Equation 10.20 (type 3). In the following sections, we present a method to compute the set $\mathcal{S} = \mathcal{S}_I \cup \mathcal{S}_S$.

With this formulation, it is also possible to visualize an isoparametric surface of the geometry mapping $\mathcal{V}(\mathbf{u})$, e.g., $\mathcal{V}(\hat{u}_1, u_2, u_3)$, where $\hat{u}_1$ is fixed and u_2, u_3 varies over the parametric domain. This can be achieved by using the NURBS representation to represent fixed parameter values. As an example, in Figure 2.2c, $\hat{u}_1 = 0.5$ where u_2 and u_3 vary cutting the respective Ω along u_1 in half. Furthermore, in Figure 10.1b, $\hat{u}_3 = 0$ where u_1 and u_2 vary to show only the boundary of Ω representing the Bimba statue.

In the following, we present an efficient subdivision-based solver to compute $\mathcal{S}$.

10.3 Ray Frustum/Isosurface Intersection

As discussed in Section 10.2, finding the roots of $f(t)$ is equivalent to determining the set $\mathcal{S}_I$ as defined in Equation 10.21. To compute all intersections between a ray frustum and the isosurface, the set S_I must be computed. Here, this is achieved through a subdivision approach combined with the Newton-Raphson method.

Before our proposed isosurface intersection is applied, we find the set $\mathbb{I} \in \mathbb{G}$ of candidate geometry subpatches $(\hat{\mathcal{V}}_{\mathbf{i}}(\mathbf{u}), \hat{\mathcal{A}}_{\mathbf{i}}(\mathbf{u}))$ that potentially may be intersected by the ray frustum constructed from $\mathbf{r}(t)$ and may contain the isosurface at the isovalue $\tilde{a}$. While the technique itself does not require this step, since the relevant parts can be found through subdivision, we perform it to make the algorithm faster and more efficient. We address the different data-dependent ways that $\mathbb{I}$ can be computed in Section 10.4. In this section, we assume that $\mathbf{r}(t)$ and $\mathbb{I}$ are given. Section 10.3.1 details our intersection algorithm.

10.3.1 Algorithm

By following the framework discussed in Section 10.2, given patch $(\hat{\mathcal{V}}_\mathbf{i}(\mathbf{u}), \hat{\mathcal{A}}_\mathbf{i}(\mathbf{u})) \in \mathbb{I}$ in perspective space, a specified isovalue $\tilde{a}$ and a pixel through whose center the ray $\mathbf{r}(t)$ is passing, the coefficients for the tuple $(\mathcal{P}_\mathbf{i}(\mathbf{u}), \delta_\mathbf{i}(\mathbf{u}))$ are determined, where

$$\mathcal{P}_\mathbf{i}(\mathbf{u}) := \sum_{\mathbf{j}=1}^{\mathbf{d}+1} Q_{\mathbf{j}+\mathbf{i}-1}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u}) = (\alpha_\mathbf{i}(\mathbf{u}), \beta_\mathbf{i}(\mathbf{u}), \gamma_\mathbf{i}(\mathbf{u})), \tag{10.22}$$

with $Q_{\mathbf{j}+\mathbf{i}-1} = (\alpha_{\mathbf{j}+\mathbf{i}-1}, \beta_{\mathbf{j}+\mathbf{i}-1}, \gamma_{\mathbf{j}+\mathbf{i}-1})$, and

$$\delta_\mathbf{i}(\mathbf{u}) := \sum_{\mathbf{j}=1}^{\mathbf{d}+1} \delta_{\mathbf{j}+\mathbf{i}-1}\, \mathcal{B}_{\mathbf{i},\mathbf{d},\tau}(\mathbf{u}). \tag{10.23}$$

$\mathcal{P}_\mathbf{i}(\mathbf{u})$ has no direct geometric meaning. We refer the reader to Figure 10.7 which shows, on the left side, the two planes defining $\mathbf{r}(t)$, the isosurface, and the boundaries of the tricubic patch. On the right side, it shows the α-, β- and γ- coefficients of $\mathcal{P}_\mathbf{i}(\mathbf{u})$ derived from the two planes, the geometry and attribute data. The parametric boundaries transformed by $\mathcal{P}_\mathbf{i}(\mathbf{u})$ are depicted as well, and parts of them may lie in the interior of the parametric domain of $\mathcal{P}_\mathbf{i}(\mathbf{u})$ while forming part of the (α, β, γ)-space boundary.

Given $(\mathcal{P}_\mathbf{i}(\mathbf{u}), \delta_\mathbf{i}(\mathbf{u}))$, intersecting the ray frustum for ray $\mathbf{r}(t)$ with the isosurface at $\tilde{a}$ is a two-step algorithm:

1. Determine the superset $\mathcal{S}^S = \mathcal{S}_I^S \cup \mathcal{S}_S^S$ of approximate parameter values $\mathbf{u}$, where $\hat{\mathcal{V}}(\mathbf{u})$ lies within the ray frustum and on the isosurface at $\tilde{a}$, using a subdivision procedure with appropriate termination. (Sections 10.3.2), and

2. Apply a filtering process to remove extra parameter values in $\mathcal{S}^S$ that represent the same root (Section 10.3.6) in order to gain $\mathcal{S}$.

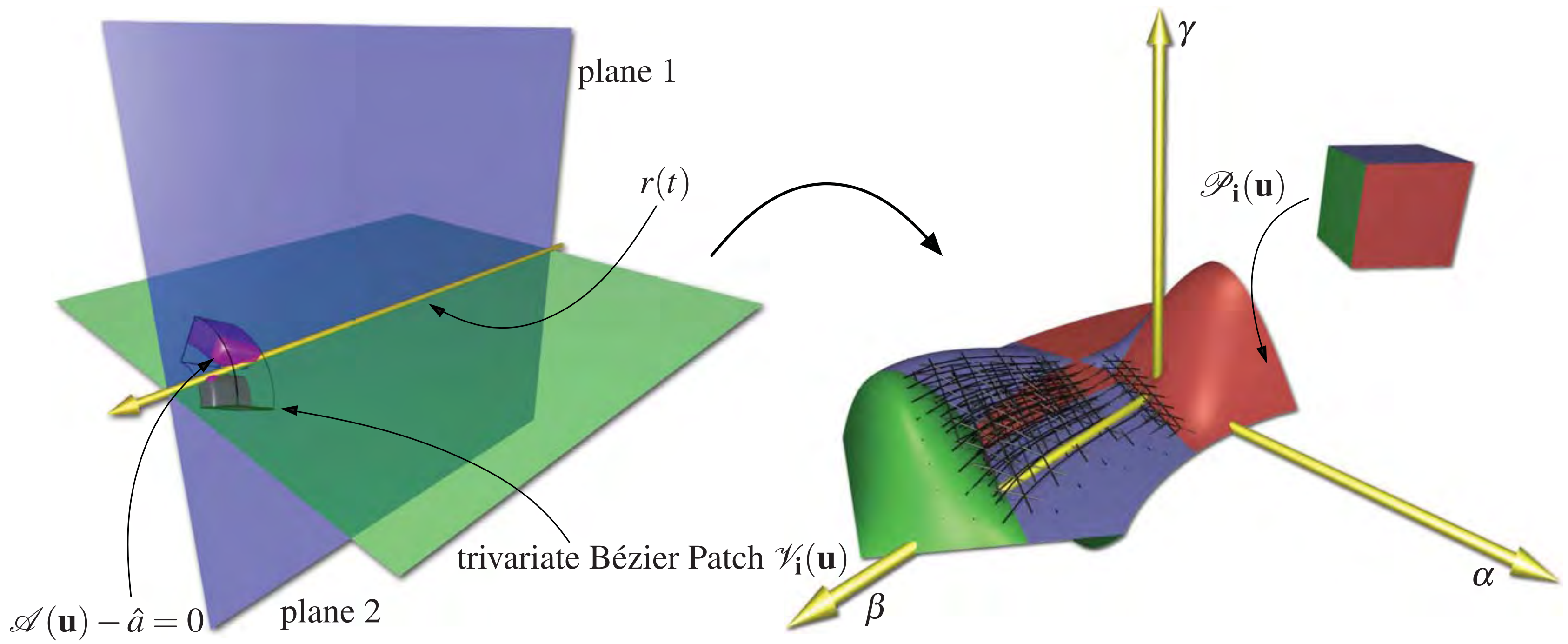

plane 1
$r(t)$
γ
$\mathscr{P}_{\mathbf{i}}(\mathbf{u})$
trivariate Bézier Patch $\mathscr{V}_{\mathbf{i}}(\mathbf{u})$
α
$\mathscr{A}(\mathbf{u}) - \hat{a} = 0$
plane 2
β

Figure 10.7. Transformation of ray and patch into (α, β, γ)-space. Left: A ray $\mathbf{r}(t)$, represented as the intersection of two planes, intersects the isosurface $\mathcal{A}(\mathbf{u}) - \hat{a} = 0$ of $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$. Right: Given $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$, $\mathcal{A}_{\mathbf{i}}(\mathbf{u})$ and the two planes, a new set of coefficients $Q_{\mathbf{k}} = (\alpha_{\mathbf{k}}, \beta_{\mathbf{k}}, \gamma_{\mathbf{k}})$ are determined to construct $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$. The ray intersects the isosurface at $\mathbf{u}_j$ where $|\mathcal{P}_{\mathbf{i}}(\mathbf{u}_j)|_\infty < \epsilon$. $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$ contains self-intersections and degeneracies depending on the number of intersections. The interior of $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$ is illustrated in wireframe. Parts of the (α, β, γ)-space boundary are formed by the interior of the parametric domain.

The following discussion details these steps.

10.3.2 Intersection Algorithm

This section presents the core of our ray frustum/isosurface intersection algorithm. Given $(\mathcal{P}_{\mathbf{i}}(\mathbf{u}), \delta_{\mathbf{i}}(\mathbf{u}))$, degeneracies and self-intersections in $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$ at the origin are related to the number of intersections between $\mathbf{r}(t)$ and the isosurface at $\tilde{a}$: Assuming there are n intersections, $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$ crosses n times within itself where $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$ evaluates to $(0, 0, 0)$. Each $\mathbf{u}$ corresponding to an intersection is an element in $\mathcal{S}_I^S$. These cases refer to type 1 and 2 intersections as illustrated in Figure 10.6.

Intersections of type 3 (see Figure 10.6) are detected by examining the signs of the coefficients of $\delta_{\mathbf{i}}(\mathbf{u})$. The $\mathbf{u}$'s corresponding to these intersections are elements in $\mathcal{S}_S^S$.

The set $\mathcal{S}^S = \mathcal{S}_I^S \cup \mathcal{S}_S^S$ is computed as follows.

The fundamental idea of our subdivision procedure is to subdivide $(\mathcal{P}_{\mathbf{i}}(\mathbf{u}), \delta_{\mathbf{i}}(\mathbf{u}))$ in all three directions at the center of its domain, which results in eight subpatches defined by the tuple $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u})) = ((\alpha_{\mathbf{i},\ell,k}(\mathbf{u}), \beta_{\mathbf{i},\ell,k}(\mathbf{u}), \gamma_{\mathbf{i},\ell,k}(\mathbf{u})), \delta_{\mathbf{i},\ell,k}(\mathbf{u}))$, where $k = 1 \ldots 8$ identifies the kth subpatch and ℓ refers to the current subdivision level; and

1. add subpatches $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u}))$ whose enclosing bounding volume contains the origin $\mathbf{0} = (0, 0, 0)$ to a list $\mathbb{L}$ and

2. examine subpatches $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ whose corresponding isosurface does not get intersected by $\mathbf{r}(t)$, but for which the corresponding isosurface potentially intersects the ray frustum (Section 10.3.4).

Depending on the geometric representation, the algorithm uses either Bézier subdivision or knot insertion [37].

The patches added to $\mathbb{L}$ in Case 1 potentially contain solutions which lie in $\mathcal{S}_I^S$. Patches examined for Case 2 potentially also contain solutions which lie in $\mathcal{S}_S^S$, i.e., Case 3 solutions. Due to properties of B-splines, note that the patch is always contained in the convex hull of its control points, and as the mesh of parametric intervals is split in half, the subdivided control mesh converges quadratically to $\mathcal{P}(\mathbf{u})$.

This procedure is recursively applied to the elements in $\mathbb{L}$ by adding new subdivision patches and removing the corresponding parent patch $(\mathcal{P}_{\mathbf{i},\ell-1,k}(\mathbf{u}), \delta_{\mathbf{i},\ell-1,k}(\mathbf{u}))$. The recursion terminates when all intersections identified with the remaining patches in $\mathbb{L}$ can be determined using the Newton-Raphson method, by using the node location (see [37]) corresponding to the coefficient in $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ closest to $\mathbf{0}$ as initial starting value. Note that initially $(\mathcal{P}_{\mathbf{i},1,1}(\mathbf{u}), \delta_{\mathbf{i},1,1}(\mathbf{u})) := (\mathcal{P}_{\mathbf{i}}(\mathbf{u}), \delta_{\mathbf{i}}(\mathbf{u}))$ and $\mathbb{L} = \{(\mathcal{P}_{\mathbf{i},1,1}(\mathbf{u}), \delta_{\mathbf{i},1,1}(\mathbf{u}))\}$; This strategy is related to the general constraint-solving technique proposed by Elber et al. in [55].

Given a subpatch $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$, a crucial issue is whether it contains the origin $\mathbf{0}$ or not. Since $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ can contain self-intersections and geometric complexity in the (α, β, γ)-space, this test is difficult to perform efficiently. The general constraint-solving technique in Elber et al. [55] looks at the signs of the coefficients in $\alpha_{\mathbf{i},\ell,k}(\mathbf{u})$, $\beta_{\mathbf{i},\ell,k}(\mathbf{u})$ and $\gamma_{\mathbf{i},\ell,k}(\mathbf{u})$ independently; that is, it investigates the properties of its Axis-Aligned Bounding Box (AABB) in the (α, β, γ)-space. Instead, we examine the geometry of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ in the (α, β, γ)-space more closely. An approximate answer to the $\mathbf{0}$-inclusion test can be given by analysing the convex hull property of NURBS [37]: If $\mathbf{0}$ does not lie within a convex set, computed from the coefficients $(\alpha_{\mathbf{k}}, \beta_{\mathbf{k}}, \gamma_{\mathbf{k}})$ defining $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$, then $\mathbf{0} \notin \mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$. However, this implies that while $\mathbf{0}$ lies within the convex boundary volume, it may not lie within its corresponding $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$. Thus, during the subdivision process, the number of elements in $\mathbb{L}$, $|\mathbb{L}|$, which contain $\mathbf{0}$, is growing or shrinking. Therefore, $\mathbb{L}$ represents a list of *potential* candidate patches which may contain $\mathbf{0}$. $|\mathbb{L}|$ at a given subdivision level ℓ is strongly dependent on how tightly the convex boundaries enclose its corresponding patches $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}) \in \mathbb{L}$. The properties of subdivision guarantee that all potential roots are kept in $\mathbb{L}$.

Generally, it can be said that given $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$'s coefficients $(\alpha_{\mathbf{k}}, \beta_{\mathbf{k}}, \gamma_{\mathbf{k}})$, a tighter convex boundary volume (e.g., convex hull) is more expensive to compute than a loose convex boundary volume (e.g., AABB), with the cost of our Oriented Bounding Box (OBB) somewhere in the middle. Given a tighter boundary volume, it is generally more expensive to test whether the origin is included in it or not. On the other hand, a tighter convex boundary will have fewer elements in $\mathbb{L}$, resulting in fewer subdivisions. Since a single subdivision step has a running time of $O((d+1)^3)$ where d is the largest degree of the three parametric directions, it is desirable to keep the number of elements in $\mathbb{L}$ as small as possible, especially as d increases. In such a scenario, a good trade-off respecting these opposing aspects is desired. Given the coefficients $(\alpha_{\mathbf{k}}, \beta_{\mathbf{k}}, \gamma_{\mathbf{k}})$ of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$, while the computation of the convex hull is more expensive compared to the much cheaper computation of an AABB, it encloses the coefficients $(\alpha_{\mathbf{k}}, \beta_{\mathbf{k}}, \gamma_{\mathbf{k}})$ much more tightly.

However, by looking locally at $\mathcal{P}_{\mathbf{i}}(\mathbf{u})$ we can adopt a much tighter bounding volume compared to the AABB, while still not as tight as the convex hull. An OBB, oriented along a given coordinate system with axes $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, is determined. Let $\mathbf{u}_c$ be the center of the parametric domain of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$. The Jacobian matrix of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ determines the first-order trivariate Taylor series. We select two of its three directions with the two largest magnitudes to form the main plane of the bounding box. Without loss of generality, suppose they are $\partial\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)/\partial u_1$ and $\partial\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)/\partial u_2$, respectively. We now form a local orthogonal coordinate system at $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ by setting $\mathbf{v}_1$ to the the unit vector in the direction $\partial\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)/\partial u_1$, $\mathbf{v}_3$ is the unit vector in the direction of $\partial\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)/\partial u_1 \times \partial\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)/\partial u_2$, and $\mathbf{v}_2 = \mathbf{v}_3 \times \mathbf{v}_1$. As in other applications, the final OBB is constructed by projecting the coefficients $(\alpha_{\mathbf{k}}, \beta_{\mathbf{k}}, \gamma_{\mathbf{k}})$ onto the planes which are located at the position $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ and have normals $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ and $-\mathbf{v}_1$, $-\mathbf{v}_2$, $-\mathbf{v}_3$, respectively.

Note that the evaluation of the derivative does not require additional computation, since it is evaluated from the coefficients computed in the subdivision process. Since $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ is a single trivariate polynomial within a patch, expanding around $\mathbf{u}_c$ is justified because the first-order Taylor series becomes a good approximation as the parametric interval decreases in size. This assumes that the determinants of the Jacobians of the neighborhood around $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ are well-behaved, i.e., do not change signs. If $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ contains self-intersections and $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ lies on a place in $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ where $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ folds into itself, then the respective determinant at $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ is equal to zero, even though the magnitudes of the partials $\partial\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)/\partial u_k$, $k = 1, 2, 3$, are well-behaved due to the smooth representation of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$. However, with increasing subdivision level ℓ, the determinants of Jacobians of the neighborhood of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ do not change signs.

Since $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ undulates through the origin multiple times depending on the number of intersections between the ray and the isosurface, this approximation is not initially useful because the bounding box is computed from the linear approximation of the Taylor series. But as the interval gets smaller, the quality of the approximation increases and the OBB encloses the coefficients of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}_c)$ more tightly (see Figure 10.8).

To compare the quality of this OBB, we used PCA on the coefficients of $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ to compute the orientation of a different OBB-bounding box on the datasets discussed in Section 10.5. Both PCA and the method discussed above result in the same order of subdivisions per pixel with PCA having slightly fewer subdivisions. However, applying PCA was on average about three times slower than our method. Table 10.1 shows the concrete timings on the various datasets.

Also, with this strategy, the number of elements in $\mathbb{L}$ is much smaller compared to the number of elements in $\mathbb{L}$ if AABB had been used. The reader is referred to Figure 10.9, which shows the glancing ray scenario with three intersections from Figure 10.7 for subdivision level $\ell = 6$. Using AABBs, on a nonsilhouette pixel of the teardrop data set, $\mathbb{L}$ has 67 elements, while by using our OBBs $\mathbb{L}$ has only 7 elements, significantly reducing subdivision effort and memory consumption. More results are given in Section 10.5.

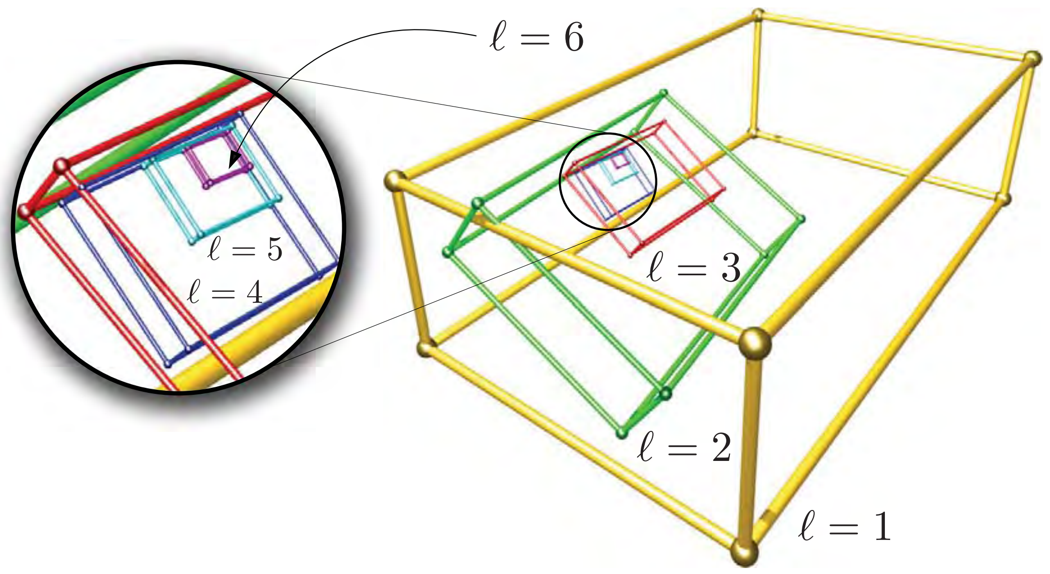


Figure 10.8. OBB hierarchy of patches, referring to a ray/isosurface intersection. With growing subdivision level ℓ, the orientation of the OBBs get closer and closer to its parent's orientation.

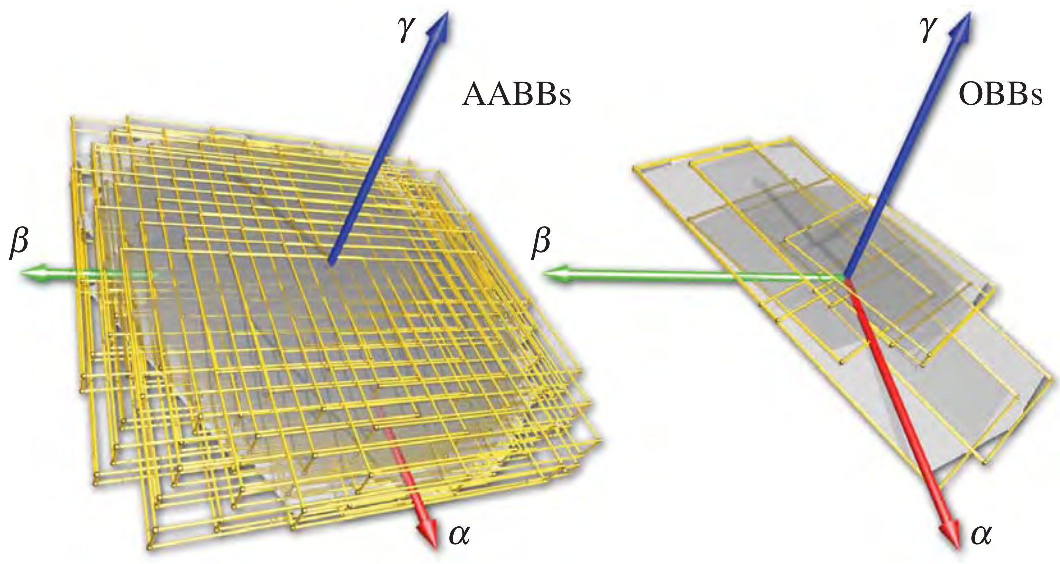


Figure 10.9. Subdivision patches stored in $\mathbb{L}$ at subdivision level $\ell = 8$. In this case, the ray glances the isosurface three times, as shown in Figure 10.7 involving more extensive subdivision and intersection tests. On the left, AABBs were used which result in $|\mathbb{L}| = 67$. On the right, our OBB computation resulting in $|\mathbb{L}| = 7$, significantly reducing subdivision work.

10.3.3 Termination of Subdivision Procedure of OBB Scheme

The previous paragraphs discussed the subdivision procedure using our OBB scheme. The termination criteria of this procedure are outlined below by answering the question: At which ℓ should the subdivision procedure terminate? A solution $\mathbf{u}_j \in \mathcal{S}_I^S$ must satisfy two requirements:

1. The patch $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ which corresponds to $\mathbf{u}_j$ must represent only one isosurface piece and must not contain folds or self-intersections so that a final application of Newton's method on $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ finds $\mathbf{u}_j$ as a unique solution;

2. $\hat{\mathcal{V}}_{\mathbf{i}}(\mathbf{u})$ has to lie within the frustum defined by the ray $\mathbf{r}(t)$ and the pixel through which $\mathbf{r}(t)$ passes.

As the number ℓ of subdivision levels increases, the geometric complexity of the patches, in $\mathbb{L}$ in terms of tangling and self-intersections, is reduced. Here, we focus on a specific OBB of one $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u})) \in \mathbb{L}$, given a subdivision level ℓ, and examine the signs of the coefficients defining $\delta_{\mathbf{i},\ell,k}(\mathbf{u})$. A sign change means that the isosurface of the patch in perspective space corresponding to $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ potentially faces towards or facing away from the ray $\mathbf{r}(t)$. This implies that $\mathbf{r}(t)$ intersects the patch at least twice and therefore $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u}))$ should be further subdivided. If there is no sign change, then the subdivision process for this patch can be terminated, and Newton's method is used to find the unique solution within the patch, such that

$$\max\left(\hat{\mathcal{V}}(\mathbf{u}_j) - proj(\hat{\mathcal{V}}(\mathbf{u}_j))\right) < \epsilon, \tag{10.24}$$

where $proj(\hat{\mathcal{V}}(\mathbf{u}_j))$ is the projection of the point $\hat{\mathcal{V}}(\mathbf{u}_j)$ onto $\mathbf{r}(t)$ and $\epsilon = 1/(2\,h)$ with h as the image resolution (see Section 10.2). More specifically, given a close enough initial solution $\mathbf{u}_0$, Newton's method tries to iteratively improve the solution and terminates when it is close enough to the exact solution. Close enough in this context means that Newton's method can terminate when the inequality equations, as defined in Equation 10.15 for a current iterative solution $\mathbf{u}_i$, are satisfied.

In the cases where the initial solution is not good enough for Newton's method, the patch $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u}))$ is further subdivided. This also guarantees that a solution associated with a ray will be within the ray's frustum and does not overlap with adjacent ray frustums. In the rare case that the solution is exactly on the pixel boundary, we use the half-open frustum to guarantee that it is included in only one of the possible adjacent pixels.

10.3.4 Ray Frustum/Isosurface Silhouette Intersection

Before a subpatch $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u}))$ whose OBB does not contain $\mathbf{0}$ is discarded, it must be examined to determine whether the subdomain it covers in $\hat{\mathcal{V}}(\mathbf{u})$ contains any isosurface silhouette intersecting the ray frustum $\mathbf{r}(t)$ in perspective space. If there is no sign change in the coefficients defining either $\gamma_{\mathbf{i},\ell,k}(\mathbf{u})$ or $\delta_{\mathbf{i},\ell,k}(\mathbf{u})$, then the patch can be discarded, because a potential intersection will be caught using the origin-inclusion-test (Section 10.3.1) since in this case the respective isosurface piece completely faces towards or faces away from $\mathbf{r}(t)$.

A sign change in both sets of the coefficients implies that a potential part of the isosurface passes through the ray frustum, facing towards and away from $\mathbf{r}(t)$. If there is such a piece of the isosurface silhouette, then a $\mathbf{u}$ is computed so that $\hat{\mathcal{V}}(\mathbf{u})$ lies on the isosurface silhouette and $\mathbf{u}$ is added to $\mathcal{S}_S^S$.

As discussed in Section 10.2, an isosurface that intersects the frustum (type 3) must have an isosurface silhouette in the frustum, i.e., it must satisfy Equation 10.20. Given $(\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u}))$ with sign changes both in the coefficients defining $\gamma_{\mathbf{i},\ell,k}(\mathbf{u})$ and defining $\delta_{\mathbf{i},\ell,k}(\mathbf{u})$, a patch $\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u})$ is constructed, where

$$\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u}) = (\gamma_{\mathbf{i},\ell,k}(\mathbf{u}), \delta_{\mathbf{i},\ell,k}(\mathbf{u}), \eta_{\mathbf{i},\ell,k}(\mathbf{u})) \tag{10.25}$$

and the number of self-intersections corresponds to the number of solutions $\mathbf{u}$.

10.3.5 Termination of Subdivision Process of Patch $\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u})$

Subdivision is used to solve $\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u}) = \mathbf{0}$, where the 3D version of the normal cone (NC) test proposed in the work [176] is used to make a faithful decision to stop the subdivision process of patch $\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u})$. This test computes the NCs for the mappings $\gamma_{\mathbf{i},\ell,k}(\mathbf{u})$, $\delta_{\mathbf{i},\ell,k}(\mathbf{u})$ and $\eta_{\mathbf{i},\ell,k}(\mathbf{u})$. Elber et al. show that when the NCs of these three mappings do not intersect, then the patch can contain at most one zero. If the NC test fails, i.e., $\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u})$ contains self-intersections, then $\mathcal{Q}_{\mathbf{i},\ell,k}(\mathbf{u})$ is further subdivided. If the NC succeeds, this implies that a subdivided patch does not contain self-intersections. Newton's method is used as above to find a solution $\mathbf{u}$ which is added to $\mathcal{S}_S^S$ when Equation 10.20 is satisfied.

Note that this additional solution step to find points on an isosurface silhouette within a ray frustum is executed only at isosurface silhouettes, when there are sign changes in the coefficients defining $\gamma_{\mathbf{i},\ell,k}(\mathbf{u})$ and $\delta_{\mathbf{i},\ell,k}(\mathbf{u})$. In most cases, as observed in our experiments, the ray $\mathbf{r}(t)$ intersects the isosurface.

10.3.6 Filtering Intersection Result

The subdivision procedure discussed in the previous section, applied to the patch $(\mathcal{V}_{\mathbf{i}}(\mathbf{u}), \mathcal{A}_{\mathbf{i}}(\mathbf{u})) \in \mathbb{I}$, outputs the superset $\mathcal{S}^S$ of approximate parameter values $\mathbf{u}_j$, i.e., where $|\mathcal{A}(\mathbf{u}_i) - \hat{a}| < \epsilon$. By following the framework from Section 10.2, our method is guaranteed to compute all roots. However, due to the approximate $\mathbf{0}$-inclusion test and the fact that it is a numerical method, it can be the case that $\mathcal{S}^S$ contains multiple solutions that represent the same root. This is because of the use of OBB to determine whether $\mathbf{0}$ is contained in its respective patch. As discussed above, a $\mathcal{P}_{\mathbf{i},\ell,k}(\mathbf{u})$ may not contain $\mathbf{0}$ while its OBB contains it. A final postprocess on $\mathcal{S}^S$, yielding the set $\mathcal{S}$, is therefore required for the removal of duplicate solutions.

In the scenario of direct isosurface visualization, multiple cases can appear (shown in Figure 10.10, computed solutions in green). In Case (I), it can happen that parts of the isosurface lie very close together. Therefore, the corresponding solutions are numerically very similar, even though they represent different solutions. In Case (II), the ray might glance or touch the isosurface tangentially, which corresponds to two solutions. In Case (III), the usual case, two solutions can represent the same true solution even though they are numerically different. We remove duplicates by examining the derivative of the function $f(t)$ given by:

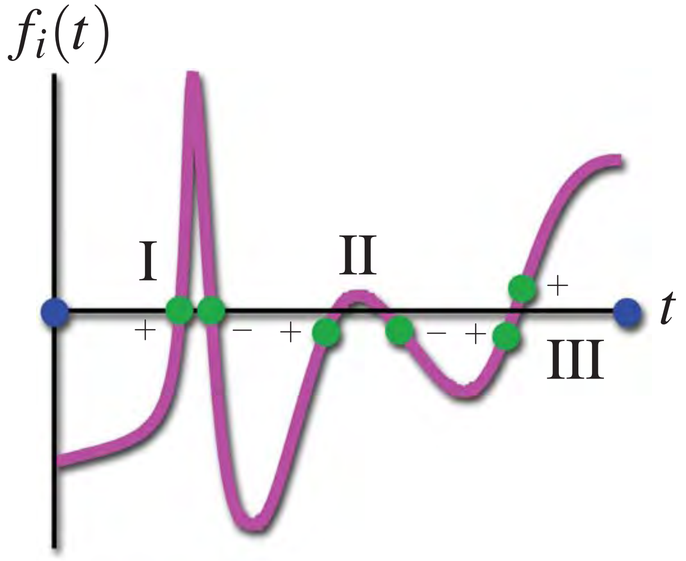


Figure 10.10. $\mathcal{S}$ can contain duplicate solutions which can arise due to the scenarios I, II and III. The derivative of the scalar function $f(t)$ is used to filter $\mathcal{S}$ to identify unique solutions and solutions representing the same root.

$$f'(t) = \langle \frac{\partial \mathbf{r}(t)}{\partial t}, J^{-1} \circ \nabla \mathcal{A}(\mathcal{V}^{-1}(\mathbf{r}(t))) \rangle, \qquad (10.26)$$

where J^{-1} is the Jacobian of $\mathcal{V}^{-1}(\mathbf{r}(t))$, and $\circ$ is the matrix/vector product. As the ray $\mathbf{r}(t)$ travels through the volume, it enters and eventually exits the isosurface. Entering means that $\mathbf{r}(t)$ intersects the isosurface at the positive side; this corresponds to a positive derivative of Equation 10.26 at the corresponding entry location. The exit point refers to a negative derivative of Equation 10.26. With this observation, Case (I) can be identified. Case (II) appears at the silhouette of the isosurface. If $f'(t) \approx 0$, then one of the corresponding solutions can be discarded. For Case (III), since the signs of $f'(t)$ for the corresponding solutions are both positive or negative, respectively, one of them can be discarded.

In our implementation, for every $\mathbf{u}_i \in \mathcal{S}^S$, we determine its corresponding t_i by solving the linear equation $t_i = \mathbf{r}^{-1}(\mathcal{V}(\mathbf{u}_i))$ and evaluate $f'(t_i)$. The resulting list of t-values is sorted in increasing order. Finally, the sorted list, which corresponds to the order in which the ray travels through the volume, is traversed by removing those elements which violate the rule of alternation of the signs of $f'(t_i)$ within the list. Note, that in some rare subpixel cases, incorrect ordering can occur and cause incorrect transparency results. This is a subpixel problem and can be resolved by further subdividing the pixel. However, we found that no visual artifacts result.

This algorithm detects intersections in the pathological case that a whole interval of $\mathbf{r}(t)$ lies on the isosurface. However, as with all numerical methods, there are no ways to determine this analytical condition, but instead, find many discrete values of t. We set a heuristic threshold on the maximum number of ray-isosurface intersections per ϵ-length of t. If the number of intersections exceeds it, we use only the smallest value and the largest value.

10.4 Determining the Set of Intersection Patches

As discussed above, $\mathbb{I} \subset \mathbb{G}$ is the set which contains the geometric subpatches $(\mathcal{V}_\mathbf{i}(\mathbf{u}), \mathcal{A}_\mathbf{i}(\mathbf{u}))$ that intersect the ray frustum constructed from $\mathbf{r}(t)$ and through which the isosurface $\mathcal{A}(\mathbf{u}) - \hat{a} = 0$ passes. There are multiple ways to determine $\mathbb{I}$, which depend on the number of coefficients defining $\mathcal{V}(\mathbf{u})$ and the geometry it describes in physical space. In our implementation, we distinguish between three different types of geometry: (1) general geometry describing a physical domain with a large number of coefficients; (2) general geometry describing a physical domain of interest with few coefficients; and (3) a uniform grid, where $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ describes the identity mapping, i.e., $\mathcal{V}_{\mathbf{i}}(\mathbf{u}) = \mathbf{u}$.

For (1) and (2) we employ a kd-tree as an acceleration structure, where an AABB is computed from the coefficients of $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ where $(\mathcal{V}_{\mathbf{i}}(\mathbf{u}), \mathcal{A}_{\mathbf{i}}(\mathbf{u})) \in \mathbb{G}$. $\mathbb{I}$ is determined by kd-tree traversal using the traversal algorithm proposed by Sung et al. [192], where the ray $\mathbf{r}(t)$ is intersected with the bounding boxes. Note the resulting $\mathbb{I}$ can contain patches that are not intersected by $\mathbf{r}(t)$. If $|\mathbb{G}|$ is small, then the AABBs do not tightly bound $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$, and $\mathbb{I}$ contains a larger number of patches that do not intersect $\mathbf{r}(t)$. In that case, we apply knot insertion to the elements in $\mathbb{G}$ to turn them into Bézier patches whose corresponding AABBs are much tighter. When $\mathcal{V}(\mathbf{u})$ consists of a large number of coefficients, the ratio between the AABB and its corresponding $\mathcal{V}_{\mathbf{i}}(\mathbf{u})$ is close to one. In that case, Bézier conversion is not a significant advantage, but a disadvantage because of its higher memory consumption and preprocessing time. In (3), where $\mathcal{V}(\mathbf{u})$ represents a uniform grid, i.e., when $\mathcal{V}(\mathbf{u}) = \mathbf{u}$, conventional uniform grid traversal is used without any data preprocessing. Also note that in this case (e.g., Figure 10.1d), the smooth representation for $\mathcal{A}(\mathbf{u})$ is generated using a B-spline [121] filter to which our method is applied.

10.5 Analysis and Results

This section is concerned with the correctness and efficiency of our approach. Verifying the correctness of an isosurface visualization technique on acquired data is difficult, especially in terms of correctness of the topology and existence of all features, since given data usually only approximate the true solution (e.g., the results of Galerkin's method or data from a CT scan). In this section, we use the fact that every rational polynomial can be represented with a NURBS representation, i.e., there are coefficients $a_{\mathbf{i}} \in \mathbb{R}$ such that

$$a(x,y,z) \equiv \mathcal{A}(x,y,z) = \sum_{\mathbf{i}=1}^{\mathbf{n}} a_{\mathbf{i}}\, \mathcal{R}_{\mathbf{i},\mathbf{d},\tau}(x,y,z), \tag{10.27}$$

defined over a rectangular parallelepiped of $\Omega \in \mathbb{R}^3$, where Ω is rectangular and where $a(x,y,z)$ is an algebraic function. Given $a(x,y,z)$ and a NURBS basis (as defined in Section 10.1.1) whose degree matches the highest degree of $a(x,y,z)$, the coefficients $a_{\mathbf{i}}$ can be derived by solving the multivariate version of Marsden's identity [123]. If $a(x,y,z)$ is a cubic algebraic function, the approach of Bajaj et al. [13] can be used to compute coefficients $a_{\mathbf{i}}$ for the NURBS basis. For our tests, we chose the isosurface at 0.0 of the teardrop function, defined as $a(x,y,z) = x^5/2 + x^4/2 - y^2 - z^2$, a common function to test correctness of a visualization technique. The thin features around the origin, as seen in Figure 10.1c, are challenging to isosurface meshing techniques where areas around the thin feature are missing (e.g., see work by [153]). Next to the coefficients $a_{\mathbf{i}}$, our method requires a choice of coefficients $P_{\mathbf{i}} = (x_{\mathbf{i}}, y_{\mathbf{i}}, z_{\mathbf{i}})$ to define $\mathcal{V}(\mathbf{u})$. If $P_{\mathbf{i}}$ are node locations as defined in [37], then $a(x,y,z) \equiv \mathcal{A}(x,y,z)$ is achieved. However, since our technique is independent of the geometric complexity, a choice can be made on the mapping $\mathcal{V}(\mathbf{u})$. A more general version of Equation 10.27 is $a(\mathcal{V}^{-1}(\mathbf{u})) \equiv \mathcal{A}(\mathbf{u})$, in which $a(x,y,z)$ undergoes a nonlinear transformation defined by $\mathcal{V}(\mathbf{u})$ deforming Ω. By referring to Figure 10.1c, Ω is stretched and perturbed, which results in a deformation of $a(x,y,z) = 0$. The deformation does not affect the accuracy of our algorithm in reproducing the thin feature discussed above, indicating robustness and topological correctness of our technique at the per-pixel level.

In Figure 10.11, the number of subdivisions per pixel of the isosurface intersection technique, using AABBs and OBBs constructed in the above section is visualized. The images are generated from the same view as the shaded version in Figure 10.1. It can be seen that major work is done only for pixels that actually correspond to a point on the isosurface and pixels on the silhouette. When employing an AABB, a large number of silhouette pixels require an average of 270 and up to 380 subdivisions per pixel. With OBBs, only a few pixels require more than 68 subdivisions, and on average, 35 subdivisions are needed for the silhouette. This means that the number of subdivision

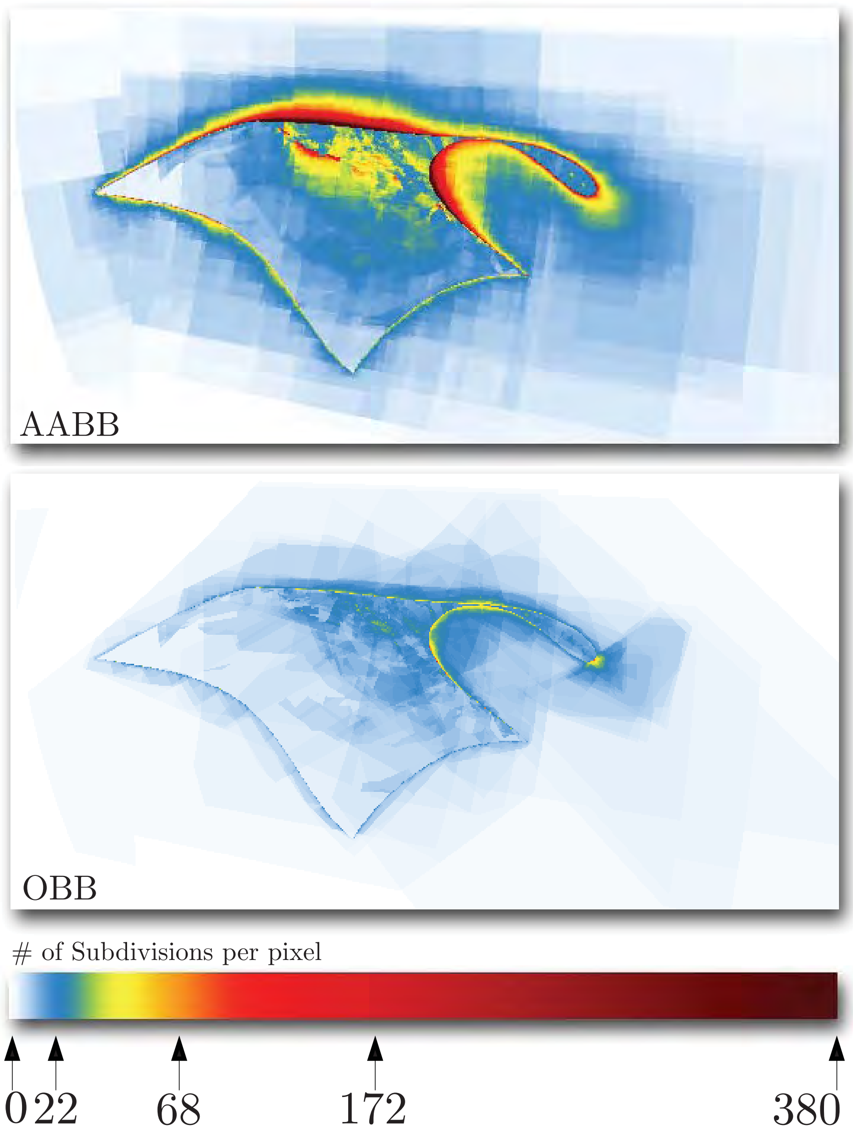
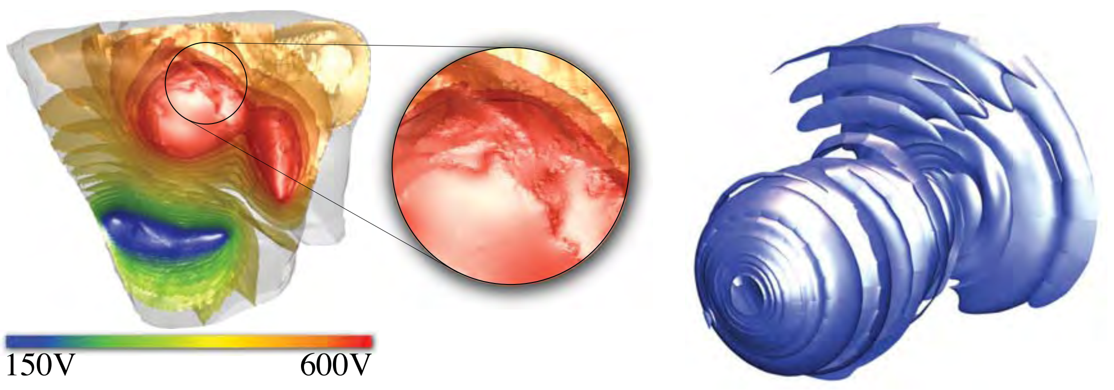


Figure 10.11. Number of subdivisions per pixel frustum using AABB and OBB for teardrop isosurface from Figure 10.1.

levels for OBB is much smaller than with AABB, resulting in a more memory efficient algorithm.

10.5.1 Timings

Figure 10.12a shows the result of our algorithm, rendering geometry of a torso with multiple isosurfaces of the potential trilinear (cubic) field. Both are represented using unstructured hex meshes. In Figure 10.12b, we present the visualization of an isosurface



(a) (b)

Figure 10.12. Additional data sets. (a) Unstructured hexahedral mesh ($\approx$ 2.3 million elements) of a segmented torso. Isosurfaces representing voltages of the potential field (using a trilinear basis) are used to specify locations of electrodes to determine efficacy of defibrillation to find a good location to implant a defibrillator into a child. (b) Wake of a rotating canister traveling through a fluid (isosurface of pressure from spectral/hp element CFD simulation data as used in the work [142, 130]). The $C^{(0)}$ nature of the boundaries of the spectral/hp elements can be seen on the isosurface and is not an artifact of our proposed method.

of pressure (isovalue = 0) generated due to a rotating canister traveling through an incompressible fluid. The data set was generated by the spectral/hp high-order finite element CFD simulation code, Nektar, and was used as test data set for visualization in the works [142, 130]. The geometry of this data is trilinear ($C^{(0)}$), and the attribute data is tricubic.

Table 10.1 provides concrete numbers of the proposed approach in comparison to the AABB and PCA as discussed in Section 10.3. The table provides average render times (μ time), additional information such as the average number of pixels per frame (μ pixel), the average number of subdivisions per frame (μ subd.), the average list size of L overall (μ list size) and the standard deviation of the list size L overall (σ list size). Due to space constraints for PCA, only the render times are presented, since the remaining values are within $\pm 1\%$ compared to our method.

The data in the table were generated by rotating the camera around the respective isosurfaces in 360 frames, using Phong shading and normals computed from the NURBS representation. The above information is generated using our method's OBBs and AABBs from the same space. Subdivision is the major work in both cases. However,

Table 10.1. Average image generation times using OBB and AABB, respectively. The table also shows the timings (in seconds) for each data set when PCA is used instead of our method to compute the OBBs. The degree column presents degrees for the geometry and attribute mapping (tl=trilinear, tc=tricubic, tq=triquintic); μ is the mean; and σ is the standard deviation. The image resolution is 512×512.

data set	**degree**	**# patches**	**μ pixel**
			(per frame)
Cylinder	tc/tc	$5 \times 2 \times 5$	57 408
Bimba	tc/tc	$27 \times 45 \times 9$	273 024
Teardrop	tq/tq	1	56 078
VisHuman	tl/tc	$253 \times 253 \times 253$	51 625
Silicium	tl/tc	$95 \times 31 \times 31$	95 425
Torso	tl/tl	2321045	123 084
CFD	tl/tc	5736	631 342

OBB

data set	**μ time PCA**	**μ time ours**	**μ subd.**	**μ / σ list size**
	(per frame)	(per frame)	(per frame)	(overall)
Cylinder	0.29	0.15	299 790	1.89/1.03
Bimba	0.58	0.27	463 281	1.17/0.49
Teardrop	1.97	0.65	371 304	3.54/1.44
VisHuman	1.06	0.40	278 317	1.04/0.24
Silicium	0.96	0.43	356 862	1.05/0.24
Torso	1.15	0.83	3 502 902	1.09/0.40
CFD	1.61	0.77	2 016 399	1.88/1.16

AABB

data set	**μ time**	**μ subd.**	**μ / σ list size**
	(per frame)	(per frame)	(overall)
Cylinder	0.31	667 000	2.70/2.56
Bimba	0.81	2 090 467	1.58/5.42
Teardrop	1.87	1 007 734	6.14/3.46
VisHuman	0.72	587 194	1.12/4.03
Silicium	0.74	738 945	1.16/3.29
Torso	1.43	14 913 568	1.36/6.26
CFD	1.02	4 124 430	2.70/3.26

both cases outperform the typical problem formulation with the four equations and four unknowns discussed in Section 10.1, since subdivision has to be performed on four parametric directions with each subdivision being $O((d+1)^4)$ versus 3 parametric subdivisions with $O((d+1)^3)$ for each subdivision, where d is the degree.

The timings were taken on interlinked Intel Xeon X7350 Processors comprised of 32 cores using gcc version 4.3 and OpenMP. Evidently, OBB is up to three times faster than AABB, depending on the isosurface complexity.

10.6 Conclusion

In this chapter, we proposed a novel direct isosurface visualization technique which computes all the intersections between a ray and an isosurface embedded in various representations, such as data-fitted geometry, rational geometry, and uniform grids. Our framework supports rendering the isosurface with view-independent transparency. The technique is robust, user friendly, and easy to implement: All the images in this chapter, which show different isosurface visualization scenarios, did not require tweaking and had no parameter readjustment. We have shown that even though the high-order geometry mapping contains parametric distortions (e.g., Figure 10.1c), important features in the isosurface are still maintained, something that is challenging for most isosurface techniques. Currently, we are working on a GPU implementation where we expect a significant speed-up of the technique. A direction for future work is to extend the approach to tessellated isosurfaces.

CHAPTER 11

CONCLUSION

This dissertation is concluded by summarizing its contributions and discuss future research directions.

11.1 Contributions

This work presents a framework to create, analyze and visualize volumetric NURBS geometries. It focuses on the methodology of IA, bridging the gap between Computer Aided Design and Finite Element Analysis. The work starts with a discussion in Chapter 3, that the quality of the model can significantly influence the quality of IA.

With these model quality concepts in mind, from Chapter 4 to Chapter 7, this work focused in particular on the *post facto* modeling path by introducing various methodologies to create volumetric representations from multiple input boundaries, representing the domain of interest with interior materials.

Creating these volumetric representations consists of several steps. The first step is to determine an appropriate decomposition strategy, to decompose the object into subvolumes, on which a tensor-product style parameterization can be established. The goal in this work was to find a decomposition where the subvolumes are as large as possible so that extraordinary points can be avoided and to ease establishing a higher continuity among the subvolumes in later datafitting stages. Three decomposition strategies based on midstructures have been proposed. The first strategy discussed in Chapter 4 allows fitting a single trivariate B-spline to the parameterization and has been shown especially useful for generalized cylinders. In the second method proposed in Chapter 5, a single midsurface is used to decompose the object and was applied to objects of higher genus. While for the first method, the user has to make only initialized choices to create the volumetric parameterization, the user input for the second decomposition strategy is more involved. To reduce user input, a more generalized midstructure called the generalized swept midstructure (GSM) has been proposed in Chapter 7, generalizing the concept of the midsurface to a structure with a sheet-by-sheet topology and 1D elements in more tubular regions. As in the first method, the user only has to choose a small number of points on the input surface from which a harmonic scalar field is computed. This scalar field is used to construct the GSM.

Lastly, a more general method has been proposed in Chapter 6, where the user specifies a midstructure representation such as a GSM. This midstructure is used to create a mixed element representation with high-order trivariate NURBS patches at the boundary and linear tetrahedra in the interior of the domain. A collocation method has been proposed to link these element types together. This method is especially useful for an extended class of objects for which the two methods above are too restricted.

Once the object has been decomposed, the corresponding subvolumes are parameterized so that the parameterization matches among the shared boundaries. This parameterization process is based on harmonic functions, where a linear problem is solved along the parameteric directions. Two methods have been proposed to improve the orthogonality of the volumetric parameterizations. In the first method, discussed in Chapter 4, two scalar fields are established on the volumetric domain such that its corresponding vector fields are orthogonal. Based on the cross product of these two vector fields, lines are integrated through the volume to complete the model. In the second method proposed in Chapter 8, the MIPS method [81] is adapted to the 3D case where the mapping from the parameteric domain to the physical space is optimized by improving isotropy. Traditionally, these nonlinear optimizations are slow, and therefore, an approach has been proposed to accelerate the optimization process of gradient-based nonlinear optimization algorithms.

Once the subvolumes have been parameterized, a method has been proposed to fit multiple high-order trivariate B-splines to sampled structured grids in Chapter 9. Special care has been taken so that oscillations, typically arising in interpolation schemes, are avoided. The method starts with an approximation and the fit is iteratively improved to enhance the accuracy of the approximation to the input data. It has been demonstrated that this method is not only useful for trivariate B-spline fitting, but is also useful for more general application such as streamline visualization of vector field input.

Once simulation has been applied to the collection of higher order trivariate B-spline or NURBS elements, one way to examine the analysis result is to visualize isosurfaces of the solution which is an attribute of the high-order representation. Several methods exist to extract the isosurface, where traditionally, often a lower order representation such as a triangle mesh is extracted using a technique such as marching cubes or marching tetrahedra. Extracting a piecewise linear representation from a high-order representation introduces error. A method has been proposed in Chapter 10 to directly visualize isosurfaces, by combining subdivision and numerical root-finding to guarantee accuruate visualization of the respective isosurfaces. In addition to demonstrating the method on simulation results, the generality of the method has been demonstrated by visualizing isosurfaces of CT data and isosurfaces of more complex algebraic functions transformed by a nonlinear transformation.

11.2 Outlook

There are various directions for future research. In particular, a more general modeling method would allow the input of interior material boundaries that are enclosed within each other. Furthermore, a more general methodolgy should take sharp features in an input representation into account. This would allow parameterization of a wider class of CAD models. It has been demonstrated that the GSM is useful to create more complex hexahedral parameterizations. However, a concrete methodology and associated decomposition strategy still must be defined. Furthermore, in the proposed framework, adjacent patches are $C^{(0)}$ continuous. A future fitting method could take the block structure of the volumetric decomposition into account to increase the continuity among adjacent patches. Higher continuity among patches will potentially improve the simulation applied to the volumetric shape representation. Initial results for the 2D case already exist. Lastly, there are possibilities to extend the visualization framework to a wider class of elements, such as tetrahedra. The main challenge would be to define a subdivision scheme based on the given element type.

APPENDIX

PUBLICATIONS

The major part of this dissertation has been published in scientific journals and/or peer-reviewed conference proceedings. The concept of analysis-aware modeling proposed in Chapter 3 is published in

- Elaine Cohen, Tobias Martin, Robert M. Kirby, Tom Lyche, Richard F. Riesenfeld, "Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis,"
  In *Computational Methods in Applied Mechanics and Engineering*, Pages 334-356, Volume 199, Number 5-8, 2010

The modeling method generalizing the concept of generalized cylinders discussed in Chapter 4 and datafitting approach discussed in Chapter 9 is published in

- Tobias Martin, Elaine Cohen, Robert M. Kirby,
  "Volumetric parameterization and trivariate b-spline fitting using harmonic functions,"
  In *Computer Aided Geometric Design*, Pages 648-664, Volume 26, Number 6, 2009 and in *Proceedings of ACM Solid and Physical Modeling Symposium*, Stony Brook, NY, June 2–4, 2008. Best Paper Award.

Modeling more complex geometries with higher genus and based on a single midsurface introduced in Chapter 5 is published in

- Tobias Martin, Elaine Cohen
  "Volumetric Parameterization of Complex Objects by Respecting Multiple Materials,"
  In *Computers & Graphics*, Pages 187-197, Volume 34, Number 3, 2010

The mixed element approach based on a userspecified midstructure discussed in Chapter 6 will be published in

- Tobias Martin, Elaine Cohen, Robert M. Kirby,
  "Mixed-Element Volume Completion from NURBS surfaces,"
  In *SMI 2012, Computers & Graphics*

The generalized swept midstructure proposed in Chapter 7 will be published in

- Tobias Martin, Guoning Chen, Suraj Musuvathy, Elaine Cohen, Charles Hansen,
  "Generalized Swept Mid-structure for Polygonal Models,"
  In *Eurographics 2012, Computer Graphics Forum*

The optimization framework to accelerate gradient-based nonlinear optimization algorithms introduced in Chapter 8 is currently under review in

- Tobias Martin, Pushkar Joshi, Miklós Bergou, Nathan Carr,
  "Efficient Nonlinear Optimization via Multiscale Gradient Filtering,"
  submitted to *Computer Graphics Forum*

The visualization method to directly visualize isosurfaces of hexahedral based higher order volume and attribute representations discussed in Chapter 10 is published in

- Tobias Martin, Elaine Cohen, Robert M. Kirby,
  "Direct Isosurface Visualization of Hex-Based High-Order Geometry and Attribute Representations,"
  In *IEEE Transactions on Visualization and Computer Graphics*, Pages 753-766, Volume 18, Number 5, May 2012

REFERENCES

[1] Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] Abert, O., Geimer, M., and Müller, S. Direct and fast ray tracing of nurbs surfaces. *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (2006), 161–168.

[3] Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Jüttler, B., and Rabl, M. Medial axis computation for planar free-form shapes. *Computer-Aided Design 41*, 5 (2009), 339–349.

[4] Aigner, M., Heinrich, C., Jüttler, B., Pilgerstorfer, E., Simeon, B., and Vuong, A. V. Swept volume parameterization for isogeometric analysis. In *Proceedings of the 13th IMA International Conference on Mathematics of Surfaces XIII* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 19–44.

[5] Aksoylu, B., Khodakovsky, A., and Schröder, P. Multilevel solvers for unstructured surface meshes. *SIAM J. Sci. Comput. 26* (April 2005), 1146–1165.

[6] Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., and Desbrun, M. Anisotropic polygonal remeshing. *ACM Trans. Gr. 22*, 3 (July 2003), 485–493.

[7] Amanatides, J. Ray tracing with cones. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 129–135.

[8] Amenta, N., Choi, S., and Kolluri, R. The power crust. In *Proceedings of 6th ACM Symposium on Solid Modeling* (2001), pp. 249–260.

[9] Arbarello, E., Cornalba, M., Griffiths, P., and Harris, J. Topics in the theory of algebraic curves. *Princeton Univ. Press* (to appear).

[10] Armstrong, C., Robinson, D., McKeag, R., Li, T., Bridgett, S., Donaghy, R., McGleenan, C., Donaghy, R., and Mcgleenan, C. Medials for meshing and more. In *IMR* (1995), pp. 277–288.

[11] Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., and Lee, T.-Y. Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27* (August 2008), 44:1–44:10.

[12] Axelsson, O. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1994.

[13] Bajaj, C. L., Holt, R. L., and Netravali, A. N. Rational parametrizations of nonsingular real cubic surfaces. *ACM Trans. Graph. 17*, 1 (1998), 1–31.

[14] Baraff, D., and Witkin, A. P. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98* (July 1998), Computer Graphics Proceedings, Annual Conference Series, pp. 43–54.

[15] Baxter, W. V., Scheib, V., and Lin, M. C. dAb: Interactive haptic painting with 3D virtual brushes. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), E. Fiume, Ed., ACM Press / ACM SIGGRAPH, pp. 461–468.

[16] Bazilevs, Y. amd Beirao da Veiga, L., Cottrell, J., Hughes, T., and Sangalli, G. Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes. *Mathematical Methods and Models in Applied Sciences 16* (2006), 1031–1090.

[17] Bazilevs, Y., and Hughes, T. Nurbs-based isogeometric analysis for the computation of flows about rotating components. *Computational Mechanics 43* (2008), 143—150.

[18] Berzins, M. Mesh quality: A function of geometry, error estimates or both. In *7th International Meshing Roundtable* (1998), pp. 229–238.

[19] Biasotti, S., Attali, D., Boissonnat, J., Edelsbrunner, H., Elber, G., Mortara, M., Baja, G., Spagnuolo, M., Tanase, M., and Veltkamp, R. Skeletal structures. *Shape Analysis and Structuring* (2008), 145–183.

[20] Binford, T. O. Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Controls* (Miami, Florida, December 1971).

[21] Blinn, J. How many ways can you draw a circle? *IEEE Computer Graphics and Applications 7*, 8 (1987), 39–44.

[22] Blinn, J. F. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[23] Botsch, M., and Kobbelt, L. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 630–634.

[24] Briggs, W. L., Henson, V. E., and McCormick, S. F. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[25] Butkov, E. *Mathematical Physics*. Addison-Wesley Publishing Company, Reading, MA, 1968.

[26] Butlin, G., and Stops, C. Cad data repair. *Proceedings of the 5th International Meshing Roundtable 1996* (1996), 7–12.

[27] C. Bernardi, Y. Maday, A. P. A new nonconforming approach to domain decomposition: the mortar element method. *Nonlinear Partial Differential Equations and their Applications, Pitman/Wiley* (1992).

[28] Casale, M. S., and Stanton, E. L. An overview of analytic solid modeling. *IEEE Computer Graphics and Applications* (1985), 45–56.

[29] Charpiat, G., Keriven, R., Pons, J.-P., and Faugeras, O. Designing spatially coherent minimizing flows for variational problems based on active contours. In *ICCV* (2005), IEEE Computer Society, pp. 1403–1408.

[30] Chazal, F., and Lieutier, A. Stability and homotopy of a subset of the medial axis. In *SM '04: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications* (2004), pp. 243–248.

[31] Chen, X., Riesenfeld, R. F., and Cohen, E. Sliding windows algorithm for b-spline multiplication. In *SPM '07: Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2007), ACM, pp. 265–276.

[32] Ciarlet, P. G. *The Finite Element Method for Elliptic Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[33] Cignoni, P., Floriani, L. D., Montani, C., Puppo, E., and Scopigno, R. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *VVS '94: Proceedings of the 1994 Symposium on Volume Visualization* (New York, NY, USA, 1994), ACM, pp. 19–26.

[34] Cohen, E., Lyche, T., and Riesenfeld, R. F. Discrete b-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing 15*, 2 (October 1980), 87–111.

[35] Cohen, E., Lyche, T., and Schumaker, L. L. Algorithms for degree-raising of splines. *ACM Transactions on Graphics 4*, 3 (July 1986), 171–181.

[36] Cohen, E., Martin, T., Kirby, R., Lyche, T., and Riesenfeld, R. Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 199*, 5-8 (2010), 334 – 356. Computational Geometry and Analysis.

[37] Cohen, E., Riesenfeld, R. F., and Elber, G. *Geometric Modeling with Splines: An Introduction.* A. K. Peters, Ltd., Natick, MA, USA, 2001.

[38] Coons, S. A. Surfaces for computer-aided design of space forms. Tech. Rep. MAC-TR-41, MIT, 1967.

[39] Cottrell, J. A., Hughes, T. R., and Reali, A. Studies of refinement and continuity in isogeometric structural analysis. *Comput. Methods Appl. Mech. Engrg. 196* (2007), 4160–4183.

[40] Cottrell, J. A., Reali, A., Bazilevs, Y., and Hughes, T. R. Isogeometric analysis of structural vibrations. *Comput. Methods Appl. Mech. Engrg. 195*, 41-43 (2006), 5257–5296.

[41] Culver, T., Keyser, J., and Manocha, D. Exact computation of the medial axis of a polyhedron. *Comput. Aided Geom. Des. 21* (January 2004), 65–98.

[42] Damon, J. Swept regions and surfaces: modeling and volumetric properties. *Theoretical Computer Science* (2008), 66–91.

[43] Davis, P. J. *Circulant Matrices.* John Wiley & Sons, Inc., New York, 1979.

[44] de Toledo, R., Levy, B., and Paul, J.-C. Iterative methods for visualization of implicit surfaces on GPU. In *ISVC, International Symposium on Visual Computing* (Lake Tahoe, Nevada/California, November 2007), Lecture Notes in Computer Science, Springer, pp. 598–609.

[45] Deville, M., Mund, E., and Fischer, P. *High Order Methods for Incompressible Fluid Flow.* Cambridge University Press, Cambridge, 2002.

[46] Dey, T. K., and Goswami, S. Tight cocone: a water-tight surface reconstructor. In *SM '03: Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2003), ACM, pp. 127–134.

[47] Díaz, J. E. F. *Improvements in the Ray Tracing of Implicit Surfaces Based on Interval Arithmetic.* PhD thesis, Universitat de Girona, 2008.

[48] Do Carmo, M. P. *Riemannian Geometry.* Birkhäuser, Boston, 1992.

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

[50] Dong, S., Kircher, S., and Garland, M. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. In *Computer Aided Geometric Design* (2005).

[51] Eckstein, I., Pons, J.-P., Tong, Y., Kuo, C.-C. J., and Desbrun, M. Generalized surface flows for mesh processing. In *SGP* (2007), Eurographics Association, pp. 183–192.

[52] Edelsbrunner, H. Dynamic data structures for orthogonal intersection queries. Technical Report F59, Inst. Informationsverarb., Tech. Univ. Graz.

[53] Efremov, A., Havran, V., and Seidel, H.-P. Robust and numerically stable bézier clipping method for ray tracing nurbs surfaces. In *SCCG '05: Proceedings of the 21st Spring Conference on Computer Graphics* (New York, NY, USA, 2005), ACM, pp. 127–135.

[54] Elber, G. *Free form surface analysis using a hybrid of symbolic and numeric computation.* Ph.D. thesis, University of Utah, Computer Science Department, 1992.

[55] Elber, G., and Kim, M.-S. Geometric constraint solver using multivariate rational spline functions. In *SMA '01: Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2001), ACM, pp. 1–10.

[56] Entezari, A., Dyer, R., and Moller, T. Linear and cubic box splines for the body centered cubic lattice. In *VIS '04: Proceedings of the Conference on Visualization '04* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 11–18.

[57] Etzmuss, O., Keckeisen, M., and Strasser, W. A fast finite element solution for cloth modelling. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), PG '03, IEEE Computer Society, pp. 244–.

[58] Farhat, C., Geuzaine, P., and Grandmont, C. The discrete geometric conservation law and the nonlinear stability of ale schemes for the solution of flow problems on moving grids. *Journal of Computational Physics 174*, 2 (2001), 669 – 694.

[59] Field, D. A. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods 4*, 6 (1988), 709–712.

[60] Floater, M. Mean value coordinates. *Computer Aided Design 20*, 1 (2003), 19–27.

[61] Floater, M. S., and Hormann, K. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin, Eds. Springer Verlag, Berlin, Heidelberg, 2005, pp. 157–186.

[62] Foskey, M., Lin, M. C., and Manocha, D. Efficient computation of a simplified medial axis. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications* (2003), SM '03, pp. 96–107.

[63] Freitag, L., and Plassmann, P. Local optimization-based simplicial mesh untangling and improvement. Technical report. Mathematics and Computer Science Division, Argonne National Laboratory.

[64] Fried, I. Condition of finite element matrices generated from nonuniform meshes. *AIAA Journal 10*, 2 (1972), 219–221.

[65] Gabow, H. N., and Tarjan, R. E. Faster scaling algorithms for general graph matching problems. *J. ACM 38* (October 1991), 815–853.

[66] Giblin, P., and Kimia, B. On the local form and transitions of symmetry sets, medial axes, and shocks. *International Journal of Computer Vision 54*, 1 (2003), 143–157.

[67] Gordon, W. J. Spline-blended surface interpolation through curve networks. *Journal of Mathematics and Mechanics 18*, 10 (April 1969), 931–952.

[68] Goswami, S., Dey, T., and Bajaj, C. Identifying flat and tubular regions of a shape by unstable manifolds. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* (2006), ACM, pp. 27–37.

[69] Gregson, J., Sheffer, A., and Zhang, E. All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum (Special Issue of Symposium on Geometry Processing 2011) 30*, 5 (2011), to appear.

[70] Grimm, C. M., and Hughes, J. F. Modeling surfaces of arbitrary topology using manifolds. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), ACM Press, pp. 359–368.

[71] Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P. Discrete shells. In *SCA* (2003), Eurographics Association, pp. 62–67.

[72] Grinspun, E., Krysl, P., and Schröder, P. Charms: a simple framework for adaptive simulation. *ACM Transactions on Graphics 21*, 3 (July 2002), 281–290.

[73] Gu, X., He, Y., and Qin, H. Manifold splines. In *SPM '05: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2005), ACM, pp. 27–38.

[74] Gu, X., and Yau, S.-T. Global conformal surface parameterization. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 127–137.

[75] Guillard, H., and Farhat, C. On the significance of the geometric conservation law for flow computations on moving meshes. *Computer Methods in Applied Mechanics and Engineering 190*, 11-12 (2000), 1467 – 1482.

[76] Han, S., Xia, J., and He, Y. Hexahedral shell mesh construction via volumetric polycube map. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2010), SPM '10, ACM, pp. 127–136.

[77] Hart, J. C. Ray tracing implicit surfaces. In *Siggraph 93 Course Notes: Design, Visualization and Animation of Implicit Surfaces* (1993), pp. 1–16.

[78] Hauth, M., Etzmuss, O., and Strasser, W. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8 (2003), 581–600.

[79] He, Y., Wang, H., Fu, C.-W., and Qin, H. Technical section: a divide-and-conquer approach for automatic polycube map construction. *Comput. Graph. 33*, 3 (2009), 369–380.

[80] Heckbert, P. S., and Hanrahan, P. Beam tracing polygonal objects. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1984), ACM, pp. 119–127.

[81] Hormann, K., and Greiner, G. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, Eds., Innovations in Applied Mathematics. Vanderbilt University Press, Nashville, TN, 2000, pp. 153–162.

[82] Hormann, K., and Greiner, G. Quadrilateral remeshing. In *Proceedings of Vision, Modeling, and Visualization 2000* (Saarbrücken, Germany, Nov. 2000), B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, Eds., infix, pp. 153–162.

[83] Hormann, K., Lévy, B., and Sheffer, A. Mesh parameterization: theory and practice. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM Press, p. 1.

[84] Hua, J., He, Y., and Qin, H. Multiresolution heterogeneous solid modeling and visualization using trivariate simplex splines. In *SM '04: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications* (Aire-la-Ville, Switzerland, Switzerland, 2004), EG Association, pp. 47–58.

[85] Huang, J., Tong, Y., Wei, H., and Bao, H. Boundary aligned smooth 3D cross-frame field. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 143:1–143:8.

[86] Hughes, T. J. R. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover, 2000.

[87] Hughes T.J., Cottrell J.A., B. Y. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering 194* (2005), 4135–4195.

[88] II, J. D. D., and Cohen, E. Surface creation and curve deformations between two complex closed spatial spline curves. In *Springer-Verlag Lecture Notes in Computer Science 4077 (GMP 2006)* (2006), pp. 221–234.

[89] Ito, Y., Shih, A. M., and Soni, B. K. Hybrid mesh generation with embedded surfaces using a multiple marching direction approach. *International Journal for Numerical Methods in Fluids 67*, 1 (2011), 1–7.

[90] Jaillet, F., Shariat, B., and Vandorpe, D. Periodic b-spline surface skinning of anatomic shapes. In *9th Canadian Conference in Computational Geometry* (1997).

[91] Johnson, D. E., Martin, T., and Cohen, E. Computing constrained laplacian navigation function paths in configuration space. *ASME Conference Proceedings 2010*, 44106 (2010), 1409–1416.

[92] Joshi, P. *Minimizing Curvature Variation for Aesthetic Surface Design*. PhD thesis, EECS Department, University of California, Berkeley, Oct 2008.

[93] Ju, T., Baker, M., and Chiu, W. Computing a family of skeletons of volumetric models for shape description. *Computer-Aided Design 39*, 5 (2007), 352–360.

[94] Kajiya, J. T. Ray tracing parametric patches. In *SIGGRAPH '82: Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1982), ACM, pp. 245–254.

[95] Kalbe, T., and Zeilfelder, F. Hardware-accelerated, high-quality rendering based on trivariate splines approximating volume data. *Comput. Graph. Forum 27*, 2 (2008), 331–340.

[96] Kallinderis, Y., Khawaja, A., and Mcmorris, H. Hybrid prismatic/tetrahedral grid generation for complex geometries. *AIAA Paper 34* (1996), 93–0669.

[97] Karátson, J., and Faragó, I. Preconditioning operators and Sobolev gradients for nonlinear elliptic problems. *Comput. Math. Appl. 50* (October 2005), 1077–1092.

[98] Karniadakis, G., and Sherwin, S. *Spectral/hp element methods for CFD - 2^{nd} Edition.* Oxford University Press, UK, 2005.

[99] Khawaja, A., and Kallinderi, Y. Hybrid grid generation for turbomachinery and aerospace applications. *International Journal for Numerical Methods in Engineering 49*, 1 (September 2000), 145–166.

[100] Kim, B., and Rossignac, J. GeoFilter: geometric selection of mesh filter parameters. *Computer Graphics Forum 24*, 3 (Sept. 2005), 295–302.

[101] Kim, M., Entezari, A., and Peters, J. Box spline reconstruction on the face-centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1523–1530.

[102] Kirby, R. M., Yosibash, Z., and Karniadakis, G. E. Towards stable coupling methods for high-order discretization of fluid-structure interaction: algorithms and observations. *J. Comput. Phys. 223* (May 2007), 489–518.

[103] Klein, P. N., Sebastian, T. B., and Kimia, B. B. Shape matching using edit-distance: an implementation. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms* (2001), SODA '01, pp. 781–790.

[104] Kloetzli, J., Olano, M., and Rheingans, P. Interactive volume isosurface rendering using bt volumes. In *I3D '08: Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2008), ACM, pp. 45–52.

[105] Knoll, A., Hijazi, Y., Hansen, C. D., Wald, I., and Hagen, H. Interactive ray tracing of arbitrary implicit functions. In *Proceedings of the 2007 Eurographics/IEEE Symposium on Interactive Ray Tracing* (2007).

[106] Knoll, A., Wald, I., Parker, S., and Hansen, C. Interactive isosurface ray tracing of large octree volumes. *Interactive Ray Tracing 2006, IEEE Symposium on* (Sept. 2006), 115–124.

[107] Knupp, P. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i - a framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering 48*, 3 (2000), 401–420.

[108] Knupp, P. M. Algebraic mesh quality metrics. *SIAM J. Sci. Comput. 23*, 1 (2001), 193–218.

[109] Krawczyk, R. Newton algorithmen zur bestimmung von nullstellen mit fehlerschranken. *Computing 4* (1969), 187–201.

[110] Lazarus, F., and Verroust, A. Level set diagrams of polyhedral objects. In *SMA '99: Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 1999), ACM Press, pp. 130–140.

[111] Levoy, M. Efficient ray tracing of volume data. *ACM Trans. Graph. 9*, 3 (1990), 245–261.

[112] Lévy, B., and Liu, Y. Lp centroidal voronoi tessellation and its applications. *ACM Trans. Graph. 29* (July 2010), 119:1–119:11.

[113] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. Least squares conformal maps for automatic texture atlas generation. vol. 21, ACM Press, pp. 362–371.

[114] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (July 2002), 362–371.

[115] Li, X., Guo, X., Wang, H., He, Y., Gu, X., and Qin, H. Harmonic volumetric mapping for solid modeling applications. In *Symposium on Solid and Physical Modeling* (2007), pp. 109–120.

[116] Lipton, S., Evans, J., Bazilevs, Y., Elguedj, T., and Hughes, T. Robustness of isogeometric structural discretizations under severe mesh distortion. *Computer Methods in Applied Mechanics and Engineering 199*, 5-8 (2010), 357 – 373. Computational Geometry and Analysis.

[117] Loop, C. Smooth spline surfaces over irregular meshes. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM Press, pp. 303–310.

[118] Loop, C., and Blinn, J. Real-time GPU rendering of piecewise algebraic surfaces. *ACM Trans. Graph. 25*, 3 (2006), 664–670.

[119] Lorensen, W. E., and Cline, H. E. Marching cubes: a high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169.

[120] Lyche, T., and Morken, K. A data reduction strategy for splines with applications to the approximation of functions and data. *IMA J. of Numerical Analysis 8*, 2 (1988), 185–208.

[121] Marschner, S. R., and Lobb, R. J. An evaluation of reconstruction filters for volume rendering. In *VIS '94: Proceedings of the Conference on Visualization '94* (Los Alamitos, CA, USA, 1994), IEEE Computer Society Press, pp. 100–107.

[122] Marsden, J. E., and Hughes, T. J. R. *Mathematical Foundations of Elasticity.* Dover Publications, New York, 1994.

[123] Marsden, M. J. An identity for spline functions with applications to variation diminishing spline approximation. *J. Approx. Theory 3* (1970), 7–49.

[124] Martin, T., and Cohen, E. Volumetric parameterization of complex objects by respecting multiple materials. *Computers & Graphics 34*, 3 (2010), 187 – 197. Shape Modelling International (SMI) Conference 2010.

[125] Martin, T., Cohen, E., and Kirby, M. A comparison between isogeometric analysis versus fem applied to a femur. *to be submitted* (2008).

[126] Martin, T., Cohen, E., and Kirby, M. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *SPM '08: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2008), ACM, pp. 269–280.

[127] Martin, W., and Cohen, E. Representation and extraction of volumetric attributes using trivariate splines. In *Symposium on Solid and Physical Modeling* (2001), pp. 234–240.

[128] Martin, W., and Cohen, E. Surface completion of an irregular boundary curve using a concentric mapping. In *Proceedings of the Fifth Conference on Curves and Surfaces* (2003), Nashboro Press, pp. 293–302.

[129] Mehlum, E., and Tarrou, C. Invariant smoothness measures for surfaces. *Advances in Computational Mathematics 8*, 1-2 (1998), 49–63.

[130] Meyer, M., Nelson, B., Kirby, R., and Whitaker, R. Particle systems for efficient and accurate high-order finite element visualization. *Visualization and Computer Graphics, IEEE Transactions on 13*, 5 (Sept.-Oct. 2007), 1015–1026.

[131] Miklos, B., Giesen, J., and Pauly, M. Discrete scale axis representations for 3d geometry. *ACM Trans. Graph. 29* (July 2010), 101:1–101:10.

[132] Milnor, J. Morse theory. In *Annual of Mathematics Studies* (Princeton, NJ, 1963), vol. 51, Princeton University Press.

[133] Mohsen, and Rezayat. Midsurface abstraction from 3d solid models: general theory and applications. *Computer-Aided Design 28*, 11 (1996), 905 – 915.

[134] Moore, R. E. *Interval analysis.* Prentice Hall, Upper Saddle River, NJ 07458, USA, 1966.

[135] Morken, K. Products of splines as linear combinations of b-splines. *Constructive Approximation 7*, 1 (1991), 195–208.

[136] Müller, M., Dorsey, J., McMillan, L., Jagnow, R., and Cutler, B. Stable real-time deformations. In *SCA* (2002), pp. 49–54.

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

[138] Musuvathy, S. *Medial Axis of Regions Bounded by B-spline Curves and Surfaces*. PhD dissertation, University of Utah, 2011.

[139] Musuvathy, S., Cohen, E., and Damon, J. Computing medial axes of generic 3d regions bounded by b-spline surfaces. *Comput. Aided Des. 43* (November 2011), 1485–1495.

[140] Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M. FiberMesh: designing freeform surfaces with 3D Curves. *ACM Transactions on Graphics 26*, 3 (July 2007), 41:1–41:9.

[141] Nealen, A., Mueller, M., Keiser, R., Boxerman, E., and Carlson, M. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum 25*, 4 (Dec. 2006), 809–836.

[142] Nelson, B., and Kirby, R. M. Ray-tracing polymorphic multidomain spectral/hp elements for isosurface rendering. *IEEE Transactions on Visualization and Computer Graphics 12*, 1 (2006), 114–125.

[143] Ni, X., Garland, M., and Hart, J. C. Fair morse functions for extracting the topological structure of a surface mesh. In *Proc. SIGGRAPH* (2004).

[144] Nieser, M., Reitebuch, U., and Polthier, K. CubeCover - parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406.

[145] Nishita, T., Sederberg, T. W., and Kakimoto, M. Ray tracing trimmed rational surface patches. *SIGGRAPH Comput. Graph. 24*, 4 (1990), 337–345.

[146] Nocedal, J. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation 35*, 151 (1980), 773–782.

[147] Oddy, A., Goldak, J., McDill, M., and Bibby, M. A distortion metric for isoparametric finite elements. *Transactions of CSME 12*, 4 (1988), 213–217.

[148] Ohtake, Y., Belyaev, A., and Seidel, H.-P. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 609–612.

[149] Okazaki, N. libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), 2010. http://www.chokkan.org/software/liblbfgs/.

[150] Owen, S. J. A survey of unstructured mesh generation technology. In *International Meshing Roundtable* (1998), pp. 239–267.

[151] Owen, S. J., and Saigal, S. H-morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering 49*, 1-2 (2000), 289–312.

[152] Paik, K. L. Trivariate b-splines. Master's thesis, Department of Computer Science, University of Utah, June 1992.

[153] Paiva, A., Lopes, H., Lewiner, T., and de Figueiredo, L. H. Robust adaptive meshes for implicit surfaces. *Computer Graphics and Image Processing, Brazilian Symposium on 0* (2006), 205–212.

[154] Pardhanani, A., and Carey, G. F. Optimization of computational grids. *Numerical Methods for Partial Differential Equations 4*, 2 (1988), 95–117.

[155] Parker, S., Shirley, P., Livnat, Y., Hansen, C., and Sloan, P.-P. Interactive ray tracing for isosurface rendering. In *VIS '98: Proceedings of the Conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 233–238.

[156] Parthasarathy, V. N., Graichen, C. M., and Hathaway, A. F. A comparison of tetrahedron quality measures. *Finite Elem. Anal. Des. 15*, 3 (1994), 255–261.

[157] Peng, J., Kristjansson, D., and Zorin, D. Interactive modeling of topologically complex geometric detail. *ACM Trans. Graph. 23* (August 2004), 635–643.

[158] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics 72*, 2 (1987), 449–466.

[159] Piegl, L., and Tiller, W. A menagerie of rational b-spline circles. *IEEE Comput. Graph. Appl. 9*, 5 (1989), 48–56.

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

[161] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[162] Puso, M. A., Laursen, T., and Solberg, J. A segment-to-segment mortar contact method for quadratic elements and large deformations. *Computer Methods in Applied Mechanics and Engineering 197*, 6-8 (2008), 555 – 566.

[163] Ramanathan, M., and Gurumoorthy, B. Constructing medial axis transform of planar domains with curved boundaries. *Computer-Aided Design 35*, 7 (2003), 619–632.

[164] Raviv, A., and Elber, G. Interactive direct rendering of trivariate b-spline scalar functions. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (2001), 109–119.

[165] Ray, N., Li, W. C., Lévy, B., Sheffer, A., and Alliez, P. Periodic global parameterization. *ACM Trans. Graph. 25*, 4 (2006), 1460–1485.

[166] Reimers, M., and Seland, J. Ray casting algebraic surfaces using the frustum form. *Comput. Graph. Forum 27*, 2 (2008), 361–370.

[167] Renka, R. J. Constructing fair curves and surfaces with a Sobolev gradient method. *Computer Aided Geometric Design 21*, 2 (2004), 137 – 149.

[168] Renka, R. J., and Neuberger, J. W. Minimal surfaces and Sobolev gradients. *SIAM J. Sci. Comput. 16* (November 1995), 1412–1427.

[169] Rivara, M.-C. Local modification of meshes for adaptive and/or multigrid finite-element methods. *Journal of Computational and Applied Mathematics 36*, 1 (1991), 79–89. Special Issue on Adaptive Methods.

[170] Sampl, P. Semi-structured mesh generation based on medial axis. In *IMR* (2000), pp. 21–32.

[171] Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. Texture mapping progressive meshes. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 409–416.

[172] Schenk, O., and Gärtner, K. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Gener. Comput. Syst. 20* (April 2004), 475–487.

[173] Schreiner, J., and Scheidegger, C. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1205–1212. Member-Claudio Silva.

[174] Schreiner, J., Scheidegger, C., Fleishman, S., and Silva, C. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum (Proceedings of Eurographics 2006) 25*, 3 (2006), 527–536.

[175] Schwab, C. *p- and hp- Finite Element Methods: Theory and Applications to Solid and Fluid Mechanics.* Oxford University Press (USA), 1999.

[176] Sederberg, T., and Zundel, A. Pyramids that bound surface patches. *GMIP 58*, 1 (January 1996), 75–81.

[177] Sederberg, T. W., Zheng, J., Bakenov, A., and Nasri, A. T-splines and t-nurccs. *ACM Trans. Graph. 22*, 3 (2003), 477–484.

[178] Sederberg T.W., Zheng J., N. A. T-splines and t-nurccs. In *Proc. SIGGRAPH* (2003).

[179] Sheffer, A., and de Sturler, E. Surface parameterization for meshing by triangulation flattening. *Proc. 9th International Meshing Roundtable, pages 161–172, 2000.* (2000).

[180] Sheffer, A., Etzion, M., Rappoport, A., and Bercovier, M. Hexahedral mesh generation using the embedded voronoi graph. In *In Proceedings of the 7th International Meshing Roundtable* (1998), pp. 347–364.

[181] Sheffer, A., Praun, E., and Rose, K. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision 2*, 2 (2006).

[182] Shewchuk, J. R. What is a good linear element? interpolation, conditioning, and quality measures. *Eleventh International Meshing Roundtable* (2002), 115–126.

[183] Shi, L., Yu, Y., Bell, N., and Feng, W.-W. A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graphics 25*, 3 (July 2006), 1108–1117.

[184] Shinagawa, Y., Kunii, T. L., and Kergosien, Y. L. Surface coding based on morse theory. *IEEE Comput. Graph. Appl. 11*, 5 (1991), 66–78.

[185] Shirley, P. *Fundamentals of Computer Graphics.* A. K. Peters, Ltd., Natick, MA, USA, 2002.

[186] Si, H. Tetgen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. http://tetgen.berlios.de/, 2005.

[187] Siddiqi, K., and Pizer, S. *Medial Representations: Mathematics, Algorithms and Applications.* Springer-Verlag, 2008. 439pp.

[188] Siddiqi, K., Shokoufandeh, A., Dickinson, S. J., and Zucker, S. W. Shock graphs and shape matching. In *Proceedings of the Sixth International Conference on Computer Vision* (1998), ICCV '98, IEEE Computer Society, pp. 222–.

[189] Sorkine, O., Cohen-Or, D., Goldenthal, R., and Lischinski, D. Bounded-distortion piecewise mesh parameterization. In *VIS* (2002), IEEE Computer Society, pp. 355–362.

[190] Sud, A., Foskey, M., and Manocha, D. Homotopy-preserving medial axis simplification. In *SPM '05: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2005), ACM, pp. 39–50.

[191] Sundaramoorthi, G., Yezzi, A., and Mennucci, A. C. Sobolev active contours. *International Journal of Computer Vision 73* (2005), 109–120.

[192] Sung, K., and Shirley, P. *Ray Tracing with the BSP Tree.* Academic Press Professional, Inc., San Diego, CA, USA, 1992, pp. 271–274.

[193] Tarini, M., Hormann, K., Cignoni, P., and Montani, C. Polycube-maps. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 853–860.

[194] Taubin, G. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 95* (Aug. 1995), Computer Graphics Proceedings, Annual Conference Series, pp. 351–358.

[195] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (July 1987), pp. 205–214.

[196] Tong, Y., Alliez, P., Cohen-Steiner, D., and Desbrun, M. Designing quadrangulations with discrete harmonic forms. In *ACM/EG Symposium on Geometry Processing* (2006).

[197] Toth, D. L. On ray tracing parametric surfaces. In *SIGGRAPH '85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1985), ACM, pp. 171–179.

[198] Venditti, D. A., and Darmofal, D. L. Adjoint error estimation and grid adaptation for functional outputs: application to quasi-one-dimensional flow. *Journal of Computational Physics 164*, 1 (2000), 204–227.

[199] Verroust, A., and Lazarus, F. Extracting skeletal curves from 3d scattered data. *The Visual Computer 16*, 1 (2000), 15–25.

[200] Wang, H., He, Y., Li, X., Gu, X., and Qin, H. Polycube splines. In *SPM '07: Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2007), ACM, pp. 241–251.

[201] Wang, H., Jin, M., He, Y., Gu, X., and Qin, H. User-controllable polycube map for manifold spline construction. In *SPM '08: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2008), ACM, pp. 397–404.

[202] Wang, Y., Gu, X., Thompson, P. M., and Yau, S.-T. 3d harmonic mapping and tetrahedral meshing of brain imaging data. In *Proc. Medical Imaging Computing and Computer Assisted Intervention (MICCAI), St. Malo, France, Sept. 26-30 2004* (2004).

[203] Wang, Y., and Murgie, S. Hybrid mesh generation for viscous flow simulation. In *Proceedings of the 15th International Meshing Roundtable*, P. P. Pbay, Ed. Springer Berlin Heidelberg, 2006, pp. 109–126.

[204] Willmore, T. J. Mean curvature of riemannian immersions. *Journal of the London Mathematical Society s2-3*, 2 (1971), 307–310.

[205] Wilson, O., VanGelder, A., and Wilhelms, J. Direct volume rendering via 3d textures. Tech. rep., University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

[206] Y. Maday, C. Mavriplis, A. P. Non-conforming mortar element methods: applications to spectral discretizations. *Domain Decompostion Methods, SIAM* (1989).

[207] Yamakawa, S., and Shimada, K. Converting a tetrahedral mesh to a prism-tetrahedral hybrid mesh for FEM accuracy and efficiency. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2008), SPM '08, ACM, pp. 287–294.

[208] Zhang, E., Mischaikow, K., and Turk, G. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics 24*, 1 (Jan. 2005), 1–27.

[209] Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C. L., and Hughes, T. J. R. Patient-specific vascular nurbs modeling for isogeometric analysis of blood flow. In *Proc. 15th Int. Meshing Roundtable* (2006), Springer, Berlin, pp. 73–92.