DYNAMIC PARTICLES FOR ADAPTIVE SAMPLING OF IMPLICIT SURFACES

by

Miriah Meyer

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

August 2008

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Miriah Meyer

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Ross Whitaker

Elaine Cohen

Mike Kirby

Hanspeter Pfister

Claudio Silva

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Miriah Meyer in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Ross Whitaker
Chair, Supervisory Committee

Approved for the Major Department

Martin Berzins
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

A ubiquitous requirement in many mathematical and computational problems is a set of well-placed point samples. For producing very even distributions of samples across complex surfaces, a dynamic particle system is a controllable mechanism that naturally accommodates strict sampling requirements. The system first constrains particles to a surface, and then moves the particles across the surface until they are arranged in minimal energy configurations. Adaptivity is added into the system by scaling the distance between particles, causing higher densities of points around surface features. In this dissertation we explore and refine the dynamics of particle systems for generating efficient and adaptive point samples of implicit surfaces.

Throughout this dissertation, we apply the adaptive particle system framework to several application areas. First, efficient visualizations of high-order finite element datasets are generated by developing adaptivity metrics of surfaces that exist in the presence of curvilinear coordinate transformation. Second, a framework is proposed that meets fundamental sampling constraints of Delaunay-based surface reconstruction algorithms. In meeting these constraints, the particle distributions produce nearly-regular, efficient isosurface tessellation that are geometrically and topologically accurate. And third, a novel analytic representation of material boundaries in multimaterial volume datasets is developed, as well as a set of projection operators, that allow for explicit sampling of nonmanifold material intersections. Using a tetrahedral labeling algorithm, the material intersections are extracted as watertight, nonmanifold meshes that are well-suited for simulations.

To my cheerleading squad.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am deeply appreciative of the commitment of my advisor, Ross Whitaker, to ensuring that I obtained the skills necessary for asking, and answering, tough questions. From helping to debug code, to hashing out ideas on the white board, to debating the proper use of hyphens, Ross's involvement with each step of the research process provided me with consistent and thoughtful guidance that is a rare educational experience. Ross also has an incessant demand for excellence that, although overwhelming and frustrating at times, has left me with an immense sense of satisfaction in my own work and abilities. I could not have imagined a more effective advisor and mentor.

My years at the U were also enriched by numerous other people. Mike Kirby provided much guidance and support, especially during my last year. His insistence on mathematical rigor helped me to more effectively realize and communicate ideas, and to also make sense of months worth of analysis and data related to the multimaterial meshing work in this dissertation. Claudio Silva provided the initial idea and subsequent encouragement for pursuing the particle-based meshing work, and his insights blossomed into a large part of my dissertation research. Also, none of my graduate school experience would have been possible without the amazing work environment that Chris Johnson has fostered in the SCI Institute. I feel incredibly lucky to have experienced Chris's vision for an open, productive, and collaborative research group, and to have benefited from his years of hard work putting it all together. And finally, it was the magic of Hanspeter Pfister's teaching that pointed me down this whole path to begin with, and his poking and prodding that got me to wrap it all up in the end.

I would like to acknowledge the vast amount of research and effort that went into generating the data sets used in this dissertation. I would like to thank Mark Ellisman, Matthew Jolley, Tolga Tasdizen, Oliver Nemitz, and David Breen.

Completing this dissertation has been a long and arduous journey, and it has been the support of my friends and family that kept me motivated, focused, and energized. I cannot say thank you enough to everyone who has been involved with SCI since my arrival. I would especially like to thank Gordon Kindlmann, Milan Ikits and Dave Weinstein for providing invaluable advice on navigating my way through the PhD process — everything they told me came true at one point

along the way. Christiaan Gribble and Aaron Lehfon were incredible study partners, as well as constant sounding boards for all issues related to school and life.

A large part of my happiness over the years was fostered by the ladies of Ladies Night — Betty Mohler, Kristi Potter, and Liz Jurrus. In these three women I found never-ending support, a wealth of wisdom, strength, and advice, and a consistent source of laughter. Our personalities meshed in an amazingly symbiotic way that I do not expect, but hope, to experience again one day.

Finally, I would like to thank my family, in Utah and in Virginia, for believing in me the whole way through. Day in and day out, Aaron Campbell encouraged me and listened to me, while providing a cozy home and amazing meals to end each day with. With him I learned how to be a better friend, a better partner, and a better wine-drinker. The vast space of the rest of my life has been supported by my parents, Bill and Suki, who have made sure that I had, and have, every opportunity in the world. I owe my independence, my skills, and my education to their love and encouragement. Mom and Dad, thank you for helping me become the person that I am.

CHAPTER 1

INTRODUCTION

This dissertation is about manipulating points that sample an implicit surface. By minimizing an energy function associated with the point distribution, a wide range of sample patterns can be achieved, from homogeneous distributions to those that are highly adaptive. The point set is described by a *particle system*, which constrains a set of dynamic particles to the surface while simultaneously associating potential energy functions with each particle that induce interparticle repulsive forces, pushing the particles across the surface. The system iteratively moves the particles along the induced force field to locally minimize the energy distributions, resulting in a regular distribution of points over the entire surface. Furthermore, the potential energy kernels can be manipulated to create higher densities of points around interesting surface features. Robust and controllable, the particle system is an implicit surface sampling strategy that can be configured to meet the needs of a broad range of applications, from visualization to mesh generation.

Because of their volumetric expression, implicit functions are a natural representation for three-dimensional (3D) digital data representing the physical world, whether those data are acquired through a simulation or through a measuring device like magnetic resonance imaging (MRI) machines. Implicit functions characterize a volume by defining whether a point is on one side of the surface or the other, implicitly specifying the surface as a constraint. These surfaces can be interpreted as a thin band of some value embedded within a volume, like contour lines on a map for indicating paths of constant altitude over a landscape, illustrated in Figure 1.1.

Oftentimes, data from scanning devices or simulations come in the form of a volumetric lattice of values, which can then be interpreted as an implicit function. In MRI data, density values for different tissue types in a patient's head, for example, are stored over a regular grid and can be used to deduce boundaries between the skull and the brain, or the brain and a tumor. Similarly, vortices in data computed from a computational fluid dynamics simulation can be located by finding surfaces of constant pressure, inferred from pressure values stored at simulation data points. The ability to locate and visualize these implicitly defined surfaces allows scientists to study effects that would otherwise be too costly, or even impossible, to create experimentally

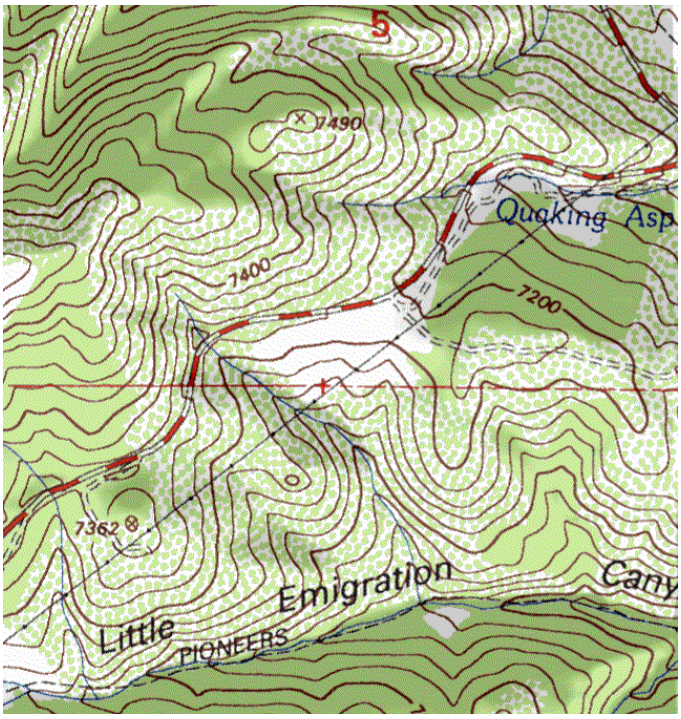


Figure 1.1. Implicit surfaces are implied bands of some constant value embedded within a volume, much like lines of constant altitude in a topographic map.

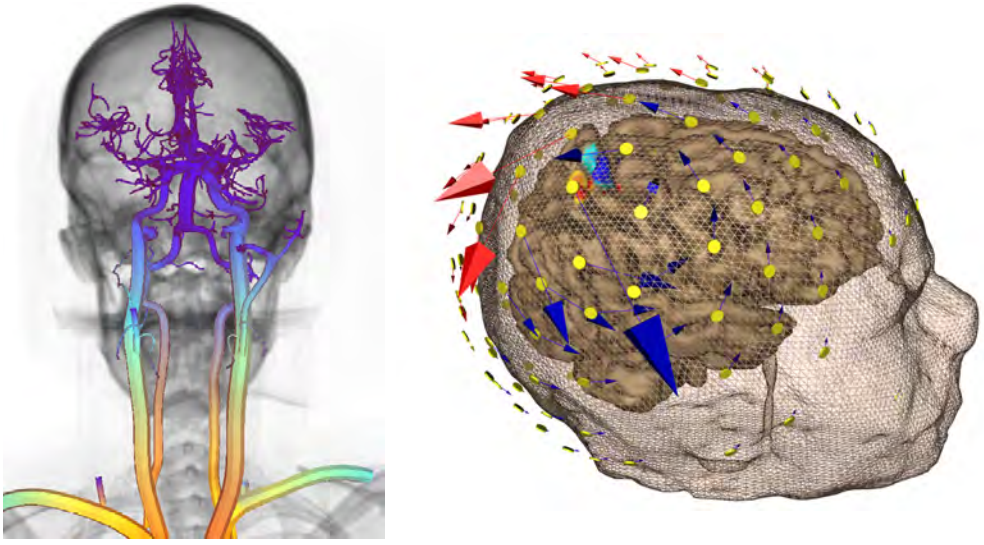
while similarly providing physicians a noninvasive lens into the physiology and pathology of their patients. Figure 1.2 provides examples of implicit surface visualizations.

Advances in the acquisition of digital data, as well as the associated ability to extract the implicit surfaces defined within, has also enabled patient-specific simulation models to greatly increase the accuracy and relevance of biomedical simulations. Simulations of bioelectric fields [151], cardiovascular fluid dynamics [144, 34], and implanted medical devices like cardiac defibrillators [1], to name a few, have benefited from technology that captures the individual geometry and pathology of a patient. To generate these models, digital data of a patient is processed to extract the implicit surface of interest, which is then parameterized into an explicit surface representation, such as a polygonal mesh like those shown in Figure 1.3. The parameterized surface model can then be used directly to study phenomena that occur over the surface, or, extended to a volumetric representation suitable for biomedical simulations.

Visualization, analysis, and extraction of implicit surfaces are all computationally challenging problems due to the lack of an explicit surface representation. Locating the surface is a root-finding problem; the surface exists as the zero crossings of a scalar function. For implicit functions that have a closed-form inverse, root-finding can be accomplished efficiently using first-

(a) (b)

Figure 1.2. Examples of implicit surface visualizations: (a) a volume rendering of a tumor extracted from MRI data using a levelset surface deformation technique [85] (image courtesy of Aaron Lefohn); (b) a raytraced image of a helium plume computed from a computational fluid dynamics simulation (image courtesy of the University of Utah Center for Accidental Fires and Explosions (C-SAFE)).



(a) (b)

Figure 1.3. Examples of implicit surface extractions for biomedical applications: (a) vasculature extraction for use in a catheter-placement simulation [34] (image used without permission); (b) patient-specific model used in an EKG simulation [151] (image used without permission).

order approximations schemes like Newton-Raphson or subdivision algorithms. More complex functions, however, like those embedded in curvilinear coordinate systems [68], require computationally intensive numerical algorithms that do not easily extend to traditional visualization and surface extraction schemes.

Locating the surface of interest embedded in an implicit function for visualization or extraction inheriently relies on effective and accurate sampling schemes. Efficacious sampling of an implicit surface comes down to how well a set of discrete points approximates the underlying continuous geometry to meet the needs of an application. In visualization, capturing small features is important for providing accurate illustrations of the data. Many traditional visualization techniques rely on piecewise-linear approximations, thus faithfully representing a curved surface thar requires a dense enough sampling that the approximation captures the topology and geometry of the surface.

A straightforward approach to adequately sampling an implicit surface is to generate a large number of samples everywhere on the surface, ensuring, by shear number, that all of the features are captured. Algorithms for producing such surface points are generally simple to implement and fast to execute, such as supersampling a surface over a finely subdivided lattice. The drawback of such methods is that most parts of the surface will be sampled by far more points then necessary, which can be particularly inefficient when the root-finding is computationally costly. A counter approach is to purposefully place the surface points according to characteristics of the implicit surface, such as generating sparser sets across flat areas. Methods that produce these adaptive point distributions are typically more computationally intensive and complex, but ultimately produce more efficient and compact representations of the underlying implicit surface.

Extracting a surface mesh for biomedical simulations further requires that the point samples be regularly spaced such that the resulting polygonalization contains well-shaped elements. In finite element simulations, the *condition number*, which is the value that quantifies how numerically well-behaved the simulation will be, is often dictated by the most poorly shaped elements — for triangular meshes, a well-shaped element is as close to an equilateral triangle as possible. Thus, sampling a surface for finite element mesh generation will ideally result in hexagonal patterns of points.

The fundamental goal of this dissertation is to provide a robust and controllable framework for adaptive sampling of implicit surfaces, specifically for the generation of efficient visualizations and high-quality polygonal representations of scientific and biomedical data. The framework posed in Chapter 3 uses a system of dynamic particles that are constrained to an implicit surface

and minimize a global energy configuration. By employing specific types of potential energy kernels, the system easily accommodates numerous optimization schemes that efficiently distribute large numbers of particles while requireing that very few parameters be tuned by the user. Furthermore, the addition of a space-warping scheme provides a tunable handle for a user to create highly controllable adaptive distributions. Combined with a simple set of mathematical constraints based on the geometry of the surface, the system can generate complex, adaptive, and regular distributions of points across arbitrarily shaped surfaces.

Using this new particle system framework, high-quality, direct visualizations of implicit surfaces can be produced for even the most complex of data sets. For example, Chapter 4 presents results from visualizing isosurfaces embedded in high-order finite element data sets. These data sets pose numerous challenges for traditional visualization techniques, but can be sampled naturally and efficiently using a particle system by warping space aniostropically according to surface metrics computed from tensor product quantities. The flexibility and controllability of dynamic particle systems also allows for a framework where particles sample the finite element basis functions directly, avoiding costly numerical inversions of high-order mapping functions which have no closed-form analytical inverse.

In Chapter 5, a pipeline that exploits the regularity of the particle distribution is presented for generating geometrically and topologically accurate surface meshes. By combining the particle system framework with surface sampling theory and PDE-based methods for controlling the local variability of particle densities, the meshing pipeline is shown to create high-quality meshes of implicit surfaces. Extending this pipeline, Chapter 6 presents a mathematical framework for describing the nonmanifold material intersections in multimaterial volumes that are then sampled with dynamic particles. The use of multiple, interacting particle systems allows for explicit sampling of topological points, lines, and planes, a feature which is then used to produce consistent meshes of intersection materials. These meshes of abutting materials are important for simulations that span multiple layers and boundaries, such as models for propagating electrical signals through the body [1].

1.1 Contributions

This dissertation seeks to provide a robust and controllable method for adaptively sampling implicit surfaces using a system of dynamic particles. In meeting the stated goal, this dissertation provides several contributions:

- *A method for adding controllable adaptivity into a dynamic particle system framework.* By combining carefully designed potential functions for controlling particle motions with mechanisms for adding and removing particles, a novel adaptivity framework is presented that alters the effective distance between particles for precise control of interparticle distances (Chapter 3) [93, 95].

- *A framework for adaptively and efficiently distributing particles in the presence of curvilinear coordinate transforms.* The controllability of the particle system is employed to adapt particle distributions to surface geometry that is influenced by curvilinear coordinate transformations. Furthermore, the particles remain in the space in which the basis functions are defined, avoiding costly inversions of coordinate transformations which (generally) have no analytic inverse. This framework is shown to be effective for visualizing data produced by high-order finite element simulations (Chapter 4) [95].

- *A pipeline for generating isosurface triangulations suitable for simulations.* Using fundamental work in computational geometry and PDE-based methods, particles can be distributed such that the resulting set of surface samples will generate nearly-regular Delaunay triangulations of volumetric data (Chapter 5) [94].

- *A mathematical representation of nonmanifold material junctions in multimaterial volumes, and a systematic approach for sampling these junctions for tessellations of the material interfaces which are suitable for simulations.* The development of novel, analytic functions for describing nonmanifold junctions of materials, as well as a corresponding set of projection operators, allows particle systems to explicitly sample corners, edges, and surfaces of material intersections. The resulting point distributions meet fundamental sampling constraints, allowing Delaunay-based meshing algorithms to reliably extract watertight meshes that include sharp features (Chapter 6) [96].

1.2 Overview

Chapter 2 is devoted to a brief overview of implicit surfaces and their history in the fields of computer graphics, scientific visualization, and computational geometry. The aim of that chapter is to establish a mathematical description of implicits that we will use throughout this dissertation, as well as an overview of reconstruction methods used to produce smooth, differentiable implicit functions from volume data (Section 2.4). Also included are several relevant concepts from the image processing literature for preprocessing segmented volumes to generate implicit functions

with controlled feature sizes (Section 2.5).

Chapter 3 provides a detailed discussion of the proposed particle system framework and its benefits over other methods posed in the literature (Section 3.2). We present new energy functions and algorithms that guide the dynamic particles (Sections 3.3 and 3.4), as well as techniques for adapting and controlling the distribution of particles (Section 3.5). The chapter concludes with a comprehensive implementation outline (Section 3.6) that will guide implemention discussions in later chapters.

Chapter 4 adapts the basic particle system framework for the visualization of isosurfaces embedded in high-order finite element data. After presenting previous work on isosurface visualization, including that specific to high-order finite elements, we present a thorough investigation on the difficulties of adapting traditional, piecewise-linear visualization techniques to these data sets (Section 4.2). Next, a particle system framework for sampling high-order finite element isosurfaces is proposed (Section 4.3) that includes a formulation of a curvature-based adaptivity metric for surfaces embedded in curvilinear coordinate systems (Section 4.3.1). Finally, we compare results generated using the particle system approach against other visualization algorithms, indicating that the proposed method is an efficient scheme for creating accurate visualizations of these challenging data sets (Section 4.6).

Chapter 5 describes a method for constructing isosurface triangulations of sampled, volumetric, 3D scalar fields by combining the particle system with surface sampling theory and PDE-based methods. The chapter first introduces previous work on generating isosurface meshes that are suitable for simulations, which is followed by a discussion of the sampling constraints posed by Delaunay-based reconstruction algorithms (Section 5.2). We then present a pipeline that outlines a scheme for generating particle distributions that conform to the sampling constraints (Section 5.3), followed by an analysis of results (Section 5.5).

Chapter 6 proposes a method for constructing geometrically accurate, nonmanifold tessellations of material intersections in multimaterial volumes. We develop a novel, functional representation of material junctions, along with a set of projection operators, such that each material junction is explicitly sampled with a dynamic particle system (Section 6.4). When sharp features in the data are explicitly sampled, we show that Delaunay-based meshing algorithms can be used to generate watertight, nonmanifold tessellations of the intersections of multimaterial datasets (Sections 6.4.3 and 6.5). The chapter concludes with results that indicate the proposed method reliably generates meshes that are well-suited for simulations (Section 6.7).

The final chapter of this dissertation presents preliminary results from extensions to the parti-

cle system and mesh generation frameworks. Included in this chapter are methods for generating quad-packings of particles, for generating sphere-packings towards the creation of tetrahedral meshes, and future avenues of work.

CHAPTER 2

TECHNICAL BACKGROUND

Although this dissertation is about *sampling* implicit surfaces, the ideas herein fundamentally rely on the description and mathematical properties of implicit functions. This chapter briefly describes and reviews implicit functions as they are used in the computer graphics and image processing literature. Details on constructing implicit functions, as well as processing volume data to control the geometry of the levelsets,are is also included.

2.1 Notation

Our notation for this disseration is as follows: bold face lower-case variables denote column vectors, such as $\mathbf{x} = [x\ y\ z]^\mathsf{T}$, while bold face upper-case variables denote matrices, such as the identity matrix $\mathbf{I}$; bold face subscripts denote partial derivatives of the function with respect to each component of the subscripted column vector, such as $F_\mathbf{x} = [\frac{\partial F}{\partial x}\ \frac{\partial F}{\partial y}\ \frac{\partial F}{\partial z}]^\mathsf{T}$; $\mathbf{x}_i$ specifies the position of the i-th particle, and other nonbold subscripts denote the evaluation of a scalar function at a specific particle's location, such as E_i. We stray from this notation only in Section 4.3.1, Equations 4.4 – 4.12, where we present the Einstein notation convention for the derivations of curvature computed in the presence of curvilinear coordinate transformations.

2.2 Implicit Surfaces

An implicit surface is mathematically characterized as the zero level set of scalar field defined by a function $F : \mathcal{R}^3 \rightarrow \mathcal{R}$. That is, F implicitly defines a locus where $\{\mathbf{x} \in \mathcal{R}^3 : F(\mathbf{x}) = 0\}$ [16]. All other points lie either inside or outside of the surface ($F(\mathbf{x}) < 0$ and $F(\mathbf{x}) > 0$, respectively, by standard convention), with the value of $F(\mathbf{x})$ frequently indicating the relative distance of the point to the surface [70].

The specification of F is typically given as either a set of discrete samples, often spaced regularly over a lattice, or as a set of mathematical functions which evaluate F at a point $\mathbf{x}$. Discrete samples are usually physical measurements such as density, temperature, or pressure, which, when convolved with functional kernels, define *isosurfaces* within F as $\{\mathbf{x} \in \mathcal{R}^3$:

$F(\mathbf{x}) = c\}$ [99] — these kernels will be described in detail in Section 2.4. Here, c is referred to as the *isovalue* of the surface, and can be varied in visualization applications for exploring scientific and medical data. These mathematical descriptions of F specify the implicit surface as the roots of a function. Recent work in the computer graphics community has also proposed methods for constructing implicit functions from scattered point data using functions such as radial basis functions [27, 108] and moving least squares descriptions [3].

Implicit surfaces defined by continuous, differentiable functions are widely used in computer graphics, visualization, and scientific computing, in part due to their mathematical properties. The gradients of a surface, that is the first partial derivative of F, $\nabla F(\mathbf{x}) = [\partial F/\partial x, \partial F/\partial y, \partial F/\partial z]^{\mathsf{T}}$, are used for advanced shading effects to enhance the perception of small surface features [116, 56, 60]. The definition of surface normals, *i.e.*, the normalized gradients, is also important for surface deformations in simulations and medical imaging [133]. Furthermore, the implicit surface representation easily accommodates morphing and surface editing [22], CSG operations [102], shape interpolation [150], and collision detection [41].

The curvature of an implicit surface can be computed as the eigenvalues of the the second derivative matrix, or *Hessian*, evaluated at a point $\mathbf{x}$ and projected onto the local tangent plane at that point. With the Hessian given as:

$$\mathbf{H} = \begin{bmatrix} \partial^2 F/\partial x^2 & \partial^2 F/\partial x \partial y & \partial^2 F/\partial x \partial z \\ \partial^2 F/\partial x \partial y & \partial^2 F/\partial y^2 & \partial^2 F/\partial y \partial z \\ \partial^2 F/\partial x \partial z & \partial^2 F/\partial y \partial z & \partial^2 F/\partial z^2 \end{bmatrix} \tag{2.1}$$

the projection of $\mathbf{H}$ onto the local tangent plane of the surface is:

$$\mathbf{G} = \frac{-\mathbf{PHP}}{|\nabla F|} \tag{2.2}$$

where $\mathbf{P} = \mathbf{I} - \mathbf{n}\mathbf{n}^{\mathsf{T}}$ is the projection operator, with $\mathbf{n}$ as the normalized gradient. The *shape matrix* $\mathbf{G}$ will have two (possibly) nonzero eigenvalues, giving the minimum and maximum curvature values, with the corresponding eigenvectors defining their respective directions. A thorough derivation of this curvature calculation is presented by Kindlmann *et al.* [78]. Curvature is used in visualization to enhance features and provide depth and orientation clues [121, 36], as well as for parameterization of implicit surfaces [129].

Blobbies [15] and metaballs [107] were the first implicit surfaces used for modeling in computer graphics, while more recent work has focused on functions that provide greater controllability, a richer set of surface characteristics, and the ability to reconstruct continuous surfaces from

scattered point data. For reconstructing continuous surfaces from discrete data sampled over a regular lattice, certain types of compact spline kernels have been shown to efficiently and accurately approximate the original data [99, 78]. Methods for reconstructing a surface by interpolating a set of surface points are used for interactive sculpting and converting polygonal models to smooth representations — example implicit functions include radial basis functions (RBF) [27], and their more scalable successors, compactly supported radial basis functions (CSRBF) [100], along with variational implicits [149]. Approximating implicits, such as multipartition of unity (MPU) functions [108], on the other hand, are ideal for reconstructing surfaces from noisy or missing surface point data.

There are several major drawbacks of implicit surfaces that limit their use in high end modeling applications. Their inability to capture sharp features, as well as the high computational overhead associated with defining many types of implicit functions such as RBFs, CSRBFs, and MPUs, limit the number of shapes and variety of surfaces that can be modeled. Furthermore, implicit functions are inherently unintuitive to deform, and require a significant amount of machinery to indirectly produce specific deformations [162]. In scientific visualization and computing, however, the biggest drawbacks for using implicit surfaces are the computational challenges of directly rendering implicits and the lack of an explicit parameterization of the surface for defining computational boundaries — methods in the literature that address these two challenges will be discussed in Chapters 4 and 5, respectively.

2.3 Distance Transforms

A commonly used implicit function is the *distance transform*, which will be used throughout this dissertation. This function returns, for anywhere in the domain of the function, the distance to the closest point on a surface [70]. The distance transform of a surface Σ at a point $\mathbf{x}$ is defined as

$$d_{\Sigma}(\mathbf{x}) = \inf_{\mathbf{s} \in \Sigma} \parallel \mathbf{x} - \mathbf{s} \parallel . \tag{2.3}$$

Oftentimes we are interested in the *signed* distance transform of a solid S, which is the distance to the closest point on the boundary of S, δS, with the sign of the value indicating inside (negative by convention) or outside of S:

$$d_{S}(\mathbf{x}) = \mathrm{sgn}(\mathbf{x}) \inf_{\mathbf{s} \in \delta \mathrm{S}} \parallel \mathbf{x} - \mathbf{s} \parallel \tag{2.4}$$

where

$$\mathrm{sgn}(\mathbf{x}) = \begin{cases} -1 & \text{if } \mathbf{x} \in \mathrm{S} \\ 1 & \text{otherwise.} \end{cases} \tag{2.5}$$

Unless stated otherwise, we will assume a signed function when discussing distance transforms in this dissertation, and will thus refer to Equation 2.4 as simply $d(\mathbf{x})$. Figure 2.1 shows a 2D distance transformation of curve.

The distance transform is an implicit representation of the surface $F(\mathbf{x}) = 0$, and has numerous interesting mathematical properties. For example, $|\nabla d(\mathbf{x})| = 1$ almost everywhere with $\nabla d(\mathbf{x})$ orthogonal to the levelset passing through $\mathbf{x}$, the exception being at locations where the gradient is undefined due to more than one closest surface point (*i.e.*, points along the *medial axis*, which will be discussed in more detail in Section 5.3.1). The distance transform is continuous everywhere, and is differentiable almost everywhere (*i.e.*, *except* the points along the medial axis). Furthermore, where the distance transform is differentiable, the gradient field is *1-Lipschitz continuous*, which is defined for a function F as $|F(\mathbf{x}) - F(\mathbf{y})| < |\mathbf{x} - \mathbf{y}|$— this property is a smoothness condition that states that the magnitude of the gradients of the distance transform will never be larger than its Lipschitz value of one.

To compute a distance transform of a solid, such as a binary classification, a grayscale classification can be initialized in a narrow band around the boundary of the solid such that

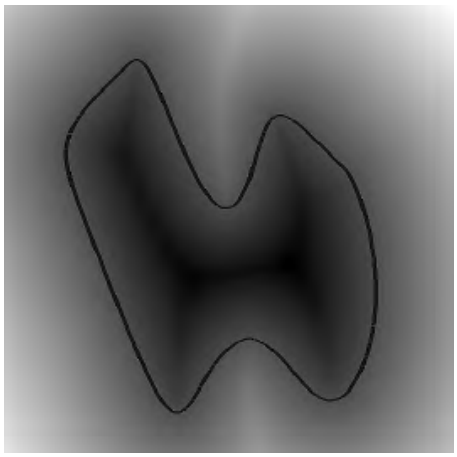


Figure 2.1. A distance transformation of the black curve, where the shade of gray represents the signed distance from the curve.

$$F(\mathbf{x}) = \begin{cases} d(\mathbf{x}) & \text{in the narrowband} \\ \infty & \text{elsewhere.} \end{cases} \tag{2.6}$$

Methods such as anti-aliasing have been shown to be effective at reducing voxelization artifacts [156]. For cases where the boundary is defined as an isosurface of a function F, a first-order approximation of d can be computed as $(F - c)/|\nabla F|$ in the narrow band region, where c is the isovalue of the surface.

Once the narrow band region has been initialized, several methods exist for computing the distance transformation over the domain of the function, such as the chamfer distance transform [124, 24] and the vector distance transform [38, 101]. A popular method for computing the distance transformation is the fast marching method (FMM) [132, 133], which computes the arrival time of a front that expands in the normal direction over a set of grid points. By solving the Eikonal equation from the boundary conditions given in the narrow band region around the surface, the FMM establishes an inverse relationship between the speed of the front and the magnitude of the gradient field over a domain. Because $|\nabla d| = 1$ at the surface, the front will move with unit speed and result in a distance transformation.

2.4 Reconstruction Kernels

The proposed system specifically targets volumetric data, such as segmentations from MRI or CT scans, which come as a set of discrete values defined over a regular lattice. We construct a continuous, differentiable implicit function from these discrete points by convolving the data with reconstruction kernels. In 1D, a set of discrete sample points v_i, located at positions $\mathbf{x}_i$, are represented continuously as $v(\mathbf{x}) = \sum_i v_i \delta(\mathbf{x} - \mathbf{x}_i)$, where $\delta(\mathbf{x})$ is the Dirac delta function. The convolution of $v(\mathbf{x})$ with a continuous kernel $f(\mathbf{x})$ is defined as [54]

$$(v * f)(\mathbf{x}) = \sum_i v_i \delta(\mathbf{x} - \mathbf{x}_i) f(\mathbf{x}) = \sum_i v_i f(\mathbf{x} - \mathbf{x}_i). \tag{2.7}$$

In visualization, reconstruction of a continuous implicit function in 2D and 3D frequently uses *separable convolution*, which treats each axis in an image (or volume) separately [76]. Thus, a 1D continuous kernel $f(\mathbf{x})$ generates a 3D continuous kernel $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) f(\mathbf{y}) f(\mathbf{z})$, resulting in a 3D convolution:

$$(v * f)(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{i,j,k} v_{ijk} f(\mathbf{x} - \mathbf{x}_i) f(\mathbf{y} - \mathbf{y}_j) f(\mathbf{z} - \mathbf{z}_k). \tag{2.8}$$

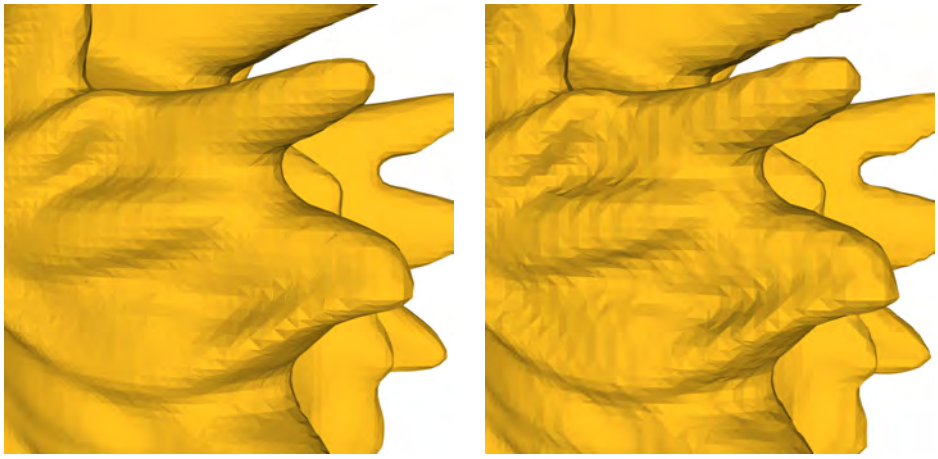
Using separable convolution, derivatives of the reconstructed function can also be evaluated and summed along each axis:

$$\begin{array}{l} \frac{\partial (v*f)(\mathbf{x},\mathbf{y},\mathbf{z})}{\partial \mathbf{x}} = \sum_{i,j,k} v_{ijk} f'(\mathbf{x} - \mathbf{x}_i) f(\mathbf{y} - \mathbf{y}_j) f(\mathbf{z} - \mathbf{z}_k) \\ \frac{\partial (v*f)(\mathbf{x},\mathbf{y},\mathbf{z})}{\partial \mathbf{y}} = \sum_{i,j,k} v_{ijk} f(\mathbf{x} - \mathbf{x}_i) f'(\mathbf{y} - \mathbf{y}_j) f(\mathbf{z} - \mathbf{z}_k) \\ \frac{\partial (v*f)(\mathbf{x},\mathbf{y},\mathbf{z})}{\partial \mathbf{z}} = \sum_{i,j,k} v_{ijk} f(\mathbf{x} - \mathbf{x}_i) f(\mathbf{y} - \mathbf{y}_j) f'(\mathbf{z} - \mathbf{z}_k). \end{array} \tag{2.9}$$

The choice of reconstruction kernel for different applications and different scanning modalities has been extensively studied in image processing [54]. In visualization, however, the most common reconstruction method is trilinear interpolation due to the ease and speed of its computation, as well as its pairing with many techniques that assume piecewise-linear data. For visualization tasks that require a higher order reconstruction, such as the design of transfer functions for volume rendering boundaries and interfaces [77], certain types of compact kernels have been shown to effectively balance computation efficiency with good approximations of derivatives [99, 78]. Using these observations, this dissertation utilizes a 4^3 cubic B-spline for approximating volume data, or a 4^3 Catmull-Rom spline for interpolation. Figure 2.2 presents two isosurface extractions of a distance transform using an approximating kernel (*left*) and interpolating kernel (*right*).

2.5 Mathematical Morphology

Volume (and image) data in the form of a *binary image*, *i.e.*, where the value at each pixel indicates the pixel is part of the material, or not, is often preprocessed to smooth features and fill gaps. Using only blurring kernels that smooth data through diffusion, such as a Gaussian kernel, can result in arbitrarily small (and sharp) features which can impose arbitrarily high sampling requirements for applications that must capture all parts of the surface. We instead propose to preprocess labeled data using mathematical morphology, first on binary images to close gaps and remove features of a certain pixel size, followed by a grayscale morphology that limits the radius of curvature of the reconstructed data. We explain each of these operators in the following two subsections. For these sections we will refer to 2D and 3D data stored over a regular lattice as an *image*.



(a) (b)

Figure 2.2. Examples of an isosurface of a distance tranform extracted using (a) an approximating 4^3 cubic B-spline reconstruction kernel, and (b) an interpolating 4^3 Catmull-Rom spline reconstruction kernel.

2.5.1 Binary Morphology

By expressing an image in a set theoretic framework, binary morphology operations can be described by unions and intersections of the shape contained in the image with morphologic stencils. As described by Gonzalez and Woods [54], binary morphology operations have two fundamental operations — *dilation*, which will add material to the shape boundary; and *erosion*, which removes material from the boundary. Dilation of a set (a shape in a binary image) A by a stencil B is

$$A \oplus B = \{\mathbf{z} | B_{\mathbf{z}} \cap A \neq \emptyset\} \tag{2.10}$$

while erosion is

$$A \ominus B = \{\mathbf{z} | B_{\mathbf{z}} \subseteq A\} \tag{2.11}$$

These two operations are shown in Figure 2.3.

Building on these two primitive operations, two other fundamental morphologic operations can be defined. *Opening* smooths the boundary by removing small features and thin regions, while *closing* smooths the boundary by closing gaps and eliminating small holes. Opening is an erosion of A by B, followed by a dilation of the result by B:

(a) (b) (c)

Figure 2.3. A binary segmentation (a) and results of the fundamental binary morphology operations *dilate* (b) and *erode* (c).

$$A \circ B = (A \ominus B) \oplus B. \tag{2.12}$$

Closing, on the other hand, is a dilation of A by B, followed by an erosion of the result by B:

$$A \bullet B = (A \oplus B) \ominus B. \tag{2.13}$$

These two operations are shown in Figure 2.4.

For 3D binary images, there are three basic, isotropic stencils. The first is a 6-pointed stencil we will call a *plus*; the second is a 22-point stencil that is a solid cube minus the corners which we will call a *ball*; and the third is a 26-point solid *cube* stencil. The operations erosion and dilation can be intuitively thought of as sliding these stencils along the inside or outside of the binary

(a) (b) (c)

Figure 2.4. A binary segmentation (a) and results of the binary morphology operations *open* (b) and *close* (c).

shape, respectively. In Figure 2.5, isosurface extractions of the three stencils are shown. These stencils can be dilated with themselves, or in combination to produce larger stencils for opening or closing features larger than one pixel.

For results in this dissertation, specifically those presented in Chapter 6, we have found that rounder stencils (like the ball stencil) produce the best binary shapes for increasing the minimum size of features in the reconstructed data. In Table 2.1 we present the stencil combinations used in this dissertation. However, other combinations can produce the similiar results. An example of the effects of morphology on the geometry and topology of an implicit surface can be seen in Figure 2.6.

2.5.2 Tightening

To produce smooth surface reconstructions from binary data, the data must be smoothed at the subvoxel level to eliminate aliasing artifacts from the voxelization. Blurring via diffusion (*i.e.*, using a Gaussian blurring kernel) can generate arbitrarily small surface features, even if small features in the binary data have been eliminated using morphology operations. On the other hand, methods that perform grayscale morphology operations based on curvature flows can smooth the data while maintaining a lower bound on the feature size. One such grayscale morphology scheme proposed by Williams and Rossignac [160], used in Chapter 6, is called *tightening*, which limits the radius of curvature of the resulting boundary using constrained, levelset curvature flow. Other methods have also been proposed that obtain similar results, such as those that use discrete Willmore flows [18] or implicit fairing [40].

Tightening eliminates high curvature on a surface while preserving low curvature parts of the boundary by defining a region around the surface called the *mortar* in which minimum length loops are computed. These loops are the boundary of a tightened surface that has the properties that the radius of curvature is greater than r everywhere and differs from the original surface only within the mortar. The mortar is defined by morphological operations called *rounding* and

Figure 2.5. Isosurface extractions of the three basic isotropic stencils — *plus* is a 6-pointed stencil, *ball* is a solid cube minus the corners, and *cube* is a solid cube.

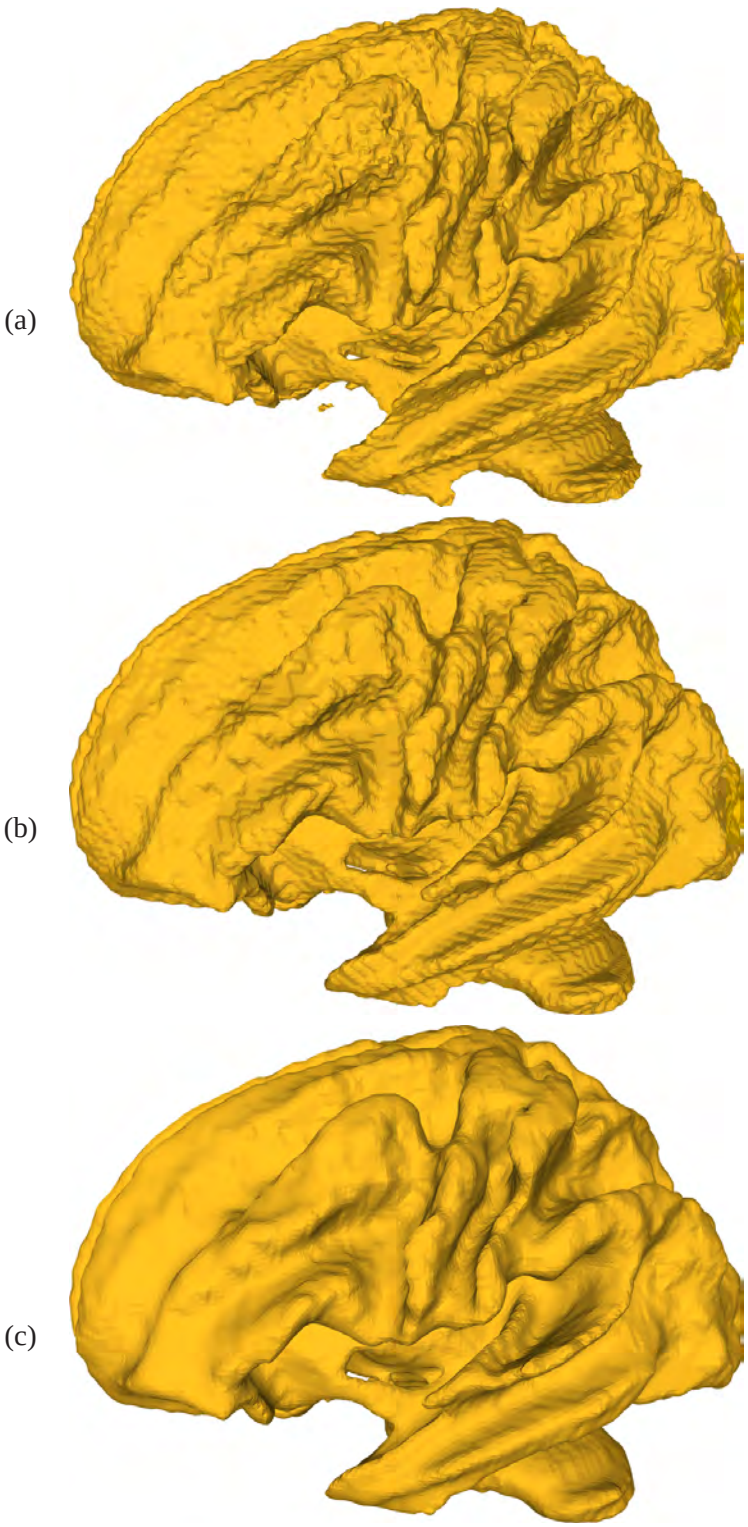
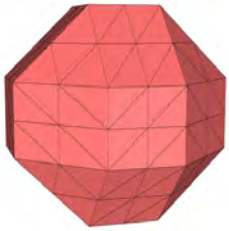
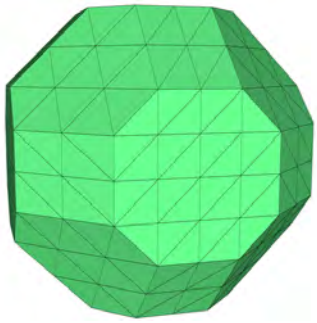


Figure 2.6. Isosurface of a brain segmentation [142] at various stages of preprocessing, extracted using an approximating kernel: (a) the original segmentation; (b) after opening and closing the volume using a ball stencil; (c) after tightening with $r = 1$.

Table 2.1. The specific, round stencils used in this dissertation for a range of feature sizes. The stencils were generated by dilating combinations of the three basic isotropic stencils, where p is the *plus* stencil, b is the *ball* stencil, and c is the *cube* stencil.

feature size	stencil morphology	stencil isosurface
1	b	
2	$b \oplus c$	
3	$(b \oplus p) \oplus c$	
4	$[(p \oplus b) \oplus c] \oplus p$	

filleting, which are analoguous to the binary morphology operations opening and closing. The mortar of the shape in Figure 2.7(a) is shown in gray in Figure 2.7(b); minimum length loops are computed in the mortar with $r = 40$ to generate the shape in Figure 2.7(c). Figure 2.6(c) shows an isosurface extraction of a brain segmentation tightened with $r = 1$.

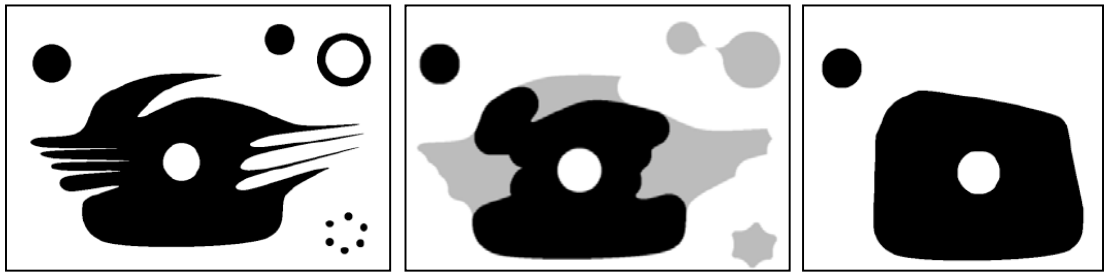


Figure 2.7. The original shape (a) has a mortar shown in gray in (b). The tightened surface, with $r = 40$ is shown in (c). (images used without permission)

CHAPTER 3

DYNAMIC PARTICLE SYSTEMS

Particle systems are a mechanism for controlling point samples and distributing them across an implicit surface, producing compact, object-space samples of the underlying geometry which can then be rendered efficiently, or, used as input to a parameterization scheme. Recent research on point-based surface representations also suggests that point sets may be a viable alternative to parametric surface representations in applications where the topological constraints of a parameterization are unwieldy or inefficient [57]. The state of the art in particle systems for sampling implicit surfaces, however, presents some shortcomings. First, most of these systems have many parameters that interact in complicated ways, making it difficult for users to tune the system to meet specific requirements. And second, these systems do not readily lend themselves to spatially adaptive sampling schemes, which are essential for efficient, accurate representations of complex surfaces.

In this chapter we first review the history of particle systems in the computer graphics literature, focusing on the work of Witkin and Heckbert [162] which established the scheme as a viable method for sampling implicit surface. We then present a new framework for distributing particles on implicit surfaces, including a new class of energy functions and a space-warping scheme for adapting the particle densities. These techniques are shown to provide stable, scalable, efficient, and controllable mechanisms for distributing particles that sample implicit surfaces within a locally adaptive framework.

3.1 Background

Particle systems for sampling implicit surfaces were introduced to the computer graphics community more than a quarter of a century ago. Modeling surfaces with particles was first proposed by Szeliski and Tonnesen [139, 140] as an oriented particle system that samples deformable surface models. They employ an energy function from the molecular dynamics literature which causes particles to exert short-range repulsion and long-range attraction, keeping particles at an appropriate distance from each other. Turk [148] uses repelling particles to resample polygo-

nalized static surfaces using curvature measurements, while De Figueiredo *et al.* [39] propose a physically-based particle method to polygonalize implicit surfaces by modeling particles with a mass-spring system. Other work presents ideas for sampling implicit surfaces for animation [41] and texture mapping [169]. In parallel, a body of work has been developed in the mathematics community that studies the discretization of surfaces via energy minimizations [127, 59].

In 1994, Witkin and Heckbert [162] introduced a novel approach to sampling and controlling implicit surfaces that builds on many of these early ideas. The system they describe constrains a set of interacting particles to lie on an implicit surface while each particle repels nearby particles to minimize a Gaussian energy function. The Gaussian energy has a characteristic length, which is adapted for each particle to suit the distribution of its neighbors. The particle interactions are constrained to the local tangent plane of the surface, while the particles are reprojected onto the implicit surface. The Witkin and Heckbert (W-H) method includes approximately 10 free parameters, and when they are carefully tuned ([162] gives an effective set of guidelines), the resulting system produces a homogeneous distribution of particles on the surface. The W-H system will be covered in more detail in the following section.

Heckbert [65] extends the original W-H method by developing a spatial binning optimization that determines the radius of influence of a particle and only calculates forces for neighboring particles within this radius. The radius of influence varies from particle to particle, so the bounding sphere must be computed for each particle. An extension to the method proposed by Hart *et al.* [62] allows the system to sample increasingly complex surfaces within an object-oriented framework that numerically differentiates implicit surfaces comprised of large numbers of control parameters. Galin *et al.* [52] propose a method that uniformly samples implicit surfaces generated from the BlobTree model [164], creating interactive visualizations of these surfaces. To decrease the number of iterations a particle system requires to converge, Levet *et al.* [86] present a geometric scheme for initializing particle positions over a surface. This method is fast to generate uniform initial positions, and Oka *et al.* [109] extend these preprocessing ideas to quickly produce initial particle positions that are adaptive to surface curvature.

3.2 The Witkin and Heckbert Method

The W-H method constrains n repulsive particles $\mathbf{P} = \{\mathbf{p}_0, \cdots, \mathbf{p}_{n-1}\}$ to lie on an implicit surface of the function $F(q)$, which is controlled by parameters $\mathbf{q}(t)$ that change over time to meet user-defined surface deformation goals. That is,

$$F(\mathbf{x}_i, \mathbf{q}(t)) = 0 \tag{3.1}$$

where $\mathbf{x}_i$ is the position of particle $\mathbf{p}_i$. In this framework, particles provide not only a way to visualize the implicit surface in real time, but also provide a handle through which surface deformations are controlled by updating the parameters $\mathbf{q}(t)$. This dissertation is concerned with only the particle placement, and thus, to simplify the discussion, we ignore the surface deformation terms from the original W-H formulation.

Once the particles lie on the surface, individual potential functions are associated with each to induce particle-particle interactions. Each particle, $\mathbf{p}_i$, creates a potential field, which is a function of the distance between $\mathbf{p}_i$ and all neighboring particles that lie within the potential field. The energy at a particle $\mathbf{p}_i$ is defined as the sum of potentials of the m particles that interact with $\mathbf{p}_i$:

$$E_i = \frac{1}{2} \sum_{j=1, j\neq i}^{m} E_{ij}(\mid \mathbf{r}_{ij} \mid), \tag{3.2}$$

where $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. In the W-H scheme, a Gaussian kernel is used for the potential function:

$$E_{ij} = \alpha \exp\left(-\frac{|\mathbf{r}_{ij}|^2}{2\sigma^2}\right) \tag{3.3}$$

where α is a global repulsion amplitude parameter, and σ, called the repulsion radius, is the standard deviation of the Gaussian.

The derivative of a particle's energy with respect to its position gives rise to the repulsive force that defines the repulsion velocity direction (steepest descent) that minimizes the local energy:

$$\mathbf{v}_i = -\frac{\partial E}{\partial \mathbf{x}_i} = -\frac{\partial E_i}{\partial \mathbf{x}_i} = -\sum_{j=1, j\neq i}^{m} \frac{\partial E_{ij}}{\partial \mid \mathbf{r}_{ij} \mid} \frac{\mathbf{r}_{ij}}{\mid \mathbf{r}_{ij} \mid}. \tag{3.4}$$

Iteratively moving particles along their energy gradients causes the system to converge toward a minimal energy configuration.

The surface constraint is enforced by projecting $\mathbf{v}_i$ onto the local tangent plane of particle $\mathbf{p}_i$, and then reprojecting the updated particle position onto the zero set of F using a Newton-Raphson gradient descent technique. The change in a particle's position, $\dot{\mathbf{x}}_i$ is characterized by

$$\dot{\mathbf{x}}_i = \left(\mathbf{I} - \frac{\nabla F_i \otimes \nabla F_i}{\nabla F_i \cdot \nabla F_i} \right) \mathbf{v}_i - \phi F_i \frac{\nabla F_i}{\nabla F_i \cdot \nabla F_i}, \tag{3.5}$$

where $\mathbf{I}$ is the identity matrix, ∇F_i is the spatial gradient of F at the particle $\mathbf{p}_i$, and $\nabla F_i \otimes \nabla F_i$ is the projection operator formed by the vector direct product of the gradient. The last term in (3.5) is the reprojection onto the implicit surface (the *feedback term* in [162]), which is scaled by a free parameter ϕ. A fraction of $\dot{\mathbf{x}}_i$ is added to the current position in the manner of a finite forward difference scheme, *i.e.*, $\mathbf{x}_i \leftarrow \mathbf{x}_i + \lambda \dot{\mathbf{x}}_i$, where λ is the gradient descent constant mentioned previously. The particle is then rendered using a disk oriented to lie in the local surface tangent plane [139].

Each particle maintains an adaptive repulsion radius, σ_i, which grows and shrinks based on the local energy values. This allows particles to quickly spread over the surface by increasing their repulsion radius in sparse regions, which in turn increases the interparticle forces that push particles to lower energy states. Accompanying the adjustable radius is a *splitting and dying* mechanism based on a target radius, $\hat{\sigma}$, that controls the insertion and deletion of particles in the system. When a particle's σ_i drops below some fraction of $\hat{\sigma}$, indicating a densely sampled region, it is removed, and when it goes above some multiple of $\hat{\sigma}$, indicating a sparsely sampled region, a new particle is inserted nearby. The system can thus quickly move particles into sparse regions by growing σ_i for particles in underpopulated regions, and then inserting new particles when σ_i becomes too high.

The *dying* mechanism is also important to keep particles from clumping together if they get closer to each other than the characteristic length of the Gaussian potential kernel. At σ_i, a particle's force function begins to dip towards zero as the derivative of the Gaussian energy reaches a maximum, creating a potential well that can trap particles together. Although the particles will not be able to force each other away, the increase in their respective energies will cause their σ_i values to decrease, either eventually moving the particles out of the potential well, or causing a particle to die as its σ_i falls below the dying criteria cut-off.

Ultimately, the system forms homogeneous distributions of particles over the surface by adapting σ_i until all particles have similar energy measures. Many applications, however, require inhomogeneous distributions based on curvature, such as using particles to polygonalize [72] or to parametrize [169] the surface. Several extensions to the original W-H particle system have been proposed in order to accommodate increased sampling in areas of high curvature [123, 35, 72]. All three extensions apply an adaptive, per-particle, curvature dependency to either the repulsion

radius, σ_i, or to the target radius, $\hat{\sigma}_i$. We have found these extensions do not provide adequate curvature dependent distributions for complex surfaces with large variations in curvature values, confirming the difficulties mentioned by Rösch *et al.* [123] and Karkanis *et al.* [72].

These difficulties arise because σ_i will grow and shrink regardless of the underlying curvature value, and $\hat{\sigma}_i$ does not control the behavior of a particle apart from its splitting and dying. For example, consider a particle in a high curvature area with a relatively low $\hat{\sigma}_i$. As σ_i increases this particle splits at faster rates than particles in nearby flatter areas. However, these new particles will merely be pushed out onto flat areas, which will, in turn, become too crowded, resulting in the deletion of particles. Meanwhile, the high-curvature particle, missing the fleeing particles it recently created, will continue to split—a never ending cycle of insertion and deletion, illustrated in Figure 3.1. In our experiments, when we tuned parameters to stop the insertion-deletion cycle by expanding the hysteresis of insertion and deletion we found that the particle distributions did not reflect the desired differences in particle densities—the W-H scheme tends toward homogeneous distributions despite variations in $\hat{\sigma}$. Local adaptation in the W-H scheme is significantly more complex than a single parameter; it also requires modifications to the particle radii interactions with the per-particle energy functions.

Further complicating fine-grained control of the system are not only the numerous free parameters in the splitting and dying mechanism, but also the tunable parameters in the numerical algorithm used for iteratively distributing the particles. The W-H method relies on a gradient descent for both particle repulsion and σ_i adaptation. Discretized gradient descent algorithms

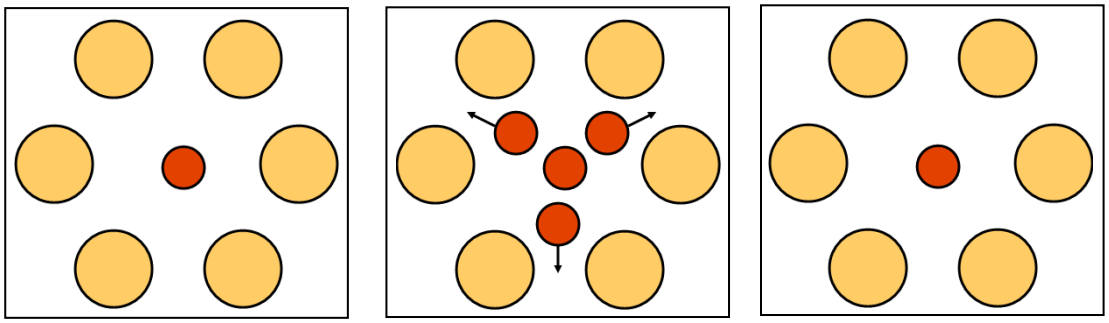


Figure 3.1. Adapting the radius of particles based on local surface curvature in the W-H system results in a cycle of insertion and deletion of particles: (a) a high curvature particle in red has a low energy value; (b) the high curvature particle splits to increase its energy, and these new particles are pushed outwards towards a lower curvature region sampled by the yellow particles; (c) the new particles are deleted as the lower curvature area becomes over crowded, leaving the high curvature particle again in a low energy state.

invariably introduce a critical free parameter (the descent rate, or unit change per iteration), and the system can easily become either too slow or unstable if the parameter is improperly tuned. As a result, changes to the W-H system often entail careful retuning of the corresponding gradient descent parameter.

In this chapter we propose a new approach to distributing particles across an implicit surface, allowing for a wide range of distribution patterns from homogeneous to highly adaptive. The proposed system is general across a broad range of shape complexity and size, and requires minimal parameter tuning from surface to surface. The new framework builds upon the constrained particle system developed by Witkin and Heckbert, but introduces a new class of energy functions accompanied by a single, global radius σ that virtually eliminates the need for insertion-deletion to ensure even distributions of particles across the entire surface. These (approximately) scale-invariant energy functions allow particles to interact in a similar fashion over a wide range of distances without adapting or tuning parameters. The particle insertion and deletion mechanism is instead applied for control of particle densities and spacing, and is combined with a space-warping scheme that causes particles to distribute with higher densities in regions of interest. To address the limitations of gradient descent we further propose an adaptive time-stepping minimization scheme, which automatically tunes the descent rate to accommodate the individual particle force magnitudes.

The result is a robust system with relatively few parameters that provides a new capability: a controllable, locally adaptive distribution of particles on implicit surfaces. Mechanisms such as neighborhood size and deletion/insertion of particles can now be adapted to meet other constraints, such as the total number of particles in the system, the average particle density, the efficiency of the computation, or update and rendering times.

3.3 A New Particle Energy Scheme

At the heart of the proposed particle system is the computation of the potential energy associated with particle-particle interactions. The minimization of this energy defines the algorithm for distributing particles across the surface and leads to a quantifiable notion of an ideal distribution. This pairwise potential energy E_{ij} is the most important aspect of any such particle system, with a bad energy function leading to numerical instabilities and uneven particle distributions, and a good function resulting in a homogeneous steady state. We have experimented with several potential energy functions from the literature and have identified three important characteristics of a well behaved potential energy function. First, energies should be C^1 continuous functions

of particle distance. Second, the energy functions should be compact to avoid global influences and allow for efficient computation. And third, to avoid characteristic lengths, and the associated parameter tuning, the energy must be scale invariant — that is, two particles at different distances should have the same ratio of energies regardless of the choice of units of the system.

The Gaussian energy used in the W-H method is smooth and nearly compact because the function can be truncated in a manner that does not significantly affect its behavior. The Gaussian, however, has a characteristic length and is not scale invariant. A particularly interesting example of a scale-invariant energy is the electrostatic potential, $E_{ij} = 1/|\mathbf{r}_{ij}|$. The electrostatic function is smooth, except at the origin (which can be fixed by adding adding a small constant to the denominator), but does not fall off quickly enough to provide local behavior. As a result, particles do not remain on flat regions but instead concentrate exclusively on convex, high-curvature areas—a well-known phenomenon from electrostatics. Furthermore, truncating the electrostatic potential yields unreliable results, and the configurations of particle steady states are very sensitive to the distance of truncation. Thus, the electrostatic function is not approximately compact. Figures 3.2 (a-b) show graphs of the Gaussian and electrostatic energy functions, respectively.

An energy function which establishes a good compromise between approximate scale invariance and compactness is a *modified cotangent*:

$$E_{ij} = \begin{cases} \cot\left(\frac{|\mathbf{r}_{ij}|}{\sigma}\frac{\pi}{2}\right) + \frac{|\mathbf{r}_{ij}|}{\sigma}\frac{\pi}{2} - \frac{\pi}{2} & |\mathbf{r}_{ij}| \leq \sigma \\ 0 & |\mathbf{r}_{ij}| > \sigma \end{cases} \tag{3.6}$$

which is shown graphically in Figure 3.2 (c). This potential has one free parameter σ, which establishes the farthest distance at which particles interact. When $\mid \mathbf{r}_{ij} \mid = 0$ Equation 3.6 goes to infinity — we avoid this numerical instability by adding a small value, $\epsilon_{\mathbf{r}} = 10^{-7}$, to the interparticle distances. The derivative of this energy with respect to particle distance is:

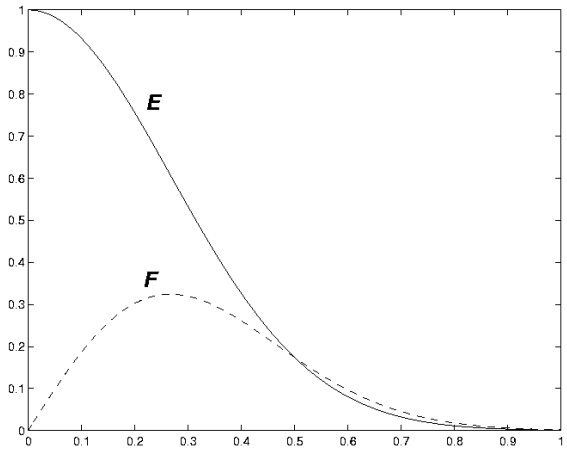
$$\frac{\partial E_{ij}}{\partial \mid \mathbf{r}_{ij} \mid} = \begin{cases} -\frac{\pi}{2\sigma}\left[1 - \sin^{-2}\left(\frac{|\mathbf{r}_{ij}|}{\sigma}\frac{\pi}{2}\right)\right] & |\mathbf{r}_{ij}| \leq \sigma \\ 0 & |\mathbf{r}_{ij}| > \sigma \end{cases} \tag{3.7}$$

The derivative shows an analogous relationship to the electrostatic potential. When the distance between particles is small relative to σ, $\sin(|\mathbf{r}_{ij}|/\sigma) \approx |\mathbf{r}_{ij}|/\sigma$, and the force behaves like $-1/r^2$, which is invariant to scale.

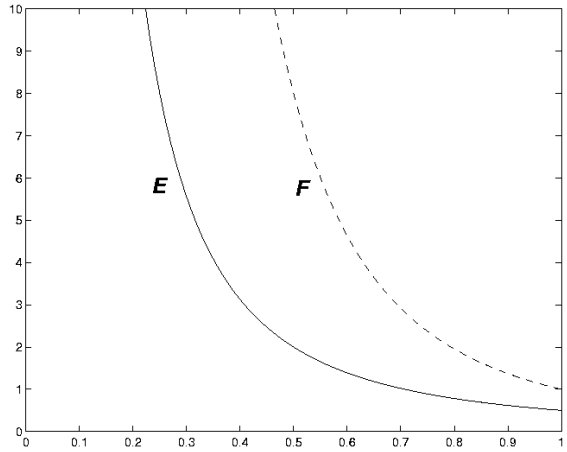
The particle system uses Euclidean distances when computing the distance between particles as opposed to the more accurate, and computationally expensive, geodesic distance. This approximation is reasonable as long as the distance between particles is small relative to the local

Figure 3.2. Plots of energy functions (E) and the corresponding force functions (F) : (a) Gaussian energy, exhibiting a characteristic length; (b) electrostatic energy, exhibiting a necessary truncation; (c) the proposed *modified cotangent* energy, exhibiting compactness and approximate scale invariance.

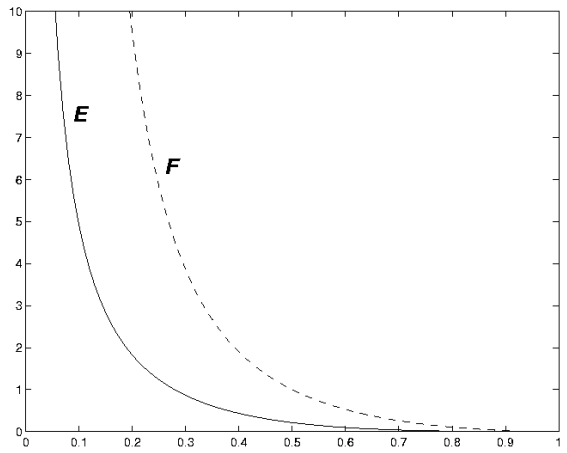
(a)



(b)



(c)



curvature of the surface such that the surface is approximately planar, which is a valid assumption when adapting the particle densities with curvature (discussed in Section 3.5.2). For σ values that are large relative to the dimensions of the domain, it is possible then to include particles in the energy and force computations that lie on non-adjacent parts of the surface. To cull some of the non-neighboring particles, we scale Equations 3.6 and 3.7 by a weighting function based on the dot product of the normals at $\mathbf{p}_i$ and $\mathbf{p}_j$:

$$w_i j = \begin{cases} 1 & \mathbf{n}_i \cdot \mathbf{n}_j \geq \gamma \\ \cos\left(\frac{\gamma - \mathbf{n}_i \cdot \mathbf{n}_j}{\gamma}\right) & \gamma > \mathbf{n}_i \cdot \mathbf{n}_j > 0 \\ 0 & \mathbf{n}_i \cdot \mathbf{n}_j \leq 0 \end{cases} \tag{3.8}$$

with $\gamma = 0.156$ for the results in this dissertation.

Another energy that is very similar to the modified cotangent function is a radial energy:

$$E_{ij} = \begin{cases} \frac{1 - \left(\frac{|\mathbf{r}_{ij}|}{\sigma}\right)^2}{\left(\frac{|\mathbf{r}_{ij}|}{\sigma}\right)^2} & |\mathbf{r}_{ij}| \leq \sigma \\ 0 & |\mathbf{r}_{ij}| > \sigma \end{cases} \tag{3.9}$$

with a derivative given as

$$\frac{\partial E_{ij}}{\partial \mid \mathbf{r}_{ij} \mid} = \begin{cases} -\frac{2}{\sigma}\left[\frac{1 - \frac{|\mathbf{r}_{ij}|}{\sigma}}{\left(\frac{|\mathbf{r}_{ij}|}{\sigma}\right)^3}\right] & |\mathbf{r}_{ij}| \leq \sigma \\ 0 & |\mathbf{r}_{ij}| > \sigma \end{cases} \tag{3.10}$$

Experiments using this function result in visually similar distributions as those generated using Equation 3.6.

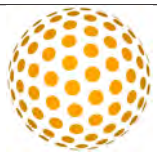
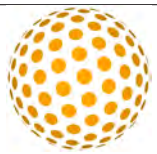
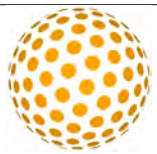
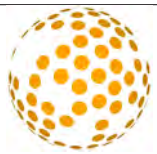
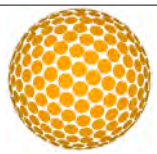
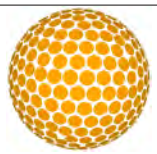
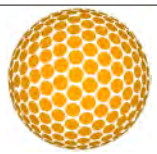
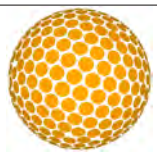
Our experiments show that the modified cotangent and radial energies homogeneously distributes particles across the surface, freeing up the need to modify σ on a per particle basis or to implement a parameter sensitive particle insertion and deletion algorithm. The particles distribute themselves in a nearly hexagonal packing, which is the general pattern for optimal configurations [127]. The system is well behaved due to the lack of a characteristic length in the energy or force function, and works across a broad range of surface shapes and sizes with no modifications. Because of this robust behavior, σ can be treated as an application dependent parameter which can be tuned in accordance with the desired density of particles and the run-time requirements of the application — these ideas will be discussed further in Section 3.5.

Table 3.1 illustrates the robustness and generality of the cotangent energy function (with the radial energy exhibiting similar results). For these examples we start with a random placement of particles, then iteratively move these particles using a gradient descent until the system converges to a homogeneous distribution. We vary the value of σ under two different scenarios: a sparsely packed system of 300 particles, and a densely packed system of 600 particles. When $\sigma = 1$ (where σ is the fraction of the domain size), the energy has a global influence over the entire surface function domain. The particle distributions continue to be homogeneous as we reduce σ, demonstrating that the cotangent energy is approximately invariant over a wide range of scales. The only restriction is that σ must be large enough such that particles interact with a ring of neighbors at the steady state ($m \approx 6$ for this example). The upper right example in Table 3.1 shows the system breaking down when σ is too small to provide sufficient interaction. As discussed in later sections, this condition can be met automatically by either increasing σ or adding more particles.

3.4 Moving Particles

Integrating the particles towards a progressively lower energy state is a nonlinear optimization problem that introduces numerical challenges. A gradient descent requires careful tuning of the step size parameter, which can vary from surface to surface, and even particle to particle. Improper values can lead to very long convergence times if the step size is too small, or irreconcilable oscillations around the minimum if the step size is too large. To avoid these problems we have implemented an adaptive gradient descent integration algorithm, in the spirit of the

Table 3.1. The effects of varying the one free parameter, σ, in the cotangent energy function. The total number of particles, n, is constant along the rows, and the value of σ is the fraction of the total domain. The average number of influenced neighbors, m, is given for each variation. The upper right example illustrates the one constraint on σ, which is that σ must be large enough to ensure adequate distributing forces at a particle.

n	$\sigma = 1$	m	$\sigma = \frac{1}{6}$	m	$\sigma = \frac{1}{9}$	m	$\sigma = \frac{1}{12}$	m
300		150.3		8.5		5.8		1.1
600		302.1		18.9		6.6		5.8

Levenberg-Marquardt integration scheme [119], that does not require any tuning of parameters, described in detail in Section 3.4.1. We also propose a Gauss-Siedel update strategy, in which particle positions are updated one at a time, and each particle update relies on the most recent positions of its neighbors. Moving particles in this method causes the distribution to converge about twice as fast as iterating using a Jacobi update where all particle movements are computed before making any positional updates [73].

A two-step update scheme keeps the moving particles constrained to the surface. First, particle positions are updated based on the repulsion velocity in the tangent plane:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \left(I - \frac{\nabla F_i \otimes \nabla F_i}{\nabla F_i \cdot \nabla F_i} \right) \mathbf{v}_i, \tag{3.11}$$

which is the result of a Lagrangian formulation of the constrained optimization that keeps particles on the zero sets of F. Movements in the tangent plane can, however, push particles off of the surface, especially in areas of high curvature. Therefore a tangent plane movement must be followed with a reprojection:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - F_i \frac{\nabla F_i}{\nabla F_i \cdot \nabla F_i}, \tag{3.12}$$

which is a Newton-Rhapson approximation to the nearby roots (zero sets) of F. We have found that a single iteration of Equation 3.12 is generally sufficient to keep particles on the surface for most well-behaved implicit functions.

3.4.1 Adaptive Gradient Descent

To overcome the numerical problems of using a fixed stepsize parameter when integrating the particle motions, we propose an adaptive gradient descent method that automatically tunes individual step sizes for each particle. Each particle maintains an individual step size parameter λ_i that is adjusted by the system based on the local energy of the particle. A particle is temporarily moved to a new position, with the motion scaled by λ_i, and the new energy E_i^{new} is compared to the old. If the energy is not lower, the λ_i value is decreased and the particle attempts another, smaller movement. Otherwise, the particle accepts the movement, and its λ_i is increased for the next iteration. When λ_i is low the particle moves in small steps in order to not over step the minimum, but when λ_i is high, the particle moves much faster, quickly minimizing poor configurations. This particle-by-particle adjustment allows the particles to take steps that con-

tinually decrease the local energy, avoid destabilizing motions, and to make smaller and smaller movements as the distribution converges to a steady state.

Particles must be on the surface before computing E_i^{new} because a particle should not be allowed to move to a lower energy position if that position is not on the surface. Also, the system is slow to converge if the particles are allowed to jump over one another, and thus we penalize a particle with a very high energy when it attempts to move too large a distance (Section 3.6 defines *too large*). This movement size penalty forces λ to decrease until the motion is on the same scale as the neighborhood. Each particle is updated and moved individually so that changes are propagated into subsequent particle updates. The entire process over all of the particles is repeated until convergence.

Each particle's λ_i value is initialized to 1.0 (although we have found the system to be insensitive to this value) and modified by factors of 10 as the system converges to an even distribution. Modifying λ_i is an iterative process governed by energy computations and comparisons, and works as follows for one particle's position update:

Step 1: Compute E_i, and compute $\mathbf{v}_i$ with Equation 3.4.

Step 2: Compute $\mathbf{v}_i^{\text{new}} = \lambda_i \mathbf{v}_i$.

Step 3: Compute the new particle position $\mathbf{x}_i^{\text{new}}$ by solving Equation 3.11 with $\mathbf{v}_i^{\text{new}}$, followed by a reprojection to the surface by solving Equation 3.12.

Step 4: Compute the new energy value, E_i^{new}, at the new particle location $\mathbf{x}_i^{\text{new}}$.

Step 5: If $E_i^{\text{new}} \geq E_i$, and $\lambda_i > \lambda_{\min}$, decrease λ_i by a factor of 10 and go back to Step 2. If $\lambda_i \leq \lambda_{\min}$, do not move the particle and skip to the next particle in the list.

Step 6: If $E_i^{\text{new}} < E_i$, update $\mathbf{x}_i = \mathbf{x}_i^{\text{new}}$. Increase λ_i by a factor of 10 if this is the first time through the loop.

We have found this integration scheme to be insensitive to the two free parameters, the initial value of λ, as well as λmin, as long as $\lambda_{\min}$ is sufficiently small (for all the results in this dissertation, $\lambda_{\min} = 10^{-14}$). The method converges over a wide range of surface shapes and sizes with no modifications.

3.5 System Control

The scheme described in the previous section ensures that particles repel each other and reach a uniform distribution in a reasonable amount of time, without modifying free parameters, or inserting and deleting particles. The only restriction of the scheme is that σ must be large enough such that all the particles can maintain at least a one-ring of neighbors. In practice, when dealing with unfamiliar or deforming surfaces it may be necessary to enforce certain relationships between the particles and the surface. For instance, we might want to maintain a minimum number of particles, a certain minimum particle density, or a minimum particle radius that covers the surface with a specified number of particles. Meeting these conditions will require modifying the number of particles or the radius of the energy function. Furthermore, we would also like to have particle distributions that adapt to local surface features. Mechanisms for meeting these conditions are discussed in the following two subsections.

3.5.1 Global Control of Particles

As with the W-H method, the particle energy quantifies the density of particles in the neighborhood of a single particle, while the system energy provides information on the global density of particles. The system energy measure provides further insight to the efficiency of the system and the locality of the particle influences. Based on energy measures, several techniques can be used to ensure an efficient and effective system.

The energy of a particle quantifies the amount of interaction the particle has with its neighbors, where a low energy implies too few neighbors, and high energy indicates to many. To determine whether the particle system contains enough energy to enforce even particle distributions without incurring unnecessary particle-particle computations, the system energy measure is compared against an ideal energy measure, nE_{ideal}, where n is the number of particles in the system and E_{ideal} is the ideal energy for a single particle. In our system, we define E_{ideal} based on a hexagonal packing of particles where the repulsion radius ends at the two-ring of neighbors, similar to the distribution described in the W-H method:

$$E_{\text{ideal}} = 6E_{ij} \quad \text{with} \quad \frac{|\mathbf{r}_{ij}|}{\sigma} = \beta = \frac{0.5}{\cos(\pi/6)} \approx 0.58 \tag{3.13}$$

The hexagonal configuration represents a natural, low local energy distribution [127], and is illustrated in Figure 3.3. When the system energy is below nE_{ideal}, particles will generally not

have enough neighbors to reach an acceptable distribution. On the other hand, when the system energy is greater than nE_{ideal}, particles are influencing more neighbors than necessary, resulting in extraneous particle-particle computations and slower global convergence.

To achieve the nE_{ideal}, several mechanisms exist to modify the system energy. Particles can be inserted or deleted to increase or decrease the system energy respectively, or σ can grow or shrink. These mechanisms can be used separately or in combination, depending on the goals of the application.

Inserting and deleting particles drives the system to maintain a specific surface density of particles, defined by the value of σ. For a specific σ value, the system energy specifies the approximate local density of neighboring particles across the entire surface. If the local densities contain more particles than the defined ideal packing, particles can be deleted, either randomly across the surface, or in a biased approach similar to the W-H deletion criteria. Conversely, low local energies can be adjusted by splitting particles in the local tangent plane.

Adjusting σ can be used when a specific number of particles is desired. The σ value grows and shrinks to ensure that particles interact with only the ideal neighborhood distribution. Growing σ is important when the system energy is too low to ensure an even distribution of particles, while shrinking σ when the system energy is too high produces the most computationally efficient system by keeping inter-particle calculations as local as possible. Changes in σ are carried out iteratively using a gradient descent or some other 1D optimization technique.

A combination of insertion and deletion of particles can be used with growing and shrinking σ to maintain a lower bound on the number of particles, as well as an upper bound on σ. This combination of constraints ensures a minimum number of particles in the system at all times, and

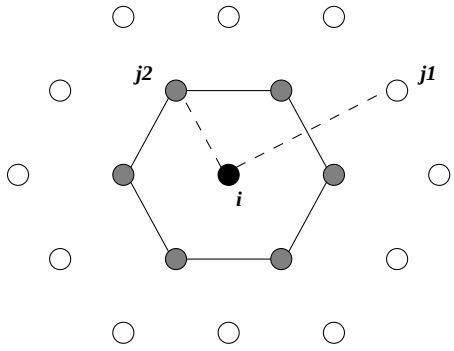


Figure 3.3. When determining the ideal energy at a particle, we want only the 6-ring neighbors to influence a particle's energy and force calculations. In this diagram, we want the distance from i to $j1$ to be σ, which makes the distance from i to $j2$ approximately 0.57σ. The white particles fall at, or just outside of, the repulsion radius of $\mathbf{p}_i$, while the gray particles exert forces.

a cap on the complexity of the interparticle calculations. Only changing σ is useful in applications that require a strict number of particles, such as using particles to set up correspondence points between multiple shapes for use in shape statistics [28]. For the applications in this dissertation, however, we rely solely on inserting and deleting particles to achieve specific densities of points. The ability to achieve specific interparticle distances is an integral component for using particle systems to generate isosurface meshes (Chapter 5 and 6).

3.5.2 Locally Adaptive Particle Distribution

The local density of particles can be controlled to achieve an adaptive sampling by introducing a repulsion amplitude, α_{ij}, between each pair of particles. This parameter is used to scale the *effective distance* between particles based on the local geometry of the surface, resulting in scaled energy and force values. For instance, if we allow surface curvature to increase the effective distance between particles, the energy and force at these particles will decrease, and the areas of high curvature will have a higher density of particles. This adaptivity mechanism maintains a small number of influential neighbors around a particle even in regions of high curvature, allowing for effective optimization strategies (discussed in Section 3.7). For results in this dissertation, $\sigma = 1$, and the vector between two particles is:

$$\mathbf{r}_{ij} = \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) = -\mathbf{r}_{ji}. \tag{3.14}$$

With this formulation, when $\mid \mathbf{r}_{ij} \mid > 1$, we have $E_{ij} = 0$, and $\mid \mathbf{v}_{ij} \mid = 0$. Equation 3.14 also ensures that the energy and force between two particles is symmetric, which is important for stability in the system. The repulsion amplitude α_{ij} could be defined as a function based on any geometric property of the surface, and we have developed one formulation based on the curvature magnitude, C (root sum of squares of the principle curvatures):

$$\alpha_{ij} = \alpha_{ji} = \frac{1 + \rho\left(\frac{s}{2\pi}\right) C_{ij}}{s\beta} \tag{3.15}$$

where s and ρ are user defined variables that specify the distance, in world units, between particles on a planar surface and the density of particles per unit angle over a curved surface, respectively, and C_{ij} is the average of the curvature magnitudes at $\mathbf{x}_i$ and $\mathbf{x}_j$. The principle curvatures are

computed analytically from the gradient and Hessian of the implicit function as described in Chapter 2, Equation 2.2.

For surfaces that contain singularities, the curvature magnitudes can become arbitrarily large at critical points and cause extremely high densities of particles. To curb this effect a maximal bound can be placed on α_{ij}. This constraint still allows particles to get very close to singularities, but does not guarantee that the critical point will be exactly sampled. We have observed, however, that particles tend to sample the sharpest points and edges of a surface — an artifact of projecting a particle's motion vector into the local tangent plane of the surface.

To illustrate the adaptivity of the particle system, Figure 3.4 presents three examples of particle distributions over a quartic implicit function with varying values of the angular density parameter, ρ.

3.6 Implementation

Here we present a basic implementation overview of the proposed particle system using an insertion and deletion mechanism, which will be referred to in later chapters. Tables 3.6 and 3.6 list all the free parameters in the system. We eliminate divisions by zero in the implementation of the the proposed curvature based sampling method by adding a very small value to all denominators.

For the results presented in this chapter we initialize our system with several hundred particles in random positions within the 3D domain of the implicit surface, while the s and ρ parameters are user-defined. In later chapters (Chapters 5 and 6) we propose seeding the particle system with the vertices of a mesh generated using the *marching cubes* algorithm [90] to ensure that the surface is adaquetly sampled. We have found that for complex surfaces, it is possible for the convergence criteria (described below) to fail and stop the particle distribution early if the

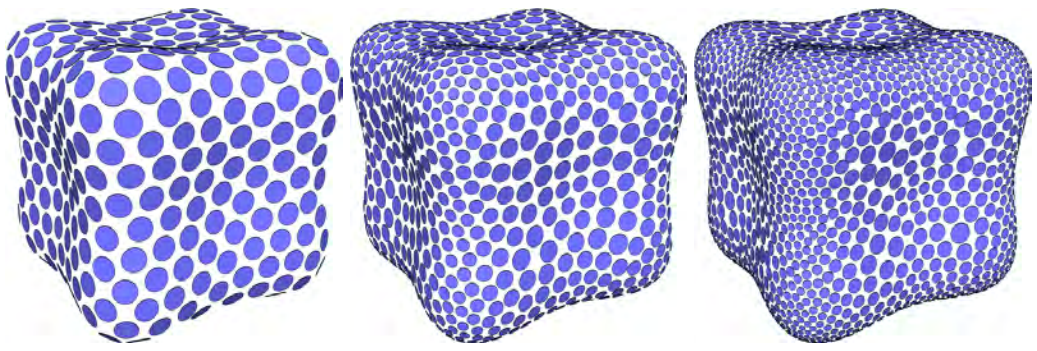


Figure 3.4. The adaptivity of the particle system is modified from left to right with $\rho = 0$, $\rho = 7$, and $\rho = 15$.

initial set of particles is not dense enough. For the results in this dissertation, however, we did not experience this problem.

First, the particles are projected onto the isosurface by performing several iterations of Equation 3.12. We have found that five iterations is usually sufficient for moving particles to within $\epsilon_F = 10^{-5}$ of the isovalue. Next, the system iterates using a Gauss-Seidel updating scheme until the particle distribution converges by repeating the following steps:

1. For each particle (taken from Section 3.4.1):

   (a) Compute E_i, and compute $\mathbf{v}_i$ with Equation 3.4.

   (b) Compute $\mathbf{v}_i^{\text{new}} = \lambda_i \mathbf{v}_i$.

   (c) Compute the new particle position $\mathbf{x}_i^{\text{new}}$ by solving Equation 3.11 with $\mathbf{v}_i^{\text{new}}$, followed by a reprojection to the surface by solving Equation 3.12.

   (d) Compute the new energy value, E_i^{new}, at the new particle location $\mathbf{x}_i^{\text{new}}$, as well as the new implicit function value, F_i^{new}.

   (e) If $E_i^{\text{new}} \geq E_i$ or $|\mathbf{x}_i - \mathbf{x}_i^{\text{new}}| > s$ or $F_i^{\text{new}} > \epsilon_T$, and $\lambda_i > \lambda_{\min}$, decrease λ_i by a factor of 10 and go back to Step 2(b)ii. If $\lambda_i \leq \lambda_{\min}$, do not move the particle and skip to the next particle in the list.

   (f) If $E_i^{\text{new}} < E_i$, update $\mathbf{x}_i = \mathbf{x}_i^{\text{new}}$. Increase λ_i by a factor of 10 if this is the first time through Step 2b.

2. Decide whether the system is at a steady state. There are numerous metrics to determine steady state, and we have chosen to use the difference of the system energy (the sum of the energy at all the particles) from one iteration to the next. When the system energy difference is less than a small fraction of the total energy (we use 0.15% for the results presented in this dissertation, although a range of values would produce similar results), we deduce that particles have reached a steady state. Otherwise, repeat Step 2b. For poor initializations, we have found it can be useful to skip to Step 2d every 50 iterations instead of waiting for the system to reach a steady state.

3. Check whether the configuration of particles is desirable. We compare each particle's energy against an ideal energy, E^{ideal}, which is defined by a hexagonal packing of neighbors on a flat surface with inter-particle distances of s. We bias E_i with a random value on the interval $[0, 1]$ to eliminate mass splitting or dying, then split particles with $E_i < 0.35E^{\text{ideal}}$,

and delete particles with $E_i > 1.75E^{\text{ideal}}$. Alternatively, if a constant number of particles is desired, the planar separation variable s could be modified to move the system energy towards the ideal system energy. While we have provided specific values for this step, we have found in practice that varying the values by up to 20% produces visually similar results, although with different convergence times.

4. If the energy of the particles is acceptable, stop iterating.

3.7 Computational Optimizations

We have implemented several strategies to increase the computational efficiency of the distribution process. First and foremost, any type of Lagrangian scheme suffers from an inherent lack of explicit spatial relationships. In the case of the particle system described in this dissertation, the problem manifests in the computation of repulsive forces from local particle interactions. Nominally the particle-to-particle interaction problem is $O(n^2)$ for each iteration. The use of compact energy kernels, however, leaves the energy and force computations null for all but a small subset of neighboring particles. Thus, we have implemented a spatial binning structure [35] that lessens the subset of potential interactions. The size of the bins is based upon the maximum possible extent of the energy kernel (which derived from Equation 3.15 is $s\beta$), with each bin maintaining a list of resident particles. Neighboring particle queries for computing forces and energies is then reduced to computing interactions with particles that reside in the $3 \times 3 \times 3$ block of bins surrounding the querying particle's bin. As such, every time a particle is moved in Step 2b above, the binning structure particle lists are updated.

Although the binning structure dramatically decreases the system convergence time, particles will still compute distances with many non-neighbor particles, especially as the adaptivity of the system is increased (*i.e.*, for large values for ρ). By not allowing particles to move further than s in any one iteration (Step 2(b)v), however, the neighborhood configurations change very little from iteration to iteration. Taking advantage of this observation, we store a list of neighbors with each particle that is updated only after several iterations (every five iterations for the results in this chapter). Although the lists may be out of date for intermediate iterations, the iterative convergence method smooths out these errors. We have found that this approach is particularly effective as the system approaches a steady state and the particle neighborhoods become stable. Storing the lists of neighbors typically decreases the computation time by a factor of 2-3, and even more so with increasing adaptivity.

Table 3.2. Table of free parameters.

Parameter	Value	Description	Comments
$\epsilon_{\mathbf{r}}$	10^{-7}	added to interparticle distances to avoid infinite energy values	system is insensitive to values near machine precision
σ	1.0	effective particle radius	constant for all results in this dissertation
γ	0.156	defines range of normals for which the particle energies are smoothed	insensitive to exact value as long as $\gamma \leq 1$
λ_0	1.0	initial stepsize value	system is insensitive to this value
λ_{min}	10^{-14}	minimum stepsize value	system is insensitive to this value as long as it is sufficiently small
s	user-defined	planar separation value that specifies desired interparticle distances across a flat plane	robust system behavior relies on this value to be small enough such that Euclidean distance measures closely approximate geodesic distances
ρ	user-defined	density of particles per unit angle over a curved surface	system is insensitive to this value
ϵ_F	10^{-5}	surface threshold	value should be around the machine precision value
	5	number of initial projections of a particle	system is insensitive to this value as long as particles get to within ϵ_F of the surface
	0.15%	system energy difference from previous iteration that indicates a steady state	value must be small enough such that the particle distribution converges to an even packing
	50	number of iterations when system automatically checks for a desirable configuration (Step 2d)	value must be large enough such that local particle neighbors can be established
	0.35	percentage of E_{ideal} that indicates a particle should be split	values with approximately 20% of this value produce visually similiar results with different convergence times

Table 3.3. Table of free parameters, *cont*.

Parameter	Value	Description	Comments
	1.75	percentage of E_{ideal} that indicates a particle should be deleted	values with approximately 20% of this value produce visually similiar results with different convergence times
	5	number of iterations when particle neighbor lists are updated	values < 10 maintain stability in the system
	$4w$	initial value of s based on the width of a voxel, w	this value must be small enough such that Euclidean distance computations closely approximate geodesic distances

We have also observed that the particle system quickly eliminates the high frequency errors in its configurations by taking large movements during the first few iterations, followed by many iterations of small movements to eliminate the low frequency errors. To exploit this trend, we have developed a hierarchical distribution mechanism. We first distribute coarse level particles across the system using a large s value, then split each of these particles into four evenly spaced particles and reduce s by half. This process is repeated until s reaches the user defined value. We have found that an initial value of s based on the width of the volume voxels works most effectively.

3.8 Rendering

Rendering a particle system with oriented disks provides the user with a visualization of accurate surface samples. The disks allow the user to infer the topology of the surface, aided by the ability to rotate and translate the surface in 3D. However, the oriented disks cannot express subtle shading cues or connectivity as effectively as rendering methods that generate water-tight surface visualizations.

Over the last several years, work on point set surfaces has established points as a popular, and effective, geometric primitive. One common approach for rendering point sets is the splatting method, first introduced by Pfister *et al.* as *surfels* [115], with numerous extensions for improving the quality of the rendering [170, 171], as well as the speed [29]. Splatting entails an additive blend of oriented, alpha-channel Gaussian kernels at each point, followed by a normalization of each pixel's color to ensure an even intensity across the surface. The normals of the points

can also be included in the projection and blending to allow for smooth shading of the surface. We have implemented the basic splatting algorithm on the GPU, achieving interactive rendering speeds — results are shown in Section 3.9. The adaptivity of the particle system allows the splat surfaces to appear smooth with relatively sparse point samples (sparse compared to typical point set surfaces that contain millions of points).

3.9 Results

In this section we present results of the proposed particle system for sampling a variety of implicit surfaces. All the data sets defined over a regular grid are reconstructed using an approximating kernel, described in Section 2.4. The distributions were computed on a 3.2GHz CPU with 2.0Gb of memory.

Figure 3.5 shows results of the proposed particle system sampling a sphere, torus, and box. The sphere and torus implicit surfaces are defined as the zerosets of analytic functions, while the box is an offset surface of the distance transform of a box. The distributions on the sphere and torus show regular, hexagonal packings of points over the surfaces, and the distribution on the box is adaptive to the curvature on the edges and corners. The plot in Figure 3.6 compares the number of particles distributed across the sphere with the convergence time of the system. From this plot, the linear behavior of the distribution time of the system with increasing numbers of particles is evident, which is due to the compact energy kernels and the system optimizations that take advantage of this characteristic.

The next example in Figure 3.7 is an implicit surface generated from a binary segmentation of white matter in a brain. Using the techniques discussed in Sections 2.5.1 and 2.5.2, the data were smoothed by first closing and then opening the binary volume using a *ball* stencil, followed

(a) (b) (c)

Figure 3.5. Particle distributions over simple surfaces. The surfaces in (a) and (b) are the zerosets of analytic functions, while (c) is a levelset of the distance transform of a box.

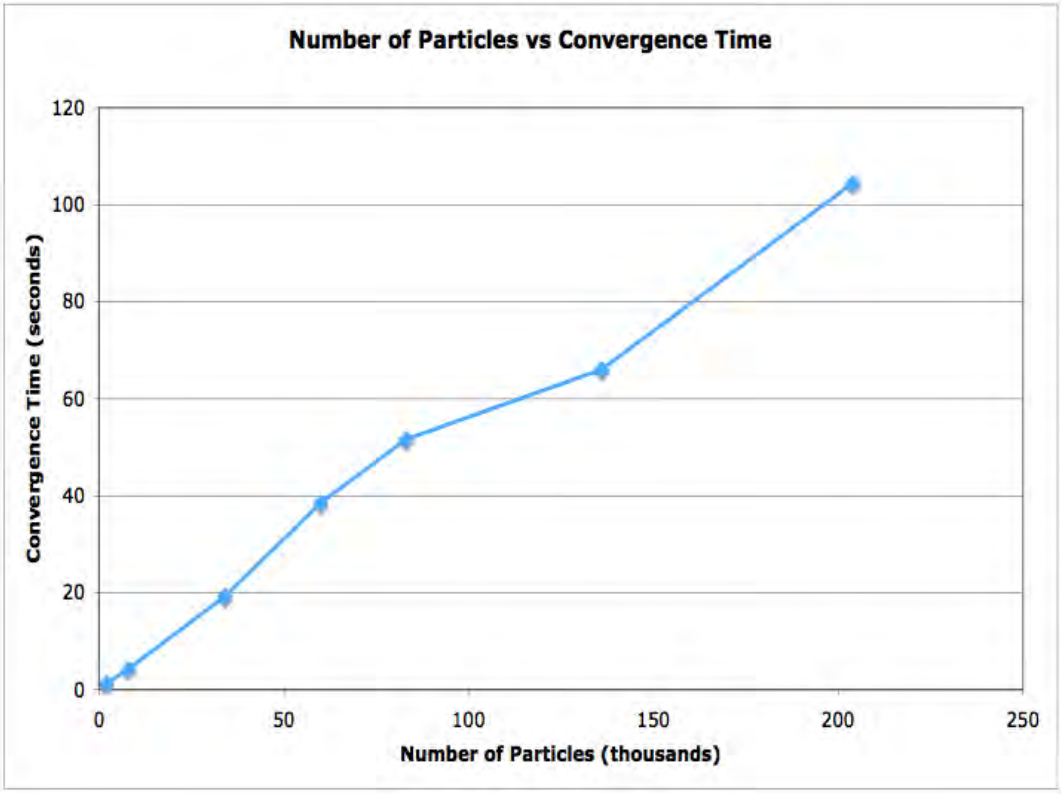


Figure 3.6. Comparison of the distribution time of the particle system over a sphere versus the total number particles in the system.

by a tightening of the surface with a maximum radius of curvature of 1.0. Figure 3.7 (a) shows a homogeneous packing of particles across the surface, and in Figure 3.7 (b) the particle system adapts to the curvature of the surface.

Figures 3.8 and 3.9 are surfaces reconstructed from the zerosets of distance transforms. These figures again present homogeneous as well as adaptive distributions of particles. Figure 3.8 (d) and Figure 3.9 (d) were generated using a GPU-based splat renderer, which allows these solid appearing surfaces to rendered at interactive rates. The GPU-based splat algorithm runs on an NVIDIA GeForce 6800 GT card with Pixel Shader 3.0.

In Table 3.9 we provide the number of particles for each of the examples in this section, as well as the time the particle systems took to converge. For the surfaces that are reconstructed from volume data, computing the curvature to adapt the particles increased the convergence time by roughly a factor of four over the distribution using the same value of s. For the surfaces in Figure 3.5, the jump in distribution time from the analytic surfaces to the box distance transform is due to the cost incurred by reconstructing volumetric data.

(a)

(b)

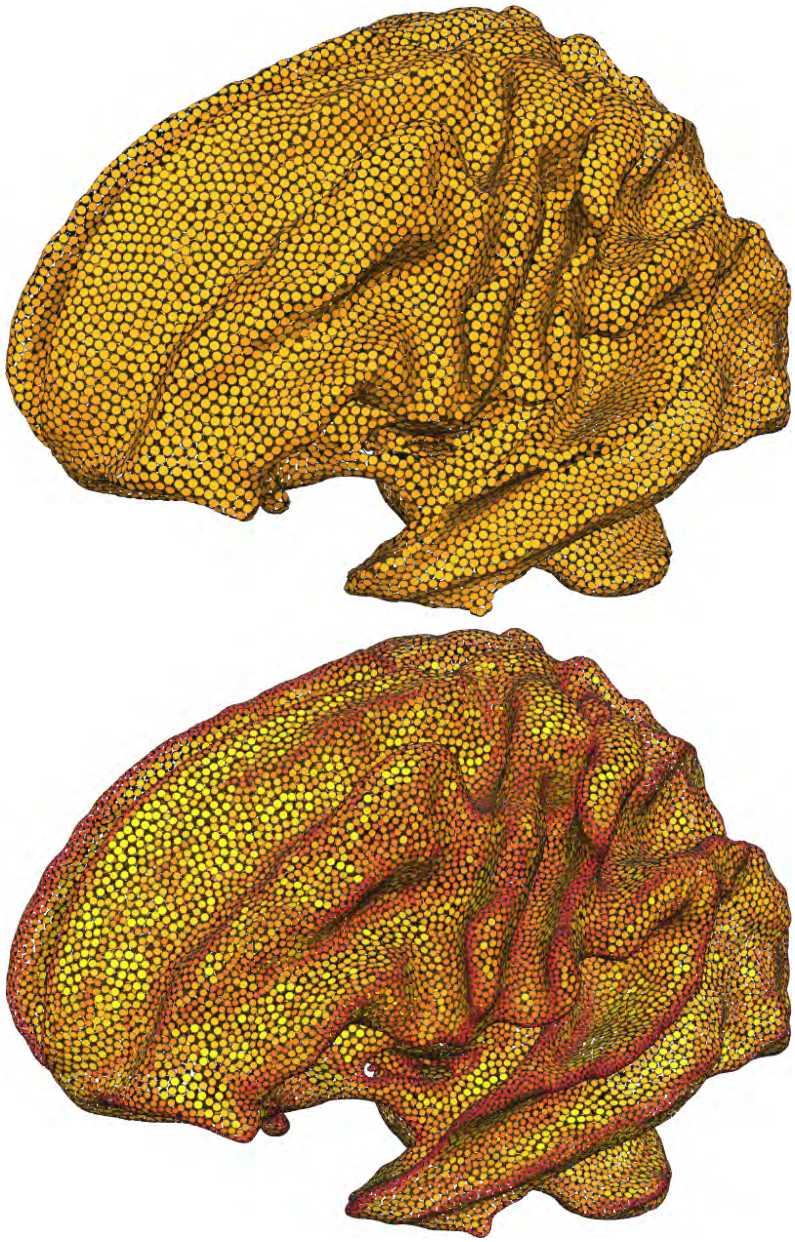


Figure 3.7. Particles on a brain reconstructed from a $149 \times 188 \times 148$ binary volume that was closed then opened with a *ball* stencil, followed by a tightening with $r = 1$. In (a) and (b) $s = 2$, with $\rho = 0$ in (a) and $\rho = 5$ in (b).

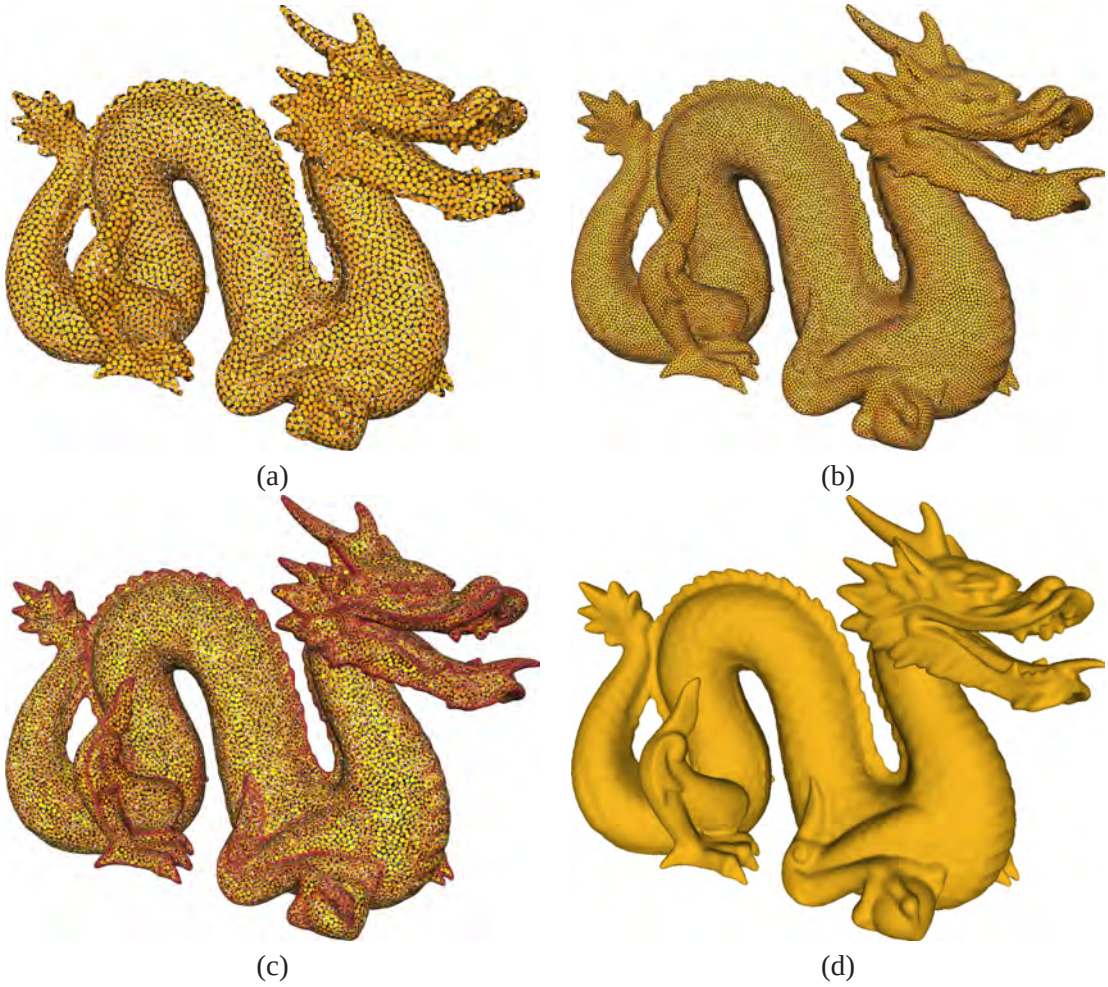
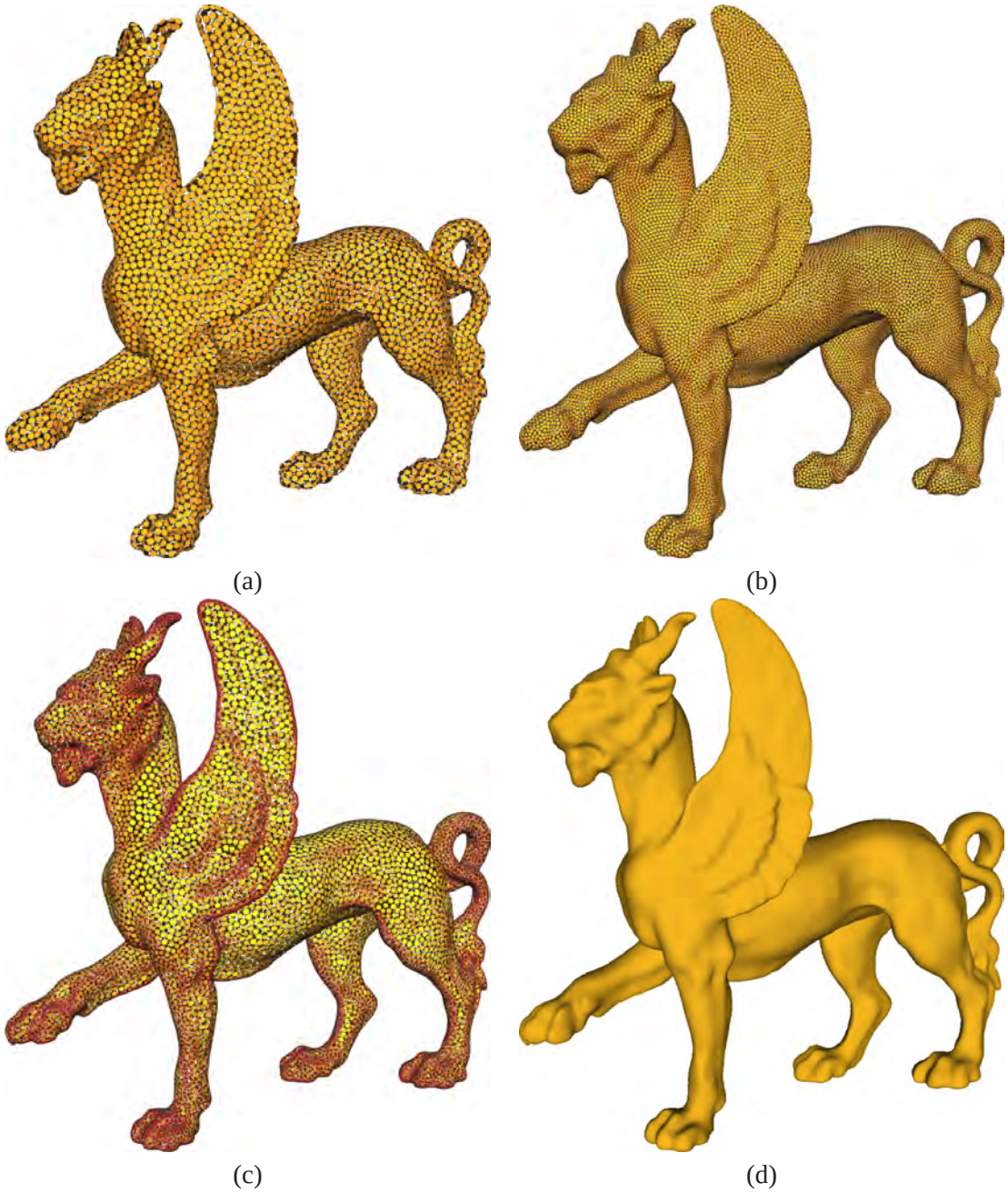


Figure 3.8. The dragon dataset is a $356 \times 161 \times 251$ distance transform. In (a) and (b) $s = 4$ and $s = 2$, respectively, with $\rho = 0$, and in (c) and (d) $s = 2$ and $\rho = 7.5$. Image (d) is a splat rendering of the particles in (c).

3.10 Discussion

In this chapter we have presented a new particle system framework for robust, adaptive sampling of complex implicit surfaces. We have developed a new class of energy functions and applied several numerical techniques to generalize, stabilize, and control the distribution of particles. The physically-based nature of the particle system inherently incurs a high computational cost, and the proposed system provides the tools necessary to automate the particle convergence reliably for the production of compact sets of surface samples.

There are numerous avenues to improve the proposed system. Although we have presented several optimization strategies, more work can be done to increase the efficiency of the distribution process. First, the development of an adaptive spatial binning structure would reduce the



(a) (b)

(c) (d)

Figure 3.9. The griffin dataset is a $104 \times 48 \times 98$ distance transform. In (a) and (b) $s = 2$ and $s = 1$, respectively, with $\rho = 0$, and in (c) and (d) $s = 2$ and $\rho = 7.5$. Image (d) is a splat rendering of the particles in (c).

Table 3.4. Number of particles for each of the distributions presented in Section 3.9, as well as the time required for the particle systems to converge.

Dataset	Number of Particles	Time
sphere	492	1 second
torus	1184	1 second
box	1049	26 seconds
brain (a)	31635	3.3 minutes
brain (b)	57139	11.3 minutes
dragon (a)	11011	2.3 minutes
dragon (b)	41879	6.2 minutes
dragon (c-d)	44042	13.5 minutes
griffin (a)	8036	1.5 minutes
griffin (b)	21189	3.6 minutes
griffin (c-d)	32144	7.2 minutes

number of potential neighbors at highly adaptive particles. Second, further exploitation of the system's tendency to quickly eliminate high frequency error in the distribution process could be done by posing the system in a hierarchical framework, such as multigrid. These advancements would help to enable visualization of dynamic data sets, where course level distributions can help guide fine level distributions from time step to time step.

CHAPTER 4

ISOSURFACE VISUALIZATION OF HIGH-ORDER FINITE ELEMENT DATA

This chapter uses the flexibility and controllability of dynamic particles for sampling implicit surfaces in the context of visualization. Visualization has become an important component of the simulation pipeline, providing scientists and engineers a visual intuition of their models. Simulations that make use of the high-order finite element method for spatial subdivision, however, present a challenge to conventional isosurface visualization techniques. High-order finite element isosurfaces are often defined by basis functions in *reference space*, which give rise to a *world space* solution through a coordinate transformation, which does not necessarily have a closed-form inverse. Therefore, world-space isosurface rendering methods such as marching cubes and raytracing must perform a computationally expensive *nested root finding*. To address these challenges we propose visualizing the isosurfaces with a particle system. This chapter describes a framework that allows particles to sample an isosurface in reference space, avoiding the costly inverse mapping of positions from world space when evaluating the basis functions. The distribution of particles across the reference space isosurface, however, is controlled by geometric information from the world space isosurface, such as the surface gradient and curvature. The resulting particle distributions can be homogeneous, or, adapted to accommodate world-space surface features, providing compact, efficient, and accurate isosurface visualizations of these challenging data sets.

The following section provides an introduction to high-order finite elements and the challenges they pose for isosurface visualization. This is proceeded by sections that provide background of isosurface visualization methods, as well as the proposed adaptations of the particle system scheme for visualizing high-order data sets. A derivation of surface curvature in the presence of curvilinear coordinate transformations is given in Section 4.3.1. The chapter concludes with a comparison of results from the proposed particle system approach with other visualization techniques.

4.1 High-Order Finite Elements

The method of finite elements [67] is a common spatial subdivision scheme used by scientists and engineers to reduce large simulation domains to sets of small subdomains over which physical simulations can be computed robustly and efficiently. While traditional finite element methods utilize only low-order, linear basis functions for representing data over the elements, they provide considerable flexibility for handling complex geometries. The geometric flexibility is aided by the transformation of individual elements constructed as identical cubes in *reference space* into unique *world space* elements, which can have not only rectangular faces, but also triangular faces. In world space, the spatial extent of each element is defined by characteristics of the domain and simulation, such as boundary conditions and features of interest. The mapping functions responsible for the transformations can distort the elements by stretching, skewing, or even collapsing the faces of the reference space cubes, as illustrated in Figure 4.1.

A number of researchers have developed methods to improve the convergence properties of finite elements through the use of high-order functions for the representation of the data, as well as the element transformations. Today, high-order finite element techniques have reached a level of sophistication such that they are commonly applied to a broad range of engineering problems [42, 74, 138]. Although there do exist some high-order finite element methods that do not rely on reference space transformations, the use of curvilinear coordinate transformations is of increasing interest [68]. This chapter is addressing the problem of finite element methods that rely on higher order (higher than linear) basis functions for the solutions as well as the coordinate transformations.

Conventional approaches to finite element isosurface visualization assume that linear data

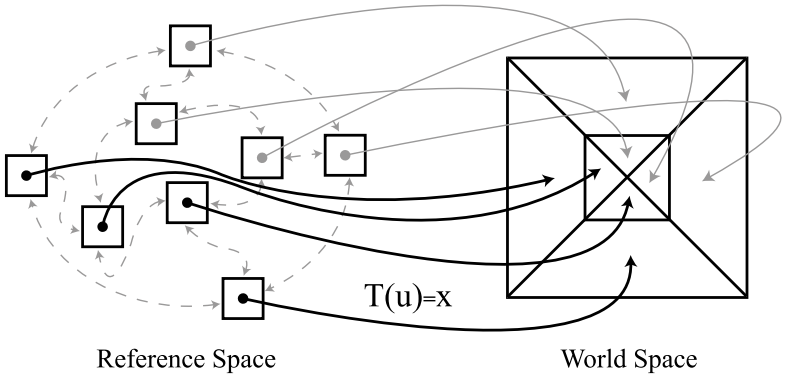


Figure 4.1. A 2D schematic of the finite element subdivision scheme. In *reference space*, elements are defined as identical squares. These squares are transformed into *world space* by a mapping function T that can stretch, skew, shrink, or even collapse edges.

Figure 4.2. An isosurface of a finite element fluid simulation pressure field sampled with a particle system. The color indicates the relative direction of the surface normal at the particle (blue indicates *outward* and red indicates *inward*).

representations can be adapted to accommodate low-order finite elements. This strategy, however, faces a number of challenges when considering high-order data sets. First, the data must be finely subsampled to ensure that features are adequately captured with linear approximation schemes. Second, there is, in general, no closed form expression for the inverse of high-order mapping functions. Numerical inversion schemes are required to transform world space locations into the reference space when sampling the data, creating a nested root-finding problem when locating an isosurface. Furthermore, determining which reference-space element in which to invert a particular point in world space adds to the computation.

Computational scientists who wish to visualize high-order finite element solutions will require visualization algorithms that are flexible enough to accommodate these constraints. These algorithms will need to have variable degrees of freedom so that users can easily control the trade-off between visualization quality and speed. For efficiency, these computations must be locally adaptive, allowing computational power to be applied to regions of the solutions that exhibit the most complexity (*i.e.*, *h-r adaptivity* in finite element terms). Furthermore, these algorithms will need to achieve the appropriate balance of computations in world space, where the metrics for adaptivity are defined, and reference space, where there are closed-form expressions for the associated geometric quantities. To address these issues we are proposing an isosurface visualization technique that relies on a particle system, exhibited in Figure 4.2. The particles are constrained to an isosurface and exert repulsive forces on each other, resulting in even distributions across the surface.

In Chapter 3, we showed how these types of systems can be made robust and controllable while also adaptive to features of interest. For isosurfaces embedded in high-order finite elements, adaptivity based on the curvature of the isosurface in world space is a function of not only the high-order basis functions, but also the mapping function. A contribution of this chapter

is the derivation of isosurface curvature in the presence of curvilinear coordinate transformations, including a reduction of the isosurface Hessian from a rank-three tensor contraction into a series of standard vector-matrix computations. Also, we propose a method for manipulating the particle positions *in reference space* to avoid a numerical inversion of the coordinate transformation, while computing the particle interactions and adaptivity in world coordinates.

The resulting particle system allows for a series of *forward* computations to obtain desirable distributions of samples that are accurate and compact over the world space surface. The resulting distribution of particles — a process that may take anywhere from a few seconds to minutes — can be rendered interactively as either simple point sprites or as a water-tight *splat* surface on the GPU, allowing a scientist or engineer to quickly explore their data from any camera location. Furthermore, the generality of this system can be broadly applied to any type of data representation that makes use of a reference space and a mapping function.

4.2 Background

This section first presents a brief overview of techniques in the literature for directly sampling implicit surfaces. The discussion is followed by an analysis of the challenges for adapting traditional visualization methods to high-order finite element data, along with a review of existing methods for visualizing these data sets.

4.2.1 Direct Visualization of Implicit Surfaces

In scientific, medical, and engineering applications, visualization has become an integral part of simulation and analysis pipelines. As the size of data sets continues to swell with the increase in computing capacities and scanning resolutions, visualization tools are have become invaluable for recognizing features in otherwise dizzying amounts of information [69]. Oftentimes, data from scanning devices or simulations come in the form of a volumetric lattice of values, which can then be interpreted as an implicit function. When directly visualizing surfaces and boundaries embedded in these implicit functions, methods typically excel in one of two orthogonal characteristics: they either produce a high quality visualization, or they are computationally efficient. These competing paradigms have brought forth two general classes of methods — those that are image-space based, and those that are object-space based.

Image-space methods visualize implicit surfaces by sampling the surface from the reference frame of the image plane in the virtual camera model. Ray tracing, for example, traces the path of individual rays from a virtual camera into each pixel of the output image and through the implicit function, computing intersections of the ray with the implicit surface [60]. Determining

the intersections is done via root-finding along the ray, either by solving for the roots directly if the functional form of the implicit function is known, or, by an iterative method such as Newton-Raphson, subdivision, or interval arithmetic [98]. Once the intersection points are computed, advancing shading effects can be generated using the derivatives of the implicit function at the point location [157], producing photo-realistic or highly stylized images [55].

Another image-space approach to visualizing implicit functions is volume rendering [89], which assigns a color as well as opacity to different levelsets of the function. By blending the values of the levelsets together, the result is a volumetric effect that allows several isosurfaces to be viewed at once. The original method, called ray casting, is a direct extension of ray tracing, and was proposed independently by Drebin and Levoy [48, 87]. Ray casting composites the color and opacity values of interpolated data as rays, originating from a virtual camera, traverse the implicit function. Methods for preprocessing the color and opacity values assigned to an implicit function, such as shear-warp [82] and splatting [155], have been developed to optimize this volume rendering algorithm.

Although image-space approaches generate stunning visualizations, they are computationally expensive for two reasons: first, the algorithms are inherently dependent upon the view point, thus every new view of the data must be recomputed; and second, the sampling of the implicit surface is dictated by the resolution of the final image and the data dimensions, not by the complexity of the surface itself. Extensions to these image-based methods exist to increase the rendering efficiency, but require multiprocessor machines [111] or specialized hardware [114, 25].

Object-space methods, on the other hand, first sample a surface in the reference frame of the virtual world, allowing the sample points to be determined as a preprocessing step. Sampling the surface directly in object-space can be done by projecting a set of points onto the zeroset of the implicit function using a technique like gradient descent. Because the samples are view-independent, the surface can be visualized from many different viewpoints without a computationally expensive resampling, making these methods effective for exploring scientific data on commodity desktop (and laptop) machines. The surface point samples can be rendered as simple point primitives, as disks oriented tangentially to the surface, or using a more sophisticated point-based rendering technique, such as splatting.

4.2.2 Low-Order Visualization Methods for High-Order Data

Until recently, much of the work in finite-element simulations has focused on linear elements. In the simplest case where the finite elements form a regular grid in world space, conventional methods like marching cubes [90] and direct volume rendering [91] are applicable. In general,

however, the finite elements produce an irregular grid in world space that is incompatible with the assumptions these methods make about the regularity of the grid. Early work by Shirley and Tuchman [136] and Williams [161] proposes a volume rendering approach for tetrahedral elements, and Bunyk *et al.* [23] propose a generalized ray-casting algorithm for irregular grids. Doi and Koide [47] present the Marching Tetrahedra method for triangulating isosurfaces defined over tetrahedral elements. More recently, work has moved the volume rendering [154, 26] and isosurface generation [122, 112] algorithms onto the GPU to obtain faster rendering speeds.

Applying these low-order, linear methods to high-order finite elements, however, presents several challenges. First, high-order basis functions represent features in the data with far fewer grid elements than an equivalent low-order representation. Thus, visualization methods that rely on linear interpolation must first finely subdivide the domain to ensure that features in the data are not missed. This increase in grid resolution can have an explosive effect on not only the storage requirements for the visualization, but also on the computation required to sufficiently sample the elements.

The second problem stems from the need to compute an inverse of the mapping function to evaluate the data in the world space. Let $F^*(\mathbf{u})$ be the functional representation of the finite element solution, which is defined in the reference space, and let T be the coordinate transformation that maps a reference space point $\mathbf{u}$ into a world space point $\mathbf{x}$, *i.e.*, $T(\mathbf{u}) = \mathbf{x}$. The world space representation of the solution is therefore $F(\mathbf{x}) = F^*(T^{-1}(\mathbf{x}))$. There is generally no closed form expression for T^{-1}, causing world-space evaluations to require iterative numerical schemes for the computation of $\mathbf{u} = T^{-1}(\mathbf{x})$. Thus, determining the location of isosurfaces in high-order finite element data becomes a nested root finding problem – $F(\mathbf{x})$ (and its derivatives) must be iteratively evaluated to determine the position of the isosurface, with each evaluation of $F(\mathbf{x})$ requiring an iterative, numerical inversion of T.

There is yet one more challenge for the general problem of visualizing high-order finite elements. The data and coordinate transformations are valid for only a single element, and in practice another layer of computation is required to determine which reference element contains the point $\mathbf{u} = T^{-1}(\mathbf{x})$. Although efficient element lookups based on regular grids [125, 53] or low order curves [159] can be applied to elements with planar or quadratic curved faces, respectively, there is no closed form solution for the general problem. Spatial partitioning schemes can be utilized to reduce the grid ambiguity to among a few elements [103], but the inverse mapping $\mathbf{u} = T^{-1}(\mathbf{x})$ will (generally) require multiple iterations across multiple elements. The problem is becoming increasingly more difficult as results from the scientific computing literature extend

the finite element methodology to more general frameworks. For instance, recent work by Hughes *et al.* [68] proposes spline-based functions for finite elements, which produces yet another class of curvilinear mappings between the reference and world domains.

Adapting marching cubes to accommodate high-order finite elements elucidates these three challenges. In Figure 4.3 we present the results of the isosurface extraction technique applied to a sphere that is transformed through a $2 \times 2 \times 2$ set of quadratic b-spline functions. To generate these results, a regular world space grid is first created. The world space location of each grid node is numerically inverted within each potential element until the associated reference space location is determined and the basis functions can be evaluated. Accurately finding the zeros of the high-order data along the grid is then accomplished via a root-trapping mechanism.

We have incorporated two different root-trapping methods into a marching cubes framework. The first is a grid refinement strategy that uses an adaptive subdivision scheme to be as efficient as possible, recursively subdividing only the grid cells that contain zero crossings. Care has been taken in the implementation of the subdivision scheme to ensure coherence across neighboring cells, avoiding redundant sampling of the data. The the second root-trapping approach uses the Newton-Rhapson method along grid edges to determine the zero crossings.

Ensuring that point samples of the isosurface—for any sampling scheme—lie on the surface to within a small error tolerance is important for generating accurate surface approximations. In Figures 4.3(b-d) the vertex locations are computed using linear interpolation over progressively more refined grids. These results indicate that using low-order interpolation schemes requires a very finely subdivided grid to accurately determine the zeros of the data and capture the geometry of the surface (Figure 4.3(d)). In Figures 4.3(e-g) the vertex locations are computed using a Newton-Rhapson root-finding method. While the grids in these images are relatively coarse, the zeros of the data are more accurately computed, generating more precise approximations of the surface.

While Figure 4.3 illustrates that capturing the geometry of high-order data is possible with low-order schemes, the results come at the cost of lengthy compute times. The computations are dominated by the large number of mapping function evaluations, which we call *forward evaluations*. The surfaces in Figures 4.3 require a numerical inversion of each grid node, and the surfaces in Figures 4.3(e-g) also incur nested root-finding evaluations along the grid edges. The number of forward evaluations and timings are given in Table 4.1 for results generated on a P4 3.2GHz CPU with 2.0Gb of memory. It is interesting to note that Figures 4.3(e-g) out perform Figure 4.3(d), indicating that root-finding along coarse grid edges is more efficient than linearly

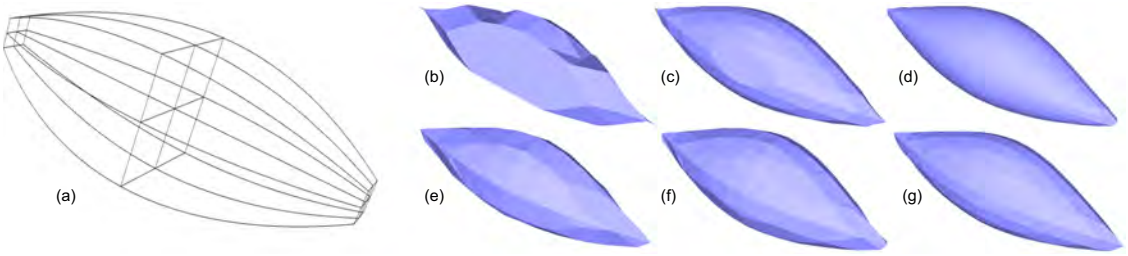


Figure 4.3. Marching cubes surfaces of a sphere mapped through quadratic b-spline functions: (a) the transformed elements; (b-d) surfaces generated using an adaptive subdivision root-trapping scheme; (e-g) surfaces generated using Newton-Raphson root-trapping. Grid dimensions, number of forward evaluations, and timings are given Table 4.1.

Table 4.1. The number of forward evaluations and timings for the marching cubes meshes in Figure 4.3.

Mesh	Grid Resolution	Root-trapping Method	Number Forward Evaluations (millions)	Time (seconds)
(b)	$5 \times 5 \times 13$	subdivision	0.6	6.7
(c)	$9 \times 9 \times 25$	subdivision	2.4	25
(d)	$17 \times 17 \times 49$	subdivision	7.7	82
(e)	$5 \times 5 \times 13$	Newton-Raphson	1.7	19
(f)	$7 \times 7 \times 17$	Newton-Raphson	3.2	35
(g)	$9 \times 9 \times 25$	Newton-Raphson	6.8	73

interpolating along refined grid edges.

4.2.3 High-Order Visualization Methods

Other researchers have also noted the challenges of efficiently adapting low-order visualization methods to high-order functions, and some work has been done to specifically address the problem of visualizing high-order finite element data. Wiley *et al.* [159, 158] formulate raycasting for curved-quadratic elements, and Brasher and Haimes [21] propose a GPU-based method for color mapping cut planes of quadratic and cubic elements. A method to subdivide elements containing high-order basis functions so that low-order visualizing methods can be used is proposed [130, 145]. Coppola *et al.* [33] address the issue of vector visualization with high-order representations by formulating the particle advection problem on high-order basis functions. Similar to our framework, this work tracks particle advection in the reference space to avoid the inverse mapping problem. More recently, Nelson and Kirby [103] present an algorithm for raytracing high-order, spectral/*hp* elements. Their method uses a world-space approximation

of the composition of the coordinate transformation and the reference space basis functions. It assumes multilinear mappings (linear element boundaries in world space), and includes a quantification of the approximation and root-finding error. They show that the image-space method compares favorably with marching cubes in compute time when the tolerances on surface position are sufficiently high.

4.3 Adaptation of the Particle System Framework

In this section curvature metrics for surfaces that exist in curvilinear are derived, as well as a method for sampling the isosurfaces in reference space that results in regular distributions of particles in the world space.

4.3.1 Isosurface Geometry in Finite Elements

To adaptively distribute particles across a finite element isosurface, we must formulate the gradient and Hessian of the world space implicit function, F, in terms of the reference space specifications that are given by the finite element basis and mapping functions. As mentioned in Section 4.2.2, the implicit function representing the simulation data, F^*, is defined over a set of finite elements in reference space and is transformed into world space through a mapping function $T(\mathbf{u}) = \mathbf{x}$. The gradient and Hessian of the world space implicit function are thus defined by not only F^*, but also by T, and therefore T must be included in all of the derivative calculations.

Used in the computation of the world space gradient and Hessian is the Jacobian of the mapping function, which describes how the space around a reference space position is stretched or squashed by the mapping function:

$$\mathbf{J}(\mathbf{u}) = \frac{\partial T(\mathbf{u})}{\partial \mathbf{u}} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial w} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial w}. \end{pmatrix} \tag{4.1}$$

Also, to simplify the following derivations we denote the inverse of the Jacobian as

$$\mathbf{K} = \mathbf{J}^{-1} \tag{4.2}$$

and provide a linear algebra identity for a matrix $\mathbf{M}$:

$$\frac{d\mathbf{M}^{-1}}{d\mathbf{z}} = -\mathbf{M}^{-1}\frac{d\mathbf{M}}{d\mathbf{z}}\mathbf{M}^{-1} \tag{4.3}$$

where $\mathbf{z}$ denotes some Cartesian basis vector.

Formulating the expressions for the world space gradient and Hessian requires meticulous derivations of the derivatives of the coordinate transformation, T. Care must be taken to correctly determine the order of the component multiplications as well as which vectors and matrices need to be transposed. Furthermore, the derivation of the Hessian includes a vector multiplication with a rank-three tensor, which is the result of computing the second derivative of the vector-valued coordinate transformation. To clarify these derivations we use the Einstein notation convention, as described in [14]. Developed for dealing with curved spaces in physics, Einstein notation identifies relationships often hidden by conventional linear algebra notation, such as transposition and order of operations.

To begin, we define the world space as X, the reference space as U, and the mapping function as $T : U \mapsto X$; let $\mathbf{x} \in X$ and $\mathbf{u} \in U$. Using Einstein notation, we define $\mathbf{x} = x^i$ and $\mathbf{u} = u_i$, utilizing upper indices for world space components and lower indices for reference space components. The number of indices of a variable indicates the rank of the tensor – v_i is a rank-one tensor (or vector); M_{ij} is a rank-two tensor (or matrix); T_{ijk} is a rank-three tensor, and so on. In this convention, repeated indices in a term indicates a summation over the range of index values. For example:

$$\underbrace{\mathbf{a}^\mathsf{T} \cdot \mathbf{b}}_{vector} = \underbrace{\sum_i a_i b_i}_{summation} = \underbrace{a_i b_i}_{Einstein\ notation} . \tag{4.4}$$

Parenthese are also dropped on functional variables, such as from the basis functions and the mapping function:

$$F(\mathbf{x}) = F^i x^i \tag{4.5}$$

$$F^*(\mathbf{u}) = F^*_i u_i \tag{4.6}$$

$$T(\mathbf{u}) = T^i_j u_j = x^i. \tag{4.7}$$

For brevity, however, we will notate Equations 4.5 and 4.6 as simply F and F^*, respectively, emphasizing the role of these equations as scalar functions throughout the derivations. Finally, Einstein notation also uses indices to represent partial derivatives, such as for the Jacobian of the mapping function:

$$J(\mathbf{u}) = \frac{\partial}{\partial u_j} T_k^i u_k = \frac{\partial x^i}{\partial u_j} = J_j^i. \tag{4.8}$$

The inverse of the Jacobian is then

$$K_j^i = (J_j^i)^{-1}. \tag{4.9}$$

The world space gradient, F^i, is:

$$F^i = \frac{\partial F}{\partial x^i} = \frac{\partial F^*}{\partial u_j} \frac{\partial u_j}{\partial x^i} = F_j^* K_j^i. \tag{4.10}$$

The expression of the Hessian includes the derivative of the inverse Jacobian which, using Equation 4.3, is written:

$$K_{jk}^i = \frac{\partial (J_j^i)^{-1}}{\partial u_k} = -K_j^l J_{mk}^l K_m^i. \tag{4.11}$$

Notice that the second derivative of the mapping function, J_{mk}^l, is a rank-three tensor. Using Equations 4.10 and 4.11 the Hessian, F^{ij}, is defined as:

$$\frac{\partial}{\partial x^i}\left(\frac{\partial F}{\partial x^j}\right) = \frac{\partial}{\partial u_k}\left(\frac{\partial F}{\partial x^j}\frac{\partial u_k}{\partial x^i}\right) = F_{lk}^* K_l^j K_k^i - F_l^* K_l^m J_{kn}^m K_n^j K_k^i. \tag{4.12}$$

By carefully matching Einstein notation indices of each component in Equations 4.10 and 4.12, the gradient and Hessian expressions are reduced to a series of standard vector-matrix operations. Introducing $[\cdot \mid \cdot \mid \cdot]$ to express the concatenation of three column vectors into a matrix, the expressions for the world space gradient and Hessian in standard notation are:

$$F_{\mathbf{x}} = K^{\mathsf{T}} F^*_{\mathbf{u}} \tag{4.13}$$

$$F_{\mathbf{xx}} = \left(F^*_{\mathbf{uu}} K\right)^{\mathsf{T}} K - \left(\left[F_{\mathbf{x}}^{\mathsf{T}} \frac{\partial J}{\partial u} \mid F_{\mathbf{x}}^{\mathsf{T}} \frac{\partial J}{\partial v} \mid F_{\mathbf{x}}^{\mathsf{T}} \frac{\partial J}{\partial w} \right] K \right)^{\mathsf{T}} K. \tag{4.14}$$

The expressions conveyed in Equations 4.13 and 4.14 allow world space adaptivity to be achieved using reference space evaluations of the basis functions, mapping functions, and their derivatives, along with any standard vector-matrix library. This formulation has general applicability and could, for instance, be used to add curvature dependencies in other applications that make use of reference space regularity, such as mesh generation [134].

4.3.2 Reference Space Particles

To avoid the computations associated with T^{-1} we have developed a strategy that maintains particle positions in reference space, while controlling the particle distributions with geometric information from the world space. This strategy is similar to the guided search algorithm proposed by Coppola *et al.* [33] for particle advection. Maintaining reference space positions allows the basis functions that define F^* to be sampled using closed form expressions. Thus, the curvature calculations and projection of particles onto the isosurface (Equation 3.12) can be computed directly. We ensure that the particle distributions are even and adaptive in the world space by using world space positions (forward mapping) when computing the repulsive force velocities in Equation 3.4. We then transform these velocities via the derivative of the mapping function to obtain an approximate reference space velocity, $\mathbf{v}^*_i$:

$$\mathbf{v}^*_i = \mathbf{K}\mathbf{v}_i \tag{4.15}$$

The reference space positions are then updated using Equation 3.11 with reference space components. We note that Equation 4.15 is a first-order approximation, and thus assumes updates that are small relative to the isosurface and the curvature of the coordinate transformation. We are not interested in the precise motion of particles, but rather that they move to lower energy states. Thus we adapt the time steps to accommodate this first-order approximation, as described in Section 3.4.

To effectively distribute particles in the reference domain, particles must also be able to move from element to element. Finite element data sets typically contain reference space information describing which elements abut one another in the world space, as shown by the dashed lines in Figure 4.4. Because reference space elements are identical cubes, we can easily determine when a particle's positional update causes it to leave an element, and use the adjacency information to compute the particle's new element and new position within that element. When a particle moves from one element into the next, we use linear interpolation—based upon the reference space coordinates of the adjacent elements' shared world space vertices—to determine the particle's location in the neighboring element's local coordinate system.

4.4 Boundary Discontinuities

Surfaces defined over finite elements are usually guaranteed to be only C^0 continuous at the element boundaries, allowing for cusps on a surface in areas that are analytically flat. This potentially results in lessened interparticle forces due to the tangent plane projection of a particle's neighborhood force field. The iterative nature of the convergence mechanism, however, smooths out this effect.

In practice we have found the discontinuities present a problem only when rendering the particles. The particles' radii do not adapt to these discontinuous features, causing artifacts to sometimes appear when disks or splats intersect each other – Figure 4.5 illustrates the phenomenon along boundaries. To more accurately capture these surface features, the particles can be adapted in Equation 3.15 by not only the curvature computed from the Hessian, but also by the particles' proximity to element boundaries. This adaptation will not remove the visual presence

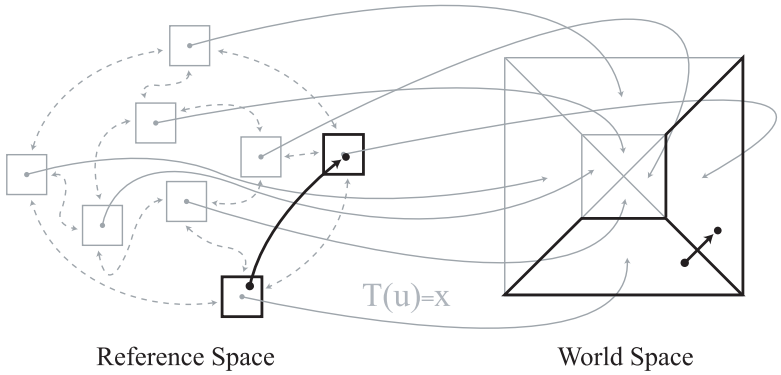


Figure 4.4. Particles move from element to element by utilizing neighboring element information (dashed lines) and linear interpolation of the reference space element vertices to determine the coordinates of the particle in the neighboring element.

of the cusps but will make the discontinuities appear smoother.

4.5 Implementation

This implementation is a modified version of that found in Section 3.6. Changes to the basic particle system implementation are in **bold**. For all of the results presented in this chapter we initialized the system with several hundred particles at random locations, evenly distributed among all finite elements. The particles maintain reference space positions as well as the corresponding world space positions.

1. For each particle:

   (a) Compute E_i and compute $\mathbf{v}_i$ with Equation 3.4 **by summing the repulsive forces and energies of all the neighboring particles in world space. Transform the velocity into reference space with Equation 4.15**.

   (b) **Compute** $\mathbf{v}_i^{*\text{new}} = \lambda_i \mathbf{v}_i^*$.

   (c) Compute the new particle position **in reference space** $\mathbf{u}_i^{\text{new}}$ **by solving Equation 3.11 with** $\mathbf{v}_i^{*\text{new}}$, followed by a reprojection to the surface by solving Equation 3.12. **If the new position is outside of the element, determine which element the new position is in, and convert the position into the new, local coordinate system.**

   (d) Compute the new energy value, E_i^{new}, at the new particle location $\mathbf{x}_i^{\text{new}}$ **by mapping the new reference space position into world space with** T, as well as the new implicit function value, F_i^{new}.

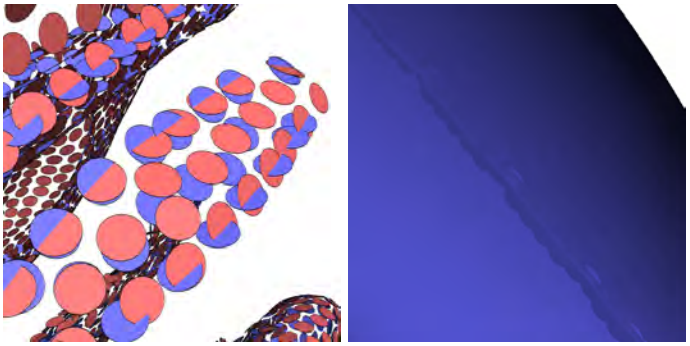


Figure 4.5. Artifacts due to boundary discontinuities where disks (left) or splats (right) intersect each other.

(e) If $E_i^{\text{new}} \geq E_i$ or $|\mathbf{x}_i - \mathbf{x}_i^{\text{new}}| > s$ or $F_i^{\text{new}} > \epsilon_T$, and $\lambda_i > \lambda_{\min}$, decrease λ_i by a factor of 10 and go back to Step 2(b)ii. If $\lambda_i \leq \lambda_{\min}$, do not move the particle and skip to the next particle in the list.

(f) If $E_i^{\text{new}} < E_i$, **update** $\mathbf{u}_i = \mathbf{u}_i^{\text{new}}$. Increase λ_i by a factor of 10 if this is the first time through Step 2b.

2. Decide whether the system is at a steady state. There are numerous metrics to determine steady state, and we have chosen to use the difference of the system energy (the sum of the energy at all the particles) from one iteration to the next. When the system energy difference is less than a small fraction of the total energy (we use 0.15% for the results presented in this dissertation, although a range of values would produce similar results), we deduce that particles have reached a steady state. Otherwise, repeat Step 2b. For poor initializations, we have found it can be useful to skip to Step 2d every 50 iterations instead of waiting for the system to reach a steady state.

3. Check whether the configuration of particles is desirable. We compare each particle's energy against an ideal energy, E^{ideal}, which is defined by a hexagonal packing of neighbors on a flat surface with inter-particle distances of s. We bias E_i with a random value on the interval $[0, 1]$ to eliminate mass splitting or dying, then split particles with $E_i < 0.35E^{\text{ideal}}$, and delete particles with $E_i > 1.75E^{\text{ideal}}$. Alternatively, if a constant number of particles is desired, the planar separation variable s could be modified to move the system energy towards the ideal system energy. While we have provided specific values for this step, we have found in practice that varying the values by up to 20% produces visually similar results, although with different convergence times.

4. If the energy of the particles is acceptable, stop iterating.

4.6 Results

We begin with a demonstration of the method on a curvilinear mapping of a simple implicit function. Figure 4.6 shows a quadratic b-spline coordinate mapping function, which maps a reference space sphere into a world space teardrop, introducing a curvature variation across the surface. Our mathematical formulation of the world space curvature correctly accounts for the effects of the mapping function.

Figure 4.7 demonstrates the particle system achieving an even distribution on a sphere in the world space, despite the irregular reference space geometry. Figure 4.6 asserts that the system

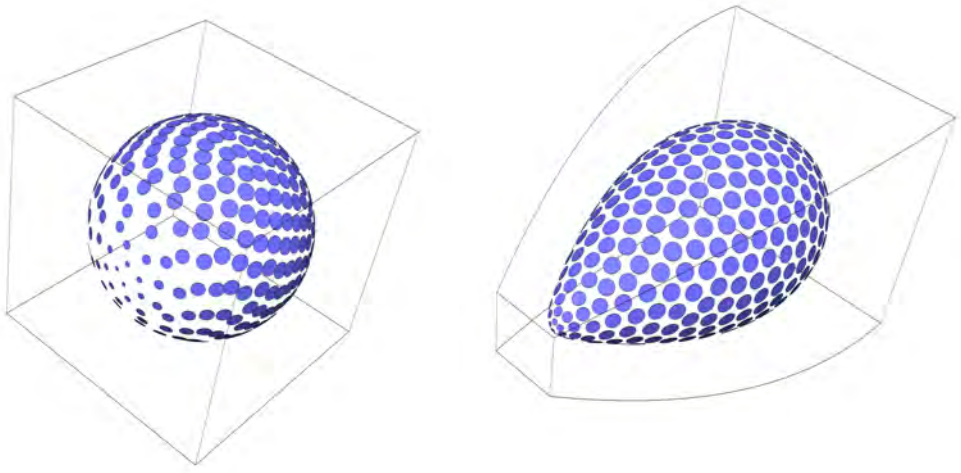


Figure 4.6. A sphere defined in reference space (*left*) is mapped to a teardrop in world space (*right*). The mapping function induces a curvature variation to the surface, and the particles adapt accordingly.

can correctly adapt to the world space curvature introduced by the mapping function. All of these results are produced by manipulating the positions of particles in the reference space, while computing inter-particle distances and curvatures using world space geometries.

The particle system visualizations in Figure 4.8 are generated from the data set used in the marching cubes example shown in Figure 4.3. These examples are provided as a comparison of what the particle system achieves in approximately the same amount of time as marching cubes with root-finding. Using the same computing resources specified in Section 4.2.2, Figure 4.8(a) requires 16 seconds and 1.1 million forward evaluations; Figure 4.8(b) requires 36 seconds and 2.5 million forward evaluations; and Figure 4.8(c) requires 70 seconds and 4.8 million forward evaluations. The particle system is able to capture the sharp tips of the surface faithfully due to the adaptivity mechanism, even at coarse resolutions – this is a significant difference from the polygonal surfaces. Furthermore, the splat renderings produce much smoother water-tight surface approximations than the marching cubes results.

These results allow us to compare directly against marching cubes, elucidating the advantage of the particle system. First, notice that the particle system produces higher quality results in similar amounts of time with *fewer* forward evaluations (compare Figures 4.3(e-g) with Figure 4.8). This is in part due to marching cubes evaluating grid nodes in regions where the surface does not exist, but also due to the numerous iterations at each grid intersection point to determine the correct placement of the vertex, a chore that consumes over half of the compute time. In contrast,

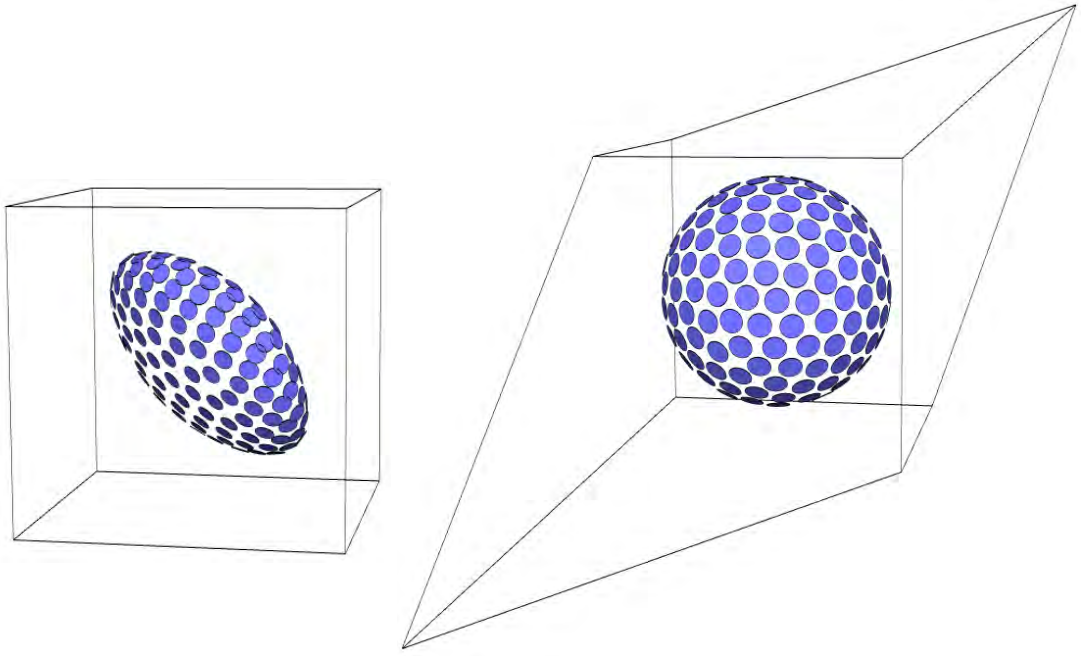


Figure 4.7. The particle system converges to an even distribution in world space (*right*) regardless of the shape of the surface in reference space (*left*).

the particle system spends most computation time evaluating the data (by these numbers, about 70%) and little time in the overhead of the particle system, *i.e.*, computing particle-particle interactions. These results, like those described in the literature [103], demonstrate that generating *accurate* visualizations of high-order data is inherently expensive due to the cost of evaluating the high-order solutions and coordinate transformations. By simultaneously finding the roots of F and distributing the surface samples in a sensible way, the particle system makes very effective use of these costly evaluations.

The raytracing method described in [103] generates nicely shaded and *water-tight images* of high-order finite element isosurfaces, but with the image-space drawbacks inherent to all raytracing methods. Data exploration is computationally expensive because each viewpoint requires the isosurface to be resampled, and the computational cost is also associated with the resolution of the resultant image. Furthermore, the accuracy of the ray-isosurface intersections are related to the degree of the polynomial used to approximate the implicit function along the ray in world space. This relationship scales with computation time as p^2 to p^3, where p is the degree of the polynomial.

Conversely, the particle system allows users the freedom to explore the data by interactively moving a distributed set of particles in space. The accuracy of the particle positions with respect

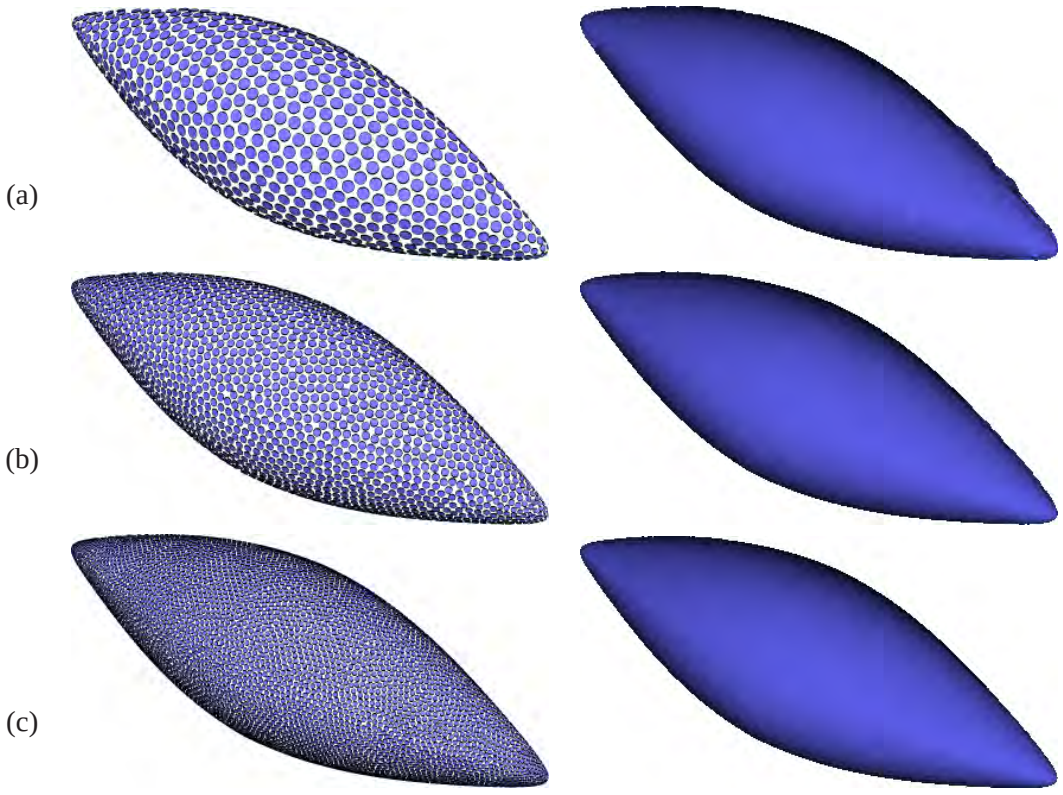


Figure 4.8. The data set from Figure 4.3 sampled with a particle system. The *right* column images are splat renderings of the particles in the *left* column: (a) 1280 particles distributed in 16 seconds with 1.1 million forward evaluations; (b) 3232 particles distributed in 36 seconds with 2.5 million forward evaluations; (c) 8647 particles distributed in 70 seconds with 4.8 million forward evaluations.

to the isosurface are controlled on a per particle basis by the error threshold, ϵ_T, and is independent of the sampling density of the particle system, unlike the raytracing method. Thus, very coarse, fast distributions of particles will be guaranteed to lie within ϵ_T of the isosurface. The accuracy of the visualization produced by the particle system is instead related to the *inferred topology* of the isosurface by the viewer. Course level distributions provide insight to the gross geometry of the isosurface, while finer distributions provide increasingly more detailed representations of the underlying geometry. Accuracy thus relates to the amount of detail that can be visualized with a specific resolution of the particle system – a relationship that scales linearly with time.

Figures 4.9(a) and 4.9(b) illustrate the results of two different resolutions of the particle system with 500 and 6800 particles, respectively. In Figure 4.9(c), the particle distribution from Figure 4.9(b) has been rendered with a GPU-based splat algorithm, and is virtually indistinguishable from a raytraced image at 512×512 resolution. Moreover, the isosurface in Figure 4.9(c) can

be rotated at interactive frame rates (greater than 30 frames per second). The GPU-based splat algorithm runs on an NVIDIA GeForce 6800 GT card with Pixel Shader 3.0. The isosurfaces in Figure 4.9 reside within a single hexahedral element, and are the zeroset of an eighth-order polynomial implicit function. The 500 particles in Figure 4.9(a) took 4 seconds to converge, and the 6800 particles in Figure 4.9(b) took 3 minutes to converge. The 512×512 raytraced image with an 25^{th}-order reconstruction polynomial required 6 minutes to render. We note that the particle system implementation uses the same finite element evaluation code as the raytracer, which is also the implementation used for results in [103]. We have found that sampling the finite element implicit function takes, on average, an order of magnitude longer than computing the inter-particle forces. This is consistent with our earlier observation, that basis-element evaluations dominate the computation.

We demonstrate the capabilities of our proposed particle system by visualizing pressure field isosurfaces of two CFD simulations. In the first example, shown in Figures 4.2, 4.10, and 4.11, we examine the wake of a rotating canister traveling through an incompressible fluid. The finite element mesh consists of 5040 hexahedra and 696 prisms, with the computational fluid mechanics problem being solved with third-order polynomials per element. In the second example, shown in Figure 4.12, we visualize the flow past a block with an array of splitter plates placed downstream of the block. This example contains 3360 hexahedra and 7644 prisms, again with the computational fluid mechanics problem being solved with third-order polynomials per element. The color of the disks in Figures 4.2, 4.10, and 4.12 indicates the relative direction of the surface normal at the particle (blue indicates *outward* and red indicates *inward*). In Figure 4.11, color specifies the size of the particles.

4.7 Discussion

In this chapter we have presented a general and robust method for visualizing isosurfaces of high-order finite element data sets that would allow scientists and engineers to efficiently explore simulation data. By sampling isosurfaces with a particle system, the method produces compact and adaptive visualizations that can be viewed at a variety of resolutions. Furthermore, the proposed system is general and easily adaptable to a broad range of finite element representations, from low-order linear elements, to complex, spline based elements. The curvature derivation presented in Section 4.3.1 is also relevant for any application that measures curvature in the presence of curvilinear coordinate transformations.

As mentioned in Section 4.3.1, the discontinuities in the derivatives at the element boundaries

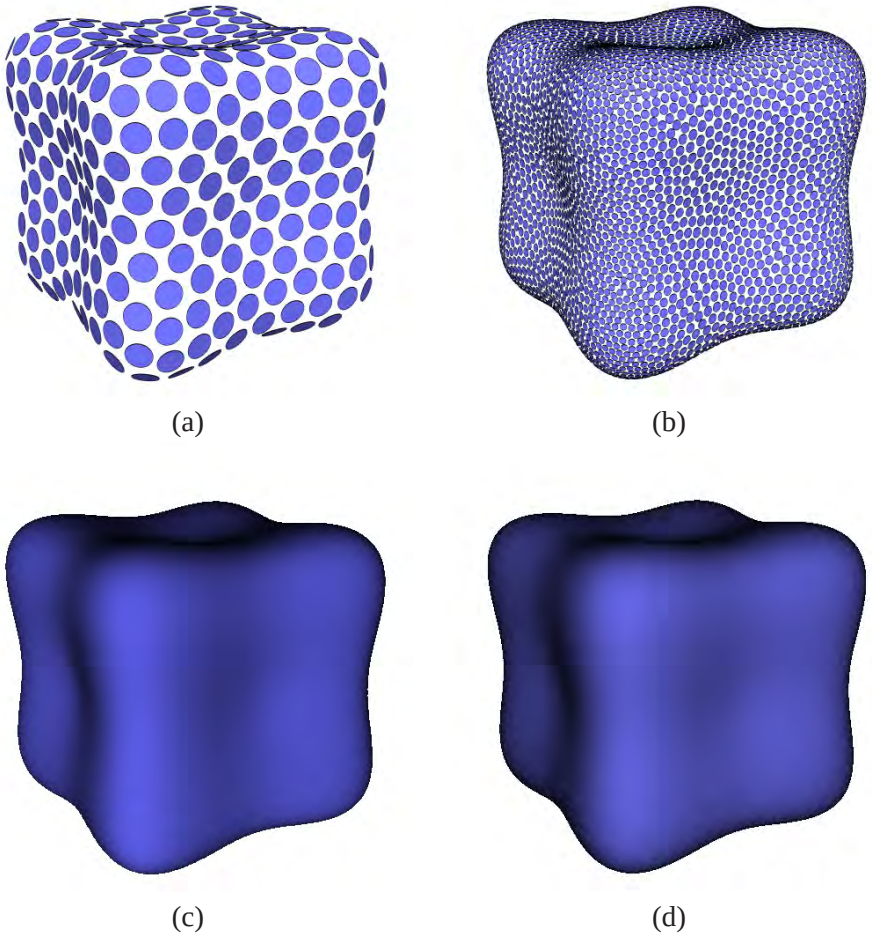


Figure 4.9. The zeroset of an eighth-order implicit function defined within a single hexahedral element: (a) 500 particles, with a distribution time of 4 seconds; (b) 6800 particles, with a distribution time of 3 minutes; (c) A GPU-based splat rendering of the 6800 particles that is visually indistinguishable from the 512×512 raytraced image in (d) that required 6 minutes to render.

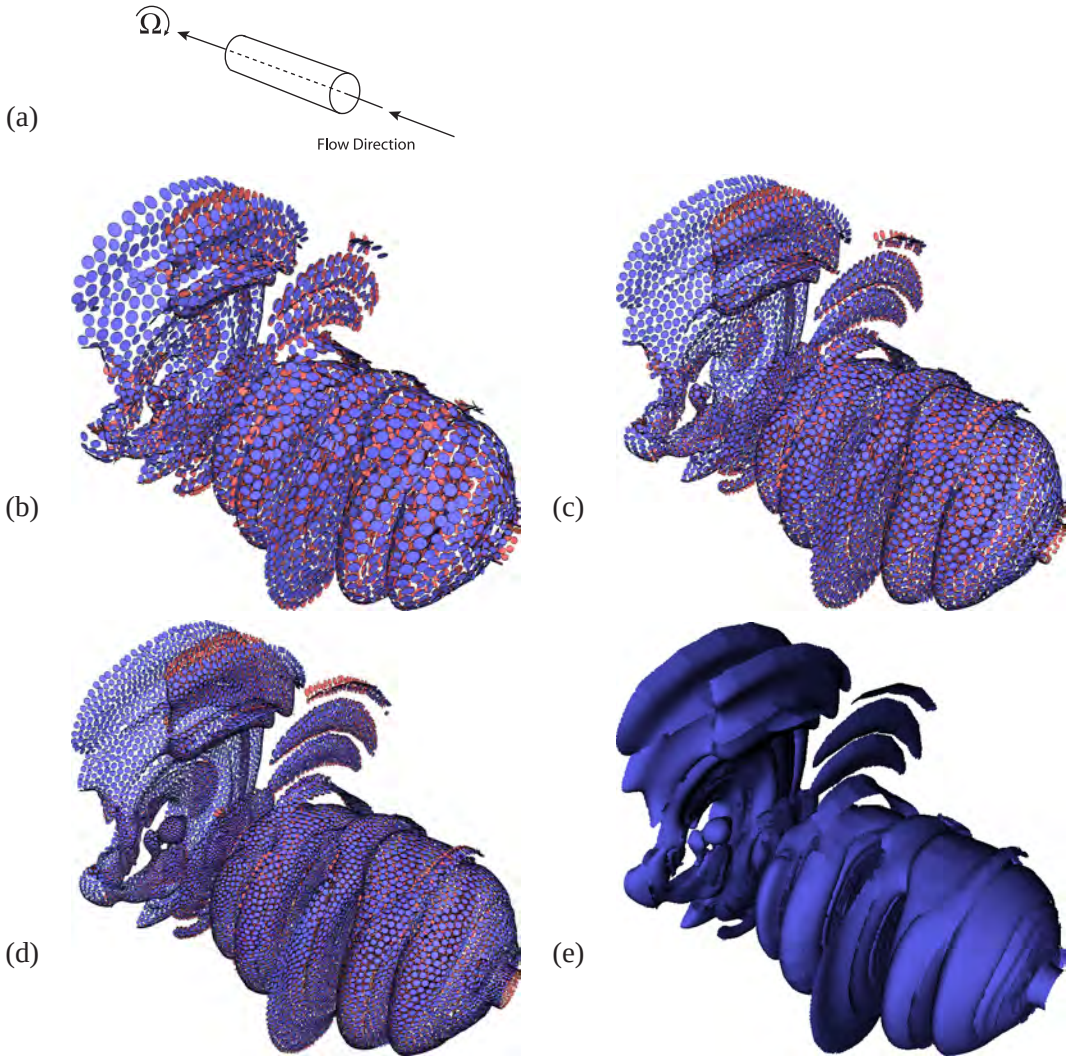


Figure 4.10. The isosurface of pressure $C = 0$ for a CFD simulation over 5736 elements with a third-order polynomial implicit function in each element: (a) Schematic for the fluid simulation; (b) 5000 particles, 55 seconds; (c) 13,000 particles, 3.4 minutes; (d) 28,000 particles, 15 minutes; (e) 59,000 particles, 39 minutes.

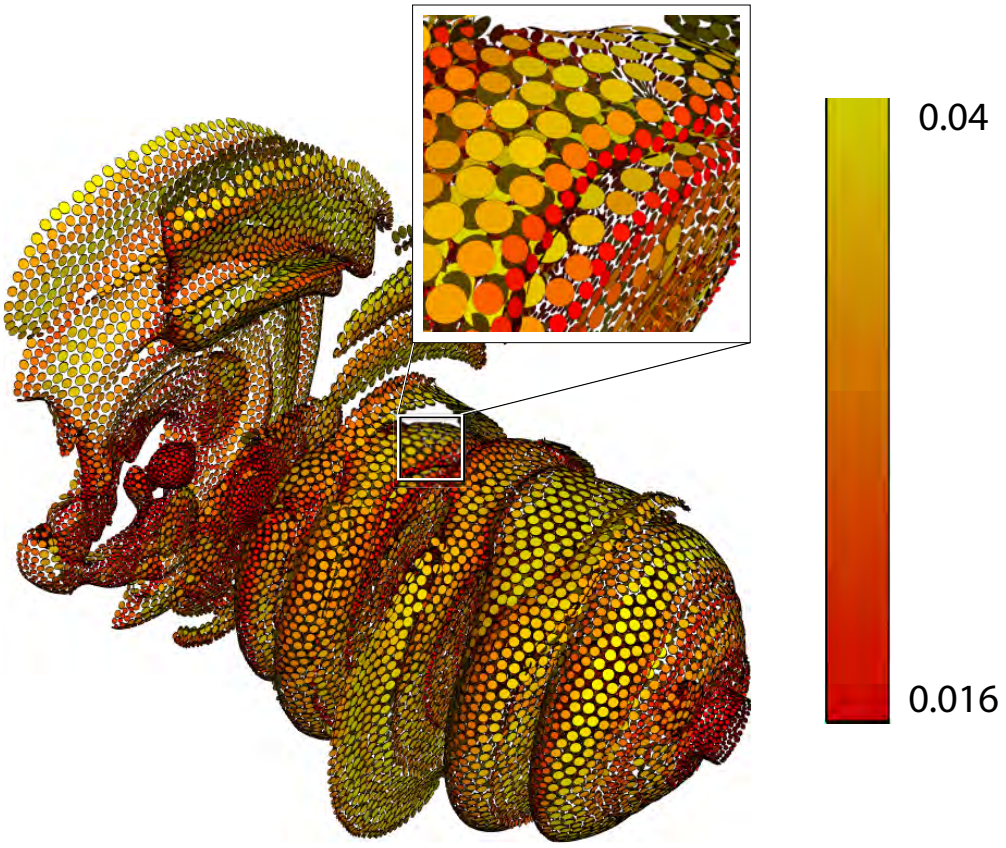


Figure 4.11. The distribution from Figure 4.10(d), color-mapped based on the radii of the particles. The colorbar provides the range of values of the radii.

can cause features in the surface that cannot be analytically detected through computation of surface curvature. These undetectable features will contain a very sparse sampling of particles, which create artifacts when the particles are splat. One solution would be to implement a more sophisticated splatting algorithm that clips splats along these boundaries [172]. Also, the particle-based visualization system could be bundled with an image-space rendering technique, such as raytracing, and serve as a preview for quickly determining isovalues and viewpoints of interest.

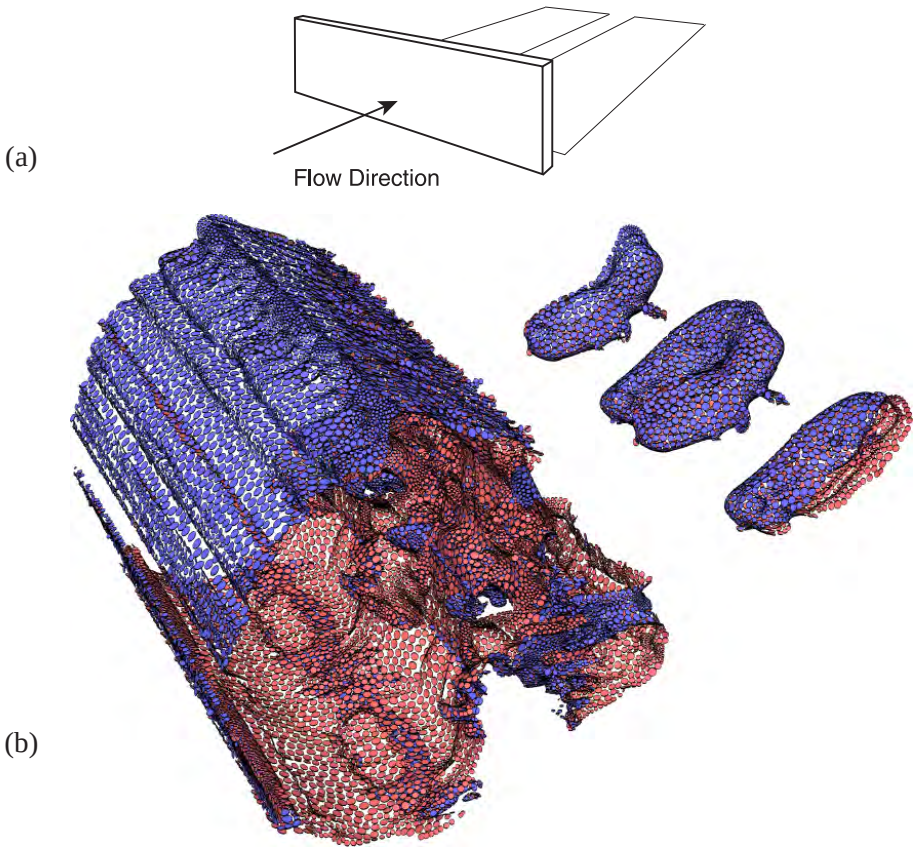


Figure 4.12. The isosurface of pressure $C = -0.1$ for a CFD simulation over 11,004 elements with a third-order polynomial implicit function in each element: (a) Schematic for the fluid simulation; (b) 43,000 particles, 25 minutes.

CHAPTER 5

ISOSURFACE MESHES USING DYNAMIC PARTICLES WITH QUALITY CONSTRAINTS

Biomedical simulations increasing rely on patient-specific models for accuratly capturing the pathology and physiology of individual patients. The generation of these models depends on a pipeline for extrating high-quality parameterizations of implicit surfaces reconstructed from MRI or computed tomography (CT) data. In this chapter we describe a method that addresses a piece of this pipeline — specifically, the construction of isosurface triangulations of implicit surfaces. The resulting meshes consist of triangles that are well-suited for accurate interpolation of scalar and vector-valued quantities, as required for numerous applications in visualization and numerical simulation. The proposed method does not rely on a local construction or adjustment of triangles as is done, for instance, in advancing wavefront or adaptive refinement methods. Instead, a system of dynamic particles optimally samples an implicit function such that the particles' relative positions produce a topologically correct Delaunay triangulation. Thus, the proposed method relies on a *global* placement of triangle vertices. The main contributions of the chapter are the integration of dynamic particles systems with surface sampling theory and PDE-based methods for controlling the local variability of particle densities, as well as detailing a practical method that accommodates Delaunay sampling requirements to generate sparse sets of points for the generation of surface tessellations.

The chapter begins with an introduction of biomedical applications that use meshing technology, and is followed by a review of existing methods for parameterizing implicit surfaces. We then present a detailed description of the proposed particle-based meshing pipeline along with results and a discussion.

5.1 Meshing for Biomedical Simulations

The problem of surface meshing has been studied extensively in a wide array of applications and contexts by a range of disciplines, from visualization and graphics to computational geometry

and applied mathematics. Existing approaches for tackling the surface meshing problem can generally be distinguished by:

- how the original surface is *specified*;
- what the *representation* of the output mesh is; and
- what the metrics are for measuring the *quality* of the mesh.

This chapter deals with a particular application, which is **the generation of nearly-regular triangle meshes from medical or biological data either for visualizing interpolated quantities or for conducting numerical simulations.**

Generating polygonal reconstructions of biological data is important in a variety of contexts. For example, in bioelectric field problems there is evidence of better source localization when inverse-problem solution techniques employ geometry and material properties that conform to those of the patient or subject [151]. Similarly, research in cardiovascular fluid dynamics and image guided medical simulations rely increasingly on models created from images of real vasculature [144, 34]. Recent work proposes using image driven geometry for the analysis of biomolecular functional models [167]. Accurate and compelling visualizations of simulated quantities over surfaces, such as the visualization of flow on manifolds [152, 88], also require high-quality geometry. In the context of mesh generation, visualization and simulation are related in the way they demand not only accurate geometric approximations but also representations that provide for accurate interpolation of other physical quantities across the surface.

The dependence of biomedical applications on image data drives several aspects of this work. First, a surface is not represented in a parameterized form but rather as a volumetric constraint, such as a levelset in a binary or grayscale mask resulting from an image segmentation. Thus, we focus on isosurfaces. Second, images have finite resolution and cannot capture small, sharp features beyond the resolution of the imaging device, which is limited by intersample distance and the point-spread function of the measurement process. Thus, the proposed method trades geometric accuracy for mesh quality and topological consistency, resulting in numerically useful meshes.

The strategy described in this chapter combines work from several disparate fields, establishing a *global approach* to meshing isosurfaces of volume data. This approach allows for local decisions about point placements to effect the global distribution over time, and generates high-quality, closed surface meshes (*i.e.*, water-tight) that adapt triangle size to closely approx-

imate the isosurface geometry. In this work, high-quality is defined by targeted measures of the regularity of the triangles. Although the proposed method is computationally expensive, the production of consistently higher quality meshes than other approaches makes the method useful for applications that demand very regular triangulations. Starting with a smooth reconstruction of the volume data, the method computes the curvature and medial axis of an isosurface, which are then used to construct a Lipschitz continuous measure of *local feature size*, a fundamental geometric quantity that governs the minimal sampling rate of a surface. A dynamic particle system then positions a large number of samples with interpoint distances that respect this local feature size. The resulting point samples are triangulated using the Delaunay-based meshing algorithm of Dey and Goswami [44], an algorithm that relies on sampling densities that exceed the minimal sampling rate. Thus, this chapter proposes a new meshing pipeline while also addressing the question of how, in practice, to make use of the fundamental work in surface sampling theory by detailing an algorithm for reliably achieving specific sampling densities.

5.2 Background

For many applications it is useful to generate a parametric representation of an implicit surface, such as a piecewise-linear polygonal mesh. In visualization, a triangular mesh can be rendered very efficiently using modern graphics hardware, while in simulations, boundaries represented in implicit functions can be linearly subdivided into a finite element mesh. Early work on isosurface meshing in the computer graphics literature focuses on efficiently generating approximate meshes, used mostly for visualization. The well-known *marching cubes* algorithm [165, 90] provides a well-defined set of rules for reliably and quickly producing first-order approximations, but does not construct tessellations that are adaptive to the isosurface geometry. Furthermore, quality measures of triangle regularity can be arbitrarily poor while vertex valences vary greatly, independent of the input surface geometry. Improvements to marching cubes are numerous, including improvements for better geometric approximations and closed meshes [106].

One general class of strategies for achieving higher quality surface polygonalizations is to start with a mesh that is either coarse or of low quality and, through some combination of mesh refinement, edge-swapping, or vertex repositioning, incrementally improve the mesh geometry and triangle quality. For example, Velho [153] proposes a curvature-based refinement method for improving the geometric accuracy of a marching cubes mesh, but does not fully address the issue of triangle quality. Wood *et al.* [163] propose another strategy that first constructs a coarse, topologically correct mesh that is then smoothly refined, producing higher quality triangles.

Other researchers have proposed refinement algorithms without the need for a base mesh [31]. In general, however, the strategy of refining a mesh to improve triangle quality produces the inefficiency of a great many samples that are dictated by mesh quality rather than the geometry of the underlying surface (for convergence rates of mesh quality see [31]).

Another scheme for generating quality isosurface meshes is to start from one or more seed points and grow triangles in the form of an expanding, or *advancing*, front [83, 63]. The basic approach is quite fast and can produce high-quality triangles, especially when triangle size is adapted to local surface curvature [72, 129]. The core algorithm procedes in two phases. First, the front is grown by adding triangles along active edges, where the triangles are sized according to local curvature metrics. Triangles are added as long as they are not too close to other preexisting ones until the initial front and the active front are separated by a thin, empty region. The second phase of the algorithm stitches these fronts together using a variety of heuristics for detecting and connecting nearby triangles.

A key element of advancing front techniques is the detection of merging fronts. Spatial subdivision schemes can be used to find any nearby triangles [72], as can other methods such as *fences* [128], which test for bounding sphere intersections. Even with these tests fronts can still overshoot or miss existing triangles. Using more stringent triangle sizing schemes, such as a *guidence field* [128], can help to limit the size of triangles and avoid misdetections of nearby fronts. The detection and stitching schemes, however, are based on a variety of heuristics and special cases that have yet to be provably correct or implementable. Furthermore, the shapes and sizes of triangles along merging fronts is determined not only by surface geometry, but also by the geodesic curvature (curvature in the tangent plane) of the moving front. This problem becomes even more acute when the data contain wavefronts that collide from opposite directions, which is unavoidable for certain topologies or shapes. Thus, advancing front algorithms must have additional built-in heuristics, such as wavefront smoothing [131], or special triangulation schemes (*e.g.*, edge swapping) that deal with collapsing or colliding wavefronts, often at the expense of triangle quality in those areas.

A third approach to surface meshing is to generate an unorganized set of surface samples and use algorithms from the computational geometry literature to create a Delaunay tessellation of the points. Early work in the field provides the algorithmic foundations for producing solid Delaunay triangulations in 2D and 3D [126, 50, 32], complemented by literature on the theory and methods for extracting the surface manifold [6, 5, 8, 43]. These methods employ a compelling bottom-up approach for constructing edges and faces from nonlocal properties of a point set, guaranteeing

closed, nonintersecting meshes. Generating the set of surface samples, which determine the topology of the resulting tessellation *and* the quality of the resulting triangles, is difficult and generally treated as either a separate problem, or as part of an adaptive scheme for iteratively improving mesh quality [31, 45].

This chapter proposes a surface sampling and triangulation algorithm that relies on fundamental sampling requirements associated with Delaunay surface reconstruction schemes. Amenta *et al.* provide the quantitative requirements, based on surface geometry, for 2D curves [6] and 3D surfaces [5], such that a unique Delaunay tessellation exists from which a subset of edges or faces have a topological equivalence to the underlying surface. After distributing a set of points based on these sampling requirements, we use methods that generate water-tight Delaunay reconstructions from such samples [44, 9] to create a tessellation.

The core of the Delaunay sampling requirements relies on a characterization of surface geometry that depends on nonlocal information. Given a smooth surface $F \subset \Re^3$, a sufficiently dense sampling P is one such that for any point $s \in F$ the Euclidean distance between s and the closest sample point $p \in P$ is no greater than ϵ times the *local feature size* at s. Any discrete set of surface points P that meets this requirement is an ϵ-sample of F. The current theoretical (3D) results show $\epsilon = 0.06$ is sufficient [5]. However, empirical results indicate that the actual bound might be looser, and several authors have conjectured that $\epsilon = 0.5$ may be closer to the necessary bound [7].

The definition of local feature size (LFS) is an important aspect of these results and of the proposed algorithm. The LFS of a point $s \in F$ is defined as the distance from s to the nearest point on the medial axis (MA) of F, shown in Figure 5.1 as the distance d to the point on the surface s. The MA has been heavily studied in the literature [80, 70] in the context of shape modeling, computational geometry, and computer vision. Although the MA has several formal definitions and many interesting and important characteristics, it is sufficient for the proposed method to consider the MA of a surface as (the closure of) the set of points $M \subset \Re^3$ such that the nearest point on the surface to $m \in M$ (*i.e.*, $\min_{s \in F} |s - m|$) is not unique. Alternatively, several authors define the MA as the set of points where there exists a sphere that does not intersect the surface and is tangent to the surface in more than one location.

The relationships between the MA, local surface geometry, and sampling requirements are important in several ways. First, the cotangency definition implies that the LFS of $s \in F$ is *no greater than* the local radius of curvature at that point. The radius of curvature is then an upper bound, as is the sampling condition, and is therefore not a suitable proxy for the LFS. For instance,

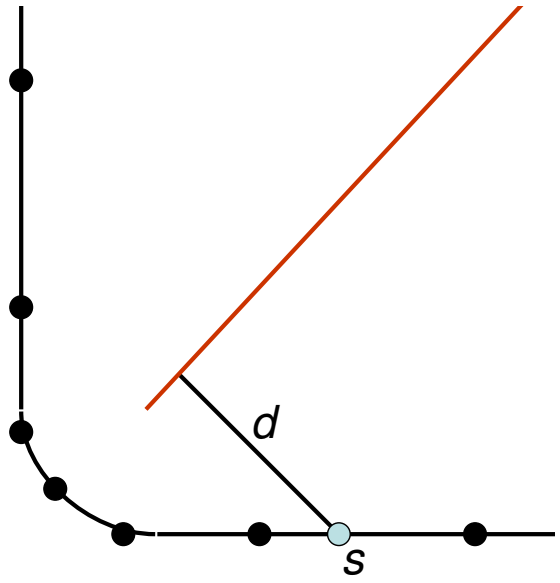


Figure 5.1. A curve (shown in black), with an ϵ-sampling of points (also shown in black), and its MA is (shown in red). The ϵ-sampling requirements state that a point on the surface, s, cannot be further away from a sample point than ϵ times the LFS at s.

the feature size can be very small on thinly separated, flat objects that have a very large radius of curvature. Second, the LFS condition is necessary for establishing the correct topology among an unorganized set of points. If the topology is somehow known *a priori* (*e.g.*, via continuity in an advancing front) the sampling density could be much more sparse. However, one common use of a surface mesh is for the construction of a body fitting tetrahedralization [4] where it is important that the triangles, which form the faces of the corresponding tetrahedra, conform to the global solid geometry. If, for example, a thin region of a surface were sampled with a sparse set of points, the corresponding tetrahedra would be very flat. However, if the point density is related to the LFS, the tetrahedra are more likely to be regularly shaped.

Algorithms for constructing an ϵ-sampling of a surface F are not immediately evident from the sampling theorems or mesh generation algorithms. Although several related schemes propose methods for sampling surfaces with less-strict or slightly different bounds [19, 31], the ϵ-sampling requirement provides guarantees necessary for subsequent simulations due to the side effect of respecting local as well as global object shape. Among the contributions of this chapter is a practical scheme for generating sets of surface points that closely conform to the ϵ-sampling requirements.

5.3 Mesh Generation with Particle Systems

The goal of the proposed meshing system is to generate nearly-regular triangular meshes of isosurfaces. By adaptively distributing a set of dynamic particles such that their positions conform to an ϵ-sampling requirement, the particles can be used to generate a Delaunay surface mesh that corresponds to the geometry and topology of the isosurface. To achieve this goal, the proposed method consists of several steps: (1) computation of a MA approximation to determine the local feature size; (2) creation of a sizing field to specify the desired distances between particles; (3) adaptive distribution of particles across the isosurface; and (4) triangulation of particle positions to create a polygonal reconstruction of the isosurface. Figure 5.2 depicts this pipeline. The following subsections will describe each of these steps in more detail.

5.3.1 Local Feature Size

As described in Section 5.2, the LFS is the distance to the MA of a surface. Accurate computation of the MA is a challenging research problem, and numerous approaches have been proposed for its solution. One approach presented in the computer vision literature is to detect discontinuities (*i.e.,* shocks) in a distance transform of a surface. Detecting the shocks, and hence the MA approximation, in the distance transform is numerically tricky due to the discontinuities in the derivatives at these points. Siddiqi *et al.* [137] propose measuring the divergence of the gradient of the distance transform, where high values indicate a significant change in the field, and thus a MA point. Another approach by Persson [113] fits local quadratics over the distance transformation grid, looking for places where these functions intersect. A set of heuristics then determine whether an intersection point should be included in the MA approximation.

For this work we have developed a medial axis detection algorithm for more general implicit surfaces which is moderately robust to free parameters, gives subgrid accuracy, and does not require the thinning or postprocessing of similar methods [137]. This scheme relies on the *foot point* method, which is the nearest point p on a surface to a given point q, and can be found using gradient descent by the method described in [64]. Here we consider only the MA proper, and not the singular points where the MA terminates. The line segment defined $\overline{pq}$ is perpendicular to the surface, and every point on $\overline{pq}$ has p as its foot point. This line segment forms a *characteristic*, which is the path of a surface point that moves inward/outward in the direction of the surface normal (to within a sign difference). As we proceed from the surface along an inward or outward characteristic, the foot point of each point along that path remains the starting point for the characteristic *until* the characteristic intersects the MA — once the characteristic intersects the

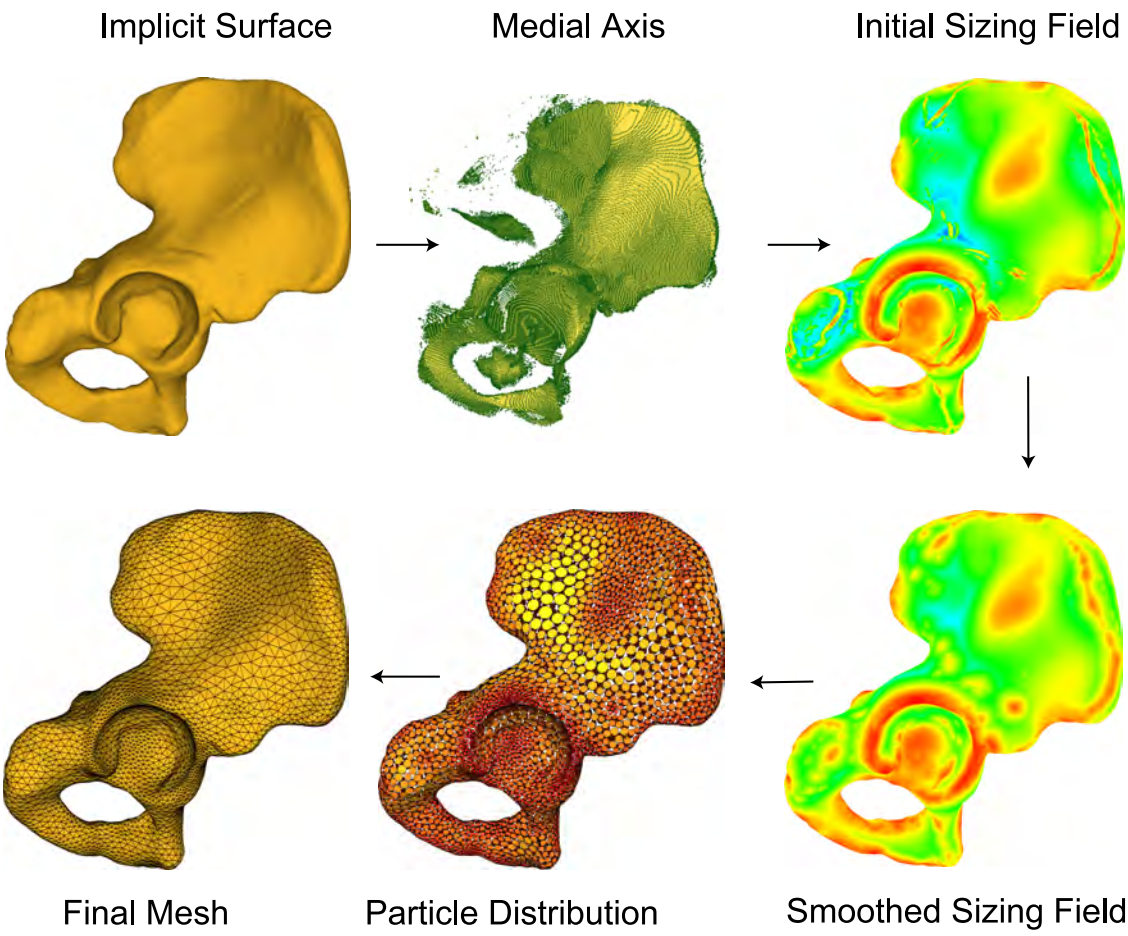


Figure 5.2. The proposed mesh generation pipeline using a dynamic particle system. First, a medial axis is computed from a distance transform of an implicit surface; next, an initial sizing field is built from the local feature size and radius of curvature; a smoothed sizing field is then generated by limiting the gradient of the initial sizing field; particles sample the sizing field and distribute themselves accordingly; and finally, the particles are triangulated using a Delaunay surface reconstruction algorithm.

MA, the position of the foot point changes. The algorithm for detecting the MA is as follows. For each point on the grid q find the foot point p, then find the point q_* along the characteristic (away from the foot point) that intersects the current voxel (far face of the cube). Find the footpoint p_* associated with q_*. If the angle between the line segments $\overline{pq}$ and $\overline{p_*q_*}$ is greater than some small threshold a, then the segment $\overline{pp_*}$ crosses the MA. The position of the MA along $\overline{pp_*}$ can be found by using a first-order (tangent plane) approximation to the surface at the point p_*. We use $\cos(a) = 0.9$ for all of the results in this paper.

After constructing M, a LFS field $\lambda(\mathbf{x})$ is created by finding the distance to the closest medial axis point $m \in M$ at the grid nodes. For efficiency, we restrict this field to the subset of grid nodes

that bound the isosurface. The λ field needs only to be a conservative estimate of the distance to the true MA as LFS (and curvature, which will be discussed in the next section) provides an upper bound on the distance between particles in the proposed system; anything less than, or equal to, the true distance will drive the final distribution of particles to be an ϵ-sampling.

The accuracy of the λ field fundamentally relies on the underlying accuracy of the medial axis detection algorithm, which is itself a sampling problem. Creating the medial axis requires a sampling of the data field, where those samples are chosen dictating the accuracy of the medial axis detection. The problem thus cycles between choosing samples to detect the medial axis, and using the detected medial axis to determine where the samples should be to accurately capture the true medial axis. For this work, we intentionally break the cycle by relegating the accuracy of the system for determining the *lfs*, and ultimately achieving a true ϵ-sampling of the surface, to that of the accuracy of the underlying medial axis detection algorithm. As research into medial axis detection progresses, the proposed sampling method can make use of new algorithms to achieve more accurate results.

5.3.2 Sizing Field

The sizing field is the mechanism by which the particle system adapts its distribution to meet an ϵ-sampling requirement. There are two geometric quantities, LFS and radius of curvature, and two parameters, ϵ and δ, that govern the construction of the field.

To establish an initial sizing field $h_0(\mathbf{x})$, the λ field is compared to the radius of curvature at each grid node in a narrow band around the isosurface. The radius of curvature is calculated as $1/|k_{max}|$ — the absolute value of the maximum curvature can be computed directly from the Hessian of F [78]. The initial sizing field is given as:

$$h_0(\mathbf{x}) = C \min(\lambda(\mathbf{x}), 1/|k_{max}(\mathbf{x})|) \tag{5.1}$$

where C is a constant based on ϵ that is discussed and defined in the following paragraphs. Including the radius of curvature in Equation 5.1 helps to ensure that small surface features that may not have been captured in M appear in the sizing field construction.

Although h_0 contains most of the core geometric information about a surface necessary for describing an ϵ-sample, it is not suitable on its own for regulating particle distances for two reason: first, the Delaunay sampling theory indicates that some fraction of the LFS is required for topologically correct reconstructions; and second, to achieve high quality triangles across the

entire mesh, the gradient of the sizing field must be limited to ensure smooth, gradual changes in triangle size. These two characteristics are controlled by the user defined parameters ϵ and δ, which modify h_0 to create an (ϵ, δ)-sizing field $h(\mathbf{x})$.

Multiplying h_0 by twice the ϵ parameter, such that $C = 2\epsilon$, causes the sampling to be a fraction of local feature size, *i.e.*, an ϵ-sampling. The Delaunay sampling requirement [6] specifies ϵ for some point on the surface *other* than the sample points, thus we include this implied factor of two in C, and the literature indicates that $\epsilon = 0.5$ may be a loose upper bound for ϵ. The second parameter, δ, is used to limit the gradient of h_0 such that the values in the resulting field h will not change faster than δ. Thus, δ dictates how quickly the edge lengths of neighboring triangles can change — $\delta < \epsilon$ will generally produce well-shaped triangles. This limiting produces a δ-Lipschitz field (described in Section 2.3), an important property for smooth triangle gradation. To limit the rate of change of h over its grid we use the following discrete operator, operating on a lattice sampling of our field $h_{ijk} = h(\mathbf{x})$ at a grid node position $\mathbf{x}$, which is shown by Persson [113] to generate a δ-Lipschitz field:

$$h_{ijk}^{n+1} = h_{ijk}^{n} + \Delta t(\min(\Delta_{ijk}^{+}, \delta) - \Delta_{ijk}^{+}) \tag{5.2}$$

where

$$\begin{aligned} \Delta_{ijk}^{+} \quad = [\quad & \max(D_x^- h_{ijk}^n, 0)^2 + \min(D_x^+ h_{ijk}^n, 0)^2 + \\ & \max(D_y^- h_{ijk}^n, 0)^2 + \min(D_y^+ h_{ijk}^n, 0)^2 + \\ & \max(D_z^- h_{ijk}^n, 0)^2 + \min(D_z^+ h_{ijk}^n, 0)^2 \ ]^{1/2} \end{aligned} \tag{5.3}$$

and where D^+ and D^- are the forward and backward difference operators, respectively, with subscripts denoting the axes along which they are operating. In our implementation of the system we consider the limiting of the gradient to have converged when the maximum relative change of any grid node is less than 10^{-5}.

While the LFS function λ is $1-$Lipschitz [126], the inclusion of the radius of curvature causes h_0 to lose this property. We have found, however, that the initial sizing field h_0 is nearly $1-$Lipschitz, producing a final sizing field h which is $\min(2\epsilon, \delta)$-Lipschitz. Notice that if $\delta \geq 2\epsilon$, the gradient limiting smoothing will have no effect on h_0. We have experimented with a range of values for both ϵ and δ, and present an illustrative example in Figure 5.3 to provide intuition

on each parameter's role in the final mesh quality. This example visually emphasizes the balance between geometric accuracy, the number of triangles, and triangle quality. The rounded box is a level-set of a sampled analytic distance transform of box, where the faces are planar, the edges are cylindrical, and the corners are spherical. The surface is reconstructed using a Catmull-Rom spline kernel and the first and second derivative of a cubic B-spline. We also note that this sizing field could be incorporated in the advancing front algorithm to determine triangle size by replacing the *guidence field* [128] with the sizing field.

5.3.3 Distributing Particles

Using the particle system framework described in Chapter 3, a set of dynamic particles can be controlled by h such that their final distribution meets the sampling requirements for F. To do this, the system is initialized with a set of particles, the positions of which are determined from a marching cubes triangulation to ensure that disconnected parts of the isosurface are seeded with particles. The particles are then projected onto F using a Newton-Raphson gradient descent method. Once on the surface, each particle is associated with an individual potential function which induces interparticle forces that push them towards lower, local energy states (see Chapter 3). To control the sampling density, we scale the distances between particles — which determines the magnitude of the interparticle forces — by the value of h at each particle's position. The distance between particles p_i and p_j becomes:

$$d_{ij} = \alpha_{ij}|(\mathbf{x}_i - \mathbf{x}_j)| = d_{ji} \tag{5.4}$$

where α_{ij} is defined by h (evaluated at particle positions $\mathbf{x}_i$ and $\mathbf{x}_j$ using linear reconstruction kernels) as follows:

$$\alpha_{ij} = \alpha_{ji} = \frac{\beta}{\min(h_i, h_j)} \tag{5.5}$$

with $\beta = 0.5/\cos(\frac{\pi}{6})$. The particles are then iteratively moved to lower local energy states until the system reaches an equilibrium.

Equation 5.5 scales the effective distance between particles based on an ideal hexagonal packing across a flat surface where the region of influence of a particle ends at the closest two-ring neighbors. The value of β is derived from this ideal packing, and allows for a population control mechanism to be defined that adds and removes particles based on their energy, driving the system

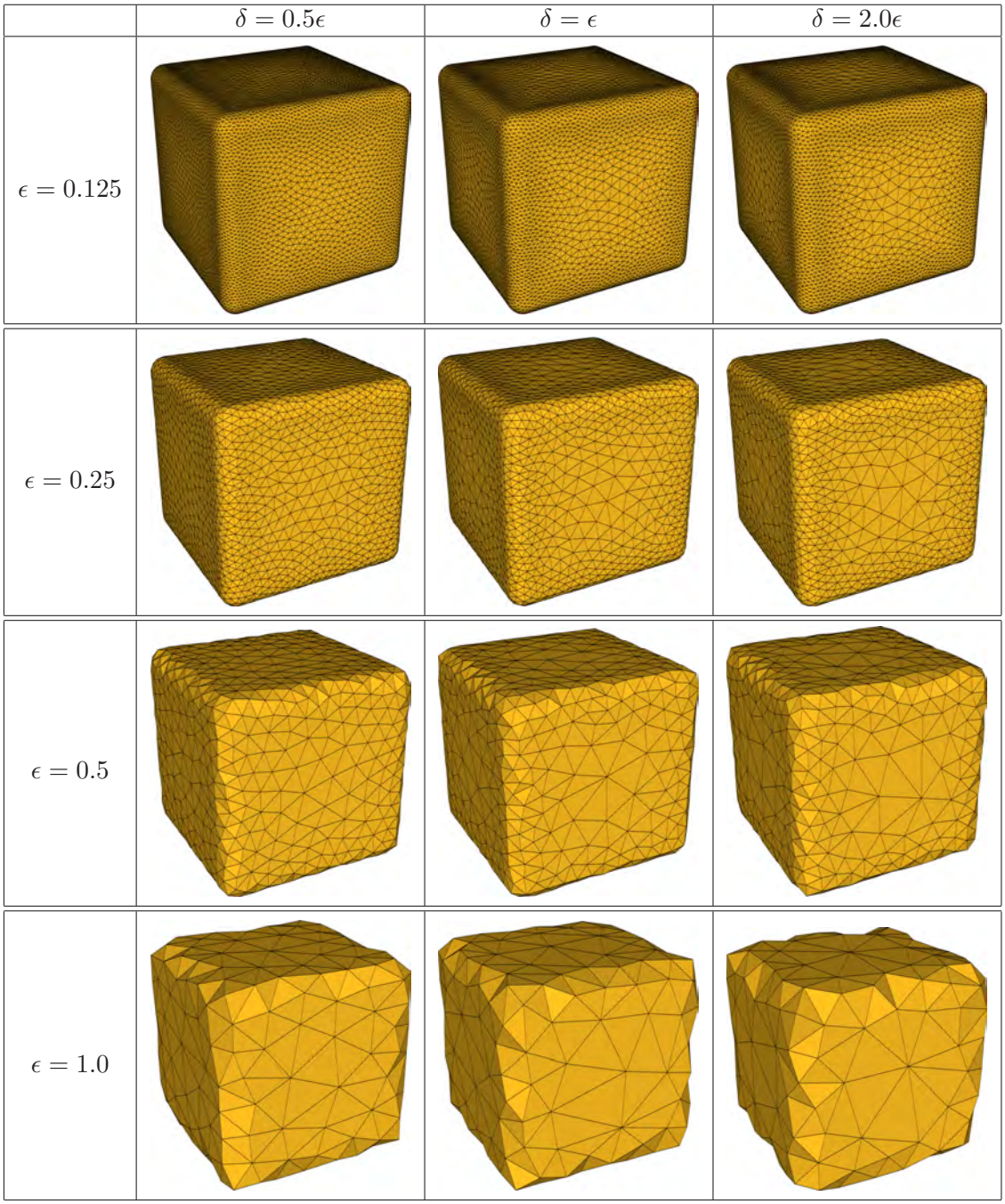


Figure 5.3. Illustrative comparison of mesh quality and number of triangles for varying values of ϵ and δ, the user defined parameters in Equations 5.1 and 5.2, respectively. The ϵ values vary down the columns while the δ values vary across the rows.

towards an ideal packing. Because h_0 establishes an upper bound on the allowed distance between particles for meeting an ϵ-sampling requirement, using the minimum sizing field value of any pair of particles establishes a conservative sampling, as do the convex, linear interpolation kernels that reconstruct h at arbitrary particle locations.

There is, however, a caveat to producing an ϵ-particle sampling. The particle system uses Euclidean distances to compute interparticle forces and energies, as opposed to the more accurate (and computationally expensive) geodesic distance. The distance approximation causes particles in a two-ring neighborhood to become influential, adversely effecting the population-control mechanism for obtaining ideal neighborhood packings. This artifact, however, is bounded by ϵ, which allows for a contraction factor g to be introduced to the system to counter-act the effects. Figure 5.4 shows the distance contraction for particles equally spaced across a circle. The distance between each adjacent particle is d — the distance between p_0 and p_2, however, is not $2d$ as the ideal packing model assumes. Instead, it is:

$$D = 2d\left(1 - \frac{(dk_{max})^2}{4}\right)^{1/2}. \tag{5.6}$$

In the proposed system, $d \leq \epsilon/k_{max}$ because the LFS is bounded from above by the radius of curvature. Thus, we can bound the contraction parameter, g, as:

$$1 \geq g \geq \left(1 - \frac{\epsilon^2}{4}\right)^{1/2}. \tag{5.7}$$

As ϵ goes to zero, g approaches one because the surface becomes locally more and more planar as the distance between particles shrinks. These bounds on g attest that the contraction effect cannot get arbitrarily worse as the surface features become smaller; the worst case is bounded by ϵ.

To counteract the contraction of distance to neighboring particles, we inversely scale the sizing field as $h \leftarrow h/g$. We empirically determined a value of g by observing histograms of the ratio of triangle edge lengths to the average of h at each edge's vertices. Over a range of ϵ and δ values for a variety of data sets the shapes of the histograms were visually identical except for the tails, which contain a relatively small number of outliers — an example of one histogram is shown in Figure 5.4. Based on our observations, we determined a conservative estimate of g to be 1.5. The inclusion of g creates a final mesh where the length of virtually every edge is at most $2\epsilon h_0$. These results are detailed in Section 5.5.

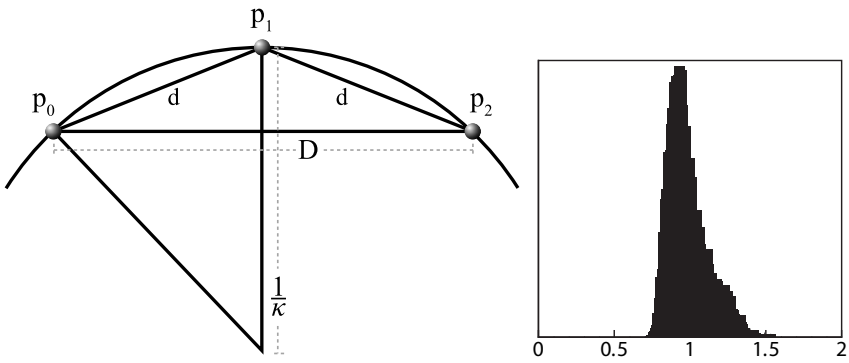


Figure 5.4. The effects of curvature cause the two-ring neighboring particles to become closer than $2d$ (*left*). This effect is bounded by ϵ, which allows for a scaling parameter to be introduced into the system. We empirically determined this value by studying histograms of the triangle edge lengths versus h, such as that of the pelvis reconstruction (*right*).

5.3.4 Triangulation

To triangulate a distribution of particles, we use the water-tight Delaunay triangulation method TIGHT COCONE [44] – a free version of the software is available from the author's website. TIGHT COCONE first builds a Delaunay tetrahedralization of the union of the set of input surface points and the Voronoi vertices of those points. The faces of the tetrahedralization are then culled such that the remaining triangles exist only in a thickened region around the surface samples. To avoid the creation of holes in the manifold extraction step from these candidate faces, the remaining tetrahedral elements are marked as *inside* or *outside*. The marking is done by first creating an adacent tetrahedral element to some candidate face with a vertex at infinity. This element is marked as outside, and a walking algorithm then determines, based on adajencies, which tetrahedral elements are inside or outside. The labeling of the elements is then considered in the final manifold extraction step that extracts a surface triangluation out of the set of candidate triangles such that the resulting mesh is gauranteed to be water-tight.

The TIGHT COCONE algorithm works well for single materials when the sampling criteria is met. There are applications of particle-based meshing, however, such as to multimaterial datasets for which the sampling criterial cannot be explicitly met. In Chapter 6 we present a simple labeling algorithm that takes advantage of the volumetric data for constructing watertight surface meshes — this algorithm could be applied to any of the sets of particles in this chapter for achieving reliable surface meshes.

5.4 Implementation

This particle system implementation is a slightly modified version of that found in Section 3.6, where changes are in **bold**, and free parameters introduced in this chapter are presented in Tables 6.6.4 and 6.6.4, where we also present the free parameters from Chapter 3 that were used in the meshing implementation. Note that almost all of these parameters use the same values as previously discussed implementations.

For all of the results presented in this chapter we initialized the system with the vertices of a marching cubes mesh. The s and ρ user-defined variables are replaced with a sizing field generated as a preprocessing step.

1. For each particle:

   (a) Compute E_i, and compute $\mathbf{v}_i$ with Equation 3.4, **where the interparticle distances are scaled by the α_{ij} given in Equation 5.5.**

   (b) Compute $\mathbf{v}_i^{\text{new}} = \lambda_i \mathbf{v}_i$.

   (c) Compute the new particle position $\mathbf{x}_i^{\text{new}}$ by solving Equation 3.11 with $\mathbf{v}_i^{\text{new}}$, followed by a reprojection to the surface by solving Equation 3.12.

   (d) Compute the new energy value, E_i^{new}, at the new particle location $\mathbf{x}_i^{\text{new}}$, as well as the new implicit function value, F_i^{new}.

   (e) If $E_i^{\text{new}} \geq E_i$ or $|\mathbf{x}_i - \mathbf{x}_i^{\text{new}}| > s$ or $F_i^{\text{new}} > \epsilon_T$, and $\lambda_i > \lambda_{\min}$, decrease λ_i by a factor of 10 and go back to Step 2(b)ii. If $\lambda_i \leq \lambda_{\min}$, do not move the particle and skip to the next particle in the list.

   (f) If $E_i^{\text{new}} < E_i$, update $\mathbf{x}_i = \mathbf{x}_i^{\text{new}}$. Increase λ_i by a factor of 10 if this is the first time through Step 2b.

2. Decide whether the system is at a steady state. There are numerous metrics to determine steady state, and we have chosen to use the difference of the system energy (the sum of the energy at all the particles) from one iteration to the next. When the system energy difference is less than a small fraction of the total energy (we use 0.15% for the results presented in this dissertation, although a range of values would produce similar results), we deduce that particles have reached a steady state. Otherwise, repeat Step 2b. For poor initializations, we have found it can be useful to skip to Step 2d every 50 iterations instead of waiting for the system to reach a steady state.

3. Check whether the configuration of particles is desirable. We compare each particle's energy against an ideal energy, E^{ideal}, which is defined by a hexagonal packing of neighbors on a flat surface with inter-particle distances of s. We bias E_i with a random value on the interval $[0, 1]$ to eliminate mass splitting or dying, then split particles with $E_i < 0.35E^{\text{ideal}}$, and delete particles with $E_i > 1.75E^{\text{ideal}}$. While we have provided specific values for this step, we have found in practice that varying the values by up to 20% produces visually similar results, although with different convergence times.

4. If the energy of the particles is acceptable, stop iterating.

5.5 Results

In this section we present results from the proposed system for generating isosurface meshes of a variety of biological data sets. The first two tessellations, shown in Figures 5.5 and 5.6, are of a pelvis [10] and brain [142] generated from binary segmentations that have been smoothed with a Gaussian kernel ($\sigma = 1.5$). Figure 5.7 illustrates a skull extracted from gray-scale CT data, and Figure 5.8 depicts vasculature represented by the zero-set of a distance transform generated by an anisotropic smoothing algorithm [104]. All four surfaces are reconstructed with approximating cubic B-spline kernels.

The proposed method was run on a P4 3.2GHz CPU with 2GB of memory. The generation of the medial axes and the λ grids took on average about an hour for each data set while limiting the gradient field of h_0 to generate h required several minutes. Run times to distribute the particle systems, along with the resulting mesh dimensions are presented in Table 5.5. We also present the ϵ and δ values used to generate the sets of particles and note that for all but one data set these two values were constant over the set of data. These values are slightly larger for the pelvis data set as the resolution of the surface was more coarse than the underlying data grid, allowing for a more spare set of particles to accurately reconstruct the surface. As mentioned in Section 5.3.4, we use TIGHT COCONE [44] to triangulate the particle distributions — the tessellations required on the order of several minutes.

The stated goal for this work is the generation of isosurface meshes suitable for simulations and the interpolation of data. To measure the quality of our results for these purposes we draw upon ideas established in the finite element literature which characterize the role of mesh quality in the simulation accuracy [67]. Here, we briefly discuss the interplay between geometric accuracy and the error of a finite element solution computed over a mesh.

Table 5.1. Table of free parameters.

Parameter	Value	Description	Comments
$\epsilon_{\mathbf{r}}$	10^{-7}	added to interparticle distances to avoid infinite energy values	system is insensitive to values near machine precision
σ	1.0	effective particle radius	constant for all results in this dissertation
γ	0.156	defines range of normals for which the particle energies are smoothed	insensitive to exact value as long as $\gamma \leq 1$
λ_0	1.0	initial stepsize value	system is insensitive to this value
λ_{min}	10^{-14}	minimum stepsize value	system is insensitive to this value as long as it is sufficiently small
ϵ_F	10^{-5}	surface threshold	value should be around the machine precision value
	5	number of initial projections of a particle	system is insensitive to this value as long as particles get to within ϵ_F of the surface
	0.15%	system energy difference from previous iteration that indicates a steady state	value must be small enough such that the particle distribution converges to an even packing
	50	number of iterations when system automatically checks for a desirable configuration (Step 2d)	value must be large enough such that local particle neighbors can be established
	0.35	percentage of E_{ideal} that indicates a particle should be split	values with approximately 20% of this value produce visually similiar results with different convergence times

Table 5.2. Table of free parameters, *cont*.

Parameter	Value	Description	Comments
	1.75	percentage of E_{ideal} that indicates a particle should be deleted	values with approximately 20% of this value produce visually similiar results with different convergence times
	5	number of iterations when particle neighbor lists are updated	values < 10 maintain stability in the system
ϵ	user-defined	fraction of LFS stored in the sizing field	values ≤ 0.5 are important for the reliability of accurate triangulations
δ	user-defined	Lipschitz value of sizing field	values must be $\leq \epsilon$
	10^{-5}	convergence threshold for gradient limiting	value should be near machine precision
	1.1	TIGHT COCONE parameter	values in the range [0.9, 1.2] produce reasonable tessellation results

Table 5.3. Details of each data set, including size of the volume, values for ϵ and δ, minutes required to distribute the particle system, and resulting number of mesh vertices and triangles.

Data	Volume Size	ϵ,δ	Time (mins)	Vertices	Triangles
brain	149x188x148	0.5, 0.3	41	91702	182188
pelvis	271x390x282	0.75, 0.5	1	4992	9984
skull	256x256x115	0.5, 0.3	232	212188	424588
vessel	265x265x171	0.5, 0.3	280	287990	576493
dendrite	270x586x154	0.5, 0.3	225	203744	406994

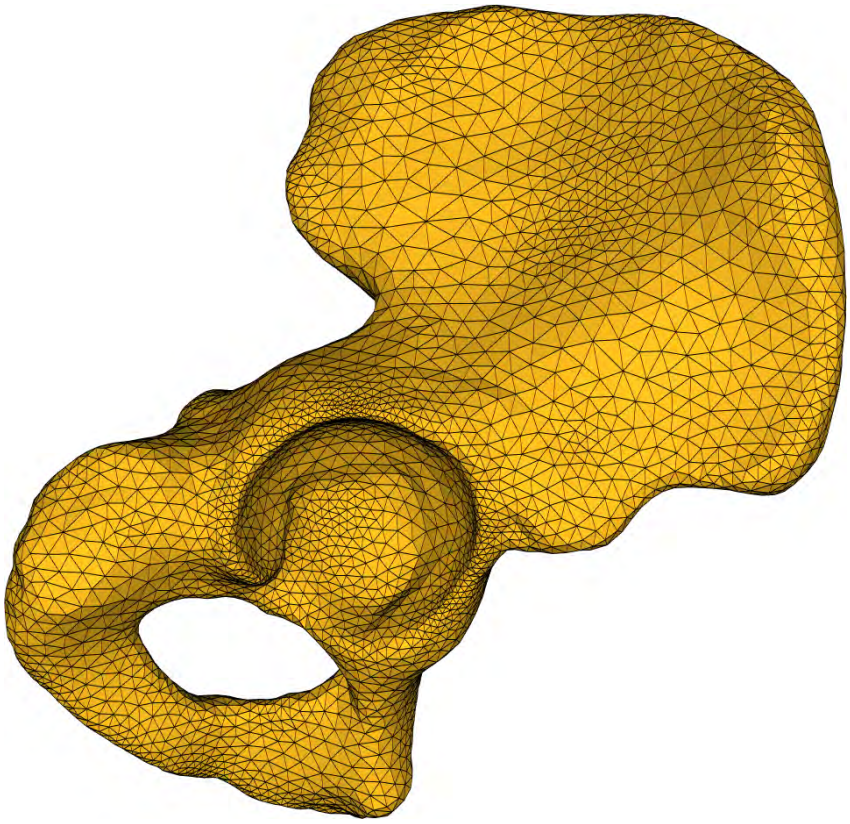


Figure 5.5. A tessellation of a pelvis segmentation [10].

Given a domain Ω and a partial differential equation (PDE) that operates on a solution u that lives over Ω, the standard finite element method attempts to construct a geometric approximation $\tilde{\Omega} = \mathcal{T}(\otimes)$ consisting of a tessellation of polygonal shapes (*e.g.* triangles and quadrilaterals for 2D surfaces) of the domain Ω, and to build an approximating function space $\tilde{\mathcal{V}}$ consisting of piece-wise linear functions based upon the tessellation [67]. Building on these two things, the goal of a finite element analysis is to find an approximation $\tilde{u} \in \tilde{\mathcal{V}}$ that satisfies the PDE operator in the Galerkin sense. The details of how this is accomplished are beyond the scope of this work. The important points, however, are that a finite element analysis must balance geometric error and approximation error while respecting stability constraints (*e.g.*, as discussed by Babuska and Aziz [11]), and that these errors are connected through the tessellation that is generated. The space of functions from which $\tilde{u}$ is generated depends on the type of elements that exist in $\tilde{\Omega}$. Thus the quality of the solution approximation is not only related to the accuracy of $\tilde{\Omega}$ for approximating Ω, but *also* to the geometric properties of the mesh elements. In the L^2 norm, the accuracy of $\tilde{u}$ is bounded by a constant that includes angles of triangular elements. Babuska and Aziz [11] show that if the largest triangle angle is bounded away from 180°, the

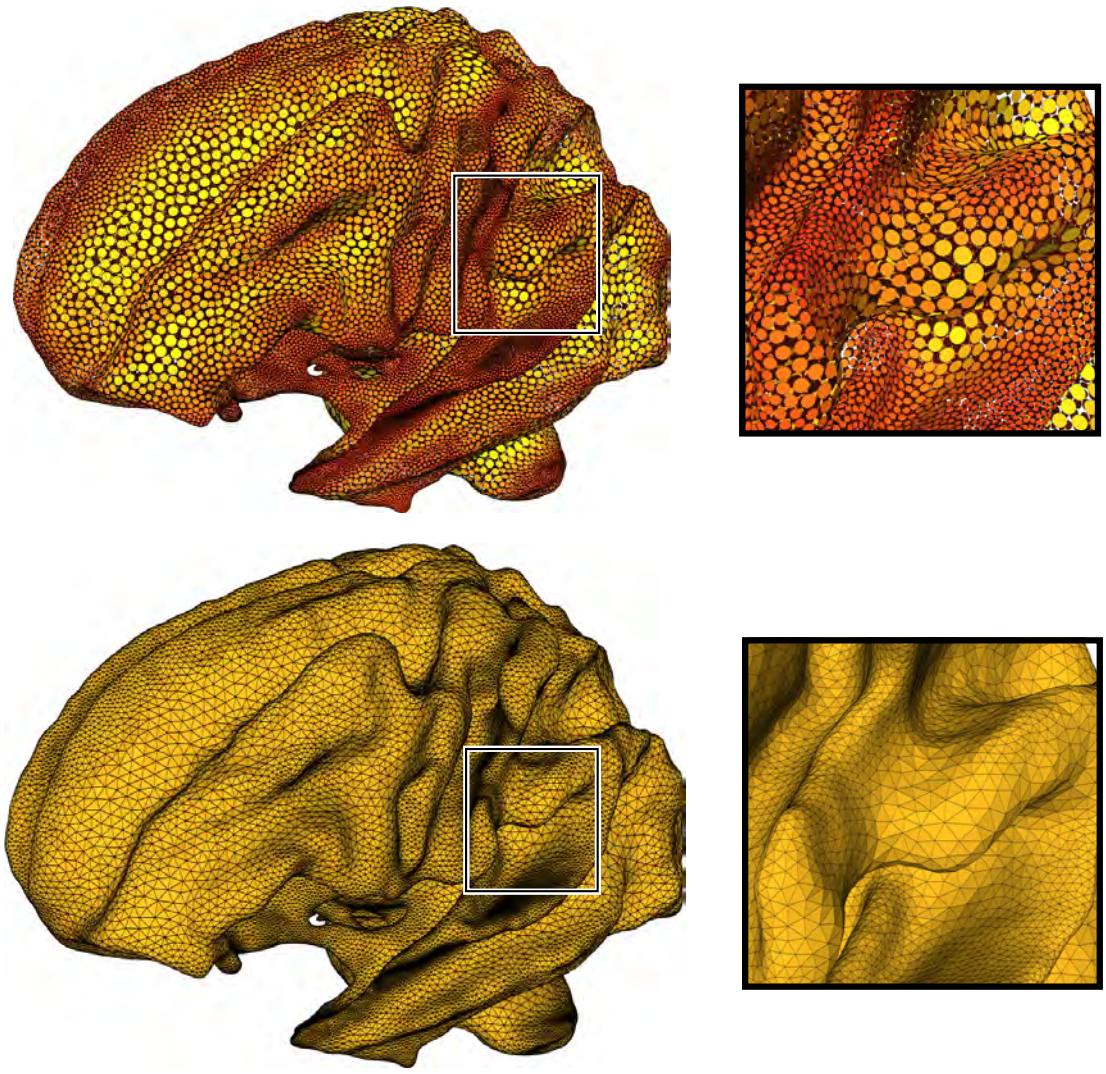


Figure 5.6. Particles on the brain and the resulting tessellation. The surface is a reconstruction of a white-matter segmentation [142].

finite element method converges as the triangle size decreases. Shewchuk [135] notes that small angles are preferable over large angles, so long as the largest angles are not too large, and extends these results to provide functions that guide mesh generation and refinement algorithms toward the production of high quality finite element tessellations. A common quality metric used in the literature for measuring this relationship of element angles is the ratio of the radii of the inscribed circle to the circumscribing circle of a triangle, r_{in}/r_{circ}. This metric penalizes triangles containing small angles, with the worst ratios going to triangles that also contain a large angle.

The proposed system addresses both aspects of geometric quality posed by the finite element method. First, the accuracy of the tessellation for capturing the topology of the domain is guar-

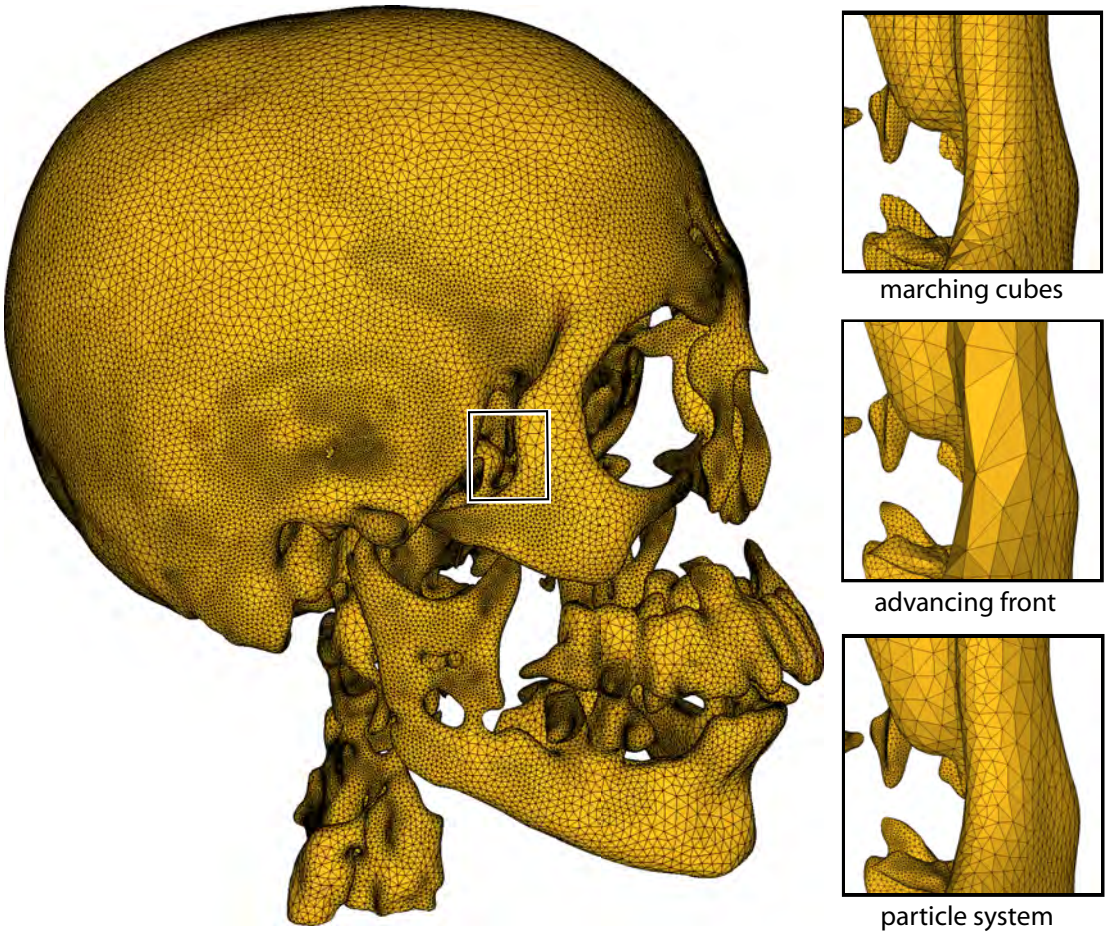


Figure 5.7. The skull mesh is generated by reconstructing a level-set of a gray-scale CT image. Close-ups are from triangulations generated using the proposed particle system method, an advancing front technique [129], and a marching cubes algorithm.

anteed by the Delaunay reconstruction algorithms for ϵ-distributions of particles. We quantify the ability of the proposed system to meet this requirement by computing the ratio of triangle edge lengths versus the average of h_0 at the edge vertices. In Figure 5.9 we present histograms of the results for each data set. The pelvis mesh contains no edges larger than h_0 dictates, and virtually every edge in the other three tessellations meets the sampling requirements defined in h_0 — less than 0.004% of the triangles in the brain and skull meshes, and less than 0.008% in the vessel mesh, contain an edge that falls above the required sampling length. We note that while the particles reliably meet the sampling requirements of h_0, the correctness of these requirements are ultimately related to the accuracy of the medial axis detection method, as discussed in Section 5.3.1. These results indicate that the proposed particle-based method is a practical scheme for generating an ϵ-sample of an isosurface that relates point density solely to the geometry of the

Figure 5.8. The vessel mesh represents the zero-set of a distance transform generated using an anisotropic smoothing algorithm [104].

surface, and not to the quality of the tessellation which is instead achieved implicitly by the low-energy configuration of particles.

The second finite element requirement for generating high quality tessellations is the production of nearly regular triangles. We compute the radius ratios for each data set to measure the quality of the triangles in the resulting meshes — Figure 5.10 displays these histograms. We present the average radius ratios for each data set in Table 5.4, along with the minimum (worst) ratio which is important for determining the condition number in a finite element simulation. We also include the radius ratios for meshes generated using a marching cubes [90] algorithm that has been modified to use the same reconstruction kernels as those used in the particle system, and from an advancing front algorithm [129] that has been supplied to us by the authors. The data indicate that the proposed method generates average radius ratios that are nearly identical to the advancing front technique, but consistently produces much better minimum ratios than either alternative triangulation method. The proposed system is able to (globally) produce very regular triangulations due to the natural, low energy, hexagonal packing of particles, avoiding the problems associated with grid-based methods (*i.e.*, restriction of vertices to grid edges) or advancing

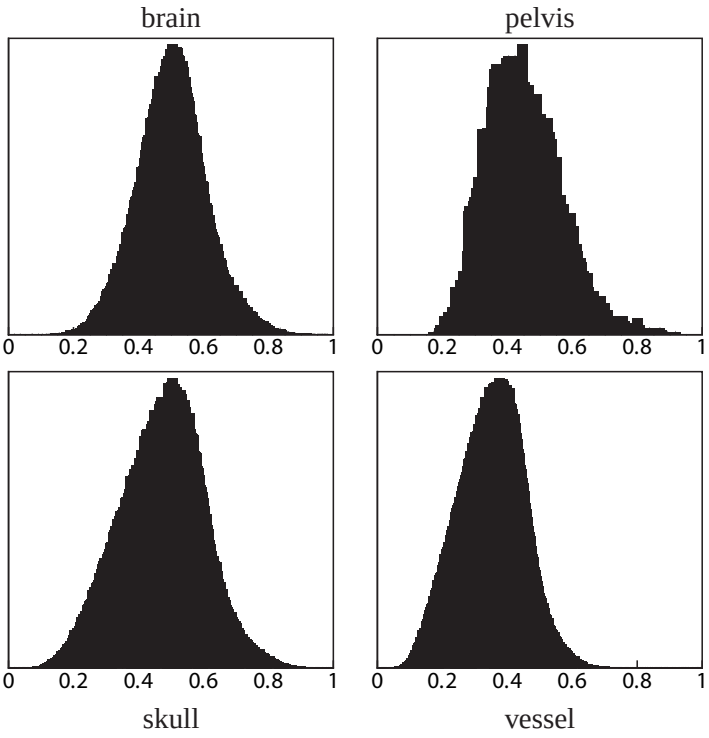


Figure 5.9. The edge length versus h ratios for the four data sets. Values greater than 1.0 were encountered at a frequency of less than 0.01% in the brain, skull, and vessel meshes.

front techniques (*i.e.*, detecting and stitching merging fronts) — poorly shaped triangles due to these problems are shown in Figure 5.7.

A third metric for measuring the quality of triangulations is a measure of the vertex valence of a mesh. For applications such as mesh compression [146] and subdivision surfaces [84] the regularity of the vertex valences across a mesh is important for efficient and accurate results. Regular triangulations tend toward a valence of six for most vertices, similar to the hexagonal properties of particle packing. Vertex valence also indirectly indicates the tendencies of a mesh to contain large and small angles. In Table 5.5 we summarize the valences of the vertices in our triangulations. The meshes indicate a good affinity for valence-six vertices (63.275%, compared with 44.15% for marching cubes and 71.75% for advancing front), with only a small fraction of vertices exhibiting valences greater than seven or less than five (0.7%, compared with 12.86% for marching cubes and 0.95% for advancing front). These numbers show that the particle-based method out-performs marching cubes while also containing a smaller percentage of extreme valences than the advancing front technique.

Practically, the proposed method relies on a lower bound for the LFS to ensure that the number of particles does not blow up. In general, however, implicit functions can have arbitrarily large

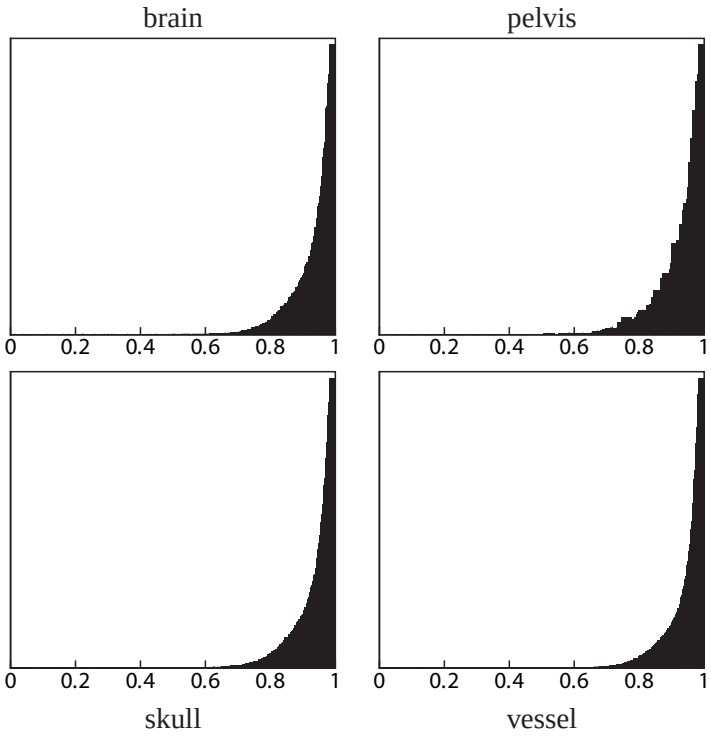


Figure 5.10. The radius ratios for the four data sets, all with an average ratio of ~ 0.94.

Table 5.4. The minimum and average radius ratios for each data set (*min/avg*) using the proposed particle-based method (*ps*), an advancing front scheme (*af*), and a modified marching cubes algorithm (*mc*).

	pelvis	brain	skull	vessel
ps	0.40/0.92	0.18/0.94	0.092/0.94	0.0195/0.94
af	0.23/0.94	0.02/0.93	0.006/0.93	0.0007/0.94
mc	0.00/0.66	0.00/0.67	0.000/0.66	0.0000/0.66

Table 5.5. Vertex valences for each data set, given as a percentage of the total number of vertices.

	< 5	5	6	7	> 7
brain	0.1	17.2	65.3	17.1	0.3
pelvis	0.3	22.3	56.2	20.8	0.4
skull	0.1	16.8	66.0	18.0	0.2
vessel	0.3	16.7	65.6	16.9	0.5

curvature, which results in very high densities of points around small features. By controlling the local configurations of binary voxels as well as the curvature of the implicit surface, *e.g.* through mathematical morphology operations, the curvature of the the reconstructed isosurface, and thus the maximum density of particles, can be controllably bounded. Levelset-based deformations [143, 133, 160], such as the *tightening* algorithm discussed in Section 2.5.2, can also systematically control the curvature of an implicit surface.

When the curvature of an implicit surface is bounded, we find that the particle-based meshing algorithm consistently generates triangulations with minimimum radius-ratios in the 0.2–0.5 range — Figure 5.11 presents one such example. This mesh consists of approximately 407,000 triangles, and has a minimum radius ratio of 0.38. The sampled implicit function was generated from a binary segmentation of a dendrite [51] that was tightened with radius of 1.0.

5.6 Discussion

In this chapter we propose a particle-based method for generating high-quality tessellations of biological data sets. By creating a sizing field to dictate the density of the particle distributions, this method produces sets of points that can meet the sampling requirements of Delaunay surface reconstruction algorithms for generating topologically accurate tessellations, the accuracy of which depends on the correctness of the underlying medial axis detection algorithm. We present results from a variety of data sets that indicate the proposed method can reliably produce meshes that closely capture isosurface geometry, as well as generate very regular triangulations. We also compare the method against other tessellation techniques and show that the particle-based scheme generates consistently higher minimum radius ratios, an important characteristic for reducing geometric error in finite element simulations.

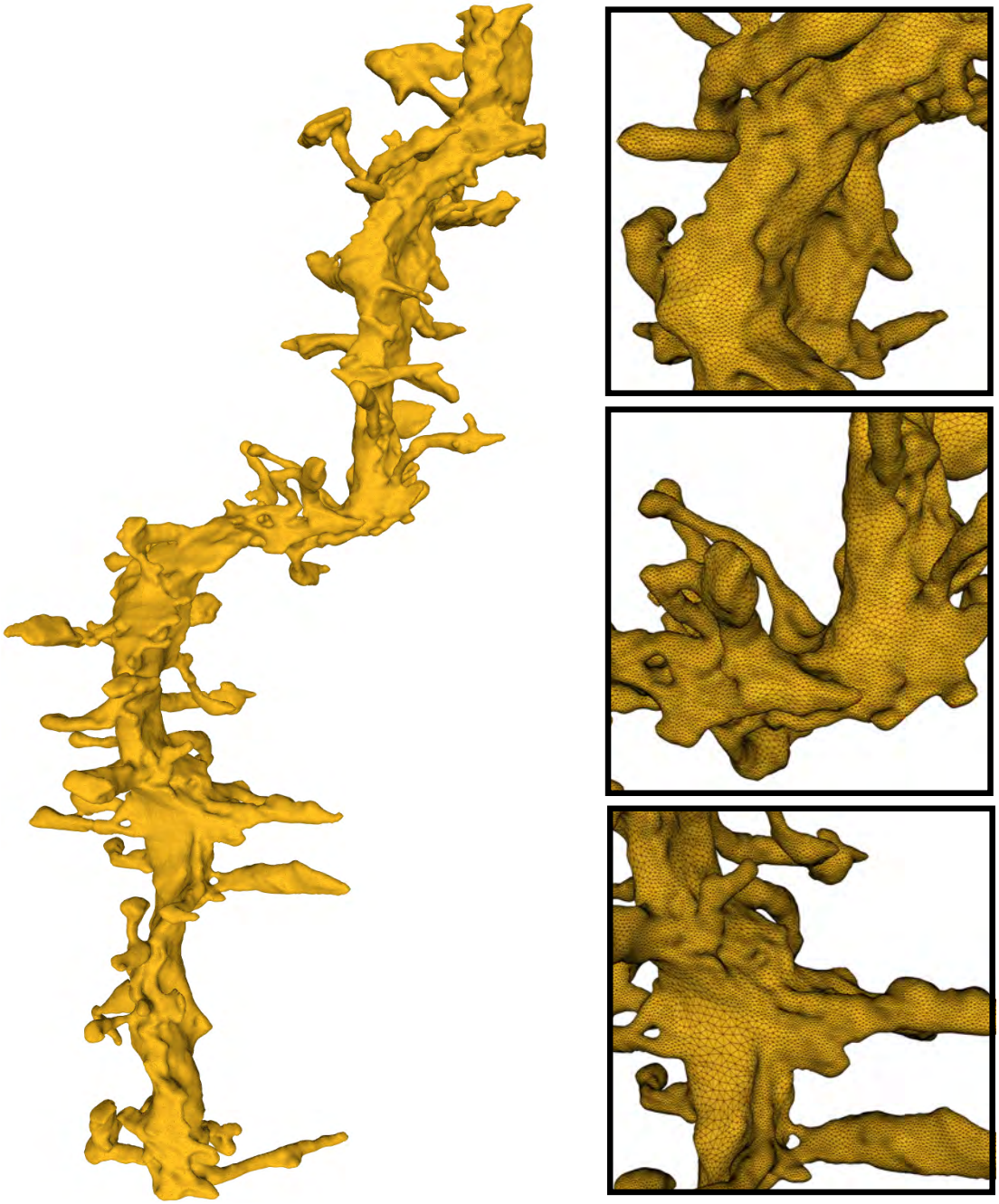


Figure 5.11. A high-quality mesh of a spinny dendrite segmentation [51]. The triangulation contains over 400,000 elements, and has a minimum radius ratio of 0.38.

CHAPTER 6

SAMPLING AND MESHING OF MULTIMATERIAL VOLUMES

Methods that faithfully and robustly capture the geometry of complex material interfaces in labeled volume data are important for generating realistic and accurate visualizations and simulations of real-world objects. The generation of such multimaterial models from measured data poses two unique challenges: first, the surfaces must be well-sampled with regular, efficient tessellations that are consistent across material boundaries; and second, the resulting meshes must respect the nonmanifold geometry of the multimaterial interfaces. In this chapter we propose a strategy for sampling and meshing multimaterial volumes using dynamic particle systems. We present a novel differentiable representation of the material junctions that allows the particle system to explicitly sample corners, edges, and surfaces of material intersections. We show that the resulting point distributions meet fundamental sampling constraints, allowing Delaunay-based meshing algorithms to reliably extract watertight meshes of consistently high-quality.

6.1 Introduction

Volumetric scans (volumes) provide an important source of information for generating realistic computer models of real-world objects. For example, biological and geophysical data are often captured using volumetric scanning methods such as magnetic resonance imaging (MRI) or ultrasound. The data from these devices is usually stored as a regular grid of values that provide information about the surface of the scanned object and its detailed internal structure. Most objects, natural or man-made, contain multiple materials with vastly different physical properties that are typically organized in complicated geometric configurations. Extracting precise geometric models of the interfaces between these materials is important both for visualization and for realistic physically-based simulations in a variety of fields, from biomedical computing and computer animation to oil-and-gas exploration and engineering.

Multimaterial volumes impose particular challenges for sampling and meshing algorithms because the boundaries between materials are typically not smooth manifolds. As a result,

intersections of materials can produce sharp features such as edges and corners (see Section 6.3). Furthermore, the development of increasingly realistic simulations dictates additional constraints, such as a sufficient number of samples to accurately represent the geometry, compact sets of nearly-regular triangles, and consistent tessellations across material boundaries. The construction of geometric models that meet these requirements for surfaces of distinct objects is well-studied. However, generating high-quality models of objects that contain multiple materials has thus far received little attention.

In this chapter we use a dynamic particle system to produce well-spaced distributions of points on material interfaces in multimaterial volumes. The particles move to minimize an objective function that is designed to produce configurations of samples that are locally adaptive, geometrically accurate, and well-suited for subsequent meshing. The set of material boundaries are described as a CW-complex [79], and this chapter proposes new, analytic representations for the different kinds of cells that form this structure. Also defined are projection operators that allow the particles to sample these material boundaries in a hierarchical fashion — 0-cells, 1-cells, and then 2-cells. The result is a set of surface points that adapt to the underlying geometry and meet fundamental surface sampling requirements. Using Delaunay-based meshing schemes, an algorithm based on labeling tetrahedron creates surface meshes that are well-suited to the generation of well-shaped volumetric elements [4].

The main contribution of the chapter is a novel scheme for representing the nonmanifold sets formed at the material interfaces in multimaterial volume data, and a corresponding set of projection operators that allow these interfaces to be sampled with dynamic particle systems. We also present an algorithm for distributing sets of particle systems such that each type of interface is sampled explicitly, and show that high-quality surface meshes of the sample points exist as well-defined subsets of a Delaunay tetrahedralization. Implementation details of the proposed algorithm are presented, and results for several multimaterial volumes generated from MRI scans of real-world objects are shown, demonstrating the proposed system's effectiveness.

6.2 Previous Work

Most of the previous work on meshing multiple material data focuses on grid-based tessellation algorithms. These algorithms work on the original, labeled volumes, and focus on extensions of the marching cubes case tables to handle the nonmanifold surface intersections. A postprocessing step is then applied to reduce the voxelization artifacts. Some of the earliest work presents methods that generate nonmanifold meshes from tetrahedral elements that are created

from the original rectilinear volume. Bloomenthal and Ferguson [17] propose a scheme that first subdivides each voxel in a multilabel dataset into six tetrahedral elements. From these elements, intersections of multiple materials along element edges, as well as in element interiors, can be determined such that a nonmanifold triangulation can be reliably extracted. The *marching tetrahedral* method [105] adapts the well-known *marching cubes* algorithm [90] for the tessellation of irregular grids, and is extended by Bonnell *et al.* [20] to incorporate fractional volume information of multiple material datasets. Dillard *et al.* [46] also extend the marching tetrahedra scheme for generating boundary meshes of data consisting of thousands of materials from polycrystal data.

Resolving the triangulation ambiguities of the nonmanifold topology when tessellating the multimaterial interfaces is straight-forward using tetrahedral elements, but results in an excessive number of triangles. Methods based on extracting interfaces in hexahedral cells can reduce the overall number of triangles, but they must handle an increased number of nonmanifold ambiguities. Reitinger *et al.* [120] propose a scheme for extracting nonmanifold meshes from a regular grid of multilabel data by specifically detecting voxels that contain more than two materials, placing a vertex inside these voxels, and, through a series of heuristics, computing a triangulation of the voxels' intersection points. Bertram *et al.* [12] produce a single multiple material dataset from time-sequenced volumes using a *fairing* procedure that produces signed distance functions of the materials from which a nonmanifold tessellation is extracted using an algorithm similar to that proposed by Hege *et al.* [66]. By preprocessing multiple material datasets to eliminate voxels with more than three materials, Bischoff and Kobbelt [13] greatly reduce the complexity of the triangulation of these datasets. Recently, Zhang *et al.* [166] proposed an octree-based approach that relies on the dual contouring method [71] to produce adaptive tetrahedral elements as well as to preserve sharp features.

In general, these irregular and regular grid-based methods are robust and efficient, but generate large numbers of triangles that are usually poorly shaped because the methods do not focus on the placement of the vertices. Furthermore, the size of the elements is related to the resolution of the grid, rather than to the geometry of the material surfaces, and the resulting meshes must be postprocessed to reduce voxelization artifacts and generate tessellations suitable for simulation [37]. To remove the dependence of the tessellation resolution from the underlying grid, Pons *et al.* [118] extend the Delaunay-based volume meshing algorithm of Oudet *et al.* [110] to multimaterial datasets. This approach instead builds a geometric model of multiple materials by subdividing a Delaunay tetrahedral mesh, generating consistent material interfaces by construction. The algorithm defines material boundaries as a subset of the tetrahedralization for all

faces that are bounded by tetrahedral elements belonging to different materials. The quality of the resulting elements are controlled through a refinement procedure, the geometric accuracy of which is proved by Boissonnat *et al.* [19]. This proof, however, requires that the surfaces be C^2 continuous, which is not the case for multiple material data — we will discuss in Section 6.3 that the surfaces of materials in a multilabel dataset are only C^0 as they contain sharp features where more than two materials intersect.

6.3 Topology of Multimaterial Interfaces

We represent interfaces in a multimaterial dataset using a model that describes each material with a smooth, volumetric *indicator function*, f_i [92]. A set of N indicator functions $F = \{f_i | f_i : V \mapsto \Re\}$ represents N materials. The model assigns a material label i to a point $x \in V$ if (and only if) $f_i(x) > f_j(x) \ \forall \ j \neq i$.

Looking at the simple case when only two materials exist in the dataset, for all points x where $f_1(x) - f_2(x) > 0$ the model will assign a label of 1, while assigning a label of 2 otherwise. Notice that this description corresponds to the conventional formulation of an implicit surface. In the multimaterial model, the set of points x where $f_1(x) - f_2(x) = 0$ forms the interface, or *junction*, between these two materials.

In the case of an arbitrary number of materials in the dataset, the configurations of interfaces become somewhat more complex. The boundaries that separate materials are no longer manifold, and can form sharp corners and edges. The topology of these junctions, however, can be characterized by certain *generic* configurations (see Figure 6.1). The term *generic*, from the field of singularity theory, refers to the cases where the set of functions F are in *general position*. This situation is analogous to the finite-dimensional spaces considered in discrete geometry —

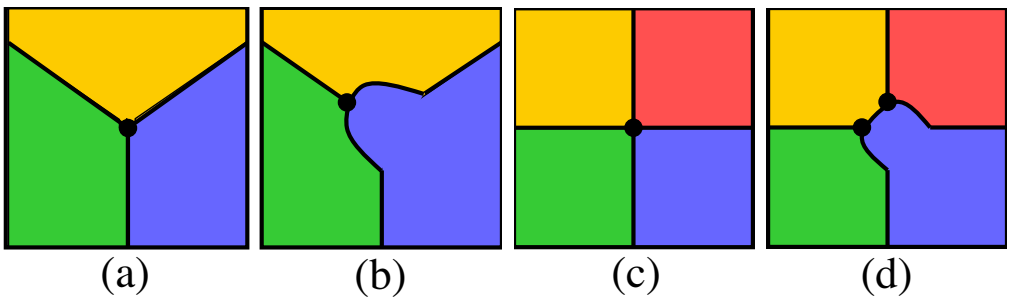


Figure 6.1. In 2D, a 3-material junction is generic and forms a 0-cell (a); it maintains its topology under small perturbations (b). A 4-material junction (c), however, is a nongeneric case, and is anihilated under small perturbations (d) to form generic 2- and 3- material junctions.

i.e., three points in general position cannot lie on a line, and if they do, a general position can be restored through very small perturbations. This work generally considers only generic configurations, which is justified by the system's reliance on measured data that inherently contains some level of noise, as well as the use of a data processing pipeline that ensures a degree of smoothness in the indicator functions (see Section 6.6.1).

Each material interface is characterized in terms of the number of material indicator functions that are maximal (and equal) at that junction. For $V \subset \Re^2$, 2-junctions and 3-junctions occur generically, as shown in Figure 6.1 (a-b), while a 4-junction is a nongeneric case, as shown in Figure 6.1 (c-d). For $V \subset \Re^d$ each K-junction forms a subset of V that is topologically equivalent (homeomorphic) to a P-disk, where $P = d - K + 1$. Thus each type of material junction can be considered a P-cell, as described in the literature on discrete topology [61]. Thus, generically, for $d = 3$ we have 4-junctions, which are 0-cells or points; 3-junctions, which are 1-cells or curves; and 2-junctions, which are 2-cells or surfaces.

The collection of cells that describe the different types of material junctions, taken together, form a CW-complex. That is, we can organize them hierarchically, such that each 2-cell is attached to a collection of 1-cells (at its border), and each one cell is attached to one or more 0-cells. Special care must be taken to describe the cases where a particular material junction is itself closed, but this formality is not important to the proposed method. The strategy in this chapter is to sample nonmanifold multimaterial boundaries using this hierarchy of manifolds, and to form the appropriate relationships between samples at each level in the hierarchy.

6.4 Representing and Sampling Junctions

Given a set of material indicator functions F, first defined are a set of analytical *cell indicator functions*, J, that approximate the cells formed by each type of material junction (Section 6.4.1). In the proposed particle system sampling scheme, each particle will be constrained to a particular material junction. The formulation for each type of cell includes a set of projection operators to enforce this constraint (Section 6.4.2). The system generates a hierarchy of particle systems so that each type of generically occurring material junction is represented in the final mesh. Thus, in 3D we begin by sampling the 0-cells (points), followed by the 1-cells (curves), and concluding with the 2-cells (surfaces). Finally, the multimaterial surface meshes are extracted as a subset of a Delaunay tetrahedralization of the samples (Section 6.4.3).

6.4.1 Differentiable Multimaterial Junctions

The individual material junctions present in the volumetric model of multimaterial datasets are analytically represent to allow sets of particles to specifically sample each junction. To do this, *inside/outside* (IO) functions for each material are defined using the volumetric model described in Section 6.3. These functions are:

$$\tilde{f}_i = f_i - \max_{j=1, j\neq i}^{n} f_j, \tag{6.1}$$

where positive values indicate the presence of material i and negative values indicate some other material. These functions have the property that the zero-set of any one IO function, $\tilde{f}_i$, coincides with the material transitions between i and some other material. This means, for instance, that for two adjacent materials, i and j, we have $\tilde{f}_i = \tilde{f}_j = 0$ along the 2-junction where these two materials meet.

This coincidence of zero-sets for adjacent materials in Equation 6.1 allows for a novel representation that approximates the different kinds of material junctions (cells) within a multimaterial volume. These junctions are detected by a *cell indicator function* that identifies points in V where a set of IO functions evaluate to zero, such as the material interface between materials 1 and 2 shown as the red dashed line in Figure 6.2. Along this curve, $\tilde{f}_1 = \tilde{f}_2 = 0$ and $\tilde{f}_3 < 0$, while in the the vicinity of this curve $\tilde{f}_1$ and $\tilde{f}_2$ will be nonzero (one negative and the other positive). Thus, in 3D, we can represent the set of 2-cells that form the interface between two materials i and j, where $i \neq j$, as the zero-set of the continuous cell indicator function:

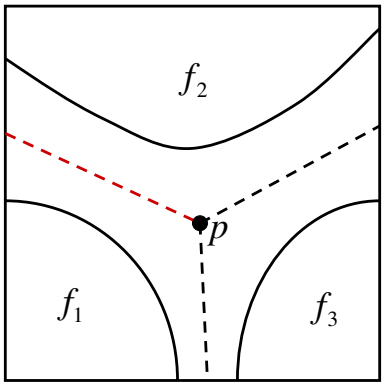


Figure 6.2. Material interfaces in multimaterial datasets exist where a volumetric model of the data transitions from one maximal material to another, shown by the dotted lines for a set of three indicator functions.

$$J_{ij} = \tilde{f}_i^2 + \tilde{f}_j^2. \tag{6.2}$$

In this scheme, the 1-cells for the set of materials i, j, k (assumed distinct) are given by the set of points $J_{ijk} = 0$ where:

$$J_{ijk} = \tilde{f}_i^2 + \tilde{f}_j^2 + \tilde{f}_k^2, \tag{6.3}$$

and likewise, the indicator for a 0-cell is:

$$J_{ijkl} = \tilde{f}_i^2 + \tilde{f}_j^2 + \tilde{f}_k^2 + \tilde{f}_l^2. \tag{6.4}$$

6.4.2 Sampling Multimaterial Junctions with Particles

To distribute a set of dynamic particles across a manifold we need to define two things: first, how particles will be projected onto the manifold; and second, how particles will be constrained to move along the manifold. The first case is usually done using a gradient descent method such as Newton-Raphson, while the latter case is most often accomplished by projecting motion vectors onto the local tangent space of the manifold. For distributing particles across multimaterial intersections, both of these tasks require first derivative information of the cell indicator functions.

The gradient of Equation 6.2 (with analogous definitions for Equations 6.3 and 6.4) is:

$$\nabla J_{ij} = 2\tilde{f}_i \nabla \tilde{f}_i + 2\tilde{f}_j \nabla \tilde{f}_j. \tag{6.5}$$

The $\max$ function is only C^0, however, and the derivative is not defined at the transition between materials. Thus, we approximate $\max$ with a smooth function that is differentiable and can be tuned (via a parameter) to be arbitrarily close to $\max$. One example of an analytic, differentiable approximation to $\max$ for a set Z of unique values $z_1, z_2, \cdots z_m$ is given by first defining a function g:

$$g(z) = 1 + \frac{z}{(z^2 + \epsilon_{\max}^2)^{1/2}}. \tag{6.6}$$

The $\max$ function is then:

$$\max(Z) = \frac{1}{2^{m-1}} \sum_{i=1}^{m} z_i \prod_{j=1, j\neq i}^{m} g(z_i - z_j) \tag{6.7}$$

with the gradient given by:

$$\nabla \max(Z) = \frac{1}{2^{m-1}} \sum_{i=1}^{m} \left[\nabla z_i \prod_{j=1, j\neq i}^{m} g(z_i - z_j) + z_i \left(\sum_{j=1, j\neq i}^{m} \nabla g(z_i - z_j) \prod_{j=1, j\neq i}^{m} g(z_i - z_j) \right) \right] \tag{6.8}$$

where:

$$\nabla g(z) = \nabla z \left[\frac{1}{(z^2 + \epsilon_{\max}^2)^{1/2}} - \frac{z^2}{(z^2 + \epsilon_{\max}^2)^{3/2}} \right]. \tag{6.9}$$

At the material transitions there is a not a unique set of a values, however. Thus, when evaluting the IO functions of the transitioning materials for a cell indicator function, the $\max$ in Equation 6.1 requires an extension of Equation 6.7 to accomodate these nonunique (maximal) values:

$$\max(F) = \frac{1}{(K-1)2^{n-K}} \sum_{i=1}^{n-1} f_i \prod_{j=1, j\neq i}^{n-1} g(f_i - f_j) \tag{6.10}$$

where n is the number of materials in V, F is the set of $(n-1)$ indicator functions evaluated by $\max$ in Equation 6.1, and K is the numer of materials in transition for a specific cell indicator function. For the results in this chapter we use $\epsilon_{\max} = 10^{-5}$.

Using the approximation to $\max$ given in Equation 6.10, we can derive an expression for the lower-bound of the cell indicator function J, and show that this bound goes to zero in the limit. The value of the transitioning materials at the junction is defined as A, and the largest value of the nontransitioning materials at the juncton is defined as B. The difference between these two values is $(A-B) > 0$. Using Equation 6.6, we define: $\alpha = g(A-B)$, which is a number slightly smaller than 2; $\beta = g(B-A)$, which is a number slightly larger than 0; and $g(A-A) = g(B-B) = 1$. The expression for the cell indicator function of a K-junction becomes:

$$J = K\left[A - \frac{1}{(K-1)2^{n-K}}\left((K-1)A\alpha^{n-K} + (n-K)B\beta^{K-1}\right)\right]^2 \tag{6.11}$$

In this expression, as the difference between A and B gets large relative to $\epsilon_{\max}$, $\alpha \to 2$ and $\beta \to 0$, which causes $J \to 0$. This expression also shows that for the special case of $n = K$ we have $J = 0$.

Notice that the cell indicator functions have the property that they are (nearly) zero on the set of interest (*i.e.*, a material junction) and positive everywhere else. Because the set of interest is locally minimal, the gradient is zero on the material junctions, and thus these cell indicators, unlike the IO functions, do not directly provide the tangent spaces that are needed to constrain the motion of interacting particles. The cell indicators are constructed, however, from combinations of implicit functions for the individual materials (*i.e.*, the IO functions), and the gradients of these IO functions give the local orientation of the cells. Thus the tangent spaces (planes or lines in 3D) of the cells are reconstructed from a series of projection operators that rely on gradients of the corresponding IO functions.

For 2-cells, the gradients of the IO functions that characterize the junction will be approximately equal and opposite near the zero set. Thus we project a motion vector of a particle, $\mathbf{v}$, onto a tangent plane that is defined by the average (for numerical robustness) of these IO function gradients:

$$\mathbf{n}_t = \frac{\nabla\tilde{f}_i - \nabla\tilde{f}_j}{|\nabla\tilde{f}_i - \nabla\tilde{f}_j|} \tag{6.12}$$

and update the motion vector as:

$$\mathbf{v} \leftarrow \mathbf{v} - < \mathbf{v}, \mathbf{n}_t > \mathbf{n}_t = (I - \mathbf{n}_t \otimes \mathbf{n}_t)\mathbf{v}. \tag{6.13}$$

On the 1-cells, particles must move along the tangent line of the zero-set of J_{ijk}. A tangent line is computed as the summation of the cross products of each pair of the three characterizing IO function normals, which all have zero-crossings along that set:

$$\mathbf{t}_{ijk} = \frac{\nabla \tilde{f}_i}{|\nabla \tilde{f}_i|} \times \frac{\nabla \tilde{f}_j}{|\nabla \tilde{f}_j|} + \frac{\nabla \tilde{f}_j}{|\nabla \tilde{f}_j|} \times \frac{\nabla \tilde{f}_k}{|\nabla \tilde{f}_k|} + \frac{\nabla \tilde{f}_k}{|\nabla \tilde{f}_k|} \times \frac{\nabla \tilde{f}_i}{|\nabla \tilde{f}_i|}. \tag{6.14}$$

The particle motion vectors are then projected onto this normalized tangent line, constraining the motions to the 1-cell:

$$\mathbf{v} \leftarrow \left\langle \frac{\mathbf{t}_{ijk}}{|\mathbf{t}_{ijk}|}, \mathbf{v} \right\rangle \frac{\mathbf{t}_{ijk}}{|\mathbf{t}_{ijk}|}. \tag{6.15}$$

The 0-cells, which are the first to be sampled, are isolated points, each sampled by a single particle (see Section 6.6.2), thus, we do not need to define a projection operator.

6.4.3 Meshing Multimaterial Samples

Fundamental work in inferring correct topology from a set of unorganized surface points relies on a sampling criteria that link the density of points to the LFS of the surface [6]. In order to guarantee topologically and geometrically correct surface reconstructions, state-of-the-art surface sampling results require an infinite sampling density (in the limit) near sharp features, such as those formed by at the 0-cells in a multimaterial dataset. The proof in Section 6.5.1, however, shows that the LFS sampling constraint can be lifted around sharp features if sample points are placed explicitly on cells, allowing for the reconstruction of geometrically and topologically correct tessellations of surfaces with sharp features using Delaunay-based meshing schemes. The proof guarantees this claim in 2D for a lower bound of 45° on the material angles formed by the tangent lines of the 1-cells at the 0-cell where they meet. Results from the levelset literature indicate that angles between 1-cells, defined by smooth indicators functions, are 120° at the 0-cells [168], making the 90° lower bound a reasonable constraint for this work. We anticipate a similar result, with a larger angle constraint, in 3D.

Based on this proof, we know that there exists a Delaunay-based method that can reconstruct topologically correct manifold surfaces of individual materials in multimaterial datasets from the set of particles. These datasets contain additional information, though — namely, a material label for almost ever point in V — that allows for a simple labeling algorithm to reliably extract the manifold material surfaces, as well as the nonmanifold intersection surface [118]. The labeling algorithm first computes a Delaunay tetrahedralization of the the sets of points sampling the 0-, 1-, and 2-cells. Next, each tetrahedron is assigned a material label by determining the material

type at the location of its circumsphere center. Finally, the algorithm generates a surface mesh by extracting all faces bounded by tetrahedra with different material labels.

6.5 Analysis and Correctness of Algorithm

6.5.1 Extension of Sampling Requirements to Include Sharp Features

Individual materials in a multimaterial dataset can be constructed as a union of junctions that bound the material. For materials that have 3- and 4-junction boundaries, sharp features appear around these boundaries at the nonmanifold material intersections. These sharp features present a problem when considering the sampling constraints imposed by Delaunay-based surface reconstruction algorithms. These constraints specify that the density of surface samples, which form the vertices of the reconstructed tessellation, is inversely proportional to the distance to the medial axis at those sample locations. This distance (the LFS) goes to zero at sharp edges where the medial axis touches the discontinuous feature. Thus, the number of particles that sample a material near a 3- or 4-junction must go to infinity to meet this constraint.

There is a way around the intractability of the sampling constraint near sharp features, which is to place sample points *directly* on these features. The following derivation shows that if the angle θ between two double-junctions near a sharp feature is large enough, and that the sharp feature is explicitly sampled, then the sampling constraints do not apply in a small region around the feature. The derivation states that in 2D, θ must be larger than 45° — we anticipate a similar result, with a larger angle constraint, in 3D. These proofs hold so long as the samples nearest the sharp features lie within a ball that contains subsets of the surface for which the lower-bound on the tangent-line angles is met.

6.5.1.1 Definitions

- A circle is *empty* if its interior contains no points.
- A *circumcircle* is a circle touching a set of points that is empty of those points.
- Given three points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, the *Voronoi vertex* of these points is the center of the circumcircle of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$.
- Given three points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, the *Voronoi vertex* of these points is at the intersection of the perpendicular bisectors of $\overline{\mathbf{ab}}$,$\overline{\mathbf{bc}}$, and $\overline{\mathbf{ca}}$.
- The edge $\overline{\mathbf{ab}}$ is in the Delaunay triangulation of the set of points containing $\mathbf{a}$ and $\mathbf{b}$ if there

exists an empty circumcircle for $\mathbf{a}$ and $\mathbf{b}$.

6.5.1.2 Delaunay Surface Reconstruction in 2D

Here we present a brief, high-level overview of the 2D Delaunay surface reconstruction algorithm CRUST [6], which builds on the fundamental LFS sampling requirements. The algorithm begins by taking a set of points $P = \mathbf{p}_1, \cdots, \mathbf{p}_n$ that sample a curve, and computing their corresponding Voronoi vertices $V = \mathbf{v}_1, \cdots, \mathbf{v}_m$. Next, a Delaunay triangulation of $P \cup V$ is generated. From the set of all edges in the Delaunay triangulation, the edges not connecting two sample points in P are culled, leaving a set of edges called the *crust* — in Figure 6.3 the crust is shown by the bold lines. In their seminal work on this algorithm [6], Amenta *et al.* show that if P is sufficiently dense with respect to the LFS of the curve, the *crust* is guaranteed to contain edges that connect only adjacent surface points, and to also be topologically homeomorphic to the original curve.

6.5.1.3 Derivation of of Material Angle Constraints

Given the set of points shown in Figure 6.4 that sample a curve with a discontinuity and a material angle $\theta > 45°$, this derivation shows that the edge $\overline{\mathbf{p}_1\mathbf{p}_3}$ will not exist in the *crust* of the points. There are several assumptions that we make:

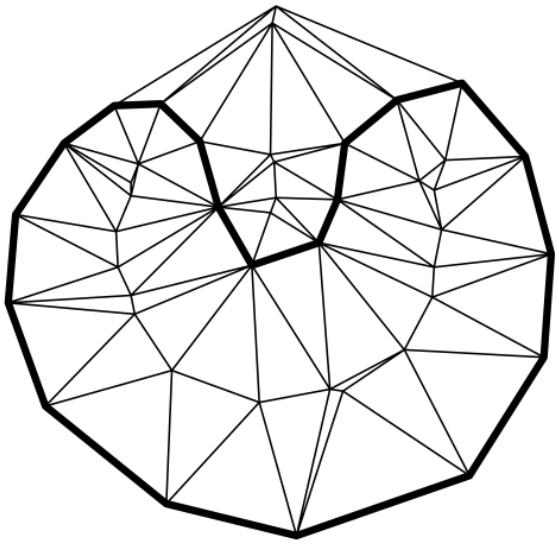


Figure 6.3. When a curve is sampled densely enough, the *crust* exists as a subset of edges of a Delaunay triangulation of the surface sample points and their Vornoi vertices (image used without permission).

- $\mathbf{p}_2$ samples the discontinuity exactly
- $|\overline{\mathbf{p}_1\mathbf{p}_2}| = |\overline{\mathbf{p}_3\mathbf{p}_2}|$
- $|\overline{\mathbf{p}_4\mathbf{p}_1}| = |\overline{\mathbf{p}_5\mathbf{p}_3}|$
- $|\overline{\mathbf{p}_1\mathbf{p}_2}| \leq |\overline{\mathbf{p}_4\mathbf{p}_1}|$

Given these assumptions, there are two cases to consider:

- $90^\circ \leq \theta \leq 180^\circ$
- $0^\circ < \theta < 90^\circ$

6.5.1.3.1 Case 1. For the first case, where $90^\circ \leq \theta \leq 180^\circ$, we have the situation depicted in Figure 6.5. The Voronoi vertex $\mathbf{v}$ is at the center of the circumcircle B of points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. In this situation the smallest circumcircle of $\overline{\mathbf{p}_1\mathbf{p}_3}$ that is empty of $\mathbf{p}_2$ is the circumcircle B. By definition, $\mathbf{v}$ will lie at the center of B. Thus, B will not be empty of points in $P \cup V$, and hence $\overline{\mathbf{p}_1\mathbf{p}_3}$ will not be an edge in the Delaunay triangulation of $P \cup V$.

6.5.1.3.2 Case 2. For the second case, where $0^\circ < \theta < 90^\circ$, we have the situation depicted in Figure 6.6. The Voronoi vertex $\mathbf{v}_1$ is at the intersection of the perpendicular bisectors of $\overline{\mathbf{p}_1\mathbf{p}_2}$, $\overline{\mathbf{p}_2\mathbf{p}_3}$, and $\overline{\mathbf{p}_3\mathbf{p}_1}$, and the Voronoi vertex $\mathbf{v}_2$ is at the intersection of the perpendicular bisectors of $\overline{\mathbf{p}_4\mathbf{p}_1}$, $\overline{\mathbf{p}_1\mathbf{p}_3}$, and $\overline{\mathbf{p}_3\mathbf{p}_4}$. The point $\mathbf{c}$ lies on $\overline{\mathbf{p}_1\mathbf{p}_3}$ and is the center of the smallest circumcircle of $\mathbf{p}_1$ and $\mathbf{p}_3$. In this situation, we want to know what is the largest θ for which the smallest circumcircle of $\mathbf{p}_1$ and $\mathbf{p}_3$ is empty of $\mathbf{v}_1$ and $\mathbf{v}_2$. The circumcircle of $\mathbf{p}_1$ and $\mathbf{p}_3$ will not be

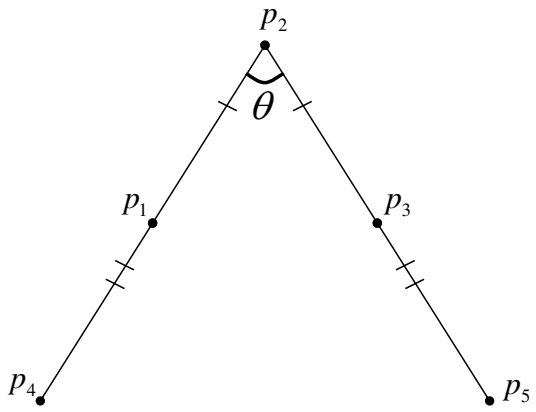


Figure 6.4. The sampling assumptions for the 2D proof. The point $\mathbf{p}_2$ explicitly samples the tip of the sharp feature, $|\overline{\mathbf{p}_1\mathbf{p}_2}| = |\overline{\mathbf{p}_3\mathbf{p}_2}|$, $|\overline{\mathbf{p}_4\mathbf{p}_1}| = |\overline{\mathbf{p}_5\mathbf{p}_3}|$, and $|\overline{\mathbf{p}_1\mathbf{p}_2}| \leq |\overline{\mathbf{p}_4\mathbf{p}_1}|$.

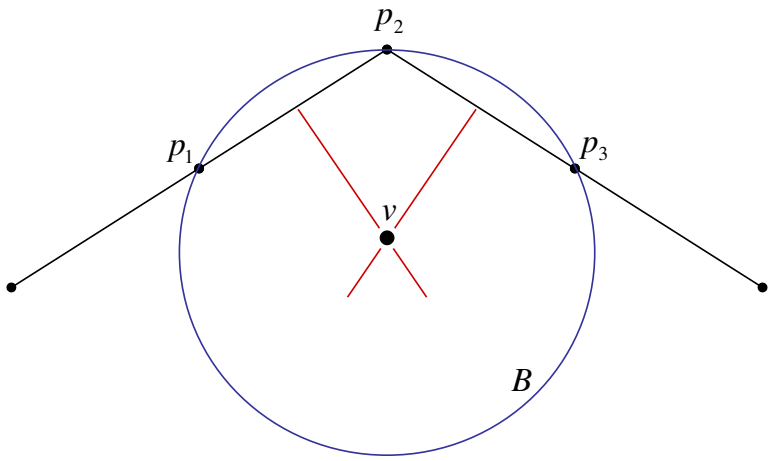


Figure 6.5. The case where $90° \leq \theta \leq 180°$.

empty when either:

- $|\mathbf{c} - \mathbf{v}_1| < \frac{1}{2}|\mathbf{p}_1 - \mathbf{p}_3|$
- $|\mathbf{c} - \mathbf{v}_2| < \frac{1}{2}|\mathbf{p}_1 - \mathbf{p}_3|$

To simplify the following discussion, we let $|\mathbf{p}_1 - \mathbf{p}_2| = 1$ without loss of generality, and let $\gamma = \frac{1}{2}\theta$.

We do not, however, need to consider the case where $|\mathbf{c} - \mathbf{v}_2| < \frac{1}{2}|\mathbf{p}_1 - \mathbf{p}_3|$ because for $\gamma > 0°$, $|\mathbf{c} - \mathbf{v}_1| < |\mathbf{c} - \mathbf{v}_2|$. To see why this is true, Figure 6.7 shows the relationship between $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{c}$. The point $\mathbf{c}$ falls halfway between $\mathbf{v}_1$ and $\mathbf{v}_2$ when $\overline{\mathbf{p}_1\mathbf{c}}$ is perpendicular to $\overline{\mathbf{p}_1\mathbf{p}_2}$

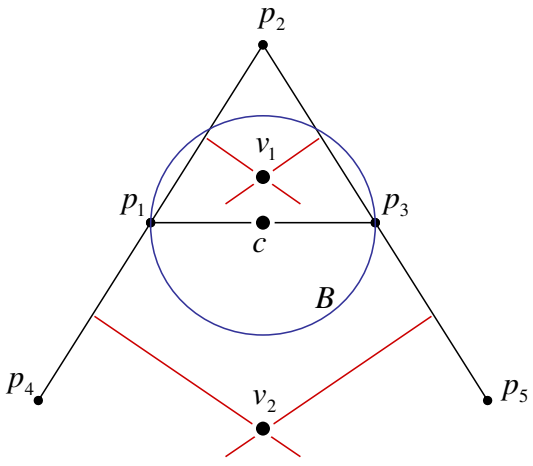


Figure 6.6. The case where $0° < \theta < 90°$.

— this assumes that $|\mathbf{p}_1 - \mathbf{p}_2| = |\mathbf{p}_4 - \mathbf{p}_1|$; if $|\mathbf{p}_1 - \mathbf{p}_2| < |\mathbf{p}_4 - \mathbf{p}_1|$ the point $\mathbf{c}$ will never fall halfway between $\mathbf{v}_1$ and $\mathbf{v}_2$. Thus, for $\gamma > 0°$, $|\mathbf{c} - \mathbf{v}_1| < |\mathbf{c} - \mathbf{v}_2|$. This means that we only need to consider when $\mathbf{v}_1$ is inside of the circumcircle of $\mathbf{p}_1$ and $\mathbf{p}_3$.

Finally, we determine when $|\mathbf{c} - \mathbf{v}_1| < \frac{1}{2}|\mathbf{p}_1 - \mathbf{p}_3|$. From Figure 6.8 we see that:

$$|\mathbf{c} - \mathbf{v}_1| = x \tan(90° - 2\gamma) \tag{6.16}$$

where $x = |\mathbf{c} - \mathbf{p}_1|$. This leaves us with the following inequality:

$$\tan(90° - 2\gamma) < 1. \tag{6.17}$$

Thus, $\overline{\mathbf{p}_1\mathbf{p}_3}$ exists in the *crust* of the sample points when $\theta < 45°$.

6.5.2 Angle of Multiple Materials

To empirically test the convergence of the 3D IO function angles towards $120°$, we generated a dataset that consists of a material in the shape of wedge that is then squeezed between two other

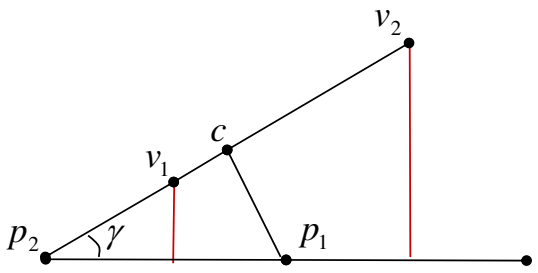


Figure 6.7. Shown here, the Voronoi vertex $\mathbf{v}_2$ will never be closer to the point $\mathbf{c}$ than $\mathbf{v}_1$.

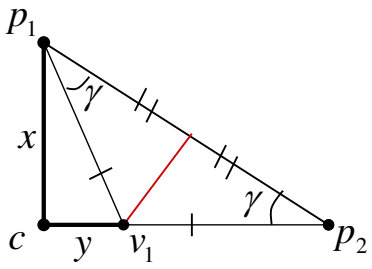
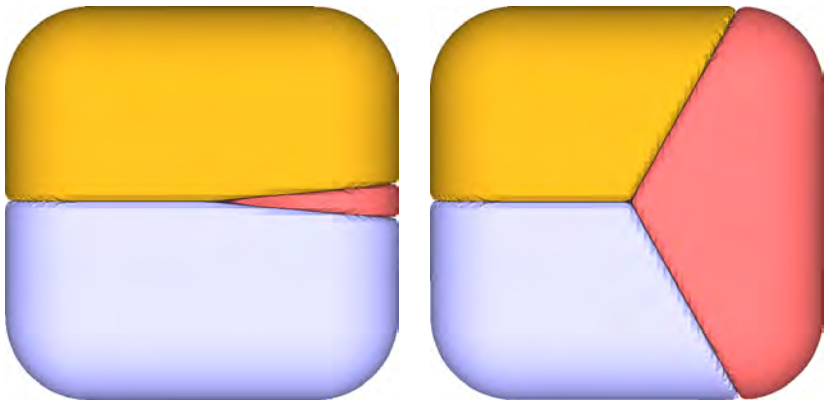


Figure 6.8. For $\theta < 45°$, the Voronoi vertex $\mathbf{v}_1$ will fall outside the smallest circumcircle of $\overline{\mathbf{p}_1\mathbf{p}_3}$.

materials inside of a box. By intersecting the reconstructed IO function of the wedge material with a circle oriented perpendicular to the sharp edge of the wedge, the angle of the material coming into the three-material boundary can be approximately computed. To do this, first analytic distance transforms of all three materials are computed, and the angle of the wedge is varied from 10° to 120°. Figure 6.9 shows isosurface extractions of the distance transform zero-sets of each material for the 10° and 120° data sets. Second, each data set is smoothed via tightening with a range of r values. We then reestablish the material boundaries with Equation 6.1 and place a point along the 3-junction. Around this point a circle of radius R is defined that is perpendicular to the 3-junction, and the two intersection points along the circle with the IO function of the wedge material are found using a root-finding scheme. Finally, to determine the angle of the wedge material near the three material junction the difference of the circle normals at the intersection points is computed.

In Figure 6.10 we present results of the experiments that vary the intersecting circle radius R, as well as the tightening radius r. The plot in Figure 6.10(a) shows the material angle for a range of wedge angles as the radius of the intersecting circle is decreased. The plot confirms that the material angle increases as the three material junction is approached, and that for even small wedge angles, the resulting IO function angles increase to values above 100°. Round-off errors prevent calculations any closer than 0.005 of the circle center (where 0.005 is in the units of the voxel width). The plot shown in Figure 6.10(b) shows the IO function angle at the smallest computed circle radius for a range of tightening radii. Again, this plot confirms that the reconstructed material angles tend toward values above 100° with numerical precision errors



(a) (b)

Figure 6.9. Isosurface extractions of analytic distance transforms of a wedge dataset of (a) 10°, and (b) 120°.

preventing accurate evaluations of the exact angle values near the three material junction.

We can show analytically that regardless of how small the initial angle of a material is going into a 3-junction, the smallest possible angle at a 3-junction point *after* the material has been tightened and then reconstructed as an IO function is $109.5°$. The geometry we are discussing is shown in Figure 6.11, where the untightened wedge with a material angle of θ is shown in blue, the tightened wedge is shown in green (where r is the tightening radius), and the reconstructed IO function for the wedge is shown in yellow. To compute the angle of the IO function material (yellow) we would like to characterize the polynomial that describes the red line and then find its slope at the 3-junction point tj.

We consider the worst-case scenario, *i.e.*, when the material angle θ goes to zero, to determine the lower-bound on the material angle of the reconstructed IO function, which is illustrated in Figure 6.12. Because the tightening algorithm results in a distance transform, we can compute the exact location of the triple-junction point tj. The tightened wedge is again shown in green with $r = 1$, while the tightened bottom and top materials are shown by the purple and blue hatch marks. The center of the tightening circle is at the location $(0, 1)$, which gives the following equation for the distance transform of the wedge material near this circle:

$$d_w(\mathbf{x}) = \sqrt{x^2 + y^2 - 2y} \tag{6.18}$$

The distance transform of the bottom material is given as

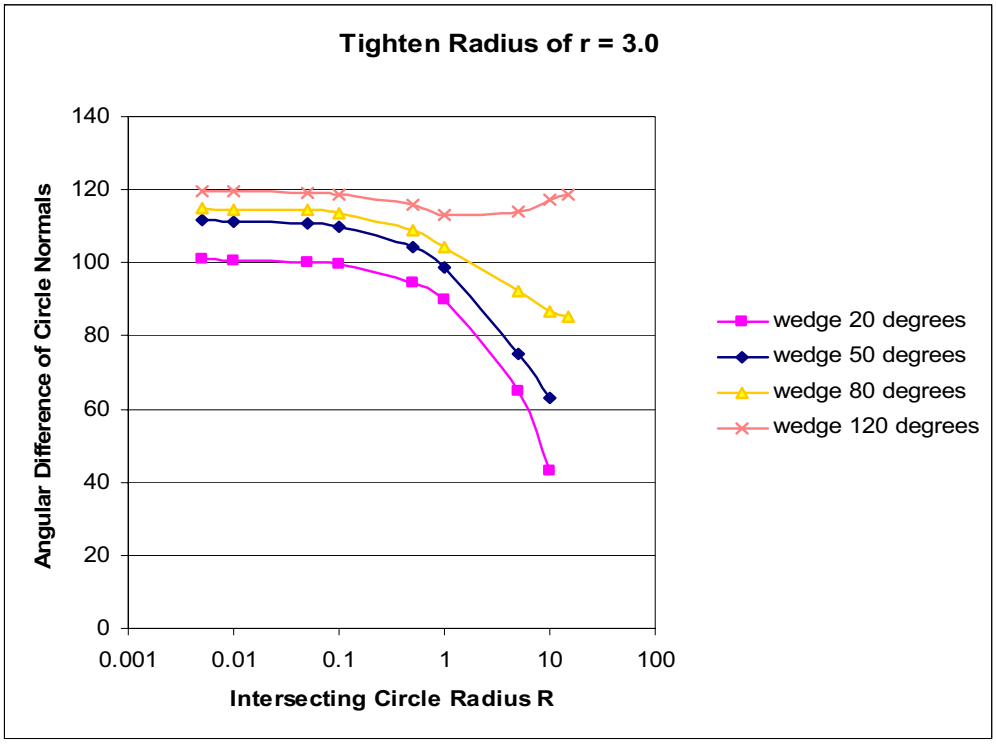
$$d_b(\mathbf{x}) = y \tag{6.19}$$

and for the top material as

$$d_t(\mathbf{x}) = 2 - y. \tag{6.20}$$

The 3-junction point will be located where $d_w(\mathbf{x}) = d_b(\mathbf{x}) = d_t(\mathbf{x})$, which is the location that the IO functions for all three materials will evaluate to zero. To determine this location, first the y coordinate is computed by setting $d_b(\mathbf{x}) = d_t(\mathbf{x})$, which results in $y = 1$. Then, $d_w(\mathbf{x}) = d_b(\mathbf{x})$ is solved with $y = 1$, leaving $x = \sqrt{2}$. Thus, the 3-junction point is at the location $(\sqrt{2}, 1)$.

Next, the polynomial of the line shown in red is determined by setting $d_w(\mathbf{x}) = d_b(\mathbf{x})$, giving the equation



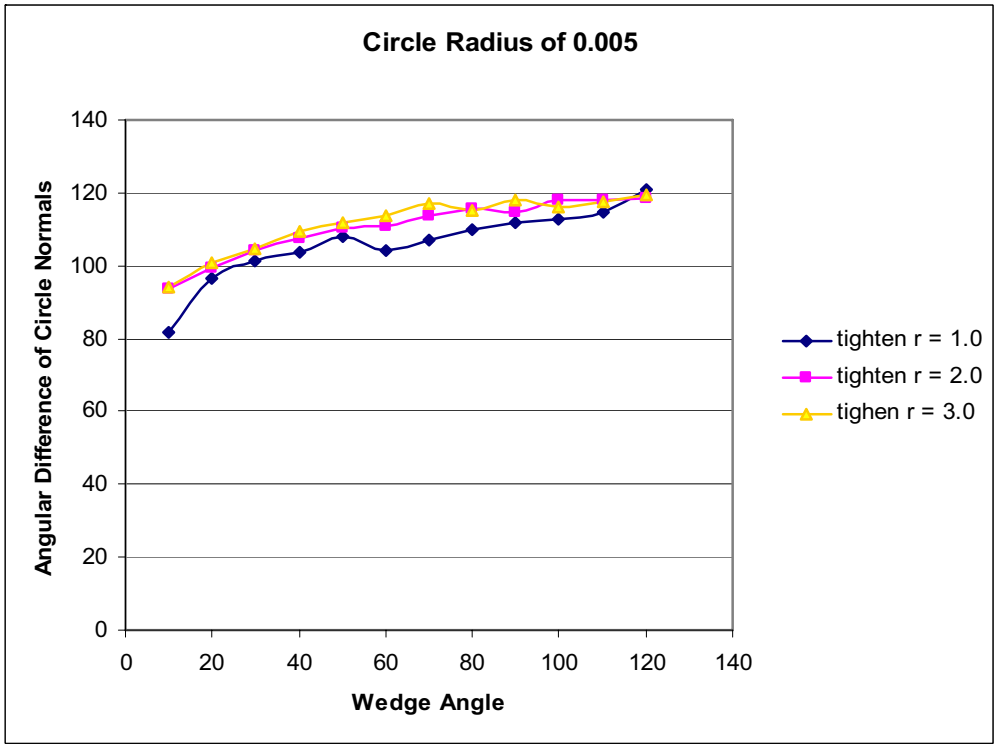


Figure 6.10. Plots of the wedge experiment that indicate the IO function angles increase as a three material point is approached: (*top*) as the intersecting circle radius R decreases, the material angles for a range of wedge angles increase to values over $100°$; (*bottom*) material angles computed at a small intersecting circle tend to values over $100°$, regardless of the amount of tightening used.

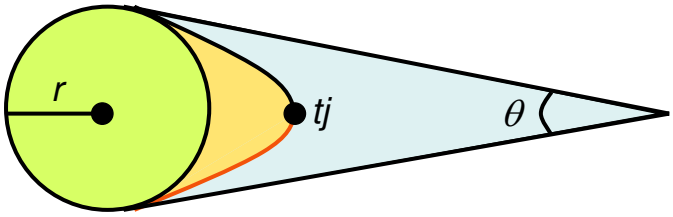


Figure 6.11. The geometry for the wedge data, where blue is the original wedge with a material angle of θ, green is the wedge tightened with a radius of r, and yellow is the reconstructed IO function.

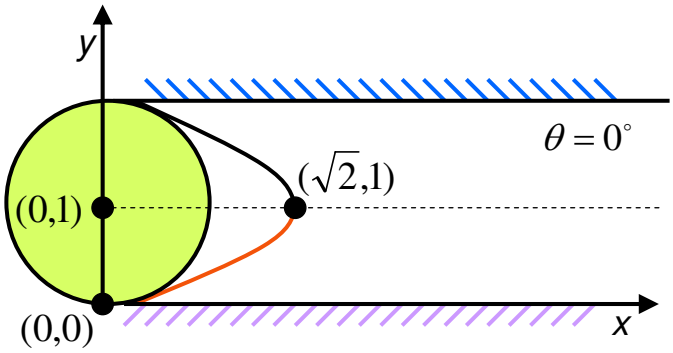


Figure 6.12. The geometry for the wedge data, where blue is the original wedge with a material angle of θ, green is the wedge tightened with a radius of r, and yellow is the reconstructed IO function.

$$y = \frac{1}{2}x^2. \tag{6.21}$$

The derivative of Equation 6.21 is $\dot{y} = x$, which is then evaluated at the 3-junction point to get the slope of the line tangent to the IO function of the wedge, $m = \sqrt{2}$. Computing $2\arctan(\sqrt{2})$ provides a lower-bound of the angle of the IO function for the wedge material as $109.471221°$.

6.6 System Overview

6.6.1 Preprocessing Data

Multimaterial data sets are often the result of a segmentation process applied to grayscale data from sources such as MRI or CT scanners. Automatic segmentation by a computer is a challenging research problem, and is thus often augmented with hand segmentations by a domain expert. For example, in the multiple material torso data segmentation shown in Figure 6.13 the heart material was hand segmented by a cardiologist, while the other materials were generated

semiautomatically using an expectation-maximization segmentation algorithm [117]. Hand segmentations can capture small features and thin regions that can sometimes be misclassified by a segmentation algorithm. These hand segmented materials, however, can often contain interslice artifacts where the material boundaries between slices are hard to distinguish. In Figure 6.14 (a) an isosurface of the hand segmented heart material shows some of these artifacts. Furthermore, these volumetric datasets are a discrete representation of some (presumably) continuous geometry, and the segmentation process makes binary decisions on what is, and is not, part of a specific material. Thus, the process of scanning and labeling is inherently lossy, but is necessary for generating 3D geometry from discrete grids. The resolution of specific simulations is often coarser than the resolution of the volume data, which further eliminates geometric information about the original, underlying shape. For these reasons, preprocessing of the volume data is necessary to eliminate small features and to reduce segmentation artifacts. This section will outline a variety of mechanisms that can be used to preprocess multiple material datasets.

Binary morphology can be used to reduce segmentation artifacts, as well as to eliminate spurious pieces of material and close small gaps. The elimination of these small features is important for generating geometry that reflects the resolution of the final simulation as efficiently as possible. As discussed in Section 2.5.1, there are numerous choices of morphology stencils, and there are the different morphology operations *opening* and *closing*. These operations are applied to each material after they have been isolated into individual volumes from the original, multilabel data set. For the torso results presented throughout this chapter, the stencils given in Table 2.1 were used to preprocess the individual materials, with the explicit morphology

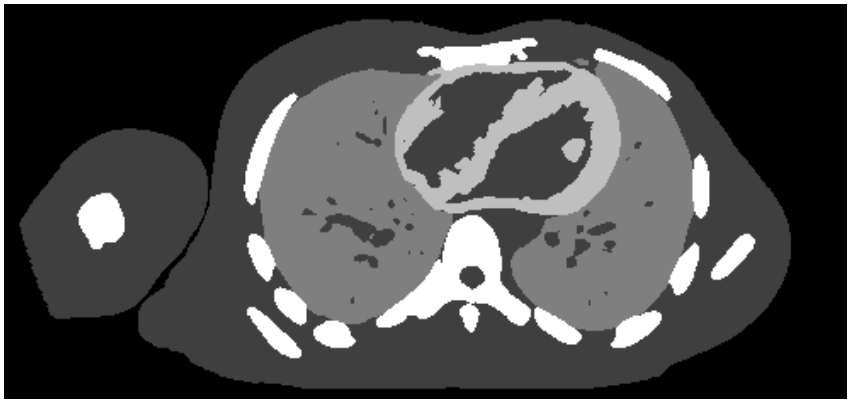
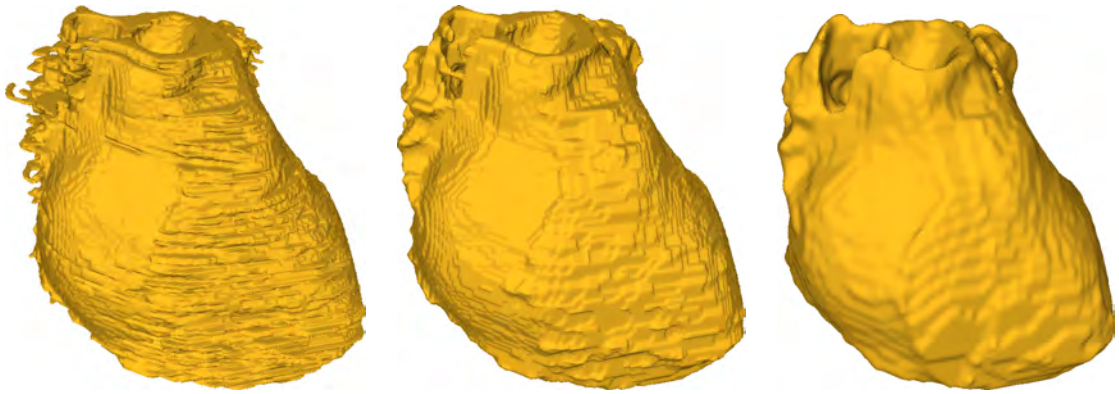


Figure 6.13. The original multilabel torso dataset, where segmentations indicate (from darkest to lightest) air, torso tissue, lung, heart, and bone. In this data set, the heart was handsegmented while all other materials were automatically segmented.



(a) (b) (c)

Figure 6.14. Segmentation artifacts show up in an isosurface extraction of the heart material (a). Binary morphology operations can eliminate many of the small features and gaps (b), while *tightening* smooths the surface by controlling the minimum feature size (c).

operations (and stencil size) listed in Table 6.1. In the isosurface extraction shown in Figure 6.14 (b), the segmentation artifacts that appear in Figure 6.14 (a) are greatly reduced after binary morphology operations are applied to the raw data.

For many types of biomedical applications, the numerical complexity of the simulations require a relatively course geometry compared with the resolution of the data coming from the scanning device. Thus, downsampling the data to meet the needs of the final simulation can reduce the size of the simulation geometry without incurring a loss of accuracy of the final result. The goal of the simulation which necessitated the generation of the torso data presented in this chapter is one such example. Thus each material was downsampled after the morphology operations were performed. One consequence of this downsampling is that empty voids may appear when the materials are concatenated back together, shown in Figure 6.15. To remedy this, a *voting* algorithm fills voids based on neighborhood information around empty nodes. For each grid node that is determined to be empty, the local 6-point neighborhood is searched, with the empty node taking on the value of the most prevalent neighborhood material value. When

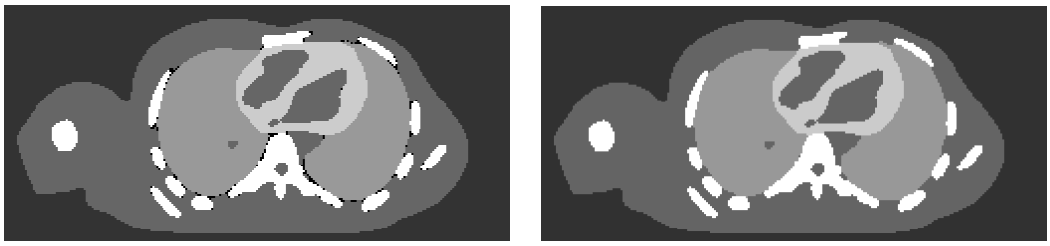
Table 6.1. The binary morphology operations used to preprocess that torso data presented in this chapter. The stencils used correspond to those presented in Table 2.1.

material	morphology operations	stencil size
air	close *then* open	4 *and* 2
torso tissue	close *then* open	4 *and* 2
bone	open *then* close	2 *and* 1
lung	close *then* open	4 *and* 1
heart	close *then* open	4 *and* 1

there is more than one prevalent value, a heuristic determines the value of the empty node — for results in this chapter, the smallest of the prevalent values is chosen. After the empty regions of the recombined data have been filled, the materials are again isolated into individual material volumes.

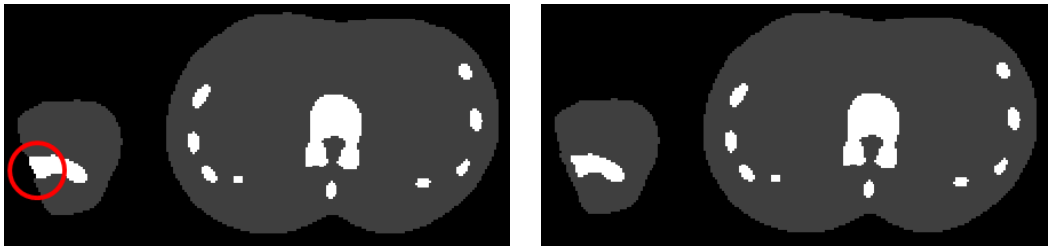
The segmentation, binary morphology, and downsampling process can sometimes produce physically implausible artifacts in the data. An example is shown in Figure 6.16, where the automatic segmentation algorithm failed to identify a region of torso material, resulting in an undesirable bone-air boundary. These types of artifacts are often due to the challenges of identifying and maintaining thin material boundaries throughout the preprocessing steps. By identifying implausible material boundaries in the segmented data, the data can be corrected by adding or removing material using a tool such as a 3D paint program.

Figure 6.17 presents the original and processed multilabel data for the individual materials for the torso data set. Although the processed materials shown in column (b) contain features of a size that correspond with the resolution of the motivating simulation, they are not adequate for



(a) (b)

Figure 6.15. The multilabel torso dataset after binary morphology is performed on the individual materials, followed by a downsampling. The materials are recombined in (a) where the black pixels indicate an empty region where no materials are specified. The downsampled data after *voting* (b) which applies heuristics to fill in empty regions.

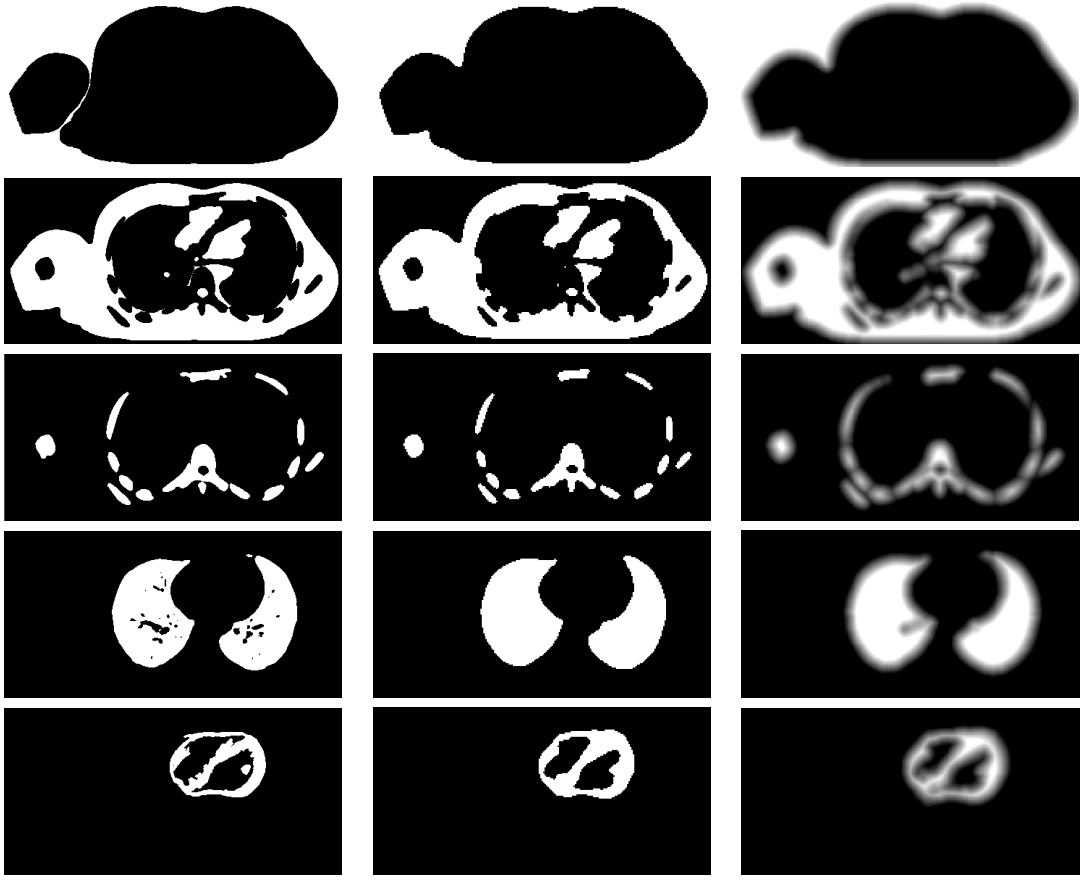


(a) (b)

Figure 6.16. Segmentation and/or binary morphology can create unrealistic material boundaries (a), such as the bone/air boundary circled in red. To remedy this, the segmentation can be fixed by hand (b).

generating high-quality geometry — these volumes must be smoothed to eliminate voxelization artifacts that would otherwise cause a stair-stepping effect in the final geometry. Using the curvature flow-based morphology operation *tightening*, described in Section 2.5.2, the individual materials are smoothed such that the resulting zeroset surfaces have a controlled minimum feature size. The results of smoothing each material with a tightening radius of $r = 1$ are shown in column (c) of Figure 6.17. Differentiable, smooth surfaces can then be extracted from these volumes using continuous reconstruction kernels.

In the process of tightening a surface to ensure a specific minimum radius of curvature, thin regions of the surface can be altered drastically by the algorithm. An example of this is shown in Figure 6.18, where a thin wall of the heart is noticeably eroded during tightening. To remedy this



(a) (b) (c)

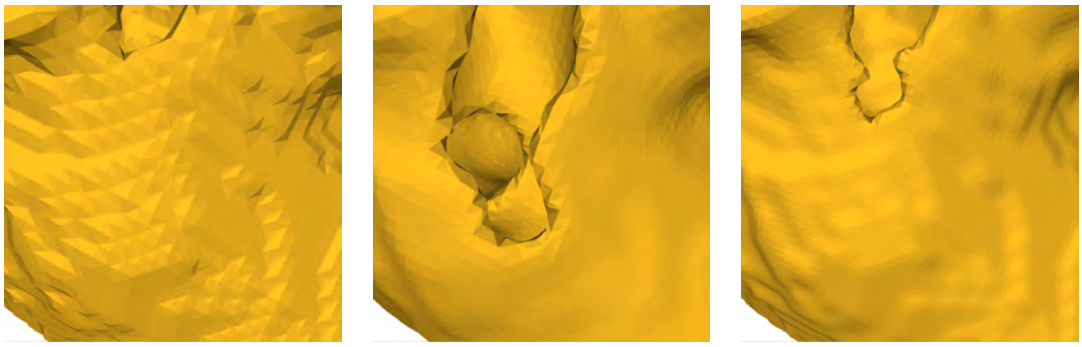
Figure 6.17. The torso data set at various stages of preprocessing, where white indicates material and black indicates nonmaterial. The materials from top to bottom are the outside, torso tissue, bone, lung, and heart: (a) the original data after each material have been isolated from the multilabeled volume; (b) the downsampled materials after performing binary morphology on each material; (c) the final, tightened materials which become the input *indicator functions*.

undesirable effect, a smaller tightening radius can be used, at the cost of less smoothing of the overall surface. However, due to the numerics of the levelset framework in which the tightening algorithm is implemented, subvoxel tightening is not possible. Instead, the material volume can be upsampled to a higher resolution grid, over which a smaller tightening radius can then be applied. For the torso results presented in this chapter, the heart material resides over a higher resolution grid then the other materials, and was tightened with $r = 0.6$ (where the units of r are given in the units of the courser grid).

The output of the algorithm is a grayscale volume that stores the signed distance to a tightened material surface, where positive values indicate material. We reconstruct continuous, differentiable f_i from the tightened volumes of each material using *separable convolution*, which convolves a 1D continuous kernel with grid points along each separate axis of a volume (see Section 2.4). For the results presented in this chapter, we use an interpolating 4^3 Catmull-Rom spline as the continuous kernel. The reconstructed implicit functions are then input to the system as the set of indicator functions F.

6.6.2 Distributing Particles on Junctions

The proposed system builds from the particle system framework from Chapter 5 for placing points along each material junction. This framework uses a *sizing field* that informs particles of how far they should be from their neighbors to meet LFS sampling requirements.A sizing field volume for a multimaterial dataset is generated by computing the LFS of each IO function (*i.e.*, the $\tilde{f}_i$). At each grid point in the sizing field volume the minimum LFS for the set evaluated at the



(a) (b) (c)

Figure 6.18. When materials have thin regions, *tightening* can remove a large amount of material to obtain the required minimum radius of curvature. In (a), an isosurface of the binary heart volume is shown for a thin region of the material. After tightening with $r = 1$ (b), a large portion of the heart wall is eroded. In these situations it can be useful to tighten the material at a higher resolution (c) with a smaller tightening radius ($r = 0.6$ in this image).

grid point location is stored. Along sharp features, however, the LFS will go to zero, causing an infinite sampling requirement. Because we are explicit sampling these sharp features, the strict LFS requirements near 0-cells and 1-cells are violated by placing a lower bound on the sizing field. This lower bound is determined by the tightening radius r (see the previous section) that drives the tightening algorithm when smoothing the material volumes. This is (from analytical results in 2D, and empirically in 3D) a good estimate of the size of the ball within which the angle constraint (from Section 6.4.3) holds. For surfaces in 3D, the preprocessing (tightening) does not guarantee a lower bound on the principle curvatures in hyperbolic regions, which could lead to problems in obtaining sampling densities that ensure the angle-constraint near 0- and 1-cells. In practice, this appears to be very rare, and we have not observed this problem in the results shown in this chapter, despite the complexity of the datasets.

Using an ordered sampling scheme for distributing sets of particle systems, the system first samples each 0-cell with a single particle, which remains fixed in place. Next, the system samples 1-cells with particle systems that interact with these 0-cell particles. Similarly, the 1-cell particles are allowed to converge to a steady state and are then fixed in place. Finally, particles are distributed on the various 2-cells that interact with 0- and 1-cell particles.

6.6.3 Meshing the Surface

We use Tetgen[1] to generate a tetrahedralization of the convex hull of the set of particles. Each tetrahedral element is labeled according to the material in which its circumscribing sphere center lies. The watertight, nonmanifold mesh of the material interfaces is the subset of faces that are bounded by tetrahedra of different material types. Faces that lie on the convex hull are labeled as having a second bounding tetrahedra of the outside material type. This nonmanifold mesh can then be used to generate volume filling elements that conform to the LFS of the boundary (see Section 6.7). Manifold meshes of each individual material can also be extracted in a similar fashion, with the shared boundaries of surface meshes having consistent triangulations by construction.

We have experimented with other Delaunay-based surface reconstruction algorithms, such as TightCocone [44], that are designed to infer topology from an organized set of points without knowledge of the underlying surface/solid. Our experiments showed these methods can sometimes fail, and thus we advocate the use of a simpler labeling algorithm, such as the one described in this paper, that includes information about the underlying multimaterial volume

[1] tetgen.berlios.de

which gaurantees conformal, watertight surfaces. In Figure 6.19 we present an example of a poor triangluation from TIGHT COCONE that is correctly meshed in Figure 6.20 using the proposed labeling algorithm.

6.6.4 Implementation

This particle system implementation is an extension of the version found in Section 3.6. Changes to the basic particle system implementation are in **bold**, and free parameters introduced in this chapter are presented in Tables 6.6.4 and 6.6.4, where we also present the free parameters from Chapter 3 that were used in the multimaterial implementation. Note that almost all of these

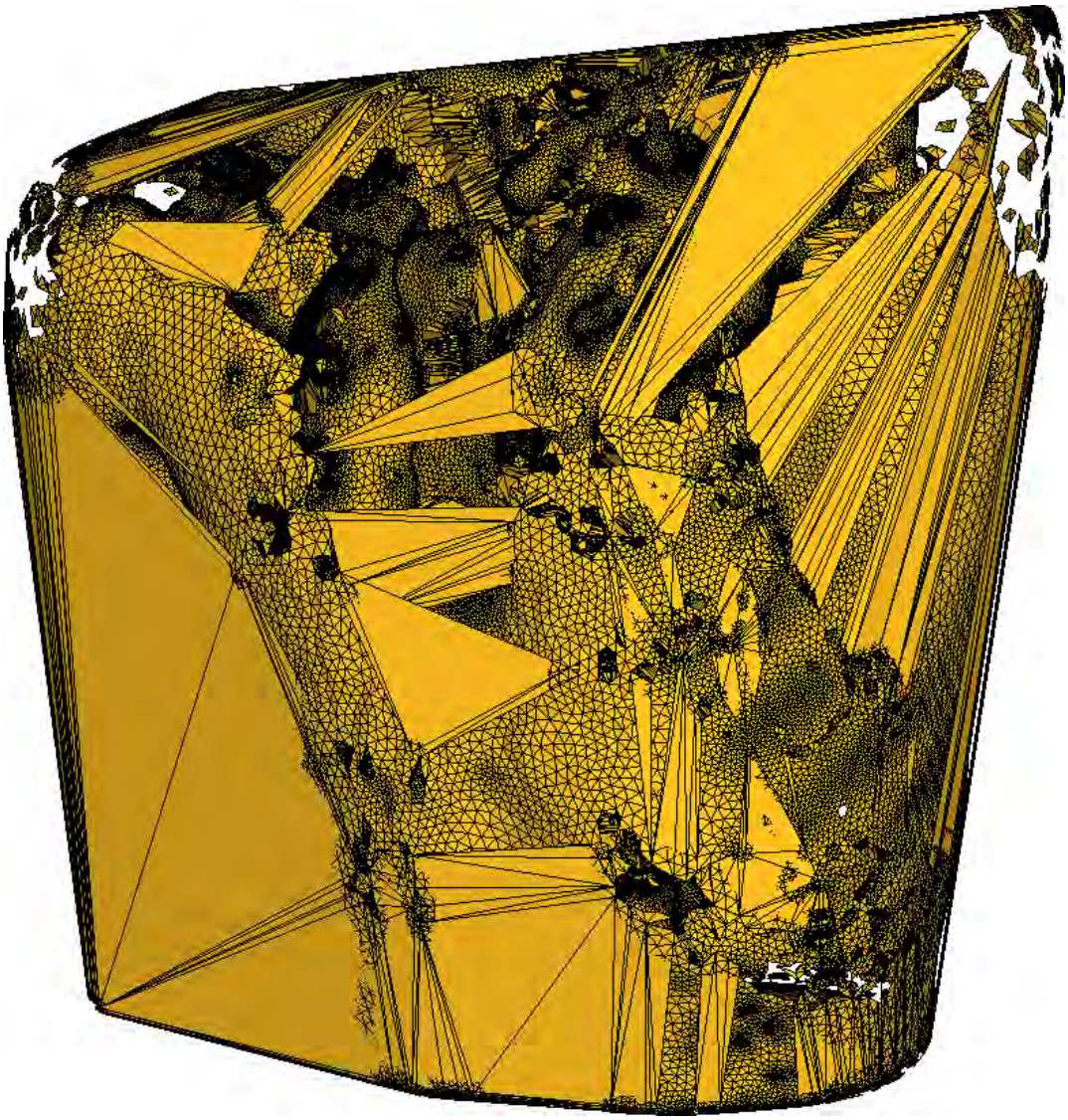


Figure 6.19. A poor triangulation from TIGHT COCONE of the torso tissue material.

parameters use the same values as previously discussed implementations.

Prior to distributing the particles, the segmented, multilabel volumes under go significant pre-processing to produce smooth surfaces that are designed specifically for the needs of a particular application. First, each material is isolated from the original volume and smoothed using binary morphology operations and any other necessary segmentation enhancements. Next, appropriate resolutions are selected for each material, and the materials undergo tightening to generate smooth representations — interpolating Catmull-Rom spline kernels (discussed in Section 2.4 are then used to reconstruct continuous represents of the tightened materials for the remainder of the pipeline. The tightening is followed by a generation of sizing fields for each IO functional representation of the materials, described in Section 5.3.2, with $\epsilon = 0.5$, $\delta = 0.4$, and $h_0^{\min} = \epsilon r$ where r is the tightening radius. Then, marching cubes meshes are generated for each IO function, the vertices of which are used to initialize the particle systems. Finally, all the possible combinations of materials are computed and initialized with a marching cubes mesh for one of the junction materials by projecting the vertices onto the junction for 10 iterations. Any particle system with zero particles on the junction after the initial projections is considered to be nonexistent in the data and removed from the set. The remaining systems are distributed using the following steps:

1. **For each quad-junction, cull all but one particle for each surface point location.**

2. **For each remaining junction, starting with the triple-junctions followed by the double-junctions:**

   (a) **Add all $(n+1)$-junctions and $(n+2)$-junctions, if they exist, to the particle system's neighborhood, where n is the number of materials at this particle system's junction.**

   (b) For each particle:

      i. Compute E_i, and compute $\mathbf{v}_i$ with Equation 3.4, **where the interparticle distances are scaled by the α_{ij} given in Equation 5.5, using the minimum $h(\mathbf{x}_i)$ value over all of the materials' sizing fields.**

      ii. Compute $\mathbf{v}_i^{\text{new}} = \lambda_i \mathbf{v}_i$.

      iii. Compute the new particle position $\mathbf{x}_i^{\text{new}}$ by solving Equation 3.11 **using the projections operators defined in Section 6.4.2** with $\mathbf{v}_i^{\text{new}}$, followed by a reprojection to the surface by solving Equation 3.12.

iv. Compute the new energy value, E_i^{new}, at the new particle location $\mathbf{x}_i^{\text{new}}$, as well as the new implicit function value, F_i^{new}.

v. If $E_i^{\text{new}} \geq E_i$ or $|\mathbf{x}_i - \mathbf{x}_i^{\text{new}}| > s$ or $F_i^{\text{new}} > \epsilon_T$, and $\lambda_i > \lambda_{\min}$, decrease λ_i by a factor of 10 and go back to Step 2(b)ii. If $\lambda_i \leq \lambda_{\min}$, do not move the particle and skip to the next particle in the list.

vi. If $E_i^{\text{new}} < E_i$, update $\mathbf{x}_i = \mathbf{x}_i^{\text{new}}$. Increase λ_i by a factor of 10 if this is the first time through Step 2b.

(c) Decide whether the system is at a steady state. There are numerous metrics to determine steady state, and we have chosen to use the difference of the system energy (the sum of the energy at all the particles) from one iteration to the next. When the system energy difference is less than a small fraction of the total energy (we use 0.15% for the results presented in this dissertation, although a range of values would produce similar results), we deduce that particles have reached a steady state. Otherwise, repeat Step 2b. For poor initializations, we have found it can be useful to skip to Step 2d every 50 iterations instead of waiting for the system to reach a steady state.

(d) Check whether the configuration of particles is desirable. We compare each particle's energy against an ideal energy, E^{ideal}, **which is defined in Section 3.5.1 with a value of $n = 2$ for triple-junctions, and $n = 6$ for double-junctions**. We bias E_i with a random value on the interval $[0, 1]$ to eliminate mass splitting or dying, then split particles with $E_i < 0.35E^{\text{ideal}}$, and delete particles with $E_i > 1.75E^{\text{ideal}}$ **for double-junctions. Or, for triple-junctions, split particles with $E_i < 0.25E^{\text{ideal}}$, and delete particles with $E_i > 2.5E^{\text{ideal}}$.**

(e) If the energy of the particles is acceptable, stop iterating.

(f) **Write out the particle positions for input to a meshing algorithm.**

6.7 Results

We present results from several real-world datasets generated from MRI scans. The dimensions of each dataset are given in Table 6.7, along with the number of materials and sample points. We also provide the ϵ and δ values used to smooth the sizing fields, and note that these values were constant over all of the data sets. The torso and brain datasets were sampled on a P4 3.2GHz CPU with 2GB of memory in approximately 12 hours and 3 hours, respectively. The

Table 6.2. Table of free parameters.

Parameter	Value	Description	Comments
$\epsilon_{\mathbf{r}}$	10^{-7}	added to interparticle distances to avoid infinite energy values	system is insensitive to values near machine precision
σ	1.0	effective particle radius	constant for all results in this dissertation
λ_0	1.0	initial stepsize value	system is insensitive to this value
λ_{min}	10^{-14}	minimum stepsize value	system is insensitive to this value as long as it is sufficiently small
ϵ_F	$10^{-4} - -10^{-5}$	surface threshold	values of 10^{-4} were used for 4-junctions, otherwise, values 10^{-5} were used
	5 – 50	number of initial projections of a particle	system is insensitive to this value as long as particles get to within ϵ_F of the surface, however, more iterations were used for increasing numbers of interacting materials
	0.15%	system energy difference from previous iteration that indicates a steady state	value must be small enough such that the particle distribution converges to an even packing
	50	number of iterations when system automatically checks for a desirable configuration (Step 2d)	value must be large enough such that local particle neighbors can be established
	0.35	percentage of E_{ideal} that indicates a particle should be split	values with approximately 20% of this value produce visually similiar results with different convergence times

Table 6.3. Table of free parameters, *cont.*

Parameter	Value	Description	Comments
	1.75	percentage of E_{ideal} that indicates a particle should be deleted	values with approximately 20% of this value produce visually similiar results with different convergence times
	5	number of iterations when particle neighbor lists are updated	values < 10 maintain stability in the system
r	user-defined	tightening radius	value should be chosen based on application specific needs
h_0^{min}	ϵr	minimum sizing field value	other than hyperbolic regions, this is the minimum feature size after tightening, which can be lowered to reduce the incidence of mistriangulations at the expense of more (and smaller) triangles
	10	number of initial particle projections	must be large enough to ensure that the initial set of particles get to within ϵ_F of the junction surface

Table 6.4. The dimensions and number of materials of each dataset, the $epsilon$ and δ values used to smooth the sizing fields, and the number of particles used to sample the material junctions.

Dataset (# Materials)	Source	Volume Dimensions	ϵ, δ	# Particles
torso (5)	MRI	260x121x169	0.5, 0.4	394k
frog (5)	MRI	260x245x150	0.5, 0.4	186k
low-res frog (5)	MRI	160x151x94	0.5, 0.4	31k
brain (3)	MRI	149x188x148	0.5, 0.4	161k
two spheres (3)	synthetic	128x128x128	0.5, 0.4	1214

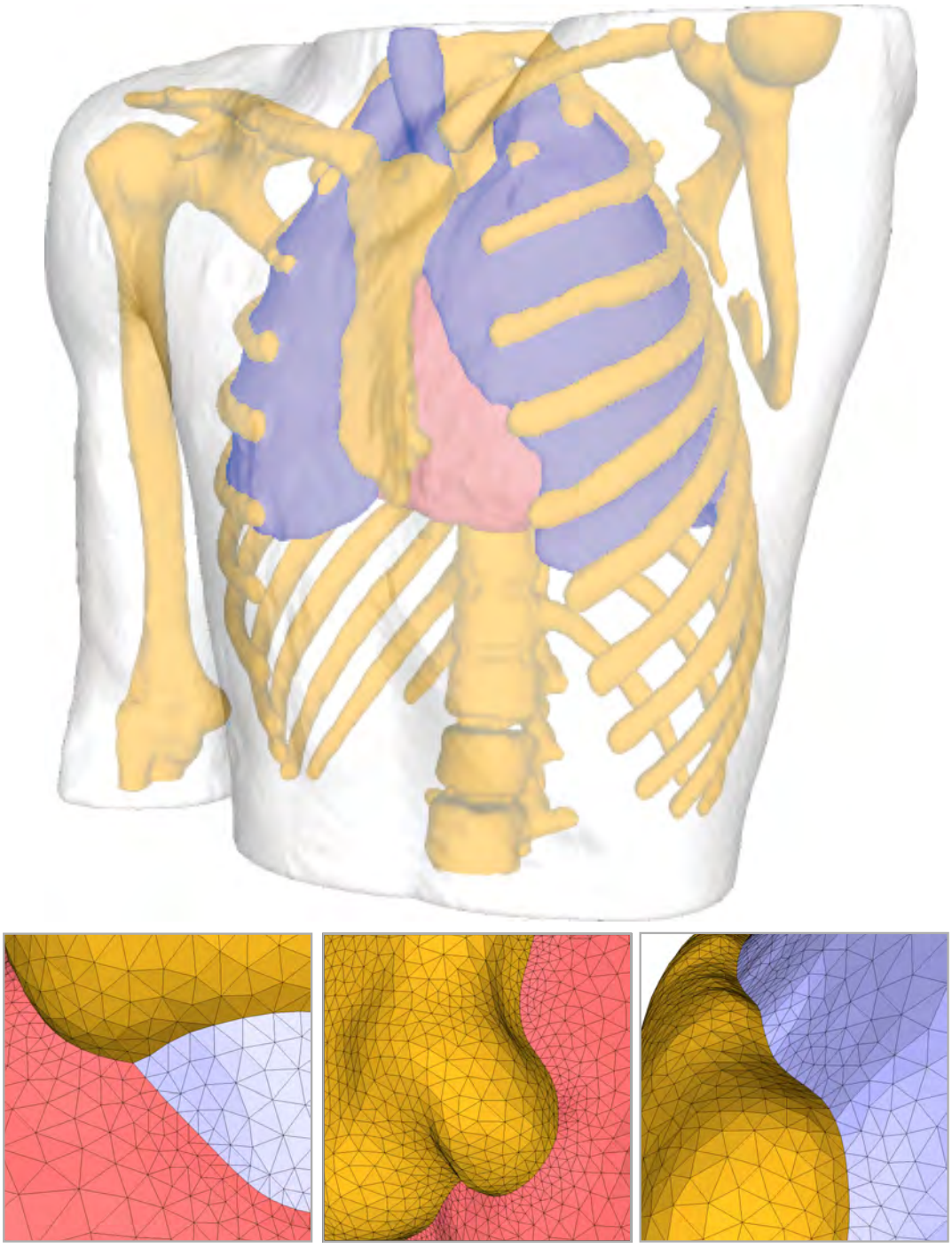


Figure 6.20. Multimaterial surfaces of a torso extracted from an MRI scan, with closeups of meshes generated using dynamic particle systems.

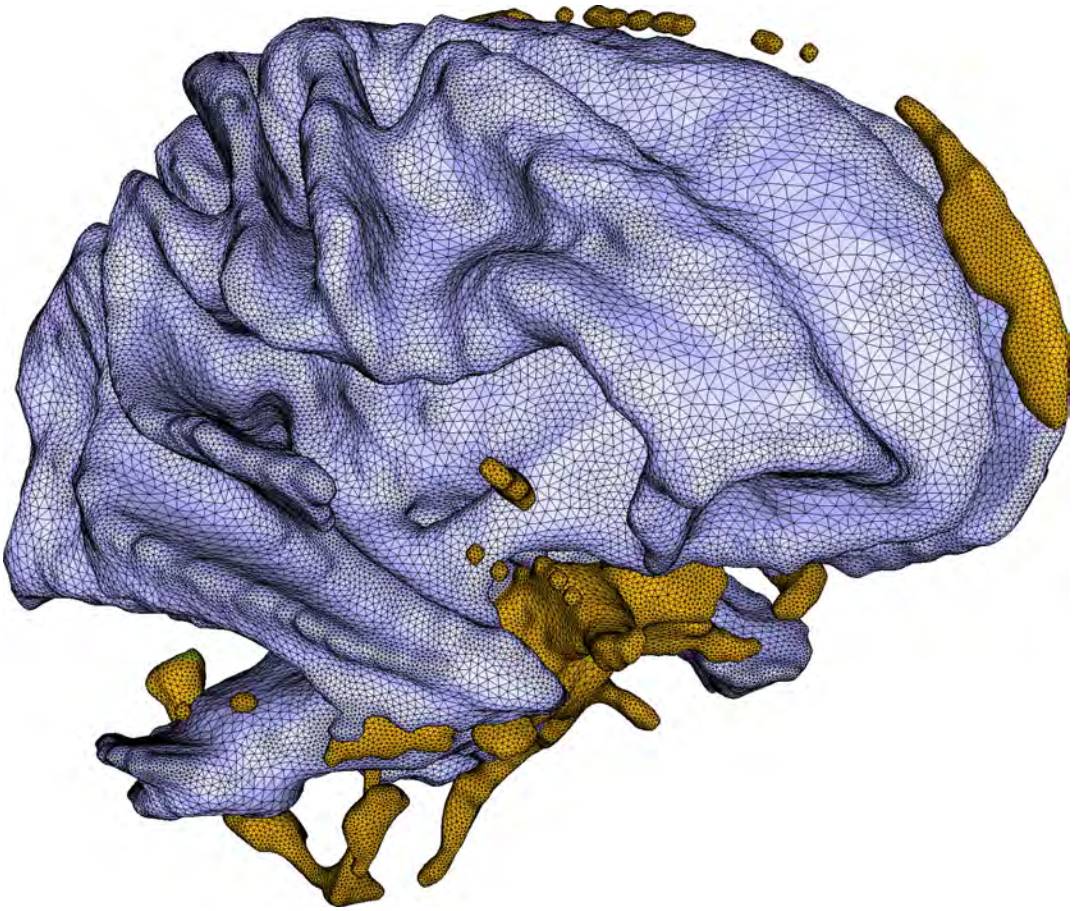


Figure 6.21. Meshes of the white matter and cerebral spinal fluid (CSF) of a brain dataset generated from an MRI scan.

frog, low-res frog, and spheres datasets were sampled on a laptop with a Celeron 1.4GHz CPU and 1GB of memory in approximately 5 hours, 1 hour, and 4 minutes, respectively.

A driving application for this work is the simulation of cardiac defibrillation in children. The goal is to generate a pipeline that will acquire a MRI scan of a child, generate patient-specific geometry from the scanned data, and to then determine an ideal placement for a cardiac defibrillator through FEM simulation. The torso dataset shown in Figure 6.20 was generated from a segmented MRI volume in this study, and consists of five materials: the torso tissue, bone, lung, heart, and air. Although other materials exist in the original MRI scan, decisions on which to include in the final simulation must be made to keep the number of elements manageable. For example, including the thin layers of fluid that exist between different organs would induce excessive numbers of elements as the feature size of this material layer is very small. Both the brain and frog datasets shown in Figures 6.21 and 6.22 were also generated from MRI scans that had been segmented

into multilabel volumes. The low-resolution frog in Figure 6.23 was downsampled from the segmented frog dataset in Figure 6.22 and then processed using morphology to eliminate small features. The synthetic two-sphere example in Figure 6.24 was generated over a grid from the difference of two analytically represented spheres.

For many FEM simulations used in biomedical computing, the *condition number*, *i.e.*, , the value that describes how numerically well-behaved a simulation will be, is directly related to the most poorly-shaped element in a tessellation. A metric that is commonly used to quantify the quality of surface meshes for FEM is the ratio of the inscribed circle to the circumscribing circle of a triangle, $2r_{in}/r_{circ}$. A ratio of one indicates an equilateral triangle, and a ratio of zero indicates a triangle that has collapsed down to an edge. In Table 6.7 we present statics for the tessellations generated with the proposed method, including the ratio of the most poorly shaped triangle for each mesh. These statistics indicate that not only are the bulk of the triangles nearly regular, but also that the worst shaped triangle is of consistently high-quality. This latter result is important for eliminating the time-consuming, and common, chore of hand tweaking mesh elements to make them suitable for simulations.

The results from the proposed system are compared against a grid-based multimaterial mesh-

Table 6.5. Statistics about each mesh and their quality.

Material	Number of Triangles	Min/Avg Radius Ratio
torso tissue	673k	0.39/0.94
torso bone	460k	0.31/0.94
torso lung	215k	0.32/0.93
torso heart	140k	0.38/0.93
frog tissue	367k	0.30/0.94
frog bone	197k	0.37/0.94
frog guts	46k	0.30/0.94
frog brain	8k	0.52/0.94
low-res frog tissue	202k	0.39/0.94
low-res frog bone	114k	0.39/0.94
low-res frog guts	25k	0.49/0.94
low-res frog brain	5k	0.52/0.93
brain white matter	255k	0.39/0.94
brain csf	92k	0.42/0.94
spheres top	1544	0.51/0.92
spheres bottom	1506	0.52/0.92

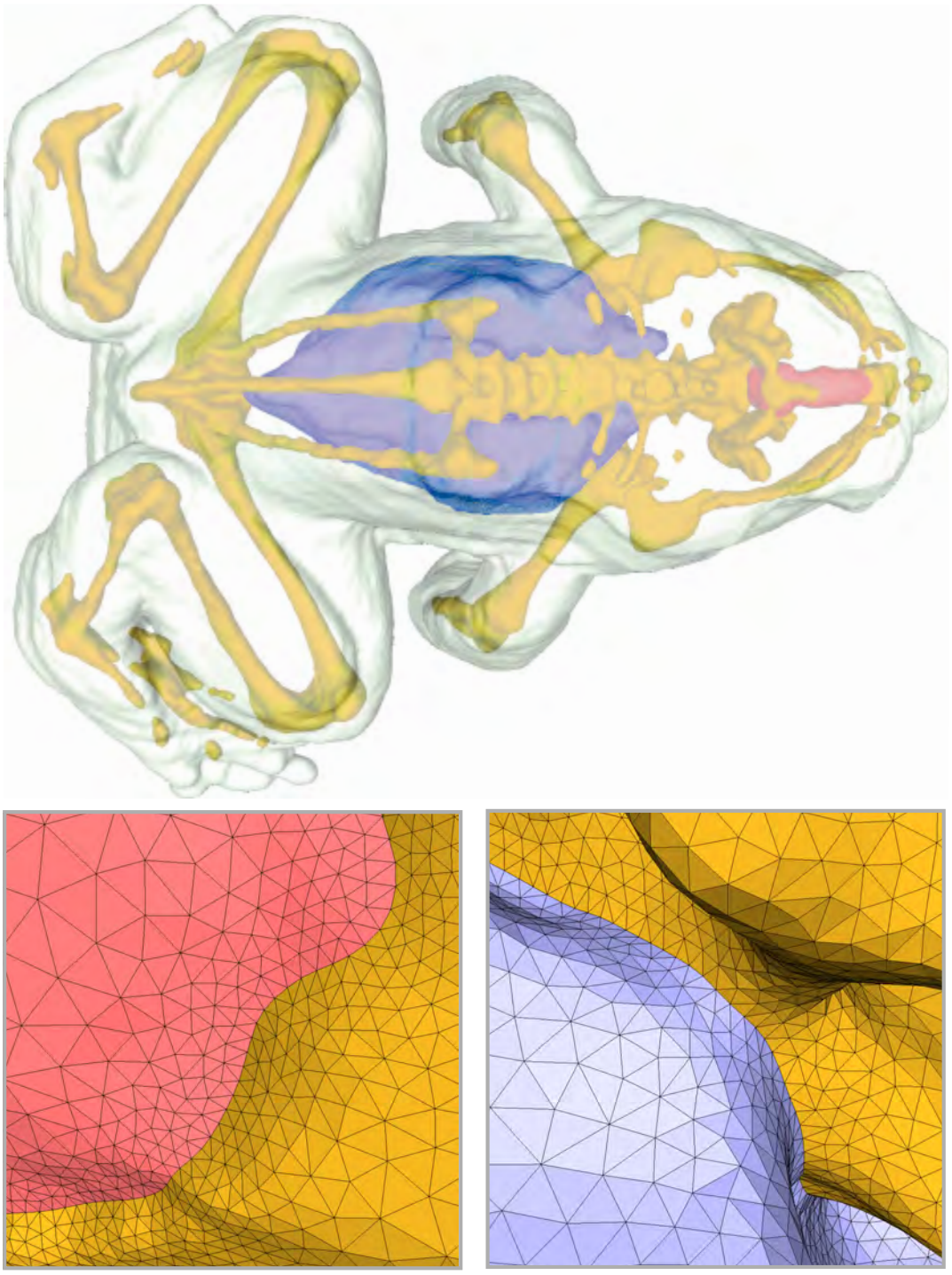


Figure 6.22. Meshes of the frog dataset generated from an MRI scan.

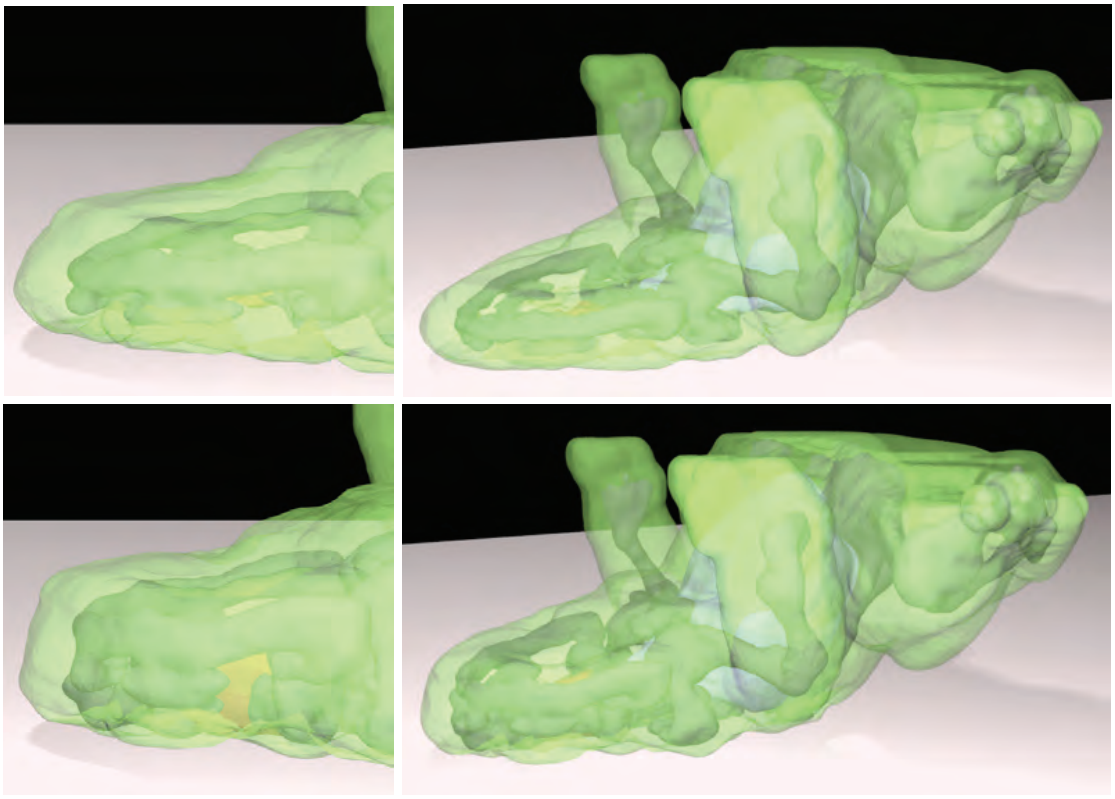


Figure 6.23. Screen shots from a point-based physics simulation. In the top row, all of the materials were assigned the same material properties, while in the bottom row, the bones and internal organs were assigned stiffer properties.

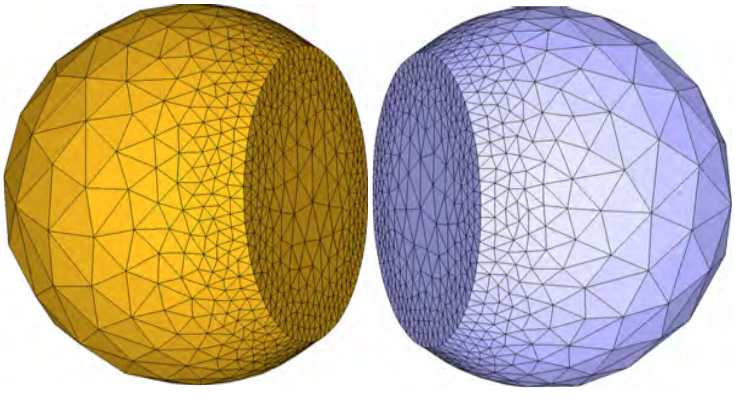


Figure 6.24. A synthetic example of two intersecting spheres, illustrating the consistency of the meshes along the shared boundary.

ing scheme using the VTK software[2]. Common to these approaches is a pipeline that first extracts a nonmanifold mesh from a discrete, multilabel volume, followed by a smoothing step to eliminate voxelization artifacts, and finally a decimation of the mesh to decrease the number of triangles [12, 46]. Our implementation uses the *vtkDiscreteMarchingCubes* class to extract a mesh of the interfaces, the *vtkWindowedSincPolyDataFilter* to smooth the voxelization artifacts, and the *vtkQuadricDecimation* to reduce the number of triangles. We generated meshes using this pipeline of the two-sphere and frog examples with approximately the same number of triangles as the analogous particle system-based examples. The minimum and average radius ratios for the spheres are 0.014 and 0.79, respectively, and for the frog 0.0 and 0.79, respectively. Not only is the quality of these meshes significantly lower than for our particle system-based meshes, but the size of the triangles does not adapt to the underlying geometry. Adaptive triangulations are important for efficiently capturing the geometry of an object with as few elements as possible. We present a visual comparison of these results in Figure 6.25.

The particle-based method is also well-suited for generating volumetric samples of multimaterial datasets. We have extended the particle system framework for packing spheres inside of sampled multimaterial interfaces. The spheres are distributed using the same sizing field that guides the surface samples, which is smoothed away from the surface such that larger values are on the inside of materials. The spheres are distributed as an additional step at the end of the ordered distribution process. These volume samples can be used to generate tetrahedral meshes that conform to the material interfaces and respect the LFS of the boundaries (see Chapter 7), as well as for point-based physics simulations. In these simulations, research has shown that both physical and geometric complexity are highly correlated [2], and that the stability and accuracy of the simulation is directly related to the ability of the surface and volume samples to capture the LFS of the material boundaries.

To test our results, we have implemented a point-based physics algorithm that simulates elastic materials [97] , extending the algorithm to handle multimaterial objects by assigning different physical properties to volume samples of different material types. Surface meshes generated from the particles sampling the multimaterial interfaces are used to include collision detection between materials of different types [75], as well as for rendering the simulation results. In Figure 6.23 we present screenshots from a simulation of the frog dropping onto a flat surface. In Figure 6.23 (*top row*) all volume samples were assigned the same material type, while in Figure 6.23 (*bottom row*), volume samples in the bones and internal organs were assigned stiffer material

[2] vtk.org

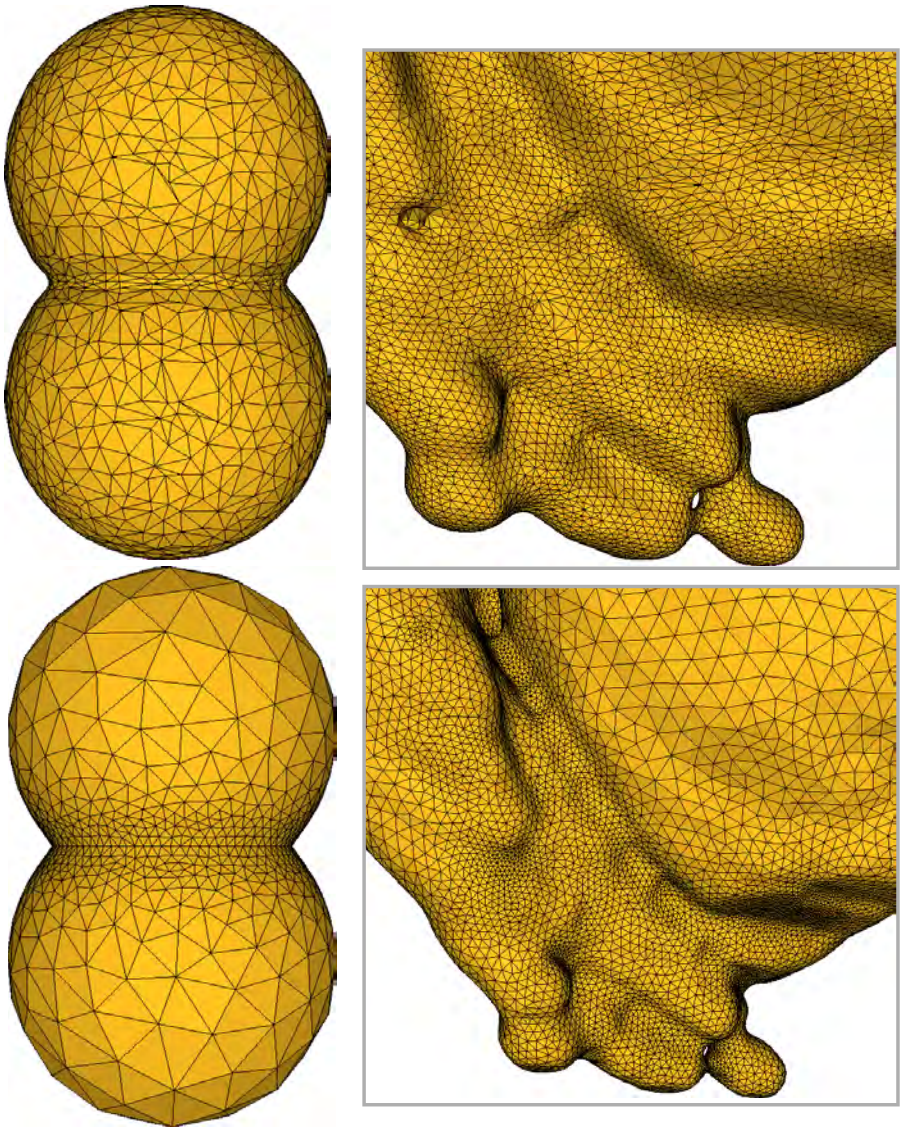


Figure 6.25. Comparisons of multimaterial meshes generated using a grid-based approach (top) and our particle system-based approach (bottom). The left column is the two-sphere example, and the right column is a closeup from the frog example.

properties than the surrounding frog tissue. By assigning different properties to each material in the frog, more complex, and realistic simulations can be achieved, including stiffness within the head and body due to the rigid bone structure. This example illustrates not only the importance of modeling multiple materials of objects for increasingly realistic simulations, but also the potential for automatically generating complex digital models from scanned, real-world objects.

6.8 Discussion

The high-quality results of the proposed method come at the cost of long computation times. Most of that time is spent in preprocessing the multilabel data and distributing the sets of particles. The number of materials in the dataset also adds to the overall computation time as the $\max$ function must evaluate every f_i that exists in the set. However, the quality of the meshes from our particle-based scheme is so high that we can avoid the usually time-consuming step of meticulous hand-editing of mesh vertices to ensure well-shaped elements.

CHAPTER 7

EXTENSIONS AND FUTURE WORK

During the course of this dissertation work we experimented with numerous extensions and applications of the particle system framework. In this section we will present some preliminary results of a few of these ideas, and provide some future directions for research.

The low-energy hexagonal packings of the particle systems' distributions are mirrored in the natural world, such as those found in many simple crystalline structures. There are also low-energy states in crystals that form quadrilateral patterns, and we were intrigued with the idea of finding adaptivity metrics that would cause the particles to settle into quadrilateral packings. In nature, quad-packings often occur when different materials interact with each other. Thus, we modified the particle system framework to accommodate particles of different types by differentiating between the adaptivity metric for like-particle interactions and that for unlike-particles. Specifically, the α_{ij} in Equation 3.14 is multiplied by $\sqrt{2}$ for interactions of like-particles, while remaining unchanged for interactions between unlike-particles. In Figure 7.1 a quad-packing of particles is shown that was generated using this scheme.

A system with different classifications of particles is sensitive to a uniform heterogeneity of the particle types. We found that the initialization of the system such that the particles are relatively uniformly mixed helped to keep regions of like-particles from forming, which would otherwise create mostly hexagonal packings of points. We are interested in looking into effective techniques for initializing the particles, as well as for maintaining uniformly heterogeneous distributions throughout the convergence process. Well-distributed sets of quad-packings could be used to create quad-meshes through a Delaunay triangulation and mesh refinement scheme that removes all the edges associated with one of the particle types.

The particle system framework is naturally extendible to any dimension. As such, we have experimented with representing particles as volumetric spheres, and packing the spherical particles into a volume. The natural low-energy state of the system occurs when each sphere has approximately 12 neighbors, forming a regular dodecahedron shape [58]. Thus, the particle system framework is modified to drive the system towards this neighborhood density by changing

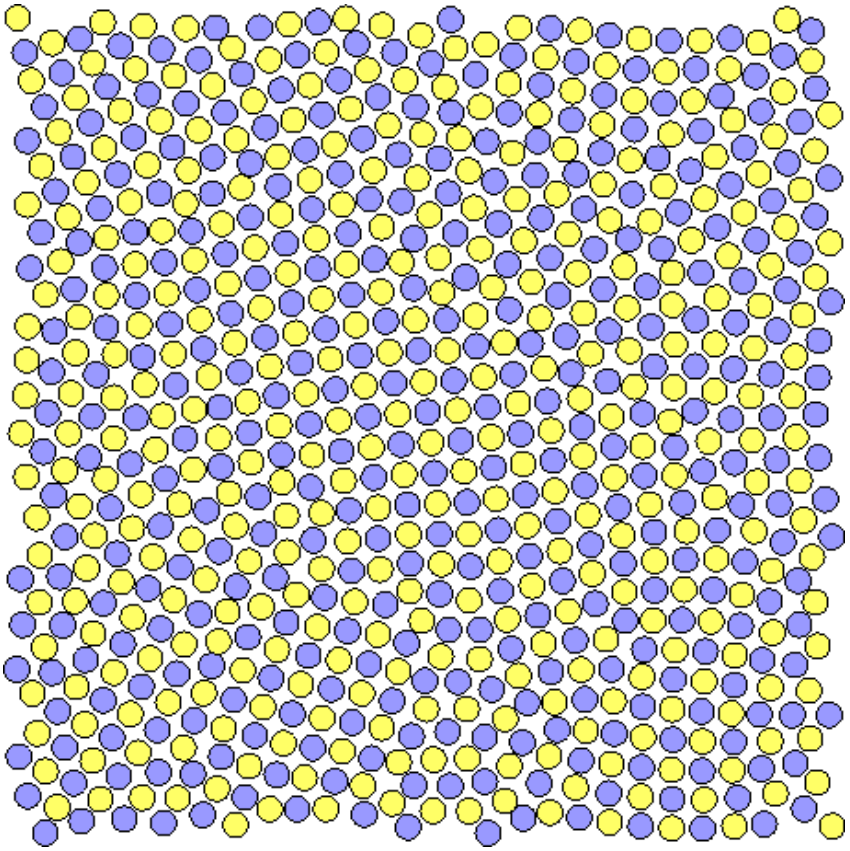


Figure 7.1. A particle system that generates quad-packings of points.

the value of n in the ideal energy measure nE_{ideal} from 6 to 12, and also eliminating the surface constraint (*i.e.*, the projection of a particle's motion vector).

Specifically, we are interested in generating volumetric, boundary conforming meshes that are well-suited for simulations. Using the volumetric particle system, spheres are packed inside of a volume after the boundary of the volume is sampled with a surface particle system. The spherical particles interact with the frozen surface samples (which are also represented as spheres) and distributed inside of the sampled boundary — Figure 7.2 shows the surface samples for the heart material from Chapter 6, along with the spherical particles packed inside. For well-shaped tetrahedral elements, the volumetric sampling should be related to the LFS near the surface [4]. Inside of the volume, however, larger tetrahedral elements are often desired for efficient simulations. Thus, the sizing field used in Chapters 5 and 6 is smoothly increased away from the surface using the gradient limiting algorithm (see Section 5.3.2), and drives the sampling density of the spherical particles.

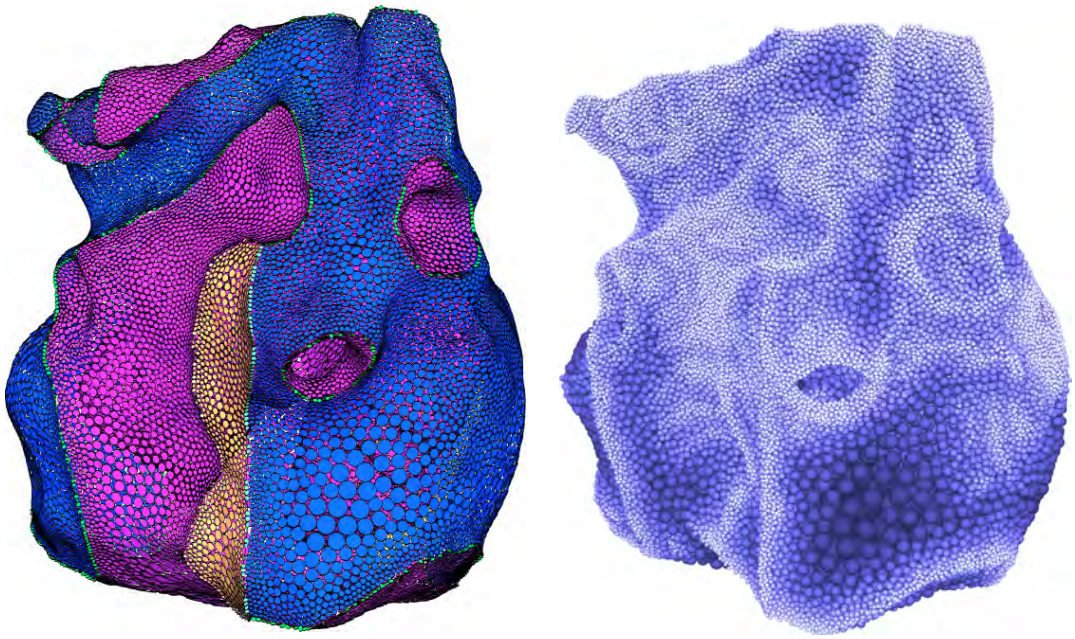


Figure 7.2. Surface samples of the heart material, along with the spherical particles packed within.

For generating tetrahedral meshes, we have experimented with Tetgen[1]. First, a boundary (surface) mesh is created (see Chapters 5 and 6), which is input to Tetgen along with the set of volumetric particles. The output is a constrained Delaunay tetrahedralization. In Figure 7.3, a tetrahedralization is shown of the sample points from Figure 7.2. Figure 7.4 shows a tetrahedralization of the spiny dendrite dataset from Chapter 5, while Figure 7.5 presents tetrahedralizations of the frog and two-sphere examples from Chapter 6.

Although well-distributed points generate well-shaped triangles in $\Re^2$, this does not extend to tetrahedral elements in $\Re^3$ [141]. The problem arises when four points are nearly coplanar and tessellated as a *sliver* tetrahedron — these slivers can have nearly equal length edges (which for triangles indicates a well-shaped element) , but are very flat. The flatness of these elements gives them poor numerical characteristics in simulations. Slivers are problem for many volumetric meshing algorithms [4], and especially for 3D Delaunay methods [49]. Thus, a significant amount of research focuses on algorithms for refining tetrahedralizations to eliminate slivers, such as *sliver exudation* [30] and *lattice refinement* [81]. The meshes created from the volumetric particle system also contain a small number of these poorly-shaped elements. However, the regularity of the majority of the tetrahedra is encouraging. We are interested in pursuing strategies for

[1] tetgen.berlios.de

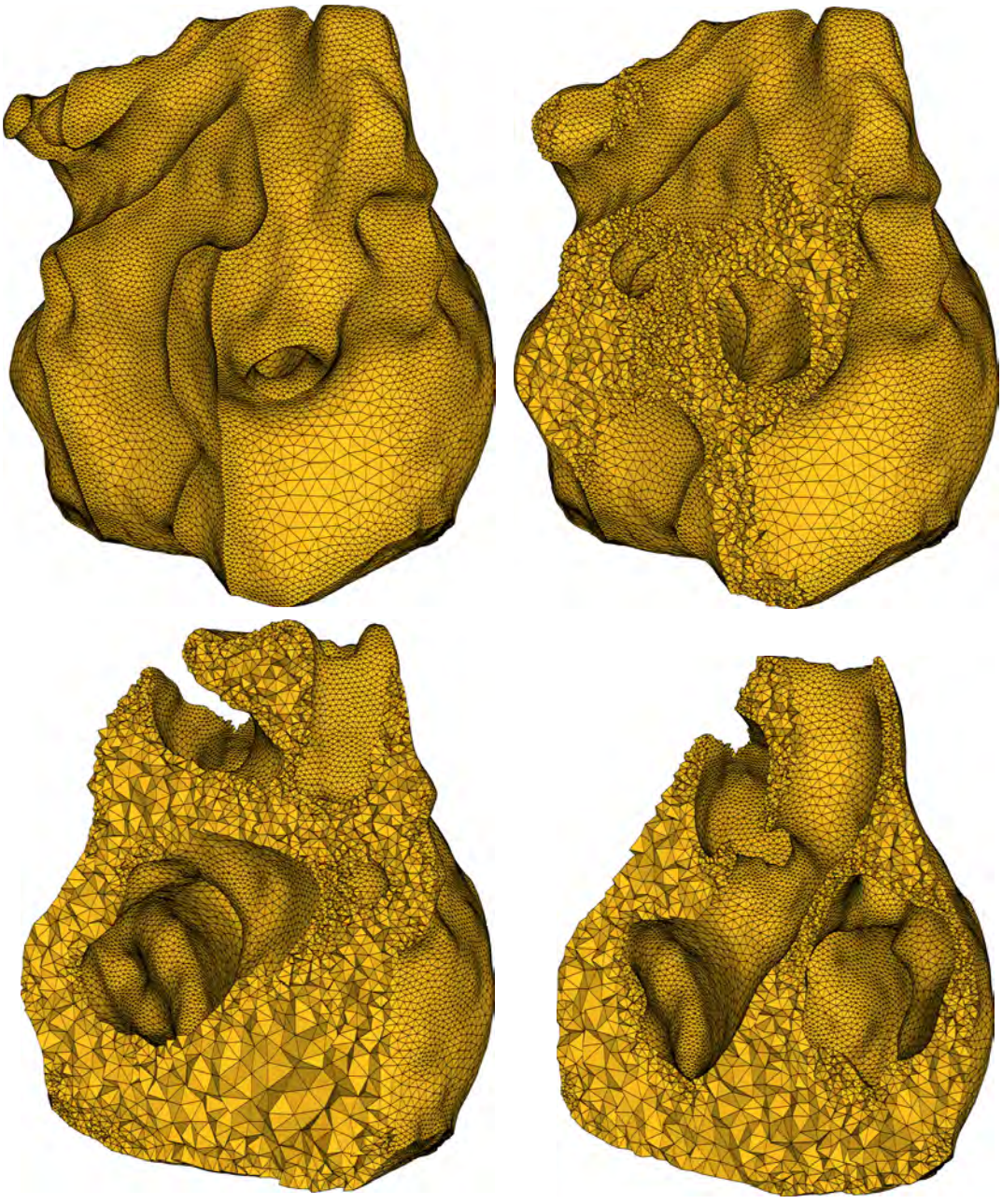


Figure 7.3. A progressive cut-away of a tetrahedralization of the heart samples shown in Figure 7.2.

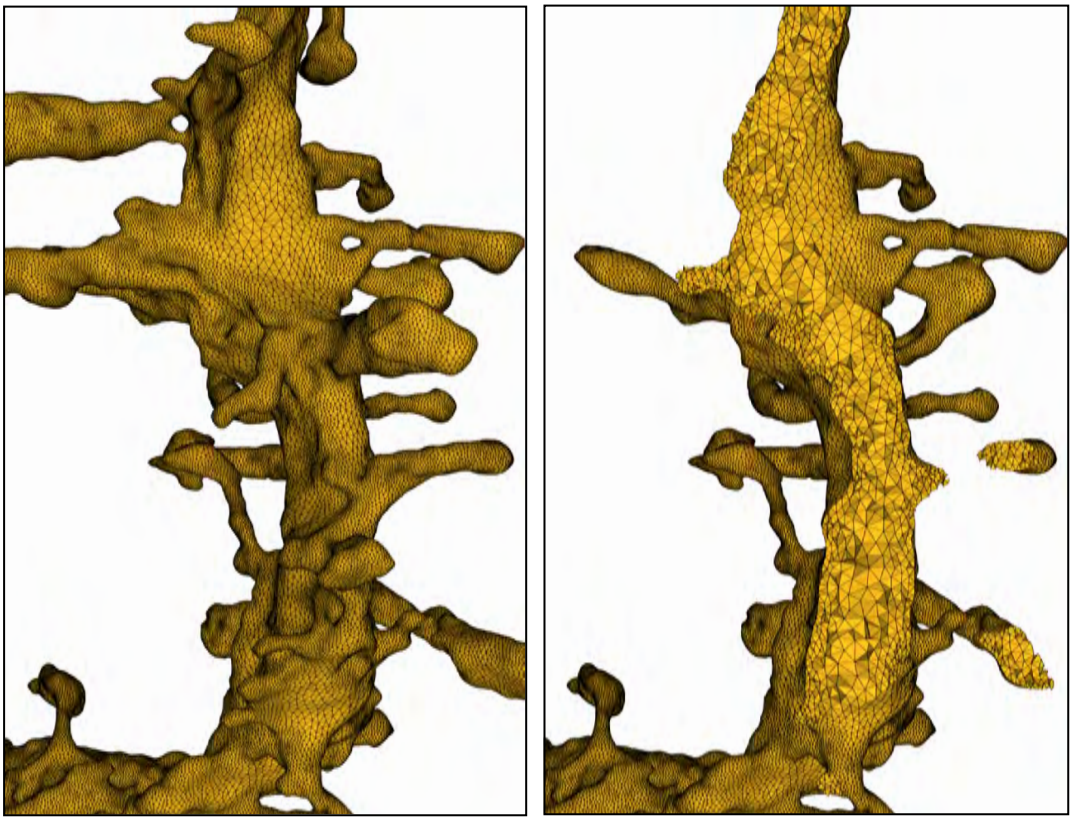


Figure 7.4. A cut-away of a tetrahedralization of the dendrite dataset from Figure 5.11.

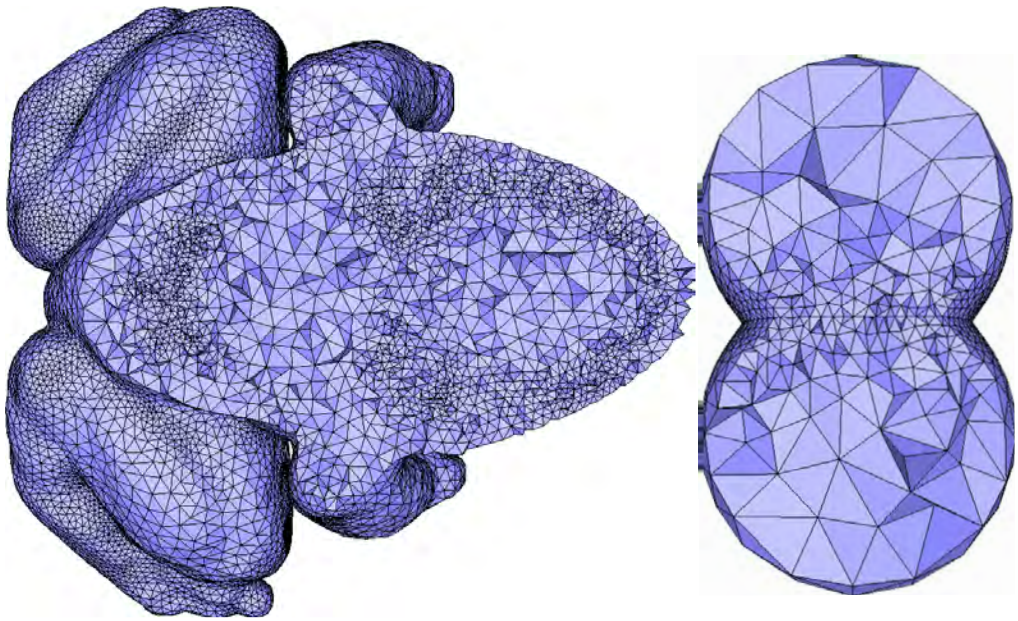


Figure 7.5. Tetrahedralizations of the interface and volume particle samples, shown with a cutting plane.

removing slivers, such as point insertion or grid refinement.

Similar in spirit to the field of artificial intelligence, the particles are, at their essence, a collection of agents governed by a few simple rules that, when interacting, produce complicated distributions. When these systems are applied to the surface sampling problem, the result is a well-distributed set of point samples over arbitrarily complex surfaces. Changing the rules that direct how particles interact with each other creates adaptive distributions that meet a specific sampling constraint, whether that constraint is for ensuring geometrically accurate meshes, or for parameterizing a set of shapes [28]. This latter example illustrates a new approach for sampling *ensembles* of shapes using dynamic particle systems for studying and computing the statistics of a set of surfaces. In this work, a a set of particles sample each shape, interacting with not only neighboring particles on the surface, but also *corresponding* particles across the entire set of surfaces. These complex interactions result in sets of point samples that balance an even distribution of points across any one surface with a similar parameterization of each surface in the set.

The flexibility of dynamic particle systems comes at the cost of long computation times compared to other sampling schemes, such as grid-based approaches. The system requires many iterations to converge, and each particle update can be long if the distribution is highly adaptive or if the particle-particle interactions are complex. Parallelization of the system is not straightforward due to the nonlocal propagation of the local particle behaviors, *i.e.*, each particle movement effects a neighborhood of other particles' movements. Multigrid methods [147] are an effective numerical scheme for solving these types of global optimization problems, and may be extendible to dynamic particle systems. Also, the GPU has been used to simulate systems of noninteracting particles, and may be applicable for simulating interacting particles as well.

This dissertation provides an empirical analysis of the behavior of dynamic particle systems, and we believe that a rigorous, theoretical analysis is possible for characterizing the system's general behaviors. However, even without a theoretical description, we have demonstrated that in practice, dynamic particle systems are an effective mechanism for approaching numerous, current research problems. As further study into the behavior of this dynamic system occurs, we expect its applicability to extend broadly.

APPENDIX A

This appendix provides a brief outline of the software used in Chapters 5 and 6 to generate meshes from particles. Included are the algorithms and sources of code or executables for each step of the pipelines.

Chapter 5

- **Downsampling and padding volumes:** The *teem*[1] software provided by Gordon Kindlmann.
- **Binary morphology for eliminating small features:** Personal implementation of the operations *erode*, *dilate*, *open*, and *close*, as defined by Gonzalez and Woods [54].
- **Greyscale morphology for volume smoothing:** Implementation by Ross Whitaker of the *tightening* algorithm of Williams and Rossignac [160].
- **Medial axis:** Implementation by Ross Whitaker of an unpublished algorithm based on *footpoints* (see Section 5.3.1).
- **Initial sizing field (LFS):** Personal implementation of an adaptive octree for quering distances to closests points.
- **Smoothing of sizing field:** Personal implementation of the *gradient limiting* algorthim of Persson [113].
- **Initialization of particle system:** Personal implementation of the *marching cubes* algorithm of Lorensen and Cline [90].
- **Particle system:** Personal implementation of the system described throughout this dissertation.

[1] teem.sourceforge.net

- **Meshing particles:** Executable provided by Tamal Dey[2] based on the algorithm of Dey and Goswami [44].

Chapter 6

- **Downsampling and padding volumes:** The *teem*[3] software provided by Gordon Kindlmann.
- **Binary morphology for eliminating small features:** Personal implementation of the operations *erode*, *dilate*, *open*, and *close*, as defined by Gonzalez and Woods [54].
- **Filing voids after morphology:** Personal implementation of the voting algorithm described in Section 6.6.1.
- **Greyscale morphology for volume smoothing:** Implementation by Ross Whitaker of the *tightening* algorithm of Williams and Rossignac [160].
- **Medial axis:** Implementation by Ross Whitaker of an unpublished algorithm based on *footpoints* (see Section 5.3.1).
- **Initial sizing field (LFS):** Personal implementation of an adaptive octree for quering distances to closests points.
- **Smoothing of sizing field:** Personal implementation of the *gradient limiting* algorthim of Persson [113].
- **Initialization of particle system:** Personal implementation of the *marching cubes* algorithm of Lorensen and Cline [90].
- **Particle system:** Personal implementation of the system described throughout this dissertation.
- **Tetrahedralization of particles:** The *Tetgen*[4] software provided by Hang Si.

[2] www.cse.ohio-state.edu/ tamaldey/cocone.html

[3] teem.sourceforge.net

[4] tetgen.berlios.de

- **Extraction of surface mesh(es):** Personal implementation of the labeling algorithm described in Section 6.4.3.

REFERENCES

[1] Open-source environment for interactive finite element modeling of optimal icd electrode placement. In *FIMH* (2007), F. B. Sachse and G. Seemann, Eds., vol. 4466 of *Lecture Notes in Computer Science*, Springer, pp. 373–382.

[2] ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3 (July 2007), 48.

[3] ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (January 2003), 3–15.

[4] ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. Variational tetrahedral meshing. *ACM Transactions On Graphics 24*, 3 (2005), 617–625.

[5] AMENTA, N., AND BERN, M. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry 22* (1999), 481–504.

[6] AMENTA, N., BERN, M., AND EPPSTEIN, D. The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphic Models and Image Processing 60*, 2 (Mar. 1998), 125–135.

[7] AMENTA, N., BERN, M., AND KAMVYSSELIS, M. A new voronoi-based surface reconstruction algorithm. In *Proceedings of SIGGRAPH* (July 1998), pp. 415–421.

[8] AMENTA, N., CHOI, S., DEY, T., AND LEEKHA, N. A simple algorithm for homeomorphic surface reconstruction. In *ACM Symposium on Computational Geometry* (2000), pp. 213–222.

[9] AMENTA, N., CHOI, S., AND KOLLURI, R. K. The power crust. In *SMA '01: Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2001), ACM Press, pp. 249–266.

[10] ANDERSON, A. E. *Computational Modeling of Hip Joint Mechanics*. PhD thesis, Department of Bioengineering, University of Utah, 2007.

[11] BABUSKA, I., AND AZIZ, A. K. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis 13*, 2 (1976), 214–226.

[12] BERTRAM, M., REIS, G., VAN LENGEN, R. H., KÖHN, S., AND HAGEN, H. Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart. In *Proceedings of EuroVis* (June 2005), pp. 199–206.

[13] BISCHOFF, S., AND KOBBELT, L. *Extracting Consistent and Manifold Interfaces from*

Multi-valued Volume Data Sets. 2006.

[14] BISHOP, R. L., AND GOLDBERG, S. I. *Tensor Analysis on Manifolds*. Dover, 1980.

[15] BLINN, J. F. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[16] BLOOMENTHAL, J. *Introduction to implicit surfaces*. 1997.

[17] BLOOMENTHAL, J., AND FERGUSON, K. Polygonization of non-manifold implicit surfaces. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), ACM, pp. 309–316.

[18] BOBENKO, A. I., AND SCHRÖDER, P. Discrete willmore flow. In *SGP '05: Proceedings of the Third Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, p. 101.

[19] BOISSONNAT, J.-D., AND OUDOT, S. Provably good sampling and meshing of surfaces. *Graphical Models 67*, 5 (Sept. 2005), 405–451.

[20] BONNELL, K. S., DUCHAINEAU, M. A., SCHIKORE, D. R., HAMANN, B., AND JOY, K. I. Material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 9*, 4 (Oct./Dec. 2003), 500–511.

[21] BRASHER, M., AND HAIMES, R. Rendering planar cuts through quadratic and cubic finite elements. In *VIS '04: Proceedings of the Conference on Visualization '04* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 409–416.

[22] BREEN, D. E., AND WHITAKER, R. T. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (2001), 173–192.

[23] BUNYK, P., KAUFMAN, A., AND SILVA, C. T. Simple, fast, and robust ray casting of irregular grids. In *Proceedings of Dagstuhl '97* (2000), pp. 30–36.

[24] BUTT, M., AND MARAGOS, P. Optimum design of chamfer distance transforms. 1477–1484.

[25] CABRAL, B., CAM, N., AND FORAN, J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS '94: Proceedings of the 1994 Symposium on Volume Visualization* (New York, NY, USA, 1994), ACM Press, pp. 91–98.

[26] CALLAHAN, S. P., IKITS, M., COMBA, J. L., AND SILVA, C. T. Hardware-assisted visibility ordering for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 285–295.

[27] CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of ACM SIGGRAPH* (Aug. 2001), pp. 67–76.

[28] CATES, J., MEYER, M., FLETCHER, T., AND WHITAKER, R. Entropy-based particle systems for shape correspondence. In *Proceedings of the Workshop on Mathematical*

Foundations of Computational Anatomy, MICCAI (2006), pp. 90–99.

[29] CHEN, W., REN, L., ZWICKER, M., AND PFISTER, H. Hardware-accelerated adaptive EWA volume splatting. In *Proceedings of IEEE Visualization 2004* (Oct. 2004).

[30] CHENG, S.-W., DEY, T. K., EDELSBRUNNER, H., FACELLO, M. A., AND TENG, S.-H. Silver exudation. *J. ACM 47*, 5 (2000), 883–904.

[31] CHENG, S.-W., DEY, T. K., RAMOS, E. A., AND RAY, T. Sampling and meshing a surface with guaranteed topology and geometry. In *SCG '04: Proceedings of the Twentieth Annual Symposium on Computational Geometry* (New York, NY, USA, 2004), ACM Press, pp. 280–289.

[32] CHEW, L. P. Guaranteed-quality delaunay meshing in 3d (short version). In *SCG '97: Proceedings of the Thirteenth Annual Symposium on Computational Geometry* (New York, NY, USA, 1997), ACM Press, pp. 391–393.

[33] COPPOLA, G., SHERWIN, S. J., AND PEIRO, J. Nonlinear particle tracking for high-order elements. *J. Comput. Phys. 172*, 1 (2001), 356–386.

[34] COTIN, S., DURIEZ, C., LENOIR, J., NEUMANN, P., AND DAWSON, S. New approaches to catheter navigation for interventional radiology simulation. pp. 534–542.

[35] CROSSNO, P., AND ANGEL, E. Isosurface extraction using particle systems. In *Proceedings of the 8th Conference on Visualization '97* (1997), IEEE Computer Society Press, pp. 495–498.

[36] CSÉBFALVI, B., MROZ, L., HAUSER, H., KÖNIG, A., AND GRÖLLER, E. Fast visualization of object contours by non-photorealistic volume rendering. *Comput. Graph. Forum 20*, 3 (2001).

[37] CUTLER, B., DORSEY, J., AND MCMILLAN, L. Simplification and improvement of tetrahedral models for simulation. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (New York, NY, USA, 2004), ACM, pp. 93–102.

[38] DANIELSSON, P. Euclidean distance mapping. 227–248.

[39] DE FIGUEIREDO, L. H., DE MIRANDA GOMES, J., TERZOPOULOS, D., AND VELHO, L. Physically-based methods for polygonization of implicit surfaces. In *Proceedings of the Conference on Graphics Interface '92* (1992), pp. 250–257.

[40] DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324.

[41] DESBRUN, M., TSINGOS, N., AND CANI, M.-P. Adaptive sampling of implicit surfaces for interactive modeling and animation. In *Implicit Surfaces'95* (April 1995), pp. 171–185. Published under the name Marie-Paule Gascuel.

[42] DEVILLE, M., MUND, E., AND FISCHER, P. *High Order Methods for Incompressible Fluid Flow*. Cambridge University Press, 2002.

[43] DEY, T. K., AND GIESEN, J. Detecting undersampling in surface reconstruction. In *SCG '01: Proceedings of the Seventeenth Annual Symposium on Computational Geometry* (New York, NY, USA, 2001), ACM Press, pp. 257–263.

[44] DEY, T. K., AND GOSWAMI, S. Tight Cocone: A water-tight surface reconstructor. *Journal of Computing and Information Science in Engineering 3*, 4 (December 2003), 302–307.

[45] DEY, T. K., AND LEVINE, J. A. Delaunay meshing of isosurfaces. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 241–250.

[46] DILLARD, S., BINGERT, J., AND THOMA, D. Construction of simplified boundary surfaces from serial-sectioned metal micrographs. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1528–1535.

[47] DOI, A., AND KOIDE, A. An efficient method of triangulating equivalued surfaces by using tetrahedral cells. *IEICE Transactions Communication, Elec. Info. Syst E74*, 1 (January 1991), 214–224.

[48] DREBIN, R. A., CARPENTER, L., AND HANRAHAN, P. Volume rendering. In *SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), ACM Press, pp. 65–74.

[49] EDELSBRUNNER, H., LI, X.-Y., MILLER, G., STATHOPOULOS, A., TALMOR, D., TENG, S.-H., ÜNGÖR, A., AND WALKINGTON, N. Smoothing and cleaning up slivers. In *STOC '00: Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2000), ACM, pp. 273–277.

[50] EDELSBRUNNER, H., AND MÜCKE, E. P. Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (1994), 43–72.

[51] ELLISMAN, M. *National Center for Microscopy Image Research*. University of California San Diego.

[52] GALIN, E., ALLEGRE, R., AND AKKOUCHE, S. A fast particle system framework for interactive implicit modeling. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)* (Washington, DC, USA, 2006), IEEE Computer Society, p. 32.

[53] GLASSNER, A. S. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications 4*, 10 (1984), 15–22.

[54] GONZALEZ, R. C., AND WOODS, R. E. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[55] GOOCH, B., AND GOOCH, A. *Non-photorealistic Rendering*. A. K. Peters, Ltd, 2001.

[56] GOURAUD, H. *Computer display of curved surfaces*. PhD thesis, 1971.

[57] GROSS, M., AND PFISTER, H., Eds. *Point-Based Graphics*. Elsevier / Morgan Kaufman, 2007.

[58] HALES, T. C. Cannonballs and honeycombs. *Notices of the American Mathematical Society 47*, 4 (April 2000), 440–449.

[59] HARDIN, D. P., AND SAFF, E. B. Discretizing manifolds via minimum energy points. *Notices of the American Mathematical Society 51*, 10 (November 2004), 1186–1194.

[60] HART, J. C. Ray tracing implicit surfaces. In *SIGGRAPH 93 Modeling, Visualizing, and Animating Implicit Surfaces course notes*. 1993, pp. 13–1 to 13–15.

[61] HART, J. C. Using the CW-complex to represent the topological structure of implicit surfaces and solids. In *Proceedings of Implicit Surfaces* (1999), pp. 107–112.

[62] HART, J. C., BACHTA, E., JAROSZ, W., AND FLEURY, T. Using particles to sample and control more complex implicit surfaces. In *Proceedings of the Shape Modeling International 2002 (SMI'02)* (2002), IEEE Computer Society, p. 129.

[63] HARTMANN, E. A marching method for the triangulation of surfaces. *The Visual Computer 14*, 3 (1998), 95–108.

[64] HARTMANN, E. On the curvature of curves and surfaces defined by normalforms. *Comput. Aided Geom. Des. 16*, 5 (1999), 355–376.

[65] HECKBERT, P. Fast surface particle repulsion. In *SIGGRAPH '97, New Frontiers in Modeling and Texturing Course* (August 1997), ACM Press, pp. 95–114.

[66] HEGE, H.-C., SEEBASS, M., STALLING, D., AND ZÖCKLER, M. A generalized marching cubes algorithm based on non-binary classifications. ZIB Preprint SC-97-05, 1997.

[67] HUGHES, T. J. R. *The Finite Element Method*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[68] HUGHES, T. J. R., COTTRELL, J. A., AND AZILEVS, Y. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. Tech. Rep. 04-50, ICES, University of Texas at Austin, October 2004.

[69] JOHNSON, C., AND HANSEN, C. *Visualization Handbook*. Academic Press, Inc., Orlando, FL, USA, 2004.

[70] JONES, M. W., BAERENTZEN, J. A., AND SRAMEK, M. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 581–599.

[71] JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. Dual contouring of hermite data. *ACM Transactions on Graphics 21*, 3 (July 2002), 339–346.

[72] KARKANIS, T., AND STEWART, A. J. Curvature-dependent triangulation of implicit

surfaces. *IEEE Computer Graphics and Applications 22*, 2 (March 2001), 60–69.

[73] KARNIADAKIS, G., AND II, R. M. K. *Parallel Scientific Computing in C++ and MPI*. Cambridge, New York, 2003.

[74] KARNIADAKIS, G. E., AND SHERWIN, S. J. *Spectral/hp element methods for CFD*. Oxford University Press, New York, NY, USA, 1999.

[75] KEISER, R., MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. Contact handling for deformable point-based objects. In *Proceedings of the 9th International Workshop "Vision, Modeling, and Visualization"* (Nov 2004), pp. 339–346.

[76] KINDLMANN, G. *Visualization and Analysis of Diffusion Tensor Fields*. PhD thesis, School of Computing, University of Utah, 2004.

[77] KINDLMANN, G., AND DURKIN, J. W. Semi-automatic generation of transfer functions for direct volume rendering. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization* (New York, NY, USA, 1998), ACM Press, pp. 79–86.

[78] KINDLMANN, G., WHITAKER, R., TASDIZEN, T., AND MÖLLER, T. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization 2003* (October 2003), pp. 513–520.

[79] KNISS, J., HUNT, W., POTTER, K., AND SEN, P. Istar: A raster representation for scalable image and volume data. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1424–1431.

[80] KRIM, H., AND YEZZI, A., Eds. *Statistics and Analysis of Shapes*. Birkhauser Boston, 2006.

[81] LABELLE, F. Sliver removal by lattice refinement. In *SCG '06: Proceedings of the Twenty-second Annual Symposium on Computational Geometry* (New York, NY, USA, 2006), ACM, pp. 347–356.

[82] LACROUTE, P., AND LEVOY, M. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM Press, pp. 451–458.

[83] LAU, T. S., AND LO, S. H. Finite element mesh generation over analytic curved surfaces. *Computers and Structures 59*, 2 (1996), 301–310.

[84] LEE, A., MORETON, H., AND HOPPE, H. Displaced subdivision surfaces. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 85–94.

[85] LEFOHN, A., KNISS, J., HANSEN, C., AND WHITAKER, R. Interactive deformation and visualization of level set surfaces using graphics hardware. In *IEEE Visualization 2003* (October 2003), pp. 75–82.

[86] LEVET, F., GRANIER, X., AND SCHLICK, C. Fast sampling of implicit surfaces by particle systems. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)* (Washington, DC, USA, 2006), IEEE Computer Society, p. 39.

[87] LEVOY, M. Display of surfaces from volume data. *IEEE Comput. Graph. Appl. 8*, 3 (1988), 29–37.

[88] LI, G.-S., TRICOCHE, X., WEISKOPF, D., AND HANSEN, C. Flow charts: Visualization of vector fields on arbitrar y surfaces. submitted to IEEE Transactions on Visualization and Computer Graphics.

[89] LICHTENBELT, B., CRANE, R., AND NAQVI, S. *Introduction to volume rendering*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[90] LORENSEN, W. E., AND CLINE, H. E. Marching Cubes: A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH* (July 1987), pp. 163–169.

[91] MAX, N. L. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph. 1*, 2 (1995), 99–108.

[92] MERRIMAN, B., BENCE, J. K., AND OSHER, S. J. Motion of multiple junctions: A level set approach. *J. Comput. Phys. 112*, 2 (1994), 334–363.

[93] MEYER, M., GEORGEL, P., AND WHITAKER, R. Robust particle systems for curvature dependent sampling of implicit surfaces. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI)* (June 2005), pp. 124–133.

[94] MEYER, M., KIRBY, R. M., AND WHITAKER, R. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (September/October 2007), 1704–1711.

[95] MEYER, M., NELSON, B., KIRBY, R. M., AND WHITAKER, R. Particle systems for efficient and accurate high-order finite element visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 1015–1026.

[96] MEYER, M., WHITAKER, R., KIRBY, R. M., LEDERGERBER, C., AND PFISTER, H. Particle-based sampling and meshing of multimaterial volumes. *submitted*.

[97] MILLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2004), ACM Press, pp. 141–151.

[98] MITCHELL, D. P. Robust ray intersection with interval arithmetic. In *Proceedings on Graphics interface '90* (Toronto, Ont., Canada, Canada, 1990), Canadian Information Processing Society, pp. 68–74.

[99] MÖLLER, T., MUELLER, K., KURZION, Y., MACHIRAJU, R., AND YAGEL, R. Design of accurate and smooth filters for function and derivative reconstruction. In *VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization* (New York, NY, USA,

1998), ACM Press, pp. 143–151.

[100] MORSE, B. S., YOO, T. S., CHEN, D. T., RHEINGANS, P., AND SUBRAMANIAN, K. R. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications* (Washington, DC, USA, 2001), IEEE Computer Society, p. 89.

[101] MULLIKIN, J. C. The vector distance transform in two and three dimensions. *CVGIP: Graph. Models Image Process. 54*, 6 (1992), 526–535.

[102] MUSETH, K., BREEN, D. E., WHITAKER, R. T., AND BARR, A. H. Level set surface editing operators. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), ACM Press, pp. 330–338.

[103] NELSON, B., AND KIRBY, R. M. Ray-tracing polymorphic multi-domain spectral/*hp* elements for isosurface rendering. *IEEE Transactions on Visualization and Computer Graphics 12*, 1 (2006), 114–125.

[104] NEMITZ, O., RUMPF, M., TASDIZEN, T., AND WHITAKER, R. Anisotropic curvature motion for structure enhancing smoothing of 3D MR angiography data. *Journal of Mathematical Imaging and Vision* (December 2006). published online.

[105] NIELSON, G., AND FRANKE, R. Computing the separating surface for segmented data. In *IEEE Visualization* (Nov. 1997), pp. 229–236.

[106] NIELSON, G. M., HUANG, A., AND SYLVESTER, S. Approximating normals for marching cubes applied to locally supported isosurfaces. In *IEEE Visualization* (2002), pp. 459–466.

[107] NISHIMURA, H., HIRAI, M., KAWAI, T., KAWATA, T., SHIRAKAWA, I., AND OMURA, K. Object modelling by distribution function and a method of image generation. In *Transactions of the Institute of Electronics and Communication Engineers of Japan* (1985), vol. J68-D, pp. 718–725.

[108] OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. Multi-level partition of unity implicits. *ACM Transactions On Graphics 22*, 3 (July 2003), 463–470.

[109] OKA, M., NAKATA, S., AND TANAKA, S. Preprocessing for accelerating convergence of repulsive-particle systems for sampling implicit surfaces. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 232–240.

[110] OUDOT, S., RINEAU, L., AND YVINEC, M. Meshing volumes bounded by smooth surfaces. In *Proc. 14th International Meshing Roundtable* (2005), pp. 203–219.

[111] PARKER, S., PARKER, M., LIVNAT, Y., SLOAN, P.-P., HANSEN, C., AND SHIRLEY, P. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (1999), 238–250.

[112] PASCUCCI, V. Isosurface computation made simple: Hardware acceleration, adaptive

refinement and tetrahedral stripping. In *Proceedings of IEEE TVCG Symposium on Visualization* (2004), pp. 293–300.

[113] PERSSON, P.-O. Mesh size functions for implicit geometries and pde-based gradient limiting. *Eng. with Comput. 22*, 2 (2006), 95–109.

[114] PFISTER, H., HARDENBERGH, J., KNITTEL, J., LAUER, H., AND SEILER, L. The volumepro real-time ray-casting system. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 251–260.

[115] PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), pp. 335–342.

[116] PHONG, B. T. *Illumination for computer-generated images.* PhD thesis, 1973.

[117] POHL, K., BOUIX, S., NAKAMURA, M., ROHLFING, T., MCCARLEY, R., KIKINIS, R., GRIMSON, W., SHENTON, M., AND WELLS, W. A hierarchical algorithm for mr brain image parcellation. *IEEE Transactions on Medical Imaging 26*, 9 (2007), 1201–1212.

[118] PONS, J.-P., SÉGONNE, F., BOISSONNAT, J.-D., RINEAU, L., YVINEC, M., AND KERIVEN, R. High-quality consistent meshing of multi-label datasets. In *Information Processing in Medical Imaging* (2007), pp. 198–210.

[119] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[120] REITINGER, B., BORNIK, A., AND BEICHEL, R. Constructing smooth non-manifold meshes of multi-labeled volumetric datasets. In *WSCG (Full Papers)* (2005), pp. 227–234.

[121] RHEINGANS, P., AND EBERT, D. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 253–264.

[122] ROETTGER, S., KRAUS, M., AND ERTL, T. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proceedings of IEEE Visualization* (Oct. 2000), pp. 109–116.

[123] RÖSCH, A., RUHL, M., AND SAUPE, D. Interactive visualization of implicit surfaces with singularities. *Eurographics Computer Graphics Forum 16*, 5 (1996), 295–306.

[124] ROSENFELD, A., AND PFALTZ, J. L. Sequential operations in digital picture processing. *J. ACM 13*, 4 (1966), 471–494.

[125] RUBIN, S. M., AND WHITTED, T. A 3-dimensional representation for fast rendering of complex scenes. In *SIGGRAPH '80: Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1980), ACM Press, pp. 110–116.

[126] RUPPERT, J. A new and simple algorithm for quality 2-dimensional mesh generation. In *SODA '93: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 1993), Society for Industrial and Applied Mathematics, pp. 83–92.

[127] SAFF, E. B., AND KUIJLAARS, A. B. J. Distributing many points on a sphere. *Mathematical Intelligencer 19*, 1 (1997), 5–11.

[128] 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.

[129] SCHREINER, J., SCHEIDEGGER, C., AND SILVA, C. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct. 2006), 1205–1212.

[130] SCHROEDER, W. J., BERTEL, F., MALATERRE, M., THOMPSON, D., PÉBAY, P. P., O'BARA, R. M., AND TENDULKAR, S. Framework for visualizing higher-order basis functions. In *IEEE Visualization* (2005).

[131] SETHIAN, J. A. Curvature flow and entropy conditions applied to grid generation. *J. Comput. Phys. 115*, 2 (1994), 440–454.

[132] SETHIAN, J. A. Fast marching methods. *SIAM Review 41*, 2 (1999), 199–235.

[133] SETHIAN, J. A. *Level Set Methods and Fast Marching Methods*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, 1999.

[134] SHERWIN, S. J., AND PEIRO, J. Mesh generation in curvilinear domains using high-order elements. *International Journal of Numerical Methods in Engineering 53*, 1 (2002), 207–223.

[135] SHEWCHUK, J. R. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures. Tech. Rep. Manuscript, 2002.

[136] SHIRLEY, P., AND TUCHMAN, A. A polygonal approximation to direct scalar volume rendering. *Proceedings of San Diego Workshop on Volume Visualization 24*, 5 (Nov. 1990), 63–70.

[137] SIDDIQI, K., BOUIX, S., TANNENBAUM, A., AND ZUCKER, S. W. The hamilton-jacobi skeleton. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2* (Washington, DC, USA, 1999), IEEE Computer Society, p. 828.

[138] SZABÓ, B., AND BABUŠKA, I. *Finite Element Analysis*. John Wiley & Sons, New York, 1991.

[139] SZELISKI, R., AND TONNESEN, D. Surface modeling with oriented particle systems. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (1992), ACM Press, pp. 185–194.

[140] SZELISKI, R., TONNESEN, D., AND TERZOPOULOS, D. Modeling surfaces of arbitrary

topology with dynamic particles. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)* (New York, NY, June 1993), pp. 140–152.

[141] TALMOR, D. *Well-Spaced Points for Numerical Methods*. PhD thesis, Carnegie Mellon University, Pittsburgh, August 1997. CMU CS Tech Report CMU-CS-97-164.

[142] TASDIZEN, T., AWATE, S. P., WHITAKER, R. T., AND FOSTER, N. L. Mri tissue classification with neighborhood statistics: A nonparametric, entropy-minimizing approach. In *MICCAI (2)* (2005), pp. 517–525.

[143] TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. Geometric surface smoothing via anisotropic diffusion of normals. In *VIS '02: Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society.

[144] TAYLOR, C. A., AND DRANEY, M. T. Experimental and computational methods in cardiovascular fluid mechanics. *Annual Review of Fluid Mechanics 36* (2004), 197–231.

[145] THOMPSON, D. C., AND PEBAY, P. P. Visualizing higher order finite elements: Final report. Tech. Rep. SAND2005-6999, Sandia National Laboratories, 2005.

[146] TOUMA, C., AND GOTSMAN, C. Triangle mesh compression. In *Graphics Interface* (1998), pp. 26–34.

[147] TROTTENBERG, U., OOSTERLEE, C. W., AND SCHÜLLER, A. *Multigrid*. Academic Press Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.

[148] TURK, G. Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), ACM Press, pp. 55–64.

[149] TURK, G., AND O'BRIEN, J. F. Variational implicit surfaces. Tech. Rep. GIT-GVU-99-15, Georgia Institute of Technology, 1999.

[150] TURK, G., AND O'BRIEN, J. F. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph. 21*, 4 (2002), 855–873.

[151] UITERT, R. V., JOHNSON, C., AND ZHUKOV, L. Influence of head tissue conductivity in forward and inverse magnetoencephalographic simulations using realistic head models. *IEEE Transactions on Biomedical Engineering 51*, 12 (2004), 2129–2137.

[152] VAN WIJK, J. J. Image based flow visualization for curved surfaces. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 17.

[153] VELHO, L. Simple and efficient polygonization of implicit surfaces. *J. Graph. Tools 1*, 2 (1996), 5–24.

[154] WEILER, M., KRAUS, M., MERZ, M., AND ERTL, T. Hardware-based ray casting for tetrahedral meshes. In *Proceedings of IEEE Visualization 2003* (Oct. 2003), pp. 333–340.

[155] WESTOVER, L. A. *Splatting: A parallel, feed-forward volume rendering algorithm*. PhD thesis, Chapel Hill, NC, USA, 1991.

[156] WHITAKER, R. T. Reducing aliasing artifacts in iso-surfaces of binary volumes. In *VVS '00: Proceedings of the 2000 IEEE Symposium on Volume Visualization* (New York, NY, USA, 2000), ACM Press, pp. 23–32.

[157] WHITTED, T. An improved illumination model for shaded display. *Commun. ACM 23*, 6 (1980), 343–349.

[158] WILEY, D. F., CHILDS, H., HAMANN, B., AND JOY, K. I. Ray casting curved-quadratic elements. In *Data Visualization 2004* (2004), Eurographics/IEEE TCVG, ACM Siggraph, pp. 201–209.

[159] WILEY, D. F., CHILDS, H. R., GREGORSKI, B. F., HAMANN, B., AND JOY, K. I. Contouring curved quadratic elements. In *Data Visualization 2003, Proceedings of VisSym 2003* (2003).

[160] WILLIAMS, J., AND ROSSIGNAC, J. Tightening: Curvature-limiting morphological simplification. In *Proceedings of the Symposium on Solid and Physical Modeling* (2005), pp. 107–112.

[161] WILLIAMS, P. L. Visibility-ordering meshed polyhedra. *ACM Transactions on Graphics 11*, 2 (Apr. 1992), 103–126.

[162] WITKIN, A. P., AND HECKBERT, P. S. Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH* (July 1994), pp. 269–277.

[163] WOOD, Z. J., SCHRODER, P., BREEN, D., AND DESBRUN, M. Semi-regular mesh extraction from volumes. In *VIS '00: Proceedings of the Conference on Visualization '00* (Los Alamitos, CA, USA, 2000), IEEE Computer Society Press, pp. 275–282.

[164] WYVILL, B., GUY, A., AND GALIN, E. Extending the csg tree - warping, blending and boolean operations in an implicit surface modeling system. *Comput. Graph. Forum 18*, 2 (1999), 149–158.

[165] WYVILL, G., MCPHEETERS, C., AND WYVILL, B. Data structure for soft objects. *The Visual Computer*, 2 (1986), 227–234.

[166] ZHANG, Y., HUGHES, T. J. R., AND BAJAJ, C. Automatic 3d mesh generation for a domain with multiple materials. In *IMR* (2007), pp. 367–386.

[167] ZHANG, Y., XU, G., AND BAJAJ, C. Quality meshing of implicit solvation models of biomolecular structures. *Comput. Aided Geom. Des. 23*, 6 (2006), 510–530.

[168] ZHAO, H.-K., CHAN, T., MERRIMAN, B., AND OSHER, S. A variational level set approach to multiphase motion. *Journal of Computational Physics 127*, 1 (1996), 179–195.

[169] ZONENSCHEIN, R., GOMES, J., VELHO, L., AND DE FIGUEIREDO, L. H. Controlling texture mapping onto implicit surfaces with particle systems. In *Proceedings of Implicit Surfaces '98* (June 1998), pp. 131–138.

[170] ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and*

Interactive Techniques (New York, NY, USA, 2001), ACM Press, pp. 371–378.

[171] ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 223–238.

[172] ZWICKER, M., ROSNEN, J., BOTSCH, M., DACHSBACHER, C., AND PAULY, M. Perspective accurate splatting. In *GI '04: Proceedings of the 2004 Conference on Graphics Interface* (2004), pp. 247–254.