PDE-based Geometric Modeling and Interactive Sculpting for Graphics

A DISSERTATION PRESENTED

BY

HAIXIA DU

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

May 2004

Abstract of the Dissertation

PDE-based Geometric Modeling and Interactive Sculpting for Graphics

by

Haixia Du

Doctor of Philosophy

in

Computer Science

Stony Brook University

2004

Advisor: Professor Hong Qin

Partial differential equations (PDEs) are widely used in Computer Graphics fields to model geometric objects, simulate natural phenomena, formulate physical laws to develop realistic behavior of objects in the virtual world, and provide means to measure features of movements, such as velocity, acceleration, the change of energy, etc. PDE techniques can be used for geometric modeling applications such as surface fairing, shape reconstruction, geometric design, etc. However, there is a lack of a systematic approach that can provide a set of comprehensive shape modeling toolkits starting from shape design from sketches or reconstruction from scattered datasets, to interactive sculpting as well as model simplification in a single PDE framework. To make full use of the advantages of PDE techniques for geometric modeling, this dissertation introduces a PDE-based modeling system for geometric and physics-based modeling including shape design and direct manipulation, free-form deformation, object reconstruction, and medial axis or skeleton extraction. In the system, objects can be defined as solutions of PDEs with generalized boundary or initial conditions either in explicit parametric domain or implicit working space. The PDE modeling system employs two types of PDEs, elliptic PDEs defining static geometric objects through boundary conditions, and parabolic PDEs modeling dynamic objects via initial values. The functionalities of the elliptic PDE model include 2D and 3D physics-based parametric shape design and sculpting, interactive modeling for arbitrary polygonal meshes or displacements, object reconstruction from arbitrary sketches or unorganized scattered data, shape blending and direct manipulation of implicit PDE objects of arbitrary topology, and free-form solid modeling and deformation with intensity and physical properties. The parabolic PDEs are used for diffusion-based medial axis extraction of geometric objects bounded by arbitrary polygonal meshes, skeleton-based shape sculpting, and shape recovery through front propagation. The PDE modeling system provides various interactive manipulation toolkits for shape editing of the PDE-governed objects satisfying geometric continuities and physical properties. After all, this dissertation offers a general and powerful PDE-based geometric modeling framework toward realizing the full potential of the PDE techniques as modeling and simulating tools in computer graphics.

To my parents and Yinghao

Contents

List of Tables **xii**

List of Figures **xiii**

Acknowledgments **xvii**

Publications **xix**

1 Introduction **1**

1.1 Problem Statement 2

1.2 Contributions 7

1.3 Dissertation Organization 15

2 Review of PDEs and Their Applications **16**

2.1 PDE Definition and Classifications 17

2.2 PDEs for Geometric Modeling 20

2.2.1 PDE Surfaces 20

2.2.2 PDE Solids 23

2.2.3 Generating Fair Meshes 24

2.3 Level Set Method 26

2.3.1 Level Set Formulation 26

2.3.2 Shape Reconstruction Using Level Set Method 28
2.3.3 Level Set Method for Shape Morphing 29
2.4 Diffusion Equations 30
2.4.1 Depicting Gaseous Phenomena Using Diffusion Processes 31
2.4.2 Reaction-Diffusion Systems for Texture Synthesis 32
2.4.3 Visualizing Vector Field 32
2.4.4 Tensor Field Visualization for MRI Data 34
2.5 Other Applications of PDE Techniques 35
2.5.1 Animation and Simulation 35
2.5.2 Image Processing 40

3 Related Work of Other Modeling Techniques 42
3.1 Spline-based Shape Representations 42
3.2 Implicit Models 46
3.2.1 Particle-based Implicit Surface Sculpting 47
3.2.2 Trivariate B-splines for Implicit Models 47
3.2.3 Level Set Method for Implicit Functions 48
3.2.4 Variational Implicit Functions 49
3.3 Physics-based Modeling 50
3.3.1 Deformable Models 51
3.3.2 Applications of Physics-based Techniques 52
3.4 Medial Axis Extraction Techinques 56

4 Physics-based PDE Surfaces 60
4.1 Introduction and Motivation 61
4.2 Formulation of PDE Surfaces 63
4.3 PDE-based Displacement Surfaces 64

4.4 Numerical Approximation Techniques 66

4.5 Combining Physics-based Modeling 69

4.6 Interactive Sculpting Techniques for PDE-based Surfaces 70

4.6.1 Surface Initialization 71

4.6.2 Generalized Boundary Constraints 71

4.6.3 Multi-Grid PDE Surface Subdivision 74

4.6.4 Manipulating Boundary Conditions 75

4.6.5 Modifying Control Function *a(u,v)* 76

4.6.6 Joining Multiple Surfaces 76

4.6.7 Sculpting Tools and Geometric Constraints for Global and Local Deformation 77

5 PDE-based Arbitrary Mesh Modeling 89

5.1 Introduction and Motivation 90

5.2 Direct Manipulation of Arbitrary Mesh Objects 93

5.2.1 Umbrella Operator 93

5.2.2 PDE Formulation for Arbitrary Meshes 95

5.2.3 Direct Manipulation of Arbitrary Meshes based on PDEs . 95

5.3 Diffusion-based Medial Axis Extraction 96

5.3.1 PDE Formulation of Medial Axis Extraction for Arbitrary Meshes 97

5.3.2 PDE-based Skeletonization and Shape Manipulation . . . 102

6 Implicit Elliptic PDE Model 109

6.1 Introduction and Motivation 110

6.2 Elliptic PDE Formulation for Implicit Models 112

6.3 Radial Basis Function 114

6.4 Numerical Simulation 115
6.5 Constrained System 116
6.6 Boundary Conditions for Different Applications 118
6.6.1 Shape Design Using Traditional Boundary Constraints . . 118
6.6.2 Shape Blending 119
6.6.3 Shape Design Using Sketch Curves 120
6.6.4 Shape Reconstruction from Unorganized Scattered Data Points 123
6.7 Sculpting and Manipulation Toolkits for Implicit PDEs 124
6.7.1 Modifying Blending Coefficients 125
6.7.2 Sketch Curve Sculpting 125
6.7.3 Local Manipulation of Implicit PDE Solids 126

7 PDE-based Free-Form Modeling and Deformation 132
7.1 Introduction and Motivation 133
7.2 PDE Solid Formulation 138
7.2.1 Formulation of PDE Solid Geometry 138
7.2.2 Formulation of PDE Solid with Intensity Attributes 140
7.3 Physics-based Modeling of Free-Form PDE Solids 141
7.4 Multi-Grid Approximation for PDE Solid 142
7.5 Interactive Editing Toolkits for Free-Form PDE Solids 143
7.5.1 Solid Geometry Initialization 143
7.5.2 Boundary Manipulation 145
7.5.3 Direct Solid Manipulation 146
7.5.4 Geometric Free-Form Deformation 149
7.6 Intensity-based Free-Form Modeling and Deformation 150
7.6.1 Initialization 150

7.6.2 Free-Form Modeling and Deformation with Intensity Distribution 151

8 Numerical Techniques for PDEs 157

8.1 Spectral Approximations Based on Fourier Transforms 158

8.2 Finite-Element Method 160

8.3 Finite-Difference Method 164

8.4 Iterative Method 166

8.5 Multi-Grid Method 168

8.6 Comparison and Discussion 169

9 System Architecture and Results 171

9.1 Physics-based PDE Surface Module 172

9.1.1 Overview 172

9.1.2 Dynamic PDE Surface Modeling Functionalities 173

9.1.3 System Performance 175

9.2 PDE-based Arbitrary Mesh Modeling Module 178

9.2.1 Overview 178

9.2.2 PDE-based Manipulation Toolkits for Arbitrary Meshes . 179

9.2.3 Discussion of Diffusion-based Medial Axis Extraction . . 181

9.3 Direct Sculpting Environment of Implicit PDEs 184

9.3.1 Overview 184

9.3.2 Implicit PDE Modeling Toolkits 185

9.3.3 System Performance and Discussion 188

9.4 PDE-based Free-Form Modeling and Deformation Structure and Results 190

9.4.1 Overview 190

9.4.2 PDE Solid Modeling Toolkits 191
9.4.3 Performance 193

10 Conclusion 196

10.1 Future Work 198

Bibliography 219

List of Tables

2.1 Illustration of the relations between the two types of PDE classifications 19

9.1 CPU time of different solvers for a PDE surface with different sampling rates 176

9.2 CPU time of different iterative methods combined with multi-grid subdivision 176

9.3 Number of iterations for various manipulation techniques 177

9.4 The parameters of physical properties for dynamic surface examples 177

9.5 CPU time in seconds for medial axis extraction. 182

9.6 CPU time of different solvers for implicit PDE objects 188

9.7 CPU time (seconds) of local direct manipulation examples of implicit PDE objects 189

9.8 CPU time for PDE solid examples using different solvers 194

List of Figures

1.1 Contributions of the PDE-based modeling system 8

2.1 Classification of PDEs based on Characteristics 18

2.2 The illustration of another type of PDE classification 19

2.3 The PDE surface with Hermite-like boundary conditions 21

2.4 Construct 3D solids from parametric space using PDE method . . 24

2.5 Example of variational surface fairing 26

2.6 Examples of shape reconstruction using level set method 29

2.7 Using level set method for shape morphing 30

2.8 Texture buttons generated by reaction diffusion system 33

2.9 Visualizing principal curvature directions 34

2.10 Texture-mapping with reaction-diffusion texture 35

2.11 Example of modeling fracture 36

2.12 Example of hot turbulent gas simulation 38

2.13 Examples of Explosion Simulation 40

2.14 Examples of Image Inpainting 40

3.1 Variational implicit shape reconstruction 50

3.2 2D illustration for medial axis 56

4.1 Illustration of the idea of displacement model 65

4.2 Mass-spring network for the discretized PDE surface 70

4.3 PDE surfaces from Hermite-like boundary conditions 72
4.4 PDE surfaces with Coons-like and Gordon-like boundary conditions 73
4.5 The multi-grid solution for a PDE surface 75
4.6 Changing boundary conditions of a PDE surface 75
4.7 Effects of changing blending coefficient $a(u, v)$ 76
4.8 Examples of point-based manipulation on PDE surfaces 79
4.9 Illustration of normal change approximation 80
4.10 Illustation of approximating curvation modification 81
4.11 Curve editing 82
4.12 Region sculpting 83
4.13 Surface manipulation with fixed regions 84
4.14 Trimming on the selected regions 84
4.15 Examples for displacement PDE surface sculpting 85
4.16 Examples for curve and region editing on PDE surface displacements 86
4.17 B-spline approximations for PDE surfaces 87
4.18 The PDE surface sculpting with physical constraints 87
5.1 Medial axis extraction using the proposed PDE technique 93
5.2 Umbrella operators 94
5.3 An example of PDE model of arbitrary meshes 96
5.4 The evaluation of Gaussian curvature for a mesh point 102
5.5 An example of PDE-based medial axis extraction for arbitrary meshes 103
5.6 Local PDE-based medial axis extraction 105
5.7 User-defined PDE-based medial axis extraction 106
5.8 An example of skeleton-based shape sculpting 107
5.9 Another example of skeleton-based shape sculpting 107

5.10 A sequence of deformed shapes through skeleton-based shape sculpting 107
5.11 Examples of skeleton extraction with different value of curvature thresholds 108
6.1 2D illustrations of different types of boundary conditions. 117
6.2 Implicit PDE objects generated from cross-sectional boundary conditions 119
6.3 Shape blending using implicit PDEs 120
6.4 Examples of shape reconstruction from sketch curves 121
6.5 Illustration of computing the gradient direction 122
6.6 Example for performing the RBF initialization locally 123
6.7 Examples of shape reconstruction from scattered data points . . . 124
6.8 Examples of enforcing curve and direct manipulation constraints . 130
6.9 Examples for manipulating gradient directions of sketch curves . . 131
6.10 Examples of enforcing gradient constraints 131
6.11 Examples of enforcing gradient and curvature constraints 131
7.1 Mass-spring network for 3D 141
7.2 PDE solids generated from given boundary surfaces 144
7.3 Examples of PDE solids subject to boundary curve network 145
7.4 Modifying PDE solids via curve constraints of boundary surfaces . 146
7.5 Examples of direct modification on trimmed PDE solids 147
7.6 Directly modify a point on the trimmed PDE solid 148
7.7 Examples using CSG trimming operation in a PDE solid 148
7.8 Free-form deformation based on PDE solids 149
7.9 Examples of intensity initialization of PDE solids 151
7.10 Iso-surface deformation 152
7.11 Shape blending by integrated PDE solid with intensity 153

7.12 PDE-based free-form shape blending and deformation 154
7.13 PDE-based free-form deformation due to intensity modification . . 155
7.14 PDE-based free-form deformation by intensity modification . . . 156
8.1 Basic finite elements 161
8.2 The point discretization for finite-difference method 164
8.3 The illustration of discretizing differential operator for $\{i, j\}$ . . . 167
9.1 System architecture for PDE-based modeling system 172
9.2 System interface for the physics-based PDE surface module 173
9.3 System architecture of the physics-based PDE surface module . . 174
9.4 System interface for the PDE-based arbitrary mesh modeling module 179
9.5 System architecture for PDE-based arbitrary mesh modeling module 180
9.6 System interface for the implicit PDE module 184
9.7 System architecture for implicit PDE module 186
9.8 System interface for the PDE-based free-form modeling and deformation module 191
9.9 System architecture of the PDE-based free-form modeling and deformation module 192

Acknowledgments

First and foremost, I am deeply grateful to my advisor, Professor Hong Qin, for his guidance for the research topics and directions, and for his invaluable advices and countless efforts guiding me during my research. I want to express my sincere gratitude to Professors Arie Kaufman, Klaus Mueller, Dimitris Samaras, and Michael Ashikhmin, who provided insightful advices and suggestions and served on various committees associated with my graduate studies. I also thank Professor Gabriel Taubin for taking the time to serve as the external member of my dissertation committee.

The members of Center for Visual Computing offer me tremendous help during my study and research. I want to thank Kevin McDonnell, Ye Duan, Jing Hua, Hui Xie, and Wei Hong for their helpful suggestions during my dissertation research. And many thanks to many of other members of the lab, especially Xiaoming Wei, Nan Zhang, Sumantro Ray, and Chris Carner for their help. The members of the excellent support staff in the Computer Science Department were also of great assistance, especially Bin Zhang, Brian Tria, Pat Tonra, Ashwin Nagrani, Anne Kilarjian, Stella Mannino, Betty Knittweis, Kathy Germana and Edwina Osmanski. I extend my gratitude to the faculty and students of the Department, who over the years have helped me in many ways.

My research was supported in part by the following grants awarded to Professor Qin: NSF CAREER award CCR-9896123, the NSF grants IIS-0082035 and IIS-0097646, the NFS grant CCR-0328930, the NSF ITR grant IIS-0326388, Alfred P. Sloan Fellowship, and Honda Initiation Award.

I also thank my parents for their understanding, inspiration, and constant support. I would not be here without them. Finally, and most importantly, I thank my husband, Yinghao, for his endless love, cheerful encouragement, and perpetual support. He stands by me and supports me all the time. When I need him, he is always there. I would not be doing my Ph.D. study without his encouragement and inspiration. My world would not be complete without him. This dissertation is dedicated to my parents and Yinghao.

Publications

Haixia Du and Hong Qin. Dynamic PDE-Based Surface Design Using Geometric and Physical Constraints. Accepted by *Graphical Models*, 2003.

Haixia Du and Hong Qin. A Shape Design System Using Volumetric Implicit PDEs. Accepted by *CAD Special Issue of the ACM Symposium on Solid Modeling and Applications, 2003*, to appear.

Haixia Du and Hong Qin. Free-Form Solid Modeling by Integrating Parametric and Implicit PDEs. Under review for journal publication, 2004.

Haixia Du and Hong Qin. PDE-based Skeletonization and Propagation for Arbitrary Topological Shapes. In preparation for journal submission, 2004.

Haixia Du and Hong Qin. Medial Axis Extraction and Shape Manipulation of Solid Objects Using Parabolic PDEs. Accepted by *The Nineth ACM Symposium on Solid Modeling and Applications 2004*.

Haixia Du and Hong Qin. Interactive Shape Design Using Volumetric Implicit PDEs. In *Proceedings of The Eighth ACM Symposium on Solid Modeling and Applications 2003, University of Washington, Seattle, WA*, pages 235-246, June 2003.

Haixia Du and Hong Qin. Integrating Physics-based Modeling with PDE Solids for Geometric Design. In *Proceedings of Pacific Graphics 2001, Tokyo, Japan*, pages 198-207, October 2001.

Haixia Du and Hong Qin. Dynamic PDE Surfaces with Flexible and General Constraints. In *Proceedings of Pacific Graphics 2000, Hong Kong*, pages 213-222, October 2000.

Haixia Du and Hong Qin. Direct Manipulation and Interactive Sculpting of PDE Surfaces. In *Proceedings of EuroGraphics 2000, Interlaken, Switzerland*, pages C261-C270, August 2000.

Chapter 1

Introduction

Partial Differential Equations, often known as PDEs, can be used to describe the physical characteristics of objects and natural phenomena in the real world using their differential properties. PDEs have been extensively employed in various visual computing applications, such as simulation, image processing, visualization, computer vision, etc. On the other hand, geometric modeling techniques are fundamental to many visual computing fields such as computer graphics, CAD/CAM, animation, and virtual environments. Previous work indicates that certain types of PDEs provide an alternative way to model geometric objects. However, the full modeling potential of geometric PDE techniques has not been realized. This dissertation aims at fully utilizing the advantages of geometric PDE techniques and presents a prototype system for design, reconstruction, simplification, and manipulation of geometric models in a single framework.

In general, modeling techniques to represent geometric objects fall into two categories, explicit models and implicit models. Explicit models represent objects by stating their precise positions, *e.g.*, coordinates of points belonged to the objects and relations among them. Parametric models and subdivision models are among popular explicit representations. Parametric models define geometric objects through building the correspondence between the parametric domain (***e.g.***, (u, v, w)) and physical space (***e.g.***, (x, y, z)), such as free-form splines [20, 35, 54, 56, 111, 112], and parametric PDE techniques [12, 13, 15, 91, 160]. Subdivision techniques directly model geometric objects through points, edges, as well as faces of the objects. There are various subdivision schemes to model geometric entities [26, 28, 36, 42, 52, 64, 66, 74, 77, 80, 86, 88, 89, 126, 133, 150, 163, 164]. Implicit models form the other category of geometric modeling techniques by representing the geometric shapes as level-sets that collect points in the physical space whose scalar value satisfies certain scalar field functions [9, 11, 23, 25, 29, 39, 37, 55, 61, 67, 99, 101, 110, 118, 120, 124, 145, 146, 153, 154, 162]. PDE techniques, in contrast, employ partial differential equations to model geometric shapes with boundary and initial conditions. They provide an alternative way of geometric modeling by representing objects as solutions of certain PDEs which can satisfy functional requirements and physical attributes automatically.

While each modeling approach exhibits certain advantages that make it useful for particular applications, different method requires different representations for geometric objects and focus on different modeling tasks respectively. Overall, none of these techniques is general enough to support interactive design and manipulation of topologically complex objects under different types of shape representations.

1.1 Problem Statement

PDE techniques are widely used in several visual computing areas, such as fluid simulation [58, 59, 60, 53, 73, 156, 158], visualization and texture synthesis [76, 143, 144, 155], and image processing[7, 109], etc. In addition, PDE methods offer an alternative but natural way to model geometric shapes, including both explicit and implicit models. They define and govern geometric objects by certain partial differential equations with given boundary/initial conditions.

In comparison with other graphical techniques, PDE methods have many attractive advantages:

- Natural physical processes are frequently characterized by PDEs. In principle, PDE models can be controlled by physical laws, so they are natural and much closer to the real world. PDE techniques are potentially ideal candidates for both design and analysis purposes.

- The formulation of differential equations is well-conditioned and technically sound. Smooth objects with high-order continuity requirements can be readily defined through certain PDEs.

- Smooth objects that minimize certain energy functionals oftentimes are associated with differential equations, hence optimization techniques can be unified with PDE models.

- Many powerful numerical techniques to solve PDEs are commercially available. Parallel algorithms can be employed for large-scale problems in industrial settings. Recently, the use of GPUs to solve PDEs provides even faster solution and makes PDEs applicable for real-time applications.

- Users can easily understand the underlying physical process associated with PDEs. Therefore, it is possible to implement intuitive and natural control through the modification of physical parameters in PDE models.

- PDE techniques can potentially unify both geometric and physical aspects of objects. They are invaluable throughout the entire modeling, design, analysis, and manufacturing processes. Various heterogeneous requirements can be enforced and satisfied simultaneously.

When formulated as boundary value problems, PDEs can be used for free-form surface modeling, shape blending, and functional surface design with given boundary conditions [12, 13, 15, 91]. Furthermore, since the energy minimization can be represented by certain PDEs, people also use PDE techniques or combine PDE methods with other modeling techniques, such as subdivision models, to generate fair surfaces with specified boundary constraints [123].

Although previous geometric PDE techniques have unique modeling advantages such as small number of control coefficients and satisfying functional requirements, they are lack of interactive and realistic sculpting and direct manipulations for geometric models. The shape deformation can only be obtained by changing boundary conditions and control coefficients for static PDEs. Such modifications only offer global control but cannot modify the shape locally. In previous work of geometric PDE modeling, only Hermite-like boundary constraints, which consist of boundary information and their derivative values, are enforced to define PDE objects. More flexible boundary conditions haven't been considered yet. This extremely limits the topological variety of geometric objects that can be modeled by PDE techniques. Furthermore, pure geometric techniques are usually non-intuitive and laborious for the sculpting and modeling of graphical objects. The geometry only has static information and does not produce realistic behavior for objects under interactive manipulations. It doesn't provide dynamic and interactive framework for time-dependent modeling requirements.

In summary, there are several modeling shortcomings that limit modeling capabilities of PDE techniques:

- **Lack of direct and interactive manipulation for PDE objects**

The elliptic PDEs previously used for geometric models only solve boundary value problems. The deformation and manipulation of PDE-governed objects are often obtained by boundary sculpting and global modification of control coefficients. There are lack of direct and interactive sculpting tools for PDE objects.

- **No arbitrary constraints for geometric PDE objects**

  In previous work of parametric PDE techniques, people mainly concentrate on seeking for analytic solutions of elliptic PDEs. Although analytic solutions are fast from performance point of view, they only take information at objects' boundaries to solve PDEs to obtain the geometry information of objects. Hence shape deformation can be only obtained through boundary manipulations and global control coefficients. The more desirable arbitrary manipulations of any part of the objects cannot be enforced, which can be achieved through numerical techniques for PDEs with ease.

- **Lack of local control for regional shape sculpting**

  The PDEs employed in conventional geometric PDE techniques are usually defined over the entire parametric domain and can model objects with global features. According to the mathematical properties of the PDEs, any modification on PDE objects will propagate to the entire shape. Therefore, only global control of objects can be achieved, but local shape modifications in small bounded regional areas of objects haven't been considered yet. This limits the flexibility of PDE shape manipulation.

- **No intuitive manipulation with physical properties**

  Geometric PDE techniques make use of static PDEs to model geometric information of objects. Physical and material properties that contribute to objects' realistic behavior are lacking from previous work. Therefore, realistic sculpting and real-time deformation couldn't be achieved easily.

- **Lack of direct manipulations for implicit PDE models**

  Although the level set method employs a time evolution PDE to model implicit models through speed functions, manipulations through speed functions are generally non-intuitive for common users. In addition, applying manipulation tools directly on intensity scalar fields for implicit objects is still under-explored.

- **Limitations of acceptable shape representations for PDE models**

  In prior work, geometric PDE techniques often support limited types of shape representations for objects to be modeled. They usually model objects of a single specific type of represnetations. There's a lack of general PDE modeling method which can model geometric entities of different popular shape structures in a single framework, including parametric objects, arbitrary polygonal meshes, as well as implicit models.

- **Lack of integration framework of different types of PDEs**

  Different PDEs have their own modeling strengths and limitations. A general PDE modeling framework which can incorporate advantages of different types of PDEs into a single system is still under-explored.

- **Application limitations of geometric PDE modeling system**

  Previously, a PDE method is often focusing on certain specific tasks and applications especially in geometric modeling. A PDE-based modeling system with a comprehensive set of manipulation toolkits which can offer various modeling functionalities such as shape design, object reconstruction, model sculpting, shape deformation, as well as simplification of geometric models associated with physical attributes in a single package hasn't been developed.

In general, it's more desirable to have a unified PDE-based modeling framework which can provide modeling capabilities of geometric objects of various types of shape representations and offer powerful design and manipulation functionalities for realistic and interactive control of objects.

1.2 Contributions

This dissertation mainly focuses on generalizing PDE techniques with interactive sculpting toolkits for various geometric tasks to maximize modeling potentials of PDE techniques for geometric models. It presents a PDE-based modeling system which integrates several types of PDEs with physics-based techniques, shape skeletons, front propagations, and free-form deformation to model objects of various types of shape representations including parametric models, arbitrary polygonal meshes, displacement maps, and implicit functions in a single framework. The system incorporates modeling advantages of these techniques and shape representations to make PDE techniques more powerful and attractive for geometric design and manipulation. It forms a general PDE-based modeling paradigm which employs elliptic and parabolic PDEs with numerical techniques for geometric and physics-based modeling and direct sculpting. The PDE-based modeling system can design objects from arbitrary curve sketches, reconstruct shape from unorganized scattered data points, interactively sculpt surfaces and displacements with physical properties, extract diffusion-based medial axes or skeletons of arbitrary polygonal meshes, manipulate and recover shapes using

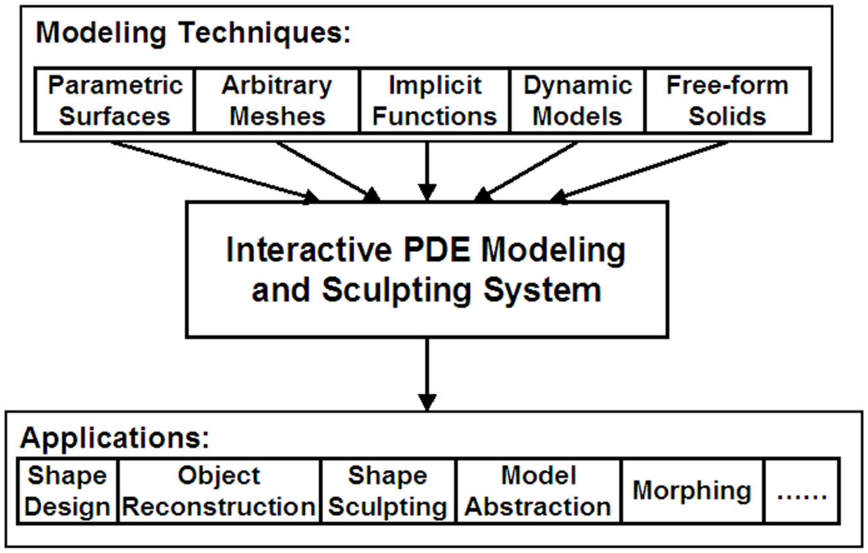


Figure 1.1: Contribution summary of the PDE-based modeling system.

skeleton-based techniques, model implicit objects with intuitive local operations, as well as model solid objects by direct manipulation and free-form deformation toolkits for both geometry and intensity attributes. The contribution of this dissertation is summarized in Fig. 1.1.

In particular, the contributions of this dissertation include:

- **Intuitive manipulation of objects of different types of shape representations**

  It develops a prototype geometric design and modeling system using PDE techniques and physics-based methods to model objects of different types of shape representations including parametric surfaces and solids, arbitrary polygonal meshes, and implicit models defined by scalar intensity fields. The system provides powerful and intuitive sculpting toolkits which are suitable for general users to manipulate geometric objects. It offers modeling advantages of both explicit and implicit models into a single geometric modeling framework.

- **Unified design and modeling platform**

  The prototype system defines geometric (parametric or implicit) objects as solutions of certain PDEs by given generalized boundary constraints including curve networks, sketches, and unorganized scattered data points. Moreover, by incorporating physical properties and material attributes with geometric information of objects into the system, designers can intuitively sculpt PDE objects by interactively modifying the shape and material properties of objects that enforces additional geometric and implicit constraints. It offers users a natural way to design and deform geometric shapes.

- **Realistic simulation of objects' behavior with physical properties**

  This dissertation presents a unified approach to couple physics-based methods and parametric PDE surfaces and solids. The integrated physics-based PDE model offers interactive sculpting of PDE objects with physical attributes such as mass, damping, stiffness, density, and other material properties. The behavior of objects under direct manipulation is governed by the integrated formulation. It provides more realistic motion simulation of real world objects than previous pure geometric PDE models.

- **Integrating implicit functions with parametric PDEs**

  It introduces a novel method which tightly couples parametric PDE techniques with implicit functions for interactive shape design and manipulation of PDE-based volumetric implicit models embedded in an implicit PDE working space. The governing elliptic PDE provides high-order intensity continuities inside the implicit objects without additional tedious specifications. It offers both direct and indirect manipulations of implicit objects for both global and local deformations. It also provides a solution for parametric PDE techniques to model objects of arbitrary topology. The integration of parametric PDE and implicit intensity field provides both geometry and intensity of the object simultaneously. It offers more degrees of freedom for shape manipulation in geometric modeling applications.

- **Defining geometric entities with small number of coefficients**

  Because objects are defined as solutions of certain type of PDEs with given boundary constraints, the interior of objects can be automatically obtained through their differential properties. This generalized PDE modeling technique is able to recover the full information of objects from partial input, which elevates the burden of specifying the large quantity of constraints for complete datasets. In addition, it also provides smooth deformation of objects during manipulation and sculpting because of the properties of differential equations.

- **Diffusion-based medial axis manipulation for simplification and animation**

  The system also employs the diffusion-based equation for medial axis or skeleton extraction as well as skeleton-based shape manipulation and reconstruction through front propagations. It offers an alternative but natural way for medial axis extraction for commonly used 3D objects bounded by polygonal meshes. The skeleton-based shape manipulation provides a fast and easy way for animation and deformation of complicated geometric objects. It further broadens applications of the PDE modeling system and provides a geometric modeling paradigm which includes design, reconstruction, sculpting, and simplification for arbitrary topological shapes under various data structures.

More specifically, the PDE modeling system includes following components:

- **Physics-based PDE surfaces and displacements** (Chapter 4)

  The PDE modeling system is able to model surfaces and surface displacements using parametric PDE formulation with integrated physical properties for direct manipulations on the surfaces[43, 44, 46]. It allows interactive design of surfaces of flexible topology as PDE surfaces or PDE displacements on the original surfaces using generalized boundary conditions as well as a variety of geometric and physical constraints. The physics-based parametric PDE supports various interactive techniques beyond the conventional boundary control. The sculpting toolkits allow users to interactively modify arbitrary point, curve span, and/or regions of interest across the entire PDE surfaces/displacements in an intuitive and physically meaningful way. To achieve real-time performance, this dissertation employs several simple, yet efficient numerical techniques, including the finite-difference discretization, the multi-grid subdivision, and the mass-spring approximation of elastic PDE surfaces/displacements. In addition, the dynamic PDE surfaces/displacements can also be approximated using standard bivariate B-spline finite elements, which can be subsequently sculpted and deformed directly in real-time subject to intrinsic PDE constraints.

- **Arbitrary topological PDE surface manipulations** (Chapter 5)

  However, traditional parametric PDE surface model can only model surfaces defined on regular parametric domain which cannot deal with arbitrary topological objects and extremely limit the modeling potential of PDE techniques. Because a large number of existing models are represented by arbitrary polygonal meshes, the PDE formulation is also considered to model arbitrary meshes to further extend the modeling coverage of the PDE modeling system. The PDE is formulated based on pre-defined mesh models to govern shape deformation. The PDE modeling system provides manipulation toolkits for interactive sculpting and facilitates data exchange with other geometric modeling techniques.

- **Diffusion-based medial axis extraction and skeleton-based manipulation** (Chapter 5)

  Medial axes or skeletons are very useful in medical image processing and analysis. They can also be used for animations of complicated objects. In previous work, certain PDE formulations such as Hamilton-Jacobi equations can detect medial axes of 2D images and volumetric data with ease. This dissertation expands the use of diffusion equations to detect medial axes of arbitrary 3D objects represented by polygonal meshes based on their differential properties. It offers an alternative but natural way for medial axis extraction for commonly used 3D objects bounded by polygonal meshes. By solving the PDE along the time axis, the system can not only quickly extract diffusion-based medial axes of input meshes, but also allow users to visualize the progressive extraction process at each time step. In addition, diffusion equations not only can be used for medial axis extraction, but also provide a way to reconstruct objects from skeletons, which enables the PDE model to provide users a set of manipulation toolkits for skeleton-based shape deformation by sculpting extracted medial axes, then using diffusion-based techniques to recover corresponding deformed shapes according to the original input datasets. This skeleton-based shape manipulation offers a fast and easy way for animation and deformation of complex objects.

- **Shape modeling with implicit PDEs** (Chapter 6)

  Considering the modeling advantages of implicit functions, the PDE formulation is further employed to model intensity fields which can reconstruct and interactively manipulate implicit objects from scattered data points or arbitrary sketches in 3D space. In particular, the unified approach can reconstruct the PDE geometry of arbitrary topology from scattered data points or a set of sketch curves. Elliptic PDEs are used to define the volumetric implicit function. The implicit PDE model has the capability to reconstruct a complete model from partial information and facilitates the direct manipulation of underlying volumetric datasets via sketch curve manipulation, iso-surface sculpting, deformation of arbitrary interior regions, CSG operations, and gradient and curvature manipulations inside the working space. The prototype system allows designers to interactively sketch the curve outlines of the object, define intensity values and gradient directions, and specify interpolatory points in the 3D working space. The governing implicit PDE treats these constraints as generalized boundary conditions to determine unknown scalar intensity values over the entire working space. The implicit shape is reconstructed with specified intensity value accordingly and can be deformed using a set of sculpting toolkits.

- **PDE-based free-form modeling and deformation** (Chapter 7)

  A 3D parametric elliptic PDE forms a PDE-governed 3D space, which can be naturally applied for free-form deformation of geometric shapes. In addition, because of the 3D PDE space can be defined by boundary surfaces surrounding the space, it provides a means of boundary representations of solid objects. According to these modeling properties, this dissertation extends the PDE techniques to 3D parametric domain to model solid objects with physical properties which constructs a mapping from 3D parametric space to 3D physical space and provides PDE-based free-form deformation of explicit objects defined in the 3D PDE parametric domian. The physics-based PDE solid formulation and its associated dynamic principle permit designers to model parametric PDE solids through free-form deformation and direct manipulation. The behavior of a physics-based PDE solid is natural and intuitive subject to imposed constraints. Users can easily model and interact with solids of complicated geometry and/or arbitrary topology from locally-defined PDE primitives through trimming operations. The PDE-based free-form solid modeling technique offers users various sculpting toolkits for solid design and manipulation, allows them to interactively modify the physical and geometric properties of arbitrary regions of interest on boundary surfaces, as well as any interior parts of modeled objects. In addition, implicit properties such as intensity attributes are incorporated into the PDE space. This integration can model geometry and intensity information of objects simultaneously. It provides a novel way for arbitrary shape modeling, blending, and deformation. The PDE-based solid offers not only a type of solid representation, but also more degrees of freedom for object manipulations with intensity attributes.

1.3 Dissertation Organization

The remainder of this dissertation is structured as follows. Chapter 2 gives a brief introduction of PDEs and reviews the prior work of PDE-based techniques and their applications. Chapter 3 reviews the related work of other popular techniques such as spline-based models, implicit functions, physics-based methods, medial axis extraction techniques, etc. Current work of this dissertation will be detailed in the following chapters. Chapter 4 presents the novel techniques of the direct manipulation of dynamic PDE surfaces and displacements with generalized boundary constraints and flexible topology. Chapter 5 introduces the extended PDE surface modeling scheme for manipulation of arbitrary topological mesh objects. It also presents diffusion-based medial axis extraction for objects bounded by arbitrary polygonal meshes, and shows how to manipulate such objects using skeleton-based sculpting and recover deformed shapes through diffusion-based front propagation techniques. Chapter 6 details the modeling functionalities and applications for implicit PDE objects. Chapter 7 introduces the sculpting toolkits for PDE-based free-form modeling and deformation for solid objects with physical and scalar intensity attributes. Chapter 8 introduces several efficient numerical methods to solve PDEs. The system structure of the prototype PDE system and experimental results are presented in Chapter 9. Finally Chapter 10 concludes this dissertation and outlines possible future research directions.

Chapter 2

Review of PDEs and Their Applications

Partial differential equations (PDEs) are at the heart of many computer analysis models or simulations of continuous physical systems, such as fluids, electromagnetic fields, the human body, and so on. Diffusion equation, wave equation, Laplace's equation, heat equation, as well as the equations of fluid dynamics, *i.e.*, Navier-Stokes equations, are all popular used PDEs [159] to model and simulate the physical world in visual environment. In addition, most of physics-based modeling techniques and many CAD/CAM applications are related to certain PDEs. PDE techniques are playing more and more important roles in the entire computer graphics area in recent years. This chapter will briefly introduce typical PDEs with classifications and their properties and applications in geometric modeling, visualization, and simulation, etc.

2.1 PDE Definition and Classifications

A partial differential equation (PDE) is an equation involving functions and their partial derivatives. (2.1) shows an example of r^{th}-order PDE over 2D parametric domain of u and v.

$$\sum_{n=0}^{r} \sum_{l,m\geq 0}^{l+m=n} \alpha_{l,m}(u,v) \frac{\partial^n}{\partial u^l \partial v^m} f(u,v) = g(u,v), \tag{2.1}$$

where $\alpha_{l,m}(u,v)$ and $g(u,v)$ are control functions, and $f(u,v)$ is the unknown function of u and v.

In most books of mathematics, PDEs are classified into three categories, *hyperbolic*, *parabolic*, and *elliptic*, on the basis of their *characteristics*, or curves of information propagation. Given a second-order PDE over 2D domain as an example:

$$A\frac{\partial^2 x}{\partial x^2} + B\frac{\partial^2 u}{\partial y^2} + C\frac{\partial^2 u}{\partial x \partial y} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = G,$$

the classification works as follows:

When $B^2 - AC > 0$, the PDE is of *hyperbolic* type, and the prototypical example is the one-dimensional *wave equation* (*e.g.*, for the coupled harmonic oscillators):

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2}. \tag{2.2}$$

When $B^2 - AC = 0$, the PDE is of *parabolic* type, and the prototypical equation is the *diffusion equation* (*e.g.*, for heat or for ink):

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(D\frac{\partial u}{\partial x}), \tag{2.3}$$

where D is the diffusion coefficient.

Finally, when $B^2 - AC < 0$, it's an elliptic PDE and the prototypical equation is the *Poisson equation* (*e.g.*, for electric fields or for fluid flow):

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x,y), \tag{2.4}$$

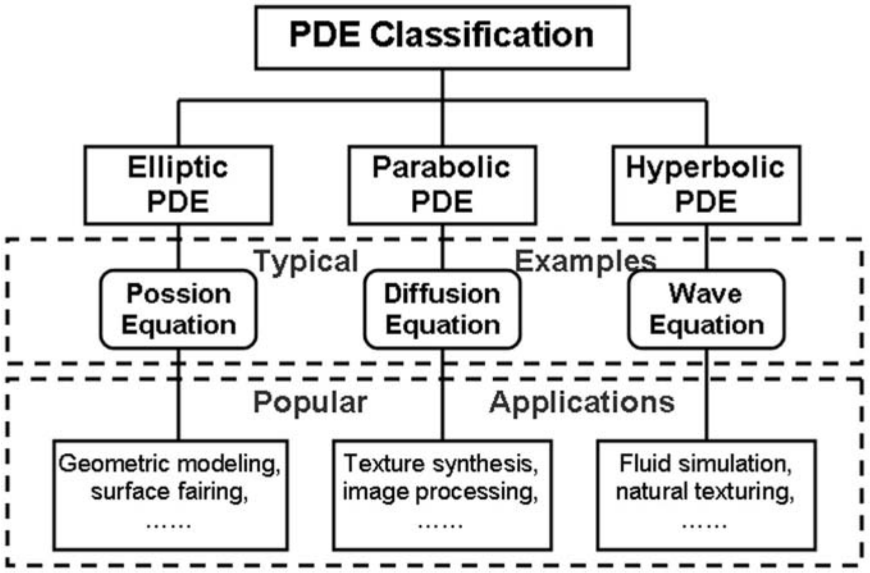
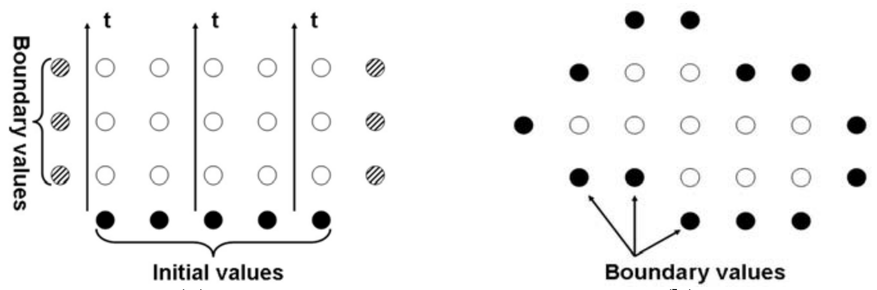


Figure 2.1: Classification of PDEs based on Characteristics.

where the source term ρ is given. If the source term is equal to zero, the equation is *Laplace's equation* [113]. Fig. 2.1 shows typical examples of such classification and some of their applications in computer graphics.

However, from a computational point of view, the above classification to distinguish PDEs into these three canonical types is not very meaningful – or at least not as important as some other essential distinctions. There is another classification to distinguish different types of PDEs, by the solution type of a PDE. If the PDE can be solved by given information at some initial time t_0, then the solution is propagating forward in time, it's called *initial value* or *Cauchy* problem, and leads to a time evolution solution. The above mentioned wave equation and diffusion equation are usually this type of PDEs. In contrast, the Poisson equation directs us to find a single "static" function $u(x, y)$ which satisfies the equation within some (x, y) region of interest, and which has some desired behavior on the boundary of the region. This type of problems are called *boundary value* problems and can be



(a) (b)

Figure 2.2: The illustration of another type of PDE classification. (a) Initial value problems; (b) boundary value problems.

PDE Classifications	Elliptic	Hyperbolic	Parabolic
Boundary Value	Laplace's Equation	N/A	N/A
Initial Value	N/A	Wave Equation	Diffusion Equation

Table 2.1: Illustration of the relations between the two types of PDE classifications.

used for geometric design. Fig. 2.2 illustrates the distinction between these two classes of problems. In Fig. 2.2 (a), initial values are given in one time slice, and it is desired to advance the solution in time, computing successive rows of open dots in the direction shown by the arrows. Boundary conditions are shown at the left and right edges of each row. On the other hand, boundary values in Fig. 2.2 (b) are specified around the edge of the grid's boundaries, and the solution is to find the values of all internal points. However, in many occasions, a PDE problem cannot be simply distinguished as of boundary-value or initial-value type, but as a combination of these types of problems.

Table 2.1 summarizes typical PDEs of these two types of classifications.

2.2 PDEs for Geometric Modeling

Elliptic PDEs are often used to model static boundary problems in geometric modeling. In 1989, Bloor and Wilson [12, 13] pioneered a novel technique that defines smooth surfaces as solutions of *elliptic* PDEs. Since the initial application for surface blending, this PDE technique has broadened its uses in surface description, solid modeling, and functional design in recent years.

2.2.1 PDE Surfaces

In principle, the PDE-based method proposed by Bloor and Wilson is a boundary value problem using an elliptic PDE, which has the property that most of the information defining a surface comes from its boundary curves. This permits a smooth surface to be generated and controlled by very few parameters such as boundary conditions and global coefficients associated with the elliptic PDE. The following equation is the fourth-order elliptic PDE introduced in [13] for PDE surfaces:

$$(\frac{\partial^2}{\partial u^2} + a^2\frac{\partial^2}{\partial v^2})^2\mathbf{X}(u,v) = \mathbf{0} \tag{2.5}$$

where u, v are parametric coordinates over 2D space, a is a blending coefficient that controls the behavior of PDE surfaces along parametric directions, and

$$\mathbf{X}(u,v) = \left[\begin{array}{ccc} x(u,v) & y(u,v) & z(u,v) \end{array}\right]^\top$$

defines the PDE surface coordinates in 3D.

(2.5) is based on *biharmonic* equation $\nabla^4\phi = 0$. The reason to choose the elliptic PDE to model surfaces is because that such equations give smooth solutions subject to boundary conditions. The required boundary conditions to find the solution are usually given in terms of the specified variation of the function and/or its normal derivatives along edges of the parametric domain over which the solution is to be found, *i.e.*, Hermite-like boundary conditions, which define the boundaries of the surface.

The four boundary conditions in their work are Hermite-like boundary conditions and comprise two curves which define a pair of the curved surface boundaries at the opposite side along u-direction and a pair of their associated derivative curves defining gradient information at the two boundaries. They are of the following form:

$$\begin{aligned}&\mathbf{X}(0,v)=\mathbf{c}_0(v), \mathbf{X}(1,v)=\mathbf{c}_1(v),\\ &\tfrac{\partial\mathbf{X}}{\partial u}(0,v)=\mathbf{d}_0(v), \tfrac{\partial\mathbf{X}}{\partial u}(1,v)=\mathbf{d}_1(v).\end{aligned} \tag{2.6}$$

This PDE formulation can be used to solve blending problems in CAD/CAM [12, 13], because the above boundary conditions contain position and gradient information of the boundaries and the fourth-order PDE provides tangential continuities inside blending parts.

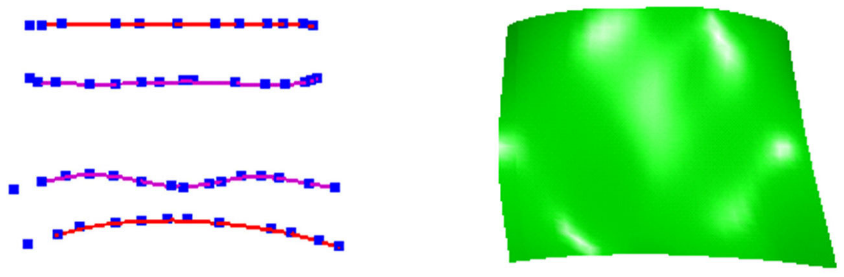

(a) (b)

Figure 2.3: The PDE surface with Hermite-like boundary conditions. (a) B-spline boundary conditions with control points; (b) the surface subject to (a).

Later on, Bloor and Wilson [15] extended the application of this PDE technique to generate piecewise free-form surfaces. By varying boundary conditions and control coefficients in PDEs, as well as connecting several PDE patches together, designers can obtain various surface shapes. Fig. 2.3 shows an example of such PDE surfaces. The control of PDE surfaces is different from conventional methods, and it offers the advantage by defining the surface in terms of a smaller number of variables.

Bloor and Wilson [14] have also developed an algorithm that approximates PDE surfaces using standard B-splines by the method of collocation. They use boundary conditions to solve control points of B-spline approximations, then use them to obtain approximate solutions of PDE surfaces. This work demonstrates that PDE surfaces are virtually compatible with other matured and well established spline-based techniques for surface design, hence PDE surface data can be readily incorporated into existing commercial design systems.

Furthermore, Lowe, Bloor and Wilson [91] presented a method with which certain engineering design criteria such as functional constraints can be incorporated into the geometric design of PDE surfaces. Therefore, it may simultaneously introduce geometric constraints, aesthetic criteria, and physical and engineering restrictions into the design process. This method of surface generation is suitable for the problem of optimum design.

For certain simple boundary conditions, the elliptic PDEs can be solved analytically, *i.e.*, PDE surfaces in these cases have a closed-form formulation that frequently involves functions of Fourier series. However, for general boundary conditions, a PDE solution will have to be sought numerically instead. Bloor and Wilson [17] have derived a set of approximate analytic solutions for PDEs subject to more general boundary conditions. The approximate solution can be made to approach the true solution up to any degree of accuracy. Their generic solutions can be decomposed into a finite sum of Fourier functions satisfying PDEs with an additional ’corrector’ term that satisfies boundary conditions.

In 1999, Ugail *et al.*[148] have developed some techniques for interactively defining and changing boundary conditions to construct PDE surfaces. We [43, 44] have integrated physics-based techniques with static PDE surfaces and developed a set of interactive and direct manipulation toolkits for dynamic PDE surfaces. Later on, we [46] further extended the PDE technique to model surface displacements for more general surface manipulations.

2.2.2 PDE Solids

In 1993, PDE solids were formulated in terms of parametric bounding surfaces by Bloor and Wilson [16], which further expands the geometric coverage of PDE methodology. Different from the traditional solid modeling methods like Constructive Solid Geometry (CSG) and Boundary representation (B-rep) [65, 100], the PDE method allows both of the shape and solid properties to be attached to an object, which can meet certain functional criteria. Here is the formulation of the second-order elliptic PDE to model solids:

$$(a^2\frac{\partial^2}{\partial u^2}+b^2\frac{\partial^2}{\partial v^2}+c^2\frac{\partial^2}{\partial w^2})\mathbf{X}(u,v,w)=\mathbf{0} \tag{2.7}$$

where u, v, and w are parametric coordinates in 3D space.

To solve this PDE, six boundary surfaces are required for a unique solution. The boundary surfaces are of following forms:

$$\begin{aligned}
&\mathbf{X}(0,v,w)=\mathbf{U}_0(v,w),\mathbf{X}(1,v,w)=\mathbf{U}_1(v,w),\\
&\mathbf{X}(u,0,w)=\mathbf{V}_0(u,w),\mathbf{X}(u,1,w)=\mathbf{V}_1(u,w),\\
&\mathbf{X}(u,v,0)=\mathbf{W}_0(u,v),\mathbf{X}(u,v,1)=\mathbf{W}_1(u,v).
\end{aligned} \tag{2.8}$$

We further incorporated the solid PDE formulation into the PDE modeling system to offer PDE-based free-form deformation and direct manipulation of solid objects [45]. We [48] also integrated the PDE solid with implicit properties for general free-form PDE modeling applications.

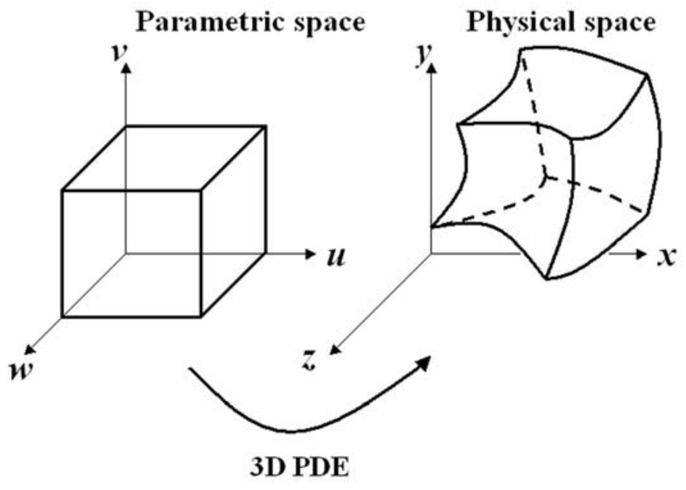


Figure 2.4: Construct 3D solids from parametric space using PDE method.

2.2.3 Generating Fair Meshes

Variational constrained optimization problems have been widely used in surface modeling [151, 157], physics-based modeling of deformable surfaces [27, 140], and subdivision models [77, 80, 152]. In variational subdivision models, with choosing appropriate functionals to be minimized iteratively during the refinement of a subdivision surface, the limit surface of high smoothness can be constructed efficiently. Because the common used minimization functionals in variational design (*e.g.*, curvature minimization functional, energy minimization functional) can be characterized in the form of PDEs, the variational modeling is closely related to PDE techniques, and sometimes variational optimization problems can be transformed to certain PDE problems.

Schneider and Kobbelt [123] presented an algorithm to create fair discrete surfaces satisfying prescribed G^1 boundary constraints by discretizing a PDE based on pure geometric intrinsics. The algorithm can construct surfaces of high aesthetic quality that have no local mean curvature extrema in the interior.

A common method for surface fairness is to minimize fairness functionals based on geometric invariants, such as curvature minimization functionals. Because the computation time is enormous for such method, people usually give up the parametric independence and approximate the geometric invariants with higher order derivatives, which can result in the construction of a solution by solving a linear system. A representant of this category is the thin plate energy:

$$\int\int f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 dxdy$$

which can be used to create surfaces satisfying C^1 boundary conditions.

Instead of minimizing a functional, another approach first applies variational calculus to transform the minimization problem into the problem of solving a differential equation with constraints. Even without using variational calculus, the PDE approach can solve the fairing problem itself, which is useful for fairing based on geometric invariants. The PDE used to solve the fairing problem is

$$\Delta_B H = 0 \tag{2.9}$$

where Δ_B is the Laplace-Beltrami operator, which extends the planar Laplacian to a smooth surface.

This algorithm can be used to solve the N-sided-hole problem and other blending problem in many fields of CAGD. Fig. 2.5 shows an example of this method.

The construction of fair surfaces from irregularly triangulated data which removes the rough features can also make use of variational methods. Desbrun *et al.*[38] developed an algorithm to remove the undesirable noise and uneven edges from imperfectly-measured data from the real world while retaining desirable geometric features. They used an implicit integration of the diffusion equation as well as the curvature flow for the smoothing of meshes.

Variational approach can find applications in shape transformation and interpolation. Turk and O'Brien [146] presented a shape transformation method

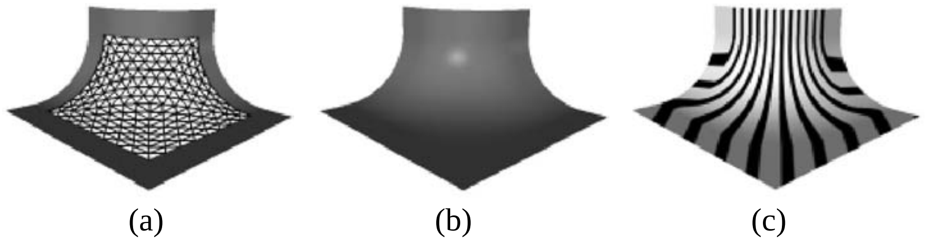


Figure 2.5: Example of variational surface fairing: the classical house corner problem. Image courtesy of Schneider and Kobbelt [123].

using variational implicit functions for N-dimensional objects. It unified the implicit function creation and interpolation into one single step and provided smooth and natural shape transformations even between objects with different topologies. Later on, they proposed a variational implicit method for shape interpolation[147].

2.3 Level Set Method

2.3.1 Level Set Formulation

The level set method was introduced by Osher and Sethian [108] to track moving interfaces in a wide variety of problems. The original idea behind the level set method is as follows: Given an interface Γ in n-dimensional space R^n, bounding a (perhaps multiply connected) open region Ω, the goal is to analyze and compute its subsequent motion under a velocity field $\mathbf{v}$. This velocity can depend on position, time, the geometry of the interface (*e.g.,* its normal or its mean curvature) and the external physics. The level set method relies on the relation between propagating interfaces and propagating shocks. The equation for a front that propagates with curvature dependent speed is linked to a viscous hyperbolic conservation law for the propagating gradients of the front. The idea is to define a smooth function $\phi(x,t)$, whose zero-level set $\phi(x,t) = 0$ represents the propagating interface.

Here $x = (x_1, \cdots, x_n) \in R^n$. The level set function ϕ has the following properties:

$$\begin{aligned} \phi(x,t) &> 0 \quad for \quad x \in \Omega, \\ \phi(x,t) &< 0 \quad for \quad x \notin \Omega, \\ \phi(x,t) &= 0 \quad for \quad x \in \partial\Omega = \Gamma(t). \end{aligned}$$

The motion for this evolving function ϕ is determined from a PDE in one higher dimension which permits cusps, sharp corners, and changes in topology in the zero-level set describing the interface.

The level set approach works as follows: suppose one wishes to follow the evolution of a curve Γ_0 as it propagates in a direction normal to itself with speed F. Then the family of moving curves Γ_t can be matched with a family of moving surfaces in such a way that the zero-level sets always yield the moving front. All that remains is to find an equation of motion for the evolving surface.

Let Γ_0 be a closed, non-intersecting curve. Assume $\phi(x,t), x \in R^n$, is a scalar function such that at time t the zero-level set of $\phi(x,t)$ is the curve Γ_t, and further assume $\phi(x,0) = \pm d(x)$, where $d(x)$ is the distance from x to the curve Γ_0. The plus sign is used if x is inside Γ_0 and the minus sign for x outside. Let each level set of ϕ flow along its gradient field $\nabla\phi$ with speed F. Then ϕ can be computed by solving the differential equation

$$\begin{aligned} \phi_t - F|\nabla\phi| &= 0 \\ \phi(x, t=0) &= \pm d(x). \end{aligned} \tag{2.10}$$

At any time, the moving front Γ_t is the zero-level set of ϕ.

If F depends on the curvature, the curvature may be expressed in terms of ϕ by

$$F = \frac{\phi_{yy}\phi_x^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{xx}\phi_y^2}{(\phi_x^2 + \phi_y^2)^{3/2}}.$$

This is called an Eulerian formulation for front propagation, because it is written in terms of a fixed coordinate system in the physical domain. There are three advantages to such an approach. First, since the underlying coordinate system is fixed, discrete mesh points do not move and the stability problems that plague the Lagrangian approximations may be avoided. Second, topological changes are handled naturally, since the zero-level set of ϕ needs not be simply connected. Third, the above formulation can be easily extended to moving surfaces in 3D with appropriate expressions for the curvature (such as mean or Gaussian curvature). The above initial value PDE may be approximated using spatial and temporal derivatives on a fixed grid.

Since its introduction, the level set approach has been used to compute and analyze a broad array of physical and mathematical phenomena, including problems in compressible/incompressible flow, flow having singular vorticity, Stefan problems, kinetic crystal growth, combustion, shape recognition, minimal surface generation, etc. In recent years, PDEs and level set motion are also explored in image analysis and computer vision. One basic idea is to view an image as $u_0(x, y)$, a function defined on a square, and obtain a (usually second-order) flow equation of the form

$$\begin{aligned} u_t &= F(u, Du, D^2u, x, t) \\ u(x, y, 0) &= u_0(x, y), \end{aligned} \tag{2.11}$$

which, for positive t, processes the image. For example, if one solves the heat equation with

$$F(u, Du, D^2u, x, t) = \nabla^2 u,$$

then $u(x, y, t)$ is the same as convolution of u_0 with a Gaussian of variance t.

2.3.2 Shape Reconstruction Using Level Set Method

Using level set method, the problem of reconstructing shapes from scattered datasets can be easily solved. Zhao *et al.*[162] used the level set method as a

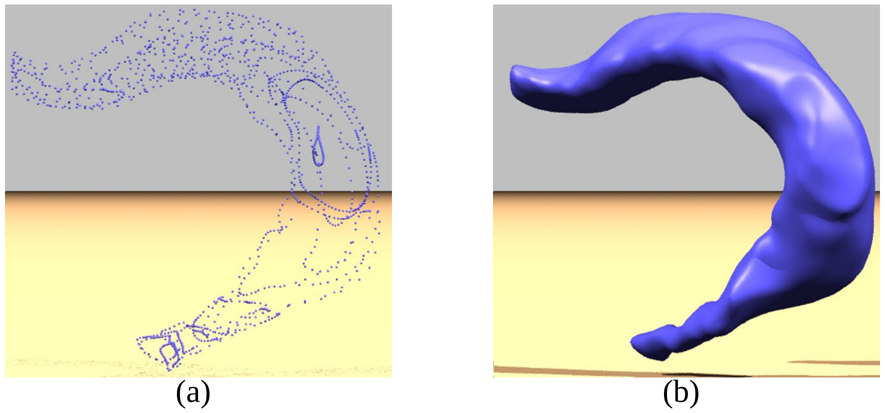


Figure 2.6: Examples of shape reconstruction using level set method. Image courtesy of Zhao *et al.*[162].

numerical technique to evolve the implicit surface continuously following the gradient descent of the energy functional for the shape reconstruction from scattered points. Fig. 2.6 shows an example.

2.3.3 Level Set Method for Shape Morphing

Level set method can also be used for shape transformation (morphing) [23] which uses a volume-based technique formulating the blending state of the morphing process as the optimization, via a hill-climbing strategy, of a similarity measure between the deforming surface and the target, utilizing level set models for the incremental shape changes. Refer to Fig. 2.7 for an example.

Comprehensive reviews of the level set approach may be found in [1, 22, 127]. The generality of this approach makes it very attractive, especially for problems in three dimensions, problems with sensitive dependence on curvature (such as surface tension problems), and problems with complex changes of topology.

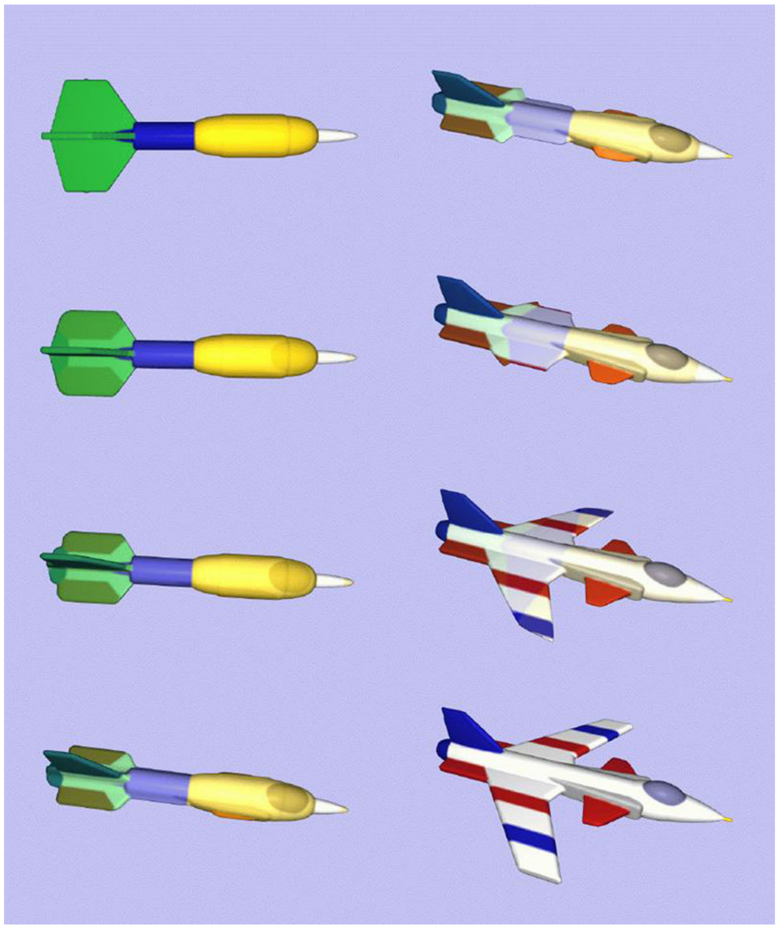


Figure 2.7: Using level set method for shape morphing. Image courtesy of Breen and Whitaker [23].

2.4 Diffusion Equations

Diffusion equations are another popular type of PDEs which have applications in texture synthesis, image processing, surface fairing, and visualization.

A Diffusion equation is defined as a PDE describing the variation in space and time of a physical quantity which is governed by diffusion. It provides a good mathematical model for the variation of temperature through heat conduction and electromagnetic wave propagation in a highly conducting medium. The diffusion equation is a parabolic PDE whose characteristic form relates the first partial derivative of a field u with respect to time t to its second partial derivatives with respect to spatial coordinates $\mathbf{x}$:

$$\frac{\partial u}{\partial t} = \kappa \nabla^2 u, \tag{2.12}$$

where $u = u(\mathbf{x}, t), \mathbf{x} = (x_1, x_2, \cdots, x_n) \in \Omega \subset \mathbf{R}^n, t \geq 0$, and κ is called the diffusion coefficient. The operator $\nabla^2 = \sum_i \frac{\partial^2}{\partial x_i^2}$ is called the *Laplacian*. When κ is not constant, but depends on spatial coordinates: $\kappa = \kappa(\mathbf{x})$, this spatial variation leads to *anisotropic diffusion equation*:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\kappa \nabla u). \tag{2.13}$$

The solutions of diffusion equations are subject to both initial and boundary conditions. Numerical solutions of diffusion equations usually refer to the finite-difference method, which uses Forward Time Centered Space (FTCS) finite-difference approximation for diffusion equations. Using such equations, people can develop visually convincing models of fire, smoke, and other gaseous phenomena. Diffusion equations can also be used in scientific visualization of medical images.

2.4.1 Depicting Gaseous Phenomena Using Diffusion Processes

In 1995, Stam and Fiume [135] discussed using diffusion processes to develop models of fire, smoke, and other gaseous phenomena, especially models of ”wispy” smoke and steam. They created new methods of animating a wide range of gaseous phenomena using far fewer primitives than before. They gave the reformulation and solution of the advection-diffusion equation for densities composed of ”warped blobs”. These blobs model the distortion that gases undergo when advected by wind fields more accurately. They also introduced a simple model for the flame of fire and its spread.

2.4.2 Reaction-Diffusion Systems for Texture Synthesis

The PDE techniques can be used to synthesizing nature textures with the consideration of *reaction-diffusion* (RD) systems, which give rises to nonlinear PDEs. The RD mechanism was first proposed by Turing [143] to account for the pattern formation in biological morphogenesis. Starting with Turing’s work, in 1991, Witkin and Kass [155] presented a method to generate textures based on the simulation of RD system. They generalized the basic RD model by introducing anisotropic and space-varying diffusion. The addition of anisotropy allows the creation of zebra stripes, and sand ripples. With allowing diffusion rates and directions to vary over space, they created more complex patterns, including the swirling patterns typical of fingerprints. They also considered the functions, which allow multiple competing orientations at each point, to create patterns of a striking woven or lattice-like appearance. The use of non-standard initial conditions or reaction-diffusion parameters can model giraffe markings. Fig. 2.8 shows several texture patterns generated by Witkin and Kass [155].

2.4.3 Visualizing Vector Field

Vector field visualization is an important topic in scientific visualization. It aims to graphically represent field data on two and three-dimensional domains and on surfaces in an intuitively understandable way. The anisotropic nonlinear diffusion can be used in flow visualization [114]. Diewald *et al.*[41] proposed a new approach based on anisotropic nonlinear diffusion for vector field visualization. It

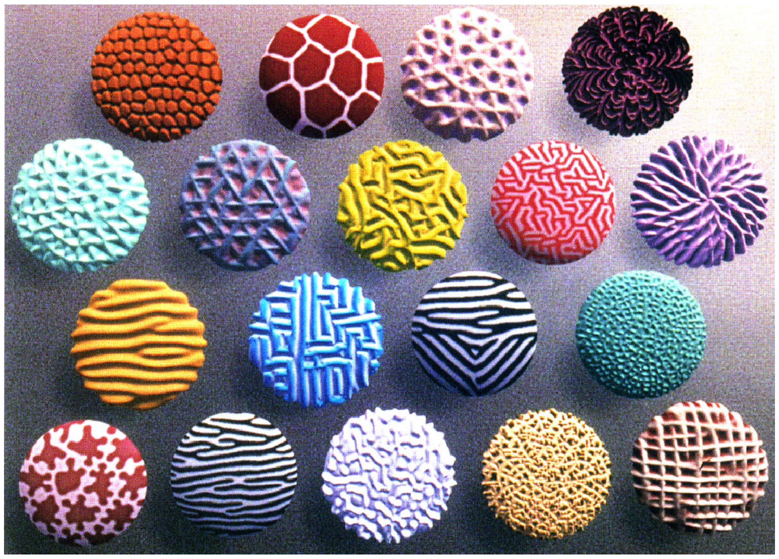


Figure 2.8: Texture buttons generated by reaction diffusion system. Image courtesy of Witkin and Kass [155].

enables an easy perception of vector field data and serves as an appropriate scale space method for the visualization of complicated flow patterns. The approach is closely related to nonlinear diffusion methods in image analysis where images are smoothed while still retaining and enhancing edges. Here, an initial noisy image intensity is smoothed along integral lines, whereas the image is sharpened in the orthogonal direction. The method is based on a continuous model and requires the solution of a parabolic PDE problem. Applications are shown for flow fields in 2D and 3D, as well as for principal directions of curvature on general triangulated surfaces. Furthermore, the provisions for flow segmentation are outlined. An example for visualizing the principle curvature directions is shown in Fig. 2.9.

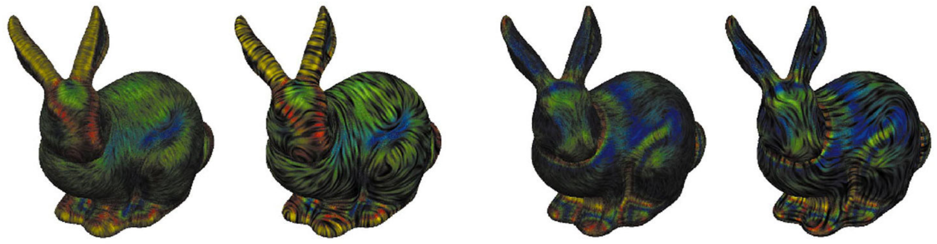


Figure 2.9: Visualizing principal curvature directions. Different time steps of the anisotropic diffusion are displayed on the surface of a pre-smoothed Stanford bunny. In addition, the corresponding principle curvature values are color coded. Image courtesy of Diewald *et al.*[41].

2.4.4 Tensor Field Visualization for MRI Data

Kindlmann *et al.*[76] also employed diffusion textures to help visualizing tensor field of the magnetic resonance imaging (MRI) brain together with other visualization tools. Diffusion-weighted MRI is capable of elucidating the fibrous structure of certain types of tissue, such as the white matter within the brain. One tool for interpreting this data is volume rendering because it permits the visualization of three dimensional structure without a prior segmentation process. In order to use volume rendering, they developed three methods for assigning opacity and color to the data, and create a method to shade the data to improve the legibility of the rendering including barycentric opacity maps, hue-balls (for color), and lit-tensors (for shading). They also proposed anisotropic reaction-diffusion volume textures as an additional tool for visualizing the structure of diffusion data. The patterns generated by this process can be visualized on their own or they can be used to supplement the volume rendering strategies. Fig. 2.10 shows an example of mapping the diffusion texture on the brain data.

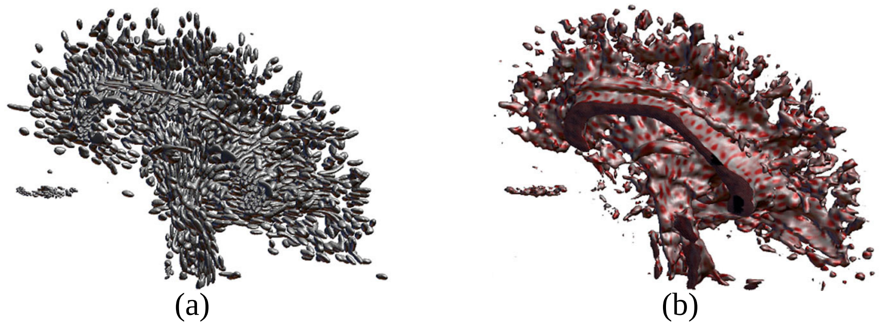


Figure 2.10: Texture-mapping with reaction-diffusion texture.(a) Segmented texture; (b) texture mapped onto surface rendering. Image courtesy of Kindlmann *et al.*[76].

2.5 Other Applications of PDE Techniques

PDE techniques also have various applications in other computer graphics related areas such as animation, simulation, image processing, etc.

2.5.1 Animation and Simulation

Modeling Fracture

PDEs can be used along with physics-based techniques to graphically model and animate the realistic behavior of materials that can undergo fracture due to deformation-induced stress [105, 104, 106]. O'Brien and Hodgins proposed an approach based on linear elastic fracture mechanics and non-linear finite-element analysis to model three-dimensional volumes using a mesh of tetrahedral elements and simulate the fracture of the volumes under stresses. With this approach, it's possible to automatically generate realistic synthetic motion for three-dimensional solid objects that can break, crack, or tear. They employed a fast, tetrahedral finite-element method that uses linear shape functions within the elements. They accommodated arbitrary propagation directions by dynamically reconstruct the mesh. Because cracks are not limited to the original element boundaries, objects can form irregularly shaped shards and edges as they shatter. To model the deformation of the material which will cause fractures, a set of differential equations that describe the aggregate behavior of the material were defined in a continuous form based on continuum mechanics, and then these equations were discretized using finite-element method for computer simulation. Fig. 2.11 is one example of shattered Stanford Bunny model.

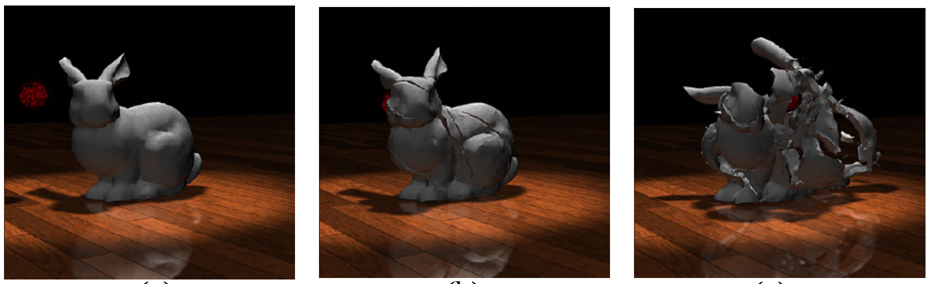


Figure 2.11: Example of modeling fracture. Image courtesy of O'Brien [104].

Fluid Dynamics

Fluid dynamics can be viewed as another type of PDE applications, because the equations governing the behavior of fluid are also a type of PDEs. Using the fluid dynamics mechanics, with different physical material properties as initial/boundary conditions, people can simulate water, gas, smoke, explosion, and so on. There are many contributions in simulating fluid materials using such techniques.

The most popular PDEs to model the fluid flow are *Navier-Stokes* Equations. A fluid whose density and temperature are nearly constant can be described by a velocity field $\mathbf{u}$ and a pressure field p. These quantities generally vary both in space and in time and depend on the boundaries surrounding the fluid. Given that velocity and pressure are known for some initial time $t = 0$, then the evolution of these quantities over time is given by Navier-Stokes equations [134]:

$$\begin{array}{lcll} \nabla \cdot \mathbf{u} & = & 0 & (a) \\ \frac{\partial \mathbf{u}}{\partial t} & = & -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u} + \mathbf{f}, & (b) \end{array} \tag{2.14}$$

where v is the kinematic viscosity of the fluid, ρ is its density and $\mathbf{f}$ is an external force. The symbol ∇ is the vector of spatial partial derivatives. More precisely, $\nabla = (\partial/\partial x, \partial/\partial y)$ in two-dimensions and $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$ in three-dimensions. Navier-Stokes equations are obtained by imposing that fluid conserves both mass (part (a)) and momentum (part (b)). These equations also have to be supplemented with boundary conditions. There are many derivations of Navier-Stokes equations, which lead to simulations of different dynamic fluids, such as hot turbulent gas by Foster and Metaxas [60], the stable fluid by Stam [134], animated explosions by Yngve *et al.*[158], etc.

Gas Simulation In 1997, Foster and Metaxas [60] proposed an animation technique to model the turbulent rotational motion that occurs when a hot gas interacts with solid objects and the surrounding medium. The method is especially useful for scenes involving swirling steam, rolling or billowing smoke, and gusting wind. It can also model gas motion due to fans and heat convection. The method combines specialized forms of equations of motion of a hot gas with an efficient method for solving volumetric differential equations at low resolutions. It's a physics-based model specifically designed to realistically animate the complex rotational component to gaseous motion, effects due to regions of different temperature within a gas, and the interaction between gas and other objects. The model accounts for convection, turbulence, vorticity and thermal buoyancy, and can also accurately model gas flowing around complex objects such as hot steam being vented into a boiler room or the rolling smoke cloud from an explosion. Fig. 2.12 is an example of this model.

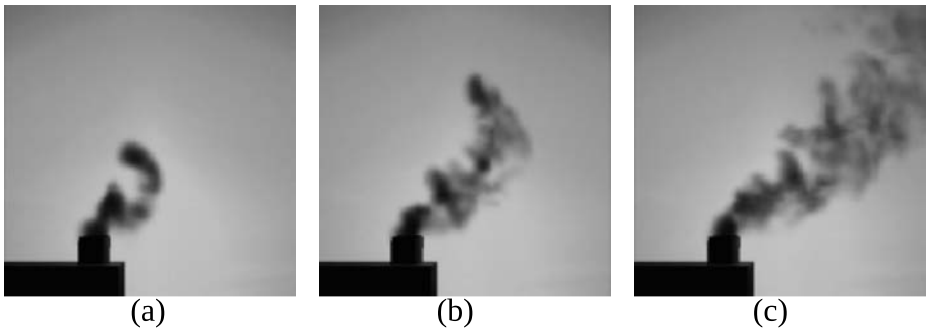


Figure 2.12: An example of turbulent smoke rolls out of a chimney into a light, gusting wind. Image courtesy of Foster and Metaxas [60].

Stam [134] improved the work of Foster and Metaxas by a simple and stable algorithm to solve the full Navier-Stokes equations with larger time steps. The improved model allows users to interact in real-time with three dimensional fluids on a graphics workstation. Both Lagrangian and implicit methods are used instead of the explicit Eulerian schemes to solve Navier-Stokes equations. It can be applied to simulate gaseous-like phenomena.

Water Simulation Kass and Miller [73] proposed a CFD simulation technique based on an approximation of the shallow water equations. These equations simplify Navier-Stokes equations by making certain assumptions and approximations according to the property of shallow water. They used a height field to represent the water's surface. The equation is transformed to a tridiagonal linear system on a uniform finite-difference grid, which can be solved by iterative methods. This model can handle wave reflections, net transport of water and boundary conditions with changing topology. It's suitable to animate flowing rivers, raindrops hitting surfaces and waves in a fish tank as well as waves lapping on a beach.

Foster and Metaxas [58] presented a realistic and stable simulation method to animate the liquid motion using finite-difference approximation to the incompressible Navier-Stokes equations. The model incorporates Lagrange equations of motion coupled by the pressure field to simulate the dynamic behavior of buoyant rigid objects. It can be used to simulate the wave effects of refraction, reflection and diffraction, as well as rotational motion (*e.g.*, vorticity). The fluid surface is represented as either a chain of massless marker particles or a height field. The liquid sources or sinks and time dependent pressure field (such as strong wind) can be included in the simulated environment. Later on, the authors [59] provided an embedded controller as an interface between the animator and a general tool for calculating three dimensional fluid flow for controlling fluid animations. It allows computer graphics animators to specify and control a three dimensional fluid animation without knowing the underlying equations.

Explosion Animation Explosions are one sort of the most dramatic phenomena in nature. According to [158], an explosion is born when a sudden burst of energy from a mechanical, chemical, or nuclear source causes a pressure wave to propagate outward throughout the air. An explosion can cause a variety of visual effects in addition to the light refraction by the blast wave, such as blinding flash of light, dust clouds, hot gases and smokes, and so on. Yngve *et al.*[158] presented a physics-based model of explosions and simulated many of the above mentioned effects. They modeled the explosion post-denotation as compressible, viscous flow, and solve the flow equations with an integrated method that handles the extreme shocks and supersonic velocities inherent in explosions. They used a fluid dynamics model of the air to capture many of the visual effects. In their model, in addition to Navier-Stokes equations for conservation of momentum, they also used governing equations for the conservation of mass and energy and for the fluid's thermodynamic state. Fig. 2.13 are some examples of the simulated explosions by [158].

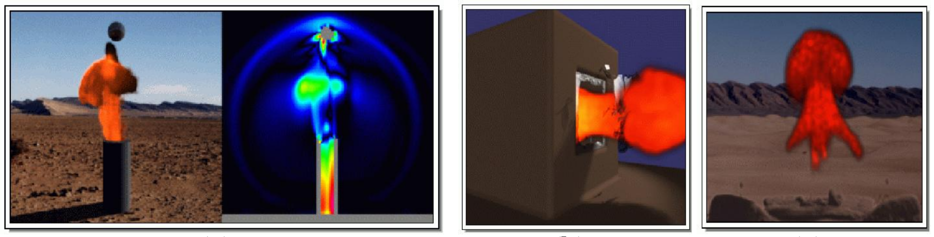


Figure 2.13: Examples of Explosion Simulation. (a) An image of a projectile propelled from a chamber by an explosion: on the right is a cross-section of three-dimensional fluid volume using a color map where hotter colors indicate higher densities; (b) a glass window is shattered by a blast wave; (c) a simulated fireball. Image courtesy of Yngve *et al.*[158].

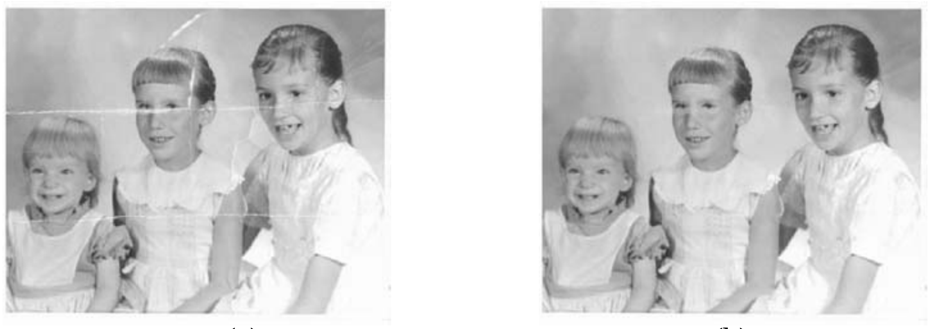


Figure 2.14: Examples of Image Inpainting. (a) Original image; (b) result image of (a). Image courtesy of Bertalmio *et al.*[7].

2.5.2 Image Processing

Image inpainting represents the technique to modify images in an undetectable way. The applications of this technique include the recovery of damaged pictures, removing of selected area in the images, and so on. Because gradient information of images, especially the information around the boundary of the image part of interests, which can be modeled by certain PDEs, is extremely helpful in the ”inpainting” process, the PDE model may also be employed for image inpainting. Bertalmio *et al.*[7] presented a method to fulfill the inpainting job using information of the surrounding area of the selected region. In particular, they considered the boundary $\partial\Omega$ of the selected area Ω to be inpainted, then smoothly propagated information from the surrounding areas in the isophotes direction using certain partial differential equations of gradient vectors. Fig. 2.14 gives an example of using such method to restore pictures.

Recently, Pérez *et al.*[109] introduced a method using Poisson equations to provide a variety of novel tools for seamless editing of image regions. The first set of tools permits the seamless importation of both opaque and transparent source image regions into a destination region. The second set is based on similar mathematical ideas and allows the user to modify the image appearance seamlessly, within a selected region. These changes can be arranged to affect the texture, the illumination, and the color of objects lying in the region.

Chapter 3

Related Work of Other Modeling Techniques

The PDE-based modeling framework presented in this dissertation is not only developed based on previous work of PDE modeling techniques, but also related to several other popular geometric modeling techniques, such as spline-based shape representations, implicit functions, physics-based interactive models, and medial axis extraction techniques.

3.1 Spline-based Shape Representations

Free-form splines can be viewed as one of the most popular shape modeling techniques which use piecewise polynomials with certain constraints of differentiability to represent geometric objects such as curves, surfaces, and solids. They can describe not only standard analytic shapes (lines, conics, circles, planes, etc.), but also free-form objects, and they also provide extra degrees of freedom for the shape manipulation. Some of the spline-based techniques can limit the deformation within specified local regions without affecting the global shape when manipulating the geometric objects. This feature of local control is often desirable for shape design.

Among the wide variety of spline families, non-uniform rational B-splines (NURBS) have gained the most popularity and become an industrial standard for geometric design. This section will introduce the NURBS formulation and applications as an example for the spline-based techniques. There are two major ingredients of NURBS, rational and B-splines.

A B-spline curve is a piecewise polynomial curve [20]:

$$\mathbf{s}(u) = \sum_i \mathbf{P}_i N_i^n(u). \tag{3.1}$$

$\mathbf{P}_i$'s are control points and $N_i^n(u)$'s are B-spline basis functions which are piecewise polynomials of degree n recursively defined over the knots sequence $\mathbf{U} = u_0, u_1, \ldots$, where $u_0 \leq u_1 \leq \ldots$:

$$N_i^r(u) = \begin{cases} 1, \text{ if } r = 0 \text{ and } u \in [u_i, u_{i+1}); \\ 0, \text{ if } r = 0 \text{ and } u \notin [u_i, u_{i+1}); \\ (u - u_i)\frac{N_i^{r-1}(u)}{u_{i+r} - u_i} + (u_{i+r+1} - u)\frac{N_{i+1}^{r-1}(u)}{u_{r+i+1} - u_{i+1}}, r > 0 \end{cases} \tag{3.2}$$

The basis functions have several important properties, such as they are non-negative, the sum of all the basis functions of same order is 1, and they provide local support, etc.

The product of B-spline basis functions $N_i^n(u)$'s and $M_j^m(v)$'s of given knots u_i, v_j provides the representation of a B-spline surface:

$$\mathbf{X}(u, v) = \sum_i \sum_j \mathbf{P}_{i,j} M_j^m(v) N_i^n(u), \tag{3.3}$$

where $\mathbf{P}_{i,j}$ are control vertices forming a control mesh.

A NURBS curve is a vector-valued piecewise rational polynomial function of the form [111]:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i N_i^p(u)}{\sum_{i=0}^{n} w_i N_i^p(u)}, \tag{3.4}$$

where w_i are weights and $\mathbf{P}_i$'s are control points. $N_i^p(u)$'s are B-spline basis functions of degree p over the non-uniform knots vector $\mathbf{U} = u_0, u_1, ..., u_m$. The relationship of the degree, number of knots, and number of control points are defined by the formula $m = n + p + 1$. For non-uniform and non-periodic B-splines, the knot vector takes the form:

$$\mathbf{U} = \alpha, ..., \alpha, u_{p+1}, ..., u_{m-p-1}, \beta, ..., \beta$$

where the end knots α and β are repeated with multiplicity $p+1$. In most practical applications $\alpha = 0$ and $\beta = 1$. With the above defined knots vector, the NURBS curve of(3.4) interpolates the endpoints and is tangential at the endpoints to the first and last legs of the control polygon.

A NURBS surface is the rational generalization of the tensor-product non-rational B-spline surface defined as follows:

$$\mathbf{S}(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} \mathbf{P}_{i,j} N_i^p(u) N_j^q(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} N_i^p(u) N_j^q(v)} \tag{3.5}$$

where $w_{i,j}$'s are the weights, $\mathbf{P}_{i,j}$'s form a control net, and $N_i^p(u)$'s and $N_j^q(v)$'s are B-spline basis functions in the u and v directions, respectively, defined over the knot vectors

$$\mathbf{U} = 0, ..., 0, u_{p+1}, ..., u_{r-p-1}, 1, ..., 1,$$

$$\mathbf{V} = 0, ..., 0, v_{q+1}, ..., v_{s-q-1}, 1, ..., 1,$$

where the end knots are repeated with multiplicities $p + 1$ and $q + 1$, respectively, and $r = n + p + 1$ and $s = m + q + 1$.

The NURBS are popular and widely accepted in the CAD/CAM and graphics community because of various properties [111]. NURBS offer a common mathematical form for representing and designing both standard analytic shapes and free-form curves and surfaces. By manipulating control points and weights, NURBS provide the flexibility to design a large variety of shapes. NURBS have clear geometric interpretations, making them particularly useful for designers, who have a very good knowledge of geometry, especially descriptive geometry. They have a set of powerful geometric toolkits (*e.g.*, knot insertion, refinement, or removal, degree elevation, splitting, etc.) to design, analyze, process, and interrogate objects. NURBS are invariant under scaling, rotation, translation and shear as well as parallel and perspective projection. They are genuine generalizations of non-rational B-spline forms as well as rational and non-rational Bezier curves and surfaces.

However, at the same time, NURBS have several drawbacks [111]. They require extra storage to define traditional curves and surfaces. Improper application of weights can result in a very bad parameterization, which may destroy subsequent surface constructions. Some interrogation techniques work better with traditional forms than with NURBS, *e.g.*, surface/surface intersection, where it is particularly difficult to handle the *just touch* or *overlap* cases. Fundamental algorithms, such as inverse point mapping, are subject to numerical instability. Furthermore, spline-based techniques are less natural and non-intuitive, primarily because free-form splines are oftentimes associated with tedious and indirect shape manipulation through time-consuming operations on a large number of control vertices, non-unity weights, and/or non-uniform knots. Users need strong mathematical sophistication to deal with such techniques. Usually spline-based techniques are restrained to model regular shapes. It is difficult to extend their geometric coverage to shapes of arbitrary topology without resorting to various non-intuitive geometric constraints.

3.2 Implicit Models

Without specification of precise locations of geometric entities as explicit models, implicit functions offer a different type of shape representations by using certain scalar field functions to define geometric objects as level-sets of specific intensity values. A general form of implicit function to represent a surface has the following form:

$$f(x, y, z) = c, \tag{3.6}$$

where $f(x, y, z)$ is a scalar function over the physical domain of x, y, z, and c is a constant scalar value. Similarly, an implicit solid can be represented by

$$\{(x, y, z)|f(x, y, z) \leq c\}, \tag{3.7}$$

where $f(x, y, z) = c$ defines the boundary surface of the solid.

The implicit representations offer designing, modeling, and interacting with 3D geometric entities in the x, y, z domain directly. In the past several years, implicit functions have been widely developed as a powerful design and manipulation tool for graphical models. They offer a totally different yet convenient and natural design and modeling paradigm (compared with parametric representations) in visual computing fields such as graphics, animation, and geometric design. This is because of their unique properties such as arbitrary topology, collision detection, free of parametric correspondence, etc. In general, every rational parametric entity can be modeled by certain implicit functions, which implies that the set of implicit objects is larger than that of rational parametric shapes.

3.2.1 Particle-based Implicit Surface Sculpting

In 1994, Witkin and Heckbert [154] introduced an approach using particles to sample and control implicit surfaces. They used a simple constraint that locked a collection of particles onto an implicit surface as *control points* for the surface. The constraint was used to make the surface follow the particles, or to make the particles follow the surface. They implemented control points for direct manipulation by specifying particle motions, then solving for surface motion that maintains the constraint. They specified and solved for velocities rather than positions, and the behavior of the system was governed by differential equations that integrate these velocities over time. For sampling and rendering, they created floater particles that roamed freely over the surface. Local repulsion was used to make floaters spread evenly across the surface. By varying the radius of repulsion adaptively, and fissioning or killing particles based on the local density, good sampling distributions can be obtained very rapidly, and maintained in rapid and extreme deformations and changes in surface topology.

3.2.2 Trivariate B-splines for Implicit Models

Raviv and Elber [118] presented an interactive sculpting technique using the zero level-set of scalar trivariate B-spline functions to represent 3D objects. Users can indirectly sculpt an object by modifying relevant scalar control coefficients of the underlying B-spline functions. Like all tensor product B-spline functions, the trivariate functions have a control volume that consists of scalar coefficients, $P_{i,j,k} \in R$. The trivariate functions are of the form:

$$f(u,v,w) = \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} P_{i,j,k} B_i(u) B_j(v) B_k(w), \tag{3.8}$$

where $B_i(u)$, $B_j(v)$, $B_k(w)$ are the uniform B-spline basis functions, $P_{i,j,k}$'s are scalar coefficients in a volumetric mesh of size $l \times m \times n$, and $f(u, v, w)$ is a scalar function. Sculpting is conducted by modifying values of scalar coefficients of the implicit trivariate functions. Arbitrarily accurate constant iso-surfaces are approximated by adaptive sampling of the exact free-form trivariate function, at some prescribed resolution, and by computing a polygonal approximation model of the sculpted shape via the Marching Cube algorithm [90].

Later on, Hua and Qin [69, 70] developed interactive solid sculpting toolkits with haptics on implicit B-spline solids defined through the use of B-spline control coefficients over the intensity field.

3.2.3 Level Set Method for Implicit Functions

Zhao *et al.* [162] proposed a *weighted* minimal surface model based on variational formulations and PDE techniques to construct a surface from scattered data. They used the level set method as a numerical technique to evolve the implicit surface continuously following the gradient descent of the energy functional for the final reconstruction. Their level set model is governed by a time evolution PDE with velocity at the level sets given by the motion law of the original surface. The level set method is based on a continuous formulation using PDEs and deforms an implicit surface according to various laws of motion depending on geometry, external forces, or certain energy minimization. It can easily handle topological changes and reduce noises in the dataset. Their level set method mainly focuses on implicit objects reconstructed from scattered datasets. Problems for interpolating curve sketches, especially open curve sketches haven't been addressed. Cutler *et al.*[33] presented a procedural framework for specifying layered solid models and applying a series of simulation operations as sculpting tools described by a script language to the models which can be tedious for complex models. Bærentzen and Christensen [5] developed an interactive volume sculpting method using the level set method which offers smoothing/un-smoothing, adding/removing blob, and dilating/eroding tools for volumetric implicit models. Museth *et al.*[102] proposed level-set-based editors using CSG operations, blending, embossing, and smoothing for implicit surfaces. However, these tools are associated with the specification of speed functions for the evolving level set, which are non-intuitive for common users.

3.2.4 Variational Implicit Functions

Implicit functions can also be used for shape reconstruction and 3D morphing process. Turk and O'Brien [146] made uses of variational implicit functions to achieve shape morphing and surface reconstruction. They employed the Radial Basis Function (RBF) method to construct an implicit function that interpolates the given dataset and minimizes the thin-plate energy. Morse *et al.*[99] proposed using compactly supported RBFs to interpolate implicit surfaces from scattered surface data. Turk and O'Brien [147] provided a variational interpolation approach for interactive implicit surface sculpting via particles, but each operation requires reformatting and recalculation of the entire system, which is difficult to model large datasets. Fig. 3.1 shows an reconstruction example.

Despite the modeling advantages of implicit functions, systematic modeling toolkits for direct manipulation of implicit surfaces and solids are still under explored.

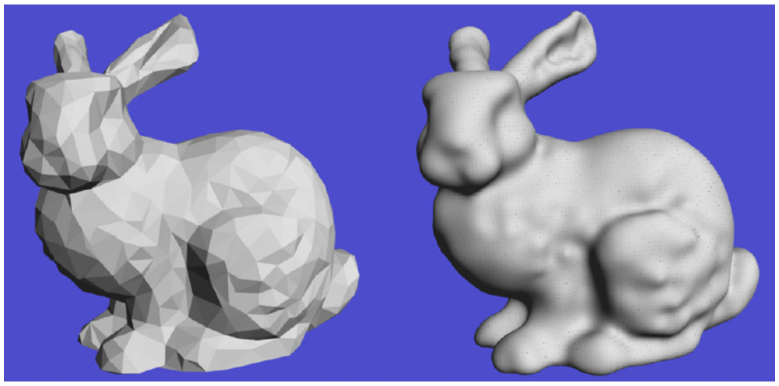


Figure 3.1: A polygonal surface (left) and the interpolating implicit surface defined by the vertices and their normals (right). Image courtesy of Turk and O'Brien [147].

3.3 Physics-based Modeling

The pure geometric methods such as spline-based techniques only consider geometric attributes of objects, which means they require indirect and non-intuitive manipulations to obtain desired shapes. In general, such modeling process is time-consuming and not interactive.

Physics-based modeling, which takes the physical attributes and material properties of objects into account and makes use of physical principles during the modeling process, offers users a way to overcome the drawback of indirect pure geometric design mechanism. It can incorporate the external forces, mass, damping, time, and constraints into a dynamic framework to produce smooth, natural, intuitive motions of objects. The physics-based modeling also provides direct and interactive manipulation tools for both professional experts and naive users. It augments (instead of replaces) the well-established geometric modeling techniques with aforementioned features, which makes the physics-based modeling more powerful for modeling and design processes.

A well-known physics-based modeling technique is the free-form deformable model introduced by Terzopoulos *et al.*[140]. The physics-based model is governed by the mechanical laws of continuous bodies which can be expressed in the form of dynamic differential equations. The dynamic and realistic behavior can be obtained by solving an associated motion equation numerically. The physics-based method can be used to interactively model and manipulate various objects dynamically. Terzopoulos *et al.*[138, 139, 140] demonstrated interactive sculpting using viscoelastic and plastic models. Celniker and Gossard [27] developed a prototype system for interactive free-form design based on the finite-element optimization of energy functions proposed by Terzopoulos and Fleischer [138].

3.3.1 Deformable Models

A deformable model is characterized by the position, velocity, and acceleration along with material properties such as mass and damping distributions. The equation governing the dynamic motion of a deformable model can be written in Lagrange's form [140] as follows:

$$\frac{\partial}{\partial t}(\mu\frac{\partial \mathbf{r}}{\partial t}) + \gamma\frac{\partial \mathbf{r}}{\partial t} + \frac{\delta\varepsilon(\mathbf{r})}{\delta \mathbf{r}} = \mathbf{f}(\mathbf{r}, t), \tag{3.9}$$

where $\mathbf{r}(\mathbf{a}, t)$ is the position of the particle $\mathbf{a}$ at time t, $\mu(\mathbf{a})$ is the mass density of the body at $\mathbf{a}$, $\gamma(\mathbf{a})$ is the damping density, and $\mathbf{f}(\mathbf{r}, t)$ represents externally applied forces. $\varepsilon(\mathbf{r})$ is a *functional* which measures the net instantaneous potential energy of the elastic deformation of the body.

The external forces are balanced against the force terms on the left-hand side of (3.9) due to the deformable model. The first term is the internal force due to the model's distributed mass. The second term is the damping force due to dissipation. The third term is the elastic force due to the deformation of the model away from its natural shape. Then the potential energy of deformation for elastic models can be used as the measure of the deformation. With applying external forces to elastic models, users can achieve realistic dynamics, thus simplify the animation of complex objects.

To achieve real-time sculpting and direct manipulation, a continuous dynamic model can be discretized into a collection of mass points connected by a network of springs across the neighbors (and/or along both diagonals). Other springs can be incorporated into the discretized model if certain types of dynamic behavior is more desirable. Hence, a set of second-order differential equations is obtained to govern the physical behavior of the underlying physics-based model:

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}, \tag{3.10}$$

where $\mathbf{p}$ is the position vector of the collection of sample points on the discretized mesh, $\mathbf{M}$ is a mass matrix, $\mathbf{D}$ is a damping matrix, $\mathbf{K}$ is a stiffness matrix, and the force at every mass point in the mesh is the sum of all possible external forces: $\mathbf{f} = \sum \mathbf{f}_{ext}$.

The deformable model then can be solved by numerical approaches such as the finite-difference method and the finite-element method.

3.3.2 Applications of Physics-based Techniques

One intriguing advantage of physics-based models is that they can be integrated with other geometric modeling techniques such as NURBS and subdivision methods for realistic shape manipulations with material properties. Dynamic

NURBS, or D-NURBS [116, 117, 141], are physics-based models that incorporate mass distributions, internal deformation energies, and other physical quantities into the NURBS geometric representation. Using D-NURBS, users can interactively sculpt curves and surfaces and design complex shapes according to required specifications not only in the traditional indirect fashion, by adjusting control points and weights, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. D-NURBS move and deform in a physically intuitive manner in response to users' direct manipulations. Their dynamic behavior results from the numerical integration of a set of nonlinear differential equations that automatically evolve the control points and weights in response to the applied forces and constraints. Lagrangian mechanics are employed to formulate the equations of motion for D-NURBS models.

Physics-based model can also be integrated with subdivision techniques to provide the benefits of both subdivision methods for modeling arbitrary topological objects and those of dynamic models for direct and interactive shape manipulation by applying synthesized forces. Qin *et al.*[115] introduced a dynamic Catmull-Clark subdivision model in 1998. Mandal *et al.*[94] further generalized this model to any subdivision scheme. Such techniques can also find applications in dynamic sculpting, data fitting, and engineering design. Fundamentally, their work is an extension of D-NURBS to surfaces of arbitrary topology. They attached physical parameters and behaviors to Catmull-Clark, Loop, and Butterfly subdivision surfaces by employing different types of finite elements. For each type of subdivision algorithm, they derived different blending functions since each type of model converges to a different type of surface in the limit. The dynamic behavior is governed by the Lagrangian equation of motion and is integrated numerically through an implicit solver. McDonnell and Qin [96, 97] then extended the dynamic subdivision techniques to solid geometry, which makes the dynamic framework of subdivision models even more powerful for shape design and manipulation. They proposed a framework for representing, manipulating, and interacting with 3D virtual solid objects of arbitrary topology. The model is an integration of the solid, subdivision, and physics-based modeling paradigms.

The physics-based techniques can be used to simulate the animation of cloth materials, such as flags, tablecloths, scarves, carpets, clothing, etc. In 1992, Carignan *et al.*[24] developed a method to animate clothes on synthetic actors in motion using physics-based models. They created many individual cloth panels, attached physical properties to the cloth, then either seamed the cloth elements together or attached them to solid objects (*e.g.*, an actor) and animated the cloths according to the motion of the actor in a physical environment. The cloth animation was performed with the internal elastic force and the external forces of gravity, wind, and collision response to make it more realistic. Baraff and Witkin [6] proposed a fast algorithm for cloth simulation with the choice of an implicit integration method. They modeled the cloth as a triangular mesh, with internal cloth forces derived from local stretching forces, compressing forces, as well as damping forces. The implicit method provided the stable simulation with large time steps.

Physics-based models also find applications of facial simulation, because the consideration of physical properties will generate more realistic facial models. Lee *et al.*[85] presented a physics-based approach constructing functional models of the human face based on laser-scanned range and reflectance data which were suitable for animation. They estimated the skull structure and inserted the major muscles for facial expressions into the dynamic facial model. The synthetic facial model consisted of five distinct layers which cover the skull and are connected by springs. Moreover, an articulated neck and synthesized subsidiary organs, such as eyes, eyelids, and teeth, were included to improve the reality of the facial animation. However, this model didn't consider the personal face tissue properties, which was generally important for facial surgical simulation. With the incorporation of individual facial data when constructing a facial model, Koch *et al.*[81] described a prototype system for surgical planning and prediction of human facial shape after craniofacial and maxillofacial surgery for patients with facial deformities. They modeled the facial data using triangular non-linear finite elements and connected the surface with nodal springs to the skull. The individual spring stiffness parameters were computed from the underlying CT data. The resulting shape was then generated from minimizing the global energy of the surface under external forces. The system also considered boundary conditions, stretching and bending forces, as well as nodal loading forces to accomplish facial simulations.

There are many other applications for physics-based modeling including providing dynamic deformations for superquadrics [98], using physics-based model to simulate artificial life (*e.g.*, fishes) [142], simulating facial surgery using physics-based models [81, 85], simulating clothes with physics-based modeling techniques [6], directly manipulating physics-based B-spline surfaces using haptics [34], etc. Physics-based methods can even be applied to implicit functions for realistic manipulation of implicit objects. For example, Hua and Qin [70] recently proposed an integrated approach to couple the physics-based methods with implicit B-spline functions. In general, the physics-based techniques can be incorporated into conventional geometric modeling framework for more realistic manipulations of geometric entities.

Because physics-based models are formulated through differential equations, it's straightforward to integrate them with PDE techniques. This dissertation has unified the physics-based techniques with PDE surface and solid models to obtain interactive sculpting of PDE objects.

3.4 Medial Axis Extraction Techinques

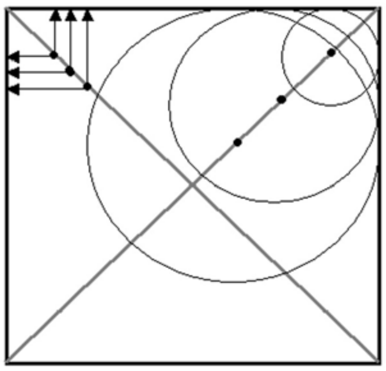


Figure 3.2: 2D illustration for medial axis.

Medial axis, also known as skeleton, offers much more simple and compact representations for arbitrary complex geometric and/or solid objects. Ever since it was first proposed and named by Blum [18, 19], medial axis has started to gain more and more popularity in visual computing areas especially in recent years. It collectively provides useful shape information such as topology, orientation, and local properties in an intuitive and compact fashion. For instance, the medial axis of a 2D polygon can be directly associated with the concept of *grassfire transform*: by igniting boundary points of the polygon, the fire propagates inward from the boundary at a uniform speed, and where the fire front meets and extinguishes itself defines the medial axis in a natural and physically plausible way. More mathematically, the medial axis can be defined as the locus of all centers of circles inside the 2D polygon (or spheres inside the 3D object) that are tangent to the boundary in two or more places [21]. The points on the medial axis (or skeleton) of an object usually have more than one closest point on the boundary of the object. Fig. 3.2 shows an illustration of the medial axis for a 2D shape. In practice, medial axis is also called medial surface and frequently referred as the

3D skeleton (especially in bio-medical applications) for 3D models. Hence the extraction of medial axis is oftentimes called skeletonization.

There are several unique advantages of using medial axis or skeleton to model geometric objects. First, it provides localization of features such as anatomical landmarks (which are extremely valuable in bio-medical applications). Second, it separates thickness information (e.g., radius of medial axis or skeleton) from orientational and topological information, i.e., shape features can be subdivided into radial, orientational and location information in order to facilitate statistical analysis. Third, shape differences between objects can be quantified in a more intuitive and accurate way. Fourth, it is more expeditious to capture coarse-scale changes from the acquired models, making it more stable and robust to handle noisy datasets.

In the past several decades, medial axis extraction has been well studied and there are various techniques for detecting medial axes of 2D and 3D objects. Brief reviews of several typical approaches computing medial axes or skeletons are listed as follows:

- **Thinning**

  To extract medial axis of an object, one intuitive way is to peel off the object's boundary layer by layer. Such *thinning* process can be performed iteratively in the discrete domain. It will retain points on skeletons and maintain object's topology [3, 84, 95]. However, thinning-based methods are fundamentally *discrete* processes and require fully segmented, compact, and connected objects. These techniques have difficulties to deal with partial data and are sensitive to Euclidean transformations of the data.

- **Distance functions**

Because the skeletal or medial surface points usually coincides with the singularities of an *Euclidean distance function* to the boundary, distance functions can be employed for medial axis extraction. The approaches based on distance functions construct distance field transformation of an object and extract the medial axis based on the distance field [4, 8, 57, 63, 87]. However, usually it's difficult to ensure homotopy with original objects using techniques based on distance functions.

- **Voronoi skeletons**

  Because the vertices of the Voronoi diagram of a set of boundary points can converge to the skeleton as the sampling rate increases under appropriate smoothness conditions [122], Voronoi diagram and its dual Delaunay triangulation have been widely adopted for medial axis extraction [2, 40, 62, 103, 107, 128, 130, 129]. Such methods can preserve topology and accurately localize skeletal or medial surface points for densely sampled object. However, for algorithms based on Voronoi diagrams, it's more time consuming to build a 3D Voronoi diagram with increasing number of sample points, thus, direct computing method for Voronoi skeletons is less suitable for large datasets.

- **Level set method**

  Another class of methods casts the surface as the level set of a 4D embedded object and finds the weak solutions of a PDE which models the wave propagation process whose singularities yield the medial axis. Kimmel *et al.* [75] introduced a level-set-based method for skeletonization using numerical approximation of distance maps of an object. Ma *et al.* [92] proposed a practical approach for extracting skeletons from general 3D models using radial basis functions (RBFs).

- **Direction testing**

  Bloomenthal and Lim [10] proposed an implicit method based on direction testing that defines the skeleton as the set of points at which the direction to the nearest point on the object undergoes a sudden transition. The geometric skeleton is derived from a static object using an implicit *direction* method. The object may be reconstructed from the modified skeleton using implicit distance and convolution techniques.

- **Hybrid techniques**

  In addition, many skeletonization techniques combine several aforementioned methods into a single framework for medial axis extraction. For instance, Siddiqi *et al.* [21] proposed a method combining the thinning process and the distance transformations and using a Hamilton-Jacobi equation to calculate medial axes of volume data. This method provides accurate medial axis extractions and preserves homotopy of objects. However, it mainly focuses on volumetric datasets. Medial axis extraction for arbitrary objects bounded by polygonal meshes hasn't been considered. And sometimes the *real* medial axis for an irregular complex model may have noisy branches which are difficult to handle in the interest of shape manipulation.

To make use of the appealing features of medial axes in the PDE-based modeling system, this dissertation employs a diffusion-based equation to approximate skeletons of objects bounded by arbitrary meshes (or other boundary representations). It also provides sculpting toolkits to manipulate skeletons and uses diffusion-based front propagation techniques to recover deformed objects according to skeleton deformations.

Chapter 4

Physics-based PDE Surfaces

PDE surfaces, which are defined as solutions of certain PDEs, offer many modeling advantages in surface blending, free-form surface modeling, and specifying surface's aesthetic or functional requirements. This dissertation presents an integrated approach that can unify static PDE surfaces with physics-based modeling method and spline-based techniques, in order to realize the full potential of PDE methodology. It provides PDE surface manipulation toolkits that allow interactive design of flexible topological surfaces as PDE surfaces and displacements using generalized boundary conditions as well as a variety of geometric and physical constraints, hence supporting various interactive techniques beyond the conventional boundary control. This work has been published in the proceedings of EuroGraphics 2000 [43] and Pacific Graphics 2000[44], and accepted by the Journal of Graphical Models[46].

4.1 Introduction and Motivation

Surface modeling techniques are fundamental for many visual computing applications including interactive graphics, CAD/CAM, animation, and virtual environments. There are various geometric techniques to model surfaces including spline-based methods, subdivision models, implicit functions, etc. Different with aforementioned techniques, PDE techniques offer a unique surface representation that defines surfaces as solutions of elliptic bivariate PDEs of 3D coordinate vectors with provided boundary conditions. The interior information of the surfaces can be automatically recovered using given boundary information, which alleviates the burden of specifying tedious control points or other means to define a surface object. Because of differential properties of the underlying PDE, the constructed surface has high-order continuities throughout the parametric domain. Moreover, the underlying PDE often relates to certain energy functionals to offer optimization properties. However, despite the rapid advances and modeling successes of PDE surfaces, there are lack of a set of novel interactive techniques to realize their full potential. Typical modeling difficulties associated with traditional PDE surfaces include:

1. The prior work on PDE surfaces mainly concentrated on static elliptic PDEs and is lack of interactive techniques for direct shape manipulation.

2. Besides simple geometric conditions along PDE surface boundaries, as well as manually editing on PDE coefficients, there is a lack of formal mechanism to directly manipulate PDE surfaces in general.

3. Traditional elliptic PDE surfaces only result from Hermite-like boundary conditions (*i.e.*, boundary curves and their corresponding derivatives up to order n at one parametric direction). More flexible and general boundary

constraints are not yet addressed.

4. Conventional PDE surface techniques are unable to support localized geometric operations. Global control is less intuitive to manipulate.

To ameliorate it, we [43] proposed an interactive method and developed novel modeling techniques that can facilitate the direct manipulation and interactive sculpting of dynamic PDE surfaces. The presented algorithms and design framework are founded upon the integrated principle of differential equations and physics-based modeling. To further promote the applicability of PDE surfaces in interactive graphics, CAD/CAM, and virtual engineering, we [44] extended both the geometric coverage and topological variation of PDE surfaces. The improved system provides users a set of more powerful sculpting tools than previously-developed point-based editing capabilities. These toolkits allow PDE surfaces to be defined through the use of general, flexible boundary constraints. PDE surfaces of complicated geometry and diverse types of topology are available in the PDE-based modeling environment. Other typical design tools in the environment include merging multiple surfaces, trimming surface parts, manipulations of isoparametric curves and/or arbitrary curve networks, editing any user-specified sub-surface, local modification of blending coefficients, etc. The PDE modeling system also offers B-spline approximation and sculpting to facilitate the data exchange with other geometric modeling techniques. Through the system, users are able to enforce both physical requirements and geometric criteria on PDE surfaces simultaneously with ease. To further extend PDE techniques for manipulation of existing models, the PDE formulation is employed to model displacements on parametric surfaces [46]. The displacement model defines surfaces as original surfaces plus PDE-governed displacements and surface deformation can be achieved by manipulating surface displacements through interactive toolkits. This extension allows users to directly model existing parametric surfaces through the system. It facilitates data exchange of PDE techniques with other parametric models.

4.2 Formulation of PDE Surfaces

The fourth-order elliptic PDE to model PDE surfaces in the PDE modeling system is a generalized version of (2.5) by replacing the constant control coefficient a with an arbitrary function $a(u, v)$ in the interest of local control:

$$(\frac{\partial^2}{\partial u^2} + a^2(u, v)\frac{\partial^2}{\partial v^2})^2\mathbf{X}(u, v) = \mathbf{0} \tag{4.1}$$

where u, v are parametric coordinates over the 2D parametric domain, $a(u, v)$ is a blending coefficient function of u and v that controls the contributions of partial derivatives along u and v directions locally, and

$$\mathbf{X}(u, v) = \left[\begin{array}{ccc} x(u, v) & y(u, v) & z(u, v) \end{array}\right]^\top$$

defines the PDE surface coordinates in 3D space. Note that, in (2.5) the control coefficient is a constant a. To offer users more flexibility for interactive manipulation, this constant coefficient is replaced by an arbitrary function of u and v, which can be defined by users. Because $a(u, v)$ varies across entire PDE surface $\mathbf{X}(u, v)$, local control on PDE surfaces can be achieved. Moreover, although the system focuses on this particular elliptic PDE, the mathematical derivation and its associated numerical techniques can be readily generalized to other PDEs. To solve (4.1), at least four boundary conditions are required in order to derive a unique solution. The PDE modeling system assumes that a PDE surface is either closed or open geometrically along its two parametric directions (*i.e.*, u and v). Therefore, PDE surfaces may be topologically flexible, yielding diverse types of surfaces equivalent to four-sided open patches, spheres, cylinders, and tori. u and

v can be restained to vary between 0 and 1, because reparameterization process can be easily conducted without changing the geometry of PDE surfaces if either u or v belongs to any $[a, b]$. Boundary conditions to define a PDE surface can be classified into three types: (1) open along both u and v directions, (2) open along u direction and closed along v direction, and (3) closed along both directions. In addition to the traditional Hermite-like boundary constraints, to enhance the cross-sectional design of PDE surfaces from a set of curves, the boundary conditions are generalized to a curve network. For instance, consider the design techniques of Gordon surface and Coons patch, the generalized boundary constraints can have the following form:

$$\mathbf{X}(u_i, v) = \mathbf{f}_i(v), \mathbf{X}(u, v_j) = \mathbf{g}_j(u), \tag{4.2}$$

where $0 \leq u_i \leq 1$ and $0 \leq v_j \leq 1$, and $\mathbf{f}_i(v)$ and $\mathbf{g}_i(u)$ are isoparametric curves. Moreover, a set of non-isoparametric curves can be easily added into our formulation.

By interactively modifying generalized boundary constraints, users are capable of manipulating the entire surface in an *indirect* manner. This property offers designers an efficient way to edit the PDE surface through a fewer number of parameters that define boundary curves.

4.3 PDE-based Displacement Surfaces

The idea of displacing a surface by a function was introduced by Cook [30]. Displacement maps are often used for texture mapping of bumped surfaces or modeling of complex detailed meshes of arbitrary topology with regular surface

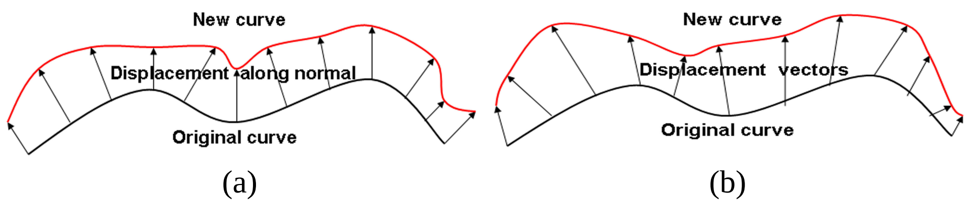


Figure 4.1: Illustration of the idea of displacement model. (a) Displacement curve model based on the displacements along the original curve normals; (b) displacement curve model based on the displacement vectors on the original curve.

patches. The complex surface can be represented as scalar/vector-valued displacements over a smooth domain surface. The displacement maps can be viewed as images, and this type of representation facilitates the use of image processing operators for manipulating the geometric detail of an object. They are also compatible with modern photo-realistic rendering system [82]. The idea is also used in subdivision techniques to produce *displaced subdivision surfaces* [83] and multiresolution surfaces [78]. Displacement maps can decrease the complexity of the model. The advantage of this representation lies in its simplicity and flexibility. The natural hierarchical division between coarse and fine features allows rapid computation of local surface features, and makes the data structure ideal for rapid collision detection for interactive operations. Furthermore, since local features can be represented by an array of scalar values, limited editing of the local geometry can be done rapidly by modifying the values in the displacement map [72]. An illustration of the idea of displacement models is shown in Fig.4.1.

To further explore the modeling potentials of PDE techniques on existing surface models, this dissertation employs the PDE formulation on surface displacements, *i.e.*, the offsets of the input surface. The target surface will be the result by adding the corresponding displacements onto the original surface. Different with popular displacement techniques, the proposed PDE method will model displacement vector maps of the surface instead of the scalar-valued map associated with surface normals and leave the underlying surface unchanged. The formulation of PDE displacement model is a slightly modified version of (4.1):

$$\begin{aligned} &\mathbf{X}(u,v) = \mathbf{X}^0(u,v) + \mathbf{O}(u,v), \\ &(\tfrac{\partial^2}{\partial u^2} + a^2(u,v)\tfrac{\partial^2}{\partial v^2})^2\mathbf{O}(u,v) = \mathbf{0}, \end{aligned} \tag{4.3}$$

where $\mathbf{X}(u,v)$ is the target surface, $\mathbf{X}^0(u,v)$ is the original surface, and $\mathbf{O}(u,v)$ is the corresponding surface displacements. Note that, (4.1) is the simplest case of (4.3) by simplifying the original surface to $\mathbf{X}^0(u,v) = \mathbf{0}$, *i.e.*, shrinking to a point at the coordinate origin.

4.4 Numerical Approximation Techniques

Prior work on PDE surfaces mainly seeks closed-form analytic solutions in order to exploit many attractive properties associated with analytic formulations for surface design. However, in the interest of allowing arbitrary boundary conditions and direct surface manipulations, this dissertation resorts to numerical techniques that guarantee approximate solutions of the integrated formulation for PDE surfaces of flexible topology. Numerical algorithms also facilitate the material modeling of anisotropic distribution and its realistic physical simulation, where there are no closed-form analytic solutions available for PDE surfaces. Among many mature techniques, two popular numerical approaches are employed for the PDE surface modeling framework: (1) finite-difference discretization, and (2) finite-element method based on B-spline approximation.

The finite-difference method (FDM) is to transform a PDE into a system of algebraic equations by sampling the parametric domain into regular grids, then replacing all the partial derivatives in the differential equation with their discretized approximations on the sample points. The algebraic equations can then be solved numerically either through an iterative process or a direct procedure in order to obtain an approximate discrete solution to the continuous PDE. The details of FDM techniques will be discussed in Chapter 8. By substituting the finite-difference representation at each grid point, (4.1) can be rewritten in matrix form as:

$$\mathbf{H}\mathbf{X} = \mathbf{z}, \tag{4.4}$$

where $\mathbf{H}$ represents the discretized differential operator in $(m \times n) \times (m \times n)$ matrix form. $\mathbf{H}$ is also controlled by the blending function $a(u, v)$. $\mathbf{X}$ is the collection of the unknown position vectors of the discretized sample points on the PDE surface, and $\mathbf{z}$ depends on the value of the boundary constraints.

$$\begin{aligned}
&\mathbf{H} = \left[\mathbf{H}_{(0,0)}, \mathbf{H}_{(0,1)}, \cdots, \mathbf{H}_{(m-1,n-1)}\right]^{\top}, \\
&\mathbf{H}_{(i,j)} = \left[H_{(i,j),(0,0)}, H_{(i,j),(0,1)}, \cdots, H_{(i,j),(m-1,n-1)}\right], \\
&\mathbf{X} = \left[\mathbf{x}_{(0,0)}, \mathbf{x}_{(0,1)}, \cdots, \mathbf{x}_{(m-1,n-1)}\right]^{\top}, \\
&\mathbf{z} = \left[\mathbf{z}_{(0,0)}, \mathbf{z}_{(0,1)}, \cdots, \mathbf{z}_{(m-1,n-1)}\right]^{\top}.
\end{aligned}$$

The matrix $\mathbf{H}$ is called ”tridiagonal with fringes” [113].

Similarly, the PDE for displacements in (4.3) can be discretized using finite-difference method in the form of

$$\mathbf{H}\mathbf{O} = \mathbf{z}.$$

Thus, (4.3) has the approximate numerical form

$$\begin{aligned}
\mathbf{X} &= \mathbf{X}^0 + \mathbf{O}, \\
\mathbf{H}\mathbf{O} &= \mathbf{z}.
\end{aligned} \tag{4.5}$$

Different topological types are available of PDE surfaces. First, the surface can be closed along one parameter direction (*e.g.*, v), in which case the points on $v = 0$ are the same as those on $v = 1$. The central-difference scheme suffices for the computation of partial derivatives with respect to v. Second, the PDE surface is open along both u and v directions. In this case, the computation of partial derivatives on two boundary curves requires special care, and forward or backward differences shall be utilized along the open boundary curves instead. Third, the PDE surface is closed along both directions, and the central-difference approximation can be applied anywhere across the surface geometry. Boundary constraints determine all the point coordinates lying on the user-specified curves. Moreover, for the Hermite-like boundary conditions, the initial derivative information across boundary curves determines additional point coordinates in the vicinity of specified boundaries (*e.g.*, $\mathbf{x}_{1,j}$ and $\mathbf{x}_{m-2,j}$) that are adjacent to two boundary curves at $u = 0$ ($\mathbf{x}_{0,j}$) and $u = 1$ ($\mathbf{x}_{m-1,j}$). Arbitrary boundary conditions can be easily enforced without any difficulty using finite-difference method. Note that, in spite of certain combinations of constraint imposition shown in the examples, in general this type of elliptic PDEs allows the boundary conditions to be explicitly formulated in arbitrary form. This permits designers to choose (various) constraints based on diverse design tasks. The same flexible topological feature also applies to PDE displacements. However, the topological type of displacements depends on the underlying original surface. This gives designers more freedom when modeling surfaces of flexible topology.

With the boundary conditions, (4.4) can be solved using direct methods like Gaussian-Elimination or finite-difference-based iterative techniques such as Gauss-Seidel iteration or SOR iteration. Nonetheless, the discretization of the parametric space results in a very large number of linear equations. This causes the slow convergence of iterative methods. To achieve a solution faster, the equations can be solved first at a coarse grid with down-sampled constraints and interpolate the solution at finer grids to compute the initial guess for iterative methods at the finer resolution. The convergent rate of this multi-grid iterative solver can be greatly increased.

4.5 Combining Physics-based Modeling

The physics-based modeling methodology can be unified with PDE approach, mainly because the dynamic behavior of physics-based models is also controlled by certain differential equations (*e.g.*, Lagrangian equations of motion). Hence, physics-based modeling augments (rather than replaces) the existing PDE methodology, offering extra advantages for shape modeling. Since the majority of physical phenomena can be characterized by PDEs, and the physical laws employed by the physics-based modeling are in the format of certain differential equations, it is natural to view the physics-based modeling as a member of the general PDE modeling and simulation category. With the incorporation of physics-based modeling approaches into the PDE framework, users can achieve real-time sculpting of PDE surfaces.

The Lagrangian mechanics are associated with the discretized PDE (refer to (4.4)) for the unified PDE modeling framework by attaching mass points on geometric grids and adding springs between immediate neighbors on the discretized PDE mesh, as shown in Fig. 4.2, then a dynamic version of PDE model can be obtained:

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{D}\dot{\mathbf{X}} + (\mathbf{K} + \mathbf{H})\mathbf{X} = \mathbf{z} + \mathbf{f}. \tag{4.6}$$

At the equilibrium, if stiffness distributions as well as the external force $\mathbf{f}$ are zero, (4.6) reduces to (4.4) with additional material properties.

The composite dynamic PDE displacement surface model can be obtained in the same way:

$$\begin{aligned} &\mathbf{X} = \mathbf{X}^0 + \mathbf{O}, \\ &\mathbf{M}\ddot{\mathbf{O}} + \mathbf{D}\dot{\mathbf{O}} + (\mathbf{K} + \mathbf{H})\mathbf{O} = \mathbf{z} + \mathbf{f}, \end{aligned} \tag{4.7}$$

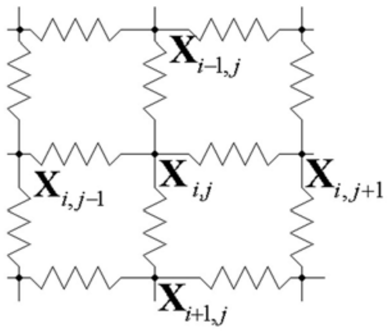


Figure 4.2: Mass-spring network for the discretized PDE surface.

where

$$\ddot{\mathbf{O}} \approx (\mathbf{O}^{t+\Delta t} - 2\mathbf{O}^{t} + \mathbf{O}^{t-\Delta t})/\Delta t^2, \dot{\mathbf{O}} \approx (\mathbf{O}^{t+\Delta t} - \mathbf{O}^{t-\Delta t})/2\Delta t.$$

By allowing the PDE model to dynamically deform in time domain, users will have a natural feeling when they interactively manipulate the PDE model, which is lacking without Lagrangian equations of motion. Furthermore, material properties can be introduced to govern the behavior of the underlying PDE model. This *hybrid* formulation permits users to obtain a surface that satisfies both geometric criteria and functional requirements at the same time.

4.6 Interactive Sculpting Techniques for PDE-based Surfaces

The PDE modeling system provides various interactive techniques for PDE-based surface sculpting.

4.6.1 Surface Initialization

The PDE modeling system supports three topological types of PDE surfaces. At the beginning of the initialization phase, users must specify the surface type, *i.e.*, whether the surface is open or closed along u and v directions. Because any direct manipulation must be based on the user-defined initial surface, users need to select boundary conditions to generate a PDE surface as an initial step. The system provides users two different ways to set up boundary conditions of the PDE surface. First, users can interactively input some control points by clicking/dragging the mouse at desired locations on the screen, and the system will calculate cubic B-spline curves as boundary curves, boundary derivative curves, or other special curves the PDE surface must interpolate. The boundary derivative curves are two curves corresponding to the two boundary curves, respectively. The difference between any point on each derivative curve and its associated point on the boundary curve will be used to determine both the magnitude and the direction of the tangent vector across the two boundary curves. Alternatively, users are allowed to define boundary conditions using certain analytic functions. The point coordinates are sampled along the analytic curves and saved into a data file. The system then can access data files and initialize the PDE surface based on analytic function curves. After the boundary conditions are determined, the PDE surface can be derived from the solution of the linear equations subject to these conditions.

4.6.2 Generalized Boundary Constraints

The solution of (4.1) is subject to boundary conditions. In general, there are several types of boundary conditions according to the information they contain. Currently, this dissertation considers three kinds of boundary constraints:

(1) Hermite-like constraints; (2) Coons-like constraints; and (3) Gordon-like constraints in analogy with their corresponding free-form surface formulation.

Hermite-like conditions include positions and the first-order or even higher-order derivatives of boundary curves. For the fourth-order PDE shown in (4.1), the boundary conditions may be Hermite-like (*i.e.*, two boundary curves at $u = 0$ and $u = 1$, and their corresponding first-order derivatives). The two boundary curves define the edges of the surface and the two derivative curves determine the gradient information across the boundaries, which outline the surface shape. Fig. 4.3 and Fig. 4.6 show examples of this type of conditions.

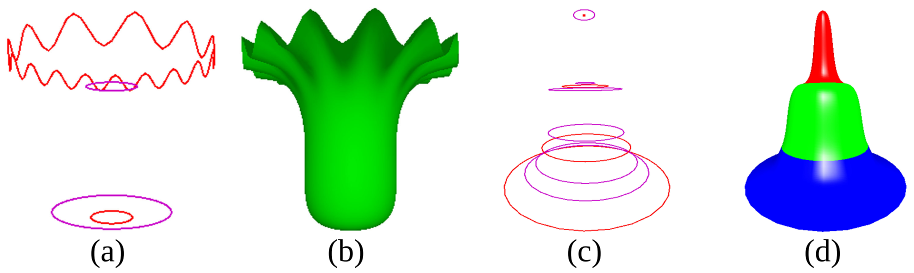


Figure 4.3: PDE surfaces from Hermite-like boundary conditions. (a) Boundary conditions, where boundary curves are in red and derivative curves in pink; (b) the surface subject to (a); (c) three sets of boundary and derivative curves for a connected PDE surface; (d) the connecting result.

For any four-sided surface patch, there are four boundary curves in general. In the parametric domain of u and v, the boundary curves are those at $u = 0$, $u = 1$, $v = 0$, and $v = 1$, respectively. This kind of conditions, in analogy with Coons patch, is considered as Coons-like boundary conditions. Using such conditions, users can easily obtain surfaces that are open along both u-direction and v-direction, or closed along v and open along u. Note that, for surfaces that are closed only along v, it is equivalent to consider that two boundary curves at $v = 0$ and $v = 1$ are the same. Fig. 4.4 has an example of this boundary type.

Although four boundary curves can provide a solution of the PDE surface, they are far from enough to define complex geometry, especially when users seek the solution for PDE surfaces that are closed along both directions of u and v (*e.g.*, tori). In this scenario, users need to define a curve network that the PDE surface must interpolate. This kind of boundary constraints is a direct generalization of Gordon surfaces [54]. Hence, the Gordon-like boundary conditions consist of a family of isoparametric curves $\mathbf{X}(u_i, v) = \mathbf{f}_i(v)$ and $\mathbf{X}(u, v_j) = \mathbf{g}_j(u)$, where $0 \leq u_i \leq 1$ and $0 \leq v_j \leq 1$. An example of this type of geometric construction is shown in Fig. 4.4. In the example, the boundary curves at $u = 0$, $v = 0$ (which is the same as $v = 1$), $v = 0.25$, $v = 0.5$, and $v = 0.75$ are specified. The control function $a(u, v) = 4.3$.

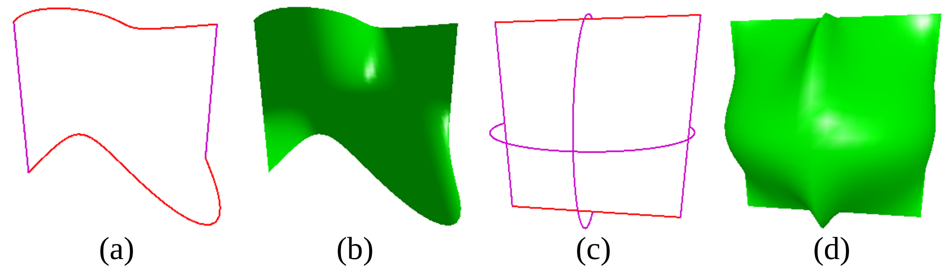


Figure 4.4: PDE surfaces with Coons-like and Gordon-like boundary conditions. (a) Coons-like boundary curves; (b) the corresponding surface of (a); (c) Gordon-like curve network with curves at $u = 0$, $u = 0.5$, $u = 1$, $v = 0$, $v = 0.5$, and $v = 1$; (d) the PDE surface from (c).

If the boundary curves are B-spline curves, users can modify the shape of the PDE surface globally by changing the B-spline control points of boundary curves. If the boundary curves are obtained through certain analytic functions, users can change them in the same way as adding additional conditions discussed in following sections.

4.6.3 Multi-Grid PDE Surface Subdivision

Although the iterative techniques are easily implemented, oftentimes the large number of sample points of a PDE surface result in the slow convergence of such techniques. To improve the computation performance, a multi-grid approximation based on popular subdivision schemes is proposed. At first, starting with a small number of sample points on the coarsest grid of a PDE surface, the coarse solution of the PDE surface can be easily obtained quickly. Second, users can refine the coarse mesh through the simple linear interpolation or more complicated subdivision schemes such as Butterfly subdivision [52] or quadrilateral interpolating subdivision [77]. Then the new subdivided mesh can be used as an initial guess for successive iterations. The finer grid is then computed iteratively to achieve a more accurate and smoother solution of the PDE surface. During the multi-grid process, the up-sampling of all generalized boundary curves is achieved through the use of four-point interpolatory subdivision scheme [52] in order to guarantee the smoothness requirement of the refined curves. Given control points $\{\mathbf{x}_i^0\}_{i=-2}^{n+2}$, the points at level $k+1$ of the subdivision are defined by

$$\begin{aligned} \mathbf{x}_{2i}^{k+1} &= \mathbf{x}_i^k && -1 \le i \le 2^k n + 1 \\ \mathbf{x}_{2i+1}^{k+1} &= (\tfrac{1}{2} + w)(\mathbf{x}_i^k + \mathbf{x}_{i+1}^k) - w(\mathbf{x}_{i-1}^k + \mathbf{x}_{i+2}^k) && -1 \le i \le 2^k n + 1 \end{aligned} \tag{4.8}$$

According to [52], the curve is tightened toward the control polygon as $w \to 0$, and for any $0 < w < (\sqrt{5} - 1)/8$, the interpolated curve is a C^1 curve. Because w influences the smoothness of the boundary curves, the system allows users to change the value of w in order to obtain satisfactory results. Fig. 4.5 shows an example using the multi-grid subdivision.

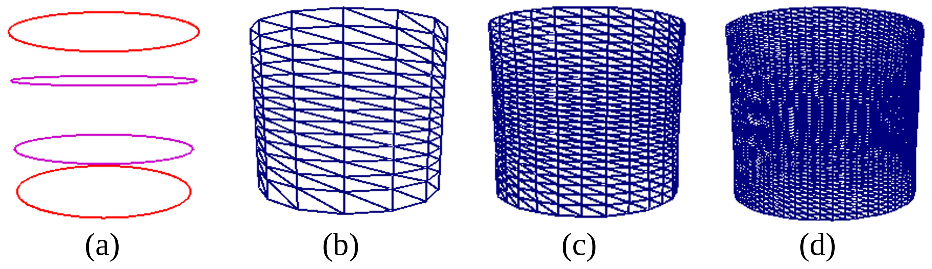


Figure 4.5: The multigrid solution for a PDE surface. (a) Initial boundary conditions; (b) PDE surface of sampling grids 15×15; (c) PDE surface of sampling grids 30×30; (d) PDE surface of sampling grids 60×60. Note that, w is 0.1 in this example.

4.6.4 Manipulating Boundary Conditions

Because boundary curves are defined by B-spline curves, or have B-spline approximation, we can modify the shape of the PDE surface globally by changing B-spline control points of boundary curves. Fig. 4.6 shows an example.

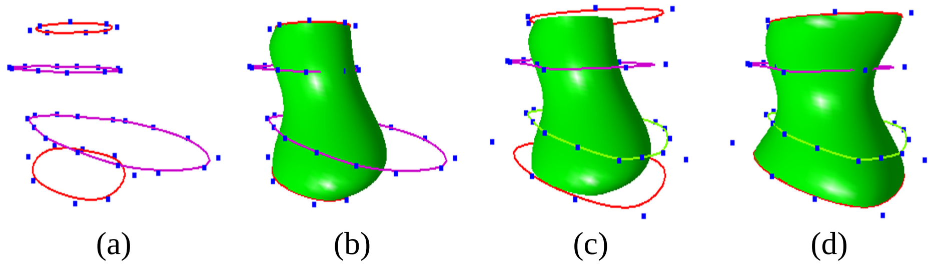


Figure 4.6: Changing direct-input B-spline boundary conditions of a PDE surface. (a) Initial B-spline boundary curves with control points; (b) the corresponding PDE surface; (c) modified boundary conditions; (d) the modified surface.

4.6.5 Modifying Control Function *a(u,v)*

The blending coefficient function $a(u, v)$ can also influence the surface shape. The value of $a(u, v)$ controls the relative smoothness and the level of variable dependence between the parametric directions of u and v. For a large $a_{i,j}$ at point $\mathbf{x}_{i,j}$, changes in the u direction occur within a relatively short length scale, *i.e.*, it is $1/a_{i,j}$ times the length scale in the v direction in which similar changes can take place. Consequently, the user can control how boundary conditions influence the interior of the surface by modifying the length scale (*i.e.*, $a_{i,j}$) at arbitrary point on the PDE surface. In general, the control function $a(u, v)$ can be interactively "painted" over the entire surface (see Fig. 4.7).

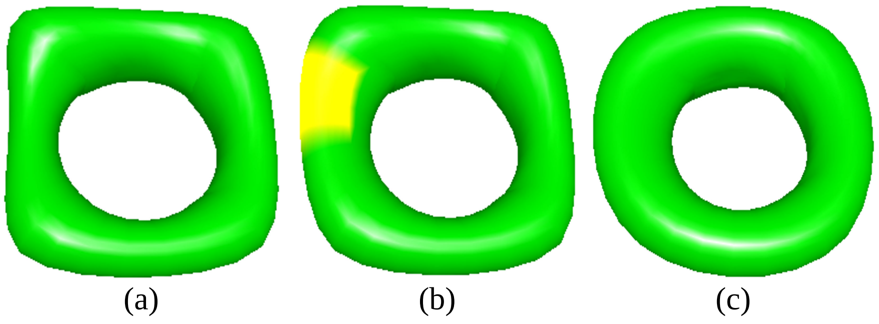


Figure 4.7: Effects of changing blending coefficient $a(u, v)$. (a) The PDE surface of $a(u, v) = 3.0$; (b) the surface after changing value of $a(u, v)$ on the yellow part on the surface to 5.0; (c) the surface by setting $a(u, v) = 5.0$ for all sample points.

4.6.6 Joining Multiple Surfaces

Oftentimes a single PDE surface may not satisfy complicated design requirements, because real-world objects exhibit both complex topological structure and irregular geometric shape. Patching multiple PDE surfaces together can provide such shapes. In the PDE modeling system, users can join $n-1$ PDE surfaces sequentially by specifying $2n$ Hermite-like boundary conditions (where $n \geq 3$). Note that, $2n$ conditions are necessary because two neighboring PDE patches share one common boundary. To satisfy C^1 continuity, the tangent vectors across the shared boundary must be the same. Note that, because the coefficient function $a(u, v)$ in (4.1) may vary throughout the $u-v$ domain, the technique of joining multiple surfaces can be considered to be equivalent to generating one "larger" PDE surface with different local control. Fig. 4.3 has an example.

4.6.7 Sculpting Tools and Geometric Constraints for Global and Local Deformation

By changing boundary curves, users can modify the entire shape of a PDE surface, *i.e.*, users obtain global deformation of the PDE surface. However, when the global appearance of a PDE surface is satisfactory, any subsequent sculpting via boundary conditions may destroy certain already-existing nice features of the underlying surface. In this situation, making small changes on a localized region is more desirable. This can be done by enforcing additional constraints on the PDE surface. Note that, the original finite-difference formulation consists of $m \times n$ equations and $m \times n$ unknowns, *i.e.*, the coefficient matrix is a square matrix. The introduction of additional conditions forces the system to incorporate a set of new equations into the original set. In this dissertation, such additional constraints are treated as hard constraints, *i.e.*, the additional equations must be satisfied. In this case, the system needs to explicitly formulate constraints and incorporate these additional constraints into the original equations. This can be done by replacing the corresponding equations in the original system with these hard constraints. For example, if users want to move a sample point on the discretized surface to a new location, $\mathbf{x}_{i,j} = \mathbf{x}_0$, the equation $\mathbf{x}_{i,j} = \mathbf{x}_0$ will be used to replace the corresponding discretized difference equation approximating the PDE at the point $\mathbf{x}_{i,j}$, *i.e.*, $H_{i\times n+j, i\times n+j} = 1$, all other $H_{(i\times n+j,k} = 0$ for $k \neq i \times n + j$, and $\mathbf{z}_{i\times n+j} = \mathbf{x}_0$ in (4.4). This method works well if the additional constraints are of linear form (*e.g.*, fixing a subset of certain unknowns or three points must be co-linear, etc.). As a result, (4.4) becomes

$$\mathbf{H}_c\mathbf{X} = \mathbf{z}_c, \tag{4.9}$$

where the additional constraints are explicitly formulated and enforced within the original equations. For such situations, the direct solver may not guarantee a satisfying result. So the iterative method can be performed to get the approximate solution. Accordingly, the physics-based PDE model after enforcing additional constraints can be formulated as:

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{D}\dot{\mathbf{X}} + (\mathbf{K} + \mathbf{H}_c)\mathbf{X} = \mathbf{z}_c + \mathbf{f}, \tag{4.10}$$

Point Editing

The PDE modeling system permits users to interactively sculpt PDE surfaces by enforcing additional constraints on a set of selected points as well as their normal and curvature:

Point Sculpting To manipulate a surface directly, one desirable way is to enforce the PDE surface interpolating certain specific locations in 3D space. This can be done by picking a point on the sampling surface grid, say $\mathbf{x}_{i,j}$, then dragging it to the position where users want the surface to pass through. The point can be arranged to stay anywhere within users' view frustum, say $\mathbf{p}$. Then the system can replace the corresponding difference equation for point $\mathbf{x}_{i,j}$ in (4.4) by the

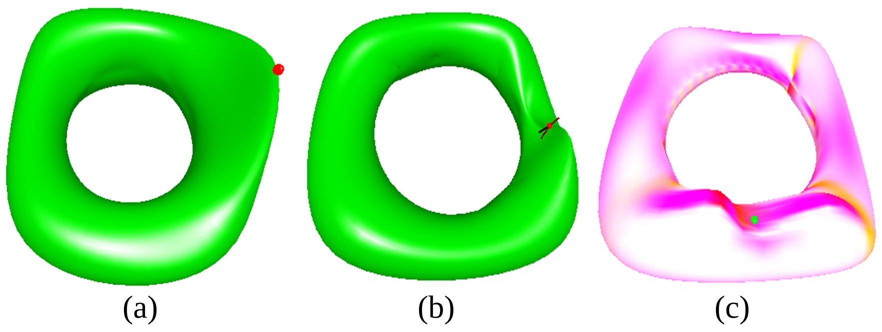


Figure 4.8: Examples of point-based manipulation. (a) Changing the location of a surface point; (b) normal modification of a selected point on the PDE surface; (c) the modified surface after changing curvature at a selected point.

equation $\mathbf{x}_{i,j} = \mathbf{p}$. The updated linear equation system are re-solved. Because the point $\mathbf{x}_{i,j}$ are specified to interpolate the position, the value of the point is not allowed to change during the approximating iteration. The surface will be deformed according to the modification. Users can edit a set of points in a sequential order, and the modified surface interpolates all the select data points. Fig. 4.8 (a) shows a modified PDE surface by changing the position of one point on the original surface. And Fig. 4.18 (a) shows a snapshot of modifying a point on a dynamic PDE surface.

Normal Rotation Users can also manipulate the surface normal on any point to achieve a local editing capability in the vicinity of the data point, as demonstrated in Fig. 4.9. This is because the normal of a continuous surface at a selected surface point can be approximated by its neighbors using finite-difference method: $\mathbf{n}_{i,j} = \frac{\mathbf{x}_{i+1,j} - \mathbf{x}_{i-1,j}}{2\Delta u} \times \frac{\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j-1}}{2\Delta v}$.

When users rotate the normal at the selected point, the system will subsequently compute the new positions of the four neighboring points according to the new normal direction. Then the new positions of these four neighboring points are

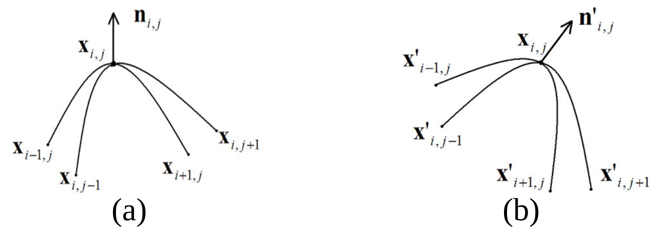


Figure 4.9: The change of the normal at a point corresponds to the change of positions of its neighbors. (a) Normal $\mathbf{n}_{i,j}$ of $\mathbf{x}_{i,j}$; (b) the new normal $\mathbf{n}'_{i,j}$ leads to the change of points $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j-1}$ and $\mathbf{x}_{i,j+1}$.

formulated into four equations and the system simply enforces four new equations within the linear equation system (4.4). By solving the constrained equations, the modified surface with the changed normal at the selected point can be obtained. An example for this kind of constraints is shown in Fig. 4.8 (b).

Curvature Manipulation Since the curvature measures the intrinsic shape of a curve/surface, users can also modify the curvature at arbitrary point. Users are allowed to modify the surface curvature at any point along u-direction and v-direction, respectively. Curvature changing can be achieved by modifying the neighboring points. The curvature of a curve can be defined by: $\kappa = \frac{\|\mathbf{x}' \times \mathbf{x}''\|}{\|\mathbf{x}'\|^3}$. Now the curvature at any surface point along u-direction and v-direction can be defined as:

$$\kappa_u = \frac{\|\frac{\partial \mathbf{X}}{\partial u} \times \frac{\partial^2 \mathbf{p}}{\partial u^2}\|}{\|\frac{\partial \mathbf{X}}{\partial u}\|^3}, \kappa_v = \frac{\|\frac{\partial \mathbf{X}}{\partial v} \times \frac{\partial^2 \mathbf{X}}{\partial v^2}\|}{\|\frac{\partial \mathbf{X}}{\partial v}\|^3}. \tag{4.11}$$

It implies that changing curvature will modify the positions of the neighboring points. But directly solving the above equations requires to deal with non-linear equations. To avoid this and keep the implementation algorithms simple for real-time geometric design, the solution can be approximated as follows. Note that any curvature modification reflects the distance between the two neighboring points

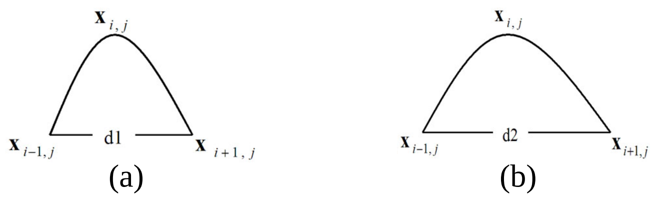


Figure 4.10: Curvature modification via changing the distance between the neighboring sample points. (a) High curvature due to reducing distance; (b) low curvature due to stretching distance.

along the corresponding parametric direction (refer to Fig. 4.10), so the curvature information can be interactively modified by attempting to move the neighboring points (*e.g.*, $\mathbf{x}_{i-1,j}$ and $\mathbf{x}_{i+1,j}$ for κ_u at $\mathbf{x}_{i,j}$). In general, increasing the distance will reduce the magnitude of the curvature, while decreasing the distance will have an opposite effect on the curvature value. After the system computes the new position of relevant neighbors corresponding to the curvature manipulation, it can incorporate these known values of data points into the system and re-solve the equations to derive the new surface that satisfies a set of curvature constraints simultaneously. Fig. 4.8 (c) shows modifying a PDE surface with curvature constraints.

Users can change the shape of a PDE surface by modifying the curvature at arbitrary points.

Curve Constraints

Although point-based conditions provide designers useful manipulation tools, point editing is less appropriate when users are faced with complicating design requirements. The PDE modeling system provides editing tools that afford the intuitive specification of curve-based constraints. First, users can select a source curve on the PDE surface by picking points on the $u - v$ domain. The curve is allowed to be of arbitrary form because the selected points may have arbitrary values of u and v, giving users more freedom for the effective surface editing. Second, users may interactively specify a cubic B-spline curve as the destination curve which will then be mapped to the selected surface curve. The B-spline curve shares the same number of sample points as that of the source curve. B-spline curves are used because of many of their nice properties. Third, the system will enforce the source curve to be in the same shape as the destination curve. The B-spline destination curve adds a number of new linear equations into (4.4), and the PDE surface will be modified accordingly. Users can freely modify or even re-define a destination curve which leads to different PDE surface geometry. In principle, boundary conditions can be special variants of curve-based constraints. Fig. 4.11 illustrates an example of non-isoparametric curve constraints.

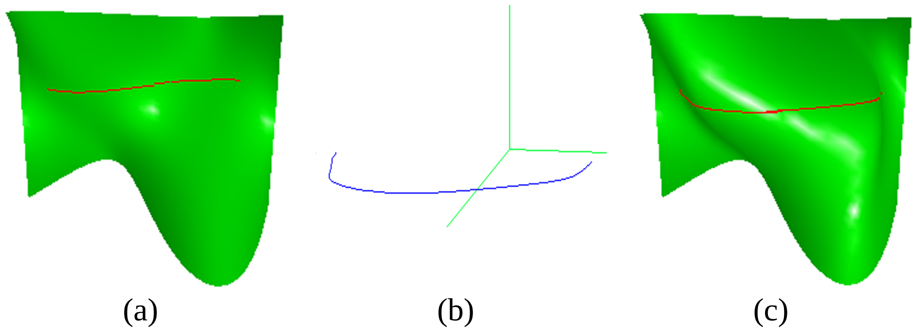


Figure 4.11: Curve editing. (a) The selected source curve shown in red; (b) the destination curve; (c) the deformed PDE surface after curve attachment.

Region Manipulation

Certain surface models exhibit special features in specific regions, hence it is more desirable to develop region-based editing tools toward the ultimate goal of feature-based design. Analogous to the aforementioned curve tool, the system can map a user-specified B-spline destination patch onto a region of interest over the PDE surface. First, users select an area over the PDE surface. Second, users can define a B-spline patch which are sampled to have the same number of grid points as those in the source region. Third, the system maps the specified area to the B-spline patch. Users can interactively deform the B-spline patch or create a new destination patch that imposes area constraints on the PDE geometry (see Fig. 4.12). Because the surface-surface mapping algorithm depends on the structure of sampling grids, the system only considers source regions with rectangular grid in the interest of simplicity.

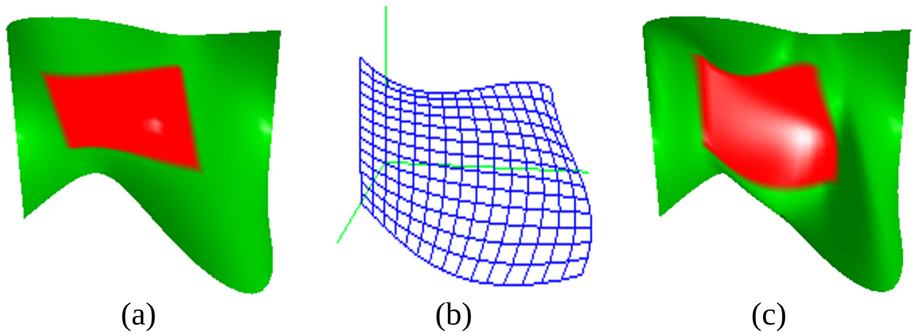


Figure 4.12: Region sculpting. (a) The selected source region shown in red; (b) the B-spline destination patch; (c) the deformed surface after area attachment.

Sculpting and Trimming Localized Regions

Conventional PDE surfaces only support global manipulation, *i.e.*, any local modification results in a new surface undergone the global deformation. This deficiency severely restrains users' freedom of arbitrary surface manipulation at any localized region(s). To overcome this difficulty, the system allows designers to freeze any specified area of a PDE surface that they do not want to change. This can be achieved in the PDE modeling system by selecting a region in $u - v$

domain, then any changes outside this region will not affect any data points inside. as shown in Fig. 4.13.

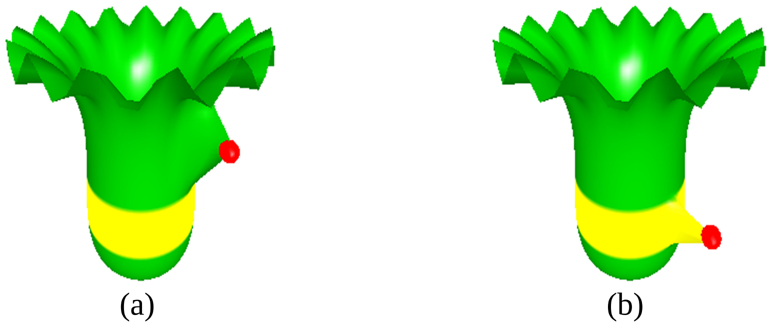


Figure 4.13: Surface manipulation with fixed regions.

In addition, the system offers users functionalities to trim a PDE surface. After the boundary of a selected region is identified, users can remove material from the PDE surface either inside or outside the specified boundary. The system also allows trimming on multiple regions simultaneously, which can be done by specifying several patches of interests on the surface. The trimming operation on PDE surfaces can greatly improve the PDE surface's utility, making it possible to obtain a PDE surface with complex boundaries and arbitrary topological shape (see Fig. 4.14).

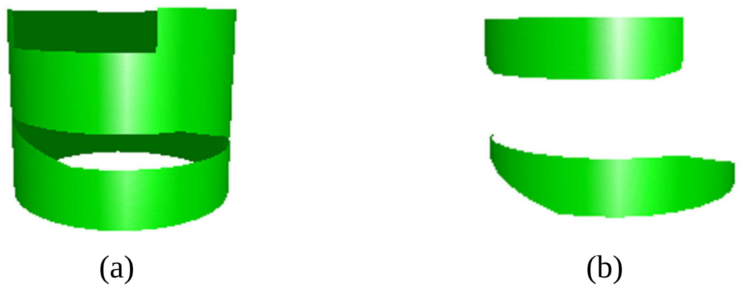


Figure 4.14: Trimming on the selected regions. (a) The surface after the removal of the selected regions; (b) the surface after the removal of all non-selected area.

Direct Displacement Manipulation

The interactive sculpting toolkits are able to manipulate the vector-valued displacements controlled by the PDE formulation. Similar to the constrained PDE surface (4.9), the formulation for constrained PDE surface displacements can be written as:

$$\mathbf{H}_c\mathbf{O} = \mathbf{z}_c. \tag{4.12}$$

The surface deformation is done through the manipulation of displacements while the original surface remains untouched. This enhancement allows the system not only to design surfaces by boundary conditions but also to manipulate existing mesh models directly, which further broadens the applications of PDE surface modeling techniques. Fig. 4.15 shows examples of local point editing on the PDE displacements. Fig. 4.16 has examples for curve manipulation and region sculpting of displacement models.

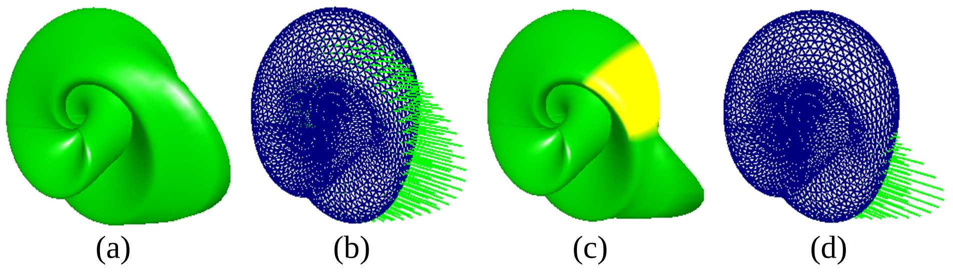


Figure 4.15: Examples for displacement PDE surface sculpting. (a) Changing a point's location on a PDE displacement model; (b) the vector-valued displacement map on the original surface is shown in lines; (c) point sculpting outside the fixed region (yellow) of displacements; (d) the original surface and the displacement vectors of (c).

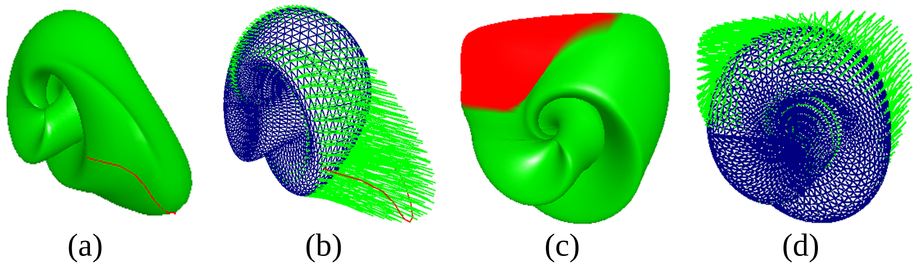


Figure 4.16: Examples for curve and region editing on PDE surface displacements. (a) A PDE surface sculpted using curve editing on displacements; (b) the corresponding vector-valued displacement map on the underlying surface; (c) region manipulation of PDE displacements; (d) the original surface and the displacement vector map of (c).

B-spline Approximation

To facilitate the data exchange capability of PDE surfaces with standard spline-based systems, the system computes B-spline finite elements to approximate PDE surfaces throughout the manipulation process. A B-spline surface over u, and v domain can be defined as (3.3) by a set of control points $\mathbf{P}_{i,j}(1 \leq i \leq k, 1 \leq j \leq l)$. Oftentimes the number of control points is less than the number of sample points on the PDE surface, therefore the B-spline approximation results in a family of over-constrained linear equations whose unknowns are fewer than the number of equations. For example, given the $m \times n$ sample points on the PDE surface, the approximation using $k \times l$ control points leads to

$$\mathbf{BP} = \mathbf{X}, \tag{4.13}$$

where there are $m \times n$ linear equations with $k \times l$ unknowns. Assuming fixed parameterization of data points in B-spline approximation, the matrix $\mathbf{B}$ is a discretization of basis functions, $\mathbf{P}$ is the collection of control points, and $\mathbf{X}$ represents the collection of sample PDE surface points. This over-constrained system can be solved by multiplying $\mathbf{B}^\top$ on both sides of (4.13). Consequently, a B-spline surface that approximates the PDE surface in a least-square sense (4.14) is obtained.

$$\mathbf{P} = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{X}. \tag{4.14}$$

Fig. 4.17 shows B-spline approximations examples.

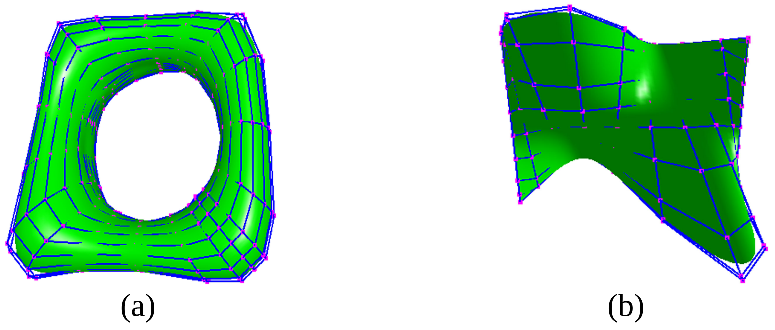


Figure 4.17: B-spline approximations for PDE surfaces. The blue lines with pink points form the B-spline control mesh.

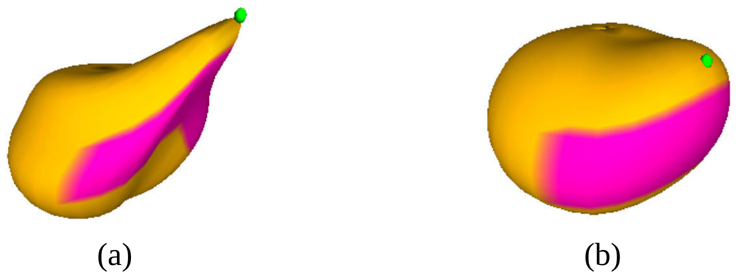


Figure 4.18: The PDE surface sculpting with physical constraints. (a) Directly changing a point of a mass-spring model; (b) point editing in B-spline approximation for the mass-spring model, and the color mapping shows different material properties (refer to Table 9.4).

Meanwhile, B-spline finite elements can also be used to approximate the dynamic model of PDE surfaces at each time step. This allows users to interactively manipulate the B-spline solution of PDE surfaces with forces in real-time. Because the B-spline control mesh is obtained using least-square fitting, the additional constraints of the approximated PDE surface are treated as soft constraints that results in a smoother solution for the PDE surface than the one obtained under hard constraints. Fig. 4.18 (b) shows a snapshot of modifying a dynamic B-spline approximation model of a PDE surface.

Chapter 5

PDE-based Arbitrary Mesh Modeling

The previous PDE modeling techniques for surface models usually focus on parametric surfaces, which extremely limits their applications in geometric modeling where large number of geometric objects are represented by polygonal meshes. To develop a general geometric PDE modeling framework, this dissertation incorporates a couple of functionalities on arbitrary polygonal meshes to further extend the coverage of PDE modeling techniques. The PDE-based modeling system offers shape modeling of arbitrary topology including using elliptic PDEs for arbitrary shape sculpting, diffusion-based equations for medial axis or skeleton extraction from arbitrary meshes, and diffusion-based front propagation models for shape manipulation and recovery based on skeletons. This work extends the geometric coverage of PDE techniques to arbitrary objects bounded by polygonal meshes. And the diffusion-based medial axis extraction and shape manipulation provides modeling operations for complex objects which are lack from previous work of PDE methods. Part of this work has been accepted by Solid Modeling

2004 [49] and Another paper based on this work is in preparation for journal submission [50].

5.1 Introduction and Motivation

There are several popular techniques to define the surface geometry in computer graphics and geometric modeling. Besides the implicit surfaces and the parametric definitions (spline-patches and previous PDE models) over regular domain, another often used shape representation is the explicit polygonal mesh model. In particular, polygonal meshes can be considered the most versatile representation for general free-form surface geometry. Due to the simplicity and robustness of algorithms operating on triangle meshes, arbitrarily complex objects can be approximated using such representation with sufficient resolution.

Subdivision methods are among the most popular modeling techniques for such models. It came from the idea of ”cutting corner” by working directly on the control mesh/lattice to generate smooth objects. It avoids the difficulties of establishing the correspondence of parametric domain and the objects coordinates. Subdivision techniques have simple underlying principles and easy-to-understand schemes to model surfaces/solids of arbitrary topology. However, conventional subdivision rules can only handle simple geometric objects. To model complicated shapes and sharp edges, corners, as well as other small details on the subdivision objects, special subdivision schemes with additional requirements have to be derived, which increases modeling difficulties for general applications.

Since PDE techniques are previously used for surface fairing of arbitrary meshes [38, 79, 137], they can also be considered for interactive manipulation of surface meshes of arbitrary topology with the integration of displacement maps. With this functionality, the PDE modeling system provides direct manipulations for surfaces of arbitrary topology.

However, when modeling complex shapes with a large number of surface vertices, applying manipulations directly on the object will slow down the time performance. In such case, a type of representation that can preserve the complex features and maintain simple structure for manipulation is more desirable. Medial axis models can be an ideal candidate to satisfy such requirements with ease.

The medial axis (or skeleton) provides a compact representation while preserving the object's genus and retaining sufficient local information to reconstruct (a close approximation to) the original shape. This type of representations are of significant interests for a number of applications in biomedicine, including object representation, shape understanding, registration, and segmentation. Such descriptions are equally popular and powerful in geometric shape modeling, model reconstruction, shape analysis, animation, object deformation and manipulation in computer-aided design and medical image processing, etc. Because of its popularity, there are various developed algorithms using different techniques for medial axis extraction in both 2D and 3D. However, the stable numerical computation of medial axis remains a challenging problem.

PDE techniques such as level set methods and Hamiltonian system have been applied for medial axis extraction in recent years [21, 75, 131] because of their modeling advantages. However, the existing work of PDE-based medial axis extraction techniques mainly focus on 2D images or volumetric data defined on discretized grids. They are often associated with Euclidean distance transformation to compute the distance field on 3D lattices. Therefore, the space complexity will increase dramatically for finer resolution. And the computation of distance field for complex models will be much more time-consuming. Using the PDE approach to detect medial axis or skeleton directly from arbitrary 3D meshes and/or B-reps is still under-explored in general. In addition, because polygonal meshes are one of the most dominant representations for geometric models and widely used in modeling and animation, the medial axis extraction to facilitate shape analysis and manipulation for such models will be strongly desirable.

This dissertation presents a PDE technique to extract medial axis (or skeleton) for arbitrary 3D objects bounded by polygonal meshes. It formulates a diffusion-based equation with differential properties of the boundary surface to approximate a simplified medial axis of the object. The diffusion-based equation is solved numerically along the time axis, therefore users can obtain visual feedback during the medial axis extraction process. Note that, it is straightforward to further generalize this method to handle other boundary representations. Users can define their own medial axis for an object by selecting desired boundary points of the object to be skeletal points on the medial axis. It provides users more degrees of freedom for shape skeletonization and further manipulation. The proposed algorithm also allows users to define local regions for skeletonization for part of the object interactively during the process, which will speed up the medial axis extraction for complex models. After the medial axis is extracted, shape manipulation of original dataset can be performed by sculpting the skeleton and using the diffusion-based front propagation techniques along with distance information between the skeleton and its boundary surface to reconstruct the deformed shape. Fig. 5.1 demonstrates some examples of extracted medial axes from several objects.

The diffusion-based medial axis extraction, skeleton-based shape deformation, and the direct manipulation using elliptic PDEs provide modeling advantages of PDE techniques for shape modeling of arbitrary polygonal meshes. With these functionalities, the PDE techniques can be used for more general surface representations and offer possible interactions with other surface modeling methods including spline-based models, subdivision techniques, etc.

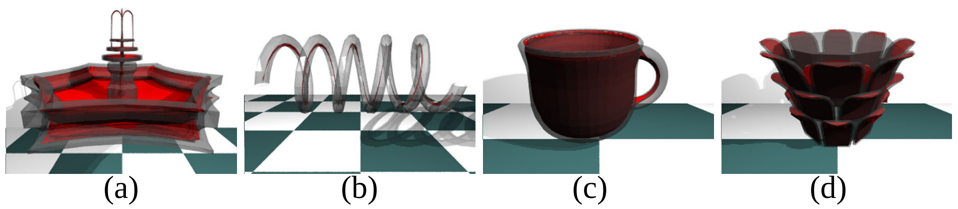


Figure 5.1: Medial axis extraction using the proposed PDE technique. The medial axes are shown in red with transparent datasets surrounding them.

5.2 Direct Manipulation of Arbitrary Mesh Objects

2D elliptic PDEs are usually used to define parametric surfaces over regular (rectangular) parametric domain. The parametric shape representations have difficulties to model arbitrary topological objects. Arbitrary polygonal meshes, on the other hand, can represent arbitrary objects with more flexible topology. Therefore, applying PDE techniques on arbitrary polygonal meshes will greatly expand the PDE application coverages and expose the modeling advantages of PDE methods to more general surface representations. However, different from regular surfaces with underlying rectangular parametric domain, an arbitrary mesh usually have arbitrary vertex connectivities. Therefore, it's a challenging task to discretize such a mesh into homogeneous grids and apply previous finite-difference approximations of partial derivatives, which implies that elliptic parametric PDEs cannot be used to govern arbitrary polygonal surfaces directly. To overcome this, a type of difference operator called *umbrella operator* to approximate the Laplacian operator is considered. It's commonly used in fair surface modeling for 2D meshes[79, 137].

5.2.1 Umbrella Operator

Umbrella operator assumes the mesh has underlying regular parametrization where edge length and angle between neighbor vertices are constant. Then the parametrization of (u_j, v_j) in the $u - v$ parametric domain can be represented by

$$(u_j, v_j) = (cos\frac{2\pi \cdot j}{n}, sin\frac{2\pi \cdot j}{n}).$$

The Laplacian operator can be approximated by the discretized umbrella operator:

$$\nabla^2 \mathbf{p}_i = \frac{1}{n} \sum_{j \in N_1(i)} \mathbf{p}_j - \mathbf{p}_i. \tag{5.1}$$

However, regular parametrization is only suitable for ideal situations, while in most occasions, the regular parametrization cannot give a satisfying result. The umbrella operator can be improved by adding weights based on the connectivity of the mesh which permits vertices drifting in parametric space and leads to non-uniform mesh parametrization. One way is to allow edge lengths between vertices not to be constant. The discretized Laplacian can then be approximated by

$$\nabla^2 \mathbf{p}_i = \frac{2}{E} \sum_{j \in N_1(i)} \frac{\mathbf{p}_j - \mathbf{p}_i}{e_{i,j}}, \tag{5.2}$$

where $E = \sum_{j \in N_1(i)} e_{i,j}$ and $e_{i,j}$ is the edge length between $\mathbf{p}_i$ and $\mathbf{p}_j$. The angles between edges in the 1-neighborhood of a vertex on the mesh can also be consider as weights in the umbrella operator. Fig. 5.2 shows the illustration of umbrella operators.

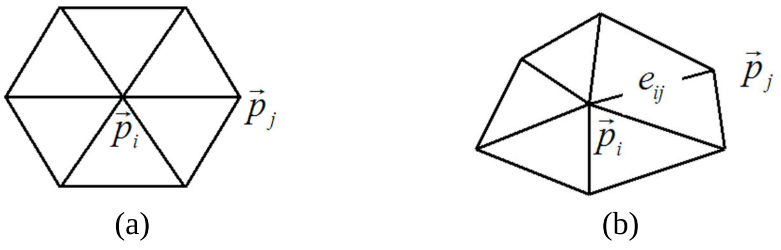


Figure 5.2: Umbrella operators. (a) Regular umbrella operator; (b) improved umbrella operator.

5.2.2 PDE Formulation for Arbitrary Meshes

The fourth-order differential operator in (4.1) can be approximated using umbrella operators by setting the blending coefficient $a = 1$, which is called Biharmonic operator $\nabla^4 = (\nabla^2)^2$. Second-order elliptic PDEs that use Laplacian operator can also be used for the arbitrary mesh model. Because the datasets of arbitrary meshes are taken as input from existing models, when calculating the partial derivatives of the datasets, the right-hand side of the governing PDEs may not be zero. Therefore, the fourth-order and second-order PDEs used to govern the behavior of the input mesh are formulated as follows:

$$(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2})^2 \mathbf{p} = \nabla^2(\nabla^2 \mathbf{p}) = \mathbf{F}, \tag{5.3}$$

and

$$(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2}) \mathbf{p} = \nabla^2 \mathbf{p} = \mathbf{F}, \tag{5.4}$$

where (5.4) has the form of Poisson equation.

Before the shape manipulation, an initialization process is performed to calculate the partial derivatives in (5.3) and (5.4) and values for the right-hand side, **F**, are obtained. For each vertex $\mathbf{p}_i$ on surface $\mathbf{p}$, $\nabla^2(\nabla^2 \mathbf{p}_i)$ can be approximated by

$$\begin{aligned} \nabla^2(\nabla^2 \mathbf{p}_i) &= \frac{1}{n} \sum_{j \in N_1(i)} \nabla^2 \mathbf{p}_j - \nabla^2 \mathbf{p}_i, && (5.5) \\ \nabla^2 \mathbf{p}_i &= \frac{1}{n} \sum_{j \in N_1(i)} \mathbf{p}_j - \mathbf{p}_i, && (5.6) \end{aligned}$$

where $N_1(i)$ is the collection of vertices in 1-neighborhood of $\mathbf{p}_i$ and n is the valence of $\mathbf{p}_i$, *i.e.*, number of vertices in $N_1(i)$.

5.2.3 Direct Manipulation of Arbitrary Meshes based on PDEs

To sculpt a polygonal mesh object to desired shape, users can pick a vertex on the discrete mesh and move it to another location, then the shape will deform

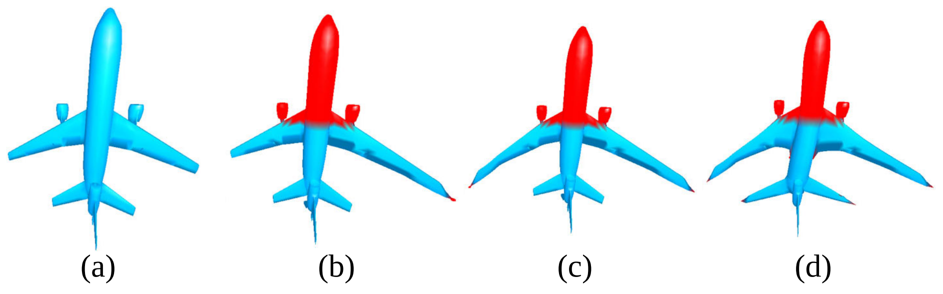


Figure 5.3: An example of PDE model of arbitrary meshes. (a) Original dataset; (b), (c), and (d) are deformed sequences from (a), where only the blue parts are allowed to deform.

according to the modification. However, because the PDE is formulated based on the initial shape, the vertices on the polygonal mesh are constrained to each other, which indicates that without additional constraints on the mesh, the entire shape will not deform with the location change of one point but only transform to the new location according to the change. Therefore, users need to specify additional constraints such as fixing selected parts of the object to allow the shape to deform. Fig. 5.3 shows examples for this kind of deformation. Alternatively, the polygonal surface can be modeled as original surfaces with displacement maps using the techniques introduced in Chapter 4. Then surface deformation can be obtained by sculpting surface displacements without touching the underlying original surface.

5.3 Diffusion-based Medial Axis Extraction

Shape skeletons (*i.e.*, medial axes) are popularly used in many visual computing applications, such as pattern recognition, object segmentation, registration, and animation. Meanwhile, certain PDE techniques such as level set methods and Hamilton-Jacobi equation have been used to detect medial axes of 2D images and volumetric data with ease. However, direct extraction methods based on PDEs to detect skeletons of 3D solid objects bounded by arbitrary meshes are still under-explored. This dissertation expands the use of diffusion equations to approximate medial axes of arbitrary 3D objects represented by polygonal meshes based on their differential properties. It offers an alternative but natural way of medial axis extraction for commonly used 3D polygonal models. By solving the initial value PDE along time axis, the system can not only quickly extract diffusion-based medial axes of input meshes, but also allow users to visualize the extraction process at each time step. In addition, this diffusion-based model provides users a set of manipulation toolkits to sculpt extracted medial axes, then uses diffusion-based techniques to recover corresponding deformed shapes according to the original input datasets. This skeleton-based shape manipulation offers a fast and easy way for animation and deformation of complicated geometric objects.

5.3.1 PDE Formulation of Medial Axis Extraction for Arbitrary Meshes

This dissertation employs a diffusion-based PDE that allows 3D objects to propagate inward their boundaries and approximate simplified skeletons with user interactions, which can provide users instant feedback and interactive control during the extraction process. The distance information from skeletal points to the boundaries is recorded for reconstruction and deformation purposes. When manipulating the skeleton, the original model can be deformed accordingly. Other immediate applications include model simplification, skeleton-driven parameterization, and animation control of complex, articulated characters.

Diffusion-based Equation

Medial axis extraction can be simulated using the idea of grassfire flow. The grassfire flow on a surface $\mathbf{S}$ in 3D space is governed by

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{N}, \tag{5.7}$$

which allows the fire front propagating at unit speed along the inward surface normal $\mathbf{N}$.

The simplest way to simulate (5.7) for medial axis extraction of a polygonal mesh is to let vertices on the boundary surface travel along their surface normal inward (*i.e.*, shrinking the boundary) at each time step, and where the points meet with each other forms the skeleton. However, the time step for this simulation process needs to be very small to guarantee a close approximation of medial axes. Furthermore, because the surface normal can only be approximates at discrete sample points on the boundary polygonal mesh where the regular parametrization is not applicable, mesh sampling qualities will directly affect simulation results and the structure of obtained medial axes will be complicated for complex datasets. In general, it's difficult to achieve satisfactory results using direct simulation of (5.7).

On the other hand, diffusion equations are frequently used for denoising in image processing. They can also provide smooth results for geometric surface fairing [38]. Because of their smoothing properties, this dissertation uses the diffusion process for medial axis extraction from polygonal meshes, which will provide simplified approximations and remove noisy branches on medial axes for easy storage and manipulation. Since the main purpose of medial axis extraction in this dissertation is to offer users a compact geometric representation for shape manipulation and deformation, such approximations can provide satisfactory results.

The diffusion-based equation for medial axis approximation can be formulated as:

$$\frac{\partial \mathbf{S}}{\partial t} = \mathbf{D}(\mathbf{N}, \kappa)\nabla^2 \mathbf{S}, \tag{5.8}$$

where $\mathbf{S} = \mathbf{S}(\mathbf{p}, t)$ is the propagating boundary surface, $\mathbf{p} = (x, y, z)$ is the coordinate vector, $t \geq 0$ is time variable, $\nabla^2\mathbf{S}$ is the Laplacian of the surface, and $\mathbf{D}$ is the diffusion coefficient function related to the surface normal $\mathbf{N}$ and curvature κ. The normal $\mathbf{N}$ provides directions for boundary propagation during the medial axis extraction process. The curvature κ is used as a threshold to detect skeletal points on the medial axis. Because the Laplacian will smooth the boundary surfaces and eliminate the sharp features during the propagation, the use of curvature as a threshold can allow the propagation process to detect sharp features of an object and preserve such properties on its simplified medial axis.

(5.8) is formulated to guide the boundary surface propagation. By solving (5.8), the object's surface will moving inward from the original boundary guided by its normal, and the Laplacian will smooth the surface to avoid noisy branches during the propagating process. The curvature acts as a threshold to preserve feature points of the object on the approximated medial axis. Therefore, after all the points on the propagating surfaces collide with others, which means they reside on a thin set, a compact structure without interior points inside can be obtained. It can be viewed as an approximation for the *real* medial axis because it's a thin set inside the object and preserve features of the original dataset. Such a compact and simple representation can provide satisfactory results for skeleton-based shape manipulation. Note that, the shape reconstruction from skeletons is a reverse process of medial axis extraction by applying the normal outward to original boundaries. The diffusion equation is suitable for continuous geometric objects including surfaces and solids. Although it is solved on discrete polygonal boundary surfaces numerically in this dissertation, the equation can be readily applied to other type of solid representations for medial axis extraction.

Numerical Simulations

Diffusion equations can be easily solved through numerical techniques. To simplify the process and provide a fast algorithm for medial axis extraction, the PDE-based modeling system employs the finite-difference discretization associated with umbrella operators for iterative computations of the evolving surface. The diffusion-based equation (5.8) can be discretized as follows:

$$\frac{\mathbf{p}_i^{n+1} - \mathbf{p}_i^n}{\Delta t} = \mathbf{D}(\mathbf{N}_i, \kappa_i)(\frac{1}{n} \sum_{j \in N_1(i)} \mathbf{p}_j^n - \mathbf{p}_i^n), \tag{5.9}$$

The surface normal at a sample vertex on an arbitrary mesh object is also calculated using numerical approximation techniques. The simplest way is first calculating the normals of surface patches around the target point, then averaging the surface patch normals to approximate the normal at the point. However, this method only provides a rough approximation of surface normals at sample points. There is another way to approximate the surface normal at a point [165] to provide more satisfying normal approximations. The normal $\mathbf{N}_i$ at vertex $\mathbf{p}_i$ can be computed using approximate tangent vectors of the surface at $\mathbf{p}_i$ that can be computed as:

$$\mathbf{t}_1 = \sum_{j=0}^{n-1} cos \frac{2\pi j}{n} \mathbf{p}_j, \mathbf{t}_2 = \sum_{j=0}^{n-1} sin \frac{2\pi j}{n} \mathbf{p}_j,$$

where n is the valence of $\mathbf{p}_i$, and $\mathbf{p}_j$ are its neighbors.

The approximate surface normal $\mathbf{N}_i$ at $\mathbf{p}_i$ can be computed as

$$\mathbf{N}_i = \mathbf{t}_1 \times \mathbf{t}_2. \tag{5.10}$$

Since the diffusion process is also influenced by the surface curvature, it's necessary to evaluate the curvature values at the boundary surface. This dissertation considers the contribution of Gaussian curvature for medial axis extraction. Note that,the curvature is used as a threshold to define skeletal points on the medial axis to preserve shape features, therefore other types of curvature instead of Gaussian curvature can also be employed for this purpose. A local approximation scheme is used to compute Gaussian curvature of sample points on the boundary surface based on Gauss-Bonnet theorem. The Gaussian curvature κ at a surface vertex is related to angles and faces connected to the point on the surface [149]:

$$\kappa = \frac{\phi}{A},$$

where ϕ is the angular defect at the point which is defined as 2π - *sum of the interior angles of faces meeting at the point* and A is the area associated to the point that is equal to $\frac{1}{3}$ *of the sum of the areas of the triangles meeting at the point*. Therefore, the Gaussian curvature κ_i at point $\mathbf{p}_i$ can be computed as follows:

$$\kappa_i = \frac{2\pi - \sum_{j=0}^{n-1} \phi_j}{\frac{1}{3}\sum_{j=0}^{n-1} A_j}, \tag{5.11}$$

where ϕ_j is the angle of the jth face connected to $\mathbf{p}_i$ and A_j is the corresponding triangle's area. (5.11) is for inner points on a mesh and suitable for any points in a closed surface. As for open surfaces, the approximation for Gaussian curvature of a boundary point can be evaluated using the following scheme:

$$\kappa_i = \frac{\pi - \sum_{j=0}^{n-1} \phi_j}{\frac{1}{3}\sum_{j=0}^{n-1} A_j}, \tag{5.12}$$

An illustration of Gaussian curvature approximation for a mesh point is shown in Fig. 5.4.

With numerical discretizations and approximations for Laplacian operator, surface normal, and curvature, (5.9) can be easily solved by iterative method along

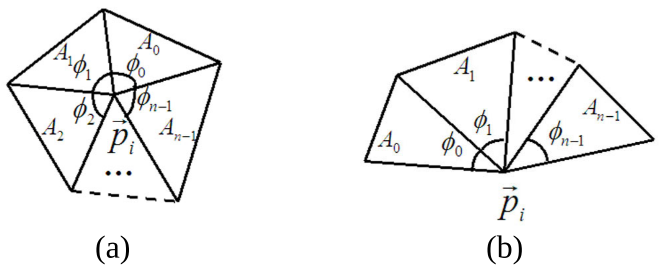


Figure 5.4: The evaluation of Gaussian curvature for a mesh point. (a) Gaussian curvature for an internal vertex; (b) Gaussian curvature for a boundary vertex.

the time axis. The diffusion equation will evolve along the time axis according to the surface curvature and normal, and the medial axis resides on the locations where different parts of the propagating surfaces meet. Because of the discretized property, users can freeze certain points on the mesh to let them stay at their current positions during the process to obtain different skeletons. Furthermore, Users can select a region to extract the medial axis inside the region for localized results.

Because the distance information between skeletons and the corresponding original boundary surfaces is already saved, the original object can be recovered easily by propagating the surface from the extracted medial axis along the normal outward. Moreover, after skeleton manipulations, the deformed shape can be reconstructed through diffusion equation to obtain a smooth result.

5.3.2 PDE-based Skeletonization and Shape Manipulation

Diffusion-based Medial Axis Extraction: An Algorithmic Outline

This dissertation uses finite-difference techniques to approximate solutions for the time-dependent diffusion-based equation numerically in order to provide users progressive results for medial axis approximation and shape reconstruction.

Starting with the original mesh, the PDE-based modeling system extracts the

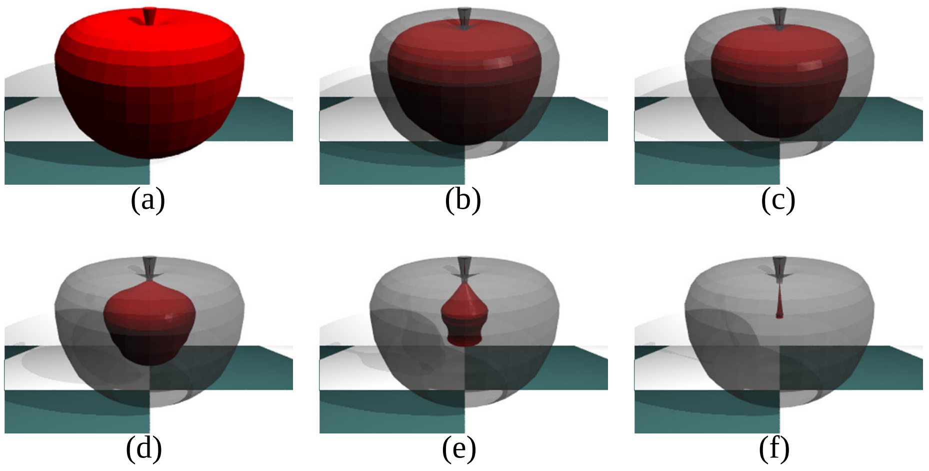


Figure 5.5: An example of PDE-based medial axis extraction for arbitrary meshes. (a) Original dataset; (b), (c), (d) and (e) are shrinking objects during medial axis extraction at different time step by performing (5.9); (f) is the final approximate skeleton.

medial axis according to differential properties of the boundary mesh, and allow the mesh to shrink to an approximate medial axis. The extraction algorithm consists of following operations:

- Initialization: at the initialization stage, the system approximates the surface normal for the boundary surface using (5.10) and other differential properties such as Gaussian curvature based on (5.11) and Laplacian using umbrella operators.

- Skeletonization: during the skeletonization process, at each time step, the system first computes the evolving surface based on (5.9). Then collision detection is performed on the resulting surface. If a surface point collides with any other point, edge, or face, it is considered as residing on the skeleton. In such case, it is marked as a skeletal point with its position fixed and the distance information from original surface point to this skeletal point is recorded. After all the points are checked for collision detection, surface optimization is applied to delete redundant points and faces with too small areas. This process is repeated until all points on the propagating surface are marked as skeletal points.

- User Interaction: in addition, during the extraction process, users can interactively select any vertices on the propagating surface to be skeletal points, thus they can define the user-controlled skeleton based on their own criteria. Users are also allowed to define local regions for local medial axis extraction.

The medial axis extraction using the proposed diffusion-based technique is a progressive process along time, which offers users visual feedback during the extraction. Fig. 5.5 shows an example of progressively extracting medial axis for an object.

After this skeletonization process, the system obtains a simplified skeleton approximating the object's medial axis associated with distance information between the skeleton and the original boundary surface. With such information, users can manipulate the object through its skeleton with ease.

Local Region Skeletonization and User Interaction

To explore local features, the system offer users local skeletonization by extracting the medial axis of a selected part of an object. This can be done by specifying a region in the 3D working space and the system will only extract a skeleton for part of the object residing in the region. By allowing medial axis extraction in local regions, it will reduce the time complexity of shape skeletonization for complex models and enable direct user control. Refer to Fig. 5.6 for an example.

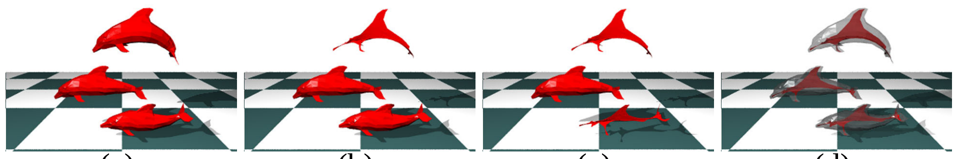


Figure 5.6: An example of PDE-based medial axis extraction for selected parts from arbitrary meshes. (a) Original dataset; (b) and (c) are two examples of extracting skeletons for part of the dataset; and (d) is the skeleton for the entire dataset.

Because the system provides a simplified approximation of medial axis for an object, the result may not satisfy users expectation sometimes. For example, there are certain points on the object that users want to be on the medial axis, but the system doesn't mark them as skeletal points during medial axis extraction process. Therefore, users are allowed to select desired points on the boundary surface to be skeletal points for any user defined skeleton during the extraction process, which can provide more degrees of freedom for later skeleton-based shape manipulation (Fig. 5.7). Since manipulations on different shape of skeletons can result in different shape deformations, these user interactions provide more flexibility/freedom and control for skeleton-based shape sculpting. Furthermore, because the diffusion-based equation is solved on polygonal meshes, the number of points on the meshes will extremely affect the performance of the medial axis extraction process. It's time consuming to extract medial axes for complex models. Therefore, local medial axis extraction in selected regions will be useful for such cases. It's also possible to integrate parallel techniques with this method for shape skeletonization of large datasets.

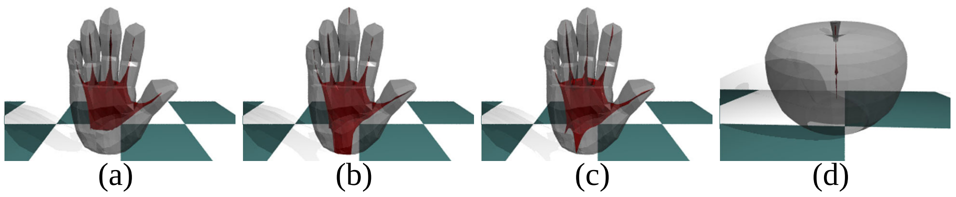


Figure 5.7: Examples of PDE-based medial axis extraction with user-defined skeletal points. (a), (b) and (c) are different skeletons obtained after fixing different surface points as skeletal points; (d) is the skeleton of Fig. 5.5 by fixing a point at the bottom of the dataset.

Skeleton-based Shape Sculpting

One of the advantages of medial axes is that they provide a much more compact and natural representation for geometric objects. Therefore, the shape deformation/manipulation and other processes based on medial axes alleviate the burden of tedious and less insightful operations for deforming and animating complex objects, as well as other shape queries and interrogations. The PDE-based modeling system provides users sculpting tools to manipulate the medial axis, then propagates the deformation to the original dataset according to the distance information. However, the deformed result may not be satisfactory if the system just simply reconstructs the object from the medial axis according to its distance to the original dataset. Therefore, the diffusion-based equation with normal outward to the original boundaries is employed to reconstruct the modified dataset. Fig. 5.8 and Fig. 5.9 have two examples of shape manipulation based on skeleton modification and shape reconstruction using diffusion propagation. Fig. 5.10 shows a deformation sequence of an object through skeleton manipulations. It may be noted that, from this point of view, this technique also serves as an aid for shape parameterization and can be potentially improved for a powerful shape analysis tool (beyond shape sculpting and synthesis).

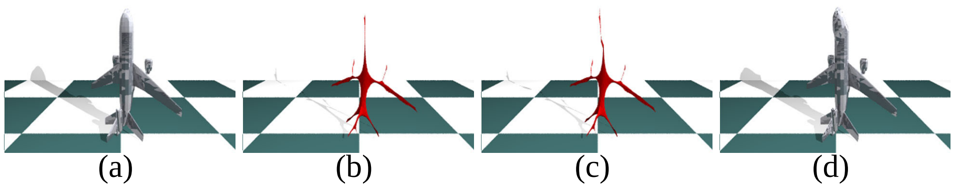


Figure 5.8: An example of skeleton-based shape sculpting. (a) Original dataset; (b) is the skeleton; (c) is the sculpted skeleton; (d) is the corresponding deformed dataset recovered from (c).

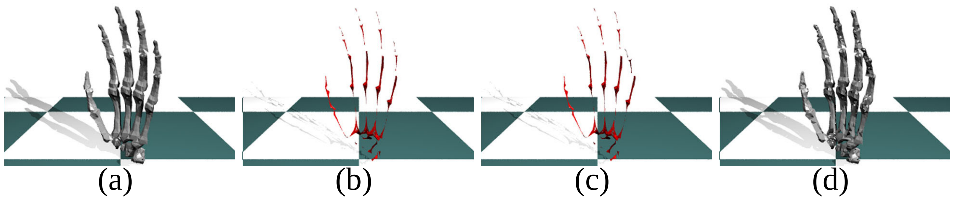


Figure 5.9: Another example of skeleton-based shape sculpting.

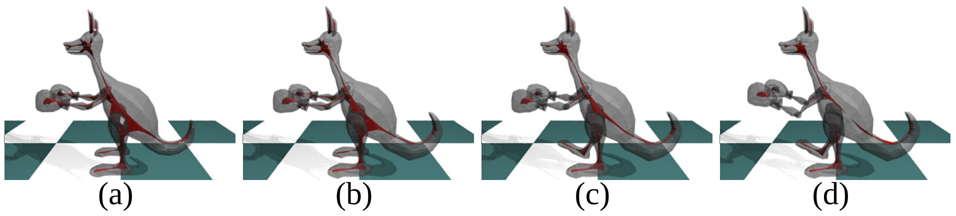


Figure 5.10: A sequence of deformed shapes through skeleton-based shape sculpting.

Curvature Manipulation

Gaussian curvature of polygonal surfaces is employed in the diffusion equation for shape skeletonization. It works as the threshold for medial axis extraction

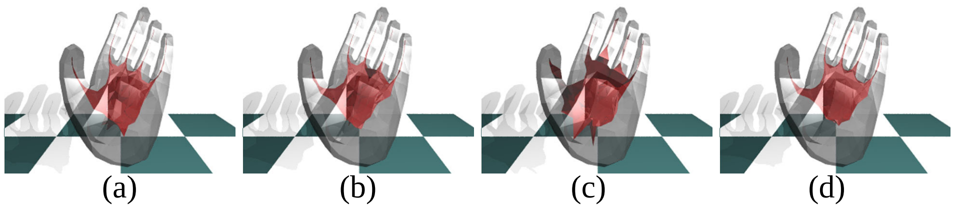


Figure 5.11: Examples of skeleton extraction with different value of curvature thresholds.

to decide which surface points will be skeletal points. Thus, different value of the threshold for Gaussian curvature of the boundary polygonal mesh will result in different shape of skeletons. By allowing users to define the threshold themselves, they can obtain medial axes for an object according to their own criteria. Fig. 5.11 shows examples for several medial axes extracted from an object with different Gaussian curvature thresholds.

Chapter 6

Implicit Elliptic PDE Model

To maximize the modeling capabilities of PDE techniques and implicit functions in geometric and visual computing areas, this dissertation proposes a PDE-based modeling paradigm which integrates PDE techniques with implicit functions into one single framework for interactive shape design and manipulation of PDE-based volumetric implicit models. This dissertation presents an implicit modeling module governed by elliptic PDEs of scalar intensity fields which can reconstruct implicit objects and the embedding implicit 3D working space as solutions of the PDEs by specifying a set of curve outlines or scattered data points of certain intensity values as *general* boundary constraints with the assistance of variational interpolating approaches and distance field approximations. Because the curves and datasets are not required to be closed, open surfaces can be modeled within the PDE modeling system. Moreover, it offers a set of sculpting toolkits to manipulate implicit objects, such as interactively modifying the geometric shape, intensity value, and gradient direction of selected sketch curves, directly changing intensity values, gradient vectors, and curvature information of selected regions in the working space, as well as deforming iso-contours at specified intensity values of objects. Because the working space is governed by PDEs, any missing information inside the space can be recovered by the PDEs according to the given constraints. It is able to recover damaged datasets using partial information, smooth the intensity distribution of volume data, and smoothly blend objects inside the working space. In general, it allows intensity manipulation anywhere in the implicit working space to model implicit objects at any iso-value either directly or indirectly, which offers users both local and global control for implicit shape manipulation. This work has been published in Solid Modeling 2003 and accepted by CAD Special Issue of Solid Modeling 2003 [47, 51].

6.1 Introduction and Motivation

The parametric PDE model simplifies the geometric design process by using only boundary conditions to recover the whole interior information and offers high-order continuity as well as energy minimization properties, like traditional parametric approaches. However, it's extremely difficult to model arbitrary shapes of general topology, because parametric PDEs are defined over regular domain.

On the other hand, implicit functions are usually characterized by zero-sets of polynomial-based algebraic equations and other commonly-used scalar equations. They form another powerful category of modeling techniques to represent surfaces/solids in graphics, geometric design, and visualization, and offer certain advantages comparing with parametric methods such as point classification and unbounded geometry. However, despite the modeling advantages of implicit functions, implicit modeling operations in previous work are usually less intuitive for common users. In general, the modeling potential of implicit functions has not been fully explored yet and there are still difficulties to design, reconstruct, and sculpt implicit models directly and intuitively.

To further make use of the attractive features of implicit functions for interactive design and sculpting, and expand the modeling coverage of PDE techniques, this dissertation applies PDE techniques to formulate implicit functions which may potentially bridge the gap between implicit functions and other geometric and physics-based modeling techniques. This unified approach provides users more powerful manipulation toolkits for volumetric implicit shape modeling. Instead of time evolution PDEs used in the level set method, static elliptic PDEs for boundary value problems are employed with the assistance of the variational energy minimization and distance field approximation. In particular, this dissertation introduces a novel technique which defines the volumetric implicit objects as the approximate solution of elliptic PDEs of scalar intensity fields under *generalized* boundary constraints. This method can be used for geometric shape design, object reconstruction, and shape blending. Implicit PDE objects can be manipulated by modifying the initial constraints or changing intensity values inside the volumetric space, as well as sculpting iso-surfaces at specified values. This implicit PDE approach has modeling advantages of both parametric PDE techniques and implicit functions. Using implicit PDEs, not only objects, but also the entire working space can be recovered from given information. The PDE-based modeling system doesn't require the constraints to be closed datasets, which provides modeling potentials for open surfaces. In addition, because implicit PDEs are formulated on a scalar intensity field and define objects by collecting points of certain iso-values, they are capable of designing arbitrary topological shapes and recovering the full information from partial input, which will reduce the burden of specifying the large quantity of constraints for complete datasets. This offers users a natural way to design objects easily with general non-isoparametric arbitrary curve outlines, reconstruct objects from scattered data points, blend shapes in the working space, and recover damaged datasets. Implicit PDE objects can be visualized by either using the Marching Cube [90] method which calculates the triangulated iso-surface at certain selected intensity value on discretized sampling grids, or through other volume rendering systems such as Pov-Ray and Vol-Vis systems.

6.2 Elliptic PDE Formulation for Implicit Models

The implicit PDE formulation employed in this dissertation is founded upon the parametric PDE solid models [45], which used a fourth-order elliptic PDEs to establish a mapping from 3D parametric domain to 3D physical space and developed a set of sculpting tools over the resulting parametric PDE solids (7.1).

In order to take advantage of the interactive feature associated with parametric PDE modeling techniques, this dissertation uses elliptic PDEs to define scalar intensity fields for modeling implicit objects. Because higher order PDE can provide higher-order continuity for the scalar intensity value distribution, a fourth-order elliptic PDE is used to model the scalar field for smooth results with tangential continuity, especially when dealing with shape blending and damage data recovery in which most of information will be specified as constraints. In particular, the unknown function is formulated as the intensity field function $d(x, y, z)$ defined in the physical space of x, y, and z instead of the vector function $\mathbf{X}(u, v, w)$ in the parametric space of u, v, and w. The corresponding implicit PDE is formulated as follows:

$$(a^2(x,y,z)\frac{\partial^2}{\partial x^2} + b^2(x,y,z)\frac{\partial^2}{\partial y^2} + c^2(x,y,z)\frac{\partial^2}{\partial z^2})^2 d(x,y,z) = 0, \tag{6.1}$$

where x, y, and z are coordinate variables of 3D physical space varying from 0 to 1, respectively, which form a unit cube as the working space. $a(x, y, z)$, $b(x, y, z)$, and $c(x, y, z)$ are arbitrary functions of x, y, z defining material properties of the implicit space, which are initially defined as constants throughout the entire working space.

Because the numerical techniques to solve fourth-order elliptic PDEs used in this dissertation are suitable for other boundary value PDEs, the system also provides solutions for a second-order PDE:

$$(a^2\frac{\partial^2}{\partial x^2} + b^2\frac{\partial^2}{\partial y^2} + c^2\frac{\partial^2}{\partial z^2})d(x, y, z) = 0, \tag{6.2}$$

which is less time-consuming to solve with less continuous intensity distribution and can be used for initial guess of intensity values of objects.

Because $a(x, y, z)$, $b(x, y, z)$, and $c(x, y, z)$ are allowed to vary across $d(x, y, z)$, *i.e.*, different locations in the physical domain may have different smoothing coefficient values, local control on implicit PDE objects can be easily achieved.

To obtain direct and local manipulation on implicit PDE objects, this dissertation solves (6.1) and (6.2) using numerical methods based on finite-difference approximations of PDEs, which require at least six boundary conditions at $x = 0, x = 1, y = 0, y = 1, z = 0, z = 1$ that define intensity values at three boundary surface pairs of the 3D working space in order to derive a unique solution. However, in most applications, there are no such boundary conditions available for modeling implicit objects, especially in the case of using implicit functions for shape reconstruction, where the constraints are usually defined by certain contouring sketch curves or scattered points assigned with specified intensity values inside the 3D working space. In such cases, the intensity distributions on the boundaries are unknown. Thus, such problems cannot be solved by traditional finite-difference methods directly. However, the intensity distribution for this type of problems can be approximated as follows. First, the sytem can find an initial guess of the volumetric working space using certain techniques. Second, the guessed boundary values are used as boundary conditions, and additional constraints are enforced according to the original data. Third, the sytem performs iterative finite-difference techniques to get an approximated solution for the entire working space based on these constraints. After that, direct manipulation and local sculpting inside the working space can be enforced by adding additional constraints to the PDEs. Variational interpolating approaches are good candidates for shape reconstruction from scattered points, such as the RBF method [99, 146] that creates a 3D implicit function to give an approximation interpolating given constraints by minimizing certain energy functional. This dissertation uses the RBF method to compute the initial guess of implicit PDE objects defined by sketch curves. The intensity values on sampling grids can also be calculated using their distance to the constraints, because implicit objects can be defined by distance functions. The algorithm used in this dissertation to compute the distance field is the fast-tagging approach proposed by [161]. Note that, because the goal here is to obtain an initial guess of the working space according to constraints, there are other techniques which can provide satisfactory results.

6.3 Radial Basis Function

Radial Basis Function (RBF) is commonly used for scattered data interpolation, which is to generate a smooth surface that passes through a given set of data points. Scattered data interpolation sometimes can also be addressed using variational analysis where the desired solution is a function, $f(\mathbf{x})$, which minimizes certain energy functionals. In principle, the energy functional measures the quality of interpolation subject to the interpolatory constraints $f(\mathbf{c}_i) = h_i$. It can be solved by a weighted sum of certain radial basis functions $\phi(\mathbf{x})$ centered at the constraint locations. Then the interpolation function can be formulated as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i \phi(\mathbf{x} - \mathbf{c}_i) + P(\mathbf{x}), \tag{6.3}$$

where $\mathbf{c}_i$'s are the coordinate vector of the constraints, w_i's are weights, and $P(\mathbf{x})$ is a polynomial only consisting of the linear and constant portions of f. According to the properties of the appropriate radial basis functions $\phi(\mathbf{x})$ ($\phi(\mathbf{x}) = |\mathbf{x}|^3$ in the system for 3D space), the interpolation function minimizes the thin-plate energy while satisfying the data interpolation requirement. Applying the constraints to (6.3) can provide a linear equation system whose unknowns are the weights and coefficients of the polynomial P. This system can be solved using standard solvers of linear equations.

However, the RBF method requires gradient information of datasets, and time and space complexity of the equation system depends on the number of constraints, so it's not suitable for reconstruction and interactive sculpting of large scattered datasets with arbitrary constraints. Because the goal here is simply making an initial guess for the implicit PDE shape, distance approximation techniques such as the fast-tagging algorithm which computes the signed distance field of the working space according to the constraints can give satisfactory results for such input.

6.4 Numerical Simulation

In order to easily apply additional constraints for direct manipulation of implicit objects, this dissertation resorts to the numerical techniques based on finite-difference approximation and iterative method for linear equations to solve the implicit PDEs with pre-defined boundary values or approximate initial guess from sketch curves/scattered points and arrive at an approximate solution with user-specified error tolerances. Numerical algorithms also facilitate the material modeling of anisotropic distribution. A multi-grid iterative solver is used to improve the system performance. By applying the finite-difference technique, a set of algebraic equations for the fourth-order implicit PDE can be achieved:

$$\mathbf{G}\mathbf{d} = \mathbf{b}. \tag{6.4}$$

Similarly, (6.2) can be rewritten as:

$$\mathbf{G}'\mathbf{d} = \mathbf{b}', \tag{6.5}$$

The implicit PDEs are open along all of x, y, and z directions, so forward/backward instead of central difference approximations shall be utilized for the computation of partial derivatives near the six boundaries. Arbitrary boundary/additional constraints can be easily enforced using the finite-difference method. In the PDE modeling system, after making the initial guess of intensity distributions of the working space, intensity values at those boundaries can be fixed, so that manipulations on implicit objects can be performed using the finite-difference iterative solver. In general, this type of elliptic PDEs allows designers to choose (various) constraints based on diverse design tasks.

6.5 Constrained System

One attractive advantage of PDE modeling techniques is that the interior of objects is controlled by PDEs without the need of extra specification for interior material distribution. More importantly, users can modify an implicit PDE object by enforcing additional hard constraints of desired intensity values anywhere inside the working space without violating previously defined conditions. Such additional hard constraints introduce a set of new equations into the system to replace the corresponding original difference equations. For example, if users want to set the intensity value $d_{(i,j,k)}$ as d_0, the equation $d_{(i,j,k)} = d_0$ will be used to replace the d ifference equation at the point $\{i, j, k\}$, *i.e.*, $G_{(i,j,k)(i,j,k)} = 1$, all other $G_{(i,j,k)(i',j',k')} = 0$, and $b_{(i,j,k)} = d_0$. After replacing all the equations according to the constraints, (6.4) becomes

$$\mathbf{G}_c \mathbf{d} = \mathbf{b}_c, \tag{6.6}$$

where $\mathbf{G}_c$ and $\mathbf{b}_c$ are obtained by replacing $k(k > 0)$ equations in the original system with those derived from additional k constraints. (6.4) and (6.6) can be solved using iterative methods.

The constrained system for the second-order (6.5) has the similar form:

$$\mathbf{G}'_c \mathbf{d} = \mathbf{b}'_c, \tag{6.7}$$

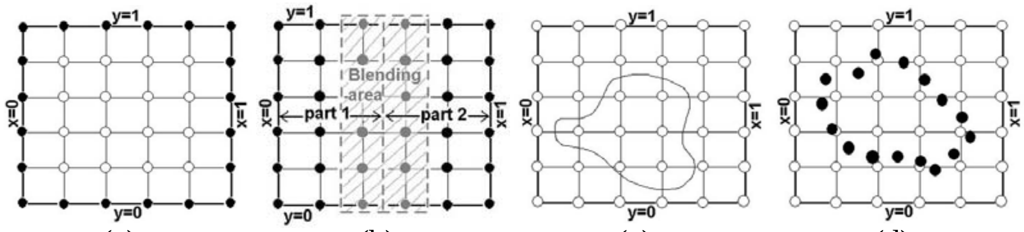


Figure 6.1: 2D illustrations of different types of boundary conditions. (a) Traditional boundary constraints; (b) boundary conditions for shape blending; (c) sketch-curve constraints; (d) scattered-point constraints.

6.6 Boundary Conditions for Different Applications

To construct an implicit PDE object, first users need to outline the rough shape of the object, which can be defined through boundary conditions or special constraints such as curve contours and scattered data points in the working space that the object interpolates. The form of boundary constraints varies for different applications. The implicit PDE technique accepts boundary conditions for applications such as shape blending, object recovery, and shape reconstruction from sketch curves and scattered data points. Fig. 6.1 illustrates different types of boundary conditions in simplified 2D cases.

6.6.1 Shape Design Using Traditional Boundary Constraints

The implicit PDE technique can model geometric shapes by computing the information of the whole working space based on traditional boundary constraints with optional cross-sectional details inside the working space. Such boundary conditions are defined as intensity values sampled at certain resolution from input or use some analytic functions to generate implicit boundary functions $d(0, y, z)$, $d(1, y, z)$, $d(x, 0, z)$, $d(x, 1, z)$, $d(x, y, 0)$, $d(x, y, 1)$ and a collection of cross-sectional scalar intensity functions $d(x_i, y, z)$, $d(x, y_j, z)$, or $d(x, y, z_k)$, where $x_i, y_j, z_k \in (0, 1)$ are constants. These functions are sampled at specified resolution to provide a set of intensity values inside the working space. Using these values as generalized boundary conditions introduces certain number of new equations and the linear equation system has the form of (6.6) or (6.7) which can be solved using above mentioned techniques. Fig. 6.2 shows examples of the fourth-order and second-order PDEs, respectively. Although (4.9) takes more time to solve, it provides higher-order continuity of intensity distributions in the working space in comparison with results from (6.7).

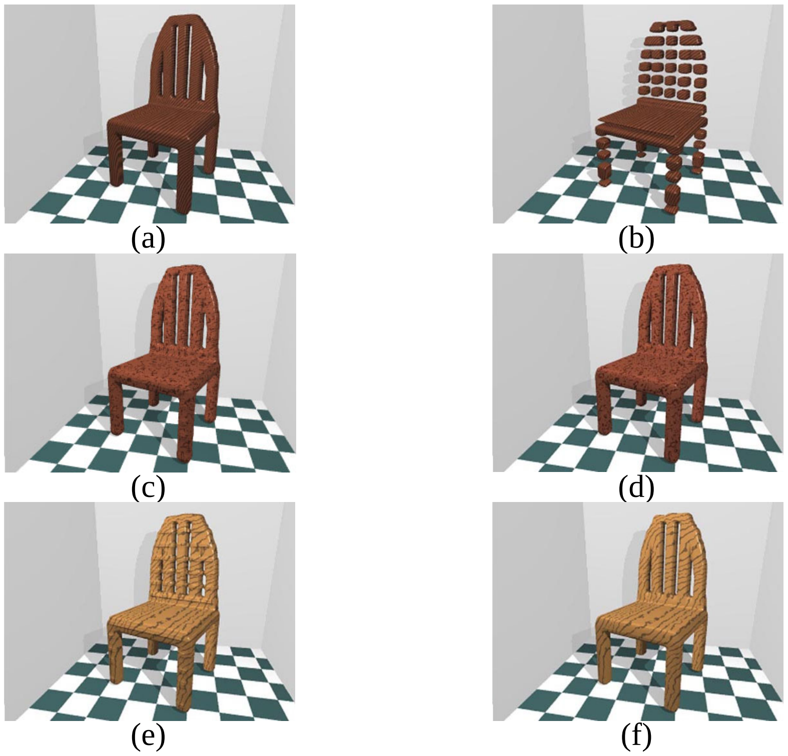


Figure 6.2: Examples of implicit PDE objects generated from cross-sectional boundary conditions. (a) Original object; (b) boundary conditions by removing several data slices along y-direction from (a); (c) and (d) are recovered fourth-order implicit PDE objects from (b) by solving (6.1) with $b = 1.2$ and $b = 4.8$ respectively; (e) and (f) are corresponding second-order objects.

6.6.2 Shape Blending

The proposed PDE formulations define the interior information of implicit objects via differential properties, which means that it is possible to automatically recover the missing information from partial data through the prototype system and guarantee intensity continuity of non-constrained parts of the working space. This feature can be applied to shape blending process by placing objects to be blended into the working space and the system will compute the connecting parts

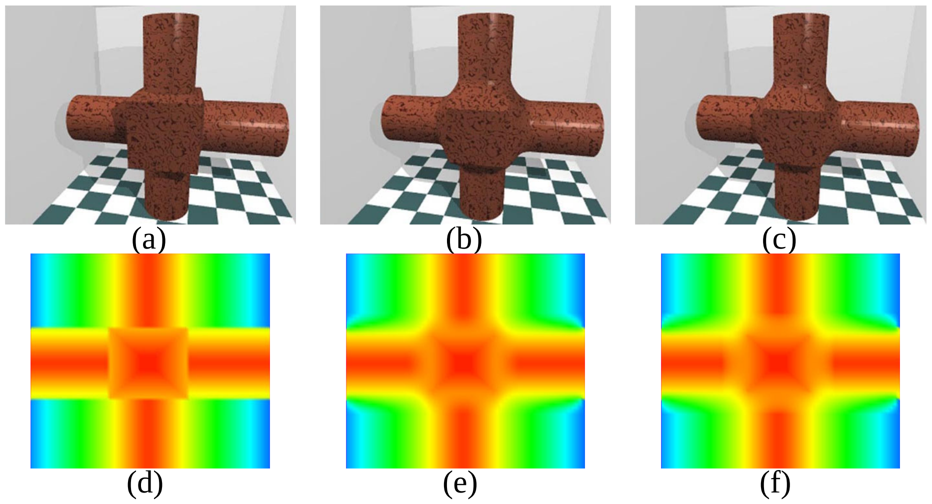


Figure 6.3: Shape blending using implicit PDEs. (a) Original dataset shown in iso-surface; (b) blended object from (a) using the fourth-order PDE; (c) blended object using the second-order PDE; (d), (e), and (f) are cross-section views of working spaces for (a), (b), and (c), respectively.

between those objects. Such kind of datasets form another type of initialization with pre-defined boundary constraints, which gives most of the information with only a small portion of the working space missing. The missing information of the working space can be approximated based on the remaining part using the PDE formulation. An example of shape blending is shown in Fig. 6.3 including blended results using different order PDEs, where the fourth-order blended shape is smoother than the second-order result. The above two types of boundary conditions allow the implicit PDE module to model volumetric datasets.

6.6.3 Shape Design Using Sketch Curves

To maximize the modeling potential of implicit PDEs, this dissertation provides a set of toolkits using the PDE techniques to reconstruct objects from spatial

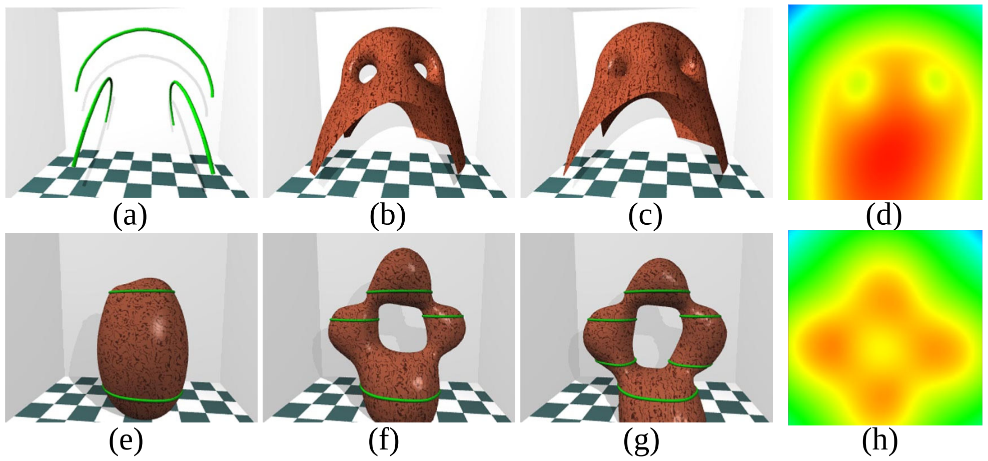


Figure 6.4: Examples of shape reconstruction from sketch curves. (a) A set of open curves; (b) and (c) are reconstructed iso-surfaces at different iso-values, respectively; (d) a cross-section view for (b) and (c); (e), (f), and (g) show an example of generating implicit shapes by incrementally defining a set of B-spline curves; (e) an object defined by two curves; (f) the refined object by adding two sketch curves; (g) the shape reconstructed from six B-spline sketch curves; and (h) a cross-section view.

sketch curves of specified intensity values. Because with this type of constraints, the boundary information around the working space is missing, it is extremely difficult to directly solve the implicit PDE under such constraints. Therefore, this dissertation employs techniques such as the RBF method for the interpolation problems to obtain an initial guess for the implicit PDE shapes subject to those constraints. Then the iterative solver is used to get a smooth solution. When performing the RBF method, the gradient information indicating the change of the intensity values around the constraints will be needed to define the inside and the outside of the reconstructed shape. If the gradient information is not provided by users, the PDE modeling system calculates the gradient at each sample point of the constraints according to the local tangent plane of the curve in the point's neighborhood, as explained in Fig. 6.5. The system also allows designers to interactively input certain sketch curves such as B-spline curves with specified intensity values, which permits the initial sketch curves to be modified directly. Note that, the sketch curves are not required to be planar curves. The sketch curves can even be open curves, which may result in open iso-surface instead of solid objects. Fig. 6.4 demonstrates several examples using sketch curves.

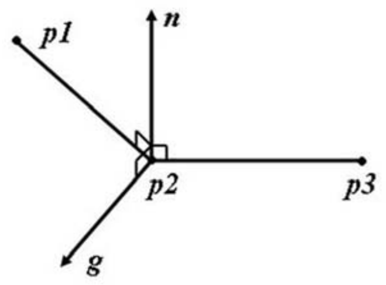


Figure 6.5: Illustration of computing the gradient direction. $p1$, $p2$, and $p3$ are neighboring points on a discretized curve; $\mathbf{n}$ is the normal of their local plane; and $\mathbf{g}$ is the gradient vector for $p2$.

When modeling more complex shapes from sketch, usually there are a large number of sketch curves to be enforced, which will increase the number of calculations dramatically. Moreover, sometimes the sketch curves are only designed to model the local area they resides, so their global contribution are not desirable. To address such issues, the system allows users to compute the initial guess of implicit PDE objects using the RBF method for the selected subset of sketch curves at any local region of the working domain without disturbing the outside area. At the initialization stage, when using RBF method to compute the initial guess of the implicit shape, users are prompted to select interested curves, define the region in the working space to reconstruct the subset of the object, as well as indicate if curves that only part of them inside the specified area can make contribution to the reconstruction. After all the sampled intensity values in each of the sub-regions of the working space are computed, the implicit PDE module can perform a global blending process to put sub-regions together. This feature can reduce the number of calculations of the RBF method, and provide fast reconstruction by sculpting sketch curves. Moreover, CSG sculpting tools can be easily enforced accordingly. Fig. 6.6 shows an example.

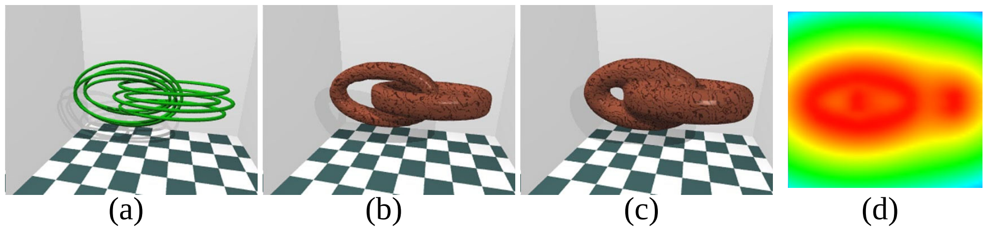


Figure 6.6: Example for performing the RBF initialization locally. (a) Two set of sketch curves; (b) and (c) are reconstructed implicit shapes rendered at different iso-values; (d) a cross-section view.

6.6.4 Shape Reconstruction from Unorganized Scattered Data Points

Implicit functions are commonly used for shape reconstruction from scattered data points. In this dissertation, the implicit PDE techniques not only reconstructs objects from unorganized scattered datasets, but also recovers information of the entire working space where objects reside, with which direct manipulations of objects can be easily applied. Similar to sketch curve constraints, intensity values at boundaries of the working space are unknown. However, for scattered datasets where the number of constraints is extremely large and there is no gradient information available, RBF method is not suitable for computing the initial guess. In

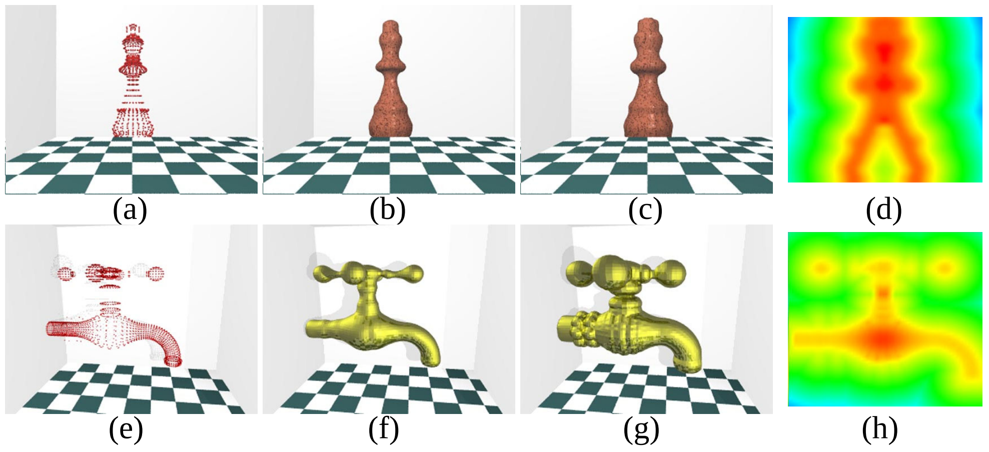


Figure 6.7: Examples of shape reconstruction from scattered data points. (b) and (c) are iso-surfaces at different iso-values of the object reconstructed from (a); (f) and (g) represent the reconstructed shape from (e) at different iso-values; (d) and (h) are cross-section views.

such case, the system uses the signed distance field approximation based on the constraints. The initial intensity values on the sampling grids are computed by the fast-tagging algorithm introduced by [161] based on their signed distances to the data point constraints and then iterative solvers are used to conduct a smoothing task. Two examples are shown in Fig. 6.7.

6.7 Sculpting and Manipulation Toolkits for Implicit PDEs

The system provides a set of toolkits for global deformation and local editing of implicit objects.

6.7.1 Modifying Blending Coefficients

The coefficient functions $a(x, y, z)$, $b(x, y, z)$, and $c(x, y, z)$ can influence the solution of the implicit PDEs. They control the relative intensity blending and the level of variable dependence among x, y, and z directions, thus they can be treated as generalized material properties over the volumetric working space. Consequently, users can control how the boundary and additional conditions influence the interior intensity distribution by modifying the length scale of these functions at arbitrary locations (i.e., $a_{i,j,k}$, $b_{i,j,k}$, and $c_{i,j,k}$). In general, users can define the control functions $a(x, y, z)$, $b(x, y, z)$, and $c(x, y, z)$ interactively over the specified grid point $\{i, j, k\}$. The system allows users to modify them locally to deform the shape. Fig. 6.2 has examples of implicit PDE objects subject to different coefficient values.

6.7.2 Sketch Curve Sculpting

Implicit objects can be defined by specifying a set of sketch curves which outline their rough shapes. The implicit PDE model provides interactive shape design toolkits that allow users to manipulate the sketch curves in order to deform underlying reconstructed implicit objects. The sketch curves defining the rough shape of the object can be obtained by either pre-defined curve network or B-spline curves from users' direct input. The PDE modeling system allows users to modify the geometric shape, intensity value, as well as gradient directions of the sketch curves interactively in order to get desired objects.

In order to modify the sketch curves smoothly, B-spline approximations for those curves are calculated at the initialization stage, then users can sculpt the curves interactively by manipulating the B-spline control points via sculpting, translation, and rotation. Because the reconstructed implicit object is required to interpolate those sketch curves which define its outlining shape approximately, it will follow the shape changes accordingly. Fig. 6.8 has an example for sculpting the shape of a selected sketch curve. The intensity value of a sketch curve determines where the final shape of the implicit object should pass through at the level set of its value. By modifying intensity values of selected curves, users can manipulate the objects accordingly.

Furthermore, according to the gradient definition, intensity values increase along gradient directions of sketch curves and decrease in the opposite directions in general. Gradient directions provide information of intensity distributions starting at the sketch curves and propagating to the neighborhood, which defines the inside and outside of the object. Without the definition of gradient directions, the PDE solution will be trivial. Therefore, gradient information of sketch curves is required to reconstruct a unique shape. Accordingly, changing gradient directions at selected sketch curves means modifying directions of intensity changes in the implicit working space and will result in different implicit shapes. The system allows users to specify the gradient direction of each individual sketch curve to construct different implicit PDE objects. Refer to Fig. 6.9 for examples of specifying and modifying gradient directions at sketch curves. Without further specification, other examples in this chapter have gradient directions point inward the curves by default.

6.7.3 Local Manipulation of Implicit PDE Solids

Usually the sketch curve sculpting will deform the entire reconstructed shape, which only offers global manipulation and is less intuitive for ordinary users to handle. Even with the specification of local areas of interests containing the sculpted sketch curve, the sculpting will affect all the points in the selected regions. Moreover, sometimes the input constraints alone can not guarantee a satisfactory solution of the constructed shape. Therefore, direct modification in selected areas is desirable, especially when the overall recovered shape is satisfactory but minor changes in small localized areas are needed. The PDE modeling system provides interactive tools of the intensity value modification in selected regions to sculpt the reconstructed shape. The modification will be enforced into (6.6) or (6.7). Using the aforementioned techniques, (6.6) and (6.7) can be solved to obtain the modified objects. For local manipulations, only the intensity updates in the neighborhood of selected regions need to be calculated. The intensity values are governed through the PDE and the selected regions usually have relatively small number of grids comparing with the entire working space. The update of the new intensity values for such regions will be quickly obtained through the finite-difference solver. Therefore, interactive manipulations for local sculpting of implicit PDE objects can be easily achieved.

Traditional implicit techniques for data reconstruction do not support direct manipulations on arbitrary locations in the volumetric working space. Changes on pre-defined constraints will cause global deformation. It is more desirable to offer users editing functionalities for the interior properties with interactive interface.

Local Intensity Modification

Besides the local RBF approximation for local sketch curve sculpting, the system also allows users to specify any interior region of the sampling grids, and applies intensity changes only within the specified region. Alternatively, users can freeze the selected region and disallow any changes in the specified region. In the system, this can be done through interactively specifying the maximum and minimum sampling grid in x, y, and z direction of the desired region in the volumetric working space. Subsequently, any change within the region will have no influence on sampling points outside the region. The localized deformation can be easily achieved because only those equations corresponding to the points in specified regions in (4.9) will be solved. In addition, the number of computations is reduced due to fewer number of equations involved in the local sculpting. In principle, all hard constraints can be viewed as some sort of local deformation. Fig. 6.8 shows examples of local deformation.

Iso-surface Sculpting

Users can also specify an iso-surface at a particular intensity value and use a cutting plane inside the volumetric working space to get a 2D iso-contour on the plane, then stretch, push, rotate the contour, as well as add desired intensity values at specified locations to modify the shape of the iso-surface and the intensity distribution of interested areas. Refer to Fig. 6.8 for illustrative examples.

CSG Operations

The system also offers several CSG sculpting tools such as using spheres and cubes to trim/extrude/sculpt implicit objects by adding more constraints on the sampling grids of the working space. This is extremely useful for such situations when there are some minor changes needed to be done in some local small regions. Such sculpting tools make our system compatible with CSG-based implicit models by treating those models as modeling tools. Examples are shown in Fig. 6.8.

Gradient Constraints

The intensity gradient ∇ at a point (x, y, z) in the intensity field can be defined as

$$\nabla d(x, y, z) = (\frac{\partial d(x, y, z)}{\partial x}, \frac{\partial d(x, y, z)}{\partial y}, \frac{\partial d(x, y, z)}{\partial z}).$$

By applying the finite-difference technique, the gradient vector $\nabla d_{i,j,k}$ at a discretized grid point $\{i, j, k\}$ can be approximated as:

$$(\frac{d_{i+1,j,k} - d_{i-1,j,k}}{2\Delta x}, \frac{d_{i,j+1,k} - d_{i,j-1,k}}{2\Delta y}, \frac{d_{i,j,k+1} - d_{i,j,k-1}}{2\Delta z}).$$

It provides information about intensity changes in the neighborhood of (x, y, z) in the working space. Therefore, changing the direction and length of the gradient vector of a selected grid point will affect the intensity distribution in its neighborhood, and as a result, deform the object. The implicit PDE module allows users to pick a point inside the working space, specify the local region surrounding the point, and modify its gradient vector interactively, then the shape bounded by the specified local region will be deformed accordingly (refer to Fig. 6.10 and Fig. 6.11(a)).

Curvature Constraints

The mean curvature at point (x, y, z) in the intensity field can be computed from the divergence of the intensity gradient of (x, y, z), i.e. $\nabla \cdot \nabla d(x, y, z)$ [119]. In the discretized form, it can be approximated as $\nabla \cdot \nabla d_{i,j,k}$. Its definition is also related to intensity values of the point's neighbors. By changing the curvature value at a point, the shape of the object will be changed. The system allows users to manipulate the curvature at a selected grid point for implicit shape deformation. Fig. 6.11 shows examples.

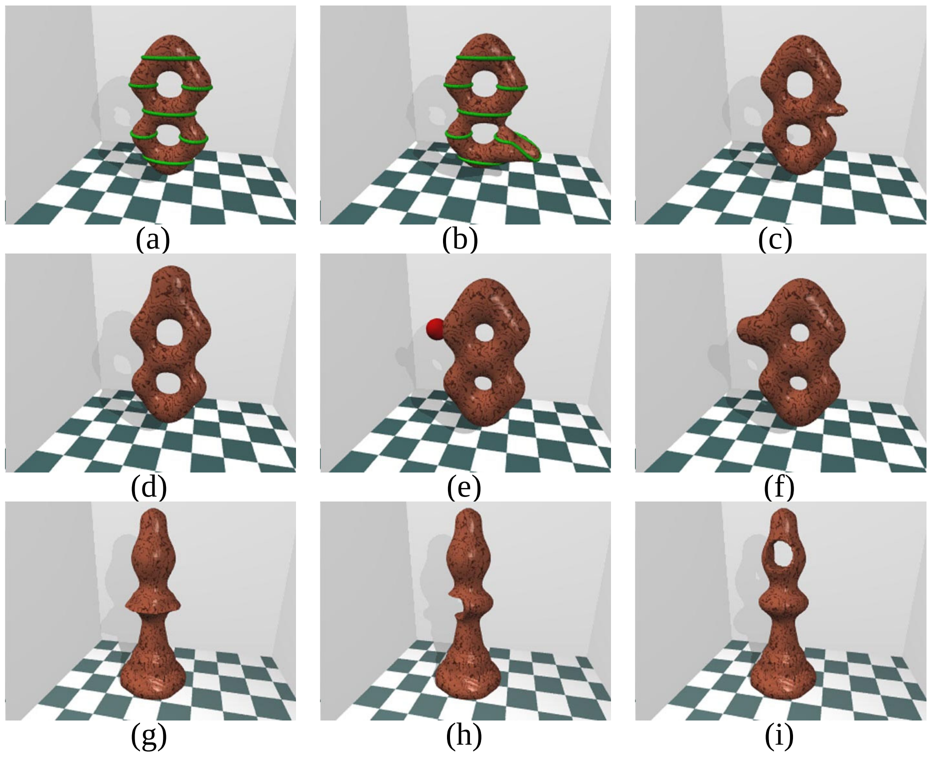


Figure 6.8: Examples of enforcing curve and direct manipulation constraints. (a) Original object with sketch curves; (b) deformed object by sculpting a selected curve; (c) changing an iso-contour; (d) deformed object subject to local region constraints; (e) adding a sphere in the working space; and (f) is the corresponding deformed object subject to (e); (g) adding constraints for an object with sharp edges; (h) and (i) are two trimming examples.

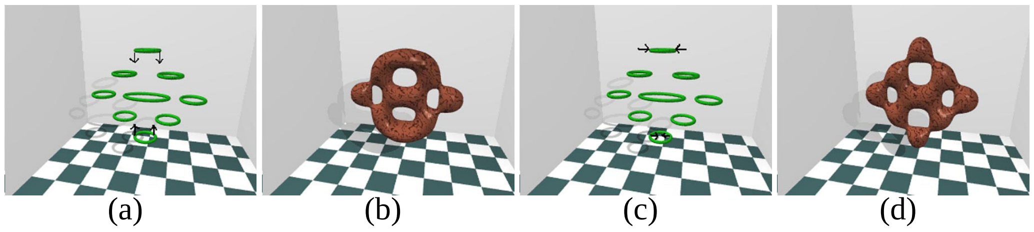


Figure 6.9: Examples for manipulating gradient directions of sketch curves. (a) and (c) are two sets of curves with the same geometric shape but different gradient directions, where the arrows show intensity increasing directions; (b) and (d) are the corresponding implicit objects.

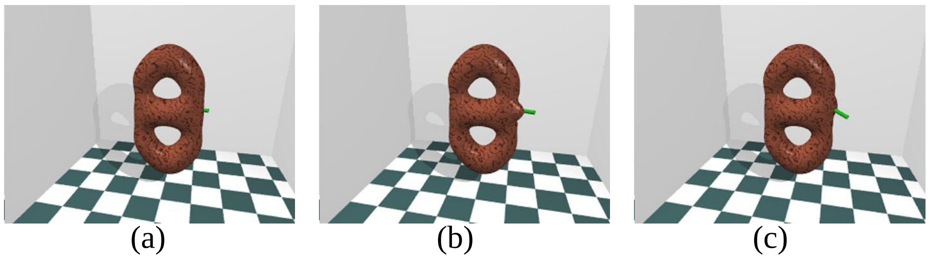


Figure 6.10: Examples of enforcing gradient constraints. (a) Original object with the gradient vector at a selected point; (b) and (c) are deformed objects by changing the gradient at the point.

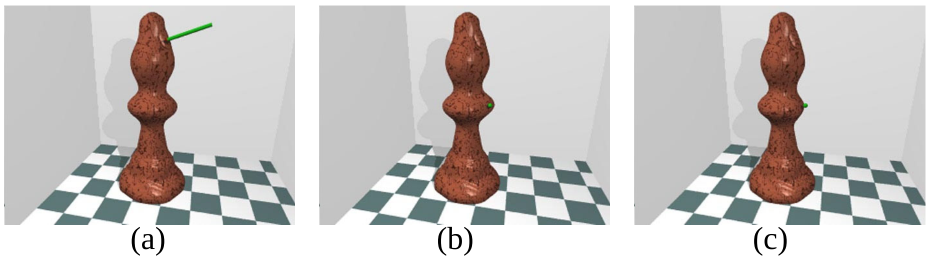


Figure 6.11: Examples of enforcing gradient and curvature constraints. (a) Deformed object by changing gradient; (b) and (c) are deformed objects by changing curvature at a selected point (shown in green) from (a).

Chapter 7

PDE-based Free-Form Modeling and Deformation

This dissertation has presented a unified dynamic approach that defines solid geometry of sculptured objects using trivariate elliptic PDEs subject to flexible boundary conditions. The proposed solid PDE formulation and its associated dynamic principle permit designers to directly deform PDE solids whose behaviors are natural and intuitive subject to imposed constraints. Users can easily model and interact with solids of complicated geometry and/or arbitrary topology from locally-defined PDE primitives through trimming operations. The system offers users various sculpting toolkits for solid design and interactive modification of physical and geometric properties of boundary surfaces, as well as any interior parts of solid objects. Furthermore, this dissertation also integrates implicit scalar intensity information with solid geometry and uses elliptic PDEs to govern both geometry and density behavior of solid objects. It offers free-form modeling and deformation for integrated PDE solid objects with density attributes. This work has been published in Proceedings of Pacific Graphics 2001[45]. Another paper based on this work is under review for journal publication[48].

7.1 Introduction and Motivation

At present, curve and surface modeling techniques are extensively used for representing a wide range of geometric shapes. However, such representations are far from adequate for modeling real-world objects when both interior properties and dynamic behaviors of the underlying shape are of prime significance to modelers. Although the use of elliptic PDEs on 2D parametric domain with physics-based techniques offers interactive and dynamic modeling of PDE surfaces and surface displacements, surface representations fall short in modeling most of the real-world objects where interior geometry and material distribution are required for both synthesis and analysis processes.

In contrast, solid modeling has emerged as a powerful, convenient, and natural paradigm for representing, manipulating and interacting with 3D objects in graphics, animation, CAD/CAM, art and entertainment, scientific visualization, and virtual environments. It greatly enhances existing surface modeling techniques primarily because a solid model offers engineers an unambiguous shape representation of a physical entity. In essence, the CAD-based solid representation of a real-world physical object is both geometrically unambiguous and topologically consistent. There is a wide array of solid modeling techniques [65, 100] including: Constructive Solid Geometry (CSG), Boundary representation (B-rep), cell decomposition, and trivariate free-form parametric *superpatches*. The CSG approach exploits semi-algebraic sets and Boolean operations on simple primitives such as cubes, spheres, cylinders, cones, and tori to construct more complex solid models. The B-rep technique typically defines a solid object via a set of its boundary surfaces along with extra topological information. The cell decomposition method usually uses 2D cross-sectional slices or cubical units (*e.g.*, voxels) to approximate complicated solids with hierarchically structured octree schemes. Despite many advances in solid modeling during past several decades, conventional solid modeling techniques can be rather rigid and inflexible. Such solid modeling representation schemes encounter difficulties in interactive sculpting of solid objects, solid geometry deformation, topology modification, and kinematic and dynamic analysis of physical solids. In general, prior representations in solid modeling fall short in offering designers an array of flexible and powerful modeling and sculpting tools.

On the other hand, free-form solid modeling combines the benefits of free-form boundary surfaces and interior geometry within a unified framework. It provides users more flexible design interfaces for modeling a large variety of objects. It also facilitates cost-effective algorithms for evaluation and manipulation of solid geometry. Typical examples of free-form solid models include trivariate B-splines, Hermite solids and NURBS solids. However, free-form solids such as trivariate B-splines typically offer users more degrees of freedom (*i.e.*, control points, weights, etc.) than what they can actually handle. Furthermore, free-form solids are restrained to model topologically regular shapes. It is difficult to extend the geometric coverage of free-form solids to shapes of arbitrary topology without resorting to various non-intuitive geometric constraints.

As for shape manipulation, free-form deformation (FFD) provides efficient shape deformation for geometric objects. In essence, shape deformation based on FFD is conducted *indirectly* by deforming the embedding space in which the shape is defined. It provides a unique parameterization for the shape to define its position in the space. When the embedding space is deformed, the embedded shape is deformed accordingly based on its parameterization. One appealing advantage of FFD in comparison with other traditional modeling techniques is that it can be applied to arbitrary geometric object since the embedding space is independent to the geometric representation and topological structure of the embedded target. In past several decades, FFD techniques are among popular modeling methods for shape modeling and manipulation. Sederberg and Parry [125] proposed a FFD technique to deform solid objects. They embedded the object in a parallel-piped lattice structure and mapped it to the deformed lattice using a trivariate Bezier spline. Later, Coquillart [31] extended their technique as EFFD by using nonparallel-piped 3D lattices. This EFFD technique offers shape deformation by bending the shape along an arbitrarily shaped curve or adding randomly shaped bumps to it. Hsu *et al.*[68] introduced direct manipulation of B-spline FFD which used least squares to calculate the control points of B-spline FFD based on the direct movement of objects. MacCracken and Joy [93] introduced a subdivision-based FFD technique that uses Catmull-Clark subdivision volumes. An underlying model can be deformed by embedding the model into the converging sequence of lattices and then tracking new positions of the model according to the deformed lattice sequences The subdivision-based FFD offers a broader range of shape deformations under various control volumes. However, the definition of control volumes is laborious and difficult. Singh and Fiume [132] presented a geometric deformation technique called ”wires” to deform objects using space curves and implicit functions. Crespin [32] presented a FFD technique using deformation primitives. The primitives provided deformations on associated parts of an object. Then the deformations are combined together by associated blending functions. Recently, Schmitt *et al.*[121] introduced a shape-driven technique for functionally defined heterogeneous volumetric objects. Hua and Qin [71] presented a scalar-field-based free-form deformation (SFD) technique by establishing deformation methods defined on implicit scalar fields instead of constructing a mapping function from geometric deformation primitives. However, the FFD-based techniques to model complex objects require large number of control vertices to define their embedding space. To achieve arbitrary shape deformation, they need complex operations on these control vertices.

By taking advantages of the boundary-value elliptic PDEs, PDE solid model can effectively model objects through the use of certain elliptic PDEs of u, v, w with only a few boundary conditions. In principle, PDE solids can be reconstructed from a small set of heterogeneous boundary conditions. The solids' interior information will be automatically provided by solving the given PDEs. Hence, fewer parameters are required than those of free-form or subdivision solids. PDE solids offer a powerful solid representation with combined advantages of conventional solid modeling techniques, such as spline-based behavior, boundary surface representations, and underlying parameterization for (generalized) cell decomposition in the interior. Therefore, PDE solids have potential to integrate CSG, B-rep and cell decomposition into a single framework. Furthermore, because PDE solids offer mapping between parametric space and physical space, natural FFD operations can be easily provided for embedded objects inside the PDE solid working space. PDE solids can also unify both geometric and physical aspects for real-world models. Various heterogeneous requirements can be enforced and satisfied simultaneously. They are invaluable throughout the entire modeling, design, analysis, and manufacturing processes.

However, besides simple geometric conditions enforced over PDE solid boundaries, there is no formal mechanism for the direct editing of PDE solids anywhere across their domain in prior work of PDE solids. Traditional elliptic PDE solids are only computed from a set of regular boundary surfaces. More flexible and general boundary constraints are yet to be addressed. Conventional PDE techniques are unable to support localized geometric operations for solid models. Global control is less intuitive to manipulate. Despite the great potential to integrate different techniques such as CSG, B-rep, cell decomposition, and free-form solids, previous PDE solid techniques only make use of boundary information and many interior properties and features have not yet been considered. The FFD features of PDE solid geometry haven't been facilitated. In addition, previous PDE solid models seek for analytic solutions based on Fourier transforms, which don't have enough degrees of freedom for designers to design and manipulate solid objects. Numerical techniques, in contrast, provide numerical discrete approximations for solid PDEs and allow the enforcement of arbitrary constraints including point interpolation and manipulation constraints and other functional requirements. They also make it possible to incorporate physical and material properties into the PDE solid framework.

To further broaden applications of PDE techniques in geometric and visual computing, this dissertation extends PDE solid techniques for more general PDE-based free-form modeling and deformation with interactive manipulations. In the framework, PDE solids are associated with geometric shapes, physical properties, and intensity attributes at the same time. The PDE solid geometry can be defined by boundary surfaces or a set of boundary curve networks. Geometric manipulation toolkits for boundary surface sculpting, local control and trimming operations using simple CSG tools and user-specified datasets are offered to obtain arbitrary topological shapes. The interactive sculpting and manipulation can be accomplished by integrating PDE solids with physics-based modeling techniques, which will offer users intuitive editing toolkits for solid modeling. Furthermore, the implicit PDE is incorporated into the parametric PDE solid formulation to govern intensity attributes of solid objects. This integration augments free-form shape modeling and deformation based on PDE solids with implicit properties. The unified PDE framework provides arbitrary deformation of solid objects with parametric geometry and implicit intensity distributions. With modeling properties of both implicit PDEs and parametric PDE solids, the integrated PDE formulation can model, blend and deform objects with arbitrary topology.

7.2 PDE Solid Formulation

7.2.1 Formulation of PDE Solid Geometry

Bloor and Wilson [16] initially employed the second-order elliptic PDE (2.7) to construct PDE solids. However, direct manipulation of PDE solids defined by boundary surfaces is lack from their work. To allow users to directly modify PDE solid objects, direct control of boundary surfaces is more desirable. In order to make previously-developed fourth-order elliptic PDE surface sculpting techniques available for solid modeling, this dissertation employs the fourth-order elliptic PDE to formulate parametric solid geometry:

$$(a^2(u,v,w)\frac{\partial^2}{\partial u^2} + b^2(u,v,w)\frac{\partial^2}{\partial v^2} + c^2(u,v,w)\frac{\partial^2}{\partial w^2})^2\mathbf{X}(u,v,w) = \mathbf{0} \qquad (7.1)$$

where $a(u,v,w)$, $b(u,v,w)$ and $c(u,v,w)$ are coefficient functions of u, v, and w that offer local control for the behavior of PDE solids. The constant smoothing coefficients a, b, and c in (2.7) are replaced by arbitrary functions over u, v, and w, to offer users more flexibilities for interactive manipulation. Usually they are set to be constant values over the parametric domain except those regions of interest during the manipulation process.

To solve (2.7) and (7.1), at least six boundary conditions are required in order to derive a unique solution. This dissertation restrains u, v, w to vary between 0 and 1, because reparametrization does not change the PDE solid geometry if u, v, or w belongs to $[a, b]$. The six boundary PDE surfaces define three surface pairs on the solid boundaries at $u = 0$, $u = 1$, $v = 0$, $v = 1$, $w = 0$, and $w = 1$ are in the form of (2.8). These six surfaces may share corresponding boundary curves with each other, and they are all open surfaces along their boundary curves. Furthermore, because a PDE surface can be derived from a set of Coons-like or Gordon-like boundary curves, boundary conditions in the form of arbitrary curve network are also possible to uniquely define PDE solids. A same order of elliptic PDE is used to model boundary surfaces of PDE solids. Thus the corresponding PDE boundary surface formulation for (2.7) has the following form:

$$(b_1^2(u_1, u_2)\frac{\partial^2}{\partial u_1^2} + b_2^2(u_1, u_2)\frac{\partial^2}{\partial u_1 \partial u_2} + b_3^2(u_1, u_2)\frac{\partial^2}{\partial u_2^2})\mathbf{X}(u_1, u_2) = \mathbf{0}, \tag{7.2}$$

where $(u_1, u_2) \in \{(u, v), (u, w), (v, w)\}$. The boundary surface formulation for fourth-order PDE solids has a similar form of (4.1).

The boundary curves to define a PDE solid have the following form:

$$\mathbf{X}(u, v_i, w_j) = \mathbf{U}_{ij}(u), \mathbf{X}(u_k, v, w_l) = \mathbf{V}_{kl}(v), \mathbf{X}(u_r, v_s, w) = \mathbf{W}_{rs}(w). \tag{7.3}$$

This type of general and arbitrary boundary conditions provide users more flexible tools to model solid objects with fewer parameters, and are capable of modeling solids that must pass through a set of curves that serve as general constraints.

Similar to (4.4), by using difference equations to approximate the discretized PDE in u, v, w parametric domain, a set of algebraic equations can be obtained:

$$\mathbf{G}\mathbf{X} = \mathbf{z}, \tag{7.4}$$

which can be solved through iterative methods with multi-grid solvers to improve the time performance of the system.

7.2.2 Formulation of PDE Solid with Intensity Attributes

The same PDE for parametric PDE solid geometry can also be used to govern the intensity properties associated with solid objects. Similar to (7.1), the corresponding fourth-order PDE to model both geometry and scalar intensity attributes of the PDE working space has the following form:

$$(\mathbf{a}^2(u,v,w)\frac{\partial^2}{\partial u^2} + \mathbf{b}^2(u,v,w)\frac{\partial^2}{\partial v^2} + \mathbf{c}^2(u,v,w)\frac{\partial^2}{\partial w^2})^2\mathbf{P}(u,v,w) = \mathbf{0}, \quad (7.5)$$

where $\mathbf{P}(u,v,w) = [\mathbf{X}(u,v,w), d(u,v,w)]^\top$, $\mathbf{a}(u,v,w)$, $\mathbf{b}(u,v,w)$, and $\mathbf{c}(u,v,w)$ are blending coefficient functions. To offer users more degrees of freedom when modeling the integrated PDE objects, the system formulates the blending coefficient functions to have different controls on geometry and intensity attributes, *i.e.*, the coefficient functions can be defined as follows:

$$\mathbf{a}(u,v,w) = (\begin{array}{cc} a_g(u,v,w) & 0 \\ 0 & a_d(u,v,w) \end{array}),$$

$$\mathbf{b}(u,v,w) = (\begin{array}{cc} b_g(u,v,w) & 0 \\ 0 & b_d(u,v,w) \end{array}),$$

$$\mathbf{c}(u,v,w) = (\begin{array}{cc} c_g(u,v,w) & 0 \\ 0 & c_d(u,v,w) \end{array}).$$

Using this formulation, (7.1) can be viewed as a special case where the intensity component $d(u,v,w)$ is constant across the entire working space.

(7.5) can be discretized and approximated using the same finite-difference scheme:

$$\mathbf{G}\text{''}\mathbf{P} = \mathbf{s}, \quad (7.6)$$

and (7.6) can be solved using similar techniques for (7.4).

7.3 Physics-based Modeling of Free-Form PDE Solids

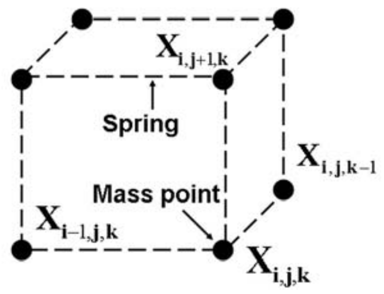


Figure 7.1: Mass-spring network for the discretized PDE solid.

This dissertation also employs the integrated mass-spring model of PDE objects whose dynamic behavior is governed by (4.6). In general, a continuous dynamic solid can be discretized into a collection of mass-points connected by a network of springs across nearest neighbors (and/or along both diagonals). Other springs can be incorporated into the discretized solid if certain types of dynamic behavior are more desirable. A mass-spring model is used because of its simplicity and efficiency. Fig.7.1 shows the illustration of the solid mass-spring model which attaches mass points to geometric grids and adds springs between immediate neighbors on the PDE discretization along u, v, and w. The external force $\mathbf{f}$ can be computed based on various additional constraints. The PDE model can be dynamically manipulated with forces by solving

$$(2\mathbf{M}+\Delta t\mathbf{D}+2\Delta t^2\mathbf{K}+2\Delta t^2\mathbf{G})\mathbf{X}^{t+\Delta t} = 2\Delta t^2(\mathbf{z}+\mathbf{f})+4\mathbf{M}\mathbf{X}^{t}-(2\mathbf{M}-\Delta t\mathbf{D})\mathbf{X}^{t-\Delta t}. \tag{7.7}$$

The behavior of the dynamic PDE solid is controlled by both of the Lagrangian equation of motion and the given PDE with boundary and additional geometric constraints. The movements of sample points of the integrated mass-spring PDE solid model under sculpting are decided by the PDE with constraints, mass, damping distributions of points, as well as the stiffness of the springs connecting those points. This *hybrid* formulation permits users to obtain a solid that satisfies both geometric criteria and functional requirements at the same time.

7.4 Multi-Grid Approximation for PDE Solid

(7.4) can be solved using finite-difference methods. The large number of sample points of a PDE surface/solid results in the slow convergence of iterative techniques. The multi-grid approximation based on simple subdivision schemes is used to improve the computation performance. Since there are two types of boundary conditions, *i.e.*, curve network and surfaces, different multi-grid approximation schemes are employed to handle two types of boundary constraints, respectively.

If boundary conditions come from curves, boundary surfaces shall be first computed. This can be done by starting with a small number of sample points at the coarsest resolution of PDE boundary surfaces, and the approximate solution of PDE surfaces can be easily derived after several iterations. Then, the PDE solid at the coarsest resolution is solved. Users can refine the coarse mesh through subdivision and use the new subdivided mesh as an initial guess for subsequent iteration steps. The finer grid is then computed iteratively to achieve a more accurate and smoother solution of boundary PDE surfaces as well as the PDE solid. For further refinement over the finest grid, the multi-grid approximation starts with the up-sampling of all boundary curves through the use of four-point interpolatory subdivision scheme [52] in order to guarantee the smoothness requirement of refined curves.

If boundary conditions come from connected surfaces, the approximation scheme should be slightly modified. The process starts with the coarsest resolution of boundary surfaces through down-sampling to obtain a coarse solution of the solid. Then during the refining process, more points are sampled over boundary surfaces until it reaches the finest resolution. After that, the subdivision process may continue to reach even finer resolution. In this scenario, the given boundary surfaces are considered as constrained PDE surfaces, requiring four curves as boundary conditions and the originally defined surface sample points as hard constraints. Then the four-point interpolatory subdivision scheme is used to subdivide boundary curves and compute unknown surface points by solving the surface PDE subject to subdivided boundary curves and hard-constrained original surface points.

7.5 Interactive Editing Toolkits for Free-Form PDE Solids

This dissertation expands the PDE-based modeling system to provide users direct manipulations and trimming operations for PDE solid models as well as free-form deformation of arbitrary objects.

7.5.1 Solid Geometry Initialization

The system supports two types of initialization for the PDE solid geometry, *i.e.*, initial boundary surfaces, or initial boundary curves. At the start of the initialization phase, users must specify the boundary type, *i.e.*, whether the boundary conditions are given as pre-defined surfaces, or connected boundary curve network for the PDE solid.

For initialization with pre-defined boundary surfaces, the system can obtain the already defined surfaces that are patched together and form the outline of PDE solid from file or use previous techniques to generate PDE surfaces. Then using the surfaces as boundary conditions, a PDE solid bounded by these surfaces can be obtained as the solution of (4.4). Fig. 7.2 shows two examples.

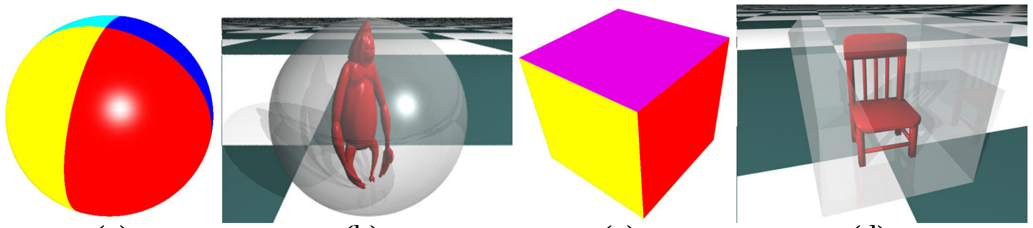


Figure 7.2: PDE solid examples generated from given boundary surfaces. (a) and (c) are two sets of boundary surfaces; (b) and (d) are the corresponding PDE solids (displayed using transparent material) subject to (a) and (c) with embedded datasets, respectively.

If users decide to employ the curve network as boundary conditions, there will be at least 12 curves required to define the Coons-like boundary conditions for the six boundary surfaces. There are two steps in this case: (1) derive boundary surfaces from the boundary curves users specified by solving (4.4); (2) solve (7.4) to obtain the corresponding PDE solid. The system uses Coons-like boundary conditions for the boundary curve network because every two neighboring surfaces share one boundary curve. To make sure the solved PDE surfaces satisfy such conditions, the shared boundary curves need to be defined. The boundary surfaces can be even defined more precisely by adding more curves as boundary conditions, which leads to the Gordon-like boundary conditions of boundary surfaces. Fig. 7.3 shows examples of curve network as boundary conditions.

The large number of sample points of a PDE surface/solid results in the slow

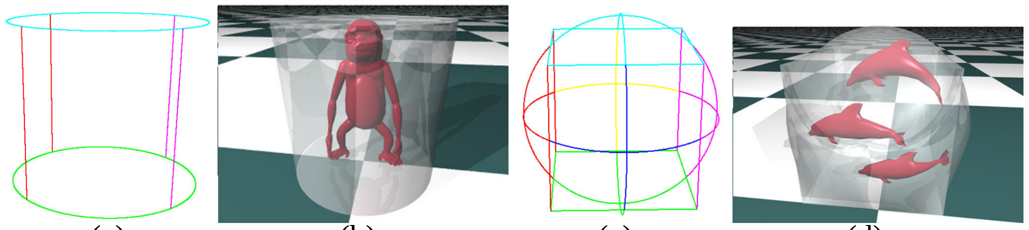


Figure 7.3: Examples of PDE solids subject to boundary curve network. (a) Coons-like boundary curves; (b) the corresponding PDE solid; (c) Gordon-like boundary conditions, and (d) the PDE solid subject to (c). The PDE solids are displayed using transparent material with embedded datasets.

convergence of iterative techniques. The system employs a multi-grid approximation based on simple subdivision schemes to improve the computation performance.

7.5.2 Boundary Manipulation

Users can modify the global shape of a PDE solid through boundary manipulation. The system permits users to directly modify boundary surfaces using the afore-mentioned PDE surfaces sculpting toolkits, then eventually define the PDE solid. To modify a PDE solid through boundary conditions, users must select a boundary surface for the editing purpose, then use the sculpting toolkits provided by the PDE modeling system to modify the selected surface. Fig. 7.4 shows two examples of boundary manipulation with curve constraints. The operations of boundary PDE surfaces provide a way for direct manipulations of PDE solids.

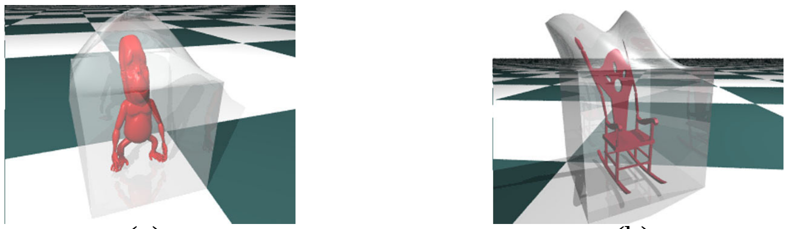


Figure 7.4: Modifying PDE solids via curve constraints of boundary surfaces.

7.5.3 Direct Solid Manipulation

One advantage of PDE solids is that the solid interior is controlled by PDEs without the need of specification on interior material distributions. PDE solids provide an integrated scheme that not only expands the B-rep method to cover the interior information but also supports Boolean operations associated with CSG models. More importantly, users can deform the interior of a PDE solid by enforcing additional constraints inside the solid without changing its boundary conditions. The interactive operations inside a PDE solid include local region sculpting and solid trimming and deformation. This can be done using the same techniques for PDE surfaces, by replacing several equations in (7.4) using additional constraints to obtain a constrained system:

$$\mathbf{G}_c\mathbf{X} = \mathbf{z}_c, \tag{7.8}$$

Region Manipulation

Traditional PDE solids only support boundary manipulations which lead to global deformation of the entire solid space. It is more desirable to offer users editing functionalities on the interior properties with interactive interface. The system provides a set of toolkits that allow designers to specify any interior region of a solid, and only enforce local deformation in the selected region. Alternatively, the selected region can be fixed and there will be no changes in the specified region. In the system, this can be done through: (1) interactively specifying a region in $[u, v, w]$ domain, (2) employing some basic CSG-based tools such as spheres and cubes to navigate the entire parametric domain to define the region of interest, or (3) embedding datasets within the PDE solid in order to define the particular region. Subsequently, any changes within the region will have no effect on points outside. The localized deformation can be achieved easily because only those equations corresponding to the points of specified regions in (4.4) will be solved. In principle, all hard constraints can be viewed as some sorts of local deformation. Fig. 7.5 and Fig. 7.6 show examples of local deformation.

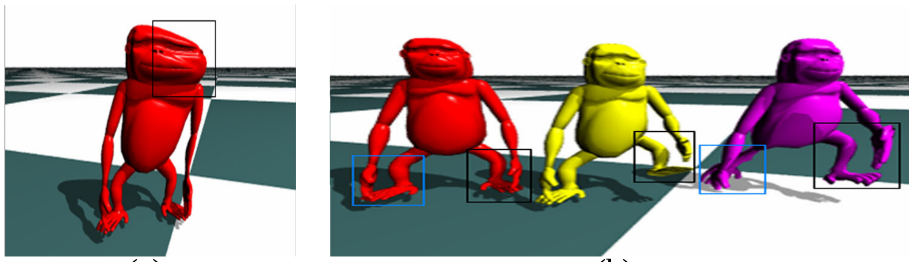


Figure 7.5: Directly modify the trimmed PDE solid. (a) A deformed object obtained by moving an interior region; (b) the deforming sequence of objects by rotating selected regions. The rectangles in the figures show the selected regions.

Solid Trimming

One of the disadvantages of parametric solids is that it is difficult to model objects of arbitrary topology. Trimming operation offers an alternative way to model objects with irregular shape. The system offers users trimming functionalities on a PDE solid for sculpting of arbitrary topological shapes. After the region of interest is selected, users can remove material from the PDE solid either inside or outside

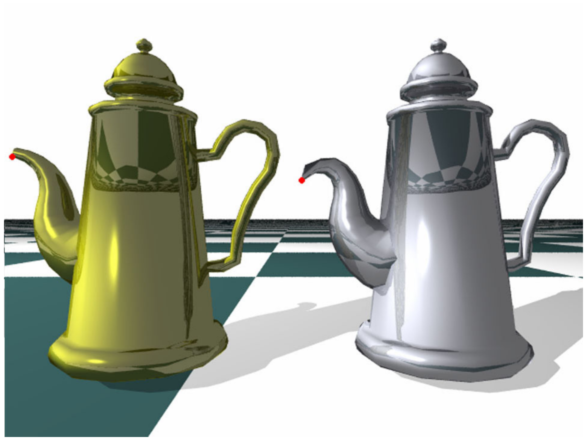


Figure 7.6: Directly modify a point on the trimmed PDE solid. The red point is selected.

the specified region. Multiple selected regions are also supported in the system, permitting the trimming on multiple regions simultaneously. Furthermore, according to the idea of CSG models, simple shape primitives such as sphere, cube, or cylinder, can be placed at any position inside the parametric domain as trimming tools, then users can move the shapes along the u, v, or w directions, and all the regions covered by their navigating path will be chosen/discarded according to the specified Boolean operations. Such tools allow the CSG construction of complex objects based on PDE solids. Fig. 7.7 shows trimming examples.

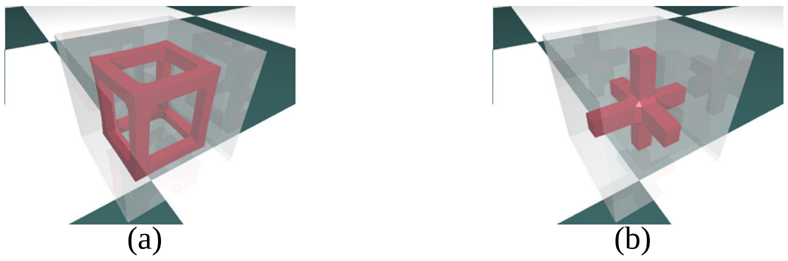
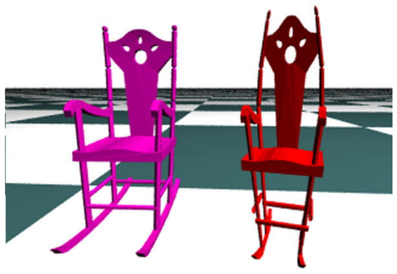


Figure 7.7: Examples using CSG trimming operation in a PDE solid. The trimmed parts are shown in red covered by transparent original solids.

7.5.4 Geometric Free-Form Deformation

Because the trivariate PDE provides a mapping between the parametric space and physical space, it's straightforward to use the PDE for FFD applications. The idea comes from trimming operations and the region-fixing method introduced in above sections, *i.e.*, users can embed datasets into PDE parametric domain and map them to the physical space to obtained deformed shapes. The mapping of embedded datasets to different PDE solids will result in different shapes. In essence, this is analogous to the principle of FFD, except the transformation between the parametric space of objects to physical working space where the objects are embedded is governed by an elliptic PDE. The free-form deformation based on PDE solids can greatly expand the coverage of PDE solid applications, making it possible to obtain PDE-governed free-form modeling and deformation for arbitrary topological shapes. Fig. 7.8 shows some examples.


(a)

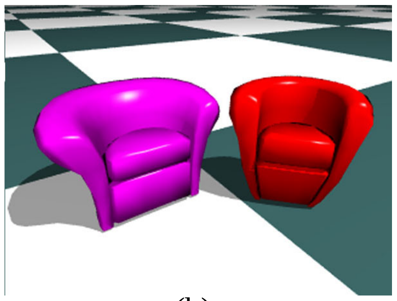

(b)

Figure 7.8: Free-form deformation based on PDE solids. In each figure, the object on the left is obtained by embedding it into a PDE cube, and the object on the right is obtained from a PDE sphere.

7.6 Intensity-based Free-Form Modeling and Deformation

To obtain more free-form modeling and deformation features for arbitrary objects, this dissertation incorporates implicit properties into the PDE solid geometry, *i.e.*, using the PDE formulated in (7.5) to model solid geometry and scalar intensity attributes at the same time. This integration will provide the modeling advantages of both implicit models and PDE solids in a single framework and offer users more degrees of freedom during free-form shape modeling and deformation operations. Manipulation toolkits provided in the implicit PDE module can be directly applied to model the intensity distribution in the parametric PDE solid space. The combination of parametric and implicit modeling based on PDEs has the potential for arbitrary modeling and deformation of geometric objects.

7.6.1 Initialization

To apply the intensity-based free-form deformation, the system needs an initial PDE working space which defines the geometry and initial intensity value distribution throughout the space. The solid geometry for the working space can be defined using initialization techniques introduced in previous sections. As for the initial intensity value distribution, it can be arbitrary intensity function defined over the parametric domain. In particular, one possible choice to initialize the intensity field is using the implicit PDE technique introduced in Chapter 6, which calculates intensity values over the working space based on given embedded datasets. Other pre-defined intensity functions or volume datasets can also be used as input for initial intensity distributions. Fig. 7.9 shows some examples. The intensity attributes can be manipulated using modeling toolkits provided by the implicit PDE module.

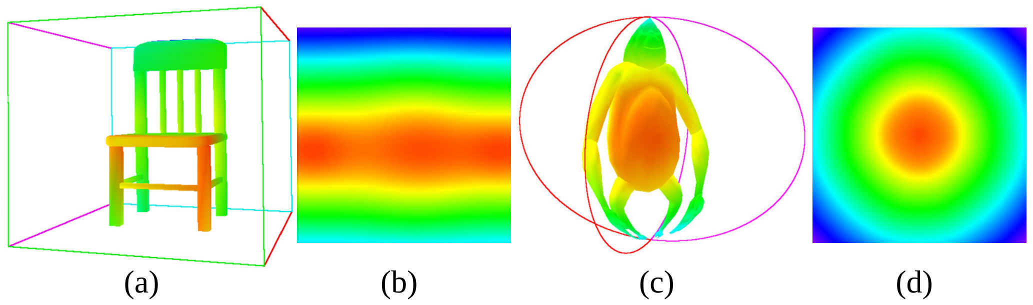


Figure 7.9: Examples of intensity initialization of PDE solids. (a) and (c) are embedded in PDE solids with color maps of intensity distribution; (b) and (d) are cross-section views from z-direction of intensity distributions in the parametric space.

7.6.2 Free-Form Modeling and Deformation with Intensity Distribution

The system provides modeling and deformation operations based on intensity values in the parametric domain to modify objects' shapes.

Iso-surface Deformation

After the initialization of intensity field for a PDE solid working space, shape deformation related to both intensity and geometry can be obtained. If the intensity distribution represents certain implicit shape, an iso-surface of this shape can be calculated using the Marching Cube method at any user specified intensity value in the parametric domain. The iso-surface can then be treated as an embedded dataset for the parametric PDE solid, and all the editing toolkits for parametric PDE solids can be employed for direct sculpting and free-form deformation of the iso-surface. It offers geometric free-form deformation and directly manipulation for implicit objects. Fig. 7.10 shows an example.

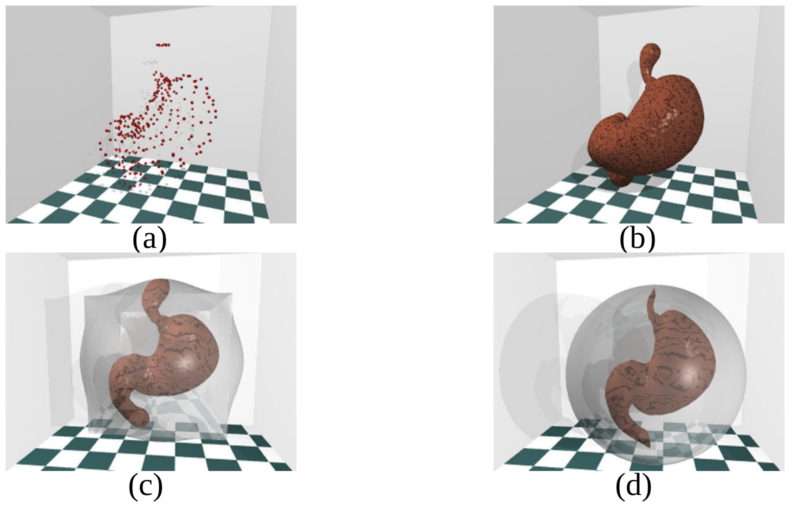


Figure 7.10: Iso-surface deformation. (a) A set of scattered points; (b) implicit iso-surface obtained from (a); (c) and (d) are deformed iso-surfaces in different PDE solids.

Free-Form Shape Blending and Deformation

Shape blending between arbitrary geometric objects are not easy for explicit models because it's hard to construct correspondence between the blending parts. However, the implicit PDE module introduced in this dissertation provides a natural way to blend implicit objects by embedding objects into the implicit PDE working space. The integration of PDE solid geometry and the implicit PDE can offer arbitrary shape blending with ease. Moreover, it can unify shape blending based on the implicit PDE and PDE-based free-form deformation to offer users more flexible shape blending and deformation for arbitrary objects.

To blend arbitrary geometric shapes, the system first constructs an embedding geometric space for each shape to be blended and calculates intensity distributions for the embedding spaces by the implicit techniques introduced in Chapter 6. Then the embedding geometric spaces and intensity distributions can be used as boundary constraints for PDE solid geometry and implicit PDE working space respectively. Solving (7.6) with these boundary constraints will provide a single geometric PDE solid working space containing shapes to be blended with a smoothly blended intensity distribution for the entire working space. The blended intensity field will provide a smoothly blended shape for input objects. With any specific iso-value, an iso-surface for the blended shape can be reconstructed accordingly. At the same time, the constructed PDE solid obtained from the geometric embedding spaces as boundary constraints provides the blended shape geometry (Fig. 7.11). Moreover, because blending operations are performed based on intensity distributions in the parametric domain, the system can provide users different blended shapes for objects embedded in deformed PDE solid working space. This allows users to obtain the shape blending and deformation at the same time. It offers more freedom of shape manipulation for objects with arbitrary topology. Refer to Fig. 7.12 for an example.

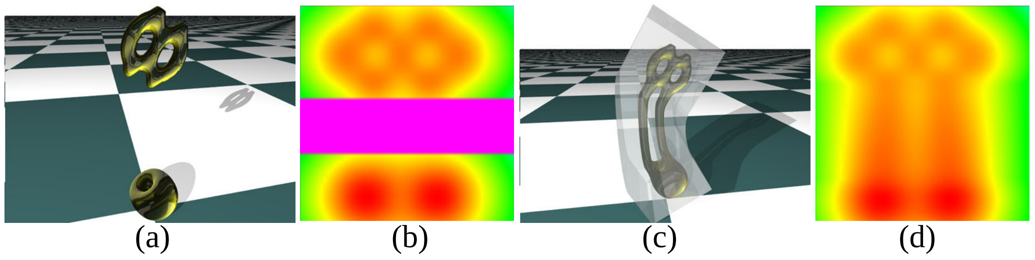


Figure 7.11: Shape blending by integrated PDE solid with intensity. (a) Two objects to be blended; (b) z-direction view of intensity field of (a); (c) the blended object embedded in the constructed PDE solid; and (d) z-direction intensity view of (c).

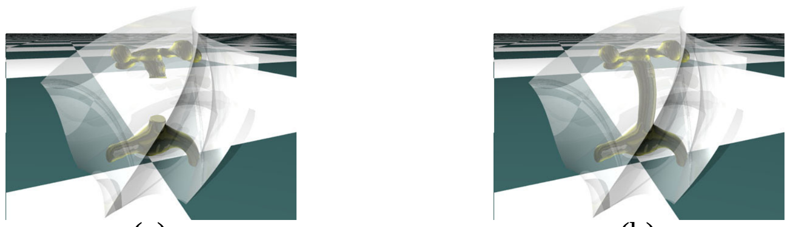

(a) (b)

Figure 7.12: PDE-based free-form shape blending and deformation. (a) is two embedded objects to be blended; (b) is the blending result.

Shape Deformation Based on Intensity Field Modification

The geometric shape of an embedded object in the PDE solid working space can be deformed by modifying intensity distributions associated with the working space. When modifying the intensity distribution in the PDE working space, intensity values of the embedded object will be changed accordingly. To preserve their original intensity values in the modified intensity field, vertices on the object will follow the intensity modifications to new locations in the working space, which will cause deformation of the object's geometric shape. The PDE-based free-form deformation techniques allow users to interactively change intensity values of the working space through the sculpting toolkits of implicit PDE models in order to deform the object. This can be done as follows. First, after initializing the intensity field in the PDE working space, the intensity distribution of an embedded dataset is obtained. Second, users can modify the distribution of the working space, and new intensity values of the dataset as well as the gradient information are recalculated according to the modified intensity field. Third, to preserve original intensity values of the dataset, vertices on the dataset are allowed to move along their intensity gradient directions to locations with their original intensity values in the modified intensity field. As results, the geometric shape of the embedded object is deformed. The system allows users to change the intensity distribution of the working space locally in selected regions and keep intensity values of other parts untouched. This will provide local deformation of the object. The initial intensity distribution of the working space can be either constructed from the embedded dataset using the implicit PDE technique, or obtained from an arbitrary implicit function to offer users more degrees of freedom for shape deformation. Fig. 7.13 and Fig. 7.14 have corresponding shape deformation examples of these two cases. In Fig. 7.14, the implicit function has the form:

$$d(x,y,z) = e^{-(\sigma_x(x-x_0)^2+\sigma_y(y-y_0)^2+\sigma_z(z-z_0)^2)}.$$

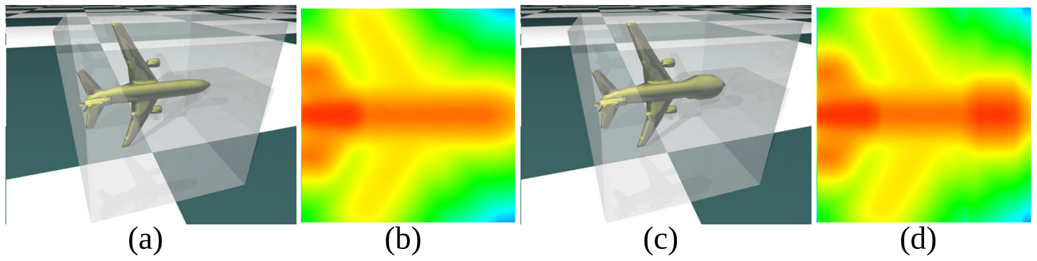


Figure 7.13: PDE-based free-form deformation due to intensity changes. (a) An embedded shape; (b) z-direction intensity view; (c) the deformed object in the locally modified intensity field; and (d) the corresponding z-direction intensity view of (c).

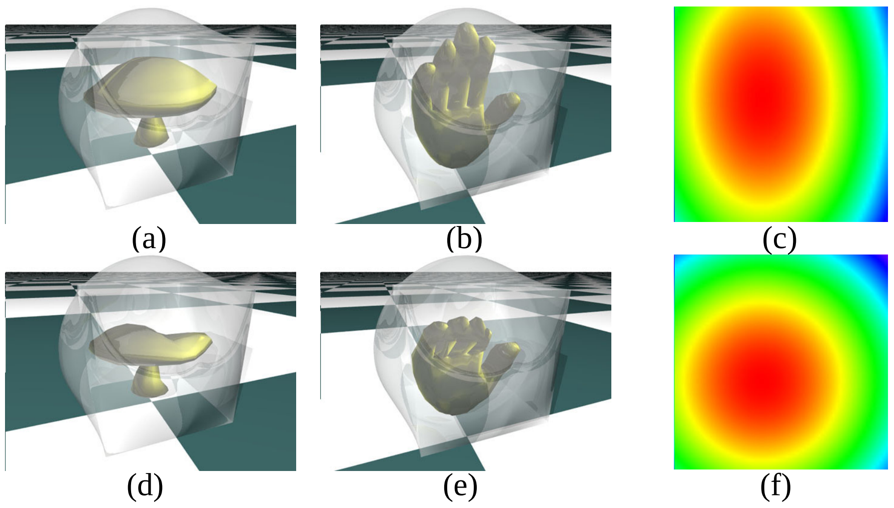


Figure 7.14: PDE-based free-form deformation due to intensity changes. (a) and (b) are embedded objects; (c) z-direction view of intensity distribution obtained from function $d(x, y, z) = e^{-(\sigma_x(x-x_0)^2+\sigma_y(y-y_0)^2+\sigma_z(z-z_0)^2)}$; (d) and (e) are deformed objects when modifying $d(x, y, z)$ by changing σ_x, σ_y, x_0, and y_0 shown in (f).

Chapter 8

Numerical Techniques for PDEs

There are various methods to solve PDEs, including analytic solutions and numerical approximations. Although there are many advantages of exact analytic solutions for PDEs, in most of occasions, they don't exist due to arbitrary boundary or initial conditions. In contrast, numerical techniques can guarantee an approximate solution at user specified accuracy in such situations. Previous work of geometric PDE modeling mainly seeks for closed-form analytic solution (*e.g.*, Fourier series functions) in order to explore certain attractive properties of analytic formulations for shape modeling. However, such methods have difficulties to handle arbitrary boundary/initial constraints. Although there are spectral approximation methods to approximate solutions for PDE objects with general boundary conditions, it's difficult to enforce additional constraints with special feature design criteria and modeling requirements which can be easily obtained using numerical discretization methods. Numerical algorithms also facilitate the material modeling of anisotropic distribution and its realistic physical simulation, where there are no closed-form analytic solutions for dynamic PDE surfaces/solids. The popular numerical methods to solve PDEs include finite-element method and finite-difference method.

8.1 Spectral Approximations Based on Fourier Transforms

Bloor and Wilson [17] introduced an approximate method whereby approximate solutions to generate PDEs satisfying associated boundary conditions may be efficiently calculated in an explicit form. The solutions are expressed in terms of a finite sum of analytic functions (*i.e.*, closed-form solutions) which individually satisfy the PDE but not boundary conditions, to which is added a *corrector* or *remainder* term. The solution can be regarded as a spectral approximation to the solution of the chosen PDE, in which it represents the solution in terms of globally defined functions which are infinitely differentiable.

They considered the solution of (2.5) over the u, v domain $\Omega : [0,1] \times [0, 2\pi]$, where v varies $0 \rightarrow 2\pi$, subject to periodic boundary conditions in the v direction, *i.e.*, topologically, the surface is like a closed *band* with the $u = 0$ and $u = 1$ isolines forming the boundary curves for the patch. Given boundary conditions in the form of (2.6), by using the method of separation of variables, the solution of (2.5) may be written as

$$\mathbf{X}(u,v) = \mathbf{A}_0(u) + \sum_{n=1}^{N} [\mathbf{A}_n(u) cos(nv) + \mathbf{B}_n(u) sin(nv)], \tag{8.1}$$

where the *coefficient* function $\mathbf{A}_n(u)$ and $\mathbf{B}_n(u)$ are of the form

$$\begin{aligned} \mathbf{A}_0(u) &= \mathbf{a}_{00} + \mathbf{a}_{01}u + \mathbf{a}_{02}u^2 + \mathbf{a}_{03}u^3, & (8.2) \\ \mathbf{A}_n(u) &= \mathbf{a}_{n1}e^{anu} + \mathbf{a}_{n2}ue^{anu} + \mathbf{a}_{n3}u^2e^{-anu} + \mathbf{a}_{n4}u^3e^{-anu}, & (8.3) \\ \mathbf{B}_n(u) &= \mathbf{b}_{n1}e^{anu} + \mathbf{b}_{n2}ue^{anu} + \mathbf{b}_{n3}u^2e^{-anu} + \mathbf{b}_{n4}u^3e^{-anu}, & (8.4) \end{aligned}$$

where $\mathbf{a}_{n1}$, $\mathbf{a}_{n2}$, $\mathbf{a}_{n3}$, $\mathbf{a}_{n4}$, $\mathbf{b}_{n1}$, $\mathbf{b}_{n2}$, $\mathbf{b}_{n3}$, $\mathbf{b}_{n4}$ are constant vectors. Note that, each of the three types of the term, $\mathbf{A}_0(u)$, $\mathbf{A}_n(u)cos(nv)$, and $\mathbf{B}_n(u)sin(nv)$ in (8.1), satisfies (2.5).

In order to determine the various constants in the solution, they transformed the boundary conditions (2.6) into finite Fourier series and identified the Fourier coefficients with values of $\mathbf{A}_n(u)$ and $\mathbf{B}_n(u)$ as well as their derivatives with respect to u at $u = 0$ and $u = 1$.

However, if boundary conditions are not expressible as finite Fourier series, the solution in the form of (8.1) will be infinite series ($N = \infty$) and not suitable for practical use. Therefore, Bloor and Wilson proposed the spectral approximation method to deal with more general boundary conditions. The basic idea is to approximate the solution by:

$$\mathbf{X}(u,v) = \mathbf{F}(u,v) + \mathbf{R}(u,v), \tag{8.5}$$

$$\mathbf{F}(u,v) = \mathbf{A}_0(u) + \sum_{n=1}^{N}[\mathbf{A}_n(u)cos(nv) + \mathbf{B}_n(u)sin(nv)], \tag{8.6}$$

where N is finite (usually $N = 5$), $\mathbf{R}(u,v)$ is a *remainder* term, which represents the contribution of high frequency modes to the surface and is negligible over most of the patch if N is large enough. $\mathbf{R}(u,v)$ is chosen to be:

$$\mathbf{R}(u,v) = \mathbf{r}_1(v)e^{wu} + \mathbf{r}_2(v)ue^{wu} + \mathbf{r}_3(v)e^{-wu} + \mathbf{r}_4(v)ue^{-wu}, \tag{8.7}$$

where coefficient functions $\mathbf{r}_1(v)$, $\mathbf{r}_2(v)$, $\mathbf{r}_3(v)$, and $\mathbf{r}_4(v)$ are determined by the difference of $\mathbf{F}(u,v)$ and its derivatives with respect to u at boundaries $u = 0$ and $u = 1$ with boundary conditions (2.6).

This method can provide very fast approximate solutions in real time and is more suitable for general boundary conditions. However, it only provides global control and couldn't handle arbitrary additional constraints.

8.2 Finite-Element Method

The finite-element method was originally introduced in the 1950's as a method to calculate elastic deformations in solids. Later the method has been developed and generalized for all kinds of PDEs. It is the dominating technique for solid-mechanics problems such as estimating stresses and strains in elastic material under prescribed loads. Finite-element methods are also commonly applied to other areas, such as calculations of electromagnetic fields and fluid flows.

The principle of the method is to replace an entire continuous domain by a number of subdomains in which the unknown function is represented by simple interpolation functions with unknown coefficients. Thus, the original problem with an infinite number of degrees of freedom is converted into a problem with a finite number of degrees of freedom, or in other words, the solution of the whole system is approximated by a finite number of unknown coefficients. Then a set of algebraic equations or a system of equations is obtained, and solution of the boundary-value problem is achieved by solving the equation system. Therefore, a finite-element analysis should include the following basic steps:

1. Discretization or subdivision of the domain;
2. Selection of the interpolation functions;
3. Formulation of the system of equations;
4. Solution of the system of equations.

Domain Discretization The discretization of the domain, say Ω, is the first and perhaps the most important step in any finite-element analysis because the manner in which the domain is discretized will affect the computer storage requirements, the computation time, and the accuracy of the numerical results. In this step, the entire domain Ω is subdivided into a number of small domains, denoted as $\Omega^e(e = 1, 2, 3, \cdots, M)$, where M stands for the total number of subdomains. These subdomains are usually called the *elements*. For a one-dimensional domain which is actually a straight or curved line, the elements are often short line segments interconnected to form the original line (Fig. 8.1(a)). For a two-dimensional domain, the elements are usually small triangles or rectangles (Fig. 8.1(b)). The rectangular elements are best suited for discretizing rectangular regions, while the triangular ones can be used for irregular regions. In a three-dimensional solution, the domain may be subdivided into tetrahedral, triangular prisms, or rectangular bricks (Fig. 8.1(c)), among which the tetrahedra are the simplest and best suited for arbitrary volumetric domains. Note that the linear line segments, triangles, and tetrahedra are the basic one-, two-, and three-dimensional elements, respectively.

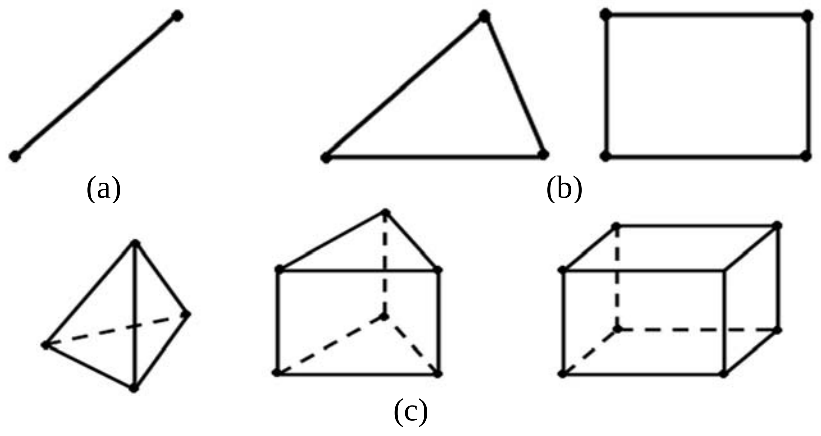


Figure 8.1: Basic finite elements. (a) One-dimensional; (b) two-dimensional; and (c) three-dimensional.

In most finite-element solutions, the problem is formulated in terms of the unknown function ϕ at nodes associated with the elements. For example, a linear line element has two nodes, one at each endpoint. A linear triangular element has three nodes, located at its three vertices, whereas a linear tetrahedron has four nodes, located at its four corners. For implementation purposes, it is necessary to describe these nodes. A complete description of a node contains its coordinate values, local number of the node that indicates its position in the element, and the global number specifies its position in the entire system.

The discretization of the domain is usually considered a pre-processing task because it can be completely separated from the other steps. Many well-developed finite-element program packages have the capability of subdividing an arbitrarily shaped line, surface, and volume into the corresponding elements and also provided the optimized global numbering.

Selection of Interpolation Functions The second step of a finite-element analysis is to select an interpolation function that provides an approximation of the unknown solution within an element. The interpolation is usually selected to be a polynomial of first (linear), second (quadratic), or higher order. Higher-order polynomials, although more accurate, usually result in a more complex formulation. Hence, the simple and basic linear interpolation is still widely used. Once the order of the polynomial is selected, one can derive an expression for the unknown solution in an element, say element e, in the following form:

$$\tilde{\phi}_e = \sum_{j=1}^{n} f_j^e \phi_j^e, \tag{8.8}$$

where n is the number of nodes in the element, ϕ_j^e is the value of ϕ at node j of the element, and f_j^e is the interpolation function. The highest order of f_j^e is referred to as the order of the element; for example, if f_j^e is a linear function, the element e is a linear element. An important feature of the functions f_j^e is that they are nonzero only within element e, and outside this element they vanish. The collection of elements which contain a specific node j forms a patch around node j. Within the patch, the nodal basis function f_j is defined piecewisely by f_j^e over each element belonging to the patch. Outside the patch, f_j is 0. One can define an approximate or trial solution $\Phi(\mathbf{x})$:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{m} f_i \Phi_i, \tag{8.9}$$

where $\mathbf{x} = (x_1, ..., x_l)$ and m is the number of nodes comprising the entire domain Ω.

Formulation of the System of Equations The third step, also a major step in a finite-element analysis, is to formulate the system of equations. In general there are two methods can be used for this purpose, *Ritz method* and *Galerkin method*. The Ritz method, also known as the Rayleigh-Ritz method, is a variational method in which the boundary-value problem is formulated in terms of a variational expression, called functional, whose minimum corresponds to the governing differential equation under given boundary conditions. The approximate solution is then obtained by minimizing the functional with respect to its variables. Galerkins method belongs to the family of weighted residual methods, which seek the solution by weighting the residual of the differential equation. In Galerkins method, the weighting function is selected to be the same as those used for the expansion of the approximate solution. This usually leads to the most accurate solution and is, therefore, a popular approach in developing the finite-element equations.

This section briefly introduces the Galerkin's method as an example: for a general PDE $\phi(\mathbf{x}) = b$, a weak form of the solution satisfies

$$\int_\Omega \phi(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} = \int_\Omega b v(\mathbf{x}) d\mathbf{x}, \tag{8.10}$$

where $v(\mathbf{x})$ is a weight function to minimize the residual $\phi(\mathbf{x}) - b$ in a *weighted* sense and each choice of v gives an equation.

Then the m unknown numbers of Φ_i in (8.9) can be determined using m different weight functions v to obtain m equations. The weight function can be set the same as the basis functions f_i. Then there is a Galerkin equation for each $v = f_i$. Substituting into (8.10), a set of linear equations can be obtained in a matrix form

$$\mathbf{K}\Phi = \mathbf{B}. \tag{8.11}$$

Solution of the Equation System The (8.11) can be solved by standard methods for linear equation systems. Details of the finite-element method can be found in [136].

8.3 Finite-Difference Method

The finite-difference method divides the 2D or 3D parametric space into discrete grids along parametric directions and transforms a continuous PDE into a set of simultaneous algebraic equations by sampling the partial derivatives in differential equations for each grid point with their discretized approximation. The algebraic equation system can then be solved numerically either through a direct procedure or an iterative process for an approximate solution of the continuous PDE.

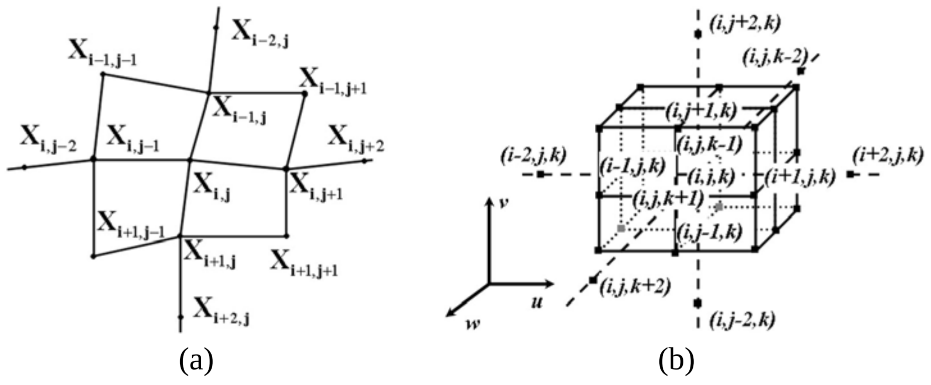


Figure 8.2: The point discretization for finite-difference method. (a) Discretization for a 2D surface; (b) discretization for 3D working space.

Based on Taylor's expansion of polynomial functions, the first-order and second-order derivatives of a univariate function can be approximated using the central-difference approximation $f'(x) = (f(x+h) - f(x-h))/2h$, $f''(x) = [f(x+h) - 2f(x) + f(x-h)]/h^2$, where h denotes the spatial interval between neighboring discrete sample points. This can be generalized to all partial derivatives on bivariate and trivariate PDEs, by dividing the parametric domain into a number of discretized grids. The partial derivatives respect to each parametric direction can be approximated by the neighbors along that direction. For example, the 2D parametric domain for bivariate function $f(u, v)$ can be divided into m and n discretized points respectively, and the 3D $u - v - w$ domain can be discretized into l, m, and n grids. Then the bivariate function $f(u, v)$ and trivariate function $g(u, v, w)$ can be represented by their values at the discrete set of points:

$$\begin{aligned} u_i &= i\Delta u \quad i = 0, 1, ..., l-1 \\ v_j &= j\Delta v \quad j = 0, 1, ..., m-1 \end{aligned}$$

and

$$\begin{aligned} u_i &= i\Delta u \quad i = 0, 1, ..., l-1 \\ v_j &= j\Delta v \quad j = 0, 1, ..., m-1 \\ w_k &= k\Delta w \quad k = 0, 1, ..., n-1 \end{aligned}$$

where $\Delta u, \Delta v, \Delta w$ are the *grid spacing* along u, v, w directions. $f_{i,j}$ is used for $f(u_i, v_j)$ and $g_{i,j,k}$ is used for $g(u_i, v_j, w_k)$ (Fig. 8.2) for sake of simplicity.

For the 2D parametric PDE discretization, the finite-difference approximation of the fourth-order partial derivatives at point $\{i, j\}$ can be written as:

$$\frac{\partial^4 f_{i,j}}{\partial u^4} = \frac{f_{i-2,j} + f_{i+2,j} - 4f_{i-1,j} - 4f_{i+1,j} + 6f_{i,j}}{(\Delta u)^4},$$

$$\frac{\partial^4 f_{i,j}}{\partial v^4} = \frac{f_{i,j-2} + f_{i,j+2} - 4f_{i,j-1} - 4f_{i,j+1} + 6f_{i,j}}{(\Delta v)^4},$$

$$\frac{\partial^4 f_{i,j}}{\partial u^2 \partial v^2} = \begin{array}{l} \frac{f_{i-1,j-1}+f_{i-1,j+1}+f_{i+1,j-1}+f_{i+1,j+1}+4f_{i,j}}{(\Delta u)^2(\Delta v)^2} \\ -\frac{2(f_{i-1,j}+f_{i+1,j}+f_{i,j-1}+f_{i,j+1})}{(\Delta u)^2(\Delta v)^2} \end{array},$$

Similarly, in the trivariate case, the approximate fourth-order partial derivatives at $\{i, j, k\}$ are as follows:

$$\frac{\partial^4 g_{i,j,k}}{\partial u^4} = \frac{g_{i-2,j,k} + g_{i+2,j,k} - 4g_{i-1,j,k} - 4g_{i+1,j,k} + 6g_{i,j,k}}{(\Delta u)^4},$$

$$\frac{\partial^4 g_{i,j,k}}{\partial v^4} = \frac{g_{i,j-2,k} + g_{i,j+2,k} - 4g_{i,j-1,k} - 4g_{i,j+1,k} + 6g_{i,j,k}}{(\Delta v)^4},$$

$$\frac{\partial^4 g_{i,j,k}}{\partial w^4} = \frac{g_{i,j,k-2} + g_{i,j,k+2} - 4g_{i,j,k-1} - 4g_{i,j,k+1} + 6g_{i,j,k}}{(\Delta w)^4},$$

$$\frac{\partial^4 g_{i,j,k}}{\partial u^2 \partial v^2} = \begin{array}{l} \frac{g_{i-1,j-1,k}+g_{i-1,j+1,k}+g_{i+1,j-1,k}+g_{i+1,j+1,k}+4g_{i,j,k}}{(\Delta u)^2(\Delta v)^2} \\ -\frac{2(g_{i-1,j,k}+g_{i+1,j,k}+g_{i,j-1,k}+g_{i,j+1,k})}{(\Delta u)^2(\Delta v)^2} \end{array},$$

$$\frac{\partial^4 g_{i,j,k}}{\partial u^2 \partial w^2} = \begin{array}{l} \frac{g_{i-1,j,k-1}+g_{i-1,j,k+1}+g_{i+1,j,k-1}+g_{i+1,j,k+1}+4g_{i,j,k}}{(\Delta u)^2(\Delta w)^2} \\ -\frac{2(g_{i-1,j,k}+g_{i+1,j,k}+g_{i,j,k-1}+g_{i,j,k+1})}{(\Delta u)^2(\Delta w)^2} \end{array},$$

$$\frac{\partial^4 g_{i,j,k}}{\partial v^2 \partial w^2} = \begin{array}{l} \frac{g_{i,j-1,k-1}+g_{i,j-1,k+1}+g_{i,j+1,k-1}+g_{i,j+1,k+1}+4g_{i,j,k}}{(\Delta v)^2(\Delta w)^2} \\ -\frac{2(g_{i,j-1,k}+g_{i,j+1,k}+g_{i,j,k-1}+g_{i,j,k+1})}{(\Delta v)^2(\Delta w)^2} \end{array}.$$

Fig. 8.3 gives an illustration for the coefficients of the difference equation at the point $\{i, j\}$ for the 2D fourth-order elliptic PDE (2.5) with uniform grid spacing along both u and v directions and blending coefficient $a = 1$.

8.4 Iterative Method

The approximate difference equations form an algebraic equation system that can be easily solved by either direct methods or iterative methods and suitable for parallel computing. For fine resolution of domain discretization, the number of difference equations will increase dramatically, which indicates the iterative

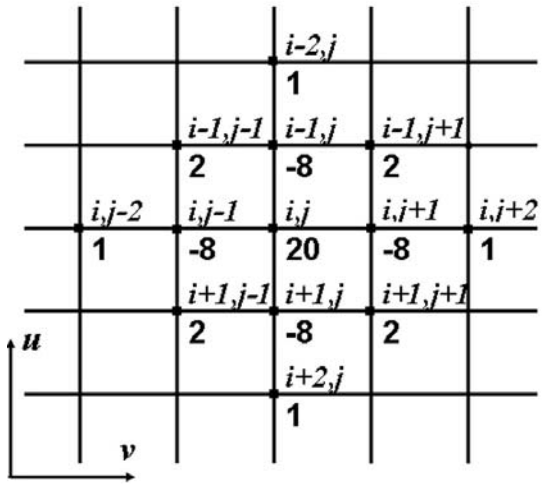


Figure 8.3: The illustration of discretizing differential operator for i, j.

solvers are more realistic choices than direct methods for algebraic equation system.

The iterative methods make use of the structure of the sparse matrix on the left-hand side of the finite-difference equation system. Given an example

$$\mathbf{AX} = \mathbf{b}, \tag{8.12}$$

the matrix $\mathbf{A}$ is split into two parts

$$\mathbf{A} = \mathbf{A}_d - \mathbf{A}_r, \tag{8.13}$$

where $\mathbf{A}_d$ consists of the diagonal elements of $\mathbf{A}$ and zeros everywhere else, $\mathbf{A}_r$ is the remainder. Then (8.12) becomes

$$\mathbf{A}_d\mathbf{X} = \mathbf{A}_r\mathbf{X} + \mathbf{b}. \tag{8.14}$$

The iterative methods start from choosing an initial guess $\mathbf{X}^{(0)}$ and then solving the equations successively by iterating $\mathbf{X}^{(s)}$ from

$$\mathbf{A}_d\mathbf{X}^{(s)} = \mathbf{A}_r\mathbf{X}^{(s-1)} + \mathbf{b}. \tag{8.15}$$

Given boundary conditions of certain PDEs for boundary value problems, one can compute the initial guess by simple linear interpolation based on the constraints. The iteration will stop at $\mathbf{X}^{(s)}$ for an approximate solution when the difference between $\mathbf{X}^{(s)}$ and $\mathbf{X}^{(s-1)}$ is less than a threshold. Certain variants of iterative techniques exist for solving the aforementioned linear equations [136]. This dissertation employs the Gauss-Seidel iteration which uses the updated value of the iteration result at a grid point on the right-hand side of (8.15) as soon as it becomes available. To further speed up the converging rate of Gauss-Seidel iteration, the error factor characterized by the difference between the approximation and the real solution is considered. This leads to the method of Successive Over-Relaxation (SOR) iteration.

8.5 Multi-Grid Method

The multi-grid method will speed up the convergent rate when solving the linear elliptic equation systems. Suppose the goal is to solve the linear elliptic problem

$$Lu = f, \tag{8.16}$$

where L is some linear elliptic operator and f is the source term. Discretizing (8.16) on a uniform grid with mesh size h, the resulting set of linear algebraic equations is of the form

$$L_h u_h = f_h \tag{8.17}$$

Let $\tilde{u}_h$ denote some approximate solution to (8.17) and u_h denote the exact solution. Then the *error* in $\tilde{u}_h$ or the *correction* v_h is $v_h = u_h - \tilde{u}_h$, and the *residual* or *defect* d_h is $d_h = L_h\tilde{u}_h - f_h$. Since L_h is linear, the error satisfies

$$L_h v_h = -d_h. \tag{8.18}$$

At this point it's necessary to make an approximation to L_h in order to find v_h, say $\hat{v}_h$. The next approximation is generated by

$$\tilde{u}_h^{new} = \tilde{u}_h + \hat{v}_h.$$

Approximation L_H of L_h can be formed on a coarser grid with mesh size H (*e.g.*, $H = 2h$). Then (8.18) is approximated by

$$L_H v_H = -d_H, \tag{8.19}$$

which will be easier to solve due to the smaller size of L_H. To define the defect d_H on the coarse grid, a *restriction operator* R that $d_H = Rd_h$ is defined. Once a solution $\tilde{v}_H$ to (8.19) is obtained, a *prolongation operator* P is needed to prolongate or interpolate the correction to the fine grid, $\tilde{v}_h = P\tilde{v}_H$. Finally the approximation $\tilde{u}_h$ can be updated:

$$\tilde{u}_h^{new} = \tilde{u}_h + \tilde{v}_h. \tag{8.20}$$

This scheme is called *coarse-grid correction*. The multi-grid method starts pre-smoothing by approximating $\tilde{u}_h$ at finest grid, then applies coarse-grid correction on coarser grid recursively, and performs post-smoothing by computing $\tilde{u}_h^{new}$ to finest grid again.

8.6 Comparison and Discussion

The techniques to solve PDEs introduced in this chapter, including spectral approximation method, finite-element method, and finite-difference method, have their own advantages and shortcomings. The spectral approximation methods can provide fast solutions for parametric PDEs. However, they are focusing on boundary conditions and global features of the entire parametric domain, while local modifications inside the domain cannot be enforced. Thus, they are not suitable for interactive manipulations of geometric PDE objects. The finite-element method and finite-difference method can deal with additional constraints easily for interactive shape design and sculpting. The finite-element method can provide numerical approximations for geometric PDEs defined on irregular domain, however, it's not as simple and easy to implement as the finite-difference method. In addition, although traditionally the finite-difference method is only applicable to regular parametric domain, it can be used in irregular space with the help of certain parametrization techniques.

Because the finite-difference method is simple to implement and suitable to most of the PDEs employed in the PDE-based modeling system, This dissertation employs this method with uniform discretizations for different application modules to offer interactions among these modules. With the finite-difference approach, flexible boundary conditions and additional hard constraints can be easily enforced for direct manipulations. Iterative methods are employed to provide fast approximations under local additional constraints. A simplified version of the multi-grid method is also used by starting at the coarsest grid to solve the difference equations by iterative method, and refine the solution to finer grids until it reaches the user-defined finest resolution. This simplified multi-grid version is quite easy to implement and guarantees approximate solution under various constraints with improved convergent rate of the iterative solvers.

Chapter 9

System Architecture and Results

This dissertation presents an integrated PDE modeling framework for parametric surfaces and solids, implicit shapes, and arbitrary polygonal mesh objects. The PDE-based prototype system offers interactive sculpting and direct manipulations for the PDE-governed objects with a set of global and local deformation toolkits. This chapter will outline the system structure of the PDE-based modeling system and discuss experimental results. The system is written in Visual C++ and runs on Windows98/2000/XP. It provides output datasets for POV-RAY rendering system. The mesh objects used in examples of this dissertation are provided by 3DCafe. By the representation types of models, the system consists of four modules to model parametric surface, PDE-governed arbitrary polygonal meshes, implicit PDE objects, and free-form PDE solids with intensity distributions, respectively. These modules may have interactions with each other through data exchanges. Fig. 9.1 shows the structure and functionalities of the prototype PDE-based modeling system with the relations of these modules.

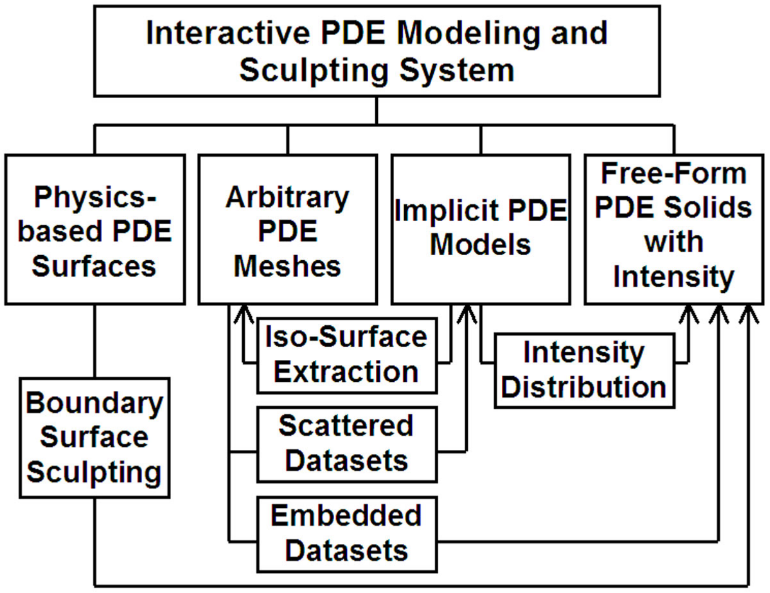

Interactive PDE Modeling and Sculpting System
Physics-based PDE Surfaces
Arbitrary PDE Meshes
Implicit PDE Models
Free-Form PDE Solids with Intensity
Iso-Surface Extraction
Intensity Distribution
Boundary Surface Sculpting
Scattered Datasets
Embedded Datasets

Figure 9.1: System architecture for PDE-based modeling system.

9.1 Physics-based PDE Surface Module

9.1.1 Overview

This dissertation presents a modeling framework for design and manipulation of dynamic PDE surfaces. Fig. 9.2 shows a snapshot of the PDE-based modeling system while manipulating a selected point on a PDE surface. When modeling PDE surfaces, the modeling toolkits are under the ”Surface_Model” in the menu bar, including surface connection, selection of surface types and boundary types in the Surface Initialization, direct manipulation, B-spline approximation, as well as surface displacement sculpting. The left window in the interface is the 3D view window for PDE surfaces, which can be displayed in wireframe, shaded surface, and curvature mapping. The window on the upper right can be switched between X-Y view and U-V discretization grids in parametric domain, and the selected point on the grid is displayed in red. The window on the lower right can

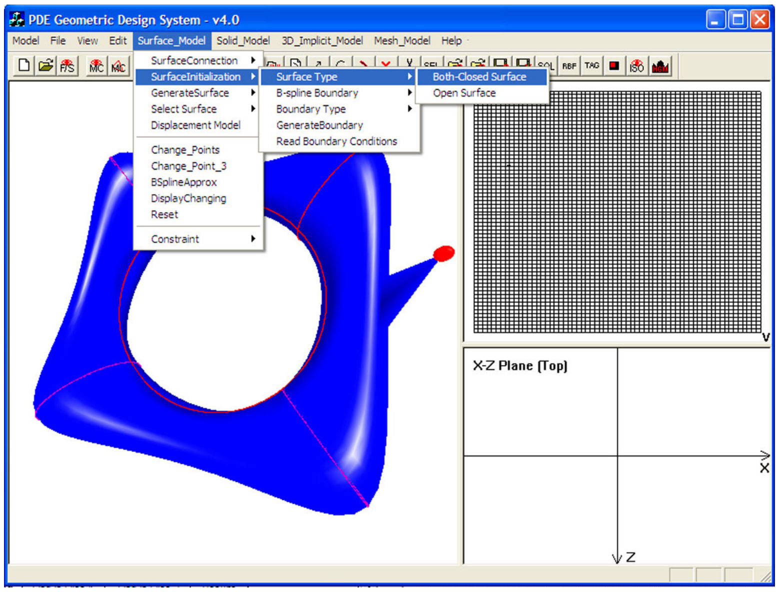


Figure 9.2: System interface for the physics-based PDE surface module.

be switched between X-Z view and curvature mapping color index.

The physics-based PDE surface module in the PDE modeling system permits users to interactively manipulate PDE surfaces/displacements with various constraints either locally or globally. Fig. 9.3 shows the structure of the PDE surface module for physics-based parametric surfaces.

9.1.2 Dynamic PDE Surface Modeling Functionalities

The physics-based PDE surface module provides the following functionalities:

Boundary Conditions. Users can interactively input and edit several types of boundary conditions defined by cubic B-spline curves or commonly-used analytic functions, and obtain PDE surfaces satisfying these constraints. Boundary conditions can be modified freely as curve-based constraints. Moreover, the system offers a multi-grid scheme to improve the time performance when modeling PDE surfaces.

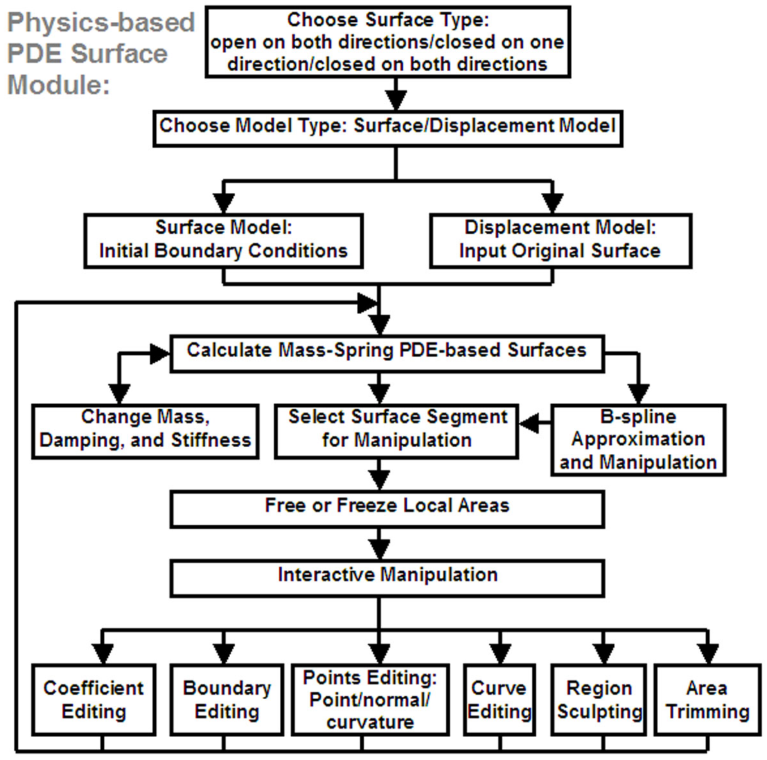

Physics-based PDE Surface Module:
Choose Surface Type: open on both directions/closed on one direction/closed on both directions
Choose Model Type: Surface/Displacement Model
Surface Model: Initial Boundary Conditions
Displacement Model: Input Original Surface
Calculate Mass-Spring PDE-based Surfaces
Change Mass, Damping, and Stiffness
Select Surface Segment for Manipulation
B-spline Approximation and Manipulation
Free or Freeze Local Areas
Interactive Manipulation
Coefficient Editing
Boundary Editing
Points Editing: Point/normal/ curvature
Curve Editing
Region Sculpting
Area Trimming

Figure 9.3: System architecture of the physics-based PDE surface module.

Displacement Models. To further broaden the applications of PDE method on parametric surfaces, this dissertation extends the PDE formulation to model surface displacements, which represent the offsets over the original surface. It enables the proposed PDE technique to model a large set of surfaces of flexible topology. The system will switch to model surface displacements by selecting ”Displacement Model” in the ”Surface_Model” menu. Displacements can be manipulated using the interactive sculpting toolkits for the PDE surface.

Dynamic Models. The PDE-based modeling system supports novel physics-based PDE surface manipulation techniques including: (1) finite-difference discretization for mass-spring models; (2) multi-grid subdivision for model refinement; and (3) finite-element approximation using B-splines for dynamic surfaces.

Material properties and dynamic behavior can greatly enhance the interactive manipulation of conventional PDE surfaces.

Sculpting Tools. The system provides various manipulation toolkits to offer users the capability of intuitive and interactive sculpting of physics-based PDE surfaces/displacements. These toolkits include: (1) patching several PDE surfaces smoothly; (2) moving (a set of) arbitrary surface points to desired locations; (3) modifying surface normals at arbitrary points; (4) editing surface curvatures of arbitrary surface points; (5) changing boundary conditions; (6) modifying the blending coefficient function (*i.e.*, $a(u, v)$) associated with the PDE; (7) specifying and enforcing a set of curve constraints; (8) deforming a set of user-specified regions to desired shapes; (9) freezing any local region(s); (10) applying local operations only on user-selected areas; (11) trimming specified parts of the surface; (12) direct manipulation on surface displacements; (13) modifying material properties such as mass, damping, and stiffness distributions locally; (14) computing the B-spline approximation of PDE surfaces; and (15) directly deforming B-spline finite elements with *forces*.

9.1.3 System Performance

Several numerical techniques are employed to solve the PDE surface subject to various constraints. Table 9.1 summarizes the CPU time of three different numerical solvers on a PDE surface discretized at different resolutions. Gaussian-Elimination represents Gaussian Elimination method, Gauss-Seidel stands for Gauss-Seidel iteration, and SOR stands for SOR iteration. The iteration threshold (0.001) for the two iterative methods is the sum of the distance between the same point in successive steps.

Table 9.1 indicates that it is generally very time consuming to solve the PDE

Grids	Gaussian-Elimination	Gauss-Seidel	SOR(σ=1.25)
15×15	0.094	1	0.593
30×30	3.187	15.016	6
60×60	122.156	64.719	34.391

Table 9.1: CPU time (in seconds) of different solvers for a PDE surface with different sampling rates.

surface of large sampling grids using direct methods such as Gaussian Elimination method. The multi-grid subdivision method will allow us to start from a coarse approximation of the surface at a low discretizing resolution, then apply subdivision refinement to the coarse sampling level in order to obtain the initial guess for a finer resolution. The PDE surface at the finer resolution can be approximated accordingly. The total CPU time of using this method is much less than directly solving the equation on same discretizing resolutions.

Grids	Gauss-Seidel	SOR(σ=1.05)	SOR(σ=1.15)	SOR(σ=1.25)
15×15	1	109	0.437	0.593
30×30	0.266	0.079	171	0.282
60×60	1.1	0.156	0.344	0.5

Table 9.2: CPU time (in seconds) of different iterative methods combined with multi-grid subdivision for a PDE surface.

Table 9.2 compares the CPU time of using Gauss-Seidel Iteration and SOR Iteration with the multi-grid subdivision. The system starts solving the surface at the sampling grid 15×15. The total CPU time to obtain a solution through the iterative approaches at selected grid is the sum of the numbers from the coarsest level to the current level in the same column. The last three columns record time performance of SOR iteration with different values of σ. The choice of σ will also influence the convergent speed of the SOR iterative solver.

Grids	Gauss-Seidel	SOR	Point	Normal	Curvature	Curve	Patch
15×15	1438	756	253	146	447	497	102
30×30	1751	838	420	146	494	1681	457
60×60	4000	1583	933	146	190	3504	2000

Table 9.3: Number of iterations for various manipulation techniques with different sampling grids. The threshold (0.001) is the sum of all distance between the corresponding points in successive steps.

Table 9.3 details PDE surface examples and their performance under the two iterative methods. Point, Normal and Curvature stand for the point, normal, and curvature manipulations, respectively. Curve denotes curve editing with 20 sampling points, while Patch stands for the regional manipulation of 10×10 sampling points attached to a B-spline patch. Note that, it generally takes more iterations on a coarse sampling grid, however, the CPU time spent on the coarser grid is far less than that on the finer grid.

Surface	B-spline	μ_1	γ_1	μ_2	γ_2	ρ	Δt
Fig. 4.18(a)	N/A	20	70	70	20	100	0.1
Fig. 4.18(b)	9×9	20	70	70	20	100	0.1

Table 9.4: The parameters of physical properties on dynamic surface examples.

Table 9.4 summarizes the physical parameters used in examples of the mass-spring model as well as the B-spline approximation for mass-spring PDE surfaces shown in Fig. 4.18. μ_i, γ_i represent the mass and damping distribution for the surface (μ_1, γ_1 for yellow parts, and μ_2, γ_2 for the pink parts). ρ represents spring stiffness distribution on the surface. The sampling grids of the surfaces are 30×30.

Besides traditional boundary conditions of PDE techniques, the system allows users to specify and enforce a large variety of additional constraints on a set of points, cross-sectional curves, and surface regions. These constraints provide more freedom of control to designers, making the design process of PDE surfaces more cost-effective, natural, and intuitive. The physics-based PDE surface module is developed using both finite-difference and B-spline finite-element techniques. The advantages of these approximation techniques are that they are simple, easy to implement, and suitable for the incorporation of complicated, flexible constraints. On the other hand, the time and space complexity are increased correspondingly with higher resolution as well as increased accuracy. The convergent rate of the iteration depends on initial values. The multi-grid subdivision method for various levels of refinements achieves anticipated results in the experiments.

9.2 PDE-based Arbitrary Mesh Modeling Module

9.2.1 Overview

Fig. 9.4 is a system snapshot while manipulating vertices on an input polygonal mesh. The PDE-based mesh modeling module can be turned on by selecting Arbitrary Mesh under the ”Model” in the menu bar. The sculpting toolkits are under the ”Mesh_Model”. The mesh modeling module offers PDE-based direct manipulation of polygonal meshes and displacements and broadens the elliptic PDE applications to modeling surfaces of arbitrary topology. It also provides a diffusion-based medial axis extraction method which combines the grassfire flow simulation and diffusion propagation to approximate skeletons for objects whose boundary surfaces are polygonal meshes. It offers an alternative but natural way for medial axis extraction for commonly used 3D polygonal models. By solving

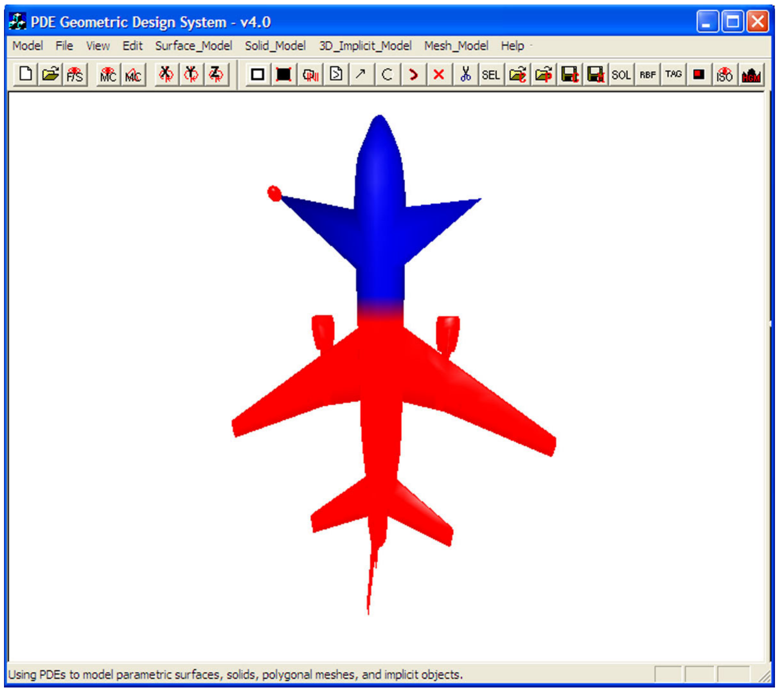

PDE Geometric Design System - v4.0
Model File View Edit Surface_Model Solid_Model 3D_Implicit_Model Mesh_Model Help
Using PDEs to model parametric surfaces, solids, polygonal meshes, and implicit objects.

Figure 9.4: System Interface for the PDE-based arbitrary mesh modeling module.

the PDE along time axis, the system can not only quickly extract diffusion-based medial axes of input meshes, but also allow users to visualize the extraction process at each time step. In addition, the module provides users a set of manipulation toolkits to sculpt extracted medial axes, then uses diffusion-based techniques to recover corresponding deformed shapes according to their original input datasets. This skeleton-based shape manipulation offers a fast and easy way for animation and deformation of complicated solid objects. Fig. 9.5 outlines this part of work in the PDE-based modeling system.

9.2.2 PDE-based Manipulation Toolkits for Arbitrary Meshes

The PDE-based arbitrary mesh modeling module provides users a set of interactive toolkits for direct shape manipulation, local/global medial-axis extraction,

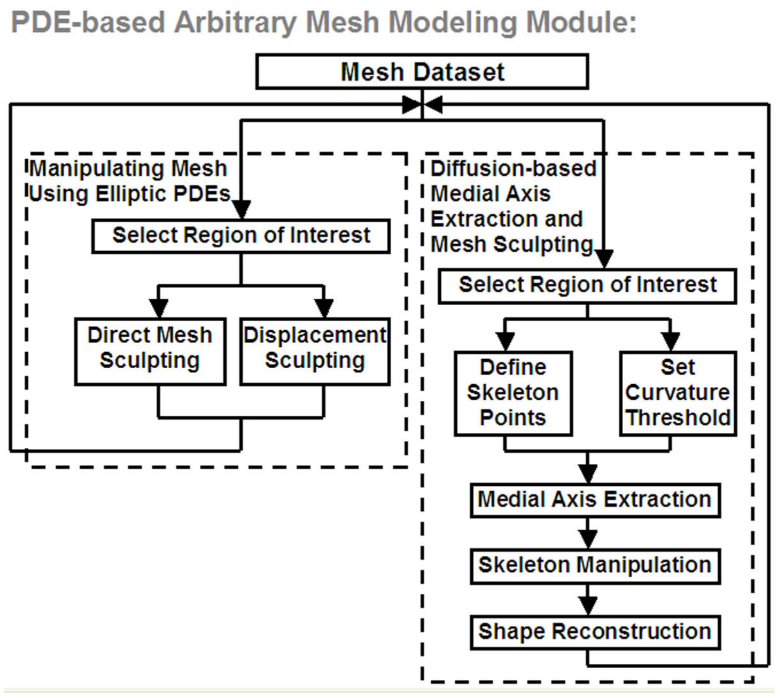

PDE-based Arbitrary Mesh Modeling Module:
Mesh Dataset
Manipulating Mesh Using Elliptic PDEs
Select Region of Interest
Direct Mesh Sculpting
Displacement Sculpting
Diffusion-based Medial Axis Extraction and Mesh Sculpting
Select Region of Interest
Define Skeleton Points
Set Curvature Threshold
Medial Axis Extraction
Skeleton Manipulation
Shape Reconstruction

Figure 9.5: System architecture for PDE-based arbitrary mesh modeling module.

and skeleton-based shape manipulation and recovery, etc. The summarized system functionalities are listed as follows: **Direct Manipulation of Mesh Surfaces and Displacements.** The system uses umbrella operator to approximate partial derivatives in elliptic PDEs and the interactive deformation of polygonal meshes can be governed by the PDEs for smooth results. Users can direct manipulate the surface for desired shape.

Progressive Diffusion-based Medial Axis Extraction. Finite-difference techniques are used to approximate the solution for the time-dependent diffusion-based equation numerically, which can provide users progressive results and visual feedback for medial axis approximation and shape reconstruction.

User Interaction for Medial Axis Extraction. During the extraction process, users can interactively select any points on the propagating surface to be skeletal points, thus they can define the user-controlled skeleton based on their own criteria. This can provide more degrees of freedom for skeleton-based shape manipulation.

Local Region Skeletonization. Users are allowed to select local regions in the 3D working space and the system will only extract skeleton for parts of the object residing in selected regions. This will reduce the time complexity for shape skeletonization of complex models and enable the mechanism for direct user control.

Curvature Manipulation. Gaussian curvature of polygonal surfaces works as the threshold for diffusion-based medial axis extraction to decide which surface points will be skeletal points. The module allows users to define the threshold themselves and obtain the medial axis for an object according to their own criteria.

Skeleton-based Shape Sculpting. The module allows users to directly manipulate the medial axis, then propagate the deformation to the original dataset according to the distance information. The shape deformation/manipulation and other processes based on medial axis alleviate the burden of tedious and less insightful operations for deforming and animating complex objects, as well as other shape queries and interrogations.

9.2.3 Discussion of Diffusion-based Medial Axis Extraction

The diffusion-based formulation naturally unifies the thinning process along surface normals with surface smoothing for the propagating fronts. It provides satisfactory results for irregular meshes. The diffusion-based technique expands the conventional notion of medial surface as it allows direct user control for shape skeletonization. In addition, the module allows users to obtain a sequence of simplified shapes satisfying different design requirements and offers shape manipulations through skeleton sculpting. The examples shown in this dissertation are provided by 3D CAFE and rendered using POV-RAY.

Example	Points	Δt	Time (seconds)
Fig. 5.1 (a)	2782	0.05	61.790
Fig. 5.1 (b)	909	0.05	17.267
Fig. 5.1 (c)	3749	0.05	136.398
Fig. 5.1 (d)	3162	0.05	252.801
Fig. 5.5	867	0.05	96.175
Fig. 5.6	855	0.05	38.098
Fig. 5.7 (a)	386	0.05	21.121
Fig. 5.8	1149	0.05	35.198
Fig. 5.9	1545	0.05	31.326
Fig. 5.10	1672	0.05	121.201
Fig. 5.11	309	0.05	15.001

Table 9.5: CPU time in seconds for medial axis extraction.

Table 9.5 summarizes the CPU time on a Pentium M 1.3GHz laptop for medial axis extraction of the examples, where ”Points” stands for the number of surface points for the dataset, Δt is the time step value used in (5.9), and ”Time (seconds)” is the CPU time for approximating the medial axis. Because the diffusion equation is discretized to approximate the medial axis, the computing time is depending on the size and complexity of the dataset as well as the time step used to solve (5.9) iteratively. In addition, the performance of the collision detection also depends on the resolution of the object.

The diffusion-based method offers smooth approximations of medial axes in a visually progressive way. For complex objects bounded by polygonal meshes, the *real* medial axes may have numerous noisy branches to preserve objects' features. Such structures are difficult to manipulate for shape sculpting. In contrast, the proposed technique provides simplified approximations for medial axes, which are smooth thin sets residing inside objects without noisy branches. The approximated results are smoothed because of the Laplacian operator, which eliminates noisy branches of the real medial axis, so that the resulting medial axis is relatively simple and easy to manipulate. In addition, different with previous techniques, this method allows user interaction during the medial axis extraction process, which provides more degrees of freedom for shape skeletonization and manipulation. For example, if users are not satisfied with the results, they can define medial axes according their own criteria by selecting skeletal points for medial axes. The approximated medial axes by user interaction along with distance information to original objects and the diffusion-based propagation technique can produce satisfactory results for sculpting and manipulating objects.

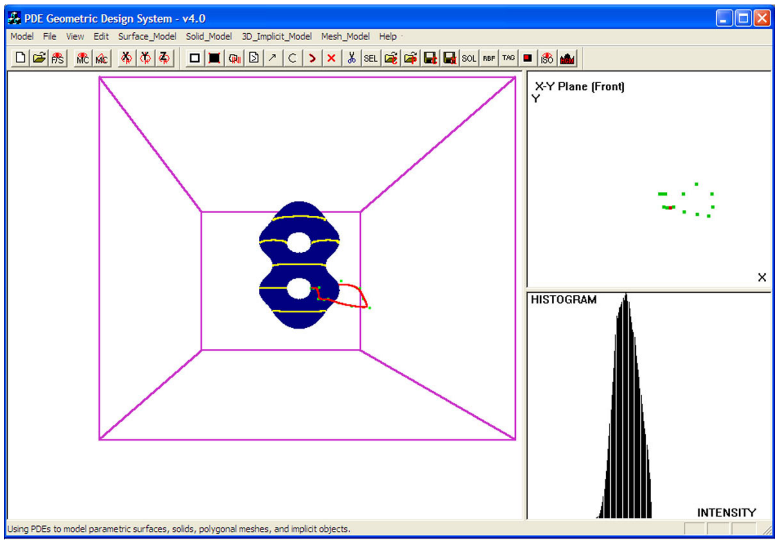
Because the proposed medial axis extraction algorithm is applied directly to arbitrary polygonal meshes, the resolution of meshes and the point distribution on meshes will affect the quality of extracted skeletons. For instance, when the two end points of a long edge on the propagating surface stop on the skeleton, all points on the edge will be assumed to be skeletal points, although there may be still spaces between them and real skeletal points. Therefore, a mesh optimization process can be considered to extract more accurate medial axes.

In addition, the approximating techniques to calculate the differential properties of the boundary surface sometimes are not accurate enough for extremely irregular meshes. On the other hand, there are techniques available to provide regular parametrization for irregular polygonal meshes. The differential calculation will be much easier under such parametrization. Thus, mesh parametrization techniques can also be applied to the diffusion-based model for better results.

A better algorithm to detect the skeletal points during the progressive medial axis extraction can also improve the process. The current algorithm employs a collision detection method that simply checks collision for sampling points and faces. Such algorithm is slow for large datasets and sensitive to the value of time intervals for the diffusion process. Faster and more accurate skeletal point detection techniques such as singularity related methods can be considered to speed up the medial axis extraction process and make this model more applicable for large models.

9.3 Direct Sculpting Environment of Implicit PDEs

9.3.1 Overview


PDE Geometric Design System - v4.0
Model File View Edit Surface_Model Solid_Model 3D_Implicit_Model Mesh_Model Help
X-Y Plane (Front)
Y
X
HISTOGRAM
INTENSITY

Using PDEs to model parametric surfaces, solids, polygonal meshes, and implicit objects.

Figure 9.6: System interface for the implicit PDE module.

This dissertation also provides a modeling framework for design, reconstruction, blending, and manipulation of implicit PDE objects. Fig. 9.6 shows a snapshot of the system while manipulating a selected sketch curve for the implicit PDE model. The implicit PDE module can be turned on by selecting Implicit Models under the ”Model” in the menu bar. The sculpting toolkits are under the ”3D_Implicit_Model”. To reduce the burden of selecting menu items during the sculpting process, the manipulation tools are associated with a set of toolbars. The functions of the toolbars include object rotation along X, Y, Z directions, Marching-Cube triangulation and view, initialization of sketch curves, input of scattered data point sets, initial guess via the RBF method and fast-tagging algorithm, iso-surface view, histogram graph, sketch curve sculpting, selection of regions of interests, etc. The left window in the interface is the 3D view window for implicit PDE objects, which offers wireframe and shaded view of iso-surface at selected intensity value, as well as discretized grid view of particular intensities. The display window on the upper right can be switched between X-Y view to modify sketch curves and 2D cutting-plane along coordinate directions for iso-contour sculpting of implicit objects. The window on the lower right can be switched between X-Z view to sculpt sketch curves and Histogram display of the entire working space for the implicit PDE module.

The system architecture of implicit PDE module is shown in Fig. 9.7.

9.3.2 Implicit PDE Modeling Toolkits

The implicit PDE module permits users to reconstruct geometric shapes defined by PDE-based implicit functions from a set of sketch curves, scattered data

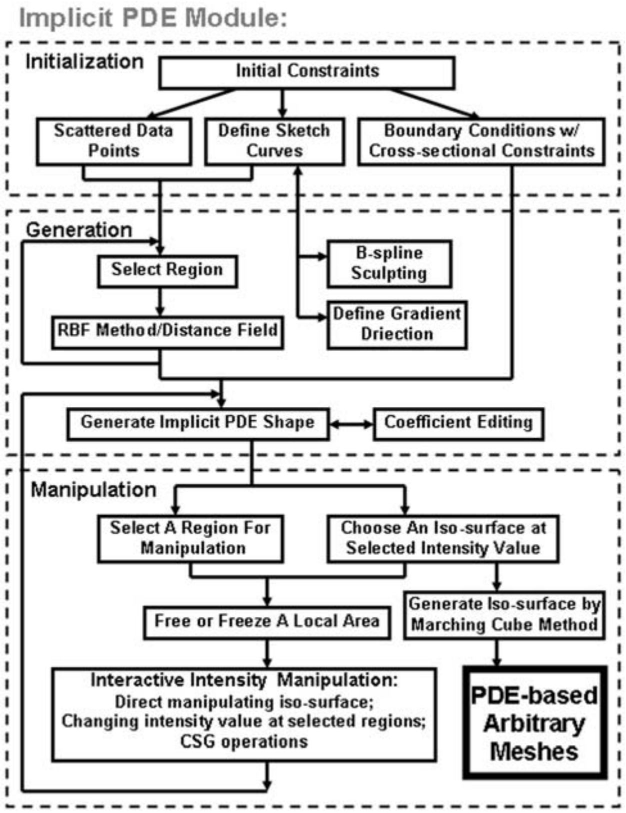

Implicit PDE Module:
Initialization
Initial Constraints
Scattered Data Points
Define Sketch Curves
Boundary Conditions w/ Cross-sectional Constraints
Generation
Select Region
B-spline Sculpting
RBF Method/Distance Field
Define Gradient Driection
Generate Implicit PDE Shape
Coefficient Editing
Manipulation
Select A Region For Manipulation
Choose An Iso-surface at Selected Intensity Value
Free or Freeze A Local Area
Generate Iso-surface by Marching Cube Method
Interactive Intensity Manipulation: Direct manipulating iso-surface; Changing intensity value at selected regions; CSG operations
PDE-based Arbitrary Meshes

Figure 9.7: System architecture for implicit PDE module.

points, or volumetric datasets. The system also allows direct manipulation of reconstructed implicit PDE objects with various intensity constraints in the volumetric working space. The direct sculpting of implicit PDE objects can be obtained via modification of pre-defined conditions and interior operations. Fig. 9.7 illustrates the architecture of the implicit PDE module of the PDE-based modeling system. In particular, the module provides the following functionalities:

Missing Information Recovery and Shape Blending. The underlying implicit PDEs of the PDE modeling system provide a simple yet systematic mechanism to obtain the volumetric information satisfying specified constraints automatically. Such an advantage allows the PDE modeling system to recover the missing information of input datasets. It can also be used to compute connecting parts between different objects in the working space which leads to shape blending.

Shape Reconstruction. Users can interactively input and edit scattered data points or sketch curves with specified intensity values, then the system uses the RBF method or distance field approximation to calculate intensity values on the sampling grids within the volumetric working space as an initial guess for the iterative solver of the discretized implicit PDE to obtain an approximate solution for implicit PDE objects satisfying these conditions. The system can model both closed and open implicit shapes.

Discrete Models. The PDE modeling system supports implicit PDE objects obtained from solving the elliptic PDEs using: (1) finite-difference discretization for the numerical solution of the fourth-order and second-order elliptic PDEs in 3D working space; and (2) RBF approximation at arbitrary sub-regions in the working space for modeling localized details and performance speedup.

Interactive and Direct Operations. Users can also work directly on the implicit PDE objects through: (1) local modification of blending coefficient functions; (2) sketch curve sculpting using B-spline manipulation; (3) gradient specification of selected curves; (4) local RBF approximation for improved time performance and interactive CSG manipulation; (5) interior deformation with additional constraints inside the working space; (6) iso-surface manipulation and direct sculpting of iso-contours at selected intensity values; and (7) gradient and curvature constraints inside the working space.

9.3.3 System Performance and Discussion

Iterative methods (e.g. Gauss-Seidel iteration) with multi-grid techniques are employed to solve the implicit PDEs subject to various constraints. Besides original datasets or predefined sketch curves, the implicit PDE module allows users to interactively define and sculpt sketch curves directly and specify gradient directions at selected curves. These constraints provide more freedom to designers and make intuitive design of implicit objects more cost-effective. Users can also enforce additional constraints directly inside the volumetric working space by applying local operations and sculpting toolkits for implicit objects. For implicit PDE models, first the RBF method or fast-tagging algorithm are used to get an initial guess of intensity distribution for the entire implicit space, then iterative methods based on finite-difference approximations are performed to get solutions with additional constraints. The initial guess can be stored to save time for further manipulations.

Examples	Constraints	Initial	2nd(s)	4th(s)
Fig. 6.2	169888	N/A	1.542	7.992
Fig. 6.3	274086	N/A	3.04	13.7
Fig. 6.4 (a)	180	5.889	N/A	379.766
Fig. 6.4 (g)	720	18.872	N/A	416.312
Fig. 6.6	960	30.584	N/A	425.688
Fig. 6.7 (a)	1218	267.925	N/A	113.432
Fig. 6.7 (e)	3154	359.657	N/A	148.283

Table 9.6: CPU time (in seconds) of different solvers for several examples of implicit PDE objects with different number of constraints.

Table 9.6 summarizes the numbers of constraints and CPU time of numerical solvers for the second-order and fourth-order implicit PDE examples when running on a Pentium 4 1.4GHz PC. The resolution of the working space is $64 \times 64 \times 64$ for Fig. 6.2 and $65 \times 65 \times 65$ for other examples. The stopping threshold (difference between two iteration steps) is 10^{-9}. ”Initial” stands for the initial guess where we use the RBF method for sketch curve datasets and the fast-tagging approximation for the scattered data points input. ”2nd(s)” and ”4th(s)” indicate the CPU time in seconds for solving the entire implicit second-order and fourth-order PDE working spaces based on initial guesses using multi-grid Gauss-Seidel iteration. The time performance of the RBF and fast-tagging algorithms depends on the number of enforced constraints, while convergent speeds of iterative methods are mainly determined by the sampling rates of the implicit working space.

Although the initialization of implicit models is time-consuming because of the approximation of the entire working space, the local sculpting afterward will be interactive because only small number of sampling grids are involved. Table

Examples	Constraints	Grids	4th(s)
Iso-contour Editing(Fig.6.8c)	8	1792	0.45
Region Deformation(Fig.6.8d)	507	6358	3.17
CSG-like Blending(Fig.6.8f)	108	1000	1.15
Sharp-feature Creation(Fig.6.8g)	98	5046	0.23
Cutting-1(Fig.6.8h)	216	1000	0.82
Cutting-2(Fig.6.8i)	216	1000	0.82
Gradient Sculpting(Fig.6.10)	7	294	0.09
Curvature Manipulation(Fig.6.11)	7	294	0.09

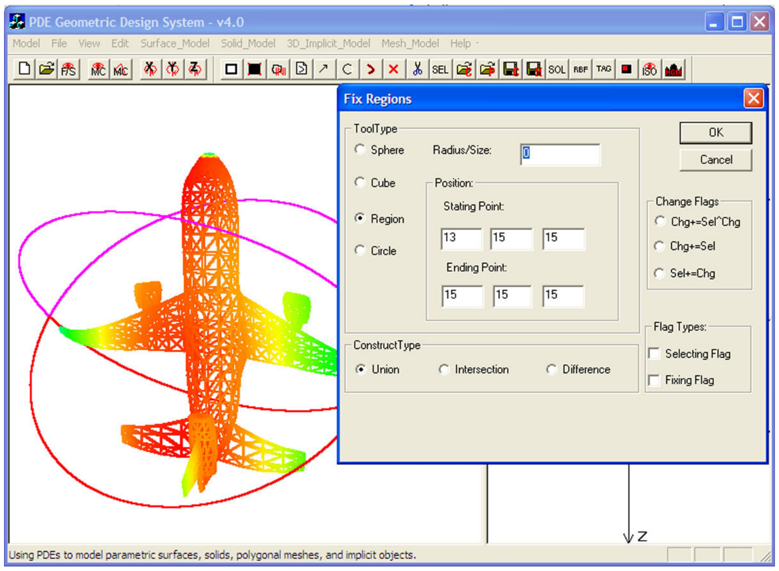
Table 9.7: CPU time (seconds) of local direct manipulation examples of implicit PDE objects.

9.7 summarizes the CPU time for direct sculpting examples in local selected regions. ”Constraints” stands for the number of constraints involved for the operation, ”Grids” represents the number of grid points in the selected region, and ”4th(s)” gives the CPU time (seconds) for updating the intensity values in the selected area using the fourth-order PDE. The CPU time depends on the scale of intensity changes by the sculpting operation as well as the number of constraints and the size of the selected region. For instance, CSG operations usually enforce relatively larger intensity changes for constraints in selected regions than other operations such as gradient and curvature sculpting, hence they need more CPU time to update the region’s intensity distribution.

9.4 PDE-based Free-Form Modeling and Deformation Structure and Results

9.4.1 Overview

The PDE-based free-form modeling and deformation module offers free-form PDE-based shape design, sculpting, blending, and deformation from geometric boundary surfaces or curve network and intensity attributes with direct manipulation toolkits. Fig. 9.8 shows a snapshot of the system interface while modeling a polygonal mesh object as an embedded dataset inside a PDE solid obtained from a set of boundary curves. The modeling toolkits are provided under ”Solid_Model” in the menu bar of the system. The dialog of ”Fix Regions” provides the specification of the sculpting tools for PDE solids. The left display window shows the PDE solids with embedded datasets, which can be displayed in wireframe, shaded model, and discretized grids. It can also show intensity distributions when they are


PDE Geometric Design System - v4.0
Model File View Edit Surface_Model Solid_Model 3D_Implicit_Model Mesh_Model Help
Fix Regions
OK
Cancel
Z

ToolType: ( ) Sphere [✓ Region] ( ) Cube ( ) Circle

Radius/Size: 0

Position:
Stating Point: 13 15 15
Ending Point: 15 15 15

Change Flags: ( ) Chg+=Sel^Chg ( ) Chg+=Sel ( ) Sel+=Chg

Flag Types: [ ] Selecting Flag [ ] Fixing Flag

ConstructType: [✓ Union] ( ) Intersection ( ) Difference

Using PDEs to model parametric surfaces, solids, polygonal meshes, and implicit objects.

Figure 9.8: System interface for the PDE-based free-form modeling and deformation module.

available. The two right windows are for manipulations of boundary conditions of PDE solids.

The PDE-based free-form modeling and deformation module provides users interactive manipulations of free-form PDE solids with various local/global constraints and intensity properties and allows interactive sculpting of PDE solids via boundary conditions and interior operations. Fig. 9.9 illustrates its architecture outline.

9.4.2 PDE Solid Modeling Toolkits

The PDE modeling system offers a set of direct manipulation toolkits for the PDE-based free-form deformation and modeling.

Geometric Boundary Representations. Users can interactively input and edit boundary surfaces or boundary curves by selecting the boundary of interests,

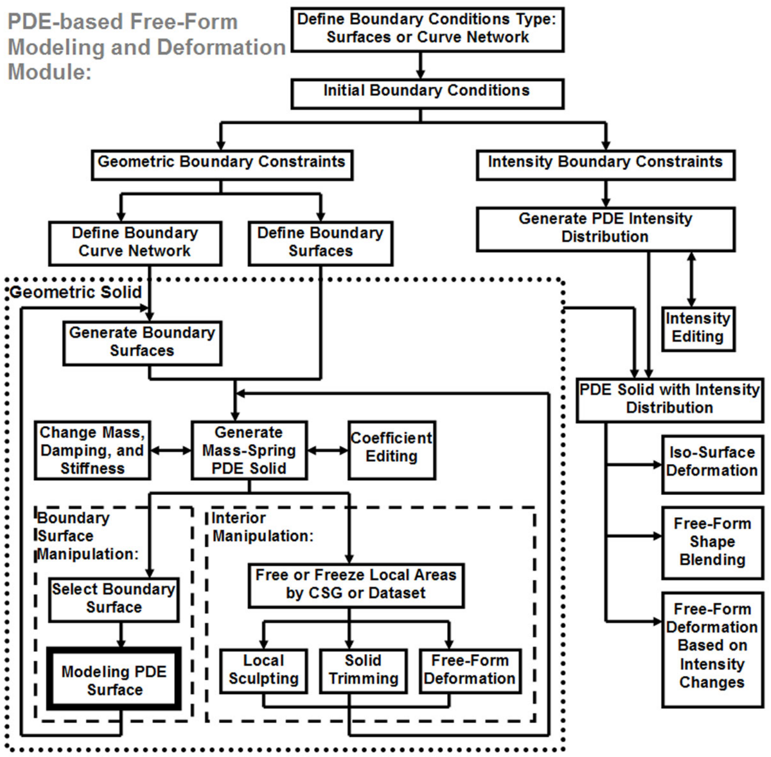

PDE-based Free-Form Modeling and Deformation Module:
Define Boundary Conditions Type: Surfaces or Curve Network
Initial Boundary Conditions
Geometric Boundary Constraints
Intensity Boundary Constraints
Define Boundary Curve Network
Define Boundary Surfaces
Generate PDE Intensity Distribution
Geometric Solid
Generate Boundary Surfaces
Intensity Editing
PDE Solid with Intensity Distribution
Change Mass, Damping, and Stiffness
Generate Mass-Spring PDE Solid
Coefficient Editing
Iso-Surface Deformation
Boundary Surface Manipulation:
Interior Manipulation:
Free-Form Shape Blending
Select Boundary Surface
Free or Freeze Local Areas by CSG or Dataset
Free-Form Deformation Based on Intensity Changes
Modeling PDE Surface
Local Sculpting
Solid Trimming
Free-Form Deformation

Figure 9.9: System architecture of the PDE-based free-form modeling and deformation module.

and obtain PDE solids satisfying these conditions. Moreover, the system offers a multi-grid subdivision scheme to improve time performance of iterative techniques.

Dynamic Models. The system supports physics-based PDE solids using numerical techniques including: (1) finite-difference discretization using mass-spring models; (2) multi-grid subdivision for model refinement and performance speedup. Material properties and dynamic behavior greatly enhance interactivities while manipulating conventional PDE solids.

Boundary Constraint Manipulations. Users can use various manipulation toolkits to deform boundary surfaces including: (1) editing points and their normal and curvature at arbitrary locations; (2)enforcing a set of curve constraints; (3) deforming user-specified regions; and (4) applying local operations only in user-selected areas.

Geometric Interior Operations. In addition, users can also work directly inside the PDE solid through: (1) interior deformation with additional constraints inside the solid; (2) trimming specified regions for complex geometry and arbitrary topology; and (3) modifying control functions as well as material properties such as mass, damping, and stiffness distributions locally. Regions of interests can be selected through the ”Fix Regions” function on interface.

Free-Form Deformation Based on Intensity Fields. In order to offer users more degrees of freedom for shape manipulation based on PDE techniques, this dissertation integrates scalar intensity properties with PDE solid geometry for arbitrary shape modeling. (1) The system allows users to model the implicit PDE shape through geometric free-form deformation and direct manipulation; (2) the system provides free-form shape blending features by integrating geometric and intensity properties in the working space; and (3) users can also obtain deformed shape by changing the intensity distribution globally/locally in the PDE solid working space.

9.4.3 Performance

Because of the finite-difference discretization, the system can model the entire 3D parametric PDE space through direct sculpting, which offers PDE-based free-form modeling and deformation for embedded objects inside the 3D parametric domain. Table 9.8 details the time performance of some PDE solid geometry examples.

In Table 9.8, ”Gauss-Seidel” stands for Gauss-Seidel iteration and ”SOR”

Model	Gauss-Seidel	SOR
Sphere-4	304.029	257.790
Sphere-2	19.989	16.404
Sphere-s	31.436	28.626
Cylinder-4	388.642	369.157
Cylinder-2	39.471	34.078
Cylinder-s	143.256	122.375

Table 9.8: CPU time (in seconds) for PDE solid examples using different solvers.

means Successive Over-Relaxation technique. ”Sphere” represents the PDE solid obtained from boundary surfaces in Fig. 7.2 (a) and (b). ”Cylinder” stands for the cylinder-like PDE solid obtained from boundary curve network in Fig. 7.3 (a) and (b). The ”-4”, ”-2” and ”-s” stand for the 4th, 2nd order PDE, and the 4th order PDE with subdivision, respectively.

Besides traditional boundary conditions of PDE techniques, the system allows users to specify and enforce a large variety of additional constraints on a set of points, cross-sectional curves, and surface areas on the geometric boundary surfaces. These constraints provide more freedom to designers, making the design process of PDE solids more cost-effective. The curve-based boundary conditions make it even easier for designers to achieve the desired shape of the PDE solid. Users can also enforce additional constraints directly inside the PDE solid and apply trimming and free-form operations, which facilitate the construction of PDE solids of arbitrary topology. The prototype system uses finite-difference techniques because they are simple, easy to implement, and suitable for the incorporation of complicated, flexible constraints. In general, the time and space complexity will increase with higher resolution as well as increased accuracy. The multi-grid subdivision method for various levels of refinement achieves anticipated results in the experiments. The free-form modeling and deformation based on PDE solid geometry and intensity distributions provide arbitrary shape blending and modification functionalities for the PDE-based modeling system.

Despite the direct and powerful modeling advantages of the PDE framework, the major difficulty associated with the proposed PDE techniques is the convergent speed of finite-difference approximation. Thus, faster numerical approximation techniques for solving PDEs may be considered to improve the time performance of the PDE-based modeling system.

Chapter 10

Conclusion

This dissertation presents a novel PDE-based modeling and interactive sculpting system that offers geometric representations with physical and material properties and a set of modeling/deformation toolkits as solutions of certain PDEs under boundary and initial conditions associated with additional constraints. The system is governed by PDEs and users can model objects without worrying about the underlying mathematical details. The PDE formulation can recover the entire shape information from partial input and alleviate the burden of specification of various control information for designers. Several currently popular and efficient modeling techniques, such as parametric representations, implicit functions, physics-based modeling, and PDE techniques, are integrated into a single framework to design smooth parametric or implicit geometric entities from generalized boundary constraints, manipulate displacements models for parametric or polygonal surfaces, extract medial axis/skeleton structures of objects bounded by polygonal meshes, deform shapes based on skeleton manipulations and diffusion-based propagation, reconstruct objects from partial information by sketching a set of arbitrary non-isoparametric curves or unorganized scattered data points, smoothly blend implicit objects, and offer physics-based interactive and direct manipulation toolkits for geometric models, such as free-form deformation and shape blending, global and local sculpting, physical and material property modification, etc. With these modeling features, the PDE-based modeling system offers a generalized PDE modeling mechanism which covers popular applications in geometric modeling, including dynamic physics-based surface and displacement sculpting of arbitrary topology, medial axis extraction for model simplification, skeleton-based shape manipulation using diffusion-based front propagation models, implicit shape reconstruction and deformation, damaged data recovery, arbitrary shape blending, and PDE-based free-form solid modeling and deformation with intensity and physical properties. The system offers modeling functionalities for geometric objects of various types of representations, including parametric surfaces, arbitrary polygonal meshes, free-form solids, implicit models, as well as volumetric data. It also provides data exchange among these types of formats. It also provides a comprehensive set of direct and interactive manipulation toolkits for common users including point-based editing, region sculpting, curve and region mapping, normal constraints, curvature manipulation, displacement modeling, intensity and material modifications, etc. These toolkits provide users more degrees of freedom to model geometric shapes than previous techniques. This dissertation employs several simple but efficient numerical techniques such as finite-difference method with multi-grid iterative techniques, SOR relaxation methods, and least-square fitting techniques to solve the elliptic PDEs and diffusion equations with additional and general constraints. These techniques provide a powerful and intriguing framework using the PDE techniques for general geometric simulation and interactive modeling.

10.1 Future Work

As for future work, I will continue to explore applications using PDE techniques in visual computing areas such as geometric modeling, visualization, image processing, simulation, animation, etc.

- Sweeping objects design: the front propagation techniques allow objects to grow from skeletons and propagate along certain specific directions. This will provide a method to generate sweeping objects and can be used for sweeping shape design.

- Implicit shape morphing: source and target objects for shape transformation can be embedded into the implicit working space by setting different intensity values respectively and using implicit PDEs to compute the transition of intensity values from the source to the target. Implicit PDEs offer high-order continuity of intensity distribution over the entire working space, which will provide a smooth transformation between shapes.

- Data reconstruction and recovery: the implicit PDE model can model volumetric datasets obtained from 3D scan devices. It can reconstruct embedded objects from volumetric datasets for further analysis. Because the underlying PDE can recover domain shape information through partial input, it can be used for damaged data recovery when only partial information of objects is available.

- Image Processing and Medical Imaging: PDE techniques are popular in image processing areas. I want to further employ PDEs in image processing especially medical imaging applications. I consider to use diffusion equations for feature enhancement and noise removal to improve image qualities at the same time. I also want to explore applications of implicit PDEs in image morphing, which then can be used for image analysis and animation. Moreover, because PDEs can reconstruct information of entire space from partial input, they are ideal candidates for model recovery from cross-sectional slicing images, which is extremely useful for medical data reconstruction. This can be considered as another possible future focus in my research.

- Simulation and Animation: because most of the natural phenomena can be formulated using differential equations, PDE techniques such as time-dependent wave equations are among commonly used methods for natural phenomena simulation such as water, smoke, fire, etc. Their applications include simulating natural scenes and creating visual realistic scenes in animation and movie production. Because there are various types of PDEs that are possible candidates for simulation and animation, another direction of my future research will focus on employing different types of PDEs in natural scene simulation and animation. My work will aim at using PDE techniques to produce more realistic and accurate results with better time performance than previous PDE-based techniques, which are normally solved by numerical approximations.

Bibliography

[1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:269–277, 1995.

[2] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pages 249–266. ACM Press, 2001.

[3] C. Arcelli and G. S. di Baja. A width-independent fast thinning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):463–474, 1985.

[4] C. Arcelli and G. S. di Baja. Ridge points in euclidean distance maps. *Pattern Recognition Letters*, 13(4):237–243, 1992.

[5] A. Bærentzen and N. J. Christensen. Volume sculpting using level-set method. In *Proceedings of International Conference on Shape Modeling and Applications 2002*, pages 175–182, Banff, Canada, 2002.

[6] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH 1998*, pages 43–54, Orlando, USA, 1998.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH 2000*, pages 417–424, New Orleans, USA, 2000.

[8] I. Bitter, A. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[9] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[10] J. Bloomenthal and C. Lim. Skeletal methods of shape manipulation. In *Proceedings of International Conference on Shape Modeling and Applications 1999*, pages 44–47, Aizu-Wakamatsu, Japan, 1999.

[11] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, 1990.

[12] M. I. G. Bloor and M. J. Wilson. Blend design as a boundary-value problem. In *Geometric Modeling: Theory and Practice*, pages 221–234, I.W. Straßer (ed), Springer-Verlag, 1989.

[13] M. I. G. Bloor and M. J. Wilson. Generating blend surfaces using partial differential equations. *Computer-Aided Design*, 21(3):165–171, 1989.

[14] M. I. G. Bloor and M. J. Wilson. Representing PDE surfaces in terms of B-splines. *Computer-Aided Design*, 22(6):324–331, 1990.

[15] M. I. G. Bloor and M. J. Wilson. Using partial differential equations to generate free-form surfaces. *Computer-Aided Design*, 22(4):202–212, 1990.

[16] M. I. G. Bloor and M. J. Wilson. Functionality in solids obtained from partial differential equations. *Computing Suppl. 8*, pages 21–42, 1993.

[17] M. I. G. Bloor and M. J. Wilson. Spectral approximations to PDE surfaces. *Computer-Aided Design*, 28(2):145–152, 1996.

[18] H. Blum. A transformation for extracting new descriptions of shape. In *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[19] H. Blum. Biological shape and visual science. *Journal of Theoretical Biology*, 38:205–287, 1973.

[20] W. Bohm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1(1):1–60, 1984.

[21] S. Bouix and K. Siddiqi. Divergence-based medial surfaces. In *Proceedings of Sixth European Conference on Computer Vision (ECCV 2000)*, pages 603–618, Dublin, Ireland, 2000.

[22] D. Breen, R. Fedkiw, S. Osher, G. Sapiro, and R. Whitaker. *Level Set and PDE Methods for Computer Graphics*. SIGGRAPH 2002 Course Notes 10, 2002.

[23] D. Breen and R. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[24] M. Carignan, Y. Yang, N. M. Thalmann, and D. Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *SIGGRAPH 1992*, pages 99–104, Chicago, USA, 1992.

[25] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, and B. McCallum. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH 2001*, pages 67–76, Los Angeles, USA, 2001.

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

[27] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics*, 25(4):165–170, 1991.

[28] G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[29] D. Cohen-Or and D. Levin. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, 1998.

[30] R. Cook. Shade trees. *Computer Graphics (Proceedings of SIGGRAPH 1984)*, 18(3):223–231, 1984.

[31] S. Coquillart. Extended free-form deformation: A sculpting tool for 3D geometric modeling. In *SIGGRAPH 1990*, pages 187–196, Dallas, USA, 1990.

[32] B. Crespin. Implicit free-form deformations. In *Proceedings of Implicit Surfaces 1999*, page 1723, Bordeaux, France, 1999.

[33] B. Cutler, J. Dorsey, L. McMillan, M. Müller, and R. Jagnow. A procedural approach to authoring solid models. In *SIGGRAPH 2002*, pages 302–311, San Antonio, USA, 2002. ACM Press.

[34] F. Dachille, H. Qin, A. Kaufman, and J. El-Sana. Haptic sculpting of dynamic surfaces. In *Proceedings of 1999 ACM Symposium on Interactive 3D Graphics*, pages 103–110, Atlanta, USA, 1999.

[35] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[36] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH 1998*, pages 85–94, Orlando, USA, 1998.

[37] M. Desbrun and M.-P. Cani-Gascuel. Active implicit surfaces for animation. In *Proceedings of Graphics Interface 1998*, pages 143–150, 1998.

[38] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 1999*, pages 317–323, Los Angeles, USA, 1999.

[39] M. Desbrun, N. Tsingos, and M.-P. Gascuel. Adaptive sampling of implicit surfaces for interactive modeling and animation. *Computer Graphics Forum*, 15(5):319–325, 1996.

[40] T. K. Dey and W. Zhao. Approximate medial axis as a voronoi subcomplex. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, pages 356–366. ACM Press, 2002.

[41] U. Diewald, T. Preußer, and M. Rumpf. Anisotropic diffusion in vector field visualization on euclidean domains and surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):139–149, 2000.

[42] D. Doo and M. Sabin. Behavior of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[43] H. Du and H. Qin. Direct manipulation and interactive sculpting of PDE surfaces. In *Proceedings of EuroGraphics 2000*, pages C261–C270, Interlaken, Switzerland, 2000.

[44] H. Du and H. Qin. Dynamic PDE surfaces with flexible and general constraints. In *Proceedings of the Seventh Pacific Conference on Computer*

Graphics and Applications (Pacific Graphics 2000), pages 213–222, Hong Kong, China, 2000.

[45] H. Du and H. Qin. Integrating physics-based modeling with PDE solids for geometric design. In *Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2001)*, pages 198–207, Tokyo, Japan, 2001.

[46] H. Du and H. Qin. Dynamic PDE-based surface design using geometric and physical constraints. *Accepted by Graphical Models*, 2003.

[47] H. Du and H. Qin. Interactive shape design using volumetric implicit PDEs. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, pages 235–246. ACM Press, 2003.

[48] H. Du and H. Qin. Free-form solid modeling by integrating parametric and implicit PDEs. *Submitted for journal publication*, 2004.

[49] H. Du and H. Qin. Medial axis extraction and shape manipulation of solid objects using parabolic PDEs. In *Proceedings of the Nineth ACM Symposium on Solid Modeling and Applications*, 2004.

[50] H. Du and H. Qin. PDE-based skeletonization and propagation for arbitrary topological shapes. *In preparation for journal submission*, 2004.

[51] H. Du and H. Qin. A shape design system using volumetric implicit PDEs. *Accepted by Computer-Aided Design, in press*, 2004.

[52] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[53] D. S. Ebert, F. K. Musgrave, P. Prusinkiewicz, J. Stam, and J. Tessendorf. *Simulating Nature: From Theory to Practice*. SIGGRAPH 2000 Course Notes 25, 2000.

[54] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide*. Academic Press, 1996.

[55] E. Ferley, M.-P. Cani, and J.-D. Gascuel. Practical volumetric sculpting. *The Visual Computer*, 16(8):469–480, 2000.

[56] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. *Computer Graphics*, 22(4):205–212, 1988.

[57] M. Foskey, M. C. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, pages 96–107. ACM Press, 2003.

[58] N. Foster and D. Metaxas. Realistic animation of liquids. In *Proceedings of Graphics Interface 1996*, pages 204–212, 1996.

[59] N. Foster and D. Metaxas. Controlling fluid animation. In *Proceedings of Computer Graphics International 1997*, pages 178–188, 1997.

[60] N. Foster and D. Metaxas. Modeling the motion of hot, turbulent gas. In *SIGGRAPH 1997*, pages 181–188, Los Angeles, USA, 1997.

[61] S. Frisken, R. Perry, A. Rockwood, and T. Jones. Adaptive sampled distance fields: A general representation of shape for computer graphics. In *SIGGRAPH 2000*, pages 249–254, New Orleans, USA, 2000.

[62] J. A. Goldak, X. Yu, A. Knight, and L. Dong. Constructing discrete medial axis of 3-D objects. *Int. J. Computational Geometry and Its Applications*, 1(3):327–339, 1991.

[63] J. Gomez and O. Faugeras. Reconciling distance functions and level sets. *Journal of Visual Communication and Image Representation*, 11(2):209–223, 2000.

[64] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *SIGGRAPH 1993*, pages 35–44, Anaheim, USA, 1993.

[65] C. M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, 1989.

[66] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. MaDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH 1994*, pages 295–302, Orlando, USA, 1994.

[67] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 1992*, pages 71–78, Chicago, USA, 1992.

[68] W. H. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformation. In *SIGGRAPH 1992*, pages 177–184, Chicago, USA, 1992.

[69] J. Hua and H. Qin. Haptic sculpting of volumetric implicit functions. In *Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2001)*, pages 254–264, Tokyo, Japan, 2001.

[70] J. Hua and H. Qin. Dynamic implicit solids with constraints for haptic sculpting. In *Proceedings of International Conference on Shape Modeling and Applications 2002*, pages 119–128, Banff, Canada, 2002.

[71] J. Hua and H. Qin. Scalar-field-guided adaptive shape deformation and animation. *The Visual Computer, in press*, 2003.

[72] R. Jagnow and J. Dorsey. Virtual sculpting with haptic displacement maps. In *Proceedings of Graphics Interface 2002*, pages 125–132, Calgary, Canada, 2002.

[73] M. Kass and G. Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH 1990*, pages 49–57, Dallas, USA, 1990.

[74] A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, pages 203–211, Ann Arbor, USA, 1999.

[75] R. Kimmel, D. Shaked, N. Kiryati, and A. M. Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding: CVIU*, 62(3):382–391, 1995.

[76] G. Kindlmann, D. Weistein, and D. Hart. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):124–138, 2000.

[77] L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13:743–761, 1996.

[78] L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Proceedings of EuroGraphics 2000*, pages C417–C427, Interlaken, Switzerland, 2000.

[79] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multiresolution modeling on arbitrary meshes. In *SIGGRAPH 1998*, pages 105–114, Orlando, USA, 1998.

[80] L. Kobbelt and P. Schröder. A multiresolution framework for variational subdivision. *ACM Transactions on Graphics*, 17(4):209–237, 1998.

[81] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. I. H. Parish. Simulating facial surgery using finite element models. In *SIGGRAPH 1996*, pages 421–428, New Orleans, USA, 1996.

[82] V. Krishnamurthy and M. Levoy. Fitting smooth surface to dense polygon meshes. In *SIGGRAPH 1996*, pages 313–324, New Orleans, USA, 1996.

[83] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *SIGGRAPH 2000*, pages 85–94, New Orleans, USA, 2000.

[84] T. Lee and R. Kashyap. Building skeleton models via 3D medial surface/axis thinning algorithm. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[85] Y. Lee, D. Terzopoulos, and K. Waters. Realistic face modeling for animation. In *SIGGRAPH 1995*, pages 55–62, Los Angeles, USA, 1995.

[86] A. Levin. Interpolating nets of curves by smooth subdivision surfaces. In *SIGGRAPH 1999*, pages 57–64, Los Angeles, USA, 1999.

[87] F. Leymarie and M. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):56–75, 1992.

[88] C. Loop. Smooth spline surfaces over irregular meshes. In *SIGGRAPH 1994*, pages 303–310, Orlando, USA, 1994.

[89] C. Loop and T. DeRose. Generalized b-spline surfaces of arbitrary topology. *Computer Graphics*, 24(4):347–356, 1990.

[90] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH 1987*, pages 163–169, Anaheim, USA, 1987. ACM Press.

[91] T. W. Lowe, M. I. G. Bloor, and M. J. Wilson. Functionality in blend design. *Advanced in Design Automation, Vol 1: Computer Aided and Computational Design ASME, New York*, pages 43–50, 1990.

[92] W.-C. Ma, F.-C. Wu, and M. Ouhyoung. Skeleton extraction of 3D objects with radial basis functions. In *Proceedings of International Conference on Shape Modeling and Applications 2003*, pages 207–215, Seoul, Korea, 2003.

[93] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH 1996*, pages 181–188, New Orleans, USA, 1996.

[94] C. Mandal, H. Qin, and B. C. Vemuri. A novel FEM-based dynamic framework for subdivision surfaces. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, pages 191–202, Ann Arbor, USA, 1999. ACM Press.

[95] A. Manzanera, T. M. Bernard, F. Preteux, and B. Longuet. Medial faces from a concise 3D thinning algorithm. In *Proceedings of the Seventh IEEE*

International Conference on Computer Vision (ICCV'99), pages 337–343, Kerkyra, Greece, 1999.

[96] K. T. McDonnell and H. Qin. Dynamic sculpting and animation of free-form subdivision solids. In *Proceedings of IEEE Computer Animation 2000*, pages 126–133, Philadelphia, USA, 2000.

[97] K. T. McDonnell, H. Qin, and R. A. Wlodarczyk. Virtual clay: A real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics*, pages 179–190, Salt Lake, USA, 2001.

[98] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *SIGGRAPH 1992*, pages 309–312, Chicago, USA, 1992.

[99] B. Morse, T. Yoo, P. Rheingans, D. Chen, and K. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings of International Conference on Shape Modeling and Applications 2001*, pages 89–98, Genova, Italy, 2001.

[100] M. E. Mortenson. *Geometric Modeling, Second Edition*. Wiley Computer Publishing, 1997.

[101] S. Muraki. Volumetric shape description of range data using blobby model. *Computer Graphics*, 25(4):227–235, 1991.

[102] K. Museth, D. Breen, R. Whitaker, and A. Barr. Level set surface editing operators. In *SIGGRAPH 2002*, pages 330–338, San Antonio, USA, 2002. ACM Press.

[103] M. Näf, O. Kübler, R. Kikinis, M. Shenton, and G. Szekely. Characterization and recognition of 3D organ shape in medical image analysis using skeletonization. In *Proceedings of the 1996 IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA '96)*, page 139. IEEE Computer Society, 1996.

[104] J. F. O'Brien. *Graphical Modeling and Animation of Fracture*. PhD Thesis, Geogia Institute of Technology, 2000.

[105] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *SIGGRAPH 1999*, pages 137–146, Los Angeles, USA, 1999.

[106] J. F. O'Brien and J. K. Hodgins. Animating fracture. *Communications of the ACM*, 43(7):68–75, 2000.

[107] R. Ogniewicz. *Discrete Voronoi Skeletons*. Hartung-Gorre, 1993.

[108] S. J. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[109] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

[110] R. Perry and S. Frisken. Kizamu: A system for sculpting digital characters. In *SIGGRAPH 2001*, pages 47–56, Los Angeles, USA, 2001.

[111] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, 1991.

[112] L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. *Computer-Aided Design*, 19(9):485–498, 1987.

[113] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, Cambridge, 1992.

[114] T. Preußer and M. Rumpf. Anisotropic nonlinear diffusion in flow visualization. In *IEEE Visualization 1999*, pages 325–332, Sanfrancisco, USA, 1999.

[115] H. Qin, C. Mandal, and B. C. Vemuri. Dynamic catmull-clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, 1998.

[116] H. Qin and D. Terzopoulos. Dynamic NURBS swung surfaces for physics-based design. *Computer-Aided Design*, 27(2):111–127, 1995.

[117] H. Qin and D. Terzopoulos. D-NURBS: A physics-based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, 1996.

[118] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, pages 246–257. ACM Press, 1999.

[119] A. Sarti and S. Tubaro. Multiresolution implicit object modeling. In *VMV 2001, Stuttgart, Germany, November 2123, 2001*, pages 93–100, 2001.

[120] V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.

[121] B. Schmitt, A. Pasko, and C. Schlick. Shape-driven deformations of functionally defined heterogeneous volumetric objects. In *Proceedings of GRAPHITE 2003*, pages 321–322, Melbourne, Australia, 2003.

[122] M. Schmitt. Some examples of algorithms analysis in computational geometry by means of mathematical morphology techniques. *Lecture Notes in Computer Science, Geometry and Robotics, Springer-Verlag*, 391:225–246, 1989.

[123] R. Schneider and L. Kobbelt. Generating fair meshes with G^1 boundary conditions. In *Geometric Modeling and Processing Conference Proceedings, 2000*, pages 251–261, 2000.

[124] S. Sclaroff and A. Pentland. Generalized implicit functions for computer graphics. *Computer Graphics*, 25(4):247–250, 1991.

[125] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH 1986*, pages 151–160, USA, 1986.

[126] T. W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform recursive subdivision surfaces. In *SIGGRAPH 1998*, pages 387–394, Orlando, USA, 1998.

[127] J. Sethian. A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed. *Journal of Differential Geometry*, 31:131–161, 1989.

[128] D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Shape description by medial surface construction. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):62–72, 1996.

[129] E. C. Sherbrooke, N. Patrikalakis, and E. Brisson. An algorithm for the medial axis transform of 3D polyhedral solids. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):44–61, 1996.

[130] E. C. Sherbrooke, N. M. Patrikalakis, and E. Brisson. Computation of the medial axis transform of 3-D polyhedra. In *Proceedings of the Third ACM Symposium on Solid Modeling and Applications*, pages 187–200. ACM Press, 1995.

[131] K. Siddiqi, A. Tannenbaum, and S. W. Zuker. A hamiltonian approach to the eikonal equation. In *Proceedings of Second International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR 99)*, pages 1–13, York, UK, 1999.

[132] K. Singh and E. Fiume. Wires: A geometric deformation technique. In *SIGGRAPH 1998*, pages 405–414, Orlando, USA, 1998.

[133] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH 1998*, pages 387–394, Orlando, USA, 1998.

[134] J. Stam. Stable fluids. In *SIGGRAPH 1999*, pages 121–127, Los Angeles, USA, 1999.

[135] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion professes. In *SIGGRAPH 1995*, pages 129–136, Los Angeles, USA, 1995.

[136] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[137] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 1995*, pages 351–358, Los Angeles, USA, 1995.

[138] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[139] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *SIGGRAPH 1988*, pages 269–278, Atlanta, USA, 1988.

[140] D. Terzopoulos, J. Platt, A. H. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH 1987*, pages 205–214, Anaheim, USA, 1987.

[141] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[142] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *SIGGRAPH 1994*, pages 43–50, Orlando, USA, 1994.

[143] A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society*, 237(B):37–72, 1952.

[144] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics*, 25(4):289–298, 1991.

[145] G. Turk, H. Q. Dinh, J. F. O'Brien, and G. Yngve. Implicit surfaces that interpolate. In *Proceedings of International Conference on Shape Modeling and Applications 2001*, pages 62–71, Genova, Italy, 2001.

[146] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH 1999*, pages 335–342, Los Angeles, USA, 1999.

[147] G. Turk and J. F. O'Brien. Modeling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, 2002.

[148] H. Ugail, M. I. G. Bloor, and M. J. Wilson. Techniques for interactive design using the PDE method. *ACM Transactions on Graphics*, 18(2):195–212, 1999.

[149] P. Veron and J. C. Leon. Static polyhedron simplification using error measurements. *Computer-Aided Design*, 29(4):287–298, 1997.

[150] H. Weimer and J. Warren. Subdivision schemes for fluid flow. In *SIGGRAPH 1999*, pages 111–120, Los Angeles, USA, 1999.

[151] W. Welch and A. Witkin. Variational surface modeling. In *SIGGRAPH 1992*, pages 157–166, Chicago, USA, 1992.

[152] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH 1994*, pages 247–256, Orlando, USA, 1994.

[153] R. Whitaker and D. Breen. Level-set models for the deformation of solid objects. In *Proceedings of Conference of Implicit Surface 1998*, pages 19–36, Seattle, USA, 1998.

[154] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH 1994*, pages 269–277, Orlando, USA, 1994.

[155] A. Witkin and M. Kass. Reaction-diffusion textures. In *SIGGRAPH 1991*, pages 299–308, Las Vegas, USA, 1991.

[156] P. Witting. Computational fluid dynamics in a traditional animation environment. In *SIGGRAPH 1999*, pages 129–136, Los Angeles, USA, 1999.

[157] X. Ye, T. R. Jackson, and N. M. Patrikalakis. Geometric design of functional surfaces. *Computer-Aided Design*, 28(9):741–752, 1996.

[158] G. D. Yngve, J. F. O'Brien, and J. K. Hodgins. Animating explosions. In *SIGGRAPH 2000*, pages 29–36, New Orleans, USA, 2000.

[159] E. Zauderer. *Partial Differential Equations of Applied Mathematics, Second Edition*. A Wiley-Interscience Publication, 1988.

[160] J. J. Zhang and L. You. Surface representation using second, fourth and mixed order partial differential equations. In *Proceedings of International Conference on Shape Modeling and Applications 2001*, pages 250–256, Genova, Italy, 2001.

[161] H. K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using level set method. In *Proceedings of IEEE Workshop on Variational and Level Set Methods (VLSM 01)*, pages 194–202, Vancouver, Canada, 2001.

[162] H. K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Computer Vision and Image Understanding*, 80(3):295–319, 2000.

[163] D. Zorin, P. Schröder, T. Derose, L. Kobbelt, A. Levin, and W. Sweldens. *Subdivision for Modeling and Animation*. SIGGRAPH 2000 Course Notes 23, 2000.

[164] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 1996*, pages 189–192, New Orleans, USA, 1996.

[165] D. Zorin and P. Shröder. *Subdivision for Modeling and Animation*. SIGGRAPH 2000 Course Notes 36, 2000.