Lecture Notes in Computational Vision and Biomechanics 3

Yongjie (Jessica) Zhang *Editor*

Image-Based Geometric Modeling and Mesh Generation

Springer

Lecture Notes in Computational Vision and Biomechanics

Yongjie (Jessica) Zhang
Editor

Image-Based Geometric Modeling and Mesh Generation

Springer

Editor
Yongjie (Jessica) Zhang
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

ISSN 2212-9391 ISSN 2212-9413 (electronic)
Lecture Notes in Computational Vision and Biomechanics
ISBN 978-94-007-4254-3 ISBN 978-94-007-4255-0 (eBook)
DOI 10.1007/978-94-007-4255-0
Springer Dordrecht Heidelberg New York London

Library of Congress Control Number: 2012941832

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

As a new interdisciplinary research area, “image-based geometric modeling and mesh generation” integrates image processing, geometric modeling and mesh generation with finite element method (FEM) to solve problems in computational biomedicine, materials sciences and engineering. It is well known that FEM is currently well-developed and efficient, but mesh generation for complex geometries (e.g., the human body) still takes about 80% of the total analysis time and is the major obstacle to reduce the total computation time. It is mainly because none of the traditional approaches is sufficient to effectively construct finite element meshes for arbitrarily complicated domains, and generally a great deal of manual interaction is involved in mesh generation.

This contributed volume book, the first for such an interdisciplinary topic, collects the latest research of experts in this area. Of the fourteen invited book chapters, three of them were selected from high quality accepted papers in MeshMed, a workshop on mesh processing in medical image analysis in conjunction with the 14th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) 2011. These papers cover a broad range of topics, including medical imaging, image alignment and segmentation, image-to-mesh conversion, quality improvement, mesh warping, heterogeneous materials, biomolecular modeling and simulation, as well as medical and engineering applications.

We would like to thank all the authors for submitting their excellent research, and also the main organizers of the MeshMed workshop, Drs. Rasmus R. Paulsen and Joshua A. Levine, for their support.

Pittsburgh, USA Jessica Zhang

Contents

Challenges and Advances in Image-Based Geometric Modeling and Mesh Generation 1
Yongjie Zhang

3D Surface Realignment Tracking for Medical Imaging: A Phantom Study with PET Motion Correction 11
Oline V. Olesen, Rasmus R. Paulsen, Rasmus R. Jensen, Sune H. Keller, Merence Sibomana, Liselotte Højgaard, Bjarne Roed, and Rasmus Larsen

Flexible Multi-scale Image Alignment Using B-Spline Reparametrization 21
Yanmei Zheng, Zhucui Jing, and Guoliang Xu

Shape Based Conditional Random Fields for Segmenting Intracranial Aneurysms 55
Sajjad Baloch, Erkang Cheng, and Tong Fang

Tetrahedral Image-to-Mesh Conversion Approaches for Surgery Simulation and Navigation 69
Andrey N. Chernikov, Panagiotis A. Foteinos, Yixun Liu, Michel Audette, Andinet Enquobahrie, and Nikos P. Chrisochoides

Surface Triangular Mesh and Volume Tetrahedral Mesh Generations for Biomolecular Modeling 85
Minxin Chen, Bin Tu, and Benzhuo Lu

A Combined Level Set/Mesh Warping Algorithm for Tracking Brain and Cerebrospinal Fluid Evolution in Hydrocephalic Patients 107
Jeonghyung Park, Suzanne M. Shontz, and Corina S. Drapaca

An Optimization-Based Iterative Approach to Tetrahedral Mesh Smoothing 143
Zhanheng Gao, Zeyun Yu, and Jun Wang

High-Quality Multi-tissue Mesh Generation for Finite Element Analysis . 159
Panagiotis A. Foteinos and Nikos P. Chrisochoides

Construction of Models and Meshes of Heterogeneous Material Microstructures from Image Data 171
Ottmar Klaas, Mark W. Beall, and Mark S. Shephard

Quality Improvement of Segmented Hexahedral Meshes Using Geometric Flows 195
Juelin Leng, Guoliang Xu, Yongjie Zhang, and Jin Qian

Patient-Specific Model Generation and Simulation for Pre-operative Surgical Guidance for Pulmonary Embolism Treatment 223
Shankar P. Sastry, Jibum Kim, Suzanne M. Shontz, Brent A. Craven, Frank C. Lynch, Keefe B. Manning, and Thap Panitanarak

Computational Techniques for Analysis of Shape and Kinematics of Biological Structures 251
Jia Wu and John C. Brigham

Finite Element Modeling of Biomolecular Systems in Ionic Solution . . . 271
Benzhuo Lu

Contributors

Michel Audette Department of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA, USA

Sajjad Baloch Siemens Corporate Research, Princeton, NJ, USA

Mark W. Beall Simmetrix Inc., Clifton Park, NY, USA

John C. Brigham Swanson School of Engineering, University of Pittsburgh, Pittsburgh, PA, USA

Minxin Chen Center for Systems Biology, Department of Mathematics, Soochow University, Suzhou, Jiangsu, China

Erkang Cheng Siemens Corporate Research, Princeton, NJ, USA

Andrey N. Chernikov Department of Computer Science, Old Dominion University, Norfolk, VA, USA

Nikos P. Chrisochoides Department of Computer Science, Old Dominion University, Norfolk, VA, USA

Brent A. Craven Penn State Applied Research Laboratory, University Park, PA, USA

Corina S. Drapaca The Pennsylvania State University, University Park, PA, USA

Andinet Enquobahrie Kitware Inc., Carrboro, NC, USA

Tong Fang Siemens Corporate Research, Princeton, NJ, USA

Panagiotis A. Foteinos Department of Computer Science, College of William and Mary, Williamsburg, VA, USA; Department of Computer Science, Old Dominion University, Norfolk, VA, USA

Zhanheng Gao Department of Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI, USA; College of Computer Science and Technology, Jilin University, Changchun, Jilin, China

Liselotte Højgaard Department of Clinical Physiology, Nuclear Medicine & PET, Rigshospitalet, Copenhagen University Hospital, University of Copenhagen, Copenhagen, Denmark

Rasmus R. Jensen Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark

Zhucui Jing LSEC, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China

Sune H. Keller Department of Clinical Physiology, Nuclear Medicine & PET, Rigshospitalet, Copenhagen University Hospital, University of Copenhagen, Copenhagen, Denmark

Jibum Kim The Pennsylvania State University, University Park, PA, USA

Ottmar Klaas Simmetrix Inc., Clifton Park, NY, USA

Rasmus Larsen Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark

Juelin Leng Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA; ICMSEC, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China

Yixun Liu Department of Computer Science, Old Dominion University, Norfolk, VA, USA

Benzhuo Lu Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China; State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

Frank C. Lynch Penn State Milton S. Hershey Medical Center, Hershey, PA, USA

Keefe B. Manning The Pennsylvania State University, University Park, PA, USA

Oline V. Olesen Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark; Department of Clinical Physiology, Nuclear Medicine & PET, Rigshospitalet, Copenhagen University Hospital, University of Copenhagen, Copenhagen, Denmark; Siemens Healthcare, Siemens A/S, Ballerup, Denmark

Thap Panitanarak The Pennsylvania State University, University Park, PA, USA

Jeonghyung Park The Pennsylvania State University, University Park, PA, USA

Rasmus R. Paulsen Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark

Jin Qian Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

Bjarne Roed Siemens Healthcare, Siemens A/S, Ballerup, Denmark

Shankar P. Sastry The Pennsylvania State University, University Park, PA, USA

Mark S. Shephard Rensselaer Polytechnic Institute, Troy, NY, USA

Suzanne M. Shontz The Pennsylvania State University, University Park, PA, USA

Merence Sibomana Department of Clinical Physiology, Nuclear Medicine & PET, Rigshospitalet, Copenhagen University Hospital, University of Copenhagen, Copenhagen, Denmark

Bin Tu Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

Jun Wang Department of Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI, USA

Jia Wu Swanson School of Engineering, University of Pittsburgh, Pittsburgh, PA, USA

Guoliang Xu ICMSEC, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China; LSEC, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China

Zeyun Yu Department of Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI, USA

Yongjie Zhang Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

Yanmei Zheng LSEC, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China

Challenges and Advances in Image-Based Geometric Modeling and Mesh Generation

Yongjie Zhang

Abstract Image-based geometric modeling and mesh generation play a critical role in computational medicine and biology. This paper presents challenges and advances in this area along with a comprehensive computational framework for analysis-suitable geometric modeling and mesh generation, which integrates image processing, geometric modeling, mesh generation and quality improvement with multi-scale analysis at molecular, cellular, tissue and organ scales. The input imaging data are passed through an image-processing module where the image quality is improved. The improved images are then fed to an in-house meshing software, LBIE-Mesher (Level-set Boundary Interior and Exterior Mesher), to construct 2D or 3D finite element meshes. Given geometry or atomic resolution data in the Protein Data Bank (PDB), we first construct volumetric density map using a signed distance function or a summation of Gaussian Kernel functions, and then use LBIE-Mesher to generate various kinds of meshes. Furthermore, the constructed unstructured meshes can be used as control meshes to construct high-order elements such as volumetric T-splines. In addition, a skeleton-based sweeping method is used to generate hexahedral control meshes and solid NURBS (Non-Uniform Rational B-Spline) or cubic Hermite for cardiovascular system. Different from other existing methods, the presented framework supports five important features: multiscale geometric modeling, automatic mesh generation for heterogeneous domains, all-hexahedral mesh generation with sharp feature preservation, robust quality improvement for non-manifold meshes, and high-order element construction.

Keywords Image-based geometric modeling · Multi-scale modeling · Heterogeneous material · All-hexahedral mesh generation · Sharp feature preservation · Quality improvement · High-order element construction

Y. Zhang (✉)
Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: jessicaz@andrew.cmu.edu

1 Introduction

With finite element method (FEM) and scanning technology seeing increased use in active research areas such as multi-scale modeling and analysis, there is an emerging need for quality mesh generation of the spatially realistic domains that are being studied. In images obtained from Computer Tomography (CT) imaging, Magnetic Resonance Imaging (MRI) or microscopy scanning, the domain of focus often possesses complicated geometry, topology, and sometimes heterogeneous materials at molecular, cellular, tissue and organ scales. For example in Fig. 1, the MRI brain data is segmented into 48 sub-areas, with each colored area demarked as possessing specific characteristic functionality. In finite element analysis, high-fidelity geometric models and quality meshes are needed, with meshes conforming at the boundaries. It is known that FEM is currently well-developed and efficient, but mesh generation for complex geometries still takes approximately 80% of the total analysis time and is the major obstacle to reduce the total computation time.

Image-based mesh generation is a relatively new field. Normally researchers first extract boundary surfaces using isocontouring [5, 9] which usually involves manual interaction, then construct tetrahedral or hexahedral (hex) meshes. The research on mesh generation is dominated by tetrahedral meshing algorithms, which can be grouped into Delaunay triangulation [3, 20], advancing front [7, 8], or grid-based methods [16, 17]. Fewer algorithms exist for automatic all-hex mesh generation due to its intrinsic complexities. But all these methods have limitations. For example, the frequently used, easy to implement block-structured method [2, 10] produces non-conforming boundaries and large number of elements; The grid-based method [14, 15], which puts structured grids inside the volume while adding elements at the boundaries afterward, cannot be extended to all-hex mesh generation for heterogeneous domains with non-manifold boundaries. Today, the key barriers scientists face are:

- A lack of automatic meshing techniques for multi-scale modeling and heterogeneous domains;
- Robust unstructured all-hex mesh generation with sharp feature preservation for complicated geometry and topology is still a challenge;
- The inability of existing methods to effectively improve the quality of non-manifold meshes with feature preservation and topology validation; and
- A lack of high-order element construction techniques for complicated domains.

Many simulations cannot hereby be effectively carried out due to the lack of analysis-suitable meshes.

In this paper, a novel computational framework is presented for analysis-suitable geometric modeling and mesh generation. Starting from scanned images, geometry or atomic resolution data in the Protein Data Bank (PDB), quality 2D/3D meshes as well as high-order elements are constructed. This comprehensive framework supports the following five unique features different from other existing methods:

1. Multi-scale geometric modeling and mesh generation at molecular, cellular, tissue and organ scales;

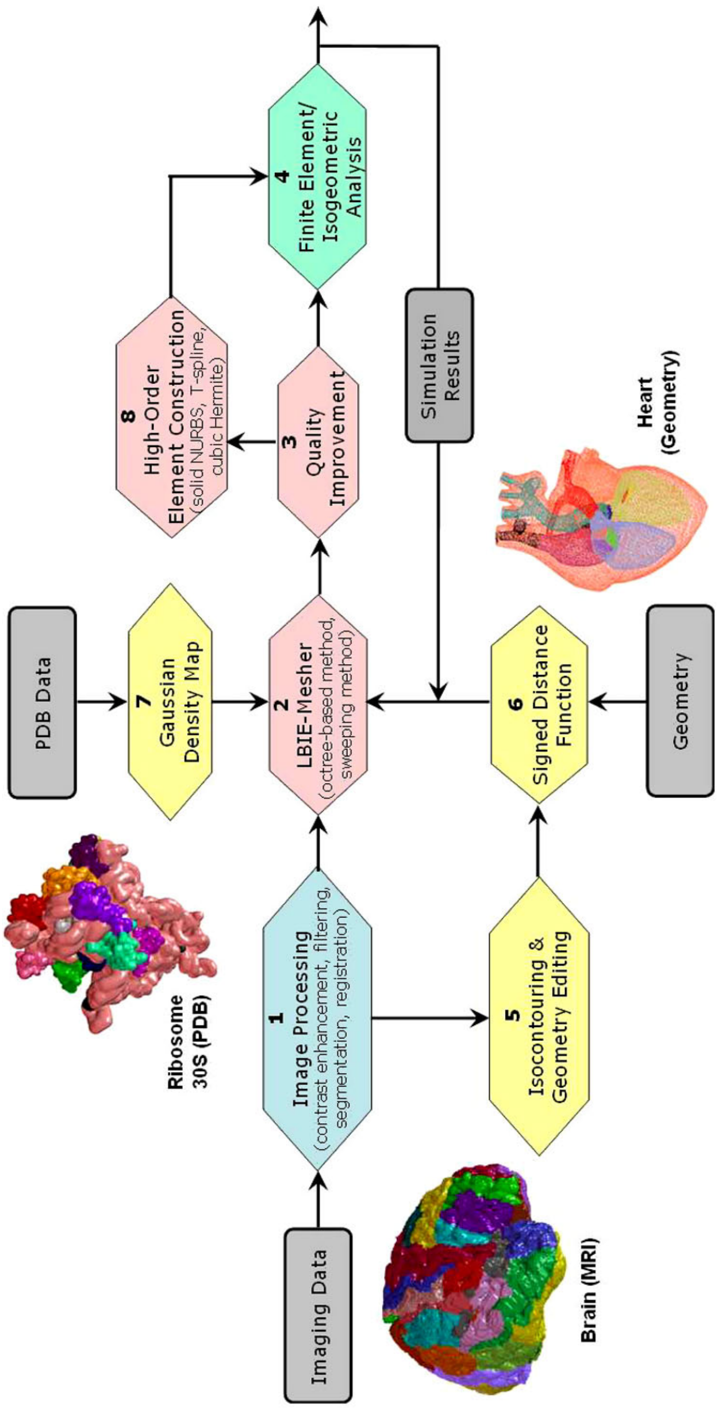

Imaging Data
1
Image Processing
(contrast enhancement, filtering, segmentation, registration)
5
Isocontouring & Geometry Editing
Brain (MRI)
Ribosome 30S (PDB)
PDB Data
7
Gaussian Density Map
2
LBIE-Mesher
(octree-based method, sweeping method)
6
Signed Distance Function
Geometry
Heart (Geometry)
3
Quality Improvement
8
High-Order Element Construction
(solid NURBS, T-spline, cubic Hermite)
Simulation Results
4
Finite Element/ Isogeometric Analysis

Fig. 1 A computational framework with four main pipelines and eight modules for image-based geometric modeling and mesh generation. Taking scanned images, geometry or PDB data as input, the meshing pipelines generate piecewise-linear and high-order elements for analysis

2. Automatic mesh generation for heterogeneous materials with non-manifold boundaries;
3. Unstructured all-hex mesh generation with sharp feature preservation for domains with arbitrarily complicated geometry and topology;
4. Robust quality improvement for non-manifold meshes with volume-preserving and feature preservation; and
5. High-order element construction for complicated domains, e.g., solid NURBS, cubic Hermite and T-spline.

The remainder of this paper is organized as follows. Section 2 presents an overview of the computational framework and then the following sections explain each module in detail. Section 3 discusses piecewise-linear mesh generation, and Sect. 4 describes high-order element construction. Finally, Sect. 5 draws conclusions.

2 Meshing Pipelines

Figure 1 shows the computational framework for mesh generation from volumetric imaging data which are scanned images, or constructed from geometry or PDB data.

2.1 Meshing Pipelines Starting from Scanned Images

The first meshing pipeline (Pipeline 1: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$) starts from imaging data, which are often of poor quality and make it difficult to generate quality meshes for regions of interest. To circumvent this problem we pass the raw images through an image-processing module (Module 1) where the image quality is improved by enhancing the contrast, filtering noise, and segmenting regions of various materials. In order to enhance the image contrast, a stretching function is designed and applied on each individual voxel based on the intensities in a suitable local neighborhood [22]. Noise may exist in the scanned imaging data, therefore we choose a bilateral pre-filtering coupled with an evolution driven anisotropic diffusion equation [1] to remove noise. The imaging data contains heterogeneous materials, here the fast marching method [21] is adopted to find the clear boundary of each material region. Registration is another important image processing technique which geometrically matches two different images. It can be used to track the organ (e.g., lung and heart) motion during breath or match one atlas with a new patient's data.

The improved imaging data is then fed to the meshing software named LBIE-Mesher (Module 2, Level-set Boundary Interior and Exterior Mesher) [23, 24], to construct piecewise-linear meshes using an octree-based isocontouring method or a sweeping method (Module 2, see Sects. 3–4). The mesh is then improved (Module 3) and imported into finite element analysis (Module 4). Mesh adaptation can be controlled by surface features, regions of interest, simulation results or physical domains. The constructed meshes can also be utilized as control meshes to construct high-order elements (Module 8, see Sect. 4).

Sometimes, applying image processing techniques may not be enough to identify the regions of interest. We need to first extract the surface model via isocontouring, and then edit the geometry to suit particular application requirements (Pipeline 2: $1 \to 5 \to 6 \to 2 \to 3 \to 4$). The edited geometry can be converted into volumetric gridded data using the signed distance function method, which puts the geometry into grids, calculate the shortest distance from each grid to the geometry surface, and assign an interior-exterior sign to the distance. LBIE-Mesher then takes the signed distance function data as its input to generate quality meshes for finite element analysis.

2.2 *Meshing Pipelines Starting from Geometry or PDB Data*

Given geometry or atomic resolution data from the PDB (Pipelines 3–4: $6/7 \to 2 \to 3 \to 4$), a volumetric density map is first constructed using a signed distance function (Module 6) or weighted Gaussian isotropic kernel functions coupled with a two-level clustering technique (Module 7) [25]. The latter enables the selection of a smooth implicit solvation surface approximation to the Lee–Richards molecular surface. Efficient and accurate computation of biomolecular surfaces is essential in computational biology. Next, LBIE-Mesher is used to extract 2D/3D meshes for the volume inside or outside the boundary surface but within a bounding sphere/box of influence. Finally, the mesh quality is improved for analysis.

3 Piecewise-Linear Mesh Generation

The imaging data V is a scalar field over sampled rectilinear grids, $V = \{F(i, j, k)|$ i, j, k are indices of x, y, z coordinates in a rectilinear grid$\}$. An isosurface is defined as $S_F(\alpha) = \{(x, y, z)|F(x, y, z) = \alpha\}$, where α represents the corresponding isovalue. In the octree-based method in LBIE-Mesher (Module 2), we analyze edges with two endpoints lying on different sides of an isosurface (sign change edge), or in different material regions (material change edge) for a heterogeneous domain [28]. Each sign/material change edge is shared by four (uniform case) or three (adaptive case) cells. Only one minimizer is calculated for one boundary cell by minimizing a predefined Quadratic Error Function, $QEF(x) = \sum_i (n_i \cdot (x - p_i))^2$ (where p_i, n_i represent the position and unit normal vectors of the intersection point), no matter how many materials are contained in it. For each sign/material change edge, one quadrilateral (quad) or triangle is constructed by connecting these minimizers, which form a manifold/non-manifold boundary and guarantee conformal meshes. Both sign/material changed edges and interior edges are analyzed to generate tetrahedral meshes for the volume of interest. For quad/hex mesh generation, a bottom-up surface topology preserving octree-based algorithm is first applied to select a

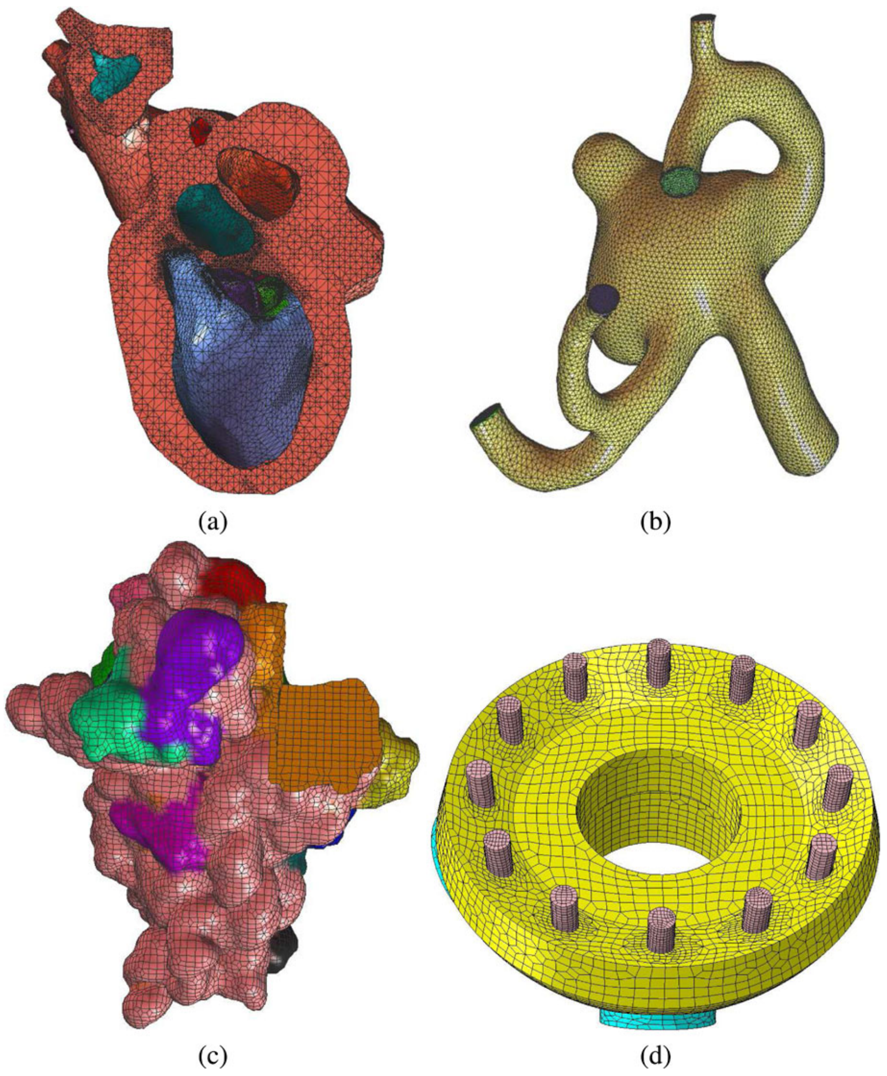


Fig. 2 (**a**–**b**): Tetrahedral meshes for a human heart (organ scale, one cross-section) and cerebral aneurysm (tissue scale); and (**c**–**d**): hex meshes of Ribosome 30S (molecular scale, the *top-right* corner shows cross-sections) and a designed geometry with sharp features

starting octree level, and a preliminary uniform mesh is generated and decomposed into finer elements adaptively without introducing any hanging nodes. Finally, all boundary vertices are projected to the boundary surface.

Figure 2 shows adaptive tetrahedral meshes of a human heart and cerebral aneurysm, as well as hex meshes of Ribosome 30S and a computer designed geometry. The surface diffusion flow [25, 26] is selected to improve the mesh quality because this geometric flow is volume-preserving and it also preserves spherical property accurately when the initial mesh is embedded and close to a sphere, while biomolecules are usually modeled as a union of hard spheres.

Sharp feature may exist in geometry such as medical devices. First, sharp curves and surface patches are extracted and then imported into mesh generation via a curve and surface parametrization [11]. Features shared by multiple material regions (demarked by different colors in Fig. 2(d)) are identified and distinguished. During quality improvement, all the mesh nodes are categorized into several groups and each group is improved differently with feature/volume preservation [6, 13]. The edge contraction and smoothing methods are used for quality improvement of triangular and tetrahedral meshes, and a combination of pillowing, geometric flow [25, 26] and optimization techniques is used for quad and hex meshes.

Challenges In image-based geometric modeling and mesh generation, image processing especially segmentation and deformable registration is still a challenge. The existing segmentation techniques always need more or less user interactions such as seed point selection. Sometimes, it is difficult to automatically detect the clear boundary for the regions of interest and accurately match two images due to the limitation of image resolution and quality. In addition, efficient, accurate and parallel computation of large, multiscale geometric modeling is often the biggest challenge or barrier in a lot of biomedical applications.

Another important challenge is topology preservation for heterogeneous domains with complicated non-manifold boundaries, which have a lot of applications in biomedical and polycrystalline materials such as human brain and beta titanium alloy. How to define the correct topology within a voxel cell especially for cases with ambiguity [12], and how to preserve and validate the topology are still not fully-understood and solved.

4 High-Order Element Construction

In addition to the octree-based method, LBIE-Mesher (Module 2) also supports a skeleton-based sweeping method to generate hexahedral control meshes for cardiovascular solid NURBS construction [27]. First, luminal surfaces are extracted and edited from the segmented images, and then the vascular skeleton is generated via Voronoi and Delaunay diagrams. Following the skeleton, hexahedral control meshes are generated. Templates are designed for various branching configurations to decompose the geometry into mapped meshable patches. Each patch is then meshed using a one-to-one sweeping technique, and boundary vertices are projected to the luminal surface. Finally, solid NURBS are constructed and used in isogeometric analysis [4]. Cubic Hermite can also be constructed from the same control mesh by assigning normal and tangential vectors at each node of one element.

The octree-based method in LBIE-Mesher generates unstructured meshes for any complicated domain. Can these meshes be directly converted to high-order elements such as T-spline? Recently, we started to work on converting any unstructured quad

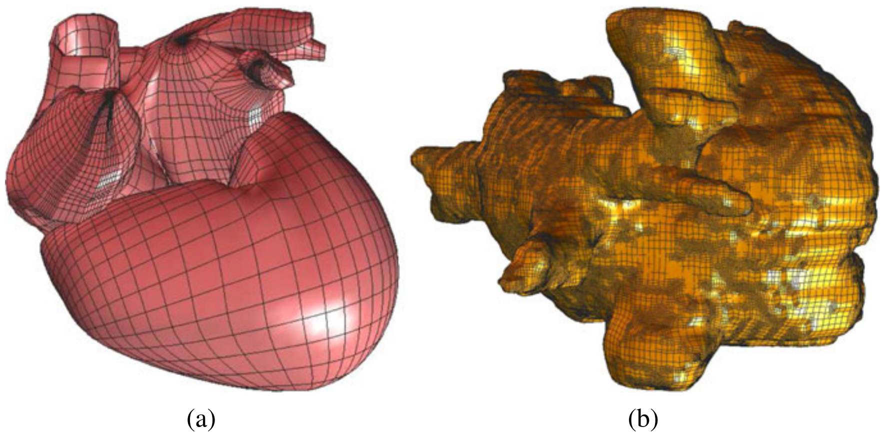


Fig. 3 (**a**): Cubic Hermite of the heart constructed from MRI data; and (**b**): T-spline model of the atria converted from unstructured meshes

or hex meshes to T-spline [18, 19] with C^2-continuous except for the local region around extraordinary nodes. There are two stages in the algorithm: the topology stage and the geometry stage. In the topology stage, templates are designed for each element type, and then generate valid T-meshes. Two sufficient conditions are proved and they serve as a theoretical basis for the template development. In the geometry stage, an efficient surface fitting technique is developed to improve the geometric accuracy. In addition, the surface continuity around extraordinary nodes are improved by adjusting surrounding control nodes. The algorithm can also preserve sharp features in the input mesh. Finally, a Bézier extraction technique is used to facilitate T-spline based isogeometric analysis. In addition, we also recently developed a novel algorithm to construct solid T-splines from boundary representations for genus-zero geometry via a parametric mapping [29]. The obtained T-spline surface is C^2-continuous everywhere except only a few extraordinary nodes.

Figure 3 shows a cubic Hermite model of the heart constructed from MRI data, and a T-spline atria model converted from unstructured meshes. The Hermite surface is C^1-continuous and the T-spline surface is C^2-continuous, except for the local region around extraordinary nodes.

Challenges A good skeleton is very important for the sweeping method. However, the luminal surface is generally noisy and sometimes aneurysm blebs exist. Moreover, the local geometry around a bifurcation or trifurcation is complicated. All these factors hinder producing a good enough skeleton for solid NURBS construction. In addition, the vessel wall-thickness and anisotropic material properties are hard to obtain from measurements, and using inaccurate material properties may produce wrong predictions sometimes.

In recently years, isogeometric analysis [4] has been developed rapidly and significantly matured as a technology combining geometry representation and computational analysis. In addition to NURBS, a standard geometric modeling tool in

CAD and isogeometric analysis, T-splines were introduced as a superior alternative to NURBS allowing for local mesh refinement. However, how to create volumetric models such as solid T-splines in an automatic manner for complicated geometry still remains a challenge. This is an active research area which is in its infancy nowadays. In addition, subdivision and hermite models can also be used in simulations via isogeometric analysis. How to create such volumetric high-order element models with good surface continuity especially around extraordinary nodes also needs a lot of study.

5 Conclusion

This paper presents insights of challenges and advances in image-based geometric modeling and mesh generation along with a comprehensive computational framework consisting of four main pipelines and eight modules. Starting from scanned images, geometry or PDB data, this comprehensive framework generates analysis-suitable piecewise-linear and high-order meshes. Different from other existing methods, this framework supports five important features: multi-scale geometric modeling, heterogeneous domains, all-hex meshing, robust quality improvement with feature preservation, and high-order element construction.

Acknowledgements The author would like to thank her students, J. Ma, J. Qian and W. Wang, for generating pictures for this paper. This research was supported in part by the ONR-YIP award N00014-10-1-0698, an ONR grant N00014-08-1-0653, an AFOSR grant FA9550-11-1-0346, a NSF/DoD-MRSEC seed grant, a UPMC-HTI gift grant, a SINTEF gift grant from Norway, and a UCSD-subaward.

References

1. Bajaj C, Wu Q, Xu G (2003) Level set based volumetric anisotropic diffusion. ICES Technical Report 301, University of Texas-Austin
2. Cailletaud G, Forest S, Jeulin D, Feyel F, Galliet I, Mounoury V, Quilici S (2003) Some elements of microstructural mechanics. Acta Mater 27:351–374
3. Delaunay BN (1934) Sur la sphére vide. Izv Akad Nauk SSSR, Otd Mat Estestvenn Nauk 7:793–800
4. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. Comput Methods Appl Mech Eng 194:4135–4195
5. Ju T, Losasso F, Schaefer S, Warren J (2002) Dual contouring of hermite data. ACM Trans Graph 21(3):339–346
6. Leng J, Zhang Y, Xu G (2011) A novel geometric flow-driven approach for quality improvement of segmented tetrahedral mesh. In: 20th international meshing roundtable, pp 347–364
7. Lo SH (1991) Volume discretization into tetrahedra II. 3D triangulation by advancing front approach. Comput Struct 39:501–511
8. Lohner R (1996) Progress in grid generation via the advancing front technique. Eng Comput 12:186–210

9. Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3D surface construction algorithm. ACM SIGGRAPH Comput Graph 21(4):163–169
10. Nygards M (2003) Number of grains necessary to homogenize elastic materials with cubic symmetry. Mech Mater 35:1049–1057
11. Qian J, Zhang Y (2010) Sharp feature preservation in octree-based all-hexahedral mesh generation for CAD assembly models. In: 19th international meshing roundtable, pp 243–262
12. Qian J, Zhang Y (2011) Dual contouring for domains with topology ambiguity. In: 20th international meshing roundtable, pp 41–60
13. Qian J, Zhang Y, Wang W, Lewis A, Qidwai MAS, Geltmacher A (2010) Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. Int J Numer Methods Eng 82(11):1406–1423
14. Schneiders R (1996) A grid-based algorithm for the generation of hexahedral element meshes. Eng Comput 12:168–177
15. Schneiders R (1996) Refining quadrilateral and hexahedral element meshes. In: 5th international conference on grid generation in computational field simulations, pp 679–688
16. Shephard MS, Georges MK (1991) Three-dimensional mesh generation by finite octree technique. Int J Numer Methods Eng 32:709–749
17. Vavasis SA QMG website. http://www.cs.cornell.edu/home/vavasis/qmg-home.html
18. Wang W, Zhang Y, Scott MA, Hughes TJR (2011) Converting an unstructured quadrilateral mesh to a standard T-spline surface. Comput Mech 48(4):477–498
19. Wang W, Zhang Y, Xu G, Hughes TJR (2011) Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline. Comput Mech. doi:10.1007/s00466-011-0674-6
20. Weatherill NP, Hassan O (1994) Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. Int J Numer Methods Eng 37:2005–2039
21. Yu Z, Bajaj C (2002) Image segmentation using gradient vector diffusion and region merging. In: 16th ICPR, vol 2, pp 941–944
22. Yu Z, Bajaj C (2004) A fast and adaptive algorithm for image contrast enhancement. In: IEEE ICIP, vol 2, pp 1001–1004
23. Zhang Y, Bajaj C (2006) Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Comput Methods Appl Mech Eng 195(9–12):942–960
24. Zhang Y, Bajaj C, Sohn B-S (2005) 3D finite element meshing from imaging data. Comput Methods Appl Mech Eng 194(48–49):5083–5106
25. Zhang Y, Xu G, Bajaj C (2006) Quality meshing of implicit solvation models of biomolecular structures. Comput Aided Geom Des 23(6):510–530
26. Zhang Y, Bajaj C, Xu G (2009) Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. Commun Numer Methods Eng 25(1):1–18
27. Zhang Y, Bazilevs Y, Goswami S, Bajaj C, Hughes TJR (2010) Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Comput Methods Appl Mech Eng 196(29–30):2943–2959
28. Zhang Y, Hughes TJR, Bajaj C (2010) An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Eng 199(5–8):405–415
29. Zhang Y, Wang W, Hughes TJR (2011) Solid T-spline construction from boundary representations for genus-zero geometry. Comput Methods Appl Mech Eng. doi:10.1016/j.cma.2012.01.014

3D Surface Realignment Tracking for Medical Imaging: A Phantom Study with PET Motion Correction

Oline V. Olesen, Rasmus R. Paulsen, Rasmus R. Jensen, Sune H. Keller, Merence Sibomana, Liselotte Højgaard, Bjarne Roed, and Rasmus Larsen

Abstract We present a complete system for motion correction in high resolution brain positron emission tomography (PET) imaging. The system is based on a compact structured light scanner mounted above the patient tunnel of the Siemens High Resolution Research Tomograph (HRRT) PET brain scanner. The structured light system is equipped with a near infrared diode and uses phase-shift interferometry (PSI) to compute 3D point clouds of the forehead of the patient. These 3D point clouds are progressively aligned to a reference surface, thereby giving the head pose changes. The estimated pose changes are used to reposition a sequence of reconstructed PET frames. To align the structured light system with the PET coordinate system, a novel registration algorithm based on the PET transmission scan and an initial surface has been developed. The performance of the complete setup has been evaluated using a custom-made phantom, based on a plastic mannequin head equipped with two positron-emitting line sources. Two experiments were performed. The first simulates rapid and short head movements, while the second simulates slow and continuous movements. In both cases, the system was able to produce PET scans with focused PET reconstructions. The system is nearly ready for clinical testing.

O.V. Olesen (✉) · R.R. Paulsen · R.R. Jensen · R. Larsen
Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 321, 2800 Kgs. Lyngby, Denmark
e-mail: ovol@imm.dtu.dk
url: http://www.imm.dtu.dk/

O.V. Olesen · S.H. Keller · M. Sibomana · L. Højgaard
Department of Clinical Physiology, Nuclear Medicine & PET, Rigshospitalet, Copenhagen University Hospital, University of Copenhagen, Blegdamsvej 9, 2100 Copenhagen, Denmark

O.V. Olesen · B. Roed
Siemens Healthcare, Siemens A/S, Borupvang 3, 2750 Ballerup, Denmark

Y.(J.) Zhang (ed.), *Image-Based Geometric Modeling and Mesh Generation*, Lecture Notes in Computational Vision and Biomechanics 3,
DOI 10.1007/978-94-007-4255-0_2, © Springer Science+Business Media Dordrecht 2013

1 Introduction

Patient head movement during high resolution brain positron emission tomography (PET) scanning will cause blurring and ghosting [3]. The low count rate and resulting low contrast makes it almost impossible to perform motion correction on raw PET data, and therefore most methods rely on external tracking of the head movement [10–12, 16]. The Polaris Vicra (Northern Digital Inc.) tracking system has been used as the reference on many PET installations [6]. While the Polaris system is well tested and accurate, it suffers from problems related to attaching optical markers to the patient's head. Experience shows that in a clinical setting, the markers are difficult to attach such that they stay in position during the entire scan. A markerless system that fits into the narrow PET tunnel will improve the clinical acceptance and the diagnostic value of PET brain scans.

We have previously described a structured light based system that is based on a small projector and two small cameras [7] for tracking patient head pose. This system has been modified as described below and it is referred to as Tracoline. The Tracoline system has been designed to fit into the patient tunnel of the Siemens High Resolution Research Tomograph (HRRT) PET brain scanner. The HRRT PET scanner has a spatial resolution down to 1.4 mm [8] and is therefore well suited for testing new motion correction methods. The Tracoline system is based on the progressive reconstruction of 3D surfaces of the upper face region of the patient in the scanner. The pose changes are found by computing the rigid transformation between the current scan and the initial surface scan. The system described in [7] was based on visible light and did not operate in real time. Visible light scanners are not suited for repeated human facial scans. Furthermore, to be functional the system needs to acquire frames sufficiently fast to faithfully capture patient head movements. In this paper, we describe a system using invisible light with a camera acquisition rate of 30 frames per second.

While the previous paper focused on measuring the accuracy of the structured light tracking system using a rotation stage as ground truth [7], the real interest is the improvement of the PET scans. To be able to evaluate the quality improvement of the PET scan, a scan using a radioactive tracer must be performed. A common approach is to use a phantom and compare the resulting PET scan with the known geometry of the phantom [8]. We have therefore designed a customized phantom with a radioactive source and used this in the evaluation of the Tracoline based motion correction. Our system setup with the Tracoline system and the custom phantom can be seen in Fig. 1. Compared to other external tracking systems, where the geometric alignment between the tracking system and the PET scanner can be problematic, we investigate a novel alignment approach based on aligning the Tracoline system scan directly to the PET transmission scan.

2 Experiments and Methods

The Tracoline system consists of two Point Grey Flea2 cameras (1288×964 pixels), each running at 30 frames per second. The Pico Digital Light Processing (DLP) pro-

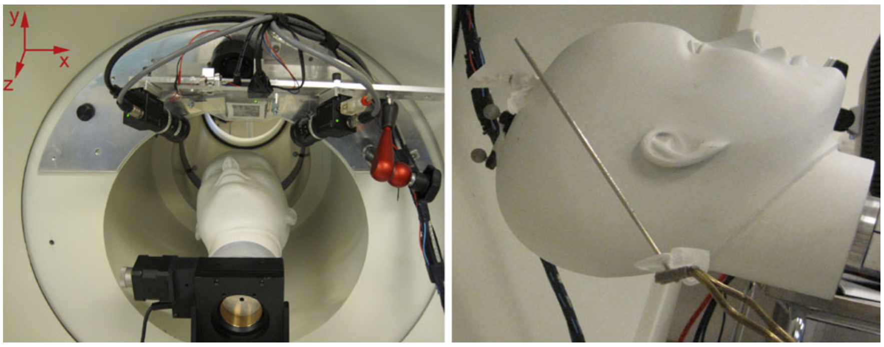


Fig. 1 *Left*: The patient tunnel of the HRRT PET scanner with the Tracoline system mounted. The phantom is mounted on a rotating stage rotated to the right (-10 degrees). *Right*: The phantom including one of the two radioactive line sources. It is placed in the head in the same angle as shown here to simulate the brain

jector from Texas Instruments is used to project phase-modulated patterns. One of the light diodes of the Pico projector has been replaced with a near infrared (NIR) diode resulting in a non-visible projected pattern. The projector is controlled by a GFM Pico developer kit board that also sends trigger pulses to the cameras, in order to synchronize the projected patterns and the shutter of the cameras. A multi-threaded C++ program running on a standard portable computer acquires the real time camera data and stores them as image files. The 3D point cloud generation, surface reconstruction, and alignment are done in a post-processing step. The Tracoline system and HRRT PET acquisition computer are synchronized through an internal network time protocol (NTP) server.

The 3D point cloud generation is based on phase-shifting interferometry (PSI) [5] where a set of 2D interferograms are projected and projector-camera correspondence can be found using phase unwrapping. This is explained in detail in [5, 7]. While three patterns are used in [7], the system is now extended to use six patterns with varying wavelengths to make the phase unwrapping more robust to discontinuities in the surface. Since each point cloud computation requires six frames, the effective tracking frequency is 5 Hz. A surface is reconstructed using a modern algorithm [9] based on the point cloud acquired in the initial position. The facial pose changes are then found by rigidly aligning the following surface scans to this reference surface using an optimized, iterative, closest point (ICP) algorithm [15].

To correct for motion, we need to know the transformation between the HRRT PET scanner coordinate system and the Tracoline system. To estimate this transformation, we use the transmission scan of the HRRT PET scanner, which is also used for the attenuation and scatter correction within normal PET reconstruction. The transmission scan is a voxel volume similar to a computed tomography (CT) scan. The initial reference surface scan is captured by the Tracoline system during the transmission scan, thus creating correspondence. The transformation is computed using a pseudo-ICP algorithm [13]. The surface scan is scaled to fit the volume, and manually rotated and translated into an initial position. To find correspondence between surface sample points and the volume, the volume is sampled in the normal direction (both positive and negative) of the surface scan to find the point with maximum gradient. Knowing the general orientation of the patient in the PET scanner, we use the absolute gradient in the x (left/right) and z (axial) direction and the negative gradient for the y (anterior/posterior) direction:

$$\Delta f = \left|\frac{\partial f}{\partial x}\right| - \frac{\partial f}{\partial y} + \left|\frac{\partial f}{\partial z}\right|$$

With a point correspondence, a rigid transformation is found using the closed-form loop to estimate the absolute transformation [4]. With an initialization, transformation, this process is iterated until the transformation of the Tracoline scan converges to the volume data.

In order to apply the motion correction to the PET data, we apply the multiple acquisition frames (MAF) method [10]. In [8] the MAF method was demonstrated on the Siemens HRRT PET scanner using the tracking input from the Polaris Vicra system. We divide the PET emission list mode data into equal time length intervals and, for each interval, a PET frame is reconstructed using the 3D ordered subset expectation maximization (3D-OSEM) algorithm with resolution modeling and incorporating a spatially invariant point spread function [14]. These frames are then repositioned to a reference position using the Tracoline tracking system.

A custom phantom with known geometry was designed. It consists of a hollow plastic mannequin head with a very low attenuation coefficient. Two radioactive line sources are inserted into the head to provide activity for the HRRT PET scanner. The activities of the line sources are 2×7.8 MBq each, created by a positron-emitting germanium-68/gallium-68 generator. As can be seen in Fig. 1, the line sources go through the head from the back of the skull to the forehead. The phantom was mounted onto a rotation stage by Thorlabs and placed in the patient tunnel of the HRRT PET scanner.

Two experiments where performed using the stage to rotate the head. In experiment one, the head was rotated from -20 to 20 degrees in steps of 5 degrees. At each position a 30 s frame was PET reconstructed and repositioned. Data with motion was excluded from the reconstruction. In experiment two, the head was rotated from -10 to 10 degrees in a continuous motion with a maximum speed of one degree per second. The PET data was reconstructed using one second frames in experiment two. Experiment one simulates the clinical situation where the patient is performing a rapid head motion followed by a stationary period. State of the art practice is to discard PET data during such rapid motions. The second experiment simulates e.g. a patient falling asleep, where the head is slowly drifting from side to side.

We evaluate the effect of the motion correction on the reconstructed PET images by calculating Dice's coefficient (percent volume overlap) [1] between a reference image recorded without phantom motion, the motion distorted image, and the Tracoline based motion-corrected image. The number of voxels, N, included in the calculation is set to a value corresponding to the number of voxels inside the tubes 2.5 times the diameter of the PET sources used (outer diameter 3.2 mm and active

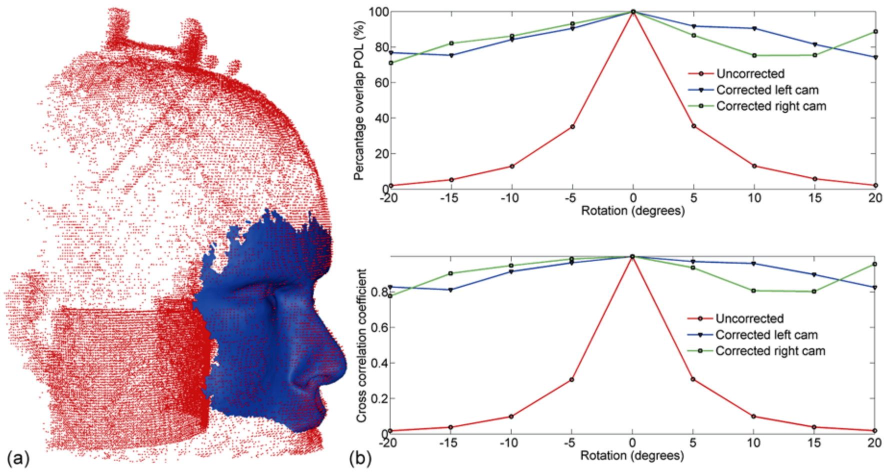


Fig. 2 (**a**) shows the alignment between maximum gradient points in the transmission scan shown as *red dots* and the Tracoline face scan shown as a *blue surface*. (**b**) shows quantitative results of the stepwise experiment. *Top*: the percentage of overlapping points in the reference PET image compared to the unaligned/aligned PET images based on either the right or the left camera. *Bottom*: the cross correlation between the reference PET image and the unaligned/aligned PET images

length 168 mm) [8]. The extended volume is used in order to compensate for partial volume effects. In each image to be studied, the set of the N most intense voxels is extracted and used for the Dice's coefficient computation, presented as the percentage of overlapping voxels. In addition we compute the normalized cross correlation between the reference image and each image frame, either motion-corrected or uncorrected [2].

3 Results and Discussion

The rigid transformation between the coordinate system of the Tracoline system and the PET image frame is obtained from using the described surface-to-volume alignment. Figure 2(a) shows the reference surface aligned to maximum gradient points in the transmission scan.

Figure 2(b) shows the results of the first experiment with stepwise rotation of the phantom. The top figure shows the percentage overlap between PET frames of the line sources in the reference position and a scan position as a function of the performed rotation of the head. Similarly, the bottom plot of Fig. 2(b) shows the correlation coefficient between the reference image and a motion-corrected/uncorrected image for the different scan positions. Results based on tracking information from the left and right camera of the Tracoline system are shown in green and blue colors respectively, while the red curve represents the uncorrected image results. The overlap and the correlation measures are in agreement. The results of the uncorrected frames decrease with the size of performed rotation from an overlap of 100%

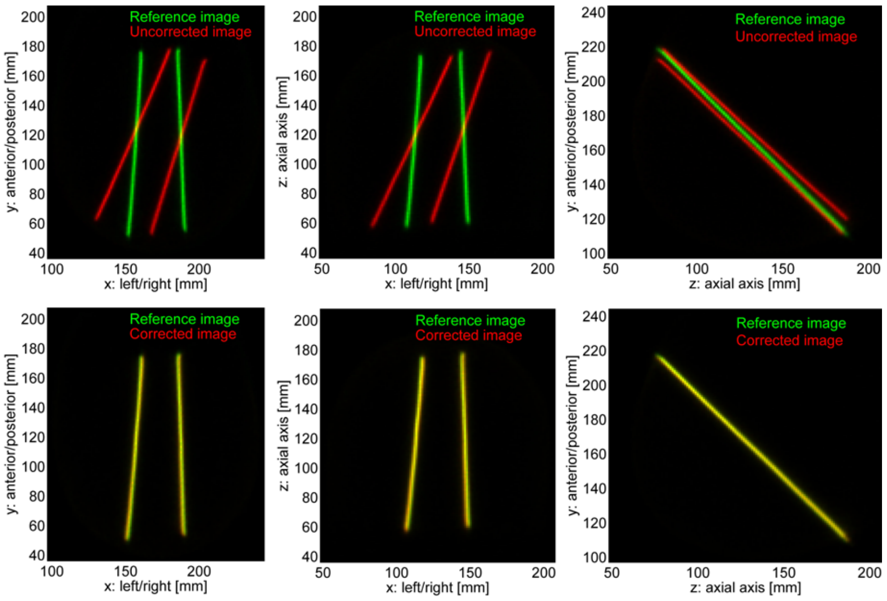


Fig. 3 The figure shows the summation of the PET images along 3 different axes for a reference image (shown in *green*) and a target image (shown in *red*) with a 20 degree rotation (overlap = 89%, shown in *yellow*). The uncorrected image is shown as captured in the *first row*, while the second row shows the image after motion correction

down to 2% at ±20 degrees. The overlap of the motion-corrected reconstruction is improved significantly for all positions with percentage overlap of 71–93%. The overlap is not 100%, which is mainly due to the internal calibration of the Tracoline system, the ICP alignment, and the geometrical alignment with the HRRT PET scanner. In addition, the interpolation error, combined with the straight and narrow line sources of the phantom (with a diameter similar to the voxel size of 1.2 mm), induces partial volume effects and thereby decreases the overlap and correlation measure. The differences between the left camera and the right camera could be explained by the construction of the reference surface scan, where left camera was chosen as the basis. The result is similar to [8], where the overlap was 65–85% for a 10 degrees corrected rotation. However, the two studies cannot be directly compared since the phantom designs are different.

A visual evaluation of the motion correction is shown in Fig. 3 for the maximum rotation of 20 degrees. The PET images are summed along one dimension and visualized on top of each other pairwise in the red and green color channels. The overlapping pixels of the two PET images appear yellow. The top row of the figure shows the reference image and the uncorrected image as two sets of rods rotated approximately 2 cm at the end points. These correspond to the relevant brain regions: the frontal lobe and cerebellum. The bottom row shows the reference image and the corrected image seen as two yellow rods, demonstrating a near-perfect mo-

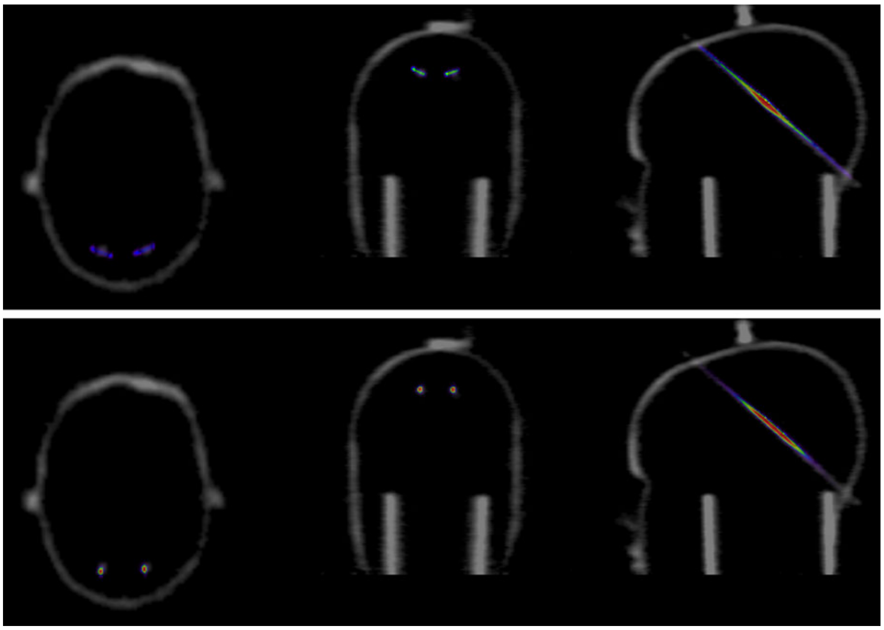


Fig. 4 Results of the dynamic PET scan. One-hundred, one-second frames uncorrected (*top*) and MAF motion corrected (*bottom*) are summed and fused with a transmission scan. The frame repositioning is based on the left camera alignment

tion correction. This position has an overlap of 89% in contrast to the rotation of −20 degrees with an overlap of 71%.

The results of the second experiment, with a continuous rotation of 20 degrees demonstrating the realtime pose registration of the Tracoline system, are presented in Fig. 4. The one second PET frames are summed and fused with the transmission image of the phantom. The top of the figure shows a row of uncorrected images, where the motion of the line sources is seen as blurred circle parts. The bottom row shows the motion-corrected image, where the previously blurred parts appear in focus and with high intensity. The cross section of the line sources shows dots with a diameter of only a few pixels. Long drift motion is a very complex problem to overcome using image registration methods for motion estimation, and this is why an external tracking system is of great value. Our latest results show that continuous motion can be tracked in real time and PET frames successfully corrected.

4 Summary and Conclusions

This paper describes a complete system for motion correction in high resolution PET brain imaging. It is based on a small and flexible structured light scanner mounted above the patient tunnel of the PET scanner. The scanner is equipped with a near infrared light source, making it suitable for future patient examinations. Furthermore, the system tracks the head pose changes with a frequency of 5 Hz, which is suitable for the head movement experienced during real clinical PET scanning. In order to align the two systems, a novel algorithm using the HRRT PET transmission scan and the initial surface scan was presented. The performance of the system was evaluated using a custom-designed phantom with two radioactive line sources mounted on a programmable rotation stage. The results of the two experiments are very promising. The first experiment simulates rapid but short head movements and the second experiment simulates slow but longer head movements. Quantitative analysis shows that the combined system is able to robustly reduce motion artifacts and greatly improve PET scans for scenarios involving both slow and rapid movements. The system is nearly ready for actual clinical testing.

Acknowledgements The John and Birthe Meyer Foundation is thanked for the generous donation of the HRRT PET scanner and the cyclotrons for isotope production.

References

1. Dice LR (1945) Measures of the amount of ecologic association between species. Ecology 26(3):297–302
2. Gonzalez RC, Woods RE (2002) Digital image processing. Prentice Hall, New York
3. Green MV, Seidel J, Stein SD, Tedder TE, Kempner KM, Kertzman C, Zeffiro TA (1994) Head movement in normal subjects during simulated PET brain imaging with and without head restraint. J Nucl Med 35(9):1538–1546
4. Horn BKP (1987) Closed form solution of absolute orientation using unit quaternions. J Opt Soc Am A 4(4):629–642
5. Huang PS, Hu Q, Jin F, Chiang FP (1999) Color-encoded digital fringe projection technique for high-speed three-dimensional surface contouring. Opt Eng 38(6):1065–1071
6. Lopresti BJ, Russo A, Jones WF, Fisher T, Crouch DG, Altenburger DE, Townsend DW (1999) Implementation and performance of an optical motion tracking system for high resolution brain PET imaging. IEEE Trans Nucl Sci 46(6):2059–2067
7. Olesen OV, Paulsen RR, Højgaard L, Roed B, Larsen R (2010) Motion tracking in narrow spaces: a structured light approach. Lect Notes Comput Sci 6363:253–260
8. Olesen OV, Svarer C, Sibomana M, Keller SH, Holm S, Jensen JA, Andersen F, Højgaard L (2010) A movable phantom design for quantitative evaluation of motion correction studies on high resolution PET scanners. IEEE Trans Nucl Sci 57(3):1116–1124
9. Paulsen RR, Bærentzen JA, Larsen R (2010) Markov random field surface reconstruction. IEEE Trans Vis Comput Graph 16:636–646
10. Picard Y, Thompson CJ (1997) Motion correction of PET images using multiple acquisition frames. IEEE Trans Med Imaging 16(2):137–144
11. Raghunath N, Faber TL, Suryanarayanan S, Votaw JR (2009) Motion correction of PET brain images through deconvolution: II. Practical implementation and algorithm optimization. Phys Med Biol 54(3):813
12. Rahmim A, Dinelle K, Cheng JC, Shilov MA, Segars WP, Lidstone SC, Blinder S, Rousset OG, Vajihollahi H, Tsui B, Wong DF, Sossi V (2008) Accurate event-driven motion compensation in high-resolution PET incorporating scattered and random events. IEEE Trans Med Imaging 27(8):1018–1033
13. Rusinkiewicz S, Levoy M (2001) Efficient variants of the ICP algorithm. In: Proc. third international conference on 3-D digital imaging and modeling, pp 145–152

14. Sureau FC, Reader AJ, Comtat C, Leroy C, Ribeiro MJ, Buvat I, Trébossen R (2008) Impact of image-space resolution modeling for studies with the high-resolution research tomograph. J Nucl Med 49(6):1000
15. Wilm J, Olesen O, Paulsen R, Højgaard L, Roed B, Larsen R (2011) Real time surface registration for PET motion tracking. Lect Notes Comput Sci 6688:166–175
16. Woo S-K, Watabe H, Choi Y, Kim KM, Park CC, Bloomfield PM, Iida H (2004) Sinogram-based motion correction of PET images using optical motion tracking system and list-mode data acquisition. IEEE Trans Nucl Sci 51(3):782–788

Flexible Multi-scale Image Alignment Using B-Spline Reparametrization

Yanmei Zheng, Zhucui Jing, and Guoliang Xu

Abstract We present a new flexible alignment method to align two images. By minimizing an energy functional measuring the difference between the initial image and the target image, an L^2-gradient flow is derived for determining a map between the images. The flow is integrated by a finite element method in the spatial direction and an explicit Euler scheme in the temporal direction. Multi-resolution representations are used for achieving efficient multi-scale alignment. The experimental results show that the proposed method is effective, robust and capable of capturing the variation of the initial and target images, from large scale to small. We show that the map of two images in the alignment model is injective and surjective under appropriate conditions, and the solution of the alignment model exists. The results on the existence and uniqueness of the solution for the ordinary differential equation derived from the finite element discretization of our flexible alignment model are established.

1 Introduction

Image alignment (or registration) is a fundamental task in image processing. It refers to the geometric alignment of a set of images. The set may consist of two or more digital images taken from a single scene at different time, different sensors, different viewpoints, or different cross sections of biological tissues. The goal of image alignment is to establish a geometric correspondence between the images so that they can be compared, interpolated for further study. Image alignment has been used in many fields such as medical diagnosis, satellite remote sensing, weather forecast

Y. Zheng · Z. Jing · G. Xu (✉)
LSEC, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100190, China
e-mail: xuguo@lsec.cc.ac.cn

Y. Zheng
e-mail: zhengym@lsec.cc.ac.cn

Z. Jing
e-mail: jingzc@lsec.cc.ac.cn

and computer vision [3, 32]. Basically, alignment methods can be classified into two categories: rigid alignment and flexible alignment. The goal of rigid alignment is to find parameters such as rotation angle, scale parameter and translation components. Rigid alignment has been widely used and studied. Several approaches have been proposed. Some of them are based on the intensity matching, such as correlation (see [26]) and Fourier transform (see [8, 15, 20, 22, 23]). Some others are based on feature matching (see [2, 6, 19, 28]). Flexible alignment [1] aims at finding a correspondence between two images with certain similarities. Compared with the rigid alignment, flexible alignment in general is more difficult.

In [9], a robust and efficient multiscale and multigrid method for the 2D flexible alignment was introduced. In addition, the existence and uniqueness of the solution were proved. The authors of [27] introduced another efficient flexible alignment method. They used bicubic B-splines to model the images, and the deformation field was estimated by solving a minimization problem. The objective function included an energy of the error between both images, the error in the mapping of corresponding landmarks and a regularization term that promoted a smooth deformation. They used the optimization method of Levenberg and Marquardt (see [17, 18]) to solve the problem. They lately added a consistency term into the energy function (see [7]). One important feature of their algorithm was that it was particularly useful when parts of the images contain very little information or when its repartition is uneven. A general review of other flexible alignment methods are given in [11, 13, 24, 25].

In this paper, we assume that the rigid alignment has been conducted, such as using the Fourier transform based method [8]. What we need to find is a correspondence $\mathbf{x}(u, v)$ between two similar images. Given an error metric measuring the similarity of the two images, we first deduce the Euler–Lagrange operator and the geometric flow. Then we obtain the filtered initial image and target image using Gaussian filter with carefully selected standard deviations, and represent $\mathbf{x}$ using B-spline basis functions. After solving the ODE systems consisting of nonlinear equations derived from using the explicit Euler scheme, we get the control points and then the updated $\mathbf{x}$. Through an iterative process we continuously change the initial image to the target image and finally obtain their correspondence.

Many used algorithms were heuristic in nature: no proof was given on their correctness, and no attempt was made at the hypotheses under which they would work or not. We also analyze in this paper our flexible alignment model from a theoretical point of view. Under appropriate conditions on the deformation $\mathbf{x}(u, v)$, we show that it is a one-to-one mapping and surjection. Based a on the well-defined functional space, the solution of the energy model is studied. Furthermore, the existence and uniqueness of the numerical solution are proved.

The main contribution of this paper includes: (i) an efficient multi-scale flexible alignment method that combines the L^2-gradient flow with multi-resolution representations of images. (ii) a method for estimating the temporal step-size in solving the ODE systems. (iii) an estimation method of the standard deviation in the Gaussian filter. (iv) a fast solving approach for the large linear system yielded from solving the ODEs. (v) theoretical analysis on the regularity of the deformation $\mathbf{x}(u, v)$ and on the existence and uniqueness of the solutions of the minimization problem and the ODE systems.

The rest of this paper is organized as follows. We describe the alignment problem and deduce the Euler–Lagrange function and L^2-gradient flows in Sects. 2 and 3, and then construct numerical solving method in Sect. 4. Multi-scale alignment is discussed in Sect. 5. In Sect. 6, we give proof details of the regularity of the mapping $\mathbf{x}(u, v)$. We consider in Sect. 7 the existence problem of the minimizer of the alignment model. In Sect. 8, we discuss the existence and uniqueness problems of the solution for the ordinary system derived from the finite element discretization. We explain the algorithm details and give several experimental results in Sect. 9. Section 10 concludes the paper.

2 Methodology

Problem Description Given two similar images $I_t(u, v)$ (target image) and $I_i(u, v)$ (initial image) in $\mathbb{R}^2$ with the same size defined on $\Omega = [0, 1] \times [0, 1]$. Suppose the size of the images is $(w + 1) \times (h + 1)$. We want to find a smooth mapping $\mathbf{x}(u, v) : \Omega \to \Omega$, satisfying

(i) $\mathbf{x}$ is a C^2 mapping;
(ii) $\mathbf{x}(0, v) = [0, v]^T$, $\mathbf{x}(1, v) = [1, v]^T$, $\mathbf{x}(u, 0) = [u, 0]^T$ and $\mathbf{x}(u, 1) = [u, 1]^T$;
(iii) For an arbitrary given ε $(0 < \varepsilon < 1)$,

$$\det[\mathbf{x}_u, \mathbf{x}_v] \geq \varepsilon \quad \text{on } \Omega, \tag{1}$$

such that

$$\mathcal{E}(\mathbf{x}) = \int_\Omega \left\| I_i\big(\mathbf{x}(u, v)\big) - I_t(u, v) \right\|^2 \mathrm{d}u\,\mathrm{d}v + \lambda \int_\Omega \big(g\big(\mathbf{x}(u, v)\big) - 1\big)^2 \mathrm{d}u\,\mathrm{d}v \tag{2}$$

is minimized, where $g(\mathbf{x}) = g_{11}g_{22} - g_{12}^2$ with $g_{11} = (\mathbf{x}_u)^T\mathbf{x}_u$, $g_{12} = (\mathbf{x}_u)^T\mathbf{x}_v$ and $g_{22} = (\mathbf{x}_v)^T\mathbf{x}_v$. $\mathbf{x}_u$ and $\mathbf{x}_u$ are partial derivatives of $\mathbf{x}(u, v)$ with respect to u and v, respectively. In this paper, we choose $\mathbf{x}(u, v)$ as a bivariate cubic B-spline defined on Ω. For the regularity of the mapping $\mathbf{x}$ and the existence of the solution of the above minimization problem, we have established the following results (the proofs are given in the Sects. 6 and 7).

Theorem 1 $\mathbf{x} : \Omega \to \Omega$ *satisfying* (i)–(iii) *is a one-to-one and surjective mapping.*

Theorem 2 *There exists a mapping* $\mathbf{x}(u, v)$ *satisfying* (i)–(iii) *such that* (2) *is minimized.*

Remark 1 If $\mathbf{x}(u, v)$ is the identity mapping, then $g(\mathbf{x}(u, v)) = 1$. Hence, the regularization term $\int_\Omega (g(\mathbf{x}(u, v)) - 1)^2\, \mathrm{d}u\,\mathrm{d}v$ can insure that the mapping $\mathbf{x}(u, v)$ is not far away from the identity mapping.

3 B-Spline Reparametrization by L^2-Gradient Flow

Now we construct an L^2-gradient flow to minimize the energy functional $\mathcal{E}(\mathbf{x})$ defined by (2). Let

$$\underline{\mathbf{x}}(u,v,\varepsilon) = \mathbf{x} + \varepsilon\Phi(u,v) : u,v \in \Omega, \quad \Phi \in C_0^1(\Omega)^2.$$

Then we have

$$\delta\big(\mathcal{E}(\mathbf{x}),\Phi\big) = \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\mathcal{E}\big(\underline{\mathbf{x}}(\cdot,\varepsilon)\big)\Big|_{\varepsilon=0},$$

where

$$\delta\big(\mathcal{E}(\mathbf{x}),\Phi\big) = 2\int_\Omega \big[\big[I_i(\mathbf{x}) - I_t\big](\nabla_{\mathbf{x}} I_i)^T\delta(\mathbf{x})\big]\,\mathrm{d}u\,\mathrm{d}v$$
$$+ 2\lambda\int_\Omega \big(g(\mathbf{x}) - 1\big)\delta(g)\,\mathrm{d}u\,\mathrm{d}v.$$

It follows from

$$\underline{\mathbf{x}} = \mathbf{x} + \varepsilon\Phi, \tag{3}$$

we have

$$\delta(\mathbf{x}) = \Phi, \qquad \delta(g) = 2\Phi_u^T(g_{22}\mathbf{x}_u - g_{12}\mathbf{x}_v) + 2\Phi_v^T(g_{11}\mathbf{x}_v - g_{12}\mathbf{x}_u).$$

Hence

$$\delta\big(\mathcal{E}(\mathbf{x}),\Phi\big) = 2\int_\Omega \big[\big[I_i(\mathbf{x}) - I_t\big](\nabla_{\mathbf{x}} I_i)^T\Phi\big]\,\mathrm{d}u\,\mathrm{d}v$$
$$+ 2\lambda\int_\Omega \big(\Phi_u^T\alpha + \Phi_v^T\beta\big)\,\mathrm{d}u\,\mathrm{d}v, \tag{4}$$

where

$$\alpha = 2\big(g(\mathbf{x}) - 1\big)(g_{22}\mathbf{x}_u - g_{12}\mathbf{x}_v), \qquad \beta = 2\big(g(\mathbf{x}) - 1\big)(g_{11}\mathbf{x}_v - g_{12}\mathbf{x}_u).$$

To construct L^2-gradient flows moving $\mathbf{x}$ in the tangential directions $D_l\mathbf{x}$, $l = 1,2$, we take

$$\Phi = (D_l\mathbf{x})(D_l\mathbf{x})^T\phi, \quad \phi \in C_0^1(\Omega), l = 1,2, \tag{5}$$

where $D_1\mathbf{x} = \mathbf{x}_u$, $D_2\mathbf{x} = \mathbf{x}_v$. Therefore, we construct the following weak-form L^2-gradient flows moving $\mathbf{x}$ in the $D_l\mathbf{x}$ directions

$$\int_\Omega \frac{\partial\mathbf{x}}{\partial t}\phi\,\mathrm{d}u\,\mathrm{d}v$$
$$= -2\int_\Omega \big[\big[I_i(\mathbf{x}) - I_t\big](D_l\mathbf{x})^T(\nabla_{\mathbf{x}} I_i)(D_l\mathbf{x})\phi\big]\,\mathrm{d}u\,\mathrm{d}v$$

$$
\begin{aligned}
&- 2\lambda \int_{\Omega} \big[\big(D_l\mathbf{x}^T \boldsymbol{\alpha}\mathbf{I}_2 + D_l\mathbf{x}\boldsymbol{\alpha}^T\big) D_{l1}\mathbf{x}\phi + \big(D_l\mathbf{x}^T \boldsymbol{\alpha}\big) D_l\mathbf{x} D_1\phi \\
&+ \big(D_l\mathbf{x}^T \boldsymbol{\beta}\mathbf{I}_2 + D_l\mathbf{x}\boldsymbol{\beta}^T\big) D_{l2}\mathbf{x}\phi + \big(D_l\mathbf{x}^T \boldsymbol{\beta}\big) D_l\mathbf{x} D_2\phi\big] \,\mathrm{d}u\,\mathrm{d}v, \qquad (6)
\end{aligned}
$$

where $l = 1, 2$, and

$$
D_{11}\mathbf{x} = \mathbf{x}_{uu}, \qquad D_{12}\mathbf{x} = D_{21}\mathbf{x} = \mathbf{x}_{uv}, \qquad D_{22}\mathbf{x} = \mathbf{x}_{vv},
$$

$\mathbf{I}_2$ represents the 2×2 unit matrix. Using the fact that $g(\mathbf{x}) = g_{11}g_{22} - g_{12}^2$ with $g_{11} = (\mathbf{x}_u)^T\mathbf{x}_u$, $g_{12} = (\mathbf{x}_u)^T\mathbf{x}_v$ and $g_{22} = (\mathbf{x}_v)^T\mathbf{x}_v$, we can rewrite (6) as

$$
\begin{aligned}
\int_{\Omega} \frac{\partial \mathbf{x}}{\partial t}\phi \,\mathrm{d}u\,\mathrm{d}v = &-2 \int_{\Omega} \big[\big[I_i\big(\mathbf{x}(u,v)\big) - I_t(u,v)\big](D_l\mathbf{x})^T(\nabla_{\mathbf{x}} I_i)(D_l\mathbf{x})\phi\big] \,\mathrm{d}u\,\mathrm{d}v \\
&- 2\lambda \int_{\Omega} \big[\big(\gamma D_{ll}\mathbf{x} + D_l\mathbf{x}D_{l1}\mathbf{x}^T\alpha + D_l\mathbf{x}D_{l2}\mathbf{x}^T\beta\big)\phi \\
&+ \gamma D_l\mathbf{x}D_l\phi\big] \,\mathrm{d}u\,\mathrm{d}v, \qquad (7)
\end{aligned}
$$

where $\gamma = 2(g(\mathbf{x}) - 1)g(\mathbf{x})$.

Remark 2 Let us explain the reason why we take Φ as (5). Taking $\Phi = D_l\mathbf{x}\phi$, we obtain the Euler–Lagrange operator for the first term of (4) as

$$
2\big[I_i(\mathbf{x}) - I_t\big](\nabla_{\mathbf{x}} I_i)^T D_l\mathbf{x}.
$$

Hence the L^2-gradient flow, for the first term of (2), moving $\mathbf{x}$ in the direction $D_l\mathbf{x}$ is

$$
\frac{\partial \mathbf{x}}{\partial t} = -2\big[I_i(\mathbf{x}) - I_t\big]\big[(\nabla_{\mathbf{x}} I_i)^T D_l\mathbf{x}\big]D_l\mathbf{x}. \qquad (8)
$$

The weak-form of this equation is the first term of the right-hand side of (6). This is the same as taking $\Phi = (D_l\mathbf{x})(D_l\mathbf{x})^T\phi$ in (4).

4 Numerical Solutions

In this section, we propose a few solving techniques for Eq. (7).

4.1 Spacial Discretization

We solve Eq. (7) in a bicubic B-spline vector-valued function space. Given two positive integers m and n, let

$$
\mathbf{x}(u,v) = \sum_{i=0}^{m+2}\sum_{j=0}^{n+2} \mathbf{p}_{ij} N_i^{(1/m)}(u) N_j^{(1/n)}(v),
$$

where $N_i^{(1/m)}(u)$ and $N_j^{(1/n)}(v)$ are one-dimensional cubic B-spline basis functions defined on the knots

$$\left[0, 0, 0, 0, \frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}, 1, 1, 1, 1\right]$$

and

$$\left[0, 0, 0, 0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, 1, 1, 1, 1\right],$$

respectively. $\mathbf{p}_{ij}$ is the corresponding two-dimensional control point.

For easy of description, we reorder the control points $\mathbf{p}_{ij}$ of the B-spline mapping into a one-dimensional array and represent them as:

$$\mathbf{x}_0, \ldots, \mathbf{x}_{n_0}, \mathbf{x}_{n_0+1}, \ldots, \mathbf{x}_{n_1},$$

where $\mathbf{x}_0, \ldots, \mathbf{x}_{n_0}$ are the control points $\mathbf{p}_{ij}$ with $0 < i < m+2$ when $l = 1$, and $0 < j < n+2$ when $l = 2$. The remaining control points are $\mathbf{x}_{n_0+1}, \ldots, \mathbf{x}_{n_1}$. It is easy to see that

$$\begin{aligned} n_0 &= (m+1)(n+3) - 1, \quad \text{if } l = 1, \\ n_0 &= (m+3)(n+1) - 1, \quad \text{if } l = 2, \\ n_1 &= (m+3)(n+3) - 1. \end{aligned}$$

The B-spline basis functions $N_i^{(1/m)}(u)N_j^{(1/n)}(v)$ are correspondingly reordered and represented as

$$\phi_0, \ldots, \phi_{n_0}, \phi_{n_0+1}, \ldots, \phi_{n_1}.$$

Using this ordering of the basis functions and control points, mapping $\mathbf{x}$ can be represented as

$$\mathbf{x}(u, v) = \sum_{j=0}^{n_0} \mathbf{x}_j \phi_j(u, v) + \sum_{j=n_0+1}^{n_1} \mathbf{x}_j \phi_j(u, v). \tag{9}$$

Substituting $\mathbf{x}(u, v)$ into (7), and then taking the test function ϕ as ϕ_i, for $i = 0, \ldots, n_0$, we can discretize (7) as a system of the ordinary differential equations (ODE) with the control points $\mathbf{x}_j$, $j = 0, \ldots, n_0$, as unknowns.

$$\sum_{j=0}^{n_0} m_{ij} \frac{\mathrm{d}\mathbf{x}_j(t)}{\mathrm{d}t} = -\mathbf{q}_i^{(l)}, \quad i = 0, \ldots, n_0, l = 1, 2, \tag{10}$$

where

$$m_{ij} = \int_\Omega \phi_i \phi_j \,\mathrm{d}u\,\mathrm{d}v, \tag{11}$$

$$\begin{aligned}\mathbf{q}_i^{(l)} &= 2\int_\Omega \big[\big[I_i\big(\mathbf{x}(u,v)\big) - I_t(u,v)\big](D_l\mathbf{x})^T(\nabla_{\mathbf{x}} I_i)(D_l\mathbf{x})\phi_i\big]\,\mathrm{d}u\,\mathrm{d}v \\ &\quad + 2\lambda \int_\Omega \big[\big(\gamma D_{ll}\mathbf{x} + D_l\mathbf{x}D_{l1}\mathbf{x}^T\alpha + D_l\mathbf{x}D_{l2}\mathbf{x}^T\beta\big)\phi_i \\ &\quad + \gamma D_l\mathbf{x}D_l\phi_i\big]\,\mathrm{d}u\,\mathrm{d}v. \end{aligned} \tag{12}$$

For the existence and uniqueness of the solution of system (10), we have the following result (the proof is given in Sect. 8).

Theorem 3 *There exists a unique solution of the problem* (10) *for a given initial mapping* $\mathbf{x}^0(u,v)$ *satisfying* (i)–(iii).

4.2 Temporal Discretization

For the temporal direction discretization of the ODE systems (10), we use the forward Euler scheme

$$\frac{\mathrm{d}\mathbf{x}_j(t)}{\mathrm{d}t} \approx \frac{\mathbf{x}_j^{(s)} - \mathbf{x}_j^{(s-1)}}{\tau_l}, \tag{13}$$

where τ_l is a temporal step-size, s is the iteration number.

Compute τ_l We can set a fixed temporal step-size τ_l in advance, but an arbitrarily chosen τ_l may not be suitable for specified images. It may cause condition (1) invalid if it is too big, or it leads to too much running time if it is too small. The ideal strategy is to compute τ_l according to the specific characteristics of the images. We first define

$$\mathbf{y}_j^{(l)} = \frac{\mathrm{d}\mathbf{x}_j(t)}{\mathrm{d}t}, \quad j = 0,\ldots,n_0, l = 1,2,$$

and solve the linear system (10) for the unknowns $\mathbf{y}_j^{(l)}$, and then let

$$\delta_l^{(s)}(u,v) = \sum_{j=0}^{n_0} \mathbf{y}_j^{(l)}\phi_j(u,v), \quad l = 1,2.$$

Using the increment $\delta_l^{(s)}(u,v)$, we can define a τ_l such that

$$\int_\Omega \big[I_i\big(\mathbf{x}^{(s,l-1)} + \tau_l\delta_l^{(s)}\big) - I_t\big]^2\,\mathrm{d}u\,\mathrm{d}v = \min.$$

From this, we can derive that

$$\tau_l = -\int_{\Omega}\left[I_i\left(\mathbf{x}^{(s,l-1)}\right) - I_t\right]\left(\nabla_{\mathbf{x}} I_i\left(\mathbf{x}^{(s,l-1)}\right)\right)^T \delta_l^{(s)}\, \mathrm{d}u\, \mathrm{d}v$$
$$\Big/ \int_{\Omega}\Big[\left[\left(\nabla_{\mathbf{x}} I_i\left(\mathbf{x}^{(s,l-1)}\right)\right)^T \delta_l^{(s)}\right]^2$$
$$+ \left[I_i\left(\mathbf{x}^{(s,l-1)}\right) - I_t\right]\left(\delta_l^{(s)}\right)^T \nabla_{\mathbf{x}}^2 I_i\left(\mathbf{x}^{(s,l-1)}\right)\delta_l^{(s)}\Big]\, \mathrm{d}u\, \mathrm{d}v. \tag{14}$$

Using the inverse of the matrix $[m_{ij}]$ (see Sect. 4.3) to solve the linear systems (10) for $l = 1, 2$, we obtain $\frac{\mathrm{d}\mathbf{x}_j(t)}{\mathrm{d}t}$ and the new control points of $\mathbf{x}$ from (13). We treat the right-hand terms in (10) as the known quantities, and obtain the following iterative algorithm.

Algorithm 1 (Explicit finite element method)

1. Set $s = 0$ and the initial B-spline representation of $\mathbf{x}^{(0)}(u, v)$ such that

$$\mathbf{x}^{(0)}(u, v) = \sum_j \mathbf{x}_j^{(0)} \phi_j(u, v) = [u, v]^T.$$

2. Set $\mathbf{x}_j^{(s,0)} = \mathbf{x}_j^{(s)}$, $j = 0, \ldots, n_1$.
3. For $l = 1, 2$, do the following
   a. Compute $\{\mathbf{q}_i^{(l)}\}$ using (12).
   b. Solve the linear system

$$\sum_{j=0}^{n_0} m_{ij}\mathbf{y}_j^{(l)} = -\mathbf{q}_i^{(l)}, \quad i = 0, \ldots, n_0, l = 1, 2, \tag{15}$$

   for the unknowns $\mathbf{y}_j^{(l)}$ using the previously computed $[m_{ij}]^{-1}$.
   c. Compute τ_l using (14), then compute

$$\mathbf{x}_j^{(s,l)} = \mathbf{x}_j^{(s,l-1)} + \tau_l \mathbf{y}_j^{(l)}, \quad j = 0, \ldots, n_0. \tag{16}$$

4. Set $\mathbf{x}_j^{(s+1)} = \mathbf{x}_j^{(s,2)}$, check the terminate condition:

$$\max_j \left\| \mathbf{x}_j^{(s+1)} - \mathbf{x}_j^{(s)} \right\| < \varepsilon.$$

If it is satisfied, stop the iteration; otherwise, set s to be $s + 1$ and return to step 3.

Remark 3 After obtaining τ_l from (14), we first get the test control points $\mathbf{x}_j^{new}$ by

$$\mathbf{x}_j^{new} = \mathbf{x}_j^{(s,l-1)} + \tau_l \mathbf{y}_j^{(l)}, \quad j = 0, \ldots, n_0, \tag{17}$$

$$\mathbf{x}_j^{new} = \mathbf{x}_j^{(s,l-1)}, \quad j = n_0 + 1, \ldots, n_1. \tag{18}$$

Then get the test mapping $\mathbf{x}^{new}(u, v)$ by (9) with control points $\mathbf{x}_j^{new}$. Finally check whether $\mathbf{x}^{new}(u, v)$ satisfies regularity constraint (1). If (1) is satisfied, use this τ_l. Otherwise, decrease τ_l with the factor 0.618 until $\mathbf{x}^{new}(u, v)$ satisfies (1).

4.3 Calculation of Coefficient Matrix of (10) and Its Inverse

Let $N_i(t)$, $i = 1, \dots, m$, $\bar{N}_j(t)$, $j = 1, \dots, n$, be any given two sets of basis functions defined on $\mathbb{R}$. Let

$$\phi_{(i-1)n+j}(x, y) = N_i(x)\bar{N}_j(y), \quad i = 1, \dots, m, j = 1, \dots, n,$$

be a set of two dimensional basis functions defined on the xy-plane. Define

$$d_{\alpha\beta} = \int_{\mathbb{R}^2} \phi_\alpha(x)\phi_\beta(y)\, \mathrm{d}x\, \mathrm{d}y, \quad \alpha, \beta = 1, \dots, mn,$$

and the matrix $D = [d_{\alpha\beta}]_{\alpha,\beta=1}^{mn}$, we want to compute D^{-1} efficiently. Suppose

$$\begin{aligned} \alpha &= (i-1)n + j, \quad 1 \le i \le m, 1 \le j \le n, \\ \beta &= (\bar{i}-1)n + \bar{j}, \quad 1 \le \bar{i} \le m, 1 \le \bar{j} \le n. \end{aligned}$$

Then

$$\begin{aligned} i &= E\big[(\alpha-1)/n\big] + 1, \quad j = \alpha - (i-1)n, \\ \bar{i} &= E\big[(\beta-1)/n\big] + 1, \quad \bar{j} = \beta - (\bar{i}-1)n, \end{aligned}$$

where $E[\cdot]$ denotes taking integer part of a real number. Then we have

$$d_{\alpha\beta} = \int_{\mathbb{R}} N_i(x)N_{\bar{i}}(x)\, \mathrm{d}x \int_{\mathbb{R}} \bar{N}_j(y)\bar{N}_{\bar{j}}(y)\, \mathrm{d}y = c_{i\bar{i}}\bar{c}_{j\bar{j}},$$

with

$$c_{i\bar{i}} = \int_{\mathbb{R}} N_i(x)N_{\bar{i}}(x)\, \mathrm{d}x, \qquad \bar{c}_{j\bar{j}} = \int_{\mathbb{R}} \bar{N}_j(y)\bar{N}_{\bar{j}}(y)\, \mathrm{d}y.$$

Therefore, D can be written as

$$\begin{aligned} D &= \begin{bmatrix} c_{11}\bar{C} & c_{12}\bar{C} & \cdots & c_{1m}\bar{C} \\ c_{21}\bar{C} & c_{22}\bar{C} & \cdots & c_{2m}\bar{C} \\ \cdots & \cdots & \cdots & \cdots \\ c_{m1}\bar{C} & c_{m2}\bar{C} & \cdots & c_{mm}\bar{C} \end{bmatrix} \\ &= C \otimes \bar{C} = (C \otimes I_n)\,\mathrm{diag}[\bar{C}, \bar{C}, \dots, \bar{C}], \end{aligned} \tag{19}$$

Table 1 The values of c_{ij} when $m > 4$

$i = 0$	$c_{i0} = \frac{1}{7}$	$c_{i1} = \frac{7}{80}$	$c_{i2} = \frac{31}{1680}$	$c_{i3} = \frac{1}{840}$	$c_{ij} = 0, j > 3$
$i = 1$	$c_{i1} = \frac{31}{140}$	$c_{i2} = \frac{5}{32}$	$c_{i3} = \frac{29}{840}$	$c_{i4} = \frac{1}{3360}$	$c_{ij} = 0, j > 4$
$i = 2$	$c_{i2} = \frac{183}{560}$	$c_{i3} = \frac{283}{1260}$	$c_{i4} = \frac{239}{10080}$	$c_{i5} = \frac{1}{5040}$	$c_{ij} = 0, j > 5$
$2 < i < m - 4$	$c_{ii} = \frac{151}{315}$	$c_{i,i+1} = \frac{397}{1680}$	$c_{i,i+2} = \frac{1}{42}$	$c_{i,i+3} = \frac{1}{5040}$	$c_{i,i+j} = 0, j > 3$

where $\otimes$ denotes the Kronecker product of two matrices, I_n stands for the $n \times n$ unit matrix, and $C = [c_{ij}]_{i,j=1}^{m}$, $\bar{C} = [\bar{c}_{ij}]_{i,j=1}^{n}$. Using (19), D^{-1} can be computed as

$$D^{-1} = C^{-1} \otimes \bar{C}^{-1} = \left(C^{-1} \otimes I_n\right) \operatorname{diag}\left[\bar{C}^{-1}, \bar{C}^{-1}, \ldots, \bar{C}^{-1}\right].$$

Hence, we only need to inverse two small-sized matrices C and $\bar{C}$. The computational complexity for computing D^{-1} is $O(m^3) + O(n^3)$. The cost for computing the multiplication of D^{-1} and a vector in $\mathbb{R}^{mn}$ is $O(mn^2) + O(m^2n)$. For the cubic B-spline basis functions, c_{ij} can be computed exactly. Now we consider two cases.

Example 1 If $N_i(t)$, $i = 1, \ldots, m$, are the cubic B-spline basis functions defined on the uniform knots with spacing 1, then

$$c_{ij} = \begin{cases} \frac{151}{315}, & i = j, \\ \frac{397}{1680}, & |i - j| = 1, \\ \frac{1}{42}, & |i - j| = 2, \\ \frac{1}{5040}, & |i - j| = 3, \\ 0, & |i - j| \geq 4. \end{cases}$$

Hence C is a banded-matrix with band width 7. If N_i are cubic B-spline basis functions defined on the uniform knots with spacing h, then c_{ij} have a factor h.

Example 2 If $N_0(t), N_1(t), \ldots, N_{m+2}(t)$ are the cubic B-spline basis on the knots

$$[0, 0, 0, 0, 1, 2, \ldots, m - 1, m, m, m, m].$$

The exact values of c_{ij} can be calculated using the closed form representations of the B-spline basis. We put these values when $m > 4$ in Table 1. Using the relations $c_{m+2-i,m+2-j} = c_{ij}$ and $c_{ji} = c_{ij}$, all the c_{ij} can be obtained from this table. The values c_{ij} for the case $m \leq 4$ can be similarly calculated.

For the cubic B-spline basis $N_i^{(h)}(t) = N_i(h^{-1}t)$, defined on the knots

$$\left[0, 0, 0, 0, h, 2h, \ldots, (m - 1)h, mh, mh, mh, mh\right],$$

we obviously have

$$c_{ij}^{(h)} := \int_{\mathbb{R}} N_i^{(h)}(t) N_j^{(h)}(t) \, \mathrm{d}t = hc_{ij}.$$

If $h = \frac{1}{m}$, then $c_{ij}^{(1/m)} = \frac{c_{ij}}{m}$.

5 Alignment Based on Multi-resolution Representations

In this section, we intend to achieve a multi-scale alignment using the multi-resolution representations of the images to be aligned. We also deduce a relationship between knots number of the used B-splines and the standard deviation σ of the Gaussian filter.

5.1 Multi-resolution Representations

Given two images I_i and I_t with the size of $(w+1) \times (h+1)$. To be able to align them effectively from large structures to fine details, we use multi-resolution representations of I_i and I_t. Let $(w_0, h_0), \ldots, (w_K, h_K)$ be a sequence (we call it N-sequence for simplicity) of positive integer pairs satisfying

$$w_0 = w, \qquad h_0 = h, \qquad w_i > w_{i+1}, \qquad h_i > h_{i+1}, \quad i = 0, 1, \ldots, K-1.$$

Each pair (w_i, h_i) is used to define B-spline representations of image I_i and I_t with the interval numbers w_i and h_i in the u and v directions, respectively. In our implementation, two types of N-sequence are used. The first one is

$$w_i = E\big[w/\lambda_w^i\big], \qquad h_i = E\big[h/\lambda_h^i\big], \quad i = 0, 1, \ldots, K, \tag{20}$$

where $\lambda_w > 1$ and $\lambda_h > 1$ are the factors in the geometric proportional sequences, $E[\cdot]$ denotes taking integer part operation. The second N-sequence is

$$w_i = w - \rho_w i, \qquad h_i = h - \rho_h i, \quad i = 0, 1, \ldots, K, \tag{21}$$

where ρ_w and ρ_h are the factors in the arithmetic sequences.

We refer a bicubic B-spline function defined on the knots

$$\left[0, 0, 0, 0, \frac{1}{w_k}, \frac{2}{w_k}, \ldots, \frac{w_k - 1}{w_k}, 1, 1, 1, 1\right] \\ \times \left[0, 0, 0, 0, \frac{1}{h_k}, \frac{2}{h_k}, \ldots, \frac{h_k - 1}{h_k}, 1, 1, 1, 1\right]$$

as (w_k, h_k)-level bicubic B-spline function. Now we describe an alignment algorithm based on the multi-resolution representations.

Algorithm 2 (Multi-resolution alignment)

1. For a given $K > 0$, set $k = K$ and $I_t^{(k+1)} = I_i$.

2. Compute (w_k, h_k)-level bicubic B-spline approximations $\tilde{I}_i^{(k)}$ and $\tilde{I}_t^{(k)}$ of $I_i^{(k+1)}$ and I_t, respectively, by smoothing $I_i^{(k+1)}$ and I_t using Gaussian filter with deviation σ_k (see Sect. 5.2) and then converting the smoothed images to B-spline representations in the least square sense (see Sect. 5.3).
3. Compute $\mathbf{x}^{(k)}(u, v)$, which is a (w_k, h_k)-level bicubic B-spline vector-valued function in $\mathbb{R}^2$, such that

$$\int_\Omega \left[\tilde{I}_i^{(k)}\big(\mathbf{x}^{(k)}(u,v)\big) - \tilde{I}_t^{(k)}(u,v)\right]^2 \mathrm{d}u\,\mathrm{d}v + \lambda \int_\Omega \left[g\big(\mathbf{x}^k(u,v)\big) - 1\right]^2 \mathrm{d}u\,\mathrm{d}v = \min.$$

4. Compute $I_i^{(k)}(u, v) = I_i^{(k+1)}(\mathbf{x}^{(k)}(u, v))$ and then convert $I_i^{(k)}(u, v)$ to the (w_0, h_0)-level bicubic B-spline representation using the least square approximation for the sake of efficient computation of next iteration (see Sect. 5.3).
5. If $k > 0$, set k as $k - 1$, go back to step 2. If $k = 0$, terminate the iteration.

The Outputs of the Algorithm The above algorithm yields a sequence of mappings

$$\mathbf{x}^{(K)}(u,v), \mathbf{x}^{(K-1)}(u,v), \ldots, \mathbf{x}^{(0)}(u,v)$$

and a sequence of in-between images

$$I_i^{(K)}(u,v), I_i^{(K-1)}(u,v), \ldots, I_i^{(0)}(u,v),$$

and it is easy to observe that

$$I_i^{(k)}(u,v) = I_i\big(\mathbf{y}^{(k)}(u,v)\big)$$

with

$$\mathbf{y}^{(k)}(u,v) = \mathbf{x}^{(K)}\Big(\mathbf{x}^{(K-1)}\Big(\cdots\Big(\mathbf{x}^{(k+1)}\big(\mathbf{x}^{(k)}(u,v)\big)\Big)\Big)\Big), \quad k = K, K-1, \ldots, 0.$$

The final mapping between I_i and I_t is $\mathbf{x}(u, v) = \mathbf{y}^{(0)}(u, v)$.

5.2 Gaussian Filter

Images usually are discontinuous, using (w_k, h_k)-level bicubic B-spline function to approximate $I_i^{(k+1)}$ and I_t may lead to the approximations $\tilde{I}_i^{(k)}$ and $\tilde{I}_t^{(k)}$ of $I_i^{(k+1)}$ and I_t having Gibbs phenomenon. To cope with this problem, we smooth the images $I_i^{(k+1)}$ and I_t using the Gaussian filter.

Since the smoothed result will be represented by (w_k, h_k)-level bicubic B-splines, the smoothing effect should be consistent with the spline representation. Therefore, we choose σ_k in two directions respectively such that

$$G_{\sigma_k}(x) * N_0(x) \approx \frac{1}{h_k} N_0\left(\frac{x}{h_k}\right), \tag{22}$$

where N_0 is the cubic B-spline basis on the knots -2, -1, 0, 1, 2, h_k is either of the two knot spacings of (w_k, h_k)-level bicubic B-spline basis. Applying Fourier transform to both sides of (22), we have

$$e^{-\omega^2\sigma_k^2/2}\left(\frac{\sin(\omega/2)}{\omega/2}\right)^4 \approx \left(\frac{\sin(h_k\omega/2)}{h_k\omega/2}\right)^4,$$

or

$$-\frac{\omega^2\sigma_k^2}{2} \approx \log\left[\frac{1}{h_k^4}\left(\frac{\sin(h_k\omega/2)}{\sin(\omega/2)}\right)^4\right].$$

Computing the second order Taylor expansion of the right-hand side with respect to ω, we obtain the following approximation of σ_k^2:

$$\sigma_k^2 = \frac{h_k^2 - 1}{3}.$$

Hence, a closed relationship between h_k and σ_k is obtained. Note that if $h_k = 1$, which is the initial image knot spacing, then we have $\sigma_k = 0$. Namely, the Gaussian filter has no smoothing effect.

5.3 *Least Square Approximations*

In this sub-section, we explain how the least square approximations of B-spline function are efficiently computed. Let $f(u, v)$ be a given function on Ω and m and n two positive integers. Suppose we intend to approximate f by a bivariate cubic B-spline function

$$F(u, v) = \sum_{i=0}^{m+2}\sum_{j=0}^{n+2} f_{ij} N_i^{(1/m)}(u) N_j^{(1/n)}(v) \tag{23}$$

in the least square sense;

$$\mathcal{E}(F) = \int_\Omega \left[f(u, v) - F(u, v)\right]^2 \mathrm{d}u\,\mathrm{d}v = \min. \tag{24}$$

From

$$\frac{\partial \mathcal{E}(F)}{\partial f_{\alpha\beta}} = 0, \quad \alpha, \beta = 0, 1, \ldots, m+2, n+2,$$

we obtain the following equations

$$c_{00}^{(1/m)} C^{(1/n)} F_0 + c_{10}^{(1/m)} C^{(1/n)} F_1 + \cdots + c_{m+2,0}^{(1/m)} C^{(1/n)} F_{m+2} = B_0,$$

$$c_{01}^{(1/m)} C^{(1/n)} F_0 + c_{11}^{(1/m)} C^{(1/n)} F_1 + \cdots + c_{m+2,1}^{(1/m)} C^{(1/n)} F_{m+2} = B_1,$$

$$\cdots$$

$$c^{(1/m)}_{0,m+2}C^{(1/n)}F_0 + c^{(1/m)}_{1,m+2}C^{(1/n)}F_1 + \cdots + c^{(1/m)}_{m+2,m+2}C^{(1/n)}F_{m+2} = B_{m+2},$$

where

$$C^{(1/n)} = \big[c^{(1/n)}_{ij}\big]^{n+2,n+2}_{ij=0},$$
$$F_k = [f_{k0}, \ldots, f_{k,n+2}]^T, \qquad B_k = [b_{k0}, \ldots, b_{k,n+2}]^T,$$

with

$$b_{ij} = \int_{\Omega} f(u,v) N^{(1/m)}_i(u) N^{(1/n)}_j(v)\, \mathrm{d}u\, \mathrm{d}v. \tag{25}$$

These equations can be written as

$$c^{(1/m)}_{00}F_0 + c^{(1/m)}_{10}F_1 + \cdots + c^{(1/m)}_{m+2,0}F_{m+2} = \big[C^{(1/n)}\big]^{-1}B_0,$$
$$c^{(1/m)}_{01}F_0 + c^{(1/m)}_{11}F_1 + \cdots + c^{(1/m)}_{m+2,1}F_{m+2} = \big[C^{(1/n)}\big]^{-1}B_1,$$
$$\cdots$$
$$c^{(1/m)}_{0,m+2}F_0 + c^{(1/m)}_{1,m+2}F_1 + \cdots + c^{(1/m)}_{m+2,m+2}F_{m+2} = \big[C^{(1/n)}\big]^{-1}B_{m+2}.$$

Then the solution is obtained as

$$[F_0, F_1, \ldots, F_{m+2}] = \big[C^{(1/n)}\big]^{-1}[B_0, B_1, \ldots, B_{m+2}]\big[C^{(1/m)}\big]^{-T}, \tag{26}$$

where $[\cdot]^{-T}$ denotes the transpose and inverse of a matrix, and $C^{(1/m)} = [c^{(1/m)}_{ij}]^{m+2}_{ij=0}$. Therefore, the least square approximation problem (24) is efficiently solved by inverting two small sized matrices $C^{(1/m)}$ and $C^{(1/n)}$, then computing the matrix multiplications in (26). The total cost is in the order $O(m^3) + O(n^3) + O(m^2n) + O(mn^2)$. The integrals in (25) can be computed efficiently using Gaussian quadrature formula (see [30]) on each of pixels.

6 Regularity Analysis of Mapping x

In this section, we first introduce the used definitions, terminologies and theorems. Then we provide the details of proof for Theorem 1.

Definition 1 ([21]) Let X and Y be topological spaces, and $f : X \to Y$ a bijection. If both the function f and the inverse function $f^{-1} : Y \to X$ are continuous, then f is called a homemorphism. f is a local homeomorphism, if for every point x in X, there exists an open set U containing x, such that $f(U)$ is open in Y and is a homeomorphism.

Definition 2 ([10]) Let $\tilde{B}$ and B be subsets of $\mathbb{R}^2$. We say that $\pi : \tilde{B} \to B$ is a covering map if

1. π is continuous and $\pi(\tilde{B}) = B$.
2. Each point $p \in B$ has a neighborhood U in B (called a distinguished neighborhood of p) such that

$$\pi^{-1}(U) = \bigcup_{\alpha} V_{\alpha},$$

   where V_{α} are pairwise disjoint open sets such that the restriction of π to V_{α} is a homeomorphism of V_{α} onto U.

Definition 3 ([31]) Assuming that (X, ρ) is a metric space, A is a subspace of X. If every sequence in A has a convergence subsequence and the limit point of the convergence subsequence lies in A, we named that A is a self-sequentially compact set.

Definition 4 ([10]) $A \subset \mathbb{R}^n$ is arcwise connected if, given two points $p, q \in A$, there exists an arc in A joining p to q.

Definition 5 ([10]) $A \subset \mathbb{R}^n$ is connected when it is not possible to write $A = U_1 \cup U_2$, where U_1 and U_2 are nonempty open sets in A and $U_1 \cap U_2 = \emptyset$.

Theorem 4 ([4]) *Let T be of class C^1 in a set D with $J(p) \neq 0$ for each $p \in D$, and let T map D one-to-one onto a set $T(D)$, where $J(p)$ denotes the Jacobian determinant of T at the point p. Then, the inverse T^{-1} of T is of class C^1 on $T(D)$ and the differential of T^{-1} is $(dT)^{-1}$, the inverse of the differential of T.*

Theorem 5 ([10]) *Let $\pi : \widetilde{B} \to B$ be a local homeomorphism, $\widetilde{B}$ compact and B connected. Then π is a covering map.*

Theorem 6 ([10]) *Let $\pi : \widetilde{B} \to B$ be a covering map, $\widetilde{B}$ arcwise connected, and B simply connected. Then π is a homeomorphism.*

Now we prove that the correspondence $\mathbf{x}(u, v)$ is an injection and surjection.

Remark 4 $[0, 1]^2$ is regarded as a topological space, i.e. $[0, 1]^2$ is a clopen set.

Lemma 1 $\mathbf{x} : [0, 1]^2 \to [0, 1]^2$ *satisfying* (i)–(iii) *is a locally one-to-one mapping.*

Proof Let a point $p \in [0, 1]^2$, we determine a neighborhood B of p in which $\mathbf{x}$ is a one-to-one map. Let p' and p'' be two points near p such that the line segment jointing p' and p'' lies in $[0, 1]^2$. According to the mean value theorem, we may choose two points p_1^*, p_2^* on this line segment such that

$$\mathbf{x}\big(p''\big) - \mathbf{x}\big(p'\big) = L\big(p'' - p'\big), \tag{27}$$

where L is the linear transformation represented by $L = (\mathbf{x}_u(p_1^*), \mathbf{x}_v(p_2^*))$. Let

$$F(p_1, p_2) = \det\bigl(\mathbf{x}_u(p_1), \mathbf{x}_v(p_2)\bigr),$$

then $F(p_1^*, p_2^*) = \det(L)$. Moreover, since $\mathbf{x}$ is C^2, then F is C^1 continuous, and $F(p, p) \geq \gamma$, there exists a circular neighborhood B of p lying in $[0, 1]^2$, such that $F(p_1, p_2) \geq \gamma$ for all choices of the points p_1, p_2 in B. We shall prove that $\mathbf{x}$ is a one-to-one mapping in B. Assuming that p' and p'' lies in B and $\mathbf{x}(p') = \mathbf{x}(p'')$, we will prove $p' = p''$. Since p' and p'' lies in B and B is convex, the entire line segment joining p' and p'' also lies in B, hence both p_1^* and p_2^* are points of B. Using the property of B, we have $F(p_1^*, p_2^*) = \det(L) \neq 0$. The linear transformation L is therefore nonsingular. According to (27) and using the assumption that $\mathbf{x}(p') = \mathbf{x}(p'')$, we have $L(p'' - p') = 0$. Since L is nonsingular, we deduce that $p' = p''$, i.e. $\mathbf{x}$ is a one-to-one mapping in B. □

Lemma 2 $\mathbf{x} : [0, 1]^2 \to [0, 1]^2$ *satisfying* (i)–(iii) *is a locally homeomorphism.*

Proof According to Lemma 1, $\mathbf{x}$ is a locally one-to-one mapping in $[0, 1]^2$. Given a point $p \in [0, 1]^2$, assuming that $\mathbf{x}$ is one-to-one in a neighborhood B of p, then it is obvious that $\mathbf{x} : B \to \mathbf{x}(B)$ is surjective, where $\mathbf{x}(B)$ denotes the range of $\mathbf{x}$ in B. From Theorem 4, we know that $\mathbf{x}^{-1}$ is continuous in $\mathbf{x}(B)$. Thus, $\mathbf{x}$ is homeomorphic in B, i.e. $\mathbf{x}$ is a locally homeomorphism. □

Proof of Theorem 1 It is obvious that $[0, 1]^2$ is connected and compact. From Lemma 2 and Theorem 5, we deduce that $\mathbf{x}$ is a covering map. Since $[0, 1]^2$ is arcwise connected and simply connected, according to Theorem 6, $\mathbf{x}$ is a homeomorphism. Thus, we obtain that $\mathbf{x}$ is an injection and surjection. The theorem is proved. □

7 Existence and Uniqueness of x

This section devotes to the proof of Theorem 2. Let

$$X = \left\{ \mathbf{x}(u, v) : \mathbf{x}(u, v) = \left[x_1(u, v), x_2(u, v)\right]^{\mathrm{T}} = \sum_{i=0}^{m+2} \sum_{j=0}^{n+2} \mathbf{p}_{ij} N_i^{(1/m)}(u) N_j^{(1/n)}(v) \right.$$
$$\left. \text{satisfying (i)–(iii)} \right\}.$$

Then we define the norm $\mathbf{x} \in X$ as:

$$\|\mathbf{x}\|_X = \left(\int_\Omega |\mathbf{x}|^2 \,\mathrm{d}u \,\mathrm{d}v \right)^{1/2} = \left(\int_\Omega \left(x_1^2 + x_2^2\right) \mathrm{d}u \,\mathrm{d}v \right)^{1/2}.$$

Lemma 3 *Let $\rho(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_X$, then X is a metric space.*

Proof It is obvious that X is a nonempty set, and $\rho(\mathbf{x}, \mathbf{y})$ satisfies,

(i) $\rho(\mathbf{x}, \mathbf{y}) \geq 0$ and $\rho(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$;
(ii) $\rho(\mathbf{x}, \mathbf{y}) = \rho(\mathbf{y}, \mathbf{x})$;
(iii)

$$\begin{aligned}
\rho(\mathbf{x}, \mathbf{z}) &= \left(\int_\Omega |\mathbf{x} - \mathbf{z}|^2 \, \mathrm{d}u \, \mathrm{d}v\right)^{1/2} \\
&= \left(\int_\Omega |\mathbf{x} - \mathbf{y} + \mathbf{y} - \mathbf{z}|^2 \, \mathrm{d}u \, \mathrm{d}v\right)^{1/2} \\
&\leq \left(\int_\Omega |\mathbf{x} - \mathbf{y}|^2 \, \mathrm{d}u \, \mathrm{d}v\right)^{1/2} + \left(\int_\Omega |\mathbf{y} - \mathbf{z}|^2 \, \mathrm{d}u \, \mathrm{d}v\right)^{1/2} \\
&= \rho(\mathbf{x}, \mathbf{y}) + \rho(\mathbf{y}, \mathbf{z}).
\end{aligned}$$

Hence X is a metric space. Under the distance ρ, we denote this metric space as (X, ρ). □

Lemma 4 (X, ρ) *is a closed set.*

Proof Suppose that $\{\mathbf{x}_k\}$ is a fundamental sequence in the space (X, ρ). This sequence can be written as $\mathbf{x}_k(u, v) = \sum_{i=0}^{m+2} \sum_{j=0}^{n+2} (\mathbf{p}_{ij})_k N_i^{(1/m)}(u) N_j^{(1/n)}(v)$. It is easy to deduce that $(\mathbf{p}_{ij})_k$ are bounded, hence there exists a subsequence $(\mathbf{p}_{ij})_{k_l}$ converging to $(\mathbf{p}_{ij})_0$. Because $\{\mathbf{x}_k\}$ is a fundamental sequence, we obtain that,

$$\begin{aligned}
\lim_{k\to\infty} \mathbf{x}_k(u, v) &= \lim_{l\to\infty} \mathbf{x}_{k_l}(u, v) \\
&= \lim_{l\to\infty} \sum_{i=0}^{m+2} \sum_{j=0}^{n+2} (\mathbf{p}_{ij})_{k_l} N_i^{(1/m)}(u) N_j^{(1/n)}(v) \\
&= \sum_{i=0}^{m+2} \sum_{j=0}^{n+2} (\mathbf{p}_{ij})_0 N_i^{(1/m)}(u) N_j^{(1/n)}(v).
\end{aligned}$$

Let $\mathbf{x}_0(u, v) = \sum_{i=0}^{m+2} \sum_{j=0}^{n+2} (\mathbf{p}_{ij})_0 N_i^{(1/m)}(u) N_j^{(1/n)}(v)$. Because the range of $\mathbf{x}_k$ is $[0, 1]^2$ which is a closed set, we obtain that $\mathbf{x}_0(u, v) \in [0, 1]^2$. On the other hand, it is obvious that

$$\begin{aligned}
\mathbf{x}_0(0, v) &= [0, v]^T, &\qquad \mathbf{x}_0(1, v) &= [1, v]^T, \\
\mathbf{x}_0(u, 0) &= [u, 0]^T, &\qquad \mathbf{x}_0(u, 1) &= [u, 1]^T.
\end{aligned}$$

Moreover, since $\det((\mathbf{x}_k)_u, (\mathbf{x}_k)_v) \geq \gamma$,

$$\det\big((\mathbf{x}_0)_u, (\mathbf{x}_0)_v\big) = \lim_{k\to\infty} \det\big((\mathbf{x}_k)_u, (\mathbf{x}_k)_v\big) \geq \gamma,$$

i.e. $\mathbf{x}_0(u, v) \in X$. Therefore, X is a closed set. □

Lemma 5 *X is a self-sequentially compact set.*

Proof Because the number of the bicubic B-spline bases is finite, X is a finite dimensional space. On the other hand, $\mathbf{x}(u, v) : [0, 1]^2 \to [0, 1]^2$, hence $\|\mathbf{x}(u, v)\| \leq 2$. According to Lemma 4, X is a closed set. Then we conclude that X is a self-sequentially compact set. □

Proof of Theorem 2 Let $\{\mathbf{x}_n\}$ be a minimizing sequence for the model (2), i.e.

$$\lim_{n\to\infty} \mathscr{E}(\mathbf{x}_n) = \inf_{\mathbf{x}\in X} \mathscr{E}(\mathbf{x}).$$

According to Lemma 5, X is a self-sequentially compact set. Hence, there exists a subsequence $\mathbf{x}_{n_k}$ and $\mathbf{x}_0$ in X such that $\mathbf{x}_{n_k} \to \mathbf{x}_0$.

Finally, since $I_i(\mathbf{x})$ is a continuous function and X is a bounded closed set, $I_i(\mathbf{x})$ is a uniformly continuous function with respect to $\mathbf{x}$. For every $\varepsilon > 0$, there exists a $\delta > 0$ such that $|I_i(\mathbf{x}) - I_i(\mathbf{y})| < \varepsilon$ when $|\mathbf{x} - \mathbf{y}| < \delta$. Therefore,

$$\begin{aligned}&\left|\int_\Omega \left|I_i(\mathbf{x}) - I_t\right|^2 \mathrm{d}u\,\mathrm{d}v - \int_\Omega \left|I_i(\mathbf{y}) - I_t\right|^2 \mathrm{d}u\,\mathrm{d}v\right| \\ &\quad \leq \int_\Omega \left|I_i(\mathbf{x}) - I_i(\mathbf{y})\right|\left|I_i(\mathbf{x}) + I_i(\mathbf{y}) - 2I_t\right| \mathrm{d}u\,\mathrm{d}v \leq M\varepsilon\end{aligned}$$

where M is a constant. Similarly, it is easy to see that $\int_\Omega (g(\mathbf{x}(u, v)) - 1)^2\, \mathrm{d}u\,\mathrm{d}v$ is a continuous functional with respect to $\mathbf{x}$. Hence the energy functional

$$\mathscr{E}(\mathbf{x}) = \int_\Omega \left|I_i(\mathbf{x}) - I_t\right|^2 \mathrm{d}u\,\mathrm{d}v + \lambda \int_\Omega \big(g\big(\mathbf{x}(u, v)\big) - 1\big)^2 \mathrm{d}u\,\mathrm{d}v$$

is continuous with respect to $\mathbf{x}$. Thus,

$$\mathscr{E}(\mathbf{x_0}) = \lim_{k\to\infty} \mathscr{E}(\mathbf{x}_{n_k}) = \inf_{\mathbf{x}\in X} \mathscr{E}(\mathbf{x}),$$

i.e., $\mathbf{x}_0$ is a minimum point of $\varepsilon(\mathbf{x})$. □

8 Existence and Uniqueness of ODE's Solution

In this section, we show that Eq. (10) has a unique solution. First, we introduce Gronwall's inequality.

Gronwall's Inequality (See [14]) If (i) $g(t)$ is continuous on $t_0 \le t \le t_1$, (ii) $g(t)$ satisfies the inequality

$$0 \le g(t) \le K + L\int_{t_0}^{t} g(s)\,\mathrm{d}s \quad \text{on } t_0 \le t \le t_1,$$

then

$$0 \le g(t) \le Ke^{L(t-t_0)} \quad \text{on } t_0 \le t \le t_1.$$

Hence, if $K = 0$, then $g(t) = 0$.

Now we prove the existence and uniqueness for the solution of Eq. (10). Equation (10) can be written as

$$\begin{cases} \frac{d\tilde{\mathbf{x}}(t)}{dt} = M^{-1}\mathbf{Q}(\tilde{\mathbf{x}}(t)), \\ \tilde{\mathbf{x}}(0) = \mathbf{C}_0, \end{cases} \tag{28}$$

where

$$\begin{aligned} \tilde{\mathbf{x}}(t) &= \left[\mathbf{x}_0^{\mathrm{T}}(t), \mathbf{x}_1^{\mathrm{T}}(t), \ldots, \mathbf{x}_{n_0-1}^{\mathrm{T}}(t), \mathbf{x}_{n_0}^{\mathrm{T}}(t)\right]^{\mathrm{T}}, \\ M^{-1} &= [m_{ij}]^{-1} \otimes I_2, \quad i, j = 1, 2, \ldots, n_0, \\ \mathbf{Q}\big(\tilde{\mathbf{x}}(t)\big) &= -\left[\mathbf{q}_0^{\mathrm{T}}, \mathbf{q}_1^{\mathrm{T}}, \ldots, \mathbf{q}_{n_0-1}^{\mathrm{T}}, \mathbf{q}_{n_0}^{\mathrm{T}}\right]^{\mathrm{T}}, \end{aligned}$$

and

$$\mathbf{C}_0 = \left[\mathbf{c}_0^{\mathrm{T}}, \mathbf{c}_1^{\mathrm{T}}, \ldots, \mathbf{c}_{n_0-1}^{\mathrm{T}}, \mathbf{c}_{n_0}^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{29}$$

with $\mathbf{C}_0$ satisfying

$$\sum_{j=0}^{n_0} \mathbf{c}_j\phi_j(u,v) + \sum_{j=n_0+1}^{n_1} \mathbf{x}_j\phi_j(u,v) = [u,v]^T. \tag{30}$$

Remark 5 Since $\mathbf{x}_{n_0+1}, \mathbf{x}_{n_0+2}, \ldots, \mathbf{x}_{n_1}$ are fixed in the iterative process, $[\mathbf{x}_{n_0+1}^{\mathrm{T}}, \mathbf{x}_{n_0+2}^{\mathrm{T}}, \ldots, \mathbf{x}_{n_1-1}^{\mathrm{T}}, \mathbf{x}_{n_1}^{\mathrm{T}}]^{\mathrm{T}}$ is a constant vector.

For simplicity of description, we denote $M^{-1}\mathbf{Q}(\tilde{\mathbf{x}}(t))$ by $\mathbf{f}(\tilde{\mathbf{x}}(t))$, where

$$\mathbf{f}\big(\tilde{\mathbf{x}}(t)\big) = \left[\mathbf{f}_0^{\mathrm{T}}\big(\tilde{\mathbf{x}}(t)\big), \mathbf{f}_1^{\mathrm{T}}\big(\tilde{\mathbf{x}}(t)\big), \ldots, \mathbf{f}_{n_0-1}^{\mathrm{T}}\big(\tilde{\mathbf{x}}(t)\big), \mathbf{f}_{n_0}^{\mathrm{T}}\big(\tilde{\mathbf{x}}(t)\big)\right]^{\mathrm{T}}.$$

Now we prove the existence and uniqueness of the solution for Eq. (28). First, a positive number δ for the upper bound of t needs to be determined. Integrating both sides of (28), we have

$$\tilde{\mathbf{x}}(t) = \mathbf{C}_0 + \int_0^t \mathbf{f}\big(\tilde{\mathbf{x}}(t)\big)\,\mathrm{d}t.$$

Taking inner product of both sides with $[\phi_0, \dots, \phi_{n_0}]^{\mathrm{T}}$ and using (30), we obtain

$$\mathbf{x}(u, v) = [u, v]^T + \sum_{j=0}^{n_0} \int_0^t \mathbf{f}_j\big(\tilde{\mathbf{x}}(s)\big)\phi_j(u, v)\,\mathrm{d}s.$$

Let $\mathbf{F}(\tilde{\mathbf{x}}(t), u, v) = \sum_{j=0}^{n_0} \mathbf{f}_j(\tilde{\mathbf{x}}(t))\phi_j(u, v)$. Then it is obvious that $\mathbf{F}(\tilde{\mathbf{x}}(t), u, v)$ is continuous with respect to t. Using mean value theorem of integrals, we have

$$\int_0^t \sum_{j=0}^{n_0} f_j^{(l)}\big(\tilde{\mathbf{x}}(s)\big)\phi_j(u, v)\,\mathrm{d}s = tF^{(l)}\big(\tilde{\mathbf{x}}(\xi_l), u, v\big), \quad \text{where } 0 < \xi_l < t, l = 1, 2,$$

where $[f_j^{(1)}(\tilde{\mathbf{x}}(s)), f_j^{(2)}(\tilde{\mathbf{x}}(s))]^T = \mathbf{f}_j(\tilde{\mathbf{x}}(s))$ and $[F^{(1)}(\tilde{\mathbf{x}}(s), u, v), F^{(2)}(\tilde{\mathbf{x}}(s), u, v)]^T = \mathbf{F}(\tilde{\mathbf{x}}(s), u, v)$. Let

$$\det(\mathbf{x}_u, \mathbf{x}_v) = \begin{vmatrix} 1 + tF_u^{(1)}(\tilde{\mathbf{x}}(\xi_1)) & tF_v^{(1)}(\tilde{\mathbf{x}}(\xi_1)) \\ tF_u^{(2)}(\tilde{\mathbf{x}}(\xi_2)) & 1 + tF_v^{(2)}(\tilde{\mathbf{x}}(\xi_2)) \end{vmatrix} = \gamma.$$

Computing the determinant above, we get

$$\begin{aligned}&\big(F_u^{(1)}\big(\tilde{\mathbf{x}}(\xi_1)\big)F_v^{(2)}\big(\tilde{\mathbf{x}}(\xi_2)\big) - F_u^{(2)}\big(\tilde{\mathbf{x}}(\xi_2)\big)F_v^{(1)}\big(\tilde{\mathbf{x}}(\xi_1)\big)\big)t^2 \\ &\quad + \big(F_u^{(1)}\big(\tilde{\mathbf{x}}(\xi_1)\big) + F_v^{(2)}\big(\tilde{\mathbf{x}}(\xi_2)\big)\big)t + 1 - \gamma = 0. \end{aligned} \tag{31}$$

For the sake of easy description, let

$$\begin{aligned} a(\mathbf{x}_{\xi_1}, \mathbf{x}_{\xi_2}) &= F_u^{(1)}\big(\tilde{\mathbf{x}}(\xi_1)\big)F_v^{(2)}\big(\tilde{\mathbf{x}}(\xi_2)\big) - F_u^{(2)}\big(\tilde{\mathbf{x}}(\xi_2)\big)F_v^{(1)}\big(\tilde{\mathbf{x}}(\xi_1)\big), \\ b(\mathbf{x}_{\xi_1}, \mathbf{x}_{\xi_2}) &= F_u^{(1)}\big(\tilde{\mathbf{x}}(\xi_1)\big) + F_v^{(2)}\big(\tilde{\mathbf{x}}(\xi_2)\big), \end{aligned}$$

where $\mathbf{x}_{\xi_1}$ and $\mathbf{x}_{\xi_2}$ are the mappings in X defined by the coefficients $\tilde{\mathbf{x}}(\xi_1)$ and $\tilde{\mathbf{x}}(\xi_2)$, respectively. Then we can get a minimal positive root of Eq. (31) as the following,

$$t(\mathbf{x}_{\xi_1}, \mathbf{x}_{\xi_2}) = \frac{2(1-\gamma)}{-b(\mathbf{x}_{\xi_1}, \mathbf{x}_{\xi_2}) + \sqrt{b^2(\mathbf{x}_{\xi_1}, \mathbf{x}_{\xi_2}) - 4a(\mathbf{x}_{\xi_1}, \mathbf{x}_{\xi_2})(1-\gamma)}}. \tag{32}$$

Let

$$\begin{aligned} &\Omega(\mathbf{x}, \mathbf{y}) \\ &\quad = \big\{[u, v]^T \in [0, 1]^2 : b^2\big(\mathbf{x}(u, v), \mathbf{y}(u, v)\big) - 4a\big(\mathbf{x}(u, v), \mathbf{y}(u, v)\big)(1-\gamma) \geq 0, \\ &\qquad \sqrt{b^2\big(\mathbf{x}(u, v), \mathbf{y}(u, v)\big) - 4a\big(\mathbf{x}(u, v), \mathbf{y}(u, v)\big)(1-\gamma)} \geq b\big(\mathbf{x}(u, v), \mathbf{y}(u, v)\big)\big\}. \end{aligned}$$

Then $\Omega(\mathbf{x}, \mathbf{y})$ is a closed set.

Lemma 6 *Let*

$$\delta = \min_{\mathbf{x},\mathbf{y}\in X} \min_{[u,v]^T \in \Omega(\mathbf{x},\mathbf{y})} t(\mathbf{x},\mathbf{y}).$$

then $\delta > 0$. *Furthermore, if* $0 \le t \le \delta$, *then* $\det(\mathbf{x}_u, \mathbf{x}_v) \ge \gamma$.

Proof Let

$$y(t) = \det(\mathbf{x}_u, \mathbf{x}_v) - \gamma = at^2 + bt + 1 - \gamma,$$

$$\mathbf{F}\big(\tilde{\mathbf{x}}(t), u, v\big) = \sum_{j=0}^{n_0} \mathbf{f}_j\big(\tilde{\mathbf{x}}(t)\big)\phi_j(u,v).$$

We can easily prove that $\mathbf{F}(\tilde{\mathbf{x}}(t), u, v)$ is a C^2 continuous vector-valued function in the space X and $[0,1]^2$. Since X and $[0,1]^2$ are closed sets, $F_u^{(l)}(\tilde{\mathbf{x}}(\xi_l), u, v)$ and $F_v^{(l)}(\tilde{\mathbf{x}}(\xi_l), u, v)$ $(l = 1, 2)$ are bounded functions. Hence, $\delta > 0$ is proved. On the other hand, since $y(0) = 1 - \gamma > 0$, $y(t) > 0$ is true when $0 \le t \le \delta$. □

Lemma 7 *Let*

$$\Re = \Bigg\{ \tilde{\mathbf{x}}(t) : \det(\mathbf{x}_u, \mathbf{x}_v) \ge \gamma, \quad \textit{where } \mathbf{x} = \sum_{j=0}^{n_0} \mathbf{x}_j \phi_j(u,v) + \sum_{j=n_0+1}^{n_1} \mathbf{x}_j \phi_j(u,v) \Bigg\}. \tag{33}$$

Then $M^{-1}\mathbf{Q}(\tilde{\mathbf{x}}(t))$ *is Lipschitz continuous with respect to* $\tilde{\mathbf{x}}$ *on* $\Re$ *when* $0 \le t \le \delta$.

Proof From

$$D_1\mathbf{x} = \frac{\partial \mathbf{x}}{\partial u} = \sum_{j=0}^{n_1} \mathbf{x}_j \frac{\partial \phi_j(u,v)}{\partial u}, \qquad D_2\mathbf{x} = \frac{\partial \mathbf{x}}{\partial v} = \sum_{j=0}^{n_1} \mathbf{x}_j \frac{\partial \phi_j(u,v)}{\partial v},$$

it is easy to see that $D_l\mathbf{x}$ are polynomials of $\mathbf{x}_j(t)$ $(j = 0, 1, \dots, n_0)$, $l = 1, 2$. From

$$(D_1\mathbf{x})_u = \frac{\partial^2 \mathbf{x}}{\partial u^2} = \sum_{j=0}^{n_1} \mathbf{x}_j \frac{\partial^2 \phi_j(u,v)}{\partial u^2}, \qquad (D_1\mathbf{x})_v = \frac{\partial^2 \mathbf{x}}{\partial u\,\partial v} = \sum_{j=0}^{n_1} \mathbf{x}_j \frac{\partial^2 \phi_j(u,v)}{\partial u\,\partial v},$$

we know that $(D_1\mathbf{x})_u$ and $(D_1\mathbf{x})_v$ are also polynomials of $\mathbf{x}_j(t)$ $(j = 0, 1, \dots, n_0)$. Similarly, $(D_2\mathbf{x})_u$ and $(D_2\mathbf{x})_v$ are polynomials of $\mathbf{x}_j(t)$ $(j = 0, 1, \dots, n_0)$. From

$$\Phi_u = (D_l\mathbf{x})_u (D_l\mathbf{x})^{\mathrm{T}} \phi_i + (D_l\mathbf{x})(D_l\mathbf{x})_u^{\mathrm{T}} \phi_i + (D_l\mathbf{x})(D_l\mathbf{x})^{\mathrm{T}} (\phi_i)_u,$$

it is easy to see that Φ_u is a polynomial of $\mathbf{x}_j(t)$. Similarly Φ_v and $g(\mathbf{x})$ are polynomials of $\mathbf{x}_j(t)$. Since $\Re$ is a closed set on the interval $[0, \delta]$, we conclude that

$\int_\Omega (\Phi_u^{\mathrm{T}}\alpha + \Phi_v^{\mathrm{T}}\beta)\,\mathrm{d}u\,\mathrm{d}v$ is Lipschitz continuous on $\Re$. Furthermore, since $I_i(\mathbf{x})$ and I_t are C^2, $\int_0^1\int_0^1((I_i(\mathbf{x}) - I_t)(D_l\mathbf{x})^{\mathrm{T}}(\nabla_{\mathbf{x}} I_i)(D_l\mathbf{x})\phi_i)\,\mathrm{d}u\,\mathrm{d}v$ is Lipschitz continuous on $\Re$. Thus, $M^{-1}\mathbf{Q}(\tilde{\mathbf{x}}(t))$ is Lipschitz continuous on $\Re$. □

Now we construct successive approximations which are defined as follows:

$$\begin{cases} \tilde{\mathbf{x}}_{k+1}(t) = \mathbf{C}_0 + \int_0^t \mathbf{f}(\tilde{\mathbf{x}}_k(s))\,\mathrm{d}s, \quad k = 0, 1, 2, \ldots \\ \tilde{\mathbf{x}}_0(t) = \mathbf{C}_0. \end{cases} \tag{34}$$

Theorem 7 *Each vector-valued function $\tilde{\mathbf{x}}_k$ defined by (34) lies in $\Re$ for $0 \le t \le \delta$, $k = 0, 1, 2, \ldots$.*

Proof From $\tilde{\mathbf{x}}_0(t) = \mathbf{C}_0$ and the equality (30), we know that $\tilde{\mathbf{x}}_0$ lies in $\Re$. Moreover, according to

$$\tilde{\mathbf{x}}_{k+1}(t) = \mathbf{C}_0 + \int_0^t \mathbf{f}\big(\tilde{\mathbf{x}}_k(s)\big)\,\mathrm{d}s, \tag{35}$$

we obtain that

$$\begin{aligned} \mathbf{x}_{k+1} &= \sum_{j=0}^{n_0}(\mathbf{x}_{k+1})_j(t)\phi_j(u,v) + \sum_{j=n_0+1}^{n_1}\mathbf{x}_j\phi_j(u,v) \\ &= [u,v]^T + \sum_{j=0}^{n_0}\int_0^t \mathbf{f}_j\big(\tilde{\mathbf{x}}_k(s)\big)\phi_j(u,v)\,\mathrm{d}s. \end{aligned}$$

Let $\mathbf{F}_k(t,u,v) = \sum_{j=0}^{n_0}\mathbf{f}_j(\tilde{\mathbf{x}}_k(t))\phi_j(u,v)$. It is obvious that $\mathbf{F}_k(t,u,v)$ is continuous with respect to t. From the mean value theorem of integrals, we have

$$\int_0^t \sum_{j=0}^{n_0} f_j^{(l)}\big(\tilde{\mathbf{x}}_k(s)\big)\phi_j(u,v)\,\mathrm{d}s = tF_k^{(l)}(\xi_l,u,v), \quad l = 1, 2,$$

where $0 \le \xi_l \le t \le \delta$. From Lemma 6, we have

$$\begin{aligned} &\det\big((\mathbf{x}_{k+1})_u, (\mathbf{x}_{k+1})_v\big) \\ &\quad = \begin{vmatrix} 1 + t(F_k)_u^{(1)}(\xi_1) & t(F_k)_v^{(1)}(\xi_1) \\ t(F_k)_u^{(2)}(\xi_2) & 1 + t(F_k)_v^{(2)}(\xi_2) \end{vmatrix} \ge \gamma \quad \text{when } 0 \le t \le \delta. \end{aligned}$$

Thus, $\mathbf{x}_{k+1} \in \Re$ for $0 \le t \le \delta$. □

The proofs of Lemmas 8, 9 in the following are similar to that of Theorem I-1-4 in [14]. For completeness, we give the details.

Lemma 8 *The successive approximations given by* (34) *satisfy the estimates*

$$\left|\tilde{\mathbf{x}}_{k+1}(t)-\tilde{\mathbf{x}}_k(t)\right| \le \frac{ML^k}{(k+1)!}t^{k+1} \quad \textit{for } 0\le t\le \delta, k=0,1,2,\ldots. \tag{36}$$

Proof We use mathematical induction. When $k=0$, we have $|\tilde{\mathbf{x}}_1(t)-\tilde{\mathbf{x}}_0(t)| = |\int_0^t \mathbf{f}(\mathbf{C}_0)\,\mathrm{d}s| \le Mt$. Assume (36) is true for k, then

$$\begin{aligned}\left|\tilde{\mathbf{x}}_{k+1}(t)-\tilde{\mathbf{x}}_k(t)\right| &= \left|\int_0^t \big(\mathbf{f}\big(\tilde{\mathbf{x}}_k(s)\big)-\mathbf{f}\big(\tilde{\mathbf{x}}_{k-1}(s)\big)\big)\,\mathrm{d}s\right| \\ &\le L\left|\int_0^t \left|\tilde{\mathbf{x}}_k(s)-\tilde{\mathbf{x}}_{k-1}(s)\right|\mathrm{d}s\right| \\ &\le \frac{ML^k}{k!}\left|\int_0^t s^k\,\mathrm{d}s\right| = \frac{ML^k}{(k+1)!}t^{k+1}. \end{aligned}\tag{37}$$

□

Lemma 9 *The sequence* $\tilde{\mathbf{x}}_k(t)$, $k=0,1,2,\ldots$, *converges to*

$$\tilde{\mathbf{x}}(t)=\mathbf{C}_0+\sum_{k=1}^{\infty}\big(\tilde{\mathbf{x}}_k(t)-\tilde{\mathbf{x}}_{k-1}(t)\big) \tag{38}$$

uniformly on $0\le t\le\delta$ *as* $k\to\infty$.

Proof According to Lemma 8, for a given $\varepsilon>0$, there exists a positive integer N such that

$$\sum_{k=N}^{\infty}\left|\tilde{\mathbf{x}}_k(t)-\tilde{\mathbf{x}}_{k-1}(t)\right| \le \frac{M}{L}\sum_{k=N}^{\infty}\frac{(Lt)^k}{k!} \le \frac{M}{L}\sum_{k=N}^{\infty}\frac{(L\delta)^k}{k!} < \varepsilon. \tag{39}$$

Thus the series $\sum_{k=1}^{\infty}(\tilde{\mathbf{x}}_k(t)-\tilde{\mathbf{x}}_{k-1}(t))$ is uniformly convergent on $0\le t\le\delta$. On the other hand, since $\tilde{\mathbf{x}}_N(t)=\mathbf{C}_0+\sum_{k=1}^{N}(\tilde{\mathbf{x}}_k(t)-\tilde{\mathbf{x}}_{k-1}(t))$, the result is proved. □

Proof of Theorem 3 Because

$$\tilde{\mathbf{x}}_{k+1}(t)=\mathbf{C}_0+\int_0^t \mathbf{f}\big(\tilde{\mathbf{x}}_k(s)\big)\,\mathrm{d}s, \tag{40}$$

from the continuity of $\tilde{\mathbf{x}}$ and Lemma 9, we conclude that

$$\tilde{\mathbf{x}}(t)=\mathbf{C}_0+\int_0^t \mathbf{f}\big(\tilde{\mathbf{x}}(s)\big)\,\mathrm{d}s. \tag{41}$$

Then

$$\frac{d\tilde{\mathbf{x}}(t)}{dt}=\mathbf{f}\big(\tilde{\mathbf{x}}(t)\big), \tag{42}$$

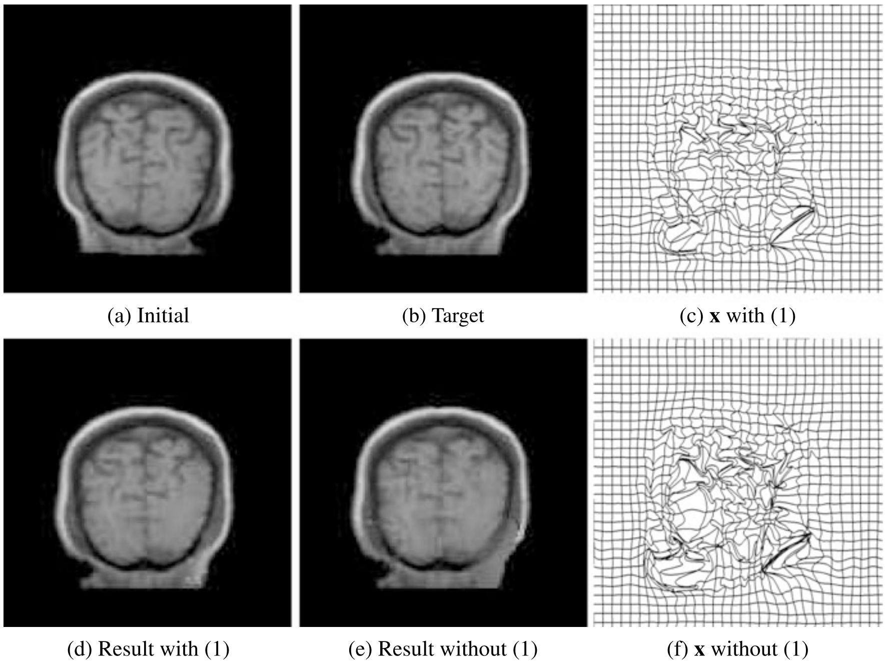


Fig. 1 Experiment to show the effect of the constraint (1)

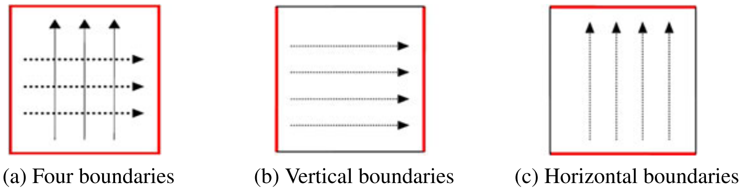


Fig. 2 Three different boundary conditions. The *thick lines* are the fixed boundaries

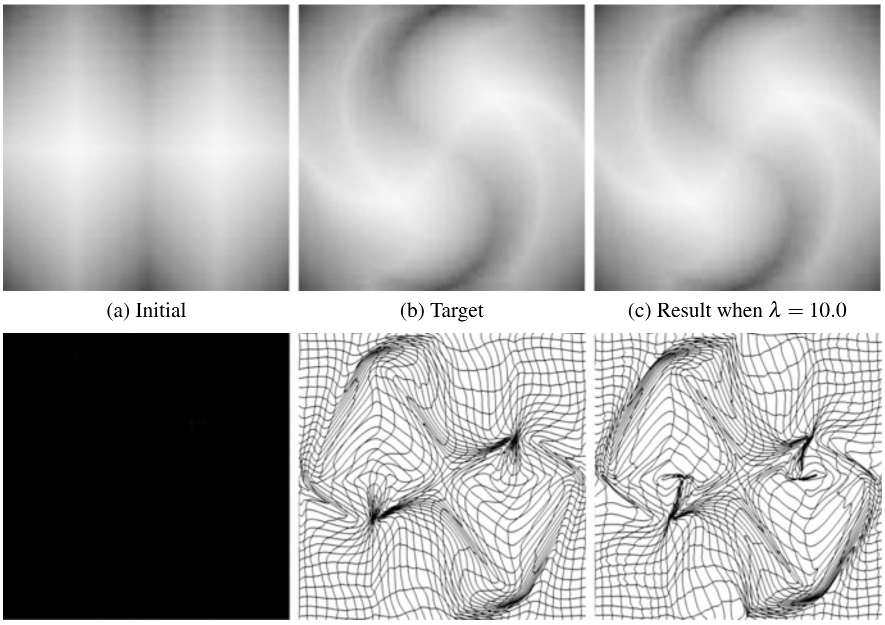
hence $\tilde{\mathbf{x}}(t)$ is a solution of (28).

To prove the uniqueness, suppose that $\tilde{\mathbf{y}}(t)$ is another solution of problem (28) on the interval $[0, \delta]$. Note that

$$\tilde{\mathbf{y}}(t) = \mathbf{C}_0 + \int_0^t \mathbf{f}\big(\tilde{\mathbf{y}}(s)\big)\,\mathrm{d}s \tag{43}$$

on $0 \le t \le \delta$. Hence using the Lipschitz continuity of $\mathbf{f}$, we have

$$\big|\tilde{\mathbf{x}}(t) - \tilde{\mathbf{y}}(t)\big| = \left|\int_0^t \big(\mathbf{f}\big(\tilde{\mathbf{x}}(s)\big) - \mathbf{f}\big(\tilde{\mathbf{y}}(s)\big)\big)\,\mathrm{d}s\right| \le L\int_0^t \big|\tilde{\mathbf{x}}(s) - \tilde{\mathbf{y}}(s)\big|\,\mathrm{d}s \tag{44}$$


(a) Initial (b) Target (c) Result when $\lambda = 10.0$

(d) Final error when $\lambda = 10.0$ (e) $\mathbf{x}$ when $\lambda = 10.0$ (f) $\mathbf{x}$ when $\lambda = 0.0$

Fig. 3 Experiment to show the effect of different λ

on $0 \le t \le \delta$. Applying Gronwall's inequality, we conclude that $|\tilde{\mathbf{x}}(t) - \tilde{\mathbf{y}}(t)| = 0$ on $0 \le t \le \delta$. Hence we complete the proof of the theorem. □

9 Experiments

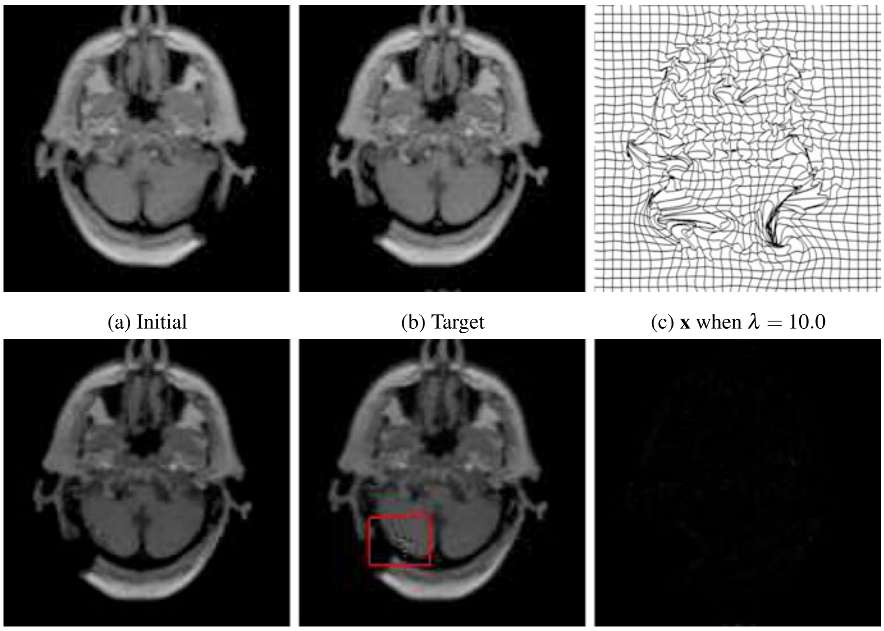
In this section, we verify the performance of our algorithm from two aspects, namely, the illustrative figures and the numerical data. In each of the experiments, we first construct the mapping $\mathbf{x}$ so that $I_i(\mathbf{x}) = I_t$ approximately, meanwhile obtain error images between the target images and our result images, that is

$$E(u, v) = \left| I_i\big(\mathbf{x}(u, v)\big) - I_t(u, v) \right|. \tag{45}$$

Finally compute the numerical data from various similarity metrics.

We first present some algorithm details including the usage of the regularity constraint of the mapping $\mathbf{x}$, the choice of N-sequences and the setting of the boundary conditions. Then compare our method with other two methods through several experiments.

As we know that, when evaluating the alignment effect of two images, similarity metric reaches the extremum when the alignment result is completely identical to the target image. Hence, we will apply the following similarity metrics (SM): Mean


(a) Initial (b) Target (c) $\mathbf{x}$ when $\lambda = 10.0$

(d) Result when $\lambda = 10.0$ (e) Result when $\lambda = 50.0$ (f) Final error when $\lambda = 10.0$

Fig. 4 Experiment to show that bigger λ will lead to a less accurate alignment result (see the part in *box* of figure (**e**)

Squared Difference (MSD); Normalized cross correlation (NCC) in [16]; Entropy of the Difference Image (EDI) in [5]; Mutual Information (MI) in [29]; Number of sites of disagreement (NSD), Largest difference (LD) and Total unsigned difference (TUD) in [12].

Regularity of Mapping x A necessary and sufficient condition for $\mathbf{x}(u, v)$ to be a regular parametric surface is that the normal vector is non-zero everywhere on the surface, i.e. $\mathbf{x}_u \times \mathbf{x}_v \neq 0$, where $\mathbf{x}_u$ and $\mathbf{x}_v$ are the two tangential vectors. We use the inequality (1) as the regularity constraint. In our examples, we take $\varepsilon = 0.001$. Figure 1 shows the effect of using constraint (1), where the target image is the mirror projection of the initial image about its vertical middle-axis. It is easy to see that the employment of the constraint leads to smoother $\mathbf{x}$. Singularities occur without using the constraint.

Boundary Conditions According to the specific feature of the given images, we can set different boundary conditions. Figure 2(a) shows a case where four boundaries are fixed. In this case, the computation can be accelerated by replacing Φ in (5) with $[(D_1\mathbf{x})(D_1\mathbf{x})^T + (D_2\mathbf{x})(D_2\mathbf{x})^T]\phi$. The other two cases are moving $\mathbf{x}$ in $D_1\mathbf{x}$ direction (see Fig. 2(b)) with fixed the vertical boundaries and in $D_2\mathbf{x}$ direction (see Fig. 2(c)) with fixed horizontal boundaries, respectively.

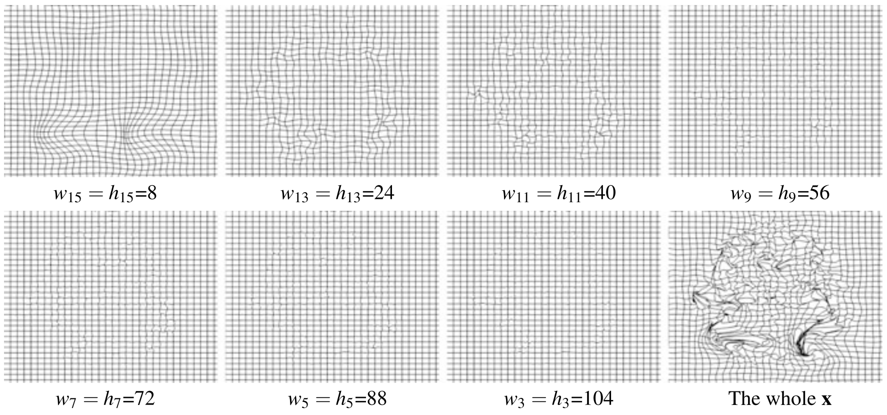


Fig. 5 $\mathbf{x}^k$ at (w_k, h_k)-level of Fig. 4(c)

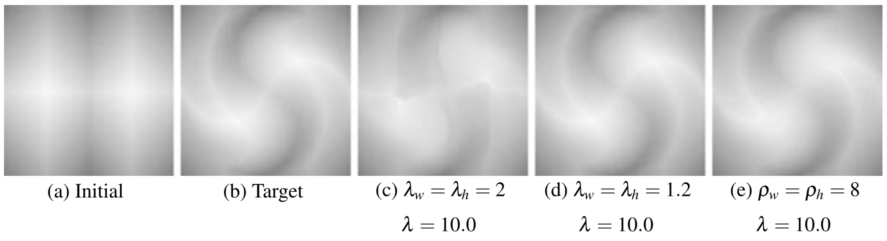

(a) Initial
(b) Target
(c) $\lambda_w = \lambda_h = 2$ $\lambda = 10.0$
(d) $\lambda_w = \lambda_h = 1.2$ $\lambda = 10.0$
(e) $\rho_w = \rho_h = 8$ $\lambda = 10.0$

Fig. 6 The effects of different N-sequences. (**c**) and (**d**) are the result of using N-sequence (20). (**e**) is the result using N-sequence (21)

Table 2 Numerical results obtained from different similarity metrics of experiment 6, the results with bold face are the best

SM	Primary metric	Figure 6(c) $\lambda = 10.0$, $\lambda_w = \lambda_h = 2$	Figure 6(d) $\lambda = 10.0$, $\lambda_w = \lambda_h = 1.2$	Figure 6(e) $\lambda = 10.0$, $\rho_w = \rho_h = 8$
NSD	8210	1890	**0**	37
MSD	0.245269	0.071634	**0.006070**	0.020314
NCC	0.710945	0.976657	**0.999823**	0.998297
EDI	3.853722	2.817571	**0.835384**	1.538645
MI	1.569308	2.246680	**4.054850**	3.459266
LD	72.00000	37.00000	**6.000000**	22.00000
TUD	365845.0	99490.00	**7183.000**	21869.00

Regularization Term $\int_\Omega (g-1)^2$ The second item of the energy functional (2) is used to make g as close to one as possible. It is well known that $\sqrt{g}$ is the area element of the surface $\mathbf{x}(u, v)$. Forcing g close to one everywhere yields a smoother $\mathbf{x}$.

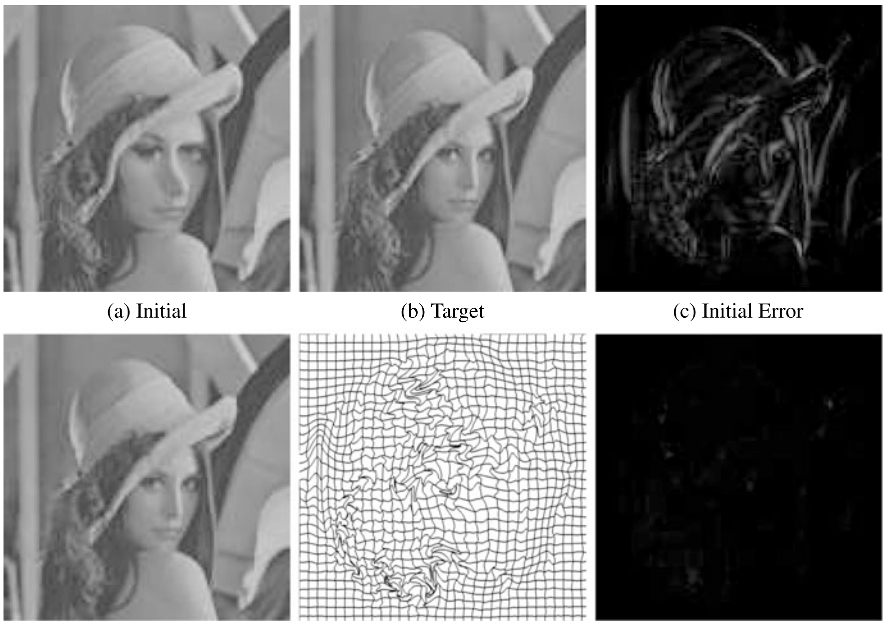


Fig. 7 Result of an artificially deformed, initial image is the distorted image of target image

Figure 3 shows the effect of the coefficient λ of $\int_{\Omega}(g-1)^2$. The target image in this figure is a rotation of the initial image. It is obvious that larger λ makes $\mathbf{x}$ smoother. However, the alignment result may be less accurate in the details. This property is also observed from Fig. 4, where the target image is the mirror projection of the initial image about its vertical middle-axis.

The Choice of the N-Sequence By choosing an appropriate N-sequence which is increasingly ordered, we can align the structures in the images from coarse to fine. Figure 5 is used to illustrate such a process for aligning the images shown in Fig. 4, where $\mathbf{x}^{(k)}$ at (w_k, h_k)-level are presented. Each figure is the plot of the isoparametric curves of $\mathbf{x}^{(k)}(u, v)$. We can easily see that the variations are aligned from large scale to small gradually. As the level number k from 15 approaching to 3, the isoparametric curves of $\mathbf{x}^{(k)}(u, v)$ close to uniform grids. We therefore stop the computation at (w_3, h_3)-level.

It should be pointed out that the choice of the N-sequence has significant influence on the alignment result. In Fig. 6, we intend to align two 128×128 images. We use N-sequence defined by (20) with $\lambda_w = \lambda_h = 2$ for figure (c), and $\lambda_w = \lambda_h = 1.2$ for figure (d). We use (21) with $\rho_w = \rho_h = 8$ for figure (e). Clearly using sequence (20) with $\lambda_w = \lambda_h = 1.2$ leads to more desirable results, similar conclusion can be seen from Table 2.

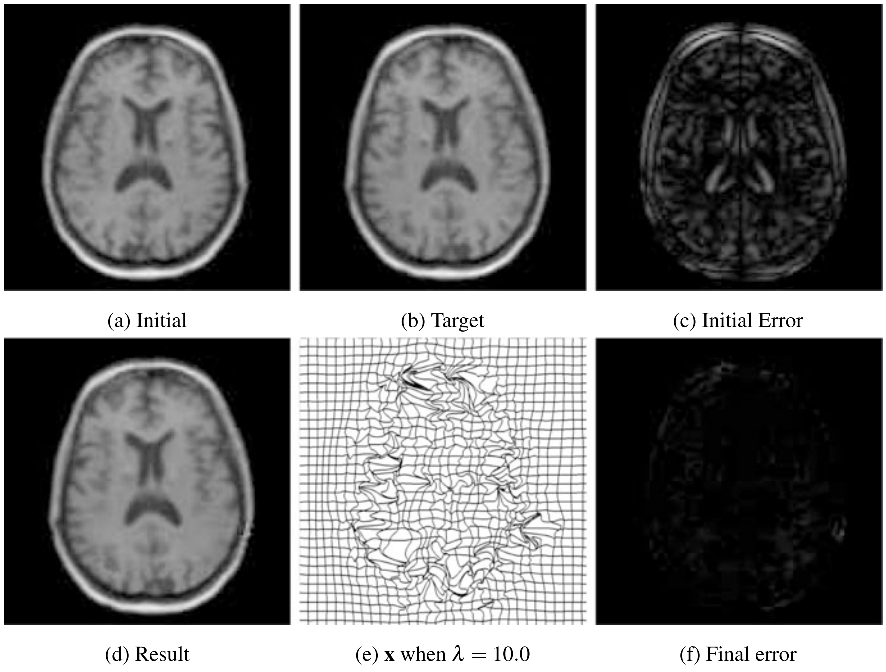


Fig. 8 Result of an artificially deformed MRI-slices, target image is the mirrored image of initial image

Image Sampling After the deformation mapping $\mathbf{x}$ is finally determined, $I_i(\mathbf{x}(u,v))$ is the aligned image of $I_t(u,v)$. To obtain a discrete version of $I_i(\mathbf{x}(u,v))$ at the grid points (u_i, v_j) with

$$u_i = \frac{i}{M}, \qquad v_j = \frac{j}{N}, \quad i = 0,\dots,M, j = 0,\dots,N,$$

we first compute $\mathbf{x}_{ij} = \mathbf{x}(u_i, v_j) \in \Omega$, then find k and l, such that

$$\mathbf{x}_{ij} \in \big[k/M, (k+1)/M\big] \times \big[l/N, (l+1)/N\big].$$

Then the discrete version of $I_i(\mathbf{x}(u,v))$ at the grid points (u_i, v_j) is computed as the bilinear interpolation of $I_i(\mathbf{x}(u,v))$ over the rectangle $[k/M, (k+1)/M] \times [l/N, (l+1)/N]$ at the point $\mathbf{x}_{ij}$.

In order to obtain reasonable results, we choose the N-sequence as (21) with $\rho_w = \rho_h = 8$, and we set $\lambda = 10.0$ and $\varepsilon = 0.001$ in the Figs. 7 and 8 below. The target image in Fig. 7 is obtained by representing the initial image as a bicubic B-spline function (see (23)), then perturbing randomly the coefficients f_{ij} and finally re-sampling the perturbed one. The target image in Fig. 8 is the mirrored image of the initial images.

From the mappings as shown in each experiments which are the plots of the isoparametric curves of maps $\mathbf{x}$ of the corresponding experiments, we can apparently

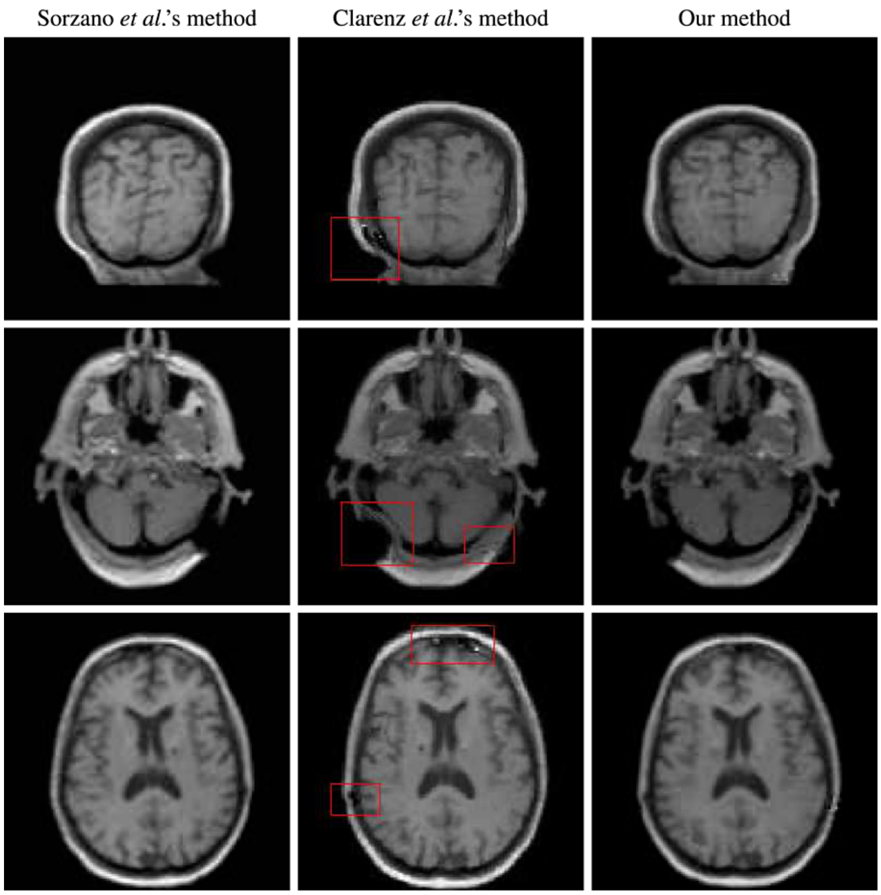


Fig. 9 Comparison of our results of Figs. 1, 4 and 8 with Sorzano et al.'s results and Clarenz et al.'s results. Sorzano et al.'s method does not yield correct results for these experiments. Some flaws (see the parts in the *boxes*) can be observed in Clarenz et al.'s results in the second column

see that our method yields C^2 smooth $\mathbf{x}$ which satisfies conditions (i)–(iii). From the error images (45) as shown, we can easily observe that the error images are almost black in the experiments conducted.

Finally, we compare the performance of our method with two popular methods of Sorzano et al. (see [27]) and Clarenz et al. (see [9]) through Fig. 9 and Table 3, 4 and 5. The initial and target images from the first row to the third row of Fig. 9 are the same as the ones of Figs. 1, 4 and 8, respectively. The first column shows that Sorzano et al.'s method does not yield correct alignment results. The figures in the second column show that Clarenz et al.'s method works for these examples but there are tiny flaws in their results. The images in the third column show that our method yields very desirable results.

Table 3 Numerical results obtained from different similarity metrics of experiment 1, the results with bold face are the best

SM	Primary metric	Figure 1(d)	Figure 1(e)	Sorzano et al.'s	Clarenz et al.'s
NSD	2936	**231**	621	3572	1301
MSD	0.147169	**0.031137**	0.053215	0.147306	0.072775
NCC	0.915203	**0.996315**	0.989065	0.956905	0.979665
EDI	2.152111	**1.428342**	1.644643	2.220481	1.860739
MI	1.178106	**1.494839**	1.362743	1.264463	1.252158
LD	145.0000	**49.00000**	100.0000	177.0000	194.0000
TUD	130177.0	**25091.00**	40830.00	173051.0	63886.00

Table 4 Numerical results obtained from different similarity metrics of experiment 4, the results with bold face are the best

SM	Primary metric	Figure 4(d)	Figure 4(e)	Sorzano et al.'s	Clarenz et al.'s
NSD	3372	**182**	254	5906	1258
MSD	0.160801	**0.030659**	0.035476	0.201012	0.070510
NCC	0.888354	**0.996038**	0.994656	0.927741	0.979117
EDI	2.714375	**1.697832**	1.803896	2.981768	2.241050
MI	1.233013	**1.769513**	1.698944	1.364450	1.455032
LD	144.0000	**55.00000**	87.00000	176.0000	102.0000
TUD	158586.0	**29530.00**	34361.00	268187.0	70375.00

Table 5 Numerical results obtained from different similarity metrics of experiment 8, the results with bold face are best

SM	Primary metric	Our method	Sorzano et al.'s	Clarenz et al.'s
NSD	4995	**177**	3317	709
MSD	0.244006	**0.029471**	0.111717	0.057463
NCC	0.822478	**0.997423**	0.974174	0.993691
EDI	3.053663	**1.618538**	2.640437	2.275920
MI	1.236784	**2.008639**	1.524631	1.815054
LD	164.0000	**68.00000**	93.00000	140.0000
TUD	262917.0	**26907.00**	147383.0	64245.00

Tables 3, 4 and 5 are the numerical results of experiments 1, 4 and 8 for different similarity metrics. Although various similarity metrics emphasize on different aspects, it can be seen from the data in the table that, our method performs the best under various similarity metrics.

10 Conclusions

A new algorithm for flexible alignment has been presented. It combines the ideas of flexible alignment based on B-spline reparametrization, L^2-gradient flow and multi-resolution. The L^2-gradient flow is efficiently solved in the B-spline finite element space. Our method is effective, robust and capable of capturing the variation of the initial and target images, from large scale to small. We have proved the regularity of the mapping $\mathbf{x}(u, v)$ under certain conditions. We also proved that there exists a mapping $\mathbf{x}_0(u, v) \in X$ satisfying (i)–(iii) such that the energy functional (2) is minimized. For the systems of ordinary differential equations derived from the finite element discretization in the spatial direction, the existence and uniqueness of solution have been proved.

Acknowledgements This work is supported in part by NSFC key project under the grant 10990013, NSFC Funds for Creative Research Groups of China (grant No. 11021101) and NSFC project under the grant 81173663.

References

1. Bajcsy R, Kovacic S (1989) Multiresolution elastic matching. Comput Vis Graph Image Process 46(1):1–21
2. Besl P, McKay H (1992) A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell 14(2):239–256
3. Brown L (1992) A survey of image registration techniques. ACM Comput Surv 24(4):325–376
4. Buck RC (1956) Advanced calculus. McGraw-Hill, New York
5. Buzug T, Weese J, Fassnacht C, Lorenz C (1997) Image registration: convex weighting functions for histogram-based similarity measures. In: Proceedings of the first joint conference on computer vision, virtual reality and robotics in medicine and medial robotics and computer-assisted surgery, pp 203–212
6. Capel D, Zisserman A (1998) Automatic mosaicing with super-resolution zoom. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp 885–891
7. Carreras I, Sorzano C, Marabini R, Carazo J, Solorzano C, Kybic J (2006) Consistent and elastic registration of histological sections using vector-spline regularization. In: Computer vision approaches to medical image analysis, vol 4241, pp 85–95
8. Chen Q, Defrise M, Deconinck F (1994) Symmetrical phase-only matched filtering of Fourier–Mellin transforms for image registration and recognition. IEEE Trans Pattern Anal Mach Intell 16(12):1156–1168
9. Clarenz U, Droske M, Rumpf M (2002) Towards fast non-rigid registration. In: Inverse problems, image analysis and medical imaging, AMS special session interaction of inverse problems and image analysis. Am Math Soc, Providence, pp 67–84
10. do Carmo MP (2004) Differential geometry of curves and surfaces. China Machine Press, Beijing
11. Droske M, Rumpf M (2003) A variational approach to non-rigid morphological image registration. SIAM J Appl Math 64(2):668–687
12. Grevera G, Udupa J (1998) An objective comparison of 3-D image interpolation methods. IEEE Trans Med Imaging 17(4):642–652

13. Haber E, Modersitzki J (2006) A multilevel method for image registration. SIAM J Sci Comput 27(5):1594–1607
14. Hsieh PF, Sibuya Y (1999) Basic theory of ordinary differential equations. Springer, Berlin
15. Keller Y, Shkolnisky Y, Averbuch A (2005) The angular difference function and its application to image registration. IEEE Trans Pattern Anal Mach Intell 27(6):969–976
16. Lemieux L, Jagoe R, Fish D, Kitchen N, Thomas D (1994) A patient-to-computed tomography image registration method based on digitally reconstructed radiographs. Med Phys 21(11):1749–1760
17. Levenberg K (1944) A method for the solution of certain problems in least squares. Q Appl Math 2:164–168
18. Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 11(2):431–441
19. McLauchlan PF, Jaenicke A (2002) Image mosaicing using sequential bundle adjustment. Image Vis Comput 20(9–10):751–759
20. Milanfar P (1999) Two-dimensional matched filtering for motion estimation. IEEE Trans Image Process 8(3):438–444
21. Munkres JR (2004) Topology. China Machine Press, Beijing
22. Porat B (1996) A course in digital signal processing, 1st edn. Wiley, New York
23. Reddy B, Chatterji B (1996) An FFT-based technique for translation, rotation, and scale-invariant image registration. IEEE Trans Image Process 5(8):1266–1271
24. Rueckert D, Sonoda LI, Hayes C, Hill DLG, Leach MO, Hawkes DJ (1999) Nonrigid registration using free-form deformations: application to breast MR images. IEEE Trans Med Imaging 18(8):712–721
25. Sdika M (2008) A fast nonrigid image registration with constraints on the Jacobian using large scale constrained optimization. IEEE Trans Med Imaging 27(2):271–281
26. Segman J (1992) Fourier cross correlation and invariance transformations for an optimal recognition of functions deformed by affine groups. J Opt Soc Am A 9(6):895–902
27. Sorzano C, Thévenaz P, Unser M (2005) Elastic registration of biological images using vector-spline regularization. IEEE Trans Biomed Eng 52(4):652–663
28. Van den Elsen P, Maintz J, Pol E, Viergever M (1995) Automatic registration of CT and MR brain images using correlation of geometrical features. IEEE Trans Med Imaging 2(14):384–396
29. Viola P, Wells W III (1995) Alignment by maximization of mutual information. In: Proceedings of the fifth international conference on computer vision, pp 16–23
30. Xu G, Shi Y (2006) Progressive computation and numerical tables of generalized Gaussian quadrature formulas. J Numer Methods Comput Appl 27(1):9–23
31. Zhang GQ (1997) Functional analysis lecture. Peking University Press, Beijing
32. Zitová B, Flusser J (2003) Image registration methods: a survey variational problems. Image Vis Comput 21(11):977–1000

Shape Based Conditional Random Fields for Segmenting Intracranial Aneurysms

Sajjad Baloch, Erkang Cheng, and Tong Fang

Abstract Studies have found strong correlation between the risk of rupture of intracranial aneurysms and various physical measurements on the aneurysms, such as volume, surface area, neck length, among others. Accuracy of risk prediction relies on the accuracy of these quantities, which in turn, is determined by the precision of the underlying segmentation algorithm. In this paper, we propose an algorithm for the separation of aneurysms in pathological vessels. The approach is based on conditional random fields (CRF), and exploits regional shape properties for unary, and layout constraints for pair-wise potentials to achieve a high degree of accuracy. To this end, we construct very rich rotation invariant shape descriptors, and couple them with randomized decision trees to determine posterior probabilities. These probabilities define *weak priors* in the unary potentials, which are also combined with *strong priors* determined from user interaction. Pairwise potentials are used to impose smoothness as well as spatial ordering constraints. The proposed descriptor is independent of surface orientation, and is richer than existing approaches due to attribute weighting. The conditional probability of CRF is maximized through graph-cuts, and the approach is validated with real dataset w.r.t. the groundtruth, resulting in the area overlap ratio of 88.1%. Most importantly, it successfully solves the "touching vessel leaking" problem.

1 Introduction

Intracranial aneurysms is a major vascular disease in the brain, attributed to local weakening of the vessel wall. It manifests in the form of bulging (saccular aneurysm) or dilation (fusiform aneurysm) as shown in Fig. 1. Intracranial aneurysms frequently occur near areas of high arterial curvature or bifurcations, as these regions usually experience more hemodynamic stress [7]. If left untreated, an aneurysm grows in size, thereby further weakening the wall strength and increasing the risk of rupture, which may lead to subarachnoid hemorrhage, neurological

S. Baloch (✉) · E. Cheng · T. Fang
Siemens Corporate Research, Princeton, NJ 08540, USA
e-mail: sajjad.baloch@siemens.com

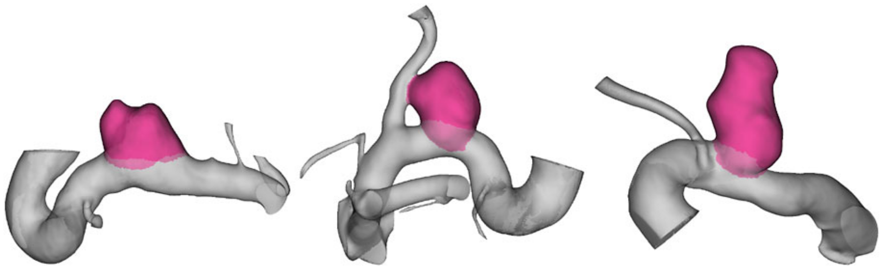


Fig. 1 Examples of intracranial sidewall aneurysms

deficits, and in up to 56% of cases mortality [14]. In order to prevent its growth and reduce the risk of rupture, surgical intervention is required, where stents, wire coils, and other embolic material or devices are placed not only to enforce the vessel wall, but also to alter the blood flow pattern, thereby reducing the pressure on regions more prone to rupture.

Once diagnosed, aneurysms are carefully monitored and examined, before making a surgical decision. To this end, geometry of aneurysm plays a crucial role. Physicians analyze various measurements of geometric primitives [5] evaluated on an aneurysm, which allows them to carry out surgical planning. Separation of the aneurysm from the healthy vessel, therefore, serves as a critical step, whose accuracy determines the eventual outcome in terms of surgical decisions, device selection, as well as patient recovery. The problem is, however, very challenging due to the complex topology and geometry of the underlying blood vessel, and its large inter-patient variation.

Early approaches on aneurysms separation mainly focused on deformable models. By grouping local shape descriptors, McLaughlin and Nobel [10] employed a region-splitting algorithm to segment the aneurysm from the neighboring vasculature. Their approach, however, fails to yield reliable results for wide-neck saccular aneurysms. Wong and Chung [16] modeled the healthy part of the vessel as a tubular deformable model, and determine the abnormal structure of aneurysms as the complement of the healthy model. Their approach does not provide protection against the leaking of the deformable model inside the aneurysm. Ford et al. [6], on the other hand, presented a method to reconstruct the parent artery by removing the aneurysm. They also utilized deformable model to model vessel of parent. The method needs smooth surface and is limited to the morphology of parent artery surface.

More recently, Mohamed et al. [11] utilized mesh based snakes for computer-aided planning for endovascular treatment of intracranial aneurysms (CAPETA). A major limitation of their approach was its inability to adapt to topological changes in the aneurysm boundary, resulting in erroneous results for meshes with touching vessels. Furthermore, due to the local nature of the snakes, their proposed method fails to guarantee a globally optimal solution. To address these issues, Sgouritsa et al. [13] proposed a curvature based graph-cut strategy for segmenting the 3D vessel mesh, guided by strong priors, which in turn were determined from user input in the form of 3 seed points. Due to its dependence on strong priors, the accuracy of this approach is determined by the accuracy of the prior computation algorithm, and remains highly sensitive to the manual user input. Also, in practical situations, the Gaussian curvature is too coarse to adequately capture underlying shape variations. It is, therefore, required to construct richer shape representations, which adequately capture the vessel-aneurysm differences.

In this paper, we propose a novel method for aneurysm separation based on conditional random fields (CRF) by augmenting both unary and pairwise potentials. Unary potentials are decomposed into strong and weak priors. The former are determined from interactive user input, and the latter are estimated via randomized decision forest as posterior probabilities of novel shape descriptors. More specifically, we propose algorithms for accurate strong prior determination. The shape descriptors, in turn, are constructed from underlying local and regional geometry. Pairwise potentials are encoded to impose spatial ordering as well as smoothness constraints. The conditional probability of CRF is maximized through graph-cuts. This formulation allows multitude of improvements over existing approaches. Like [13], our method is not limited by topological variations. Unlike [13], we construct shape descriptors specifically tailored for the problem under consideration, i.e., for the separation of blob like aneurysms on tubular vessels. Inferencing from examples allows us to specify weak priors, thereby allowing more flexibility. Smoothness and layout constraints penalize the assignment of inconsistent labels. We validate the proposed method with real dataset comprising of 27 3D digital subtraction angiographic (DSA) images in the CAPETA framework [11], and carry out a comparison with [13]. The experimental results demonstrate that (1) in all cases our improved prior seeds were in accordance with the groundtruth, and (2) our aneurysm method consistently outperforms [13].

The rest of the paper is organized as follows: We first formulate aneurysm separation as CRF in Sect. 2, followed by the construction of shape descriptors in Sect. 3. Unary and pairwise potentials are developed in Sect. 4, along with the algorithms for finding strong priors. Experimental results are presented in Sect. 5, before we concluded in Sect. 6.

2 Problem Formulation

For robust aneurysm separation algorithm, we formulate the problem as Conditional Random Fields (CRF) that are driven by rich shape descriptors.

2.1 Conditional Random Fields

Given a triangular mesh $\mathcal{T} := (P, E)$ representing a surface $\mathcal{M}$ embedded in $\mathbb{R}^3$, with $P := \{p_i = p(v_i)\}$ denoting the set of positions at vertices $V = \{v_i\}$, and $E = \{e_k\}$ denoting the edges connecting the respective vertices. $p : V \to P, v_i \mapsto p_i$, therefore, forms an isomorphism from the undirected graph $\mathcal{G} := (V, E)$ to $\mathcal{T}$. The problem under consideration is to find a binary labeling $l : V \to L = \{l_v, l_a\}$, $v_i \mapsto l_i := l(v_i)$ that partitions $\mathcal{G}$ into two segments. For aneurysm separation problem, this amounts to segmenting a mesh into the healthy vessel and the aneurysm regions, and is dictated by some feature properties of the underlying geometry.

Suppose x_i defines a shape or geometric descriptor of v_i in $\mathcal{T}$ possibly with a non-local region of support, then the isomorphism $x : V \to X$, $v_i \mapsto x_i := x(v_i)$ captures the underlying geometric description of v_i and the corresponding graph $X := (x_i, e_i)$ may be exploited to find the partitioning. The optimal partition is the one that maximizes the joint distribution of (X, L):

$$\begin{aligned} l^* &= \arg\max_l P(X, l; \theta) \\ &= \arg\max_l P(X; \theta) P(l|X; \theta) \\ &= \arg\max_l P(l|X; \theta), \end{aligned} \tag{1}$$

where θ represents a distribution parameter. Conditional random field setting allows one to simplify the above expression, by considering a local (1-ring) neighborhood $N(v_i)$ at each vertex, v_i:

$$P(l|X; \theta) = \frac{1}{Z(\theta, X)} \prod_i \phi_i(l_i, X; \theta) \prod_{(i,j)} \psi_{ij}(l_i, l_j, X; \theta), \tag{2}$$

where the *unary potential*, ϕ_i, captures the posterior distribution of labels at v_i, and the *pairwise potential*, ψ_{ij}, models the neighborhood labeling relations allowing one to impose spatial constraints. Depending on the application, one may incorporate various constraints, such as smoothing or spatial ordering, in the form of soft layout consistency or hard layout consistency [18].

The problem is, therefore, reduced to constructing appropriate shape descriptors x, estimating posterior probabilities, and specifying a pairwise potential that is suitable for the application. Maximization of Eq. (2) is identical to the minimization of the following energy functional:

$$E(l|X) = -\sum_i \log \phi_i(l_i, X; \theta) - \sum_{(i,j)} \log \psi_{ij}(l_i, l_j, X; \theta). \tag{3}$$

3 Shape Descriptors

We are interested in separating blob like structures, such as aneurysms, from somewhat tubular regions, such as blood vessels. For each point p_i on $\mathcal{T}$, we extract surface features, $\mathbf{F}_i$, that are highly discriminating between these kind of regions. They include regional shape as well as local geometry.

3.1 Local Descriptors

For local descriptors, we rely on the curvature information. We exploit Gaussian curvature κ, mean curvature H, and maximum and minimum principal curvatures, κ_1 and κ_2. In addition, we also consider maximum and minimum principal directions, $\mathbf{v}_1$ and $\mathbf{v}_2$, and the shape index, s:

$$s = \frac{2}{\pi} \arctan \frac{\kappa_2 + \kappa_1}{\kappa_2 - \kappa_1}, \quad \kappa_2 \geq \kappa_1. \tag{4}$$

3.2 Regional Shape Descriptors

The regional shape information is captured through various shape descriptors, namely (1) Wilmore energy, and (2) regional attribute weighted geodesic shape contexts.

3.2.1 Wilmore Energy

The Wilmore energy of a vertex $v_i \in \mathcal{G}$ is defined in terms of the isomorphism $\mathcal{T}$, and its n-ring neighborhood, S_i:

$$W := \int_{S_p} \left(H^2 - \kappa\right) dA, \tag{5}$$

where dA is a surface area element of S_p.

Note that $W(p) \geq 0$, with $W(p) = 0$ if and only if p is convex, and v and all of its neighbors, S_p lie on a common sphere [2]. Consequently, big blob like structures, such as aneurysms, are characterized by small Wilmore energy.

3.2.2 Regional Attribute Weighted Geodesic Shape Contexts

Shape contexts [1] are defined by creating bins of various spatial parameters, followed by constructing a histogram that counts points falling in each bin. [12] proposed 3D shape contexts for surface matching, which bin the 3D space via spherical coordinates. Such descriptors, however, are not invariant to the orientation of the surface. To overcome this problem, we carry out geodesic binning for each point $p \in \mathcal{M}$, as illustrated in Fig. 2. Geodesic distances are intrinsic to a surface, and, therefore, lead to rotation invariance. If $g(p, \cdot)$ is the geodesic distance from point p, then geodesic binning, within a local neighborhood $\mathcal{G}_r(p) := \{\forall q \in \mathcal{M} : g(p,q) \leq r\}$, is defined as $\{i = 0, \ldots, k-1 : g_i \leq g_i(p,q) < g_{i+1}\}$ with $g_k = r$. Histograms are then generated by computing the concentration of various surface attributes within each bin. Although apparently similar to [18], which has also utilized

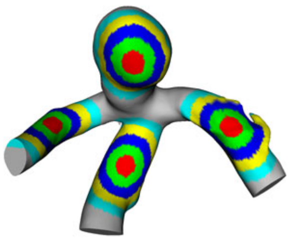


Fig. 2 Geodesic binning at selected vertices

geodesic binning for layout consistent segmentation of ear impressions, we consider regional binning instead of doing it globally. In problem under consideration, global binning causes a confounding effect, as random branching across individuals introduces noise in the sample data. Furthermore, we consider various histograms of diverse surface attributes, such as Gaussian curvature weighted point distribution, area distribution, and the distribution of connected components.

The Gaussian curvature weighted point distribution shape context $\mathbf{f}^g = (f_0^g, \ldots, f_{k-1}^g)$ computes the number of points falling within each bin normalized by the total number of points in all bins to create a distribution. Finally each bin is weighted by the Gaussian curvature averaged within it. For meshes with non-uniform triangulation, the discrete number of points does not truly represent the underlying geometry. For this reason, we augment our feature set, by computing the surface area of each bin to create a second shape context distribution $\mathbf{f}^a = (f_0^a, \ldots, f_{k-1}^a)$. It should be noted that this does not make the previous shape context redundant, since for non-uniform triangulation the point density is related to interesting features. We, therefore, retain both descriptors, and will later carry out feature selection strategy to retain the most distinguishing features.

The third and final shape context $\mathbf{f}^c = (f_0^c, \ldots, f_{k-1}^c)$ captures the number of connected components in each bin. This shape context is particularly important to differentiate flat or thick regions from narrow tubular areas. The hypothesis is that such a descriptor will help in segmenting a touching vessel from an aneurysm.

Visibility from Reference Point Due to their almost convex shape, most points on an aneurysm are visible from their centroid r. This allows us to consider a very powerful feature in our feature design:

$$\nu_r(p) := \begin{cases} 1 & \text{if } p \text{ is visible from } c \\ 0 & \text{o.w.} \end{cases} \tag{6}$$

Since the centroid is not known a priori, we assume that its rough location is specified by a user as a reference point.

All these features are combined in a feature vector $\mathbf{F}_p := (\kappa, H, \kappa_1, \kappa_2, s, \mathbf{v}_1, \mathbf{v}_2, W, \mathbf{f}^g, \mathbf{f}^a, \mathbf{f}^c, \nu_r)$ as a local and regional descriptor of a point $p \in \mathcal{T}$.

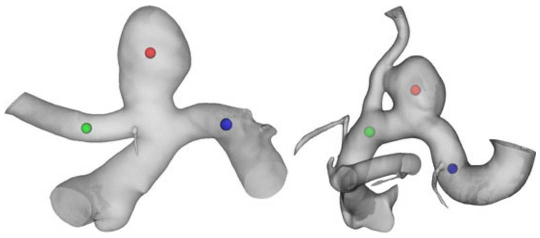


Fig. 3 Reference input provided by a user in the form of dome, proximal, and distal points

4 Potential Specification

After constructing the shape descriptors, we are in a position to specify unary potentials and pairwise potentials. Unary potentials are determined from the probability of aneurysm and pairwise potentials are used to impose smoothness and layout constraints. In this paper, we decompose the unary potentials into strong and weak priors; the former is determined from the user input and latter incorporates posterior probabilities learned from shape descriptors. The term strong highlights high confidence, and hence, large weights for such priors.

Major differences of our approach from [13] include: (1) rich shape descriptors, as opposed to scalar descriptor (Gaussian curvature) in [13]; (2) we employ CRF framework, which naturally encodes the posterior probabilities of aneurysm; (3) weak prior unary potentials are estimated by randomized decision trees; (4) new robust algorithms for finding strong priors, where [13] finds erroneous priors; (5) smoothness and layout constraints in pairwise potentials.

4.1 Strong Unary Potentials

For strong priors, we adopt an approach similar to recently proposed CAPETA framework [11], where a user specifies so-called *dome*, *proximal* and *distal* points as shown in Fig. 3. The dome point p_D very roughly provides the location of the aneurysm relative to the vessel, and proximal p_p and distal p_d points specify the region of interest for subsequent analysis in CAPETA. The aneurysm, therefore, always falls between the proximal and distal point input. It should be noted that these points do not lie on the surface, but are centered inside the vessel in the viewing direction. The closest point projection $\tilde{p}_D$ of p_D is used as p_{ref} in Eq. (10), and p_D is employed as the reference point r for the visibility feature of Eq. (6).

Vessel Prior We exploit these points for establishing strong priors in our unary potential. [13] used two kind of contours for strong priors. The strong prior for the vessel was determined from the proximal p_p and distal p_d points, by projecting them to the mesh, $\tilde{p}_p$ and $\tilde{p}_d$, and then using the geodesic h between them as the vessel prior. The limitation of this approach is that frequently touches the boundary between the aneurysm and the healthy vessel, therefore, leading to incorrect

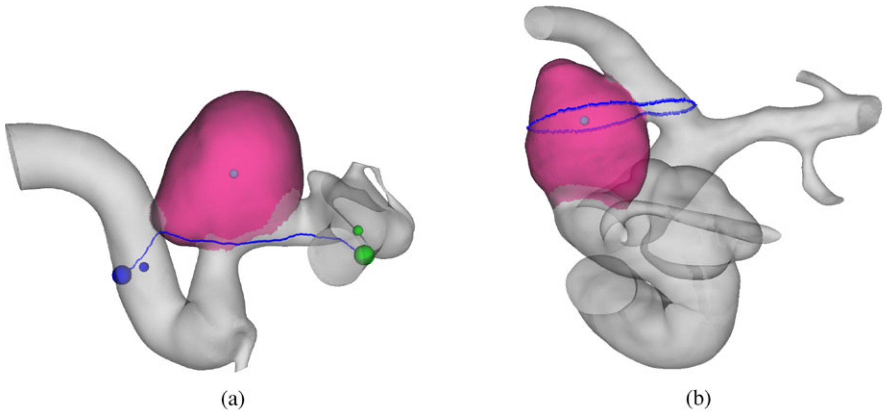


Fig. 4 Problems with the strong priors of [13]: (**a**) Vessel prior; (**b**) Aneurysm prior. *Small spheres* represent the reference points that lie inside the vessel. *Big spheres* are the mapping of the references points to the vessel surface

specification of the strong vessel priors as illustrated in Fig. 4(a). To overcome this limitation, we enforce the geodesic to stay away from the boundary. The optimal geodesic is the one that simultaneously minimizes the distance between $\tilde{p}_p$ and $\tilde{p}_d$, as well as the minimum mean curvature along the path:

$$h^*(\tilde{p}_p, \tilde{p}_d) := \arg\min_{g \in \mathcal{M}} \int_{\tilde{p}_p}^{\tilde{p}_d} \varphi(H)\pi\big(h(\tilde{p}_p, \tilde{p}_d)\big)\, dh, \tag{7}$$

where φ is a decreasing functional of mean curvature H, and π is the length of the geodesic. As shown in Fig. 4(b), the modified geodesic leads to more reliable vessel prior A_v.

Aneurysm Prior In [13], aneurysm prior was based on a planar contour $\mathcal{C} := \mathcal{T} \cap P$ found as an intersection between $\mathcal{T}$ and a plane P centered at the dome point p_D. The plane normal $\mathbf{n}$ was defined as the direction of a vector found as follows. First, the closest point p_c to the dome point p_D is determined on the vessel centerline between p_p and p_d. $\mathbf{n}$ is then:

$$\mathbf{n} := \frac{p_D - p_c}{\|p_D - p_c\|_2}. \tag{8}$$

Finally, the intersection contour is employed as the strong prior for the aneurysm region. The utility of this prior is limited to the extent that there is only one intersection contour, which is usually not true. This problem can be resolved by selected the correct single connected component, in the case of touching vessels. In addition, the plane may cut across at the touching point, and the intersection contour may consist of both aneurysm and the touching vessel part. We exploit the minimum principal curvature to decompose the contour into various segments, and eventually use the largest connected segment as the strong aneurysm prior A_a.

4.2 Weak Unary Potential

Weak unary potentials are learned from examples. In general, x_i can reside in some high dimensional space, and the learning process may be quite challenging. Given a dataset of pathological vessels, and the associated expert labeled groundtruth, a randomized decision forest is constructed for the classification similar to [15, 18], based on the extraction of the above mentioned shape descriptors. The advantages of randomized forests include: (1) its built-in feature selection mechanism, where maximum information gain is used for node splitting, and (2) its ability to avoid over-fitting without pruning. For test vessels, posterior probabilities are computed and used as weak priors.

The strong priors are incorporated in the unary potential as follows:

$$-\log \phi_i(l_i, X; \theta) = \begin{cases} \gamma_3 & \text{if } p_i \in A_a \text{ and } l_i = l_v \\ \gamma_4 & \text{if } p_i \in A_v \text{ and } l_i = l_a \\ -\log \phi_i(l_i, X; \theta) & \text{o.w.} \end{cases} \tag{9}$$

where γ_3, γ_4 are the penalization costs for incorrect label assignments.

4.3 Pairwise Potential

In order to derive pairwise potential, we assume that a reference point p_{ref} is given in the region l_a. A spatial layout constraint is then introduced that penalizes the assignment of l_v to a vertex that is closer in geodesic distance sense to p_{ref} than its neighbor with a label l_a. Similarly, we penalize the assignment of different labels to neighboring vertices to ensure a smoothness constraint:

$$\begin{aligned} -\log \psi_{ij}(l_i, l_j, X; \theta) = {} & \exp\big(\alpha(\kappa_i + \kappa_j)\big) \\ & + \begin{cases} 0 & \text{if } l_i = l_j \\ \gamma_1 & \text{if } l_i \neq l_j, l_i = l_a, \\ & g(p_i, p_{\text{ref}}) \geq g(p_j, p_{\text{ref}}) \\ \gamma_2 & \text{if } l_i \neq l_j \end{cases} \end{aligned} \tag{10}$$

where κ_i and κ_j denote the Gaussian curvature at points $p_i, p_j \in \mathcal{T}$. The first term on the right hand side, therefore, penalizes the assignment of identical labels to neighboring vertices with high curvature edges. γ_1 and γ_2 are the costs assigned empirically or inferred from data. α is an exponential shaping coefficient.

Eventually segmentation is carried out by minimizing Eq. (3) through the α-expansion algorithm [3, 4, 9].

5 Experiments

In this section, we compare the proposed method with [13]. 3D Digital Subtraction Angiographic (DSA) images of 30 patients are acquired, and subsequently thresholded as described in [11], to extract the pathological vessels. 3D triangular meshes

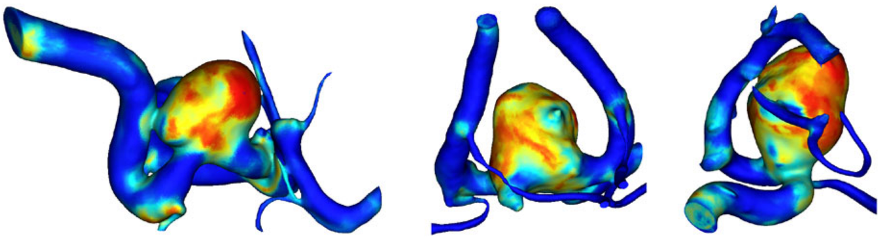


Fig. 5 Posterior label probabilities for example cases

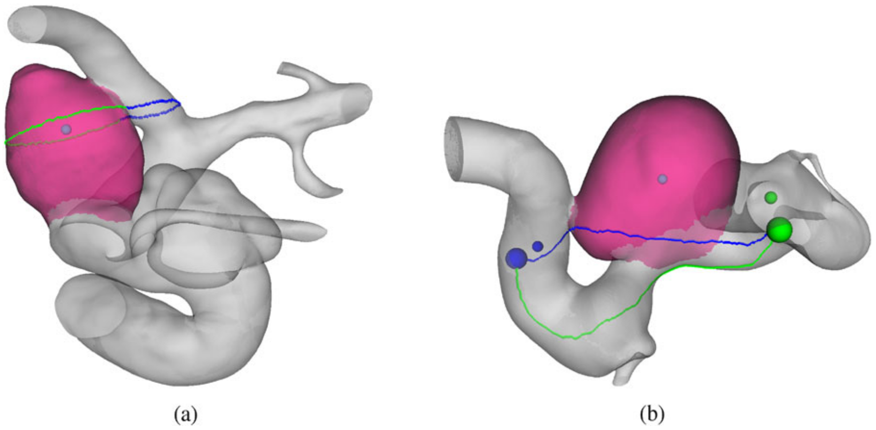


Fig. 6 Strong prior labels determined by the method proposed in Sect. 4.1 for the problem cases of Fig. 4: (**a**) Aneurysm strong prior labels; (**b**) Vessel strong prior labels. Note that the modified aneurysm prior (*green contour*) does not go inside the touching vessel, and the vessel prior (*green contour*) stays away from aneurysm region

were constructed through marching cubes, and were then decimated using quadric decimation [8]. An expert is asked to provide manual labeling on these meshes. Various features were computed to construct regional shape descriptors at each mesh vertex. Curvature estimates were based on adaptive ball approach [17] due to its robustness to noise. 10-ring neighborhood was utilized for Wilmore energy, and 3 mm neighborhood was considered for computing geodesic shape contexts.

4 problem cases were identified, where [13] had failed to provide reasonable results. These examples, along with 4 additional randomly selected meshes were considered as the test dataset. The remaining meshes were added to the training dataset, and training of randomized forest (comprising of 60 trees) was carried out, to compute coarse aneurysm separation and posterior estimates. Some examples are given in Fig. 5.

For the test cases, vessel and aneurysm labels for strong priors were determined as described in Sect. 4.1. For comparison, results are shown in Fig. 6 for the problem cases highlighted in Fig. 4.

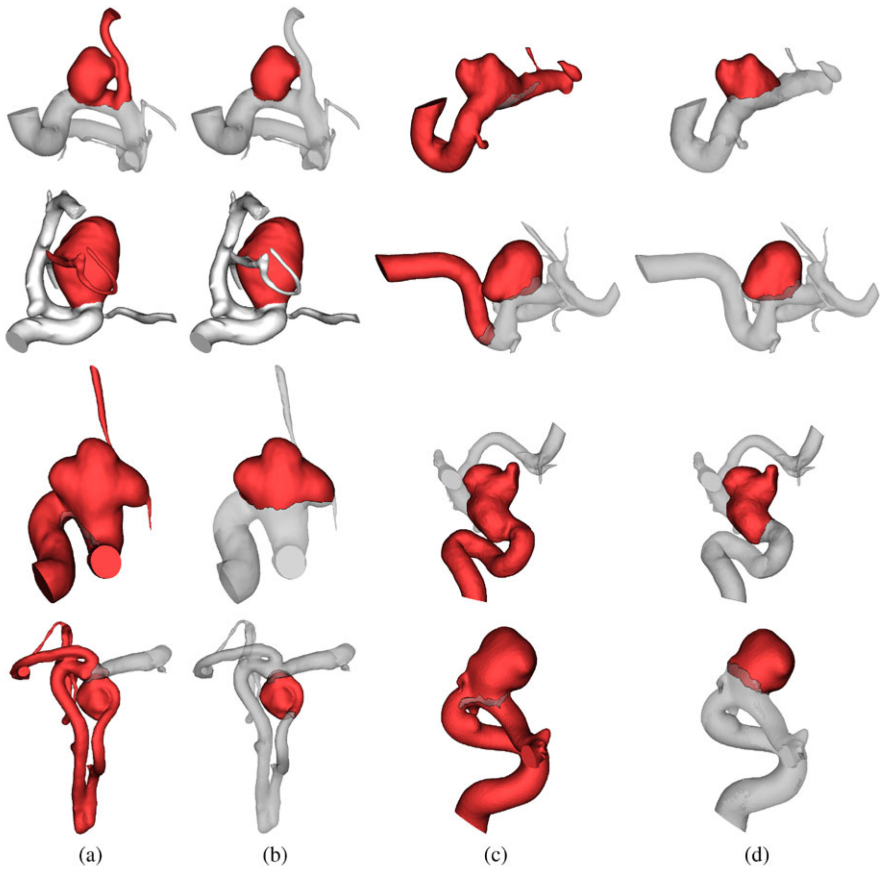


Fig. 7 Comparison with [13]. (**a**) and (**c**) [13]; (**b**) and (**d**) The proposed method. Latter outperforms [13] especially for the touching vessels

Energy functional of Eq. (3) was minimized via two iterations of α-expansion algorithm. A quantitative measure was defined as $QM := (A \cap B)/(A \cup B)$, where A is an automatic separation, whereas B is the groundtruth. $\gamma_1 = 3.5$, $\gamma_2 = 5$, $\gamma_3 = \gamma_4 = 10^9$ in Eqs. (9) and (10) were determined by maximizing QM via alternating variables. Results are given in Fig. 7, providing a qualitative comparison with [13]. Overall, the average QM for our method was computed to be 88.1%, compared with 73.0% for [13]. Most importantly, the leaking problem with [13] into the touching vessels is completely resolved by our approach.

6 Conclusions

We have proposed a CRF based method for the separation of aneurysms from healthy vessel regions. A unique strength of the method is that it effectively combines two sources of information, patient specific strong priors in the form of user input, and data-driven shape priors learned from a large number of aneurysm examples. Along with strong prior obtained from user's interactive input, posterior probabilities learned by randomized decision trees with rich shape descriptors are considered as unary potentials. Smoothness of the segmentation is ensured through pairwise potentials, which are also enriched with spatial ordering constrains. Final segmentation is achieved by minimizing the resulting energy functional with graph-cuts. The proposed method is validated with a real dataset with outstanding results (an accuracy measure of 88.1%). It also perfectly resolves the touching vessel leaking problem.

Acknowledgements We gratefully acknowledge Dr. Michael E. Mawad, St. Luke's Episcopal Hospital, for providing with the valuable data used in the experiments in the paper.

References

1. Belongie SJ, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Trans Pattern Anal Mach Intell 24:509–522
2. Bobenko AI (2005) A conformal energy for simplicial surfaces. In: Combinatorial and computational geometry, vol 52, pp 135–145
3. Boykov Y, Kolmogorov V (2004) An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans Pattern Anal Mach Intell 26:1124–1137
4. Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 23:1222–1239
5. Debrun GM, Aletich VA, Kehrli P, Misra M, Ausman JI, Charbel F, Shownkeen H (1998) Aneurysm geometry: an important criterion in selecting patients for Guglielmi detachable coiling. Neurol Med-Chir 38:1–20
6. Ford M, Hoi Y, Piccinelli M, Antiga L, Steinman D (2009) An objective approach to digital removal of saccular aneurysms: technique and applications. Br J Radiol 82(Special Issue 1):55–61
7. Foutrakis G, Yonas H, Sclabassi R (1999) Saccular aneurysm formation in curved and bifurcating arteries. Am J Neuroradiol 20(7):1309
8. Garland M, Heckbert P (1997) Surface simplification using quadric error metrics. In: Annual conference on computer graphics, pp 209–216
9. Kolmogorov V, Zabih R (2002) What energy functions can be minimized via graph cuts? In: ECCV, pp 65–81
10. McLaughlin RA, Noble JA (2002) Demarcation of aneurysms using the seed and cull algorithm. In: MICCAI, pp 419–426
11. Mohamed A, Sgouritsa E, Morsi H, Shaltoni H, Mawad ME, Kakadiaris IA (2010) Computer-aided planning for endovascular treatment of intracranial aneurysms (CAPETA). In: SPIE medical imaging, vol 7625
12. Park G-J, Novotni M, Klein R (2003) 3D shape matching with 3D shape contexts. In: International conference in central Europe on computer graphics and vision
13. Sgouritsa E, Mohamed A, Morsi H, Shaltoni H, Mawad ME, Kakadiaris IA (2010) Neck localization and geometry quantification of intracranial aneurysms. In: ISBI, pp 1057–1060
14. Weir B (2002) Unruptured intracranial aneurysms: a review. J Neurosurg 96(1):3–42
15. Winn JM, Shotton J (2006) The layout consistent random field for recognizing and segmenting partially occluded objects. In: CVPR, pp 37–44
16. Wong WCK, Chung ACS (2006) Augmented vessels for quantitative analysis of vascular abnormalities and endovascular treatment planning. IEEE Trans Med Imaging 25(6):665–684

17. Yang Y-L, Lai Y-K, Hu S-M, Pottmann H (2006) Robust principal curvatures on multiple scales. In: Eurographics symposium on geometry processing, pp 223–226
18. Zouhar A, Baloch S, Tsin Y, Fang T, Fuchs S (2010) Layout consistent segmentation of 3-D meshes via conditional random fields and spatial ordering constraints. In: MICCAI, pp 113–120

Tetrahedral Image-to-Mesh Conversion Approaches for Surgery Simulation and Navigation

Andrey N. Chernikov, Panagiotis A. Foteinos, Yixun Liu, Michel Audette, Andinet Enquobahrie, and Nikos P. Chrisochoides

Abstract In this paper we evaluate three different mesh generation approaches with respect to their fitness for use in a surgery simulation and navigation system. The behavior of such a system can be thought of as a trade-off between material fidelity and computation time. We focus on one critical component of this system, namely non-rigid registration, and conduct an experimental study of the selected mesh generation approaches with respect to material fidelity of the resulting meshes, shape of mesh elements, condition number of the resulting stiffness matrix, and the registration error. We concluded that meshes with very bad fidelity do not affect the accuracy drastically. On the contrary, meshes with very good fidelity hurt the speed of the mesher due to the poor quality they exhibit. We also observed that the speed of the solver is very sensitive to mesh quality rather than to fidelity. For these reasons, we think that mesh generation should first try to produce high quality meshes, possibly sacrificing fidelity.

A.N. Chernikov (✉) · P.A. Foteinos · Y. Liu · N.P. Chrisochoides
Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
e-mail: achernik@cs.odu.edu

Y. Liu
e-mail: yxliuwm@gmail.com

N.P. Chrisochoides
e-mail: nikos@cs.odu.edu

P.A. Foteinos
Department of Computer Science, College of William and Mary, Williamsburg, VA 23187, USA
e-mail: pfot@cs.wm.edu

M. Audette
Department of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA 23529, USA
e-mail: maudette@odu.edu

A. Enquobahrie
Kitware Inc., Carrboro, NC 27510, USA
e-mail: andinetenqu@kitware.com

Y.(J.) Zhang (ed.), *Image-Based Geometric Modeling and Mesh Generation*, Lecture Notes in Computational Vision and Biomechanics 3,
DOI 10.1007/978-94-007-4255-0_5, © Springer Science+Business Media Dordrecht 2013

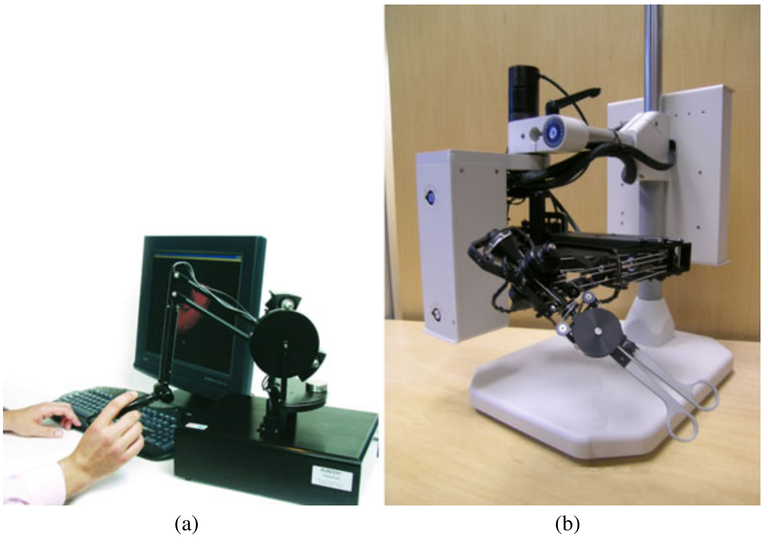


Fig. 1 Commercial haptic devices: (**a**) Sensable's 6 degree-of-freedom (d.o.f.) Phantom 6S/1.5; (**b**) MPB Technologies' 7 d.o.f. Freedom 7S

1 Introduction

Surgical simulation is the application of computers to synthesizing an anatomical response to a simulated therapy. This is achieved through a software program that synthesizes tissue response to virtual surgical tools, typically a mechanical response to cutting or manipulation. This behavior can be thought of as a trade-off between material fidelity and computation time, whose weighted emphasis on one or the other can be characterized as a spectrum. At one end of the spectrum we have *predictive* simulation, which consists of highly faithful off-line computations used by expert surgeons to predict the outcome of, and optimize, an intervention, on the basis of an anatomical model of the patient derived from that individual's preoperative image dataset. At the other end of the spectrum, the objective of *interactive* simulation is to offer a means of training surgical residents in order to improve their skill without risk to a real patient, by way of a haptic device manipulated by the user to position a virtual surgical tool, while producing a force feedback that simulates tissue resistance and a real-time graphical rendering of an anatomical model at that point in simulated time. Figure 1 illustrates some commonly used haptic devices.

Typically, the biomechanics engine used to achieve a response at near-haptic rates (some interpolation is feasible for haptic rates of 500 Hz or more), in the context of interactive simulation, is less constitutively faithful than that of predictive simulation, although much recent work is devoted to reconciling the conflicting requirements of interactivity and material faithfulness.

Irrespective of whether a medical simulator emphasizes interactivity or predictive computation, the simulation requires an anatomical model on which to carry out its synthesized therapy. For most clinical applications, such a model is not drawn with 3D CAD software, but rather extracted by image analysis from a patient dataset. As a result, the starting point for this model is one or more MR or CT volumes, which in the multi-modal case can be co-registered and resampled, which leads to a volumetric scalar or vector image, typically of several hundred voxels along each axis. For example, a 1 mm isotropic MR image of the head is usually at least $256 \times 256 \times 256$, which equates with more than 16 million voxels, which in turn precludes efficient computation directly based on raw or segmented image data. In addition, many biomechanical engines require the decomposition of a geometrically complex body into simple shapes, e.g.: elements, given that the computation itself is typically a matrix equation based on simple, well understood elemental expressions.

These requirements, computational efficiency and geometric decomposition, motivate the need for a representation of the anatomy in terms of simple shapes, such as triangles and tetrahedra. It is worth noting that in the mesh generation community, the generation of tetrahedra corresponds to unstructured mesh generation as contrasted from structured meshes which are typically comprised of hexahedra. The latter elements are not generally used in medical simulation, because this meshing approach requires a significant amount of user interaction (in contrast with tetrahedral meshing, which can be automated). The reason is that hexahedral meshes are more rigid structures and cannot be always automatically constructed for complex geometries [23]. Moreover, the subdivision of a hexahedron does not reduce to more hexahedra, which limits their applicability to interactive simulation, whereas a tetrahedron ultimately is divisible into more tetrahedra.

Finally, recent surgery simulation research emphasizes so-called meshless methods [4], which involve a system of equations derived from point-centered shape functions. Meshless methods discretize partial differential equations, including continuum mechanics expressions, through shape functions with compact support defined on a local cloud of points (or nodes), rather than on non-overlapping elements. Despite the name that implies that no mesh is involved, the latter approach requires a preliminary meshing that establishes neighboring vertices in the point cloud used in the discretization.

2 Background

2.1 Non-rigid Registration

We used the non-rigid registration method described by Clatz et al. [7] which is shown to be robust enough to be usable to clinical studies. Below, we outline the main aspects of this NRR method.

The method consists of three steps, namely, *feature points selection*, *block matching*, and *system solution*. See Fig. 2 for an illustration. During feature points selection, a sparse set of points is chosen from the pre-operative image. These points are

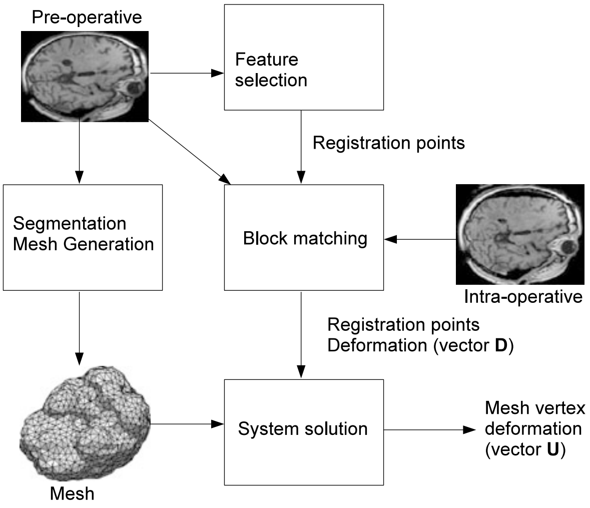

Pre-operative
Feature selection
Registration points
Segmentation Mesh Generation
Block matching
Intra-operative
Registration points Deformation (vector **D**)
System solution
Mesh vertex deformation (vector **U**)
Mesh

Fig. 2 The non-rigid registration procedure

called *registration* points. Then, the correspondence of these points into the intra-operative image is found via a block matching scheme. Specifically, for a given registration point r, a small window around it in the intra-operative image is searched; the corresponding point r' reported is the one that maximizes the correlation coefficient between r' and r.

Having computed the deformation vector **D** on the registration points (as a result of the block matching step), the deformation vector on the mesh vertices **U** (the unknowns) is calculated so that the following energy is minimized:

$$W = \underbrace{(\mathbf{HU}-\mathbf{D})^\top(\mathbf{HU}-\mathbf{D})}_{\text{Error energy}} + \underbrace{\mathbf{U}^\top\mathbf{KU}}_{\text{Mechanical energy}} \tag{1}$$

In the above equation, **K** is the $|U| \times |U|$ mechanical stiffness matrix. **H** is the linear interpolating matrix of size $|D| \times |U|$; this matrix contains the measurements of the linear shape functions on every registration point. The contributing shape functions for each registration point r_i are those defined over the mesh nodes whose forming mesh element includes r_i.

The block matching deformation d_i of a registration point r_i affects the deformation of a mesh node v_j, only if v_j is incident upon a mesh element e that contains r_j. In fact, if the minimization of the error energy (also known as matching energy) was perfect (i.e., if it vanished), then the linear interpolation (of the solution of the mesh nodes of e) on r_i would give the value d_i. As Clatz et al. show [7], this method tries to minimize this exact error energy E:

$$E = \sqrt{(\mathbf{HU}-\mathbf{D})^\top(\mathbf{HU}-\mathbf{D})} = |\mathbf{HU}-\mathbf{D}| \tag{2}$$

which is the interpolation error on the registration points $r_1, r_2, \ldots, r_{|D|}$.

Since the minimization of only the error energy is under-constrained, the mechanical energy in Eq. (1) is used to model the deformation of the brain as a physical body based on FEM. This, in turn, is used to discover and discard the outlier registration points, i.e., points whose deformation estimation from block matching contradicts the physical properties of the brain. For information about the construction of the mechanical stiffness matrix $\mathbf{K}$, see Delingette and Ayache [8].

The deformation vector $\mathbf{U}$, over which energy W is minimized, is computed through the following iterative equations:

$$\begin{aligned}\mathbf{F}_0 &= 0,\\ \left(\mathbf{K}+\mathbf{H}^\top\mathbf{H}\right)\mathbf{U}_i &= \mathbf{H}^\top\mathbf{D}+\mathbf{F}_{i-1}, \quad i=1,2,\ldots,\\ \mathbf{F}_i &= \mathbf{K}\mathbf{U}_i, \quad i=1,2,\ldots\end{aligned}$$

Clatz et al. [7] showed that the system above converges. Also, observe that $\mathbf{K}+\mathbf{H}^\top\mathbf{H}$ is the matrix responsible for the robustness of NRR; its condition number affects both the accuracy and the speed of the solution.

2.2 Image-to-Mesh Conversion

The problem of unstructured Image-To-Mesh conversion (I2M) is the following. Given an image as a collection of voxels, such that each voxel is assigned a label of a single tissue or of the background, construct a tetrahedral mesh that overlays the tissues and conforms to their boundaries. In this paper we study three I2M algorithms with respect to their suitability for real-time finite element analysis, based on the following requirements:

- The mesh offers a reasonably close representation (fidelity) of the underlying tissues. Our approach is to expose parameters that allow for a trade-off between the fidelity and the final number of elements with the goal of improving the end-to-end execution time of the FE analysis codes.
- The number of tetrahedra in the mesh is as small as possible provided the two requirements above are satisfied. This requirement is based on the cost of assembling and solving a sparse system of linear equations in the finite element method, which directly depends on the number of tetrahedra.
- Elements do not have very small angles which lead to poor conditioning of the stiffness matrix in Finite Element (FE) Analysis for biomechanics applications.

There is a large body of work on constructing guaranteed quality meshes for Computer Aided Design (CAD) models. The specificity of CAD-oriented approaches is that the meshes have to match exactly to the boundaries of the models. In contrast, the I2M problem allows for a certain distance between the mesh boundary and the image boundary, usually specified as a fidelity tolerance.

Labelle and Shewchuk [15] described an Isosurface Stuffing method for guaranteed quality tetrahedral meshing of domains defined by general surfaces. They offer a one-sided fidelity guarantee (from the mesh to the model) in terms of Hausdorff distance, and, provided the surface is sufficiently smooth, also in the other direction (from the model to the mesh). Their algorithm first constructs a body-centered cubic

(BCC) lattice that covers the model, then fills the BCC with high quality template elements, and warps the mesh vertices onto the model surface, or inserts vertices on the surface, and modifies the mesh. Using interval arithmetic, they prove that new elements have dihedral angles above a certain threshold. However, images are not smooth surfaces, and to the best of our knowledge, this technique has not been extended to mesh images. One approach could be to interpolate or approximate the boundary pixels by a smooth surface, for example using the *m-reps* segmentation technique [20], but it would be complicated by the need to control the maximum approximation (interpolation) error. On the other hand, an I2M solution can benefit from the fact that images provide more information on their structure than general surfaces. For example, the tasks of finding the local feature size [12] and all connected components can be done relatively easily on images since they already provide the finest known sampling of the space.

There are also heuristic solutions to the I2M problem, some of them developed in our group [10, 16], that fall into two categories: (1) first coarsen the boundary of the image, and then apply CAD-based algorithms to construct the final mesh, (2) construct the mesh which covers the image, and then warp some of the mesh vertices onto the image surface. The first approach tries to address the fidelity and then the quality requirements, while the second approach does it in reverse order. Unfortunately, neither of these approaches can guarantee the quality of elements in terms of dihedral angles. Both of them face the same underlying difficulty which consists in separating the steps that attempt to satisfy the quality and the fidelity requirements. As a result, the output of one step does not produce an optimal input for the other step. An approach based on filling in brick elements with quality tetrahedra was developed by Hartmann and Kruggel [14], however, it keeps an over-refined mesh near the boundaries. Another method by Dogan et al. [9] produces a mesh as a by-product of an iterative segmentation procedure, by an application of a CAD-oriented mesh generator Triangle [24] to the segmented boundaries.

Zhang et al. [28] described an algorithm to construct adaptive and quality 3D meshes from imaging data. Similar to our approach, they create an initial octree-based mesh, and then improve its quality using iterative edge contraction. Specifically, their approach removes tetrahedra with the worst ratio of the longest to shortest edge length by contracting their shortest edges; however, when it is detected that a requested ratio threshold cannot be reached the strategy is reversed to point insertion through longest edge bisection. Another approach proposed by Reid et al. [21] and Goksel et al. [13] is to iteratively deform an initial mesh by vertex movement and other operations to conform to the boundaries in the image.

In Computer Aided Surgery (CAS) and specifically in image guided neurosurgery, Magnetic Resonance Images (MRI) obtained before the procedure (pre-operative) provide extensive information which can help surgeons to plan a resection path. Careful planning is important to achieve the maximal removal of malignant tissue from a patient's brain, while incurring the minimal damage to healthy structures and regions of the brain. However, current practices of neurosurgical resection involve the opening of the scull and the dura. This results in a deformation of the brain (known as the brain shift problem) which creates discrepancies between the

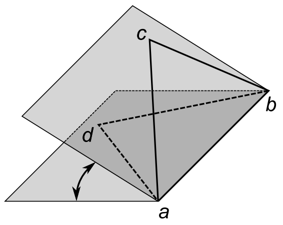


Fig. 3 The dihedral angle between two triangular faces *abc* and *abd* is the angle between two planes containing each of these faces

pre-operative imaging data and the reality during the operation. A correction is possible using non-rigid registration (NRR) of intra-operative MRI with pre-operative data.

In this paper, we target Finite Element (FE) based approaches for the non-rigid registration [7]. These methods use real-time landmark tracking across the entire image volume which makes the non-rigid registration more accurate but computationally expensive, as compared to similar methods that use surface tracking [11]. The non-rigid registration problem should be solved fast enough, so that it can be usable in clinical studies [1, 2].

Image-to-Mesh (I2M) conversion is a critical component of real-time FE-based non-rigid registration of brain images. In this paper one of the I2M evaluation criteria is the wall-clock time to construct the mesh. While in the current formulation the NRR approach makes use of a single mesh constructed before the surgery, we aim to address a general scenario, i.e., when the changes in the object geometry caused by the surgical intervention cannot be accommodated by a pre-existing mesh.

3 Evaluation of Mesh Generation Techniques

3.1 Mesh Fitness Criteria

A mesh is characterized by its *fidelity* and *quality*. Fidelity measures how well the mesh boundary resembles the surface of the biological object. Quality assesses the shape of mesh elements; the higher the minimum dihedral angle of the mesh elements is, the higher the quality. See Fig. 3 for an illustration of a dihedral angle.

It is well known that the quality of the mesh affects both the accuracy and the speed of the solver [25], because the angles of the elements influence the condition number of the stiffness matrix. In the literature, a good deal of effort has been put toward high-quality mesh generation [6, 12, 15, 18, 27].

It is not clear, however, what the impact of fidelity on the accuracy and speed of the solver is. The reason is because there is a complicated trade-off between quality and fidelity. The need for a better surface approximation always implies a deterioration of mesh quality, simply because well-shaped elements cannot fill the space formed by sharp surface creases or by surface parts of high curvature. Also, higher fidelity usually results in an increase of the number of mesh elements which in turn affects both the mesher's and the solver's speed.

In this paper, we evaluated the impact of three public mesh generators [12, 16, 26] on the accuracy and speed of NRR. The meshers were chosen carefully to cover a wide range of mesh generation approaches. The Delaunay mesh algorithm in [12] offers simultaneous meshing of the surface and the volume of the object. The algorithm in [26] is Delaunay but requires the surface of the object as input. Finally, the algorithm in [16] is an optimization-based technique which compresses an initial body-centered cubic lattice (BCC) to the surface. (See Sect. 4 for more details.) For each mesher, we conducted an extensive series of experiments controlling the fidelity of the output mesh used for the subsequent NRR [7].

3.2 Mesh Generation Libraries

In this paper, we tested the influence of three meshers on NRR, namely, *High Quality Delaunay* mesher (HQD) [12], *Tetgen* [26], and *Point Based Matching* mesher (PBM) [17]. Below, we briefly describe each of them.

HQD meshes both the surface and the volume of the object at the same time without an initial dense sampling of the object surface, as is the case in other Delaunay volume techniques [19, 22]. As a result, the number of elements of the output mesh is small.

Tetgen is a Delaunay mesh generator as well. However, it assumes that the surface of the object is already meshed and represented as a polyhedron. This polyhedron is also known as a *Piecewise Linear Complex* (PLC). Tetgen requires a PLC of the object surface as its input. We used the algorithm in [3] for the PLC generation, implemented in the *Computational Geometry Algorithms Library* (CGAL) [5].

PBM is an optimization-based approach. It starts with a triangulation of a regular grid, i.e., a body-centered cubic lattice (BCC), and then it compresses the outer nodes closer to the object surface as a result of energy minimization. In fact, the smaller the energy achieved, the better the fidelity of the output mesh. This method is able to recover the surface of multi-tissue objects. In this paper, only the single-tissue version of PBM is considered.

3.3 Evaluation Methodology

As mentioned in Sect. 2.1, registration computes the deformation on the mesh nodes, so that the error energy $E = |\mathbf{HU} - \mathbf{D}|$ is minimized. Mesh generation affects how accurately the error energy is minimized. Therefore, we assess the accuracy of registration by keeping track of this error E. There are two nested loops in the registration algorithm. The outer loop, which is run 10 times following the original paper by Clatz et al. [7], is used to discard outlier registration points. The inner loop runs the FE solution and has a fixed threshold on the convergence of the linear solver. This threshold, however, does not translate to a fixed error in the registration result due to the influence of mesh fidelity. Below we describe and report the results of two types of experiments. In the experiments of the first type we vary mesh fidelity and measure the registration error for the selected meshing algorithms. In the experiments of the second type we fix both the fidelity and the registration error, and measure the wall-clock time for the mesher and the solver.

Observe, however, that the outcome of the registration depends on the accuracy of the block matching step (vector **D**). Also, notice that the mesh does not affect the result of block matching (see Fig. 2). Since we are interested in evaluating the impact of mesh generation on registration, we wanted to make registration independent of block matching. For this reason, we synthetically deformed the pre-operative image according to the bio-mechanical properties of the brain. More specifically, we initially ran the registration procedure to register the pre-operative with the intra-operative image as shown in Fig. 2, but that time we did not focus on the behavior of the mesh. We just wanted the solution on the mesh nodes. Then, by (linearly) interpolating the solution of the mesh nodes on any point of the image, we obtained a synthetically deformed (intra-operative) image. After this initial registration, all the other registrations (aiming at evaluating mesh generation) are performed between the pre-operative and the synthetically deformed image; that is, the real intra-operative image is replaced by the deformed one. In this way, we achieve two things:

- we know the "true" deformation on any point, and therefore we know the "true" block matching result on any set of registration points, and
- we do not simulate an arbitrary deformation, but rather a realistic one, because the deformed image was obtained taking into account the elasticity properties of the brain through the stiffness matrix **K** of Eq. (1).

Since we want to measure the influence of mesh generation, only the mesh changes in every experiment. That is, for all the various meshes, the pre-operative image and the set of registration points (together with their deformation **D** of course) remain fixed.

As mentioned above, we wish to have control over the fidelity of the output mesh produced by the different meshers. In this paper, we use the *two-sided Hausdorff distance* D_H to measure fidelity.

In our case, metric D_H is defined upon two finite sets A, B as follows:

$$D_H(A,B) = \max\big\{h(A,B), h(B,A)\big\}, \quad \text{where}$$
$$h(A,B) = \max_{a\in A}\min_{b\in B} |a-b|$$

The lower the value of $D_H(A,B)$, the more similar sets A, B are. In fact, $D_H(A,B)$ is equal to 0 if and only if sets A, B are identical.

Fidelity of a mesh is measured as the 2-sided Hausdorff distance D_H of the following sets:

- set A: a densely sampled point set on the surface of the biological object, and
- set B: a densely sampled point set on the boundary facets of the mesh.

Notice that the mesh boundary point set B does not consist of only boundary mesh vertices. The reason is because otherwise, the Hausdorff distance of the meshes produced by HQD would always be 0 (or very close to 0), since this method guarantees that the boundary mesh vertices lie precisely on the object surface.

Having defined fidelity, we proceed by explaining how we control fidelity for each mesher.

For HQD, this is possible through the parameter δ (see [12] for a more detailed explanation). Low values of δ increase the sampling on the object surface which yields better fidelity. High values of δ produce meshes whose boundary crudely approximates the real surface.

For Tetgen, we had to change the fidelity of the PLC given by CGAL. We, therefore, had to adjust two parameters responsible for the PLC's fidelity. The first imposes an upper bound on the circumradius of the *Delaunay balls* and the second forces an upper bound on the distance between the circumcenter of the boundary facets and the corresponding center of their Delaunay balls. More information can be found in [3].

For PBM, control of fidelity is accomplished by adjusting the parameter λ. This parameter defines the trade-off between quality and fidelity: high values of λ make the optimization more sensitive to good fidelity, while low values do not change a lot the position of the initial (high-quality) BCC. However, we observed that λ does not offer a very flexible control over flexibility. Therefore, to get meshes of substantially different fidelity, we had to change not only λ but also the density of the initial BCC.

An important indicator of solution accuracy is the numerical conditioning of the linear system measured by the condition number. The condition number measures the extent by which a relative perturbation of the input affects the relative perturbation of the output. In the experimental evaluation below, we used Matlab's `cond(A)` function which computes the condition number as the ratio of the largest singular value of `A` to the smallest.

3.4 Results

Figure 4 presents the results obtained by various meshes produced by High Quality Delaunay (HQD), Tetgen+CGAL, and Point Based Matching (PBM) approaches. On all plots, the x-axis measures mesh fidelity in terms of the Hausdorff distance D_H between the mesh and the object surface. All distances are shown with respect to the unit voxel width, and each voxel has physical dimensions 1 mm $\times$ 1 mm $\times$ 1 mm. The condition number depicted is of the matrix $\mathbf{K} + \mathbf{H}^\top \mathbf{H}$ which is responsible for the accuracy and speed of the NRR solver (see Sect. 2.1). The registration error—as defined in Eq. (2)—obtained after the end of the registration process. Figure 5 illustrates the meshes obtained by the three meshing approaches for the best and the worst fidelity.

For HQD, we observe that the error does not fluctuate considerably. All the errors are about less than half the size of a voxel, even when the D_H distance is very large.

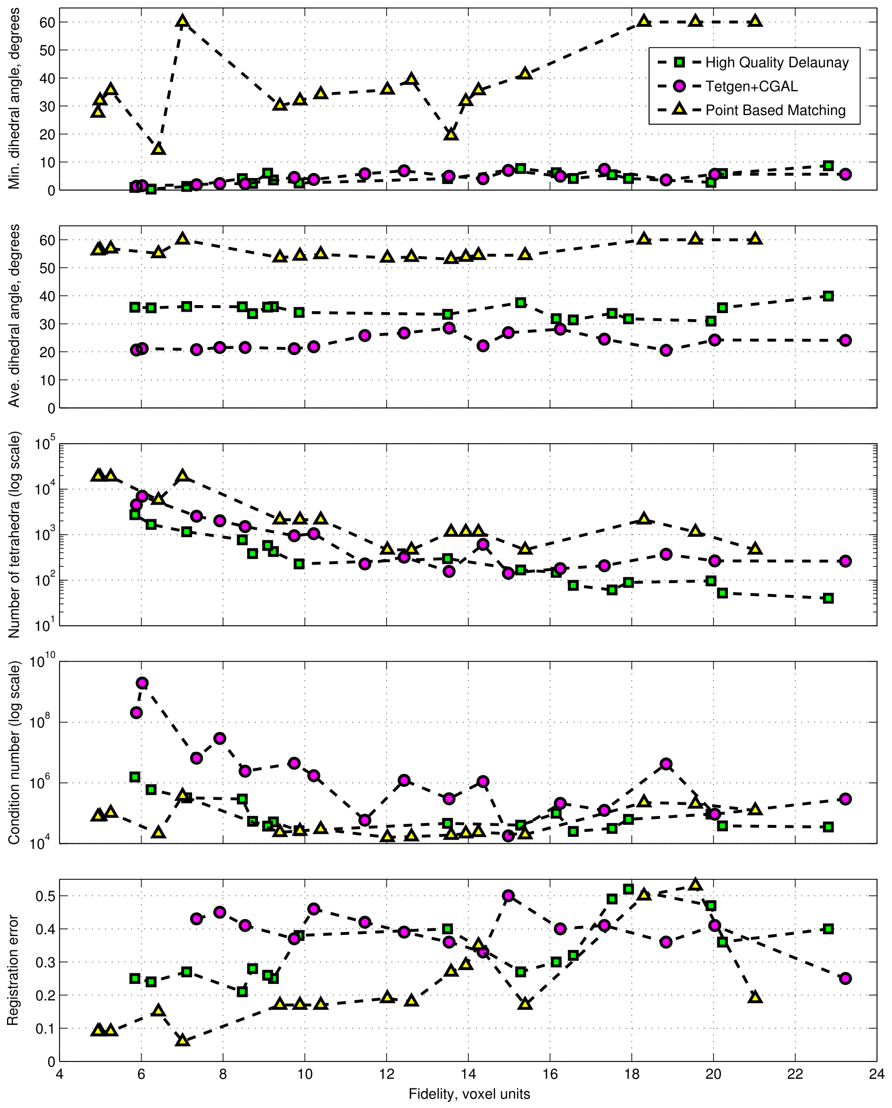


Fig. 4 Mesh properties and the resulting solution characteristics, depending on mesh fidelity measured in terms of symmetric Hausdorff distance. The x-axis measures the same fidelity values for all plots, and therefore is annotated only once

For Tetgen+CGAL, similarly, fidelity does not seem to affect the error considerably. Also, although the minimum dihedral angles are larger than those in HQD, the average minimum dihedral angles are 10 to 15 degrees less than those in HQD. This results in generally higher error than the error in HQD, but still the differences in accuracy are not very obvious. However, the much larger condition numbers affect

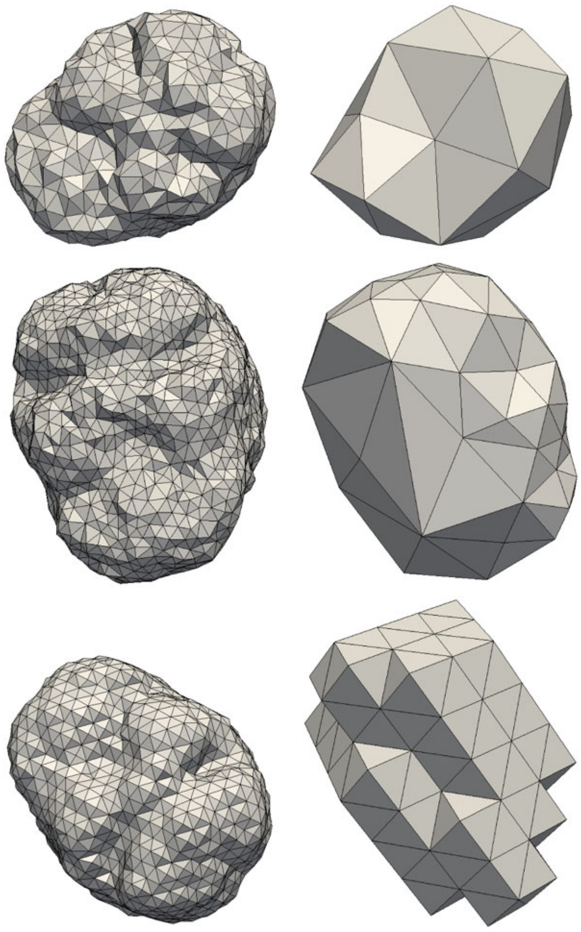


Fig. 5 The rows show the meshes obtained with the three studied approaches, from top to bottom: HQD, Tetgen+CGAL, and PBM. In each row the *left* image shows the mesh with the lowest D_H value, and the *right* image shows the mesh with the highest D_H value

the speed of the FEM solver a lot. The FEM solver we use relies on the `bicgstab` linear solver of the GMM library. Actually, for the two runs corresponding to the meshes with the two best fidelity values and with the two higher condition numbers, the solver could not even converge. For the PBM mesh we observe that the quality is very good: the minimum and the average minimum dihedral angles reach perfection. This results in much lower condition numbers and generally lower error than HQD and Tetgen. Again, we observe that fidelity does not play that important role in the accuracy of the NRR. Even meshes with very bad fidelity yield an error less than half the size of the voxel.

Also, see that for the two runs when the solver using the Tetgen meshes did not converge, the condition number is extremely large. We wanted to look into the timings of both the meshers and the solver in more depth, and to determine what the influence of fidelity on speed is. We selected 5 meshes from each method of approximately the same fidelity respectively and measured the time for meshing

Table 1 Timings (in seconds) for various meshes obtained by different methods. Both the mesh and the solver execution times are reported

D_H	HQD			Tetgen+CGAL			PBM		
	Mesher	Solver	Total	Mesher	Solver	Total	Mesher	Solver	Total
15–16.5	6.89	0.04	6.93	0.01	0.06	0.07	132.34	0.05	132.39
14–15.5	6.4	0.05	6.45	0.01	0.17	0.18	165.02	0.06	165.08
13–14.5	10.23	0.06	10.29	0.02	0.16	0.18	164.93	0.06	164.99
8.5–9.5	21.57	0.08	21.65	0.09	4.88	4.97	189.19	0.09	189.28
7–8	17.62	0.46	18.08	0.13	**45**	**45.13**	263.39	0.19	263.58

and the time for solving the registration problem. For each case, the solver has been running until the error becomes less than 0.5 (half the size of the voxel). Table 1 summarizes the results.

We observe that the meshing time of PBM is extremely large: more than 2 minutes in all cases. Actually, most of this time is spent for the initial BCC creation. On the other hand, the Tetgen+CGAL scheme is very fast: less than 2 seconds in all cases, even for the bottom mesh which consists of 2,539 elements.

As far as the solver's time is concerned, PBM yields the best meshes. Overall, however, the registration process is much slower than the other methods due to the time consuming mesh generation time. For Tetgen, the solver took much time, when the Hausdorff distance dropped below 8.5 (see bold entries). The minimum dihedral angle for this fidelity is more than $1°$, but the very low average minimum dihedral angle (the lowest among all the methods) seems to affect the condition number a lot and consequently the speed of the solver. Although the HQD meshes have elements with very small angles, the average minimum angle is much better than Tetgen (10 to 15 degrees larger). This is why when the solver ran on HQD's meshes, its execution time was less than 2 seconds in all cases, yielding a good overall execution time, even when the D_H distance drops below 8.5.

4 Discussion

In this section, we summarize our findings.

The two Delaunay meshes (i.e., HQD and Tetgen) exhibit low quality when the fidelity increases substantially (when the Hausdorff distance drops below 8 units approximately, in our case studies). This quality deterioration yields a very large condition number which affects the execution time of the solver (see Table 1). We also observe that not only the minimum but also the average minimum dihedral angle plays an important role to the solver's speed. To see it, compare the solver's speed of HQD to the solver's speed of Tetgen when the Hausdorff distance of the meshes is between 7 and 8 units. When Tetgen's mesh was used, the solver was 45 times slower. For these values of fidelity, Tetgen meshes have better minimum dihedral angles than HQD meshes, but they also have much lower average minimum dihedral angles (15 degrees smaller), which is likely to be the reason for a much worse condition number and the consequent large solver's speed.

The accuracy of the solver on the meshes produced by the two Delaunay meshers does not fluctuate significantly by the different fidelity values. That means that the need for good surface approximation does not seem to affect the accuracy of the solver. Meshes approximating very crudely the object surface yielded an error less than half the voxel size.

The main characteristic of the optimization-based mesher (i.e., PBM) is the high minimum and average dihedral angles, even in the case of very good fidelity. The reason is because relatively dense initial BCCs can easily capture the object surface without so much compression, thus preserving the good angles of the BCC triangulation. Of course, the number of elements increases significantly, which makes the mesh generation time extremely slow (see Table 1). We also observe that the solver on PBM's meshes exhibit the least error which in fact is achieved when fidelity is very good (less than 5 units approximately). This is reasonable because good fidelity does not deteriorate the quality as much as is the case of the two Delaunay meshes. Notice, however, that even when the PBM meshes have very bad fidelity, the error does not increase significantly.

Acknowledgements This research is supported in part by NSF grants CCF-1139864, CCF-1136538, CSI-1136536, and by NIH/NINDS Grant No. 5R43NS067742-02. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NSF and National Institute of Neurological Disorders and Stroke or the National Institutes of Health. A preliminary version of this work appeared and is partially reproduced (including original Figs. 1, 2, 3, 4) with kind permission from Springer Science+Business Media: Computational Biomechanics For Medicine (A. Wittek et al., eds.), An Evaluation of Tetrahedral Mesh Generation for Nonrigid Registration of Brain MRI, Part 2, 131–142, 2011, Panagiotis A. Foteinos, Yixun Liu, Andrey N. Chernikov and Nikos P. Chrisochoides, ©Springer Science+Business Media, LLC 2011.

References

1. Archip N, Clatz O, Whalen S, Kacher D, Fedorov A, Kot A, Chrisochoides N, Jolesz F, Golby A, Black P, Warfield S (2007) Non-rigid alignment of preoperative MRI, FMRI, DT-MRI, with intra-operative MRI for enhanced visualization and navigation in image-guided neurosurgery. NeuroImage 35(2):609–624
2. Bajaj C, Oden JT, Diller KR, Browne JC, Hazle J, Babuška I, Bass J, Bidaut L, Demkowicz L, Elliott A, Feng Y, Fuentes D, Kwon B, Prudhomme S, Stafford RJ, Zhang Y (2007) Using cyber-infrastructure for dynamic data driven laser treatment of cancer. In: Proceedings of the 7th international conference on computational science, part I: ICCS 2007. Springer, Berlin, pp 972–979
3. Boissonnat JD, Oudot S (2005) Provably good sampling and meshing of surfaces. Graph Models 67(5):405–451
4. Bordas S, Rabcsuk T, Zi G (2008) Three-dimensional crack initiation, propagation, branching and junction in non-linear materials by an extended meshfree method without asymptotic enrichment. Eng Fract Mech 75(5):943–960
5. CGAL: computational geometry algorithms library. http://www.cgal.org

6. Cheng SW, Dey TK, Edelsbrunner H, Facello MA, Teng SH (2000) Sliver exudation. J ACM 47(5):883–904. doi:10.1145/355483.355487
7. Clatz O, Delingette H, Talos IF, Golby AJ, Kikinis R, Jolesz FA, Ayache N, Warfield SK (2005) Robust non-rigid registration to capture brain shift from intra-operative MRI. IEEE Trans Med Imaging 24(11):1417–1427
8. Delingette H, Ayache N (2004) Soft tissue modeling for surgery simulation. In: Computational models for the human body, pp 453–550
9. Doğan G, Morin P, Nochetto RH (2008) A variational shape optimization approach for image segmentation with a Mumford–Shah functional. SIAM J Sci Comput 30:3028–3049. doi:10.1137/070692066
10. Fedorov A, Chrisochoides N (2008) Tetrahedral mesh generation for non-rigid registration of brain MRI: analysis of the requirements and evaluation of solutions. In: Proceedings of the 17th international meshing roundtable, Pittsburgh, PA. Springer, Berlin, pp 55–72
11. Ferrant M (2011) Physics-based deformable modeling of volumes and surfaces for medical image registration, segmentation and visualization. PhD thesis, Universite Catholique de Louvain. http://www.spl.harvard.edu/archive/spl-pre2007/pages/papers/ferrant/thesis/ferrant.pdf
12. Foteinos P, Chernikov A, Chrisochoides N (2010) Guaranteed quality tetrahedral Delaunay meshing for medical images. In: 7th international symposium on Voronoi diagrams in science and engineering, Laval University, Quebec City, Canada
13. Goksel O, Salcudean SE (2011) Image-based variational meshing. IEEE Trans Med Imaging 30(1):11–21. doi:10.1109/TMI.2010.2055884
14. Hartmann U, Kruggel F (1998) A fast algorithm for generating large tetrahedral 3D finite element meshes from magnetic resonance tomograms. In: Proceedings of the IEEE workshop on biomedical image analysis, WBIA. IEEE Comput Soc, Washington, pp 184–192
15. Labelle F, Shewchuk JR (2007) Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Trans Graph 26(3):57.1–57.10
16. Liu Y, Foteinos P, Chernikov A, Chrisochoides N (2010) Multi-tissue mesh generation for brain images. In: Proceedings of the 19th international meshing roundtable, Chattanooga, TN. Springer, Berlin, pp 367–384
17. Liu Y, Foteinos P, Chernikov A, Chrisochoides N (2011) Mesh deformation-based multi-tissue mesh generation for brain images. Eng Comput. In press
18. Molino NP, Bridson R, Teran J, Fedkiw R (2003) A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: Proceedings of the 12th international meshing roundtable, pp 103–114
19. Oudot S, Rineau L, Yvinec M (2005) Meshing volumes bounded by smooth surfaces. In: International meshing roundtable, San Diego, CA. Springer, Berlin, pp 203–220
20. Pizer SM, Fletcher PT, Joshi S, Thall A, Chen JZ, Fridman Y, Fritsch DS, Gash AG, Glotzer JM, Jiroutek MR, Lu C, Muller KE, Tracton G, Yushkevich P, Chaney EL (2003) Deformable m-reps for 3D medical image segmentation. Int J Comput Vis 55:85–106. doi:10.1023/A:1026313132218
21. Reid AC, Langer SA, Lua RC, Coffman VR, Haan SI, Garcia RE (2008) Image-based finite element mesh construction for material microstructures. Comput Mater Sci 43(4):989–999. doi:10.1016/j.commatsci.2008.02.016
22. Rineau L, Yvinec M (2007) Meshing 3D domains bounded by piecewise smooth surfaces. In: International meshing roundtable, pp 443–460
23. Schneiders R (1998) Quadrilateral and hexahedral meshes. In: Thompson JF, Soni BK, Weatherill NP (eds) Handbook of grid generation. CRC Press, Boca Raton
24. Shewchuk JR (1996) Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: Lin MC, Manocha D (eds) Applied computational geometry: towards geometric engineering. Lecture notes in computer science, vol 1148. Springer, Berlin, pp 203–222
25. Shewchuk JR (2002) What is a good linear element. In: Proceedings of the 11th international meshing roundtable, Sandia National Laboratories, pp 115–126

26. Si H Tetgen a quality tetrahedral mesh generator and a 3D Delaunay triangulator. http://tetgen.berlios.de/
27. Tournois J, Srinivasan R, Alliez P (2009) Perturbing slivers in 3D Delaunay meshes. In: Clark BW (ed) Proceedings of the 18th international meshing roundtable. Springer, Berlin, pp 157–173
28. Zhang Y, Bajaj C, Sohn BS (2005) 3D finite element meshing from imaging data. Comput Methods Appl Mech Eng 194(48–49):5083–5106. doi:10.1016/j.cma.2004.11.026

Surface Triangular Mesh and Volume Tetrahedral Mesh Generations for Biomolecular Modeling

Minxin Chen, Bin Tu, and Benzhuo Lu

Abstract Qualified, stable and efficient molecular surface/volume meshing appears to be necessitated by recent developments for realistic mathematical modeling and numerical simulation of biomolecules, especially in implicit solvent modeling. The chapter first describes a tool, TMSmesh, for surface meshing through tracing a molecular Gaussian surface. The method computes the surface points by solving a nonlinear equation directly, polygonizes by connecting surface points through a trace technique, and finally outputs a triangulated mesh. TMSmesh has a linear complexity with respect to the number of atoms and is shown to be capable of handling molecules consisting of more than one million atoms, which is usually difficult for the existing methods for surface generation used in molecular visualization and geometry analysis. Then, based on the surface mesh, a tool chain is built up to generate high-quality biomolecular volume tetrahedral mesh. The performances of these meshing tools are analyzed, and the surface/volume meshes are shown to be applicable to boundary element/finite element types of simulations of Poisson–Boltzmann electrostatics.

1 Introduction

Molecular surface mesh is widely used for visualization and geometry analysis in computational structural biology and structural bioinformatics. Volume mesh also emerges to be useful in finite element modeling of molecular structure, interac-

M. Chen (✉)
Center for Systems Biology, Department of Mathematics, Soochow University, Suzhou 215006, Jiangsu, China
e-mail: chenmx@gmail.com

B. Tu · B. Lu (✉)
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China
e-mail: bzlu@lsec.cc.ac.cn

B. Tu
e-mail: tubin@lsec.cc.ac.cn

tion and dynamics. Recent developments in realistic mathematical modeling and numerical simulation of biomolecular systems raise new demands for qualified, stable, and efficient surface and volume meshing, especially in implicit-solvent modeling (e.g., see a review in [30]). Main concerns for improvement on existing methods for molecular surface/volume mesh generation are efficiency, robustness, and mesh quality. Efficiency is necessary for simulations/computations requiring frequent mesh generation or requiring mesh of large systems. Robustness here means the meshing method is stable and can treat various, even arbitrary, sizes of molecular systems within computer power limitations. Mesh quality relates to mesh smoothness (avoiding sharp solid angles, etc.), uniformness (avoiding elements with very sharp angles or zero area), and topological correctness (avoiding isolated vertices, element intersection, single-element-connected edges, etc.). The mesh quality is critical for some numerical techniques, such as boundary element/finite element methods, to achieve converged and reasonable results, which makes it a more demanding task in this respect than the mesh generations only for the purposes of visualization or some structural geometry analysis.

Various definitions of molecular surface, including the van der Waals (VDW) surface, solvent accessible surface (SAS), solvent excluded surface (SES), molecular skin surface [14], minimal molecular surface [1] and Gaussian surface, etc., have been proposed to describe the shapes of molecular structure. The VDW surface is defined as the surface of the union of the spherical atomic surfaces with the VDW radius of each atom in the molecule. The SAS and SES are represented by the trajectory of the center and the interboundary of a rolling probe on the VDW surface, respectively. The molecular skin surface is the envelope of an infinite family of spheres derived from atoms by convex combination and shrinking. The minimal molecular surface is defined as a result of the surface free energy minimization. Different from these definitions, the Gaussian surface is defined as a level set of the summation of the Gaussian kernel functions as follows

$$\left\{\mathbf{x} \in \mathbb{R}^3, \phi(\mathbf{x}) = t_0\right\}, \tag{1}$$

where

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} e^{d(\|\mathbf{x}-\mathbf{c}_i\|^2/r_i^2-1)}, \tag{2}$$

$\mathbf{c}_i$ and r_i are the location and radius of atom i, the parameter d is negative and controls the decay speed of the kernel functions. When $|d|$ increased, the resulting Gaussian surface is closer to the VDW surface. In this work, the value of d and t_0 are set as -1 and 1, respectively. Compared with other definitions of molecular surface, Gaussian surface is smooth and more suitable to represent the electron density of a molecule [13]. The VDW surface, SAS, and SES can be approximated well by the Gaussian surface with proper parameter selection [2, 13]. The Gaussian surface has been widely used in many problems in computational biology, such as docking problems [33], molecular shape comparisons [18], calculating SAS areas [48] and the generalized Born models [49].

With various definitions of molecular surface that have been proposed, numerous works have been devoted to the computation of molecular surface. The representative ones are described as follows. In 1983, Connolly proposed algorithms to calculate the molecular surface and SAS analytically [9, 10]. In 1995, a popular program, GRASP, for visualizing molecular surfaces was presented [36]. In 1997, Vorobjev et al. proposed SIMS, a method of calculating a smooth invariant molecular dot surface, in which an exact method for removing self-intersecting parts and smoothing the singular regions of the SES was presented [45]. Sanner et al. presented a tool based on α shapes [15], named MSMS, for meshing the SES [39]. Ryu et al. proposed a method based on β shapes that are a generalization of α shapes [38]. More recently, Zhang et al. used a modified dual contouring method to generate mesh for biomolecular structures [52], and a later tool, GAMer, was developed for improving the mesh quality [50]. Can et al. proposed LSMS to generate the SES on grid points using level-set methods [5]. Chavent et al. presented MetaMol to visualize the molecular skin surface using ray-casting method [6], and Cheng et al. used restricted union of balls to generate mesh for molecular skin surface [8]. So far, these methods or tools usually successfully calculated different surfaces of small- or medium-sized biomolecules, but they are not suitable for large molecules with more than hundreds of thousands of atoms. Moreover, most of these methods, such as GRASP, MSMS, and LSMS were designed for molecular visualization and geometry analysis in computational structure biology or structural bioinformatics. Among those, MSMS is the most widely used one for molecular surface triangulation because of its high efficiency. However, the generated mesh is not a manifold and is composed of very irregular triangles. For some numerical modeling using, for instance, finite element/boundary element methods, the mesh quality usually needs to be improved through mesh topology checking (picking out the irregular nodes/edges/elements and rearranging the mesh), surface mesh smoothing, and so on [30]. In this chapter, we describe a recently developed robust method, named TMSmesh [7, 43] that is capable of meshing the Gaussian surface for biomolecules consisting of more than one million atoms in 30 min on a typical 2010 PC, and the mesh quality is shown to be applicable to boundary element method simulations of biomolecular electrostatics.

As the Gaussian surface is an implicit surface, the existing techniques for triangulating implicit surface can be used for the Gaussian surface. These methods are divided into two main categories: spatial partition and continuation methods. The well-known marching cubes [28] and dual contouring methods [24] are examples of the spatial partition methods. This kind of method divides the space into cells and polygonizes the implicit surface in the cell whose vertices have different signs of the implicit function. An assumption is required that the implicit function is almost linear in the cell. As shown in the following sections, TMSmesh does not require this assumption. The continuation methods [19, 20, 25] are of another category. These methods mesh the implicit surface by growing current polygonization's border through the predictor–corrector method which predicts the next surface point in the tangent direction of the current one and corrects it on the surface. The predictor–

corrector method is used in TMSmesh to generate the next corrected point on the surface from current one, and the topology connection is confirmed by checking the continuity between the corrected and current points, otherwise we restart the predictor–corrector from current point with a smaller step size, until the continuity is fulfilled. The above process is defined as the trace technique in this chapter, and it can be seen as a generalization of the predictor–corrector method. The quality of mesh triangles is well controlled in continuation methods, but techniques for avoiding overlapping, filling the gap between adjacent branches, and selecting proper initial triangles are required. In TMSmesh, no problems of overlapping, gap filling, and selecting initial seeds need to be considered, because the Gaussian surface is polygonized by connecting presampled surface points.

Once a surface mesh is obtained, we can generate a volume mesh conforming to the molecular surface. The Advancing Front technique (AFT) [27, 32, 34, 37], Octree methods [40] and Voronoi Delaunay based methods [3, 4, 16] are some of the well studied techniques in unstructured mesh generation. There are some software packages for tetrahedral mesh generation such as TetGen [42], NetGen [35], and CAMAL [11]. TetGen corresponds to a suite of techniques to generate different tetrahedral meshes from three-dimensional point sets or domains with piecewise linear boundaries. NetGen is an automatic 3D advancing-front tetrahedral mesh generator that accepts input from constructive solid geometry (CSG) or boundary representations (BRep) from the STL file format. CAMAL (the CUBIT Adaptive Meshing Algorithm Library) contains several of the CUBIT projects mesh generation algorithms. The tetrahedral mesh generation tool included in CAMAL is TetMesh-GHS3D [17], a package to automatically create tetrahedral meshes from closed triangular surface meshes, with little or no user interaction.

A tetrahedral mesh is created by two steps: first, generating a triangular mesh of the boundary of the target volume, then filling the inside of the triangular mesh with tetrahedral elements. A surface mesh may include two types of defects, geometric defects and topological defects. A self intersection is one of the common geometric defects, and a common topological defect is a gap or a hole located where the mesh needs to be closed. Due to complexity of molecular structure, the surface mesh is often of poor quality, even has defects. This makes it difficult to get high-quality tetrahedral mesh. In this chapter, we have built a tool chain to generate high-quality biomolecule mesh by combining a number of mesh generation tools. TMSmesh is first used to generate the molecular surface triangular mesh. The mesh quality is then improved through topology check and smoothing. Finally, the volume mesh is generated using TetGen.

This chapter is organized as follows. In Method section (Sect. 2), we present our method for polygonizing the Gaussian surface and a tool chain for generating tetrahedral volume mesh. Some examples and applications are presented in the Results section (Sect. 3). The final section, Conclusion (Sect. 4), gives some concluding remarks.

2 Method

2.1 Surface Triangular Mesh Generation

In this section, we describe the algorithm for meshing the Gaussian surface. Our algorithm contains two stages. The first stage is to compute the points on the surface by solving a nonlinear equation $\phi(\mathbf{x}) = t_0$. The second stage is to polygonize the Gaussian surface by connecting the generated points. In the following subsections, each stage is described in detail.

2.1.1 Computing the Points on the Gaussian Surface

From the definition of Gaussian surface, the points on the Gaussian surface are the roots of nonlinear equation $\phi(x, y, z) = t_0$, where $\phi(x, y, z)$ is defined in (2). Therefore, solving $\phi(x, y, z) = t_0$ is equivalent to computing the points on the Gaussian surface. In this method the equation is solved by the following steps.

Suppose the molecule is placed on a three-dimensional orthogonal grid consisting of $n_x \times n_y \times n_z$ cubes. For an arbitrary cube $[x_i, x_i + h] \times [y_i, y_i + h] \times [z_i, z_i + h]$, where (x_i, y_i, z_i) is the lower-left front corner, and h is the edge length of the cube. To decide whether the cube has intersection with the surface, we proposed the following lower and upper bounds of $\phi(x)$ in the cube for Gaussian surface:

$$L_i = \sum_{k=1}^{N} e^{-d} L_{k,i}^{x} L_{k,i}^{y} L_{k,i}^{z} \le \phi(x, y, z) \le \sum_{k=1}^{N} e^{-d} U_{k,i}^{x} U_{k,i}^{y} U_{k,i}^{z} = U_i, \tag{3}$$

for $(x, y, z) \in [x_i, x_i + h] \times [y_i, y_i + h] \times [z_i, z_i + h]$, where

$$U_{k,i}^{\alpha} = \begin{cases} 1, & c_{\alpha}^{k} \in [\alpha_i, \alpha_i + h] \\ \max\{e^{d(\alpha_i - c_{\alpha}^{k})^2/r_k^2}, e^{d(\alpha_i + h - c_{\alpha}^{k})^2/r_k^2}\}, & c_{\alpha}^{k} \notin [\alpha_i, \alpha_i + h] \end{cases} \tag{4}$$

$$L_{k,i}^{\alpha} = \min\left\{e^{d(\alpha_i - c_{\alpha}^{k})^2/r_k^2}, e^{d(\alpha_i + h - c_{\alpha}^{k})^2/r_k^2}\right\}, \tag{5}$$

with $\alpha \in \{x, y, z\}$ and $\mathbf{c}_k = (c_x^k, c_y^k, c_z^k)$. $U_{k,i}^{\alpha}$ and $L_{k,i}^{\alpha}$ are the upper and lower bounds along α-dimension of the kernel located at atom k in the cube, respectively. $U_{k,i}^{\alpha}$ and $L_{k,i}^{\alpha}$ take either 1 or a value of the kernel at the boundary of the cube. If $t_0 \in [L_i, U_i]$, then the Gaussian surface $\phi(x, y, z) = t_0$ may have an intersection with cube $[x_i, x_i + h] \times [y_i, y_i + h] \times [z_i, z_i + h]$, otherwise there is no surface point in the cube. Note that the upper-bound U_i and the lower-bound L_i depend on the edge length of the cube h. The bounds are sharper when h is smaller. Above estimation is easy to combine with an octree data structure to decide intersection more adaptively. Figure 1 shows a two dimensional case that uses a quadtree data structure with bounds in Eqs. (4) and (5) to find the cubes intersecting the surface. In Fig. 1, the black curves represent the surface $\phi(\mathbf{x}) = c$, and the red cubes are the

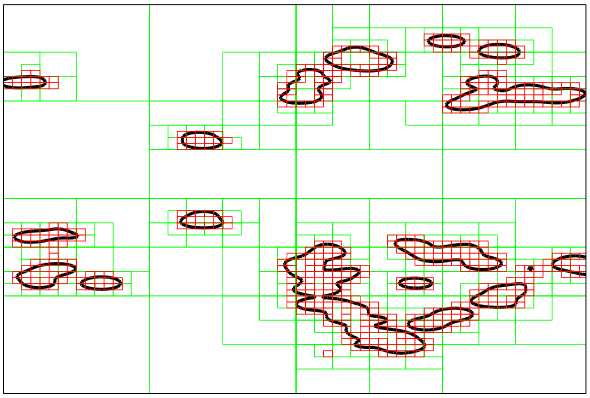


Fig. 1 A two dimensional case that using a quadtree data structure with bounds in Eqs. (4) and (5) to find the cubes intersecting the surface. The *black curves* represent the surface $\phi(\mathbf{x}) = c$, and the *red cubes* are the left cubes whose lower bounds are smaller than c and upper bonds are larger then c. Most of the *red cubes* intersect the surface

left cubes whose lower bounds are smaller than c and upper bonds are larger then c. Most of the red cubes intersect the surface. This figure shows that the smaller the cube size, the shaper the bounds are in Eqs. (4) and (5).

The above estimation technique allows the deletion of the majority of cubes, which do not intersect the surface. In each of the left cubes, some surface points are sampled through root finding. Suppose the cube $[x_{i_0}, x_{i_0} + h] \times [y_{i_0}, y_{i_0} + h] \times [z_{i_0}, z_{i_0} + h]$ is one of them, we solve the nonlinear equation $\phi_{ij}(x) \triangleq \phi(x, y_{i_0} + i\tilde{h}, z_{i_0} + j\tilde{h}) = t_0$, for each $\{i, j\}$, $i, j = 1, \ldots, [h/\tilde{h}]$. Then $\tilde{h}$ is to control the vertex density of the mesh. To find the roots, $\phi(x, y_i, z_j)$, $x \in [x_{i_0}, x_{i_0} + h]$, is approximated by the following Mth-degree polynomial

$$p_{ij}\big(2(x - x_{i_0})/h + 1\big) = \sum_{n=1}^{M} (2i + 1)a_n L_n\big(2(x - x_{i_0})/h + 1\big)/2, \tag{6}$$

where $a_n = \int_{-1}^{1} \phi[hx/2 + (x_{i_0} + h/2), y_i, z_j] L_n(x)\, dx$, $L_n(x)$ is the nth-degree Legendre polynomial, M is set as 10, and h is 4 Å in our work. Then $p_{ij}[2(x - x_{i_0})/h + 1] = t_0$ is solved using Jenkins–Traub method [22]. The real roots of $p_{ij}[2(x - x_{i_0})/h + 1] = t_0$ in $[x_{i_0}, x_{i_0} + h]$ should be checked by $|\phi(x, y_i, z_j) - t_0| < \varepsilon$ (ε is an error tolerance) and be improved by Newton iterations, if needed. This process may lose some roots of $\phi_{ij}(x) = t_0$, due to approximation of $p_{ij}[2(x - x_{i_0})/h + 1]$ to $\phi_{ij}(x)$, but they would be found through trace processes in the polygonization stage.

2.1.2 Trace Step

In this subsection, the trace step employed in polygonization stage is described in detail. The objective of the trace step is to connect two (previously identified) grid–surface intersection points. In the trace step, the next connected surface point is predicted and corrected from the initial point with an initial step size, and the connection is confirmed through checking the continuity between the two points. If the

Algorithm 1 Trace step

```
Input: Initial step size $h_1$ and initial surface point $\mathbf{p}_0$. The y-coordinates of two adjacent lines on xy-plane, $y_0$ and $y_1$. User defined small positive value $\varepsilon$ and the bound for the cosine value $\delta$ ($0< \delta <1$).
Step 1, initialize $h_2 = h_1$ and $\mathbf{p}_0 = (x_0, y_0)$.
Step 2, let $\mathbf{p}_0' = \mathbf{p}_0 + h_2(-\phi_y(\mathbf{p}_0), \phi_x(\mathbf{p}_0))$.
Step 3, use Newton iterations to find $t$, s.t.
    $\phi\big(\mathbf{p}_0' + t\big(\phi_x\big(\mathbf{p}_0'\big), \phi_y\big(\mathbf{p}_0'\big)\big)\big) = t_0.$
Let $\mathbf{p}_1 = \mathbf{p}_0' + t(\phi_x(\mathbf{p}_0), \phi_y(\mathbf{p}_0))$.
Step 4, if $|\phi_x(\mathbf{p}_1)| < \varepsilon$, $\mathbf{p}_1$ is an extreme point along $x$ direction, add it to the extreme point list.
Step 5, if $\cos((\phi_x(\mathbf{p}_1), \phi_y(\mathbf{p}_1)), (\phi_x(\mathbf{p}_0), \phi_y(\mathbf{p}_0))) < \delta$ (condition $\star_1$) or ($\phi_x(\mathbf{p}_1)\phi_x(\mathbf{p}_0) < 0$ and $\min(|\phi_x(\mathbf{p}_1)|, |\phi_x(\mathbf{p}_0)|) > \varepsilon$) (condition $\star_2$), let $h_2 = h_2/2$ and go to step 2.
Step 6, if $(\mathbf{p}_0(y) - y_0)(\mathbf{p}_1(y) - y_0) < 0$ (or $(\mathbf{p}_0(y) - y_1)(\mathbf{p}_1(y) - y_1) < 0)^a$, interpolate $\mathbf{p}_0$ and $\mathbf{p}_1$ to get the connected point $(x_1, y_0)$ (or $(x_1, y_1)$), let $\mathbf{p}_1 = (x_1, y_0)$ (or $\mathbf{p}_1 = (x_1, y_1)$) and stop.
Step 7, let $\mathbf{p}_0 = \mathbf{p}_1$ and go to step 1.
Output: The final connected surface point $(x_1, y_0)$ on $y = y_0$ (or $(x_1, y_1)$ on $y = y_1$), and the extreme point(s) if exist.
```

[a]Where $\mathbf{p}_0(y)$ and $\mathbf{p}_1(y)$ denote y-coordinates of point $\mathbf{p}_0$ and $\mathbf{p}_1$, respectively.

continuity is not fulfilled, then restart the prediction and the correction process from the initial point with a smaller step size. Because every trace step is performed either on xy or yz-planes between lines parallel to x-axis in the polygonization stage, we discuss the details of the trace step on the xy-plane between two lines parallel to x-axis as an example. Suppose there are two lines $y = y_0$, and $y = y_1$ on the xy-plane with four surface points (x_0, y_0), (x_1, y_0), (x_2, y_1), (x_3, y_1) on them and $y_1 > y_0$, $\phi_x(x_0, y_0) > 0$, Algorithm 1 connects (x_0, y_0) and (x_1, y_0) (see Fig. 2). Moreover, the extreme point (x^*, y^*) is caught during the trace step to preserve the details of the surface.

In the step 2 of Algorithm 1, $\mathbf{p}_0'$ is the predicted surface point from $\mathbf{p}_0$ along the tangent direction at $\mathbf{p}_0$ with step size h_2. Step 3 is to correct $\mathbf{p}_0'$ back to the surface along the gradient direction of $\phi(x)$ at $\mathbf{p}_0'$. Step 4 is to check whether $\mathbf{p}_1$ is an extreme point along the x-direction. In step 5, condition ($\star_1$) is used to determine if the step-size h_2 is acceptable through checking whether the angle between the normal directions at $\mathbf{p}_0$ and $\mathbf{p}_1$ is small enough, otherwise restart the prediction and correction from $\mathbf{p}_0$ with a smaller step size; δ is a user-specified bound for cosine value of the angle. If the condition is not sufficient in some cases, then other conditions can be added, such as continuity of higher order derivatives. In the case that an extreme point along the x-direction exists between $\mathbf{p}_0$ and $\mathbf{p}_1$, condition ($\star_2$) is used to detect it. In step 6, if the line segment connecting $\mathbf{p}_0$

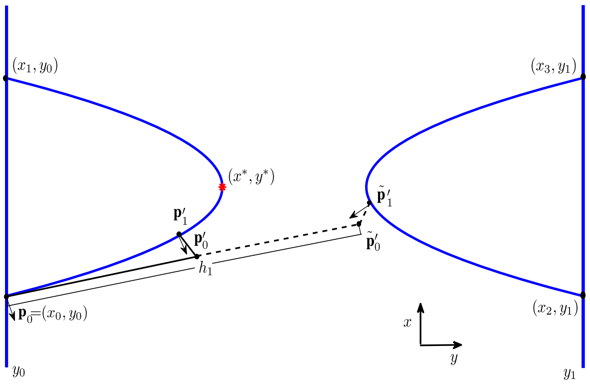


Fig. 2 Schematic picture of Algorithm 1. Curved lines indicate the surface on the xy-plane. *Arrows* on the surface denote the normal directions of the Gaussian surface on xy-plane pointing to the outside of the molecule. The $\mathbf{p}_0$ is the initial point, (x^*, y^*) is an extreme point along x-direction, and (x_1, y_0) is the final connected point on the surface obtained through the trace step from $\mathbf{p}_0$. As an illustration, $\tilde{\mathbf{p}}_0'$ and $\tilde{\mathbf{p}}_1$ on the *dashed line* that kinks near the curve on the right-hand side are the predicted and corrected points with a larger initial step h_1, which can be avoided through conditions $(\star_1)$ in Algorithm 1. While $\mathbf{p}_0'$ is the predicted point along the *tangent* direction of $\mathbf{p}_0$ with a smaller step size $h_1/2$, and $\mathbf{p}_1$ is the corrected surface point from $\mathbf{p}_0'$

and $\mathbf{p}_1$ crosses line $y = y_0$ (or $y = y_1$), then the point of intersection is the final trace point. In step 7, $\mathbf{p}_0$ is replaced by $\mathbf{p}_1$ and starts tracing the next connect surface point from step 1. This process indicates that the final connected point can be located through trace step from initial point, therefore, the position of the final point needs not be known before the trace process. For this reason, a disjointed part of the whole surface will not be missed after the polygonization stage unless no points from the disjointed part are found in the stage of surface point sampling.

2.1.3 Polygonization

This section is devoted to the polygonization step which connects the presampled points obtained through the process described in the section of computing surface points. Because solving $\phi(x, y, z) = t_0$ for different y, z values is equivalent to finding the intersection points of the surface and the different lines parallel to x-axis, polygonization of the whole surface can be achieved through connecting these points on every adjacent four lines. The problem is how to connect the surface points on the adjacent four lines. Suppose we have surface points set P consisting of four lists of points:

$$\{x_1^i, y_0, z_0\}, \quad i = 1, \ldots, n_1, \tag{7}$$

$$\{x_2^i, y_0 + \tilde{h}, z_0\}, \quad i = 1, \ldots, n_2, \tag{8}$$

Algorithm 2 Connecting the points on the four adjacent lines from the initial point $\mathbf{p}_0$ to form one polygon

```
Input: The set $P$ containing coordinates of all sampled points on the four adjacent lines. The grid space $\tilde{h}$. The yz-coordinates of lower-left line $\{y_0, z_0\}$. The initial point $\mathbf{p}_0$.

Find the connected point $\mathbf{p}_1$ on adjacent line using trace step on xy-plane from the initial point $\mathbf{p}_0$ along the tangent direction $(-sig_x\phi_y(\mathbf{p}_0), sig_x\phi_x(\mathbf{p}_0), 0)$, where $sig_x$ is the sign of $\phi_x(\mathbf{p}_0)$.
Let $idx = 1$, $i = 1$ and $P = P - \{\mathbf{p}_0\}$.
while $\mathbf{p}_i \neq \mathbf{p}_0$ do
  if $\mathbf{p}_i$ is in set $P$ then
    Let $P = P - \{\mathbf{p}_i\}$.
  end if
  if $idx = 1$ then
    if $\mathbf{p}_i(y) = y_0$ then
      Find the connected point $\mathbf{p}_{i+1}$ on adjacent line using trace step on xz-plane from $\mathbf{p}_i$ along the tangent direction $(-sig_x\phi_z(\mathbf{p}_i), 0, sig_x\phi_x(\mathbf{p}_i))$.
    else if $\mathbf{p}_i(y) = y_0 + \tilde{h}$ then
      Find the connected point $\mathbf{p}_{i+1}$ on adjacent line using trace step on xz-plane from $\mathbf{p}_i$ along the tangent direction $(sig_x\phi_z(\mathbf{p}_i), 0, -sig_x\phi_x(\mathbf{p}_i))$.
    end if
    Let $idx = 2$.
  else if $idx = 2$ then
    if $\mathbf{p}_i(3) = z_0$ then
      Find the connected point $\mathbf{p}_{i+1}$ on adjacent line using traces step on xy-plane from $\mathbf{p}_i$ along the tangent direction $(-sig_x\phi_y(\mathbf{p}_i), sig_x\phi_x(\mathbf{p}_i), 0)$.
    else if $\mathbf{p}_i(3) = z_0 + \tilde{h}$ then
      Find the connected point $\mathbf{p}_{i+1}$ on adjacent line using trace step on xy-plane from $\mathbf{p}_i$ along the tangent direction $(sig_x\phi_z(\mathbf{p}_i), -sig_x\phi_x(\mathbf{p}_i), 0)$.
    end if
    Let $idx = 1$.
  end if
  Let $i = i + 1$.
end while
Output: The polygon whose vertices are $\mathbf{p}_j$, $j = 1, \ldots, i$, and the extreme points (if they exist) along x-direction obtained during the trace steps.
```

$$\left\{x_3^i, y_0, z_0 + \tilde{h}\right\}, \quad i = 1, \ldots, n_3, \tag{9}$$

$$\left\{x_4^i, y_0 + \tilde{h}, z_0 + \tilde{h}\right\}, \quad i = 1, \ldots, n_4, \tag{10}$$

in the four adjacent lines:

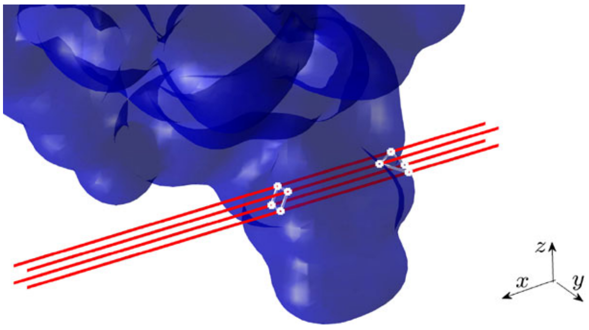


Fig. 3 Schematic picture of connecting surface points on the four adjacent *lines* parallel to x-axis to form small close loops, i.e. polygons

$$\begin{cases} y = y_0 \\ z = z_0 \end{cases}, \qquad \begin{cases} y = y_0 + \tilde{h} \\ z = z_0 \end{cases},$$
$$\begin{cases} y = y_0 \\ z = z_0 + \tilde{h} \end{cases}, \qquad \begin{cases} y = y_0 + \tilde{h} \\ z = z_0 + \tilde{h} \end{cases}. \tag{11}$$

Without loss of generality, assume $n_1 > 0$ and $(x_1^1, y_0, z_0) \triangleq \mathbf{p}_0$ is chosen to be the initial point. Through invoking the trace step described in former subsection, Algorithm 2 is to connect the points on the four adjacent lines to form small closed loops, i.e. polygons, on the surface. This is illustrated in Fig. 3.

Algorithm 2 is repeated until all points are connected, i.e., P is empty. This algorithm traces vertices of polygons in xy- and xz-planes alternately. The variable idx records the location of the last trace step and the sig_x records the direction of the first trace step. If the traced vertex is not in the same xy-plane or xz-plane as $\mathbf{p}_0$, then the next trace direction will be reversed. After Algorithm 2 is finished, $\mathbf{p}_j$, $j = 0, \ldots, i-1$, and the extreme points (if they exist) along x-direction obtained during the trace steps are connected and form a polygon on the surface. Figure 4 shows some examples of polygons with a different number of vertices obtained with Algorithm 2. The polygon can be simpler when the distances between the adjacent lines are shorter due to the smoothness of Gaussian surface. Based on the polygonized surface, the triangulation of the surface can be produced using standard polygon triangulation methods [12].

2.2 Volume Tetrahedral Mesh Generation

We have built a tool chain for high-quality biomolecule volume mesh generation by using a number of existing mesh generation tools. The tool chain has essentially three components: surface meshing, quality improving, and volume mesh generation. First, a triangulation of the Gaussian surface is generated using the program TMSmesh. The unaltered TMSmesh surface meshes for large molecules sometimes have a few geometric defects. Therefore, in the second step, we firstly use the program ISO2Mesh [21] to simplify the surface mesh, which is the process of reducing the number of faces or adding some points used in the surface while keeping

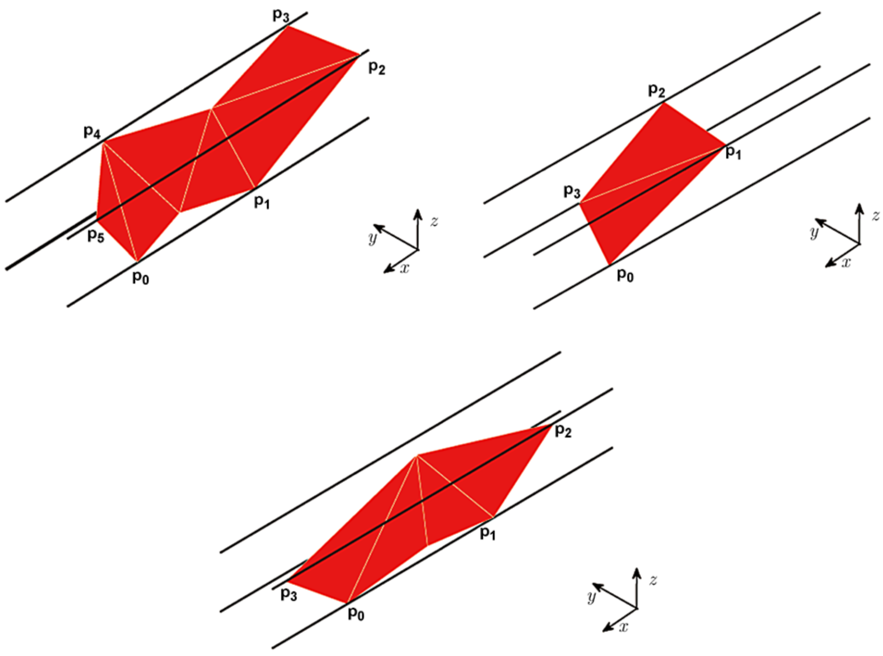


Fig. 4 Some polygons with different numbers of vertices obtained from Algorithm 2. The unlabeled vertices are extreme points obtained from the trace steps

manifold, the overall shape, volume and boundaries preserved as much as possible. Subsequently, the program TransforMesh [44] is used to remove self-intersecting faces. Finally, in the third step, the tetrahedral volume mesh is generated using the program TetGen. TetGen uses a set of switches to control the behavior of TetGen. In general, we use the switch command "-pq" to get high-quality tetrahedral mesh. The -p switch reads a piecewise linear complex (PLC) stored in file .poly or .smesh and generates a constrained Delaunay tetrahedralization (CDT) of the PLC. The -q switch performs quality mesh generation by Shewchuk's Delaunay refinement algorithm [41].

3 Results

3.1 Performance

In this section, we will present the performance of TMSmesh and the tool chain for tetrahedral mesh generation. The performance of TMSmesh is compared with those of LSMS and MSMS. LSMS is a very fast program using a level-set method to present the surface based on cubic grids. MSMS is a typical and efficient software to triangulate the SES in modeling area. A set of biomolecules with different

Table 1 Description of molecules in the PQR benchmark

Molecule (name or PDB code)	Number of atoms	Description
GLY	7	A single glycine residue
ADP	39	ADP molecule
2JM0	589	PDB code, chicken villin headpiece subdomain containing a fluorinated side chain in the core
FAS2	906	fasciculin2, a peptidic inhibitor of AChE
AChE monomer	8280	Mouse acetylcholinesterase monomer
AChE tetramer	36638	The structure of AChE tetramer, taken from Ref. [51]
30S ribosome	88431	30S ribosome, the PDB code is 1FJF
70S ribosome	165337	Obtained from 70S_ribosome3.7A_model140.pdb.gz on http://rna.ucsc.edu/rnacenter/ribosome_downloads.html
3K1Q	203135	PDB code, a backbone model of an aquareovirus virion
2X9XX	510727	A complex structure of the 70S ribosome bound to release factor 2 and a substrate analog, which has 4 split PDB entries: 2X9R, 2X9S, 2X9T, and 2X9U [23]
1K4R	1082160	PDB code, the envelope protein of the dengue virus [26]

sizes is chosen as a test benchmark. The meshing software run on the molecular PQR files (PDB + atomic charges and radii information). For tests of TMSmesh and the tool chain, we prepare a PQR benchmark (see Table 1) that can be found and is downloadable at our web page http://lsec.cc.ac.cn/~lubz/Meshing.html. It is worth making a note here about the vertex density used in TMSmesh and LSMS for comparison with MSMS surface density. For TMSmesh, grid spaces 1.0 and 0.7 Å are chosen to approximate the molecular surface vertex densities 1/Å^2 and 2/Å^2, respectively. For LSMS, the current implementation works only on the following grid sizes: 16^3, 32^3, 64^3, 128^3, 256^3, and 512^3 (requiring a 4 GB memory machine). Therefore, for each molecule, a proper grid size in LSMS is chosen to achieve the approximate density of 1/Å^2 or 2/Å^2, according to the maximum molecular length in xyz directions.

Table 2 shows the CPU time and memory use for these methods with 1 and 2 vertex/Å^2 mesh densities. All computations run on Dell Precision T7500 with Intel(R) Xeon(R) CPU 3.3 GHz and 48 GB memory under 64 bit Linux system. As shown in Table 2, TMSmesh costs less memory than LSMS and MSMS but much more CPU time for small- or medium-sized molecules. The main cost of TMSmesh is in the polygonization stage that connects the presampled surface points on parallel lines through invoking the trace steps intensively. During each trace step, prediction and correction need to be performed several times with a small step size about 0.1 to 0.2 Å, i.e., there needs 5–10 prediction–correction steps within 1 Å distance on the surface to ensure the continuity of curves connecting vertices. However, LSMS directly searches and approximates the molecular surface based on cubic grid points using the level-set method. MSMS analytically computes molecular sur-

Table 2 CPU time and memory use for molecular surface generation by TMSmesh, LSMS and MSMS[a]

Molecule	Number of atoms	CPU time (s)			Memory use (MB)		
		TMSmesh	LSMS[b]	MSMS	TMSmesh	LSMS	MSMS
FAS2	906	1.2	0.05	0.1	2	10	2
		1.8	0.1	0.1	4	19	2
AChE monomer	8280	12	0.1	0.6	11	21	21
		19	0.4	0.8	19	33	21
AChE tetramer	36638	40	0.5	5.9	36	346	75
		62	3.6	7.1	54	257	79
30S ribosome	88431	92	3.6	16.2	64	260	198
		151	28.1	19.1	100	2016	212
70S ribosome	165337	180	3.8	46.2	127	262	469
		283	28.6	Fail	185	2100	–
3K1Q	203135	226	4.0	51.5	131	262	383
		359	28.9	55.1	192	2100	410
2X9XX	510727	577	30.5	Fail	271	2100	–
		910	Fail	Fail	410	–	–
1K4R	1082160	1260	30.5	Fail	630	2168	–
		2080	Fail	Fail	890	–	–

[a]The data in the first and second row for each molecule are corresponding to density 1 vertex/Å^2 and 2 vertex/Å^2, respectively

[b]The fail cases in this column require grid size 1024^3 or larger, which is not supported by LSMS

face by first generating the so-called reduced surface that is obtained directly from atomic geometry information. Both LSMS and MSMS avoid the time-consuming step, polygonization. This makes LSMS and MSMS cost less CPU time than that of TMSmesh. Nevertheless, either the surface topology or the smoothness may not be guaranteed in LSMS and MSMS. TMSmesh is expected to be speeded up using an adaptive box structure, parallel computing, and more sophisticated polygonization algorithm. For LSMS, the cost is proportional to L^3, where L is the number of grids in one dimension. Therefore, the memory requirement and the CPU time increase dramatically when L becomes large. For MSMS, the computational complexity is $O(N \log(N))$, where N is the number of atoms, but the singularity of the molecular surface may cause numerical instability and produce incorrect results. In TMSmesh, the number of cubes is proportional to the number of atoms, since the edge length of cube is fixed to be 4 Å in this work. In addition, the calculations are done locally, and no global information is needed during the process of estimating the bounds of $\phi(x)$ in each cube, computing surface points, and tracing of the left cubes intersecting the surface. The reason is that calculating the values of $\phi(x)$ and its gradients only need to sum Gaussian kernels for near atoms, as the Gaussian

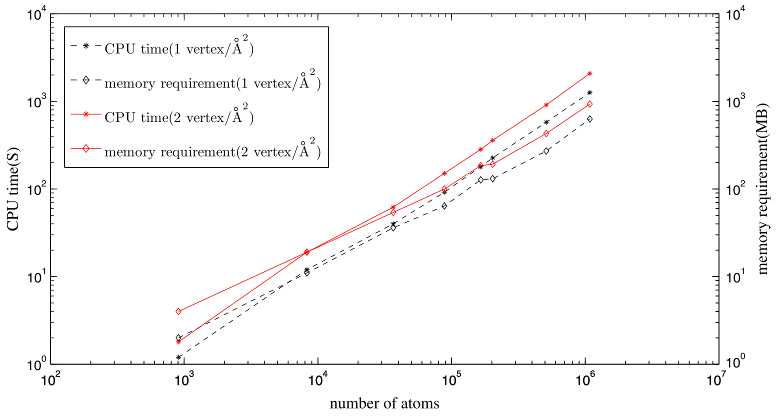


Fig. 5 Computational performance of TMSmesh

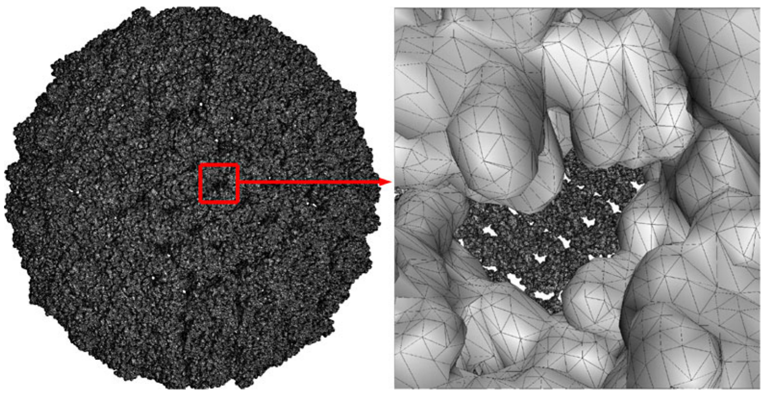


Fig. 6 A Surface triangular mesh of the envelope protein of the dengue virus (PDB code 1K4R [26]). The *left-hand side* is the whole surface mesh, and the *right-hand side* is a close view of a selected part with a gap on the surface (surrounded by the *box*). Because the structure is a shell, the inner surface of the other side of the shell is also shown through the gap

kernel $e^{d(\|\mathbf{x}-\mathbf{c}_i\|^2/r_i-1)}$ decreases to 0 faster when $\|\mathbf{x}-\mathbf{c}_i\|$ is larger. As shown in Table 2 and Fig. 5, the complexity of TMSmesh is $O(N)$. Compared to LSMS and MSMS, TMSmesh produces triangulations of a smooth surface, and it can be successfully applied to biomolecule consisting of more than one million atoms, such as dengue virus as shown in Fig. 6. Because the virus structure is among the largest ones in the Protein Data Bank (PDB), together with consideration of good algorithm stability, TMSmesh can be expected to be capable of handling the arbitrary size of molecules available in PDB.

Because MSMS is a widely used tool for surface meshing in molecular modeling, we compare the qualities, in particular uniformness, of triangles produced by TMSmesh and MSMS. The distributions of angles of each triangle are used to

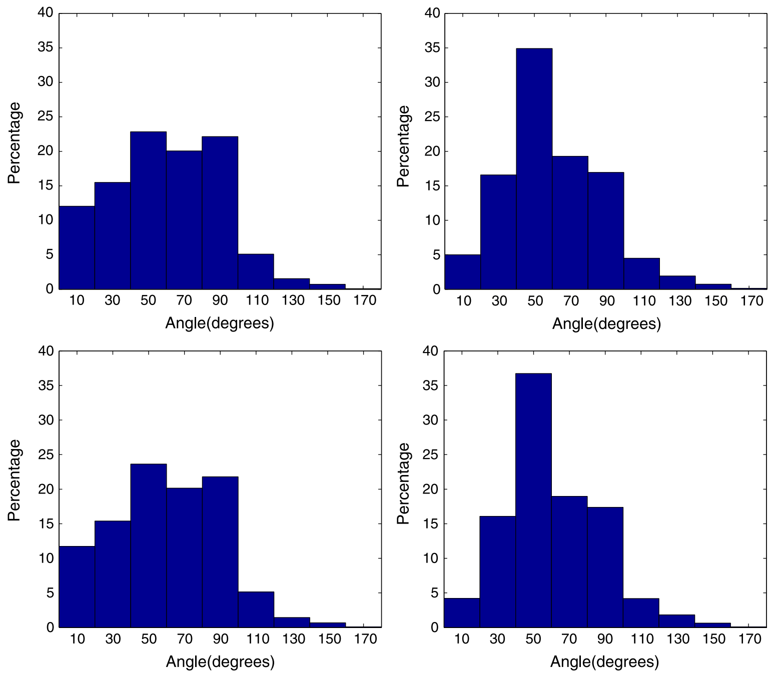


Fig. 7 Distributions of angles of each triangle produced by MSMS (*left column*) and TMSmesh (*right column*). The *first row* is for AChE tetramer meshes with densities 2 vertex/Å^2. The *second row* is for FAS2 meshes with densities 2 vertex/Å^2

describe the uniformness of meshes. The angle distributions of meshes for a large molecule, AChE tetramer, and a relatively small one, FAS2, are shown in Fig. 7. TMSmesh and MSMS meshes with density 2 vertex/Å^2 are compared. It is shown that at 2 vertex/Å^2, the angles of TMSmesh meshes are clustered around 50 degrees. Comparatively, at a low density of 2 vertex/Å^2 the angles of MSMS meshes distribute more evenly in [0, 180]. In addition, successful applications (see the following subsection) of all of our meshes to boundary element simulations also indicate improvement in mesh quality relative to MSMS mesh in the sense of right topology (e.g., without single-element-connected edges, isolated points), uniformness, and smoothness.

It is worth making a note of the molecular cavity as explored by many other surface meshing tools. TMSmesh does not differentiate the outer surfaces and interior surfaces of cavities in the meshing process. The cavities can be located through the connectivities of the triangle elements, because an internal cavity is a disjointed component of Gaussian surface (Eq. (1)) and its normal directions $\nabla\phi(x)$ are inward. The same method of finding internal cavities is used in GRASP [36].

The volume tetrahedral mesh is generated by TetGen, whose quality closely relies on the TMSmesh mesh quality and the sequential simplification/smoothing treatment of our tool chain. Figure 8 shows an example of the unstructured tetrahedral volume mesh and triangulated surface mesh of AChE tetramer that has 36638 atoms

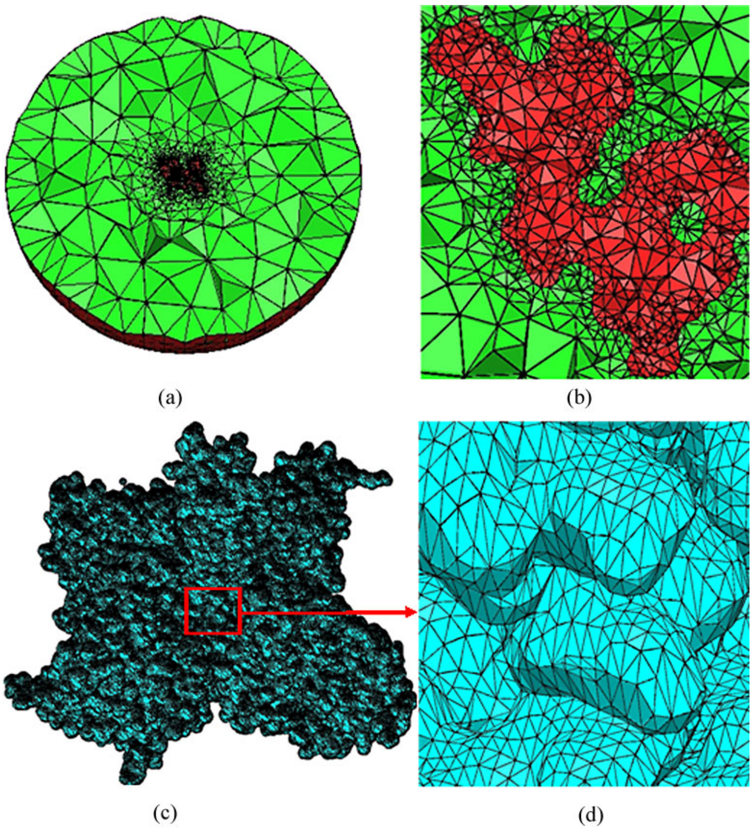


Fig. 8 An example of mesh generation for AChE tetramer. (**a**) Cross section of the whole tetrahedral volume mesh. (**b**) A close-up view of the fine mesh around the molecule, whose body is colored by *red*. (**c**) The triangular boundary mesh conforming to the molecular surface. (**d**) A close-up view of the molecular surface mesh

(see in Table 1). The molecular surface mesh is smoothed from the original surface mesh created by TMSmesh. The figures are mainly done by TetGen.

Mesh quality is an important factor influencing the convergence and stability of finite element solvers. We choose aspect ratio to measure the quality of tetrahedral meshes. The aspect ratio is defined as the ratio of the longest edge length to the smallest side height in a tetrahedron. The aspect ratio distributions of tetrahedral meshes for a large molecule, AChE tetramer, and a relatively small one, FAS2, are shown in Fig. 9. The tetrahedral mesh of AChE tetramer has 940588 vertices and 5948823 simplices, and that of FAS2 has 45098 vertices and 280786 simplices. Fig. 9 displays the aspect ratio distributions for the AChE tetramer and FAS2 meshes, which shows that most tetrahedrons have aspect ratios between 1 to 4.

As the CPU time observed in our test cases, the volume meshing takes about double of that spent on surface meshing stage using TMSmesh for most molecules.

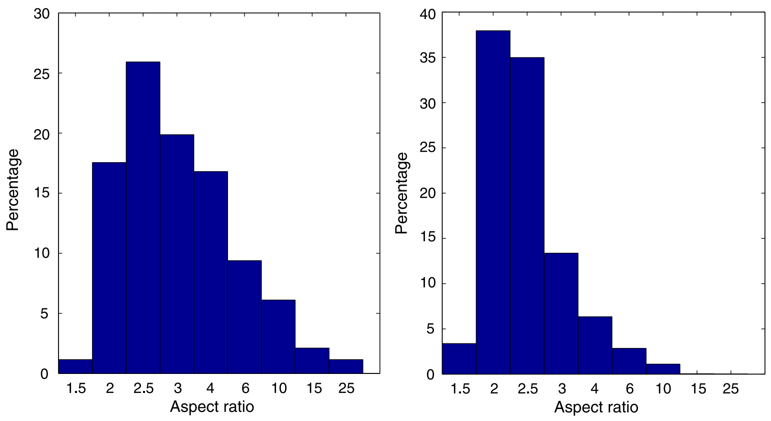


Fig. 9 Distributions of the aspect ratio for an AChE tetramer mesh (*left*) and a FAS2 mesh (*right*)

3.2 Applications

Similar to other surface generation software, such as MSMS, the surface mesh generated by TMSmesh preserves molecular surface features and thus can be applied to molecular visualization and analysis of surface area, topology, and volume in computational structure biology and structural bioinformatics. Furthermore, the goal of this work is to extend applications to some advanced mathematical modeling of biomolecules, which places demands upon the quality and the rigorous topology of the surface and volume meshes.

In this part, we test the meshes in computation of the Poisson–Boltzmann electrostatics. It is known that the PB electrostatic energy is sensitive to molecular surface definition and meshing method. Since the MSMS meshes based on SES have already been used in many previous works for small and medium sized molecules and have demonstrated to generate reasonable results, we first quantitatively compare the properties of surface meshes generated by TMSmesh and MSMS. A detailed comparison of the surface areas and molecular volumes computed from the two types of meshes is shown in Fig. 10 for three small molecules, GLY, ADP, and 2JM0 (see Table 1) using different mesh densities. In our previous work [7, 43], we have already shown that the mesh generated by TMSmesh can be successfully applied to boundary element method calculations (for example, see [31]) with better convergence performance and lead to reasonable results.

The volume tetrahedral mesh generated from the TMSmesh surface mesh by the tool chain described in this chapter can also show good performance in the usage of finite element method. Figure 11 shows the FEM results of Poisson–Boltzmann electrostatic potential of AChE tetramer. A very smooth numerical solution is obtained over the whole domain.

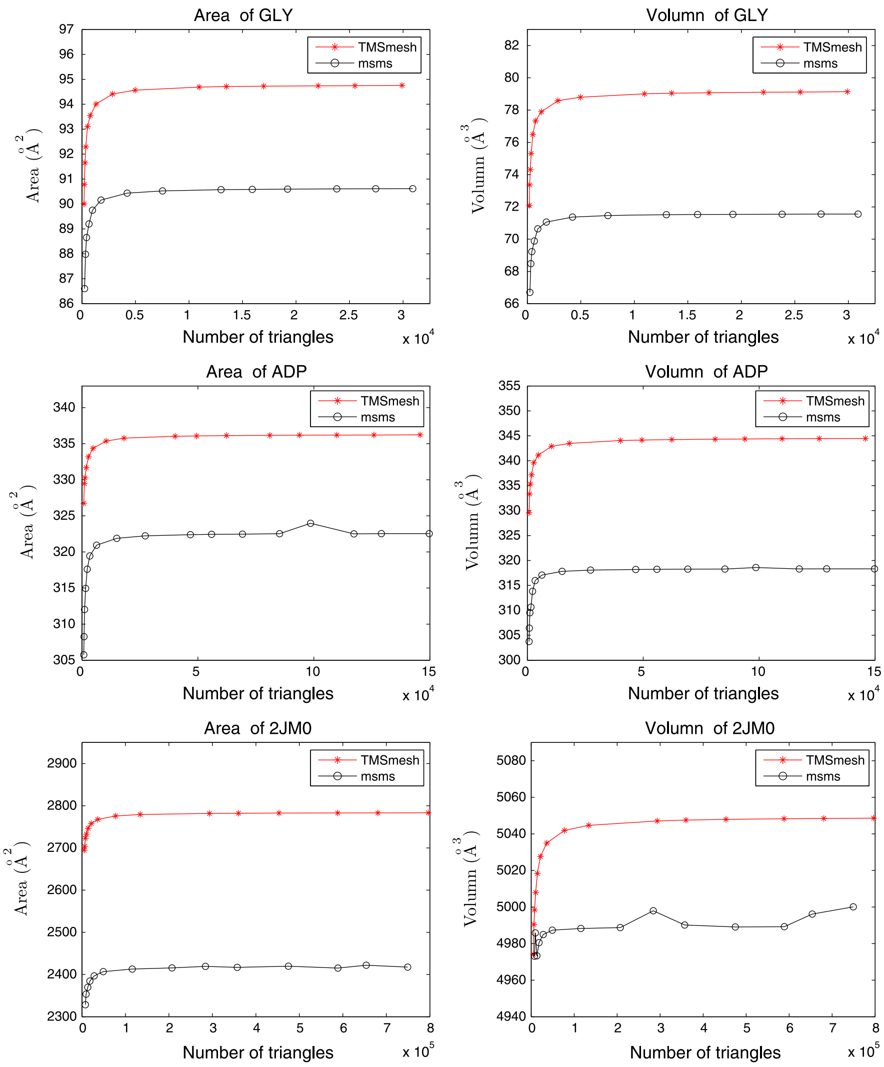


Fig. 10 Area (*first column*) and volume (*second column*) for GLY (*top row*), ADP (*middle row*), and 2JM0 (*bottom row*)

4 Conclusion

We have described a method for molecular surface meshing by a tracing surface technique and a tool chain for generating high-quality tetrahedral mesh based on the surface mesh. The implemented software TMSmesh is shown as a robust tool for meshing molecular Gaussian surfaces in the sense that: (1) It can stably handle arbitrary sizes of molecules available in PDB on a typical desktop or laptop machine, even for the not “good” molecular structures (such as ones with strong atomic

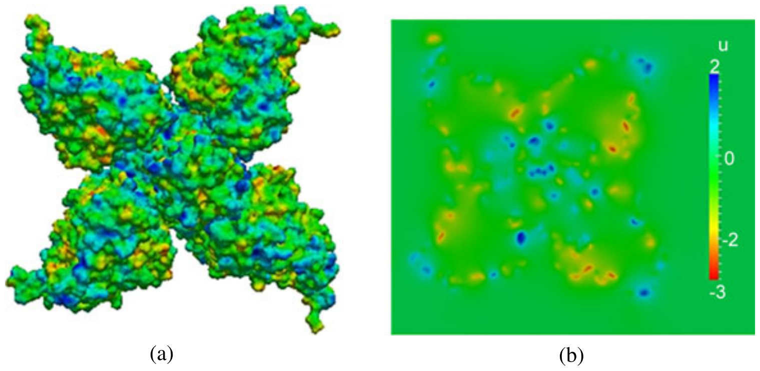


Fig. 11 Electrostatic potential of AChE tetramer. (**a**) Surface electrostatic potential from the Poisson–Boltzmann solution. The color scale is from -3 (*red*) to 2 (*blue*) kcal/mol·e. (**b**) A cross section view of the electrostatic potential (note: the potential shown inside the molecule is only corresponding to the reaction field part)

clash) and (2) the generated mesh has good quality (smoothness, uniformness, and topological correctness). The mesh converges to the smooth Gaussian surface when the mesh resolution increased and from which the calculations of surface area and molecular volume show good convergence performance and reasonable results. The tool chain described in this chapter is an effective way to get high-quality volume tetrahedral mesh. In addition to usual applications of molecular visualization and geometry analysis, the meshes are also shown to be applicable for numerical simulations with boundary element/finite element methods.

In order to simulate more complicated and wider ranges of biophysical processes using a variety of numerical techniques and modeling approaches, the current meshing methods need further improvements. First, efficiency seems to be the current bottleneck in some possible applications where the mesh needs to be either generated for large systems or generated frequently, such as in multiple-conformational analysis, BEM or FEM-based implicit solvent MD simulations [29, 46] (whereas in some finite difference-based MD simulations [47], surface meshing is not required), or elastic modeling of conformational changes. Second, mesh quality needs to be further improved, especially for the surface mesh as its quality also influences the generation and quality of volume mesh.

Finally, it is worth a mention regarding PB calculations. It is hard to conclude so far which surface specification is the best for biophysical studies due to being complicated by some other factors (like the atomic radii) in the setup of a PB calculation that can also affect the final results. Likewise, the Gaussian surface model and the meshing approach adopted in this work for PB electrostatic calculations will need further systematic studies and comparisons with experiments or other computational methods.

Acknowledgements M.X. Chen was supported in part by the China NSF (NSFC11001062) and Collegiate NSF of Jiangsu Province (No. 11KJB110010). B. Tu and B.Z. Lu was supported by the Chinese Academy of Sciences, the State Key Laboratory of Scientific/Engineering

Computing, National Center for Mathematics and Interdisciplinary Sciences, and the China NSF (NSFC10971218).

References

1. Bates PW, Wei GW, Zhao S (2008) Minimal molecular surfaces and their applications. J Comput Chem 29(3):380–391
2. Blinn JF (1982) A generalization of algebraic surface drawing. ACM Trans Graph 1(3):235–256
3. Borouchaki H, George PL (1996) Optimal Delaunay point insertion. Int J Numer Methods Eng 39(2):3407–3437
4. Borouchaki H, Laug P, George PL (2000) Parametric surface meshing using a combined advancing-front generalized Delaunay approach. Int J Numer Methods Eng 49:233–259
5. Can T, Chen CI, Wang YF (2006) Efficient molecular surface generation using level-set methods. J Mol Graph Model 25(4):442–454
6. Chavent M, Levy B, Maigret B (2008) Metamol: high-quality visualization of molecular skin surface. J Mol Graph Model 27:209–216
7. Chen MX, Lu BZ (2011) TMSmesh: a robust method for molecular surface mesh generation using a trace technique. J Chem Theory Comput 7(1):203–212
8. Cheng HL, Shi X (2009) Quality mesh generation for molecular skin surfaces using restricted union of balls. Comput Geom 42(3):196–206
9. Connolly ML (1983) Analytical molecular surface calculation. J Appl Crystallogr 16(5):548–558
10. Connolly ML (1983) Solvent-accessible surfaces of proteins and nucleic acids. Science 221(4612):709–713
11. CUBIT: the adaptive meshing algorithm library. Sandia National Laboratories. http://cubit.sandia.gov/
12. de Berg M, van Kreveld M, Overmars M, Schwarzkopf O (2000) Computational geometry: algorithms and applications, 2nd edn. Springer, Berlin
13. Duncan BS, Olson AJ (1993) Shape analysis of molecular surfaces. Biopolymers 33(2):231–238
14. Edelsbrunner H (1999) Deformable smooth surface design. Discrete Comput Geom 21(1):87–115
15. Edelsbrunner H, Mucke EP (1994) Three-dimensional alpha shapes. ACM Trans Graph 13(1):43–72
16. Frey PJ, Borouchaki H, George P (1998) 3D Delaunay mesh generation coupled with an advancing-front approach. Comput Methods Appl Math 157(1–2): 115–131
17. George PL, Hecht F, Saltel E TetMesh-GHS3D V3.1, the fast, reliable, high quality tetrahedral mesh generator and optimiser. White paper. http://www-roc.inria.fr/gamma/gamma/ghs3d/ghs.php
18. Grant JA, Gallardo MA, Pickup BT (1996) A fast method of molecular shape comparison: a simple application of a Gaussian description of molecular shape. J Comput Chem 17(14):1653–1666
19. Hartmann E (1998) A marching method for the triangulation of surfaces. Vis Comput 3:95–108
20. Hilton A, Stoddart AJ, Illingworth J, Windeatt T (1996) Marching triangles: range image fusion for complex object modelling. In: IEEE international conference on image processing, vol 2, pp 381–384
21. ISO2Mesh: a free 3D surface and volumetric mesh generator for MATLAB/Octave. http://iso2mesh.sourceforge.net/cgi-bin/index.cgi/
22. Jenkins MA, Traub JF (1970) A three-stage variables-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. Numer Math 14(3):253–263

23. Jin H, Kelley AC, Loakes D, Ramakrishnan V (2010) Structure of the 70s ribosome bound to release factor 2 and a substrate analog provides insights into catalysis of peptide release. Proc Natl Acad Sci USA 19:8593–8598
24. Ju T, Losasso F, Schaefer S, Warren J (2002) Dual contouring of hermite data. ACM Trans Graph 21(3):339–346.
25. Karkanis T, Stewart J (2001) Curvature dependent triangulation of implicit surfaces. IEEE Comput Graph Appl 2:60–69
26. Kuhn RJ, Zhang W, Rossmann MG, Sergei JC, Pletnev V, Lenches E, Jones CT, Mukhopadhyay S, Strauss EG, Chipman PR, Baker TS, Strauss JH (2002) Structure of dengue virus: implications for flavivirus organization, maturation, and fusion. Cell 5:715–725
27. Lohner R, Parikh P (1998) Generation of three-dimensional unstructured grids by the advancing-front method. Int J Numer Methods Fluids 8:1135–1149
28. Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3D surface construction algorithm. Comput Graph 21(4):163–169
29. Lu BZ, Wang CX, Chen WZ, Wan SZ, Shi YY (2000) A stochastic dynamics simulation study associated with hydration force and friction memory effect. J Phys Chem B 104(29):6877–6883
30. Lu BZ, Zhou YC, Holst MJ, McCammon JA (2008) Recent progress in numerical methods for the Poisson–Boltzmann equation in biophysical applications. Commun Comput Phys 3(5):973–1009
31. Lu BZ, Cheng XL, Huang JF, McCammon JA (2010) AFMPB: an adaptive fast multipole Poisson–Boltzmann solver for calculating electrostatics in biomolecular systems. Comput Phys Commun 6:1150–1160
32. Marcum DL, Weatherill NP (1995) Unstructured grid generation using iterative point insertion and local reconnection. AIAA J 33(9):1619–1625
33. McGann MR, Almond HR, Nicholls A, Grant AJ, Brown FK (2003) Gaussian docking functions. Biopolymers 68(1):76–90
34. Moller P, Hansbo P (1995) On advancing front mesh generation in three dimensions. Int J Numer Methods Eng 38:3551–3569
35. NetGen: an automatic 3D tetrahedral mesh generator. http://www.netgen.hr/eng
36. Nicholls A, Bharadwaj R, Honig B (1995) GRASP: graphical representation and analysis of surface properties. Biophys J 64:166–167
37. Rassineux A (1998) Generation and optimization of tetrahedral meshes by advancing front technique. Int J Numer Methods Eng 41:647–651
38. Ryu J, Park R, Kim DS (2007) Molecular surfaces on proteins via beta shapes. Comput Aided Des 39:1042–1057
39. Sanner MF, Olson AJ, Spehner JC (1996) Reduced surface: an efficient way to compute molecular surfaces. Biopolymers 38:305–320
40. Shephard MS, Georges MK (1991) Automatic three-dimensional mesh generation technique by the finite element octree technique. Int J Numer Methods Eng 32:709–749
41. Shewchuk JR (1998) Tetrahedral mesh generation by Delaunay refinement. In: Proceedings of the fourteenth annual symposium on computational geometry, Minneapolis, MN, pp 86–95
42. TetGen: a quality tetrahedral mesh generator and a 3D Delaunay triangulator. http://tetgen.berlios.de/
43. TMSmesh: a robust method for molecular surface mesh generation using a trace technique. http://lsec.cc.ac.cn/~lubz/Meshing.html
44. TransforMesh: a self-intersection removal package for triangular meshes. http://mvviewer.gforge.inria.fr/
45. Vorobjev YN, Hermans J (1997) SIMS: computation of a smooth invariant molecular surface. Biophys J 73(2):722–732
46. Wang CX, Wan SZ, Xiang ZX, Shi YY (1997) Incorporating hydration force determined by boundary element method into stochastic dynamics. J Phys Chem B 101(2):230–235
47. Wang J, Tan CH, Tan YH, Lu Q, Luo R (2008) Poisson–Boltzmann solvents in molecular dynamics simulations. Commun Comput Phys 3(5):1010–1031

48. Weiser J, Shenkin PS, Still WC (1999) Optimization of Gaussian surface calculations and extension to solvent-accessible surface areas. J Comput Chem 20(7):688–703
49. Yun Z, Jacobson MP, Friesner RA (2005) What role do surfaces play in GB models? A new-generation of surface-generalized born model based on a novel Gaussian surface for biomolecules. J Comput Chem 27(1):72–89
50. Yu ZY, Holst MJ, McCammon JA (2008) High-fidelity geometric modeling for biomedical applications. Finite Elem Anal Des 44(11):715–723
51. Zhang D, McCammon JA (2005) The association of tetrameric acetylcholinesterase with ColQ tail: a block normal mode analysis. PLoS Comput Biol 1(6):484–491
52. Zhang YJ, Xu GL, Bajaj R (2006) Quality meshing of implicit solvation models of biomolecular structures. Comput Aided Geom Des 23(6):510–530

A Combined Level Set/Mesh Warping Algorithm for Tracking Brain and Cerebrospinal Fluid Evolution in Hydrocephalic Patients

Jeonghyung Park, Suzanne M. Shontz, and Corina S. Drapaca

Abstract Hydrocephalus is a neurological disease which occurs when normal cerebrospinal fluid (CSF) circulation is impeded within the cranial cavity. As a result, the brain ventricles enlarge, and the tissue compresses, causing physical and mental problems. Treatment has been mainly through CSF flow diversion by surgically implanting a CSF shunt in the brain ventricles or by performing an endoscopic third ventriculostomy (ETV). However, the patient response to either treatment continues to be poor. Therefore, there is an urgent need to design better therapy protocols for hydrocephalus. An important step in this direction is the development of predictive computational models of the mechanics of hydrocephalic brains. In this paper, we propose a combined level set/mesh warping algorithm to track the evolution of the ventricles in the hydrocephalic brain. Our combined level set/mesh warping method is successfully used to track the evolution of the brain ventricles in two hydrocephalic patients.

1 Introduction

Hydrocephalus (also called water on the brain) is a serious neurological disorder which occurs when normal cerebrospinal fluid (CSF) circulation is impeded within the cranial cavity. If hydrocephalus develops in infancy, the intracranial pressure is raised, and, as the CSF accumulates in the ventricles, the brain tissue compresses, and both the ventricles and the skull expand. The most common cause of infantile hydrocephalus in the U.S. is hemorrhage in the neonatal period, particularly in premature infants [13]. With approximately four million births occurring annually in the U.S., it is estimated that about 20%–74% of the approximately 50,000 very-

J. Park (✉) · S.M. Shontz · C.S. Drapaca
The Pennsylvania State University, University Park, PA 16802, USA
e-mail: jxp975@cse.psu.edu

S.M. Shontz
e-mail: shontz@cse.psu.edu

C.S. Drapaca
e-mail: csd12@psu.edu

low-birth-weight infants born yearly will develop post-hemorrhagic hydrocephalus. On the other hand, in the Sub-Saharan Africa which has one of the world's greatest disease burdens of bacterial meningitis, post-infectious hydrocephalus is the most common form of infantile hydrocephalus with more than 100,000 cases arising each year.

The treatment of hydrocephalus is based on CSF flow diversion. The dilation of the ventricles can be reversed by either CSF shunt implantation or by performing an endoscopic third ventriculostomy (ETV) surgery, resulting in a relief from the symptoms of hydrocephalus. Despite the technical advances in shunt technology and endoscopy, the two treatment options display no statistically significant difference in the efficacy for treating hydrocephalus [84]. Endoscopic third ventriculostomy works well only in appropriately selected clinical cases of hydrocephalus [11], whereas shunt failure occurs in over 60% of patients [29]. Considering that many shunt recipients are children, and that shunts are lifelong commitments, these statistics underscore the importance of improving therapy. Furthermore, the postoperative persistence of the ventricular dilation constitutes a diagnostic limit for verifying the adequate functioning of ventriculostomy procedures in comparison with the treatment based on the placement of CSF shunt devices [13]. Therefore, there is an earnest need to design better therapy protocols for hydrocephalus.

An important step in this direction is the development of predictive mathematical and computational models of the mechanics of hydrocephalic brains. Many mathematical models have been proposed to explore the pathophysiology of hydrocephalus. The Monro–Kellie hypothesis [34, 49] simplifies the dynamics of the cranium to an underlying competition for space between CSF, blood, and brain parenchyma. This idea leads to numerous pressure-volume models where the CSF is contained within one compartment surrounded by compliant walls representing the brain parenchyma. These time-dependent models [16, 42, 78] (and references therein) are incapable of representing the complex dynamics of the cranium and provide little insight toward a more fundamental understanding of the development of hydrocephalus. Hakim [23] introduced a mechanical model describing the brain parenchyma as a porous sponge of viscoelastic material that compressed due to a pressure gradient causing the sponge cells to collapse. Nagashima [51] extended this model by applying Biot's theory of consolidation [7] and carried out simulations of the resulting mathematical model using the finite element method. This introduced one of the two current approaches to modeling the biomechanics of the brain parenchyma, namely the poroelastic model [32, 59, 79, 82, 85], in which the brain is considered to be a porous linearly elastic sponge saturated in a viscous incompressible fluid. These models account for the interaction of CSF with the brain parenchyma and thus can be used to model long-time scale phenomena such as the development of hydrocephalus. The second main approach is to model the brain parenchyma as a linear viscoelastic material [44–46, 89, 91]. Both linear viscoelastic and poroelastic models are based on the assumption of small strain theory which means that they are capable of predicting only small deformations. To correctly model the large deformations seen in hydrocephalus, nonlinear material laws are required and such models for brain parenchyma have been recently proposed in

[14, 21]. These models are able to successfully predict the large ventricular displacements seen in hydrocephalus. However, most of the above mentioned mechanical models suffer from the assumption that the brain's geometry is either a cylinder or a sphere.

In order for mechanical models of brain to be of clinical relevance their corresponding computational algorithms and software must incorporate the anatomical geometry of the brain as seen in medical images as well as efficient and robust numerical solvers. Therefore, the aim of our paper is to propose an elegant computational approach that combines medical image processing, level set methods, and moving meshes to simulate the mechanical response of hydrocephalic brains to treatments.

We propose a computational pipeline approach for evolution of the brain ventricles involving the following steps: image denoising, image segmentation based on a threshold method, prediction of the ventricular boundaries via the level set method, generation of computational meshes of the brain, mesh deformation using the finite element-based mesh warping (FEMWARP) method, and mesh quality improvement of the deformed meshes.

We review the literature in these areas in Sect. 2. We give a general introduction to level set methods and mesh warping methods and describe the specific methods being used in our combined level set/mesh warping approach in Sect. 3. Our computational pipeline approach for tracking the evolution of the brain ventricles is given in Sect. 4. Simulation results for the evolution of the brain ventricles in hydrocephalic patients post-treatment via shunt insertion are reported in Sect. 5. Section 6 explains our conclusions and future research plans.

2 Generation of Dynamic Biomedical Computational Models and Simulations

2.1 Generation of Image-Based Computational Models

Biomedical computational models which are derived from images are often created by following a specific pipeline approach [12]: (1) image processing and (2) surface/volume mesh generation. Dynamic computational models also require the inclusion of a third step, i.e., (3) mesh motion. We highlight several of the existing techniques in these three areas found in the literature. A comprehensive review is not the focus of this section, as the literature is extensive; hence, we focus on the most similar approaches to the proposed approaches.

2.2 Image Segmentation

The image segmentation problem [64] is to partition an image into nonoverlapping regions whose union is the entire image; each identified region shares a characteristic such as image intensity or texture [22, 24, 56]. For segmentation of the image, it is also important that each region be connected. (Pixel classification [35] is a related problem whereby the constraint that each region be connected is removed. Although this can sometimes be desirable in medical image analysis, we seek to determine a classical segmentation of the image into regions as opposed to a discrete, pixel classification segmentation.) In medical image segmentation, each region would ideally represent an anatomical structure.

There are numerous image segmentation techniques available to researchers today. Popular approaches for medical image segmentation [40, 64, 92] include: thresholding methods (e.g., [66]), region growing methods (e.g., [27] and [28]), classifier methods from pattern recognition (e.g., [2] and [6]), clustering methods (e.g., [6]), Markov random field model methods (e.g., [25]), artificial neural network approaches (e.g., [41]), deformable models (e.g., [33, 43, 68]), and atlas-guided approaches (e.g., [19, 69]).

Level-set methods (e.g., [1, 17, 48, 52–54, 60, 68, 88]) represent one very popular deformable model approach that have been used extensively for image segmentation and for other image processing problems, such as image registration. In particular, the level set approach delineates region boundaries using closed parametric curves (or surfaces) that deform under the influence of a PDE; the problem is cast as a front evolution problem. This front propagation approach is different from the earlier energy minimization evolution approaches, such as snakes (e.g., [33, 43]). The speed of the deformation is essential to the position of the final contours. Local curvature of the contour, intensity gradient, shape, and position contours have all been used for the speed term [68]. One important advantage of level set methods is they permit easier handling of topological changes.

2.3 Mesh Generation

The classical approach for generating computational meshes from images that have been segmented and registered has been to perform a surface interpolation between the contours describing the segmented volume [18, 65, 67]. Often this is done using the marching cubes (MC) algorithm [39]. However, the regularized marching tetrahedron (RMT) algorithm [83] yields topologically-consistent surface meshes, whereas the MC method does not.

Computational biomedical meshes with tetrahedral [12, 20, 30, 37, 81, 95] or hexahedral [47, 70, 87, 96] elements are typically created for finite element or finite volume simulations based on the surface mesh input. Hybrid meshes [15, 58, 94] have also been used when high accuracy is required but hexahedral mesh generation is infeasible due to biomedical data complexity. Another possibility is to generate a mesh based on the input of a level set [9]. Mesh optimization methods are used to improve the quality of biomedical meshes [26, 38, 86]; only recently have such techniques been designed for hybrid meshes [15].

2.4 Moving Meshes

Persson et al. [61–63, 80] developed a moving mesh technique based on the incorporation of level sets into an adaptive mesh refinement technique. They applied their moving mesh technique to image-based problems [61, 62]. Despite the fact that their applications involved a mesh, their algorithm does not compute the motion based upon the mesh; rather the motion is computed based upon the use of a Cartesian or octree background mesh and adaptive mesh refinement for mesh density control.

Mesh warping algorithms compute the deformation of the mesh from the source to the target domain based upon interpolation and/or extrapolation of the vertex coordinates. Typically the topology of the mesh is held fixed in order to allow for seamless integration with a numerical partial differential equation solver. Several mesh warping techniques for biomedical applications have been developed [3, 5, 36, 75–77]. However, they cannot be used for the development of computational models for tracking the evolution of the brain ventricles for hydrocephalus, as they do not incorporate the physics of the brain deformation due to the disease or its treatment.

2.5 Motivation for Current Study

Despite all of the research that has been performed in the areas of image processing, level sets, and dynamic mesh generation, there is no algorithm or software package which combines level sets with mesh warping. In addition, no such algorithms have been developed for tracking the evolution of the brain ventricles pre- and post-treatment of hydrocephalus.

3 Introduction to the Level Set Method and FEMWARP

In this section, we give an introduction to level set methods and mesh warping methods for applications with deforming domains. In addition, we describe the specific level set and mesh warping methods used to develop our combined level set/mesh warping approach in this paper. These methods are the energy minimization formulation of the level set method due to Chan and Vese and the finite element-based mesh warping (FEMWARP) method due to Shontz and Vavasis.

3.1 Level Set Methods

In this section, we give an introduction to the level set method and describe the particular level set method used in our work. The level set method is a numerical technique for tracking evolving interfaces, shapes, curves, or surfaces. The level set method was developed by Osher and Sethian in 1987 initially for problems in fluid dynamics [55]. However, there has been a significant amount of research on level set methods by numerous researchers, which has allowed for numerous extensions of the method and its applications to numerous other fields. Two significant advantages of level set methods are: (1) the deforming curve or shape need not be parametrized and (2) deforming shapes undergoing topological changes can easily be tracked. These advantages make the level set method ideal for tracking the evolution of the ventricles of the hydrocephalic brain.

3.1.1 The Chan and Vese Level Set Method for Curve Evolution

Here we describe the basic level set method for curve evolution [55]. First we define some notation. Let Ω prescribe a bounded open subset of $\mathbb{R}^2$, with boundary given by $\delta\Omega$. Let $u_0 : \bar{\Omega} \to \mathbb{R}$ denote a given image and $C(s) : [0,1] \to \mathbb{R}^2$ denote a parametrized curve. In level set methods, implicit representation of the curve C is given by a Lipschitz function ϕ. That is, $C = \{(x,y) | \phi(x,y) = 0\}$. The zero-level curve of the function at time t of the function $\phi(t,x,y)$ is used to evolve C. In order to evolve the curve C, a speed and direction must be prescribed.

The level set method we use in this paper is due to Chan and Vese and evolves the level set curve based on a level set formulation of an energy functional minimization [10]. In particular, suppose that $C \subset \Omega$ is given by the zero level set of a Lipschitz function $\phi : \Omega \to \mathbb{R}$ satisfying the following properties:

$$\begin{aligned} C = \partial\omega &= \big\{(x,y) \in \Omega : \phi(x,y) = 0\big\} \\ \text{inside}(C) = \omega &= \big\{(x,y) \in \Omega : \phi(x,y) > 0\big\} \\ \text{outside}(C) = \Omega \setminus \bar{\omega} &= \big\{(x,y) \in \Omega : \phi(x,) < 0\big\}, \end{aligned} \tag{1}$$

where C is the boundary of ω.

Let c_1 and c_2 denote constants depending on C which denote the averages of u_0 inside and outside of C, respectively, and let $F(c_1, c_2, \phi)$ denote the energy functional to be minimized. Furthermore, denote by $\mu \geq 0$, $\nu \geq 0$, λ_1, and λ_2 fixed parameters and by H_ε and δ_ε regularized Heaviside functions and one-dimensional Dirac measures.

Then the regularized energy functional which is minimized to obtain the curve evolution is defined as follows:

$$\begin{aligned} F_\varepsilon(c_1, c_2, \phi) = \mu &\int_\Omega \delta_\varepsilon\big(\phi(x,y)\big)\big|\nabla\phi(x,y)\big|\,dx\,dy \\ &+ \nu \int_\Omega H_e\big(\phi(x,y)\big)\,dx\,dy \\ &+ \lambda_1 \int_\Omega \big|u_0(x,y) - c_1\big|^2 H_\varepsilon\big(\phi(x,y)\big)\,dx\,dy \\ &+ \lambda_2 \int_\Omega \big|u_0(x,y) - c_2\big|^2 \big(1 - H_\varepsilon\big(\phi(x,y)\big)\big)\,dx\,dy. \end{aligned} \tag{2}$$

Letting $\phi(0,x,y) = \phi_0(x,y)$ denote the initial contour, then the speed of the level set is given by

$$\begin{aligned}
\frac{\partial \phi}{\partial t} &= \delta_\varepsilon(\phi)\left[\mu \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \nu - \lambda_1(u_0-c_1)^2 + \lambda_2(u_0-c_2)^2\right] \\
&= 0 \quad \text{in } (0,\infty)\times\Omega, \\
\phi(0,x,y) &= \phi_0(x,y) \quad \text{in } \Omega, \\
\frac{\delta_\varepsilon(\phi)}{|\nabla\phi|}\frac{\partial\phi}{\partial\mathbf{n}} &= 0 \quad \text{on } \partial\Omega,
\end{aligned} \tag{3}$$

where $\mathbf{n}$ denotes the exterior normal boundary $\partial\Omega$, and $\partial\phi/\partial\mathbf{n}$ denotes the normal derivative of ϕ at the boundary. More details are given in [10].

We solve the energy minimization problem given by (2) and (3) for the curve evolution using the Matlab implementation by Wu in [93].

3.2 Mesh Warping Methods

In this section, we describe mesh warping techniques for biomedical applications and the three-step finite element-based mesh warping (FEMWARP) method (see, e.g., [4]). Mesh warping methods are numerical techniques for deforming a mesh from a source to a target domain. Such techniques are needed when the geometric domain of interest deforms as a function of time, and the mesh must be updated at each time step in response to the deforming domain boundary in order for the mesh to remain a valid approximation of the geometry.

3.2.1 The Shontz and Vavasis Finite Element-Based Mesh Warping (FEMWARP) Algorithm

We base our description of the FEMWARP method upon the presentation of the algorithm given in [74]. Let M denote a triangular finite element mesh on a two-dimensional domain, Ω. Let b and m denote the numbers of boundary and interior vertices of M, respectively, and let $n = m + b$ denote the total number of vertices.

The first step of the FEMWARP algorithm is to represent each interior vertex in M as a specific linear combination of its neighboring vertices. In order to determine the weights the linear combination for each interior vertex, the $(m+b)\times(m+b)$ global stiffness matrix A for the following boundary value problem

$$\triangle u = 0 \quad \text{on } \Omega$$

with $u = u_0$ on $\partial\Omega$ is formed, where A is computed based on piecewise linear finite elements on M. Because only the relevant matrix is kept, any u_0 may be prescribed. (For a mathematical description of the entries in A, see [31].)

For simplicity, assume that the interior vertices of M are labeled $1, 2, \ldots, m$ and that the boundary vertices are labeled $m+1, m+2, \ldots, n$. Next, partition A as

$$A = \begin{pmatrix} A_I & A_B \\ A_B^T & X \end{pmatrix},$$

where A_I is $m \times m$, A_B is $m \times b$, and X is $b \times b$.

Next, let x be a vector containing the x-coordinates of the initial mesh vertices. Because any linear function of the coordinates lies in the null-space of the discretized Laplacian operator, it follows that $[A_I, A_B]x = 0$. A similar identity holds for the y-coordinates. Equivalently,

$$A_I x_I = -A_B x_B. \tag{4}$$

To represent each interior vertex as a linear combination of its neighboring vertices, one can divide each row of $[A_I, A_B]$ by the diagonal element in that row. This yields a linear system whose diagonal entries are 1's and whose row sums are 0's. Hence, each interior vertex is represented as a linear combination of its neighboring vertices.

The second step of the FEMWARP method is to transform the boundary vertices to new positions by applying a user-prescribed boundary deformation. Denote the new boundary vertex positions by $[x_B, y_B] \rightarrow [\hat{x}_B, \hat{y}_B]$.

The third step of the FEMWARP algorithm is to solve the above linear system of equations, i.e., (4) with a new right-hand side vector based on the new positions of the boundary vertices for the new coordinates of the interior vertices of $\hat{M}$ on $\hat{\Omega}$. In particular, we solve (5)

$$A_I[\hat{x}_I, \hat{y}_I] = -A_B[\hat{x}_B, \hat{y}_B] \tag{5}$$

for $[\hat{x}_I, \hat{y}_I]$.

FEMWARP maintains the topology of the mesh when warping M to $\hat{M}$; hence the mesh is fully satisfied after solving (5).

4 Ventricular Deformation for Boundaries Obtained from the Level Set Method and FEMWARP in Hydrocephalic Patients

Assuming as input medical images of the brain of a hydrocephalic patient taken at different times, a combined level set/mesh warping approach can be designed in order to track the evolution of the brain ventricles. In particular, the level set method given by (2) and (3) can be used in order to segment the medical images and determine the ventricular boundaries. The FEMWARP mesh warping technique described in Sect. 3.2 can used to deform the ventricular geometry from the source to the target ventricular boundary. Our combined approach is used in a computational pipeline involving the following steps: image denoising, image segmentation, obtaining boundary vertices via the level set method, mesh generation, mesh warping, and mesh quality improvement. Pseudocode for our proposed computational pipeline is shown in Algorithm 1. The following subsections give more details about each step in our computational pipeline.

Algorithm 1 Mesh warping with the level set method

```
1: **Input**: X ← medical image having initial ventricular boundary
2:          Y ← medical image having goal ventricular boundary
3: Image denoising using mask filters
4: Image segmentation via thresholding method
5:
6: Obtain ventricular boundary vertices from segmented X and Y medical images
   via level set method
7: A ← ventricular boundary vertices for X
8: B ← ventricular boundary vertices for Y
9:
10: Generate initial mesh with A using Triangle
11:
12: LOOP 1: Deform mesh from A to B using FEMWARP
13: if mesh is valid then
14:    Mesh quality improvement on the deformed mesh
15:    return mesh
16: else
17:    n = 1          // Intermediate mesh deformation
18:    while mesh is invalid do
19:       C ← $(\frac{1}{2})^n(B - A)$
20:       Deform mesh from A to C using small-step FEMWARP
21:       n = n + 1
22:    end while
23:    Mesh quality improvement on the deformed mesh
24:    A ← C
25: end if
26:
27: Go to LOOP 1
```

4.1 Image Denoising

In order to obtain the boundary vertices of the ventricles in the medical images, image denoising is first performed. By applying the appropriate mask filter to the image, the image is denoised and the image becomes easy to recognize the object.

4.2 Image Segmentation

After image denoising is performed, the image is segmented by a thresholding method. The method selects an appropriate threshold value for the image and divides the image into two parts: the ventricles and outside the ventricles based on the threshold value. In the segmented image (Fig. 1), the white-colored parts represent the ventricles we want to deform.

4.3 Obtaining Boundary Vertices

The level set method is then employed in order to obtain the boundary vertices of the ventricles in the post-treatment image. The contour constructed by the boundary vertices for the ventricles in the segmented pre-treatment image is used as the zero level set which is the input of the level set method.

Given the zero level set, the level set method moves the contour along its interior normal, and the contour evolves until it matches the boundary of the ventricles in the segmented post-treatment image.

4.4 Mesh Generation

The mesh used as an input for mesh warping is generated based on the boundary vertices of the ventricles in the segmented pre-treatment image. To generate the initial mesh, Triangle [71] is employed.

4.5 Mesh Deformation

To track the movement of the ventricles in the brain when hydrocephalus is treated by shunt insertion, the initial mesh is deformed until it matches the target ventricular boundary vertices for post-treatment. To deform the mesh, the FEMWARP [74] algorithm (see Sect. 3.2 for a description) is used.

When mesh deformation is performed, the deformed mesh may become tangled due to a large deformation. In this case, intermediate deformation steps are generated based on performing a backtracking line search between the pre- and post-treatment ventricular boundary vertices. Thus, a target which will yield a valid deformed mesh can be designed. An invalid mesh element can be detected by computing the sign of the determinant of an element's Jacobian matrix and comparing it to its original sign. If a tangled mesh is generated by the deformation, a backtracking line search brings the deformed vertices back to the halfway point of the deformation between the source and target vertex locations. This reduces the deformation size and prevents tangling. Small-step FEMWARP can then be used to successfully deform the mesh.

4.6 Mesh Quality Improvement

The deformed meshes often have poor mesh quality since large deformations cause poorly-shaped elements near to the moving ventricular boundary. It is well-known that the mesh quality effects the time to solve PDE, the condition number of the numerical linear system, and the accuracy of the PDE solution [72].

Hence, in order to improve the quality of the deformed meshes, mesh quality improvement is performed using the Mesh Quality Improvement Toolkit (Mesquite) Version 2.1.4 [8]. The objective function used in this study is

$$f(x) = \frac{1}{n} \sum_{1 \le i \le n} q_i^2, \tag{6}$$

where f is the overall mesh quality as measured by the average of the sequence of the element qualities, q_i is the quality of element i, and n is the number of elements in the mesh.

The inverse mean ratio metric [50] was used as the mesh quality metric for the mesh optimization in (6). The formula for computing the inverse mean ratio of an element is given by

$$q = \frac{\|AW^{-1}\|_F^2}{2\det(AW^{-1})}, \tag{7}$$

where A is the Jacobian matrix for the physical triangle, and W is a Jacobian matrix for the reference triangle. The range for this quality metric is 1 to ∞ for non-inverted elements. Inverted elements correspond to a negative value of the metric. Since an equilateral triangle is the ideal element for the mesh optimization procedure in this study, 1 is the ideal value of the metric. Hence, lower values of the metric correspond to meshes with better quality.

During the mesh optimization process, the boundary vertices are held fixed. In addition, the initial meshes and subsequent meshes are not allowed to contain any inverted elements. In order to minimize (6), a local implementation of the feasible Newton method [50] is employed. We terminate the mesh optimization process after obtaining the same value of the objective function to six digits of accuracy on successive iterations.

5 Simulations of the Evolution of the Brain Ventricles in Hydrocephalic Patients

Three simulations were designed to track the evolution of the brain ventricles upon treatment of hydrocephalus via shunt insertion. The simulations were performed based on our proposed combined level set/mesh warping algorithm (see Algorithm 1). The level of difficulty varied in these simulations, from simple to complex based on the specific deformation of the ventricles. Two medical image sets derived from [90] were used in simulations. The first set of medical images included two CT images, corresponding to pre- and post-treatment. The second set included three CT images, corresponding to pre-treatment and two time periods post-treatment. The Solaris machine used for the simulations of the evolutions of the ventricles was an UltraSPARC-III CPU with a 750 MHz processor, 1 GB SDRAM of memory, and an 8 MB L2 cache.

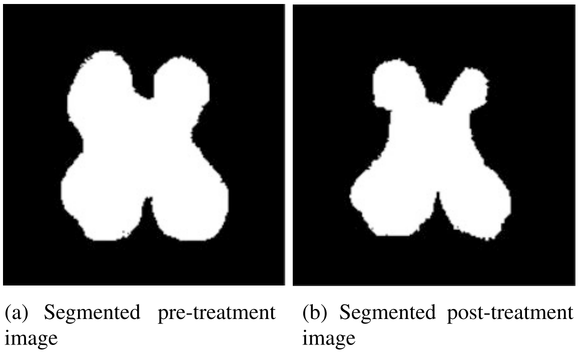


Fig. 1 Two segmented images: pre- and post-treatment. The image was denoised with a 4 × 4 mask filter. After image denoising, the images were segmented

(a) Segmented pre-treatment image (b) Segmented post-treatment image

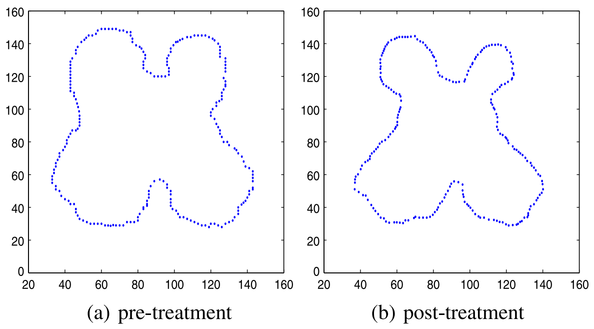


Fig. 2 Boundary vertices obtained from pre- and post-treatment segmented images. The ventricular boundary movement due to shunt insertion is shown

(a) pre-treatment (b) post-treatment

5.1 Simulation 1: Small Decrease in the Area of the Ventricles

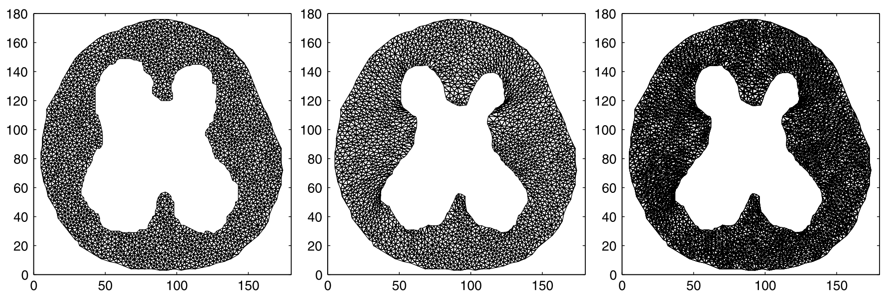
We obtained pre- and post-treatment (three months later) CT images [90] for a hydrocephalic patient who was treated by shunt insertion. In this simulation, the sixth line in Algorithm 1 was performed manually without using the level set method. Since we tested a simple case of the deformation in this simulation, the lines from 16 to 25 in Algorithm 1 (intermediate mesh deformation) were not performed. To track the evolution of the brain ventricles, the images were first denoised by using a 4 × 4 mask filter for both images.

After denoising the images, the images were segmented based on a threshold value for the pixels. For this simulation, a threshold value of 20 was applied. The images obtained from image denoising and image segmentation are shown in Fig. 1. In the segmented images, the white-colored parts represent the ventricles containing CSF whose evolution was tracked in our numerical simulation.

Once image segmentation was performed, the boundary vertices were easily obtained. By tracing the shape of the ventricles, the boundary vertices used in the mesh generation step were obtained. Figure 2 shows the boundary vertices obtained from the segmented images. Each ventricular boundary contains 225 vertices in the boundary. Each vertex has the same Euclidean distance from itself to its neighboring vertices.

Table 1 Inverse mean ratio mesh quality statistics for several meshes used in the simulation. The feasible Newton method [50] implemented in Mesquite [8] was used for average mesh quality improvement

Mesh	Inverse mean ratio mesh quality				
	min	avg	rms	max	std
Initial mesh	1.00002	1.05555	1.05816	2.0110	0.07427
Deformed mesh	1.00001	1.31565	2.36839	82.6890	1.96934
Optimized mesh	1.00000	1.14067	1.15345	3.8571	0.18090



(a) Initial mesh (b) Deformed mesh (c) Optimized deformed mesh

Fig. 3 (**a**) Initial mesh generated by Triangle [71] and (**b**) the deformed mesh generated by FEMWARP mesh warping algorithm. (**c**) Mesh quality improvement was performed on the deformed mesh to improve the mesh quality

After the boundary vertices were obtained from the segmented images, the initial mesh for the boundary vertices of pre-treatment image was generated using Triangle [71]. The mesh contained 2298 vertices and 4245 elements, and its initial quality is given in Table 1. Based on the initial mesh and the final ventricular boundary vertices obtained from the post-treatment image, mesh deformation for tracking the shape evolution of the brain ventricles was performed via FEMWARP [74].

The initial mesh and the deformed meshes resulting from mesh deformation and mesh quality improvement procedures are shown in Fig. 3. Since the area of the ventricles was decreased as keeping similar shapes in this simulation, the deformation was small. Because of this, FEMWARP easily controlled the deformation, so a tangled mesh was not generated.

As the final step of the simulation, the deformed mesh was optimized to improve the mesh quality by using Mesquite [8]. Table 1 and Fig. 4 show the mesh quality statistics and the quality distribution for each mesh generated during the simulation.

The average mesh quality of the initial mesh was 1.05555. After mesh deformation, the mesh quality of the deformed mesh was 1.31565. The number of good quality elements decreased from 1550 to 1050. By performing mesh quality improvement, the quality of the deformed mesh was improved to 1.14067. The number of poor quality elements decreased by more than half compared to the unoptimized

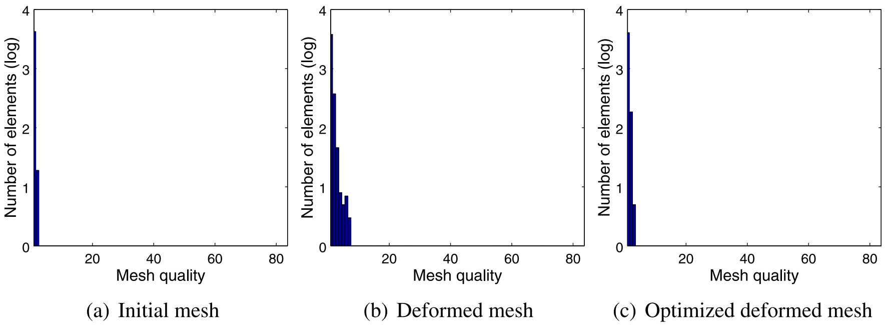
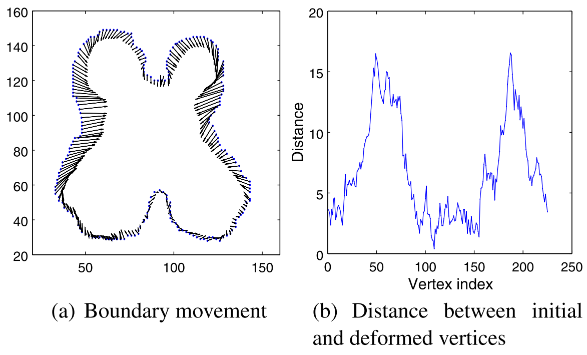


Fig. 4 Inverse mean ratio mesh quality distribution for meshes generated in Simulation 1

Fig. 5 Boundary movement of each vertex and their corresponding distances between pre- and post-treatment deformation



deformed mesh. The number of high quality elements (i.e., those with quality values ranging from 1 to 1.2) increased from 1700 to 2400 due to the mesh optimization process. Also, the quality of the worst element in the deformed mesh improved significantly from 82.689 to 3.857.

The distances for each boundary vertex during the deformation are visualized in Fig. 5. The distances were computed by calculating the Euclidean distance between corresponding ventricular boundary vertices in the pre- and post-treatment images. During the mesh deformation, the shapes of the ventricles containing the CSF changed. Since the sizes of the vectors in the middle-left and the upper-right parts of the ventricles were bigger than the corresponding vectors for the other parts, more boundary vertex movement occurred in the middle-left and the upper-right parts of the ventricles compared to the other parts. It is also shown in Fig. 5(b) that the boundary vertex movements around the vertices indexed 50 and 200 were larger than those of other vertices. Note that the boundary vertices were indexed from the starting point (90, 59) and moving in a clockwise direction.

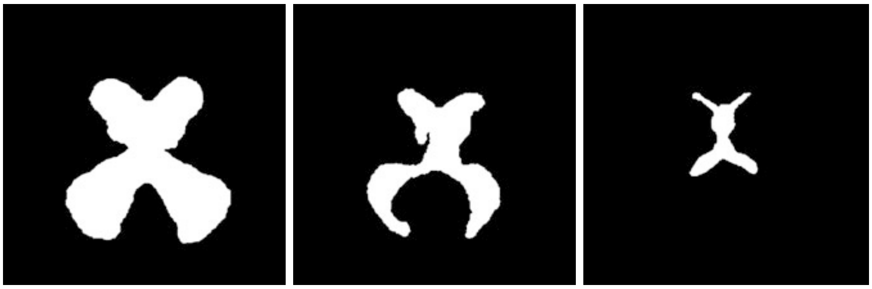

(a) Pre-treatment (b) Period 1 (c) Period 2

Fig. 6 Segmented images for obtaining the boundary vertices from the given three CT images: (**a**) pre-treatment, (**b**) period 1 (six months later), and (**c**) period 2 (one year later). The *white-colored* parts represent the ventricles which deform as their fluid volume decrease after shunt insertion

5.2 Simulation 2: Asymmetric Ventricular Shape Change

Three CT images [90] were used in the simulation: pre-treatment, period 1 (six months later), and period 2 (one year later). The goal of this experiment was to simulate ventricular deformation from pre-treatment to period 1 and from period 1 to period 2. In this simulation, the intermediate mesh deformation in Algorithm 1 (i.e., lines 16 through 25) was performed. Also, similar to the previous simulation, the ventricular boundary vertices were obtained with a manual process, and not with the level set method specified in the sixth line in Algorithm 1.

To reduce the noise in the image, a 3×3 mask filter for the pre-treatment CT image, a 6×6 mask filter for the period 1 CT image, and a 4×4 mask filter for the period 2 CT image were applied. After image denoising was performed, the images were segmented based on an appropriate threshold value. In this simulation, the threshold values 20, 77, 45 were used for the pre-treatment, period 1, and period 2 images. The segmented images to be used for obtaining the boundary vertices are shown in Fig. 6. The white-colored parts in the segmented images represent the brain ventricles which deform as their fluid volumes decrease after the treatment via shunt insertion.

The boundary vertices obtained from the segmented images are shown in Fig. 7. For each boundary vertex, the next vertex is selected from the boundary vertices with a fixed Euclidean distance from the given vertex. The boundary vertices were computed and ordered by repeating this process.

Using the ventricular boundary vertices for pre-treatment, the initial mesh was generated using Triangle [71]. The mesh contained 4311 vertices and 8166 elements; its mesh quality in given in Table 2. By using the boundary vertices obtained from the segmented images, the first mesh deformation step from pre-treatment to the period 1 boundary was performed via FEMWARP [74]. Mesh deformation results from the initial mesh to the period 1 boundary are shown in Fig. 8.

As can be seen in Fig. 8, since inverted elements were generated during mesh warping, the deformed mesh cannot be used as an input mesh for the next deforma-

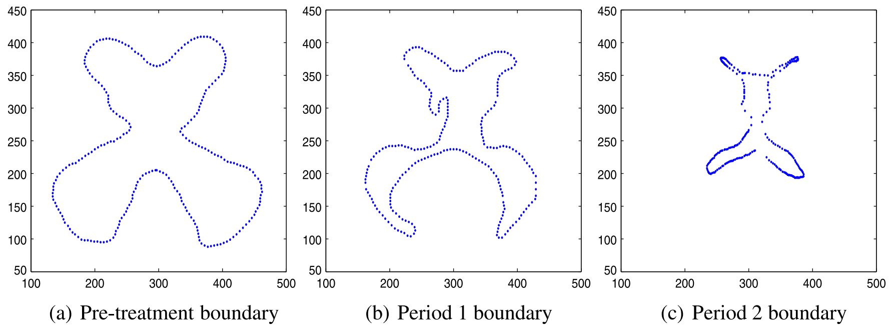


Fig. 7 Boundary vertices obtained from the segmented images: pre-treatment, period 1, and period 2

Table 2 Inverse mean ratio mesh quality statistics for several meshes used in the simulation of the ventricular mesh deformation from pre-treatment to period 1. The feasible Newton method [50] implemented in Mesquite [8] was used for average mesh quality improvement

Mesh	Inverse mean ratio mesh quality				
	min	avg	rms	max	std
Initial mesh	1.00000	1.11483	1.12096	2.4723	0.11708
6th deformation	1.00000	1.13918	1.17003	10.9184	0.26693
Opt. 6th deformation	1.00002	1.13266	1.16431	5.1703	0.26964
9th deformation	1.00001	1.16464	1.19760	14.5821	0.27905
Opt. 9th deformation	1.00001	1.15143	1.19106	6.5797	0.30469

tion step, as tangled meshes are not allowed to be used as an input for finite element methods, including methods such as FEMWARP which are based on a finite element method. The majority of the inverted elements were located in the marked area shown in Fig. 8(b). Compared to the other ventricular boundary areas, the mesh deformation in the marked areas was larger. To avoid tangled mesh generation, several intermediate mesh deformation steps for use with small-step FEMWARP [73] were designed. The intermediate steps showed more details of the ventricular evolution when the hydrocephalus was treated by shunt insertion.

To generate the intermediate mesh deformation steps, the new boundary vertices between the pre-treatment and period 1 boundary vertices were obtained by manual selection of the intermediate boundary vertices. By using the newly obtained boundary vertices, the mesh deformation was performed. Since adding the intermediate mesh deformation steps decreased the size of the deformation, small-step FEMWARP easily handled the deformation without generating inverted elements in the deformed meshes.

Nine total intermediate mesh deformation steps were designed from the initial mesh generated by pre-treatment boundary vertices to the deformed mesh to be generated by period 1 boundary vertices. Figure 9 shows the mesh deformation re-

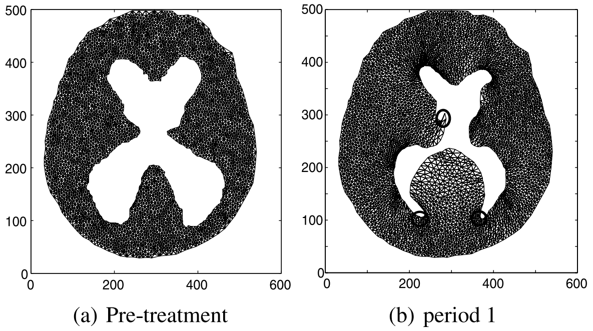


Fig. 8 The first mesh deformation of the ventricular boundary from pre-treatment to period 1. Inverted elements exist in the *marked areas* where a large deformation occurred

sults from pre-treatment to the period 1 ventricular boundaries with intermediate deformation steps. In Fig. 9, the initial mesh, the meshes resulting from the sixth deformation, and the meshes resulting from the ninth deformation are shown.

Mesh quality improvement was performed for each deformed mesh before deforming the mesh again in order to improve the mesh quality. The inverse mean ratio mesh quality statistics and element quality distribution results are shown in Table 2 and Fig. 10.

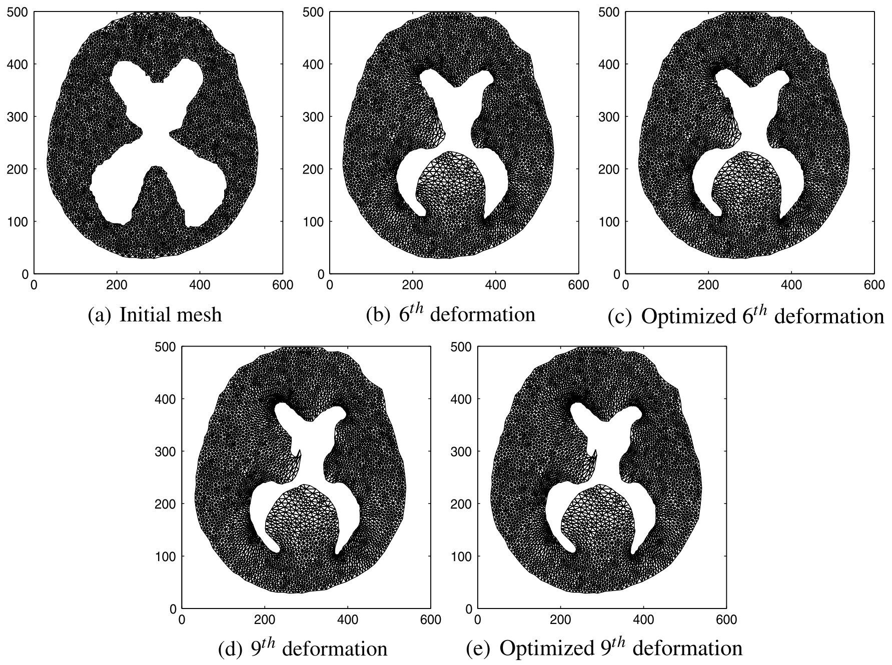
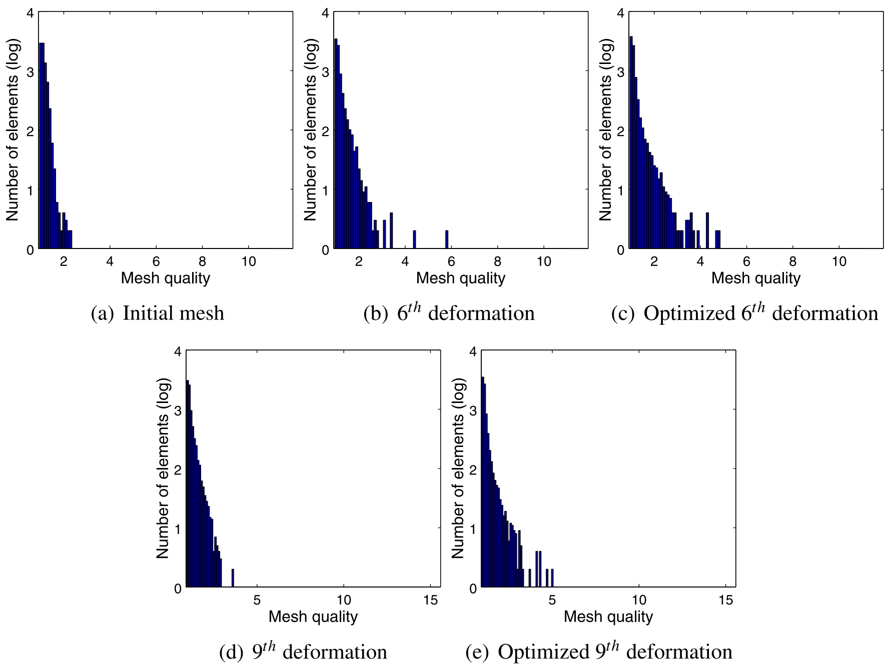


Fig. 9 (**a**) The initial mesh generated by using Triangle [71] and (**b**) and (**d**) the deformed meshes generated by the FEMWARP algorithm [74]. The mesh resulting from the ninth deformation matched to the boundary vertices of the ventricles for period 1. (**c**) and (**e**) The improved deformed meshes after use of mesh quality improvement



(a) Initial mesh (b) 6th deformation (c) Optimized 6th deformation

(d) 9th deformation (e) Optimized 9th deformation

Fig. 10 Inverse mean ratio element quality distribution for the meshes generated in Simulation 2 (ventricular mesh deformation from pre-treatment to period 1). Quality distributions for (**a**) the initial mesh, (**b**) the sixth intermediate mesh, (**c**) the sixth intermediate mesh after mesh quality improvement was performed, (**d**) the ninth intermediate mesh, and (**e**) the ninth intermediate mesh after mesh quality improvement was performed

The average quality of the initial mesh was 1.11483 according to the inverse mean ratio mesh quality metric. The average quality of the sixth intermediate mesh improved from 1.13918 to 1.1326 by performing mesh quality improvement. The worst mesh quality of an element in the sixth intermediate mesh showed a noticeable improvement as a result of mesh quality improvement, as it decreased from 10.9184 to 5.1703.

The ventricular boundary vertices in the ninth intermediate mesh matched exactly the ventricular boundary vertices obtained from the segmented image for period 1. Due to mesh quality improvement, the average mesh quality improved from 1.16464 to 1.15143; the worst mesh quality of an element in the ninth intermediate mesh improved from 14.5821 to 6.57973.

In Fig. 10, it can be seen that most of the mesh elements (approximately 90% of the elements) for the ninth intermediate mesh had good mesh qualities after mesh quality improvement was performed. Also, the number of poor quality mesh elements decreased. Since the poor quality mesh elements tended to generate inverted elements if the mesh was used as an input for the mesh deformation, reducing the number of poor quality mesh elements makes the next intermediate mesh deformation step more likely to succeed.

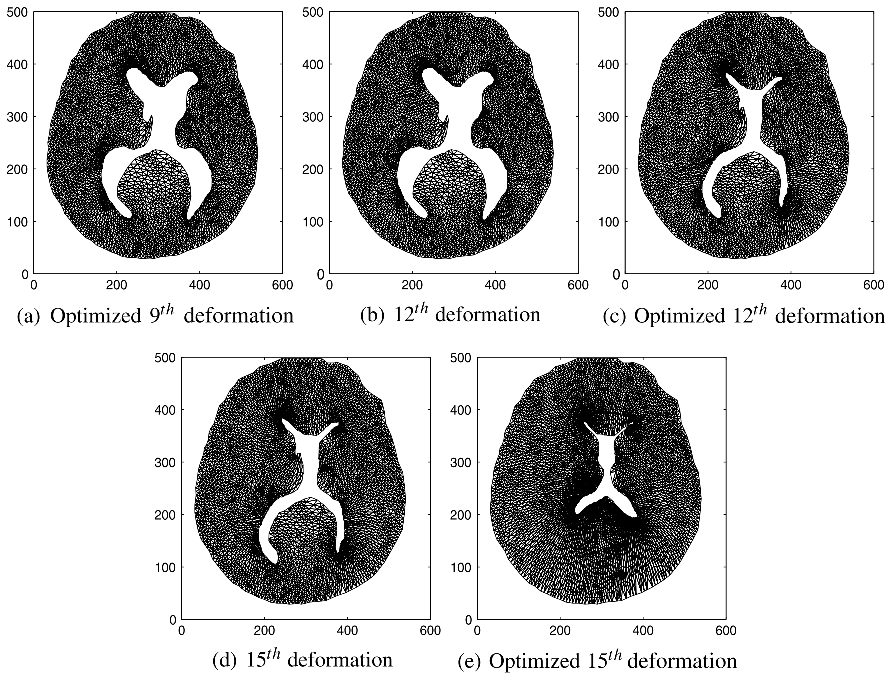

(a) Optimized 9th deformation
(b) 12th deformation
(c) Optimized 12th deformation
(d) 15th deformation
(e) Optimized 15th deformation
0
100
200
300
400
500
600

Fig. 11 **(a)** The mesh with the ventricular boundary vertices matched to the period 1 ventricular boundary vertices. **(b)** and **(d)** The deformed meshes generated by FEMWARP [74]. The fifteenth intermediate deformed mesh result matched exactly to the boundary vertices for the ventricles in the segmented period 2 image. **(c)** and **(e)** Mesh quality improvement was performed to improve the quality of the meshes at each step of the ventricular deformation

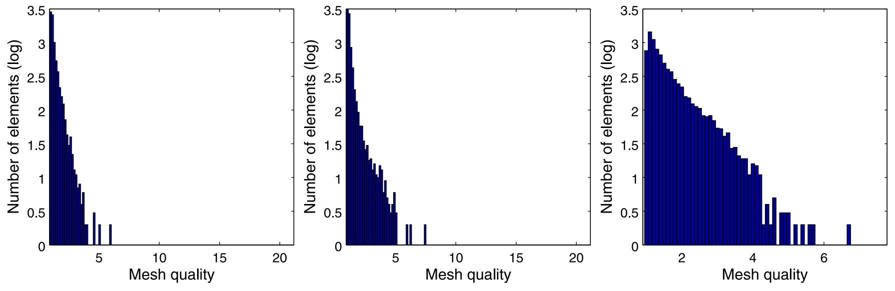
From the ninth intermediate mesh, the deformation to the ventricular boundary for period 2 was performed. After six of the intermediate meshes were generated, the boundary vertices of the deformed mesh matched to the boundary vertices of the ventricles in the segmented period 2 image. Figure 11 shows the mesh deformation results from period 1 to period 2. Six intermediate deformation steps were generated for this part of the simulation. In this figure, the mesh which having the boundary vertices matched to the ventricular boundary vertices for period 1, the twelfth intermediate step, and the fifteenth intermediate step are shown. The ventricular boundary vertices in the fifteenth intermediate mesh matched exactly the ventricular boundary vertices for period 2.

Mesh deformation from period 1 to period 2 boundary of the ventricles was easier than that for pre-treatment to period 1. Mesh deformation from period 1 to the period 2 ventricular boundary required fewer intermediate mesh deformation steps than did the earlier part of the simulation. This is because the ventricles shrunk significantly, allowing for more feasible possibilities for interior vertex positions.

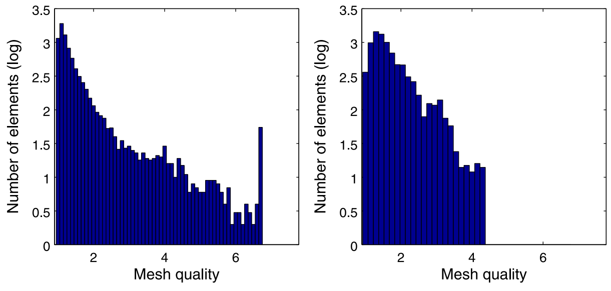
Table 3 and Fig. 12 show the inverse mean ratio mesh quality improvement results and the quality distribution for the deformed meshes from the simulation of the ventricular deformation from period 1 to period 2.

Table 3 Inverse mean ratio mesh quality statistics for several meshes used in the simulation. The feasible Newton method [50] implemented in Mesquite [8] was used for average mesh quality improvement. For improving the worst quality of the final mesh, the PS mesh quality improvement algorithm [57] was applied to obtain further improvement

Mesh	Inverse mean ratio mesh quality				
	min	avg	rms	max	std
12th deformation	1.00001	1.26625	1.36000	20.1674	0.49619
Opt. 12th deformation	1.00001	1.24535	1.34960	11.8054	0.52010
15th deformation	1.00020	1.57492	1.70020	6.7762	0.64054
Opt. 15th deformation	1.00002	1.53606	1.81146	18.4508	0.96015
Opt. 15th deformation (with PS)	1.00015	1.68952	1.78281	4.3440	0.56913



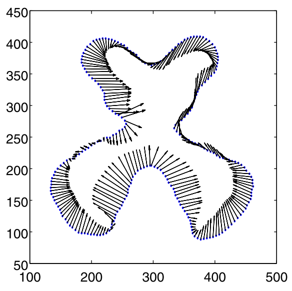
(a) 12^{th} deformation (b) Optimized 12^{th} deformation (c) 15^{th} deformation



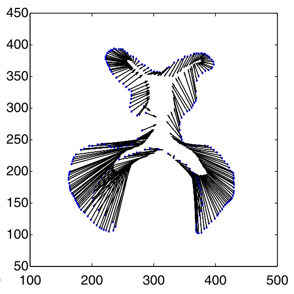
(d) Optimized 15^{th} deformation (e) Optimized 15^{th} deformation (with PS)

Fig. 12 Inverse mean ratio mesh quality distribution for meshes generated in Simulation 2 (ventricular mesh deformation from period 1 to period 2). Quality distribution for (**a**) the twelfth intermediate mesh, (**b**) the twelfth intermediate mesh after mesh quality improvement was performed, (**c**) the fifteenth intermediate mesh, (**d**) the fifteenth intermediate mesh after mesh quality improvement was performed, and (**e**) the fifteenth intermediate mesh after the worst quality element improvement was performed

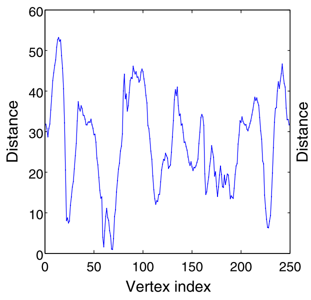
Fig. 13 (**a**) Boundary movement of each vertex from pre-treatment to period 1; (**b**) boundary movement of each vertex from period 1 to period 2; (**c**) the distance between the pre-treatment and period 1 vertices; (**d**) the distance between the period 1 and period 2 vertices



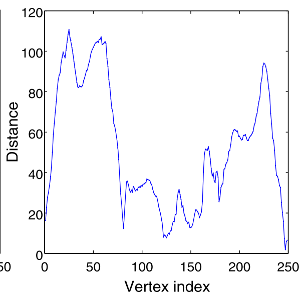
(a) Boundary movement from pre-treatment to period 1



(b) Boundary movement from period 1 to period 2



(c) Distance between pretreatment and period 1 ventricular boundary vertices



(d) Distance between period 1 and period 2 ventricular boundary vertices

As shown in Table 3, the average mesh quality for the twelfth intermediate mesh improved from 1.2625 to 1.24535. The worst mesh quality also improved from 20.1674 to 11.8054. Figure 12 shows that approximately 90% of the mesh elements in the twelve intermediate mesh have good mesh qualities after mesh quality improvement was performed.

Although the average mesh quality of the fifteenth intermediate mesh improved from 1.57492 to 1.53606 by performing mesh quality improvement with Mesquite [8], still many poor mesh quality elements existed in the mesh, and the worst mesh quality of the fifteenth intermediate mesh degraded from 6.7762 to 18.4508. Thus, to reduce the number of poor quality mesh elements, mesh quality improvement of the worst mesh element was performed by using the pattern search (PS) mesh quality improvement algorithm [57]. By using this algorithm, the quality of the worst mesh element improved from 18.4508 to 4.3440. Most of the mesh elements had good mesh qualities, in spite of the fact that the average quality of the mesh increased slightly to 1.68952.

The distances for boundary vertex movement during the deformations are visualized in Fig. 13. For the deformation from pre-treatment to period 1, most of the ventricular boundary vertices moved symmetrically except in the middle-left parts of the ventricles. The shapes for the middle-left parts of the ventricles changed significantly more than that of middle-right parts. This is because the shunt was inserted in the ventricles in this spot in order to treat the hydrocephalus. The sizes of the vectors

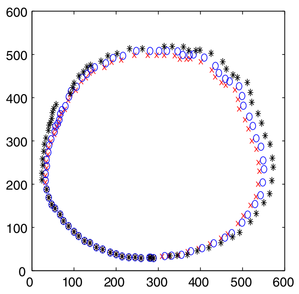


Fig. 14 Brain boundary vertices obtained from the segmented images. The $\times$, $\circ$, and $*$ symbols represent the pre-treatment, period 1, and period 2 brain boundaries, respectively

in the two lower parts and upper-left parts of the ventricles were bigger than the sizes of the corresponding vectors for the other parts. Figure 13(c) shows the distances from the pre-treatment to the period 1 ventricular boundary vertices. The distances are computed by calculating the Euclidean distance from pre-treatment ventricular boundary vertices the corresponding vertices in period 1. Note that they do not represent the cumulative distances moved during the mesh warping procedure. As can be seen in Fig. 13(c), the boundary vertex movements around the vertices indexed 5, 100, and 240 were larger than those of the other vertices.

In the case of the deformation from period 1 to period 2, most of the ventricular boundary vertices showed large movements during the deformation. In Fig. 13(b), the changing shapes of the lower parts of the ventricles were especially noticeable. As can be seen in Fig. 13(d), although the boundary vertex movements for most of the vertices were large, the boundary vertices located in the lower parts of the ventricles (indexed between 0 and 70 and between 230 and 250) showed the largest movement in the deformation from period 1 to period 2.

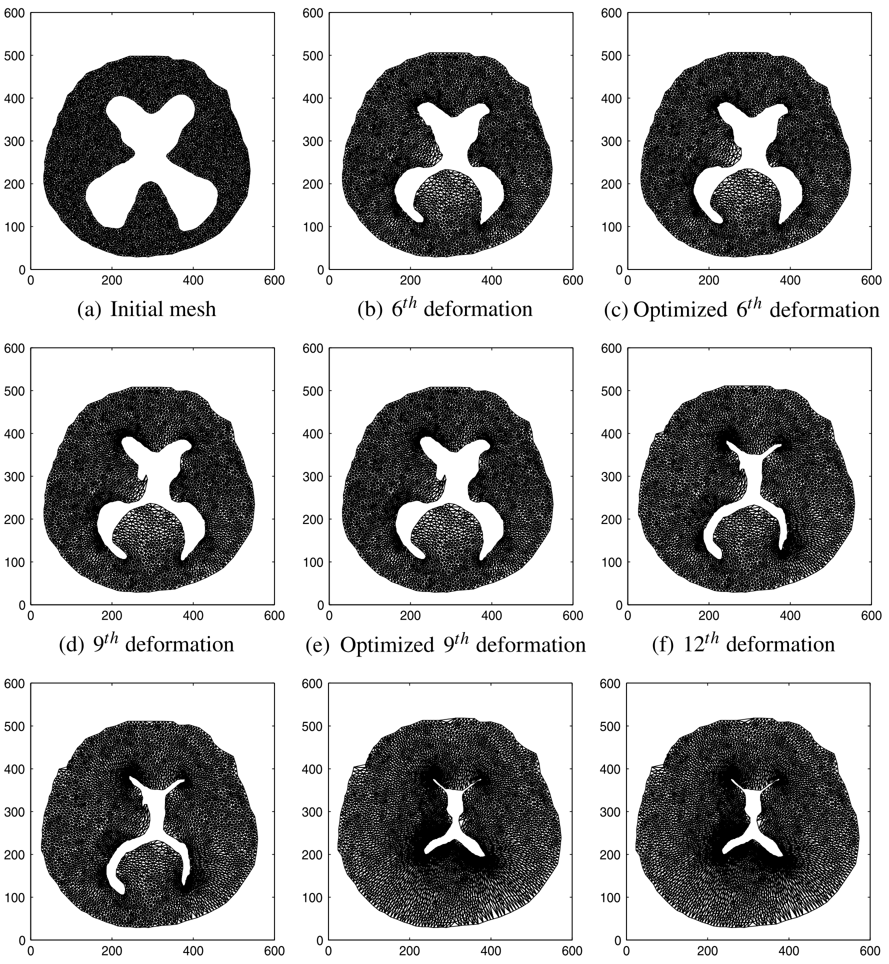
During the treatment of hydrocephalus, the brain size also changed. In particular, the brain size increased due to infant growth. The brain boundaries obtained from the segmented images are shown in Fig. 14.

Mesh deformation results from pre-treatment to period 1 and from period 1 to period 2 are shown in Fig. 15. In each step, deformation of both the ventricles and the brain were performed.

5.3 Simulation 3: Ventricular Deformation with Boundaries Obtained via the Level Set Method

In simulation 2, the ventricular boundary vertices in the segmented images were manually determined. Furthermore, the intermediate steps for avoiding the generation of inverted elements in the meshes used for the simulation of the ventricular deformations were computed by manually selecting the vertices between the pre-treatment and period 1 or between the period 1 and period 2 boundary vertices of the ventricles.

The main new aspect of this simulation is that the level set method [68] was applied to automatically obtain the boundary vertices of the ventricles in the seg-



(a) Initial mesh (b) 6th deformation (c) Optimized 6th deformation

(d) 9th deformation (e) Optimized 9th deformation (f) 12th deformation

(g) Optimized 12th deformation (h) 15th deformation (i) Optimized 15th deformation

Fig. 15 (**a**) The initial mesh generated by Triangle [71] and (**b**), (**d**), (**f**), and (**h**) the deformed meshes generated by the FEMWARP algorithm [74]. The fifteenth intermediate mesh deformation result matched exactly to the boundary vertices of the ventricles and the brain in segmented period 2 image. (**c**), (**e**), (**g**), and (**i**) Mesh quality improvement was performed to improve the quality of the meshes at each step of ventricular deformation for hydrocephalus

mented images. Using the boundary vertices, the initial mesh was generated using Triangle [71]. The mesh contained 2392 vertices and 4422 elements; its quality is given in Table 4. Also, with the boundary vertices of the ventricles in the pre-treatment, period 1, and period 2 images, the intermediate steps were automatically computed by using a backtracking line search method to obtain valid intermediate meshes.

Table 4 Inverse mean ratio mesh quality statistics for several meshes used in the simulation. The feasible Newton method [50] implemented in Mesquite [8] was used for average mesh quality improvement. For improving the worst quality of the final mesh, the PS mesh quality improvement algorithm [57] was applied to obtain further improvement

Mesh	Inverse mean ratio mesh quality				
	min	avg	rms	max	std
Initial mesh	1.00001	1.05380	1.05731	2.3513	0.08604
6th deformation	1.00000	1.20431	1.50967	36.3021	0.91035
Opt. 6th deformation	1.00001	1.18070	1.22330	5.5055	0.32000
9th deformation	1.00005	1.38846	1.51427	11.3377	0.60432
Opt. 9th deformation	1.00007	1.37350	1.53049	9.1011	0.67520
12th deformation	1.00043	1.50317	1.65771	21.9930	0.69892
Opt. 12th deformation	1.00004	1.47862	1.67494	9.0810	0.78682
16th deformation	1.00061	1.86964	2.24460	14.0210	1.24204
Opt. 16th deformation	1.00002	1.53606	1.81146	18.4508	0.96015
Opt. 16th deformation (with PS)	1.00073	1.91943	2.37001	8.9290	1.24050

Similar to the previous simulation, image denoising was performed as the first step of the simulation. The same mask filters were applied to denoise each medical image. After the images were denoised, image segmentation was performed based on the same threshold values used in the previous simulation. The segmented images were the same as before and are shown in Fig. 6.

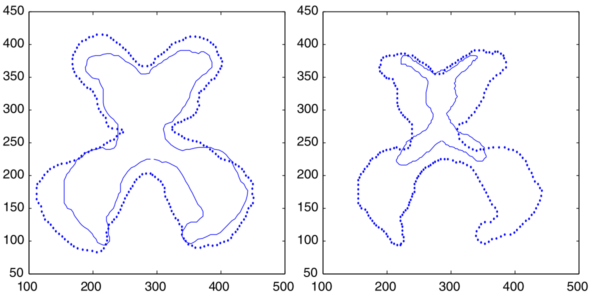
The level set method was applied to obtain the boundary vertices of the ventricles in the period 1 and period 2 images. To use the level set method, the zero level set was defined and used as an input. The contour constructed by the boundary vertices for the ventricles in the segmented pre-treatment image was used as the zero level set.

To obtain the boundary vertices of the ventricles in the segmented pre-treatment image, the contours were first generated in the segmented pre-treatment image. The pixel values of the segmented images were used to generate the contours. Since the ventricles in the segmented image were represented as pixels with a value of zero and all the other parts were represented as pixels with a value of one, the contours were plotted around the ventricles in the segmented image. The contour for the zero function matched the ventricular boundary in the segmented pre-treatment image.

The contour with the zero function was used as an input of the level set method and has used to obtain the boundary of the ventricles in the segmented image for period 1. Given the zero level set, the level set method moved the contour toward its interior normal with constant speed (a value of approximately $8.854e^{-12}$ was used for ε_0 in this simulation). The evolution was stopped when the contour matched the ventricular boundary in the segmented period 1 image. Sixty iterations of the level set method were required to evolve the contour.

The level set method was also used to generate the boundary of the ventricles for period 1 with complicated changes of shape (where the shunt was inserted). With

Fig. 16 Pre-treatment, period 1, and period 2 boundaries obtained from the level set method. The *dotted boundaries* represent the initial contour, and the *solid line* boundaries represent the evolved contour, which is matched to the boundary



(a) Pre-treatment and period 1 boundaries obtained from the level set method

(b) Period 1 and period 2 boundaries obtained from the level set method

the ventricular boundaries obtained from the level set method, several intermediate steps were generated automatically via linear interpolation. However, it was challenging to compute the intermediate boundaries near the inserted shunt via linear interpolation. This technique generated meshes with several inverted elements; most of them were located near where the shunt was inserted. Also, determining other types of interpolation automatically would be difficult since it is hard to know what type to use. Thus, in order to avoid generation of inverted elements and to reduce the number of intermediate steps for ventricular deformation from pre-treatment to period 1, the boundary of the ventricles for period 1 was obtained by deleting the shunt in the segmented period 1 image.

When the ventricular boundary for period 1 was obtained, the boundary was used as an input contour of the level set method which was used to obtain the ventricular boundary for period 2. Similar to the process performed for obtaining the ventricular boundary vertices for period 1, the level set method evolved the boundary of the ventricles for period 1 along its interior normal with the constant speed given above. After 53 iterations, the level set method was terminated after the new boundary matched the ventricular boundary for the segmented period 2 image.

The contours obtained by the level set method for representing the boundary of the ventricles in each segmented image included a different number of vertices. However, the same number of boundary vertices for each ventricular boundary is required for deformation using FEMWARP.

To represent the contours with the same number of vertices, an identical number of vertices were chosen from the starting positions in each segmented image. For the pre-treatment boundary, the vertex with the xy-coordinates $(300, 200)$ was used as a starting point, and every fourth vertex in the contour was selected as vertices used for representing the boundary of the ventricles for pre-treatment. From the vertex with the xy-coordiates $(300, 230)$, every third vertex in the contour was selected to use as vertices used for the period 1 ventricular boundary. For the period 2 boundary, the vertex with the xy-coordinates $(295, 250)$ was used as a starting point for the vertex selection. Every second vertex in the contour was selected as the ventricular boundary vertices for period 2.

The boundaries of the ventricles for pre-treatment, period 1, and period 2 obtained from the level set method are shown in Fig. 16. The dotted plot in Fig. 16(a) represents the boundary for pre-treatment generated by the contour plot with the zero function value, and the solid line plot shows the boundary for period 1 obtained by evolving the contour (dotted plot) by the level set method. In Fig. 16(b), the dotted plot is the contour which represents the boundary for period 1, and the solid line plot shows the evolved contour for the period 2 boundary obtained from the level set method.

With the boundary vertices obtained from the level set method, the evolution of the ventricles containing cerebrospinal fluid was simulated. Since the ventricular deformation both from the pre-treatment to period 1 boundaries and from the period 1 to period 2 boundaries was too large to be handled by FEMWARP [74] in just one step, intermediate small deformations were performed (i.e. using small-step FEMWARP [73]).

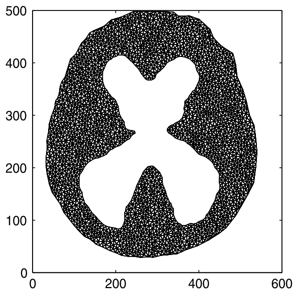
The intermediate steps were computed based on application of a backtracking line search method for each boundary vertex. The pseudocode for computing the intermediate steps is shown in Algorithm 1 line from 17 to 22. Mesh quality improvement for the intermediate step is performed once the deformed mesh is valid. The optimized mesh is used as an input in the computation of the next intermediate step.

Figure 17 shows the initial mesh generated using Triangle [71] with the ventricular boundary vertices for pre-treatment and the intermediate meshes generated during ventricular mesh deformation. For each intermediate mesh, mesh quality improvement was performed.

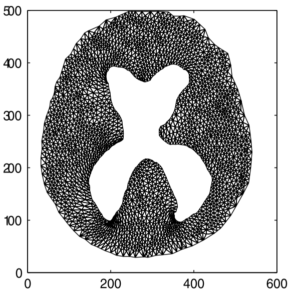
To obtain the mesh matched to the ventricular boundary vertices for period 2, sixteen intermediate steps were performed to deform the mesh from pre-treatment to period 2. Stretched triangular elements occurred near the brain boundary in the sixteenth intermediate mesh are seen in the Fig. 17(i). This is because the significant shrinkage of the ventricles happened in the sixteenth intermediate step compared to the initial mesh. Thus, the interior vertices and correspondingly the triangular elements were strained toward the ventricular boundary for period 2 in order to cover the increased area.

The inverse mean ratio mesh quality statistics and quality distribution for the deformed meshes generated during the simulation are shown in Table 4 and Fig. 18. The average mesh quality of the initial mesh was 1.0538. When each mesh deformation was performed, the average mesh quality observed for the meshes on the intermediate steps degraded. By performing mesh quality improvement with Mesquite [8], the average mesh qualities improved from 1.20431 to 1.18070, from 1.38840 to 1.37350, and from 1.50317 to 1.47862 for the sixth, ninth, twelfth, and sixteenth intermediate meshes, respectively.

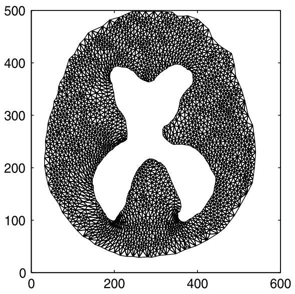
Table 4 shows that the worst mesh qualities of the intermediate meshes also improved by performing mesh quality improvement with [8]. In the case of the sixteenth intermediate mesh, the worst mesh quality was degraded from 14.021 to 19.4508, even though mesh quality improvement was performed via Mesquite [8]. To obtain further mesh quality improvement for the final mesh, mesh quality improvement of the worst quality element was performed by using the PS algorithm



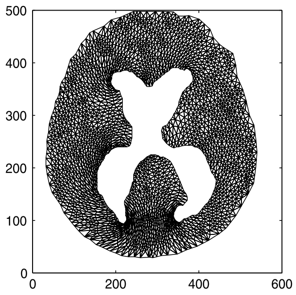
(a) Initial mesh



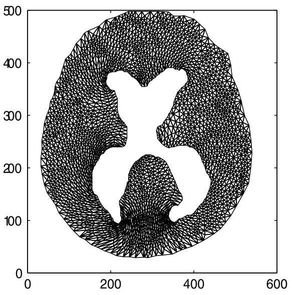
(b) 6th deformation



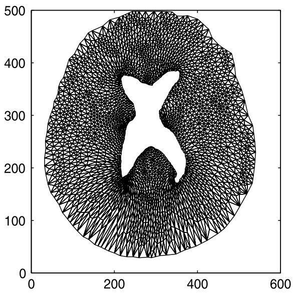
(c) Optimized 6th deformation



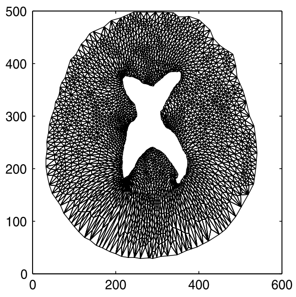
(d) 9th deformation



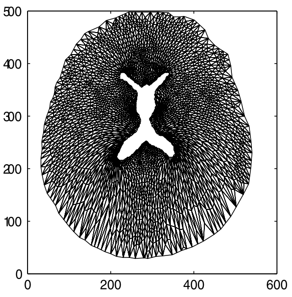
(e) Optimized 9th deformation



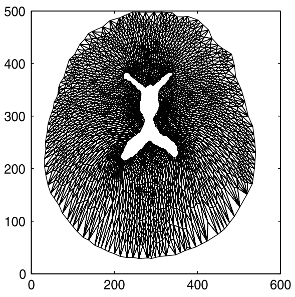
(f) 12th deformation



(g) Optimized 12th deformation



(h) 16th deformation

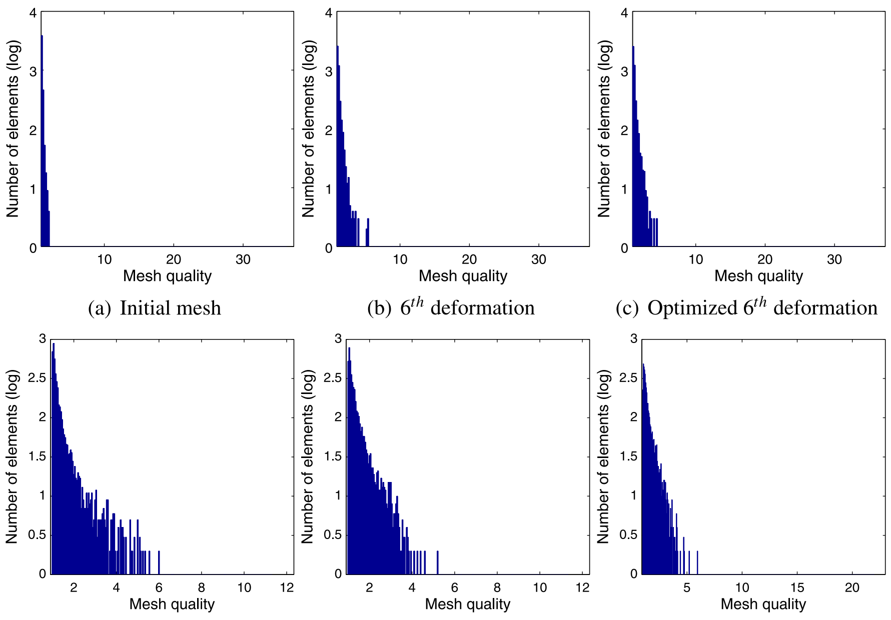


(i) Optimized 16th deformation

Fig. 17 (**a**) Initial mesh generated by Triangle [71] and (**b**), (**d**), (**f**), and (**h**) the deformed meshes generated by FEMWARP [74]. The sixteenth intermediate mesh deformation result matched exactly the boundary vertices of the ventricles for period 2. (**c**), (**e**), (**g**), and (**i**) Mesh quality improvement was performed to improve the quality of the meshes at each intermediate deformation step of ventricular deformation

[57]. The worst mesh quality of the sixteenth intermediate mesh improved from 18.4508 to 8.9290, despite the fact that the average mesh quality of the mesh was slightly degraded to 1.91943.

Figure 18(i) shows that the number of poor mesh quality elements of the sixteenth intermediate mesh increased regardless of performing mesh quality improvement. After the PS algorithm for improving the worst quality element was performed, the number of poor mesh quality elements of the mesh decreased (seen in Fig. 18(j)).



(a) Initial mesh (b) 6th deformation (c) Optimized 6th deformation

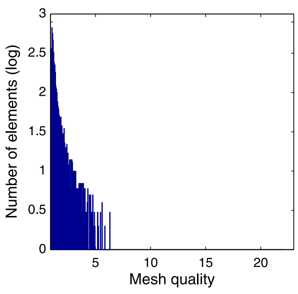
(d) 9th deformation (e) Optimized 9th deformation (f) 12th deformation

Fig. 18 Inverse mean ratio mesh quality distribution for meshes generated in Simulation 3. Quality distribution for the (**a**) initial mesh, (**b**) the sixth intermediate mesh, (**c**) the sixth intermediate mesh after mesh quality improvement was performed, (**d**) the ninth intermediate mesh, (**e**) the ninth intermediate mesh after mesh quality improvement was performed, (**f**) the twelfth intermediate mesh, (**g**) the twelfth intermediate mesh after mesh quality improvement was performed, (**h**) the sixteenth intermediate mesh, (**i**) the sixteenth intermediate mesh after mesh quality improvement was performed, and (**j**) the sixteenth intermediate mesh after the worst quality element improvement was performed

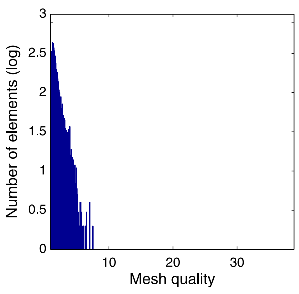
The distances for boundary vertex movement during the deformations are visualized in Fig. 19. The distances were computed by calculating the Euclidean distance between the vertices in each boundary, and not by summing each distance traveled in an intermediate step. The arrows shown in Fig. 19 were generated by connecting two vertices having the same index in each boundary.

When ventricular mesh deformation from pre-treatment to period 1 was performed, the boundary vertices in the lower-left parts of the ventricles moved more than that of the vertices in the other parts of the ventricles. These vertices, which were indexed between 40 to 60, were included in the lower-left parts of the ventricles, and their movement was approximately 80.

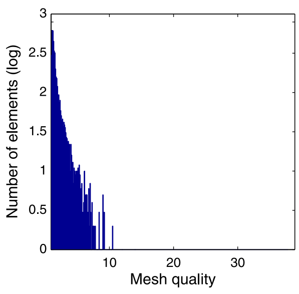
The boundary vertex movement from period 1 to period 2 was significantly large. The size of vectors for most of the vertices were larger than that of the previous deformation from pre-treatment to period 1. As can be seen in Fig. 19(d), the distances for most of the vertices were larger than 50 except the vertices indexed between 120 and 130, whose movement was around 10.



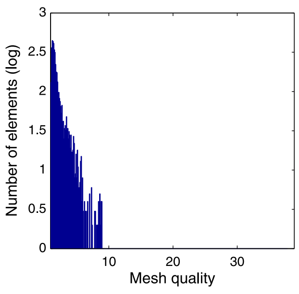
(g) Optimized 12^{th} deformation



(h) 16^{th} deformation



(i) Optimized 16^{th} deformation



(j) Opt. 16^{th} deformation (with PS)

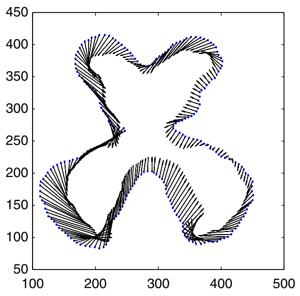
Fig. 18 (Continued)

6 Conclusions and Future Work

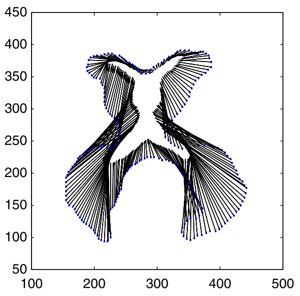
We proposed an image-based computational technique for use in tracking the evolution of the brain ventricles in hydrocephalic patients pre- and post-treatment. Such simulations could be used by neurosurgeons in order to design personalized medical treatments for a given patient in that the simulations could be used to determine how a particular treatment may perform on a given patient. Our image-based computational technique is based on a combination of the level set method and the finite element mesh warping (FEMWARP) method [74]. Our computational pipeline involves the pre-processing steps of image denoising and segmentation. The segmented medical images are then used as input to the combined level set/mesh warping algorithm. Next, the level set method is used to predict the next position of the brain ventricles, and the mesh warping method, i.e., FEMWARP, is used to deform the mesh to the new target. The prediction and deformation steps are performed several times until the final target of the brain ventricles is reached.

Using our approach, we were able to perform three numerical simulations in order to track the evolution of the brain ventricles post-treatment of hydrocephalus via shunt insertion in two patients. In Simulation 1, the ventricular mesh deformation was performed for a case in which the ventricular area decreased while the ventricular shape was preserved. In this case, the ventricular mesh deformation was

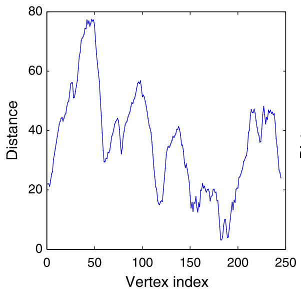
Fig. 19 (**a**) Boundary movement of each vertex from pre-treatment to period 1; (**b**) boundary movement of each vertex from period 1 to period 2, and (**c**) the distance between the pre-treatment and period 1 vertices; (**d**) the distance between the period 1 and period 2 vertices



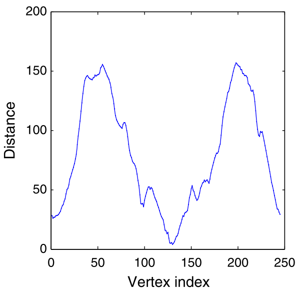
(a) Boundary movement from pre-treatment to period 1



(b) Boundary movement from period 1 to period 2



(c) Distance between pretreatment and period 1 ventricular boundary vertices



(d) Distance between period 1 and period 2 ventricular boundary vertices

easy to perform using FEMWARP, and coupling with the level set method was not necessary.

The brain ventricles changed their shapes asymmetrically during their evolution in Simulation 2. Because the deformation of the ventricles was rather large in this case, intermediate mesh deformation steps were designed, and small-step FEMWARP [74] was used in order to deform the mesh. In total, fifteen intermediate deformation steps were performed to track the evolution of the ventricles from pre- to Period 2 of post-treatment. The ninth intermediate mesh matched the ventricular boundary for period 1; the fifteenth mesh matched the ventricular boundary for period 2.

Finally, the level set method was coupled with FEMWARP in Simulation 3 in order to obtain the ventricular boundary vertices in the segmented medical images. Linear interpolation based on the level sets was performed in order to predict the location of the brain ventricles on the intermediate steps. Mesh deformation was again performed via small-step FEMWARP in combination with the predicted target locations. Sixteen intermediate deformation steps were designed, and the ninth intermediate mesh and the sixteenth mesh matched the ventricular boundary for period 1 and period 2, respectively.

Mesh quality improvement was performed on the deformed meshes from all three simulations in order to improve their overall quality. For the sixteenth deformed mesh in Simulation 3, it was also necessary to prove worst element quality mesh improvement in order to further improve the quality of the mesh to an acceptable level for computational purposes.

Our combined level set/mesh warping technique performed semi-automatic ventricular mesh deformation and evolution of the brain ventricles. Future research will focus on fully automating the level set method in its ability to predict the location of the brain ventricles at the next time step. This will require a technique for determining which speed should be applied to each boundary vertex in the level set contour to prescribe its evolution. We will also extend our approach to handle three-dimensional ventricular evolution. It should be noted that a 3D version of FEMWARP is already in existence [74] and can be used for this purpose. In addition, we will extend the FEMWARP method so that it can handle topological changes during the ventricular deformation; mesh adaptation will need to be added to FEMWARP for this purpose. It should be noted that the level set method can already handle topological changes. Finally, it should be noted that FEMWARP is a geometric mesh warping approach. We plan to extend FEMWARP so that it also incorporates brain biomechanics which can be used to predict the evolution of the brain ventricles.

It should also be mentioned that our combined level set/mesh warping method can also be used to analyze other medical conditions for which medical images are acquired. For example, such an approach could also be used to analyze geometric changes in the brain due to normal brain growth or the growth of a tumor, the change in the brain due to a stroke, or the impact of a traumatic injury on the brain. Our approach can also be applied to other medical conditions involving medical images and deformations including several applications in cardiology or orthopaedics.

Acknowledgements The work of the first two authors was funded in part by NSF CAREER Award OCI-1054459; the work of the second author was also funded in part by NSF grant CNS-0720749.

References

1. Adalsteinsson D, Sethian JA (1995) A fast level set method for propagating interfaces. J Comput Phys 118:269–277
2. Amato U, Larobina M, Antoniadas A, Alfano B (2003) Segmentation of magnetic resonance images through discriminant analysis. J Neurosci Methods 131:65–74
3. Bah MT, Nair PB, Browne M (2009) Mesh morphing for finite element analysis of implant positioning in cementless total hip replacement. Med Eng Phys 31:1235–1243
4. Baker T (2001) Mesh movement and metamorphosis. In: Proc. of the 10th international meshing roundtable, Sandia National Laboratories, pp 387–396
5. Baldwin MA, Langenderfer JE, Rullkoetter PJ, Laz PJ (2010) Development of subject-specific and statistical shape models of the knee using an efficient segmentation and mesh-morphing approach. Comput Methods Programs Biomed 97:232–240
6. Bezdek JC, Hall LO, Clarke LP (1993) Review of MR segmentation techniques and using pattern recognition. Med Phys 20:1033–1048
7. Biot M (1941) General theory of three-dimensional consolidation. J Appl Phys 12:155–164

8. Brewer M, Diachin L, Knupp P, Leurent T, Melander D (2003) The Mesquite mesh quality improvement toolkit. In: Proc. of the 12th international meshing roundtable, Sandia National Laboratories, pp 239–250
9. Bridson R, Teran J, Molino N, Fedkiw R (2005) Adaptive physics based tetrahedral mesh generation using level sets. Eng Comput 21:2–18
10. Chan TF, Vese LA (2001) Active contours without edges. IEEE Trans Image Process 10:266–277
11. Choi JU, Kim DS, Kim SH (1999) Endoscopic surgery for obstructive hydrocephalus. Yonsei Med J 40:600–607
12. de Putter S, Laffargue F, Breeuwer M, van de Vosse FN, Gerritsen FA (2006) Computational mesh generation for vascular structures with deformable surfaces. Int J Comput Assisted Radiol Surg 1:39–49
13. Di Rocco C, Massimi L, Tamburrini G (2006) Shunts vs. endoscopic third ventriculostomy in infants: are there different types and/or rates of complications? A review. Child's Nerv Syst 22:1573–1589
14. Drapaca CS, Tenti G, Rohlf K, Sivaloganathan S (2006) A quasilinear viscoelastic constitutive equation for the brain: application to hydrocephalus. J Elast 85:65–83
15. Dyedov V, Einstein DR, Jiao X, Kuprat AP, Carsona JP, del Pin F (2009) Variational generation of prismatic boundary-layer meshes for biomedical computing. Int J Numer Methods Eng 79:907–945
16. Egnor M, Rosiello A, Zheng L (2001) A model of intracranial pulsations. Pediatr Neurosurg 35:284–298
17. Evans LC, Spruck J (1991) Motion of level sets by mean curvature. J Differ Geom 33:635–681
18. Fillinger M, Raghavan M, Marra S, Cronenwett J, Kennedy F (2002) In vivo analysis of mechanical wall stress and abdominal aortic aneurysm risk. J Vasc Surg 26:589–597
19. Fischl B, Salat DH, Busa E, Albert M, Dietrich M, Haselgrove C, van der Kouwe A, Killiany R, Kennedy D, Klaveness S, Montillo A, Makris N, Rosen B, Dale AM (2002) Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. Neuron 33:341–355
20. Frey P (2004) Generation and adaptation of computational surface meshes from discrete anatomical data. Int J Numer Methods Eng 60:1049–1074
21. Fritz JS, Drapaca CS (2009) A study of the effect of the pulsatile ventricular pressure in the development of hydrocephalus. In: Proceedings of the 3rd international conference in computational mechanics and virtual engineering COMEC2009, vol 1, pp 254–259
22. Gonzalez RC, Woods RE (1992) Digital image processing. Addison-Wesley, Reading
23. Hakim S, Venegas J, Burton J (1976) The physics of the cranial cavity, hydrocephalus and normal pressure hydrocephalus: mechanical interpretation and mathematical model. Surg Neurol 5:187–210
24. Haralick RM, Shapiro LG (1985) Image segmentation techniques. Comput Vis Graph Image Process 29:100–132
25. Held K, Rota Kops E, Krause BJ, Wells III WM, Kikinis R, Muller-Gartner HW (1997) Markov random field segmentation of brain MR images. IEEE Trans Med Imaging 16:878–886
26. Hilton A, Illingworth J (1997) Marching triangles: Delaunay implicit surface triangulation. Technical report, University of Surrey
27. Hojjatoleslami SA, Kittler J (1998) Region growing: a new approach. IEEE Trans Image Process 7:1079–1084
28. Hojjatoleslami SA, Kruggel F (2001) Segmentation of large brain lesions. IEEE Trans Med Imaging 20:666–669
29. Hydrocephalus statistics. http://www.ghrforg.org/faq.htm
30. Ito Y, Shum PC, Shih AM, Soni BK, Nakahashi K (2006) Robust generation of high-quality unstructured meshes on realistic biomedical geometry. Int J Numer Methods Eng 65:943–973
31. Johnson C (1987) Numerical solutions of partial differential equations by the finite element method. Studentlitteratur, Lund

32. Kaczmarek M, Subramaniam R, Neff S (1997) The hydromechanics of hydrocephalus: steady-state solutions for cylindrical geometry. Bull Math Biol 59:295–323
33. Kass M, Witkin A, Terzopoulos D (1988) Snakes: active contour models. Int J Comput Vis 1:321–331
34. Kellie G (1824) Appearances observed in the dissection of two individuals; death from cold and congestion of the brain. Trans Med-Chir Soc Edinburgh 1:1–84
35. Langan DA, Modestino JW, Zhang J (1998) Cluster validation for unsupervised stochastic model-based image segmentation. IEEE Trans Image Process 7:180–195
36. Liu Y, D'Arceuil H, He J, Duggan M, Gonzalez G, Pryor J, de Crespigny A (2006) A nonlinear mesh-warping technique for correcting brain deformation after stroke. Magn Reson Imaging 24:1069–1075
37. Liu Y, Foteinos P, Chernikov A, Chrisochoides N (2010) Multi-tissue mesh generation for brain images. In: Proc. of the 19th international meshing roundtable, pp 367–384
38. Lohner R (1996) Regridding surface triangulations. J Comput Phys 126:1–10
39. Lorensen W, Cline H (1987) Marching cubes: a high resolution 3D surface construction method. Comput Graph 21:163–169
40. Ma Z, Tavares JM, Jorge RN, Mascarenhas T (2010) A review of algorithms for medical image segmentation and their applications to the female pelvic cavity. Comput Methods Biomech Biomed Eng 13:235–246
41. Magnotta VA, Heckel D, Andreasen AC, Cizadlo T, Corson PW, Ehrhardt JC, Yuh WTC (1999) Measurement of brain structures with artificial neural networks: two- and three-dimensional applications. Radiology 211:781–790
42. Marmarou A, Shulman K, Rosende R (1978) A nonlinear analysis of the cerebrospinal fluid and intracranial pressure dynamics. J Neurosurg 48:530–537
43. McInerey T, Tarzopoulos D (1996) Deformable medical image analysis: a survey. Med Image Anal 1:91–108
44. Mendis K, Stalnaker R, Advani S (1995) A constitutive relationship for large deformation finite-element modeling of brain-tissue. J Biomech Eng 117:279–285
45. Miller K (1999) Constitutive model of brain tissue suitable for finite element analysis of surgical procedures. J Biomech 32:531–537
46. Miller K, Chinzei K (1997) Constitutive modeling of brain tissue: experiment and theory. J Biomech 30:1115–1121
47. Miller K, Taylor Z, Wittek A (2006) Mathematical models of brain deformation behaviour for computer-integrated neurosurgery. Technical Report ISML/01/2006, The University of Western Australia
48. Mitich A, Ayed IB (2010) Variational and level set methods in image segmentation. Springer topics in signal processing, vol 5. Springer, Berlin
49. Monro A (1783) Observations on structure and functions of the nervous system. Creech and Johnson, Edinburgh
50. Munson T (2007) Mesh shape-quality optimization using the inverse mean-ratio metric. Math Program 110:561–590
51. Nagashima T, Tamaki N, Matsumoto S, Horwitz B, Seguchi Y (1987) Biomechanics of hydrocephalus: a new theoretical model. Neurosurgery 21:898–904
52. Osher S, Fedkiw RP (2001) Level set methods: an overview and some recent results. J Comput Phys 169:463–502
53. Osher S, Fedkiw R (2003) Level set methods and dynamic implicit surfaces. Applied mathematical sciences, vol 153. Springer, Berlin
54. Osher S, Paragois N (2003) Geometric level set methods in imaging, vision, and graphics. Springer, Berlin
55. Osher S, Sethian JA (1988) Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. J Comput Phys 79:12–49
56. Pal NR, Pal SK (1993) A review on image segmentation techniques. Pattern Recognit 26:1277–1294

57. Park J, Shontz SM (2010) Two derivative-free optimization algorithms for mesh quality improvement. In: Proc. of the 2010 international conference on computational science, Amsterdam, Netherlands, pp 387–396
58. Peiro J, Giordana S, Griffith C, Sherwin S (2002) High-order algorithms for vascular flow modelling. Int J Numer Methods Fluids 40:137–151
59. Pena A, Harris N, Bolton M, Czosnyka M, Pickard J (2002) Communicating hydrocephalus: the biomechanics of progressive ventricular enlargement revisited. Acta Neurochir 81:59–63
60. Peng D, Merriman B, Osher S, Zhao H, Kang M (1999) A PDE-based fast local level set method. J Comput Phys 155:410–438
61. Persson P (2004) Mesh generation for implicit geometries. PhD thesis, MIT
62. Persson P (2006) Mesh size functions for implicit geometries and PDE-based gradient limiting. Eng Comput 22:95–109
63. Persson P, Strang G (2004) A simple mesh generator in MATLAB. SIAM Rev 46:329–345
64. Pham DL, Xu C, Prince JL (2000) Current methods in medical image segmentation. Annu Rev Biomed Eng 2:315–337
65. Quatember B, Muhlthaler H (2003) Generation of CFD meshes from biplane angiograms: an example of image-based mesh generation and simulation. Appl Numer Math 46:379–397
66. Sahoo PK, Soltani S, Wong AKC (1988) A survey of thresholding techniques. Comput Vis Graph Image Process 41:233–260
67. Schmidt J, Johnson C, Eason J, McLeod R (1994) Applications of automatic mesh generation and adaptive methods in computational medicine. Springer, Berlin
68. Sethian JA (1999) Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge monographs on applied and computational mathematics
69. Shattuck DW, Mizra M, Adestiyo V, Hojatkashani C, Salamon G, Narr KL, Poldrack RA, Bilder RM, Toga AW (2008) Construction of a 3D probabilistic atlas of human cortical structures. NeuroImage 39:1064–1080
70. Shepherd JF, Johnson CR (2009) Hexahedral mesh generation for biomedical models in SCIRun. Eng Comput 25:97–114
71. Shewchuk J (1996) Triangle: engineering a 2D quality mesh generator and Delaunay triangular. In: Applied computational geometry: towards geometric engineering. Lecture notes in computer science, vol 1148. Springer, Berlin, pp 203–222
72. Shewchuk J (2003) What is a good linear element? Interpolation, conditioning, and quality measures. In: Proc. of the 11th international meshing roundtable, Sandia National Laboratories, pp 115–126
73. Shontz SM (2005) Numerical methods for problems with moving meshes. PhD thesis, Cornell University
74. Shontz SM, Vavasis SA (2010) Analysis of and workarounds for element reversal for a finite element-based algorithm for warping triangular and tetrahedral meshes. BIT Numer Math 50:863–884
75. Sigal IA, Whyne CM (2010) Mesh morphing and response surface analysis: quantifying sensitivity of vertebral mechanical behavior. Ann Biomed Eng 38:41–56
76. Sigal IA, Hardisty MR, Whyne CM (2008) Mesh-morphing algorithms for specimen-specific finite element modeling. J Biomech 41:1381–1389
77. Sigal IA, Yang H, Roberts MD, Downs JC (2010) Morphing methods to parameterize specimen specific finite element model geometries. J Biomech 43:254–262
78. Sivaloganathan S, Tenti G, Drake J (1998) Mathematical pressure volume models of the cerebrospinal fluid. Appl Math Comput 94:243–266
79. Smillie A, Sobey I, Molnar Z (2005) A hydroelastic model of hydrocephalus. J Fluid Mech 539:417–443
80. Strang G, Persson P (2004) Circuit simulation and moving mesh generation. In: Proc. of international symposium on communications and information technologies 2004 (ISCIT 2004)
81. Szczerba D, McGregor R, Szekely G (2007) High quality surface mesh generation for multiphysics bio-medical simulations. In: Proc. of the 2007 international conference on computa-

tional science. Lecture notes in computer science, vol 4487. Springer, Berlin, pp 906–913
82. Tenti G, Sivaloganathan S, Drake J (1999) Brain biomechanics: steady-state consolidation theory of hydrocephalus. Can Appl Math Q 7:111–124
83. Treece GM, Prager RW, Gee AH (1999) Regularised marching tetrahedra: improved iso-surface extraction. Comput Graph 23:583–598
84. Tuli S, Alshail E, Drake J (1999) Third ventriculostomy versus cerebrospinal fluid shunt as a first procedure in pediatric hydrocephalus. Pediatr Neurosurg 30:11–15
85. Tully B, Ventikos Y (2009) Coupling poroelasticity and CFD for cerebrospinal fluid hydrodynamics. IEEE Trans Biomed Eng 56:1644–1651
86. Ulrich D, van Rietbergen B, Weinans H, Ruegsegger P (1998) Finite element analysis of trabecular bone structure: a comparison of image-based meshing techniques. J Biomech 31:1187–1192
87. Verma CS, Fischer PF, Lee SE, Loth F (2005) An all-hex meshing strategy for bifurcation geometries in vascular flow simulation. In: Proc. of the 14th international meshing roundtable, Sandia National Laboratories, pp 11–14
88. Vese LA, Chan T (2002) A multiphase level set framework for image segmentation using the Mumford and Shah model. Int J Comput Vis 50:271–293
89. Wang H, Wineman A (1972) A mathematical model for the determination of viscoelastic behavior of brain in vivo. I. Oscillatory response. J Biomech 5:31–46
90. West JJ (2004) Application of the level set method to hydrocephalus: simulating the motion of the ventricles. Master's thesis, University of Waterloo
91. Wilkie K, Drapaca CS, Sivaloganathan S (2010) A theoretical study of the effect of ventricular pressure pulsations on the pathogenesis of hydrocephalus. Appl Math Comput 215:3181–3191
92. Withey DJ, Koles ZJ (2008) A review of medical image segmentation: methods and available software. Int J Bioelectromagn 10:125–148
93. Wu Y (2011) Matlab implementation of the Chan Vese active contour without edges method. http://www.mathworks.com/matlabcentral/fileexchange/23445-chan-vese-active-contours-without-edges
94. Yamakawa S, Shimada K (2008) Converting a tetrahedral mesh to a prism-tetrahedral hybrid mesh for FEM accuracy and efficiency. In: Proc. of the 2008 ACM symposium on solid and physical modeling, pp 287–294
95. Yu Z, Holst MJ, McCammon JA (2008) High-fidelity geometric modeling for biomedical applications. Finite Elem Anal Des 44:715–723
96. Zachariah SG, Sanders JE, Turkiyyah GM (1996) Automated hexahedral mesh generation from biomedical image data: applications in limb prosthetics. IEEE Trans Rehabil Eng 4:91–102

An Optimization-Based Iterative Approach to Tetrahedral Mesh Smoothing

Zhanheng Gao, Zeyun Yu, and Jun Wang

Abstract The optimal Delaunay triangulation (ODT) is an effective approach in improving the quality of inner vertices of a tetrahedral mesh. Recently it had been extended boundary-optimized Delaunay triangulation (B-ODT), in which both inner and boundary vertices are repositioned by analytically minimizing the $\mathcal{L}^1$ error between a paraboloid function and its piecewise linear interpolation over the neighborhood of each vertex. In the present work, we describe a smoothing method that is based on the B-ODT method but has better performance. We smooth the mesh in an edge-by-edge fashion by adjusting each pair of vertices of every edge. This method has the volume-preserving and sharp-feature-preserving properties. A number of experiments are included to demonstrate the performance of our method.

1 Introduction

The finite element method (FEM) has been a very popular numerical approach for solving partial differential equations (PDEs) in many applications. In the method, the domain over which the PDEs are defined is partitioned into a mesh containing a large number of simple elements, such as triangles and quadrilaterals in 2D cases and tetrahedra and hexahedra in 3D cases [1–6]. The quality of the mesh, typically measured by the minimum and maximum angles, can significantly affect the interpolation accuracy and solution stability of the FEA [7, 8]. Therefore, improving the mesh quality has been an active research area in computational mathematics and computer science. Due to the great popularity in the FEM, 3D tetrahedral meshes will be the focus of our present work.

The methods of mesh quality improvement can be classified into three categories as follows. (1) topology optimization, which modifies the connectivity be-

Z. Gao · Z. Yu (✉) · J. Wang
Department of Computer Science, University of Wisconsin-Milwaukee, 3200 N. Cramer St, Milwaukee, WI 53211, USA
e-mail: yuz@uwm.edu

Z. Gao
College of Computer Science and Technology, Jilin University, 2699 Qianjin St, Changchun, Jilin 130012, China

tween mesh vertices while keeping vertex positions unchanged. The edge- or face-swapping methods are commonly used in topology optimization [9, 10]. (2) vertex insertion/deletion, which inserts/deletes vertices to/from the mesh [10–13]. (3) vertex smoothing, which repositions the coordinates of the vertices while keeping the connectivity unchanged [14–16]. Generally speaking, mesh quality improvement is best achieved when all the three methods are properly combined in the mesh smoothing scheme [10]. In our method described below, we shall focus on the vertex repositioning strategy, i.e. vertex smoothing.

One of the most popular vertex smoothing method is *Laplacian* smoothing, which moves a mesh vertex to the weighted average of its incident vertices [17–19]. If the neighborhood of the vertex is not a convex polyhedron, the Laplacian smoothing may not lead to a well-positioned mesh. Some angle-based methods were proposed for smoothing 2D triangular and 3D surface meshes [20–22]. However, these methods are difficult to extend to 3D tetrahedral meshes. [23] presented a method based on the Centroid Voronoi Tessellation (CVT) concept that is restricted to inner vertices of a mesh. A peeling off operation has to be taken to improve bad tetrahedra on boundaries. [24] proposed a method of smoothing planar quadrilateral meshes. Some researchers presented methods for smoothing hexahedral mesh [25–29]. More recently, some new techniques of vertex smoothing were proposed. [30, 31] presented methods of stretching the vertices of a tetrahedron at one time. The methods were extended by [32] to hexahedral mesh. [33] assigned a quality coordinate for every vertex and calculated the new position by maximizing the combined quality of tetrahedra incident to it. [34] used a metric non-conformity driven method to smooth hybrid meshes such as a mesh with hexahedral and tetrahedral elements.

In addition to the above methods, approaches using numerical optimization to compute the new position of a vertex has been an important branch of the vertex smoothing category. The new position of a vertex is computed by optimizing a function that measures the local or global quality of the mesh [35–44]. In particular, the optimal Delaunay triangulation (ODT) approach [45] tries to minimize the $\mathcal{L}^1$ error between a paraboloid function and its piecewise linear interpolation over the neighborhood of a vertex. This idea has been extended to 3D tetrahedral mesh smoothing in [46]. Despite its great success in mesh quality improvement, the original ODT method was derived to optimize the positions of inner vertices only. In other words, the tetrahedral mesh to be smoothed must possess quality triangles on boundaries. In many real mesh models, however, "bad" tetrahedra often occur near or on the boundaries of a domain [47, 48]. Therefore, in our previous work, we provided an analytical method named boundary-optimized Delaunay triangulation (B-ODT) to find the optimal positions of all mesh vertices, including those on boundaries, by minimizing an $\mathcal{L}^1$ error function that is defined in the incident neighborhood of each vertex. The minimization is an unconstrained quadratic optimization problem and has an exact analytic solution when the coefficient matrix of the problem is positive definite.

In this work, we extend our previous B-ODT method by performing it edge by edge. The new method achieves better results than the original B-ODT method by considering the local configuration of every vertex before performing the B-ODT

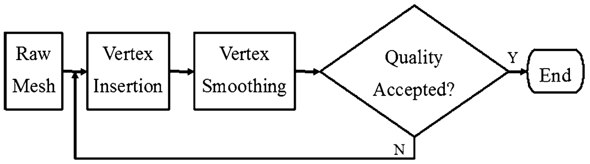


Fig. 1 The framework of our mesh quality improvement method

algorithm. The remainder of the present work is organized as follows. In Sect. 2, we start with a brief introduction to our tetrahedral mesh generation from an initial triangular surface mesh. We then review the ODT and B-ODT methods, provide the edge-based B-ODT (so-called eB-ODT) method. We present some experimental results and quality analysis in Sect. 3, followed by our conclusions in Sect. 4.

2 Methods

The framework of our mesh quality improvement method is shown in Fig. 1. The vertex insertion operation is performed prior to the vertex repositioning. We try to insert as few vertices as possible in order to maintain the size of the original mesh. We give a brief introduction to our tetrahedral mesh generation using an octree-based method in Sect. 2.1. As for vertex smoothing, the algorithms of original B-ODT and eB-ODT are given in Sects. 2.2 and 2.3 respectively.

2.1 Tetrahedral Mesh Generation Algorithm

Our tetrahedral mesh generation algorithm is based on the body centered cubic (BCC) tetrahedral lattice, a common crystal structure in nature with many desirable properties [49]. The BCC lattice is constructed by adding a new node at each cell center and connecting it to the eight vertices of the cell and six neighboring cell centers. The BCC lattice is highly structured and computationally efficient, and has been utilized in various types of numerical simulation. When dealing with a bounded domain, however, the BCC lattice must be carefully remeshed near the domain boundary so that the tetrahedral mesh generated agrees with the given boundary. To this end, our method consists of the following four steps (see Fig. 2 for a two dimensional illustration):

1. Subdivide the octree of an input surface mesh based on Euclidean distance transformation. A few geometric properties of the input mesh are utilized to refine the subdivision adaptively from interior to boundary, and from low curvature to high curvature areas.
2. Compute the sign of every node in the BCC lattice. For each edge of the BCC grid, if the corresponding signs of the two endpoints are different, then calculate the cutting (intersecting) point where the edge crosses the input surface mesh.

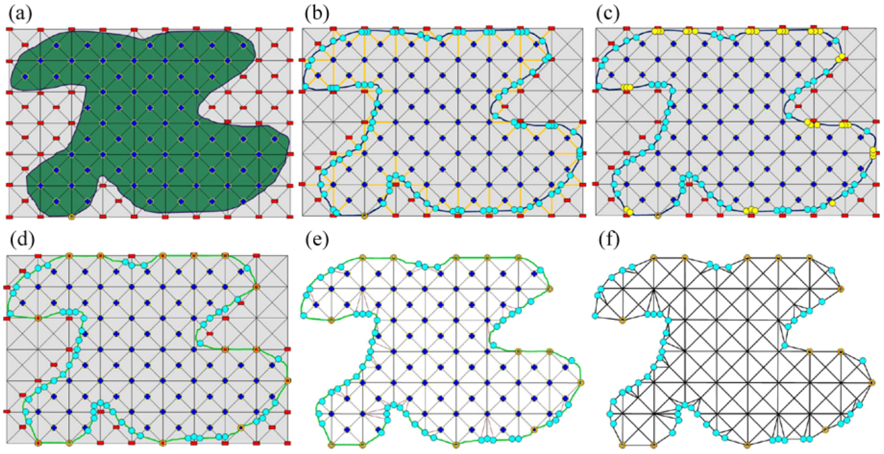


Fig. 2 A two dimensional illustration of our tetrahedral generation algorithm. Note that the octree subdivision is adaptive in our algorithm. However, we do not show the adaptivity here for simplicity. (**a**) Computing the signs for each BCC grid; (**b**) Calculating the cutting points; (**c**) Detecting the "too close" cutting points; (**d**) Snapping the "too close" cutting points to the corresponding BCC lattice grids; (**e**) Decomposing the boundary polyhedra into tetrahedra; (**f**) Obtaining the final tetrahedral mesh

3. Detect the cutting points that are "too close" to the original BCC nodes and snap them to the corresponding nodes. Equivalently, we adjust the sign of that node to zero. We refer to this process as cutting point snapping.
4. Decompose the boundary polyhedra into tetrahedra. For each BCC tetrahedron, if all signs of its vertices are negative (meaning "outside"), we ignore it (we assume that only the interior tetrahedralization is of interest). If all signs are positive (meaning "inside"), we leave it as the final tetrahedron. Otherwise, the tetrahedron is split by the input surface mesh into inside and outside parts and we further decompose the inside part (a polyhedron) into tetrahedra.

2.2 ODT and B-ODT Algorithms

For any vertex $\mathbf{x}_0$ in a tetrahedral mesh $\mathcal{T}$, suppose the neighborhood of $\mathbf{x}_0$ is Ω_0 consisting of a set of tetrahedra $\{\tau\}$. Let $\mathbf{x}_*$ be the smoothing result of $\mathbf{x}_0$ and Ω_* the neighborhood of $\mathbf{x}_*$ (or the union of tetrahedra incident to $\mathbf{x}_*$) in $\mathcal{T}$.

If $\mathbf{x}_0$ is an inner vertex, $\mathbf{x}_*$ can be computed by the following ODT formula [45]:

$$\mathbf{x}_* = \mathbf{x}_0 - \frac{1}{2|\Omega_0|} \sum_{\tau \in \Omega_0} \left(\frac{1}{3} S_\tau \mathbf{n}_\tau \sum_{i=1}^{3} \|\mathbf{x}_{\tau,i} - \mathbf{x}_0\|^2 \right). \tag{1}$$

Here S_τ and $\mathbf{n}_\tau$ are the area and unit normal vector of t_τ, which is the opposite triangle of $\mathbf{x}_0$ in τ, $\mathbf{n}_\tau$ points to the inside of τ, $\mathbf{x}_{\tau,i}$ are the (three) vertices of t_τ.

If $\mathbf{x}_0$ is a boundary vertex, $\mathbf{x}_*$ can be computed by the following B-ODT formula:

$$\mathbf{x}_* = \mathbf{x}_0 + u\mathbf{s} + v\mathbf{t} \tag{2}$$

where $\mathbf{s}$ and $\mathbf{t}$ are two orthonormal vectors in the tangent plane of the boundary surface of $\mathcal{T}$ at $\mathbf{x}_0$, the coefficients u and v are computed by solving the following linear equation system:

$$\begin{bmatrix} 2E & G \\ G & 2F \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -H \\ -I \end{bmatrix}. \tag{3}$$

The calculations of E, F, G, H, I are given below in Algorithm 1. Here, we directly give the algorithm of smoothing inner and boundary vertices of $\mathcal{T}$ using ODT and B-ODT methods respectively (Algorithm 1). Note that the tangent plane restriction of $\mathbf{x}_*$ guarantees the volume of the smoothed mesh coincides with that of the original mesh.

Theoretically, both (1) and (2) are the unique solutions of the optimization problem which minimizes the $\mathcal{L}^1$ interpolation error between a paraboloid function $f_I(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_0\|^2$ and its piecewise linear interpolation over Ω_*:

$$Error_* = \|f - f_I\|_{L^1} = \int_{\mathbf{x}\in\Omega_*} \left|f(\mathbf{x}) - f_I(\mathbf{x})\right| \mathrm{d}\mathbf{x}. \tag{4}$$

Algorithm 1 (ODT and B-ODT smoothing for inner and boundary vertices)

```
for every vertex $\mathbf{x}_0$ do
  a. if $\mathbf{x}_0$ is an inner vertex, then
       $\mathbf{x}_*$ is computed by (1).
  b. else
       $\mathbf{x}_*$ is computed using the following scheme:
       i. Compute the normal vector of the tangent plane at $\mathbf{x}_0$, then select
          two orthogonal unit vectors $\mathbf{s}$, $\mathbf{t}$ on the tangent plane.
       ii. Compute the following coefficients:
          A. $E = \frac{1}{4}|\Omega_0| - \frac{1}{60}\sum_{i=1}^{m} \mathbf{s}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{s}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
          B. $F = \frac{1}{4}|\Omega_0| - \frac{1}{60}\sum_{i=1}^{m} \mathbf{t}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{t}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
          C. $G = -\frac{1}{60}\sum_{i=1}^{m}[\mathbf{s}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{t}(\mathbf{Y}_i \times \mathbf{Y}_{i+1}) + \mathbf{t}(\mathbf{Y}_i + \mathbf{Y}_{i+1}) \times \mathbf{s}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})]$
          D. $H = \frac{1}{12}\mathbf{s}\sum_{\tau\in\Omega_*} S_\tau \mathbf{n}_\tau L_\tau - \frac{1}{60}\sum_{i=1}^{m}(\mathbf{Y}_i^2 + \mathbf{Y}_{i+1}^2 + \mathbf{Y}_i\mathbf{Y}_{i+1}) \times \mathbf{s}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
          E. $I = \frac{1}{12}\mathbf{t}\sum_{\tau\in\Omega_*} S_\tau \mathbf{n}_\tau L_\tau - \frac{1}{60}\sum_{i=1}^{m}(\mathbf{Y}_i^2 + \mathbf{Y}_{i+1}^2 + \mathbf{Y}_i\mathbf{Y}_{i+1})\mathbf{t}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
       iii. Solve the linear system (3).
       iv. Compute $\mathbf{x}_*$ using $\mathbf{x}_* = \mathbf{x}_0 + u\mathbf{s} + v\mathbf{t}$.
```

Here, $\mathbf{Y}_i = \mathbf{y}_i - \mathbf{x}_0$, $\{\mathbf{y}_i\}_{i=1}^{m}$ are the neighboring vertices of $\mathbf{x}_0$ on the boundary of the tetrahedral mesh $\mathcal{T}$. The order of $\mathbf{y}_i$ is determined in the following way: for any $i = 1, \dots, m$, the cross product between $\overrightarrow{\mathbf{x}_0\mathbf{y}_i}$ and $\overrightarrow{\mathbf{x}_0\mathbf{y}_{i+1}}$ points to the outside of Ω_0 (let $\mathbf{y}_{m+1} = \mathbf{y}_1$), $L_\tau = \sum_{j=1}^{3} \|\mathbf{x}_{\tau,j} - \mathbf{x}_0\|^2$.

For a boundary vertex $\mathbf{x}_0$, in order to preserve the sharp features, we further restrict $\mathbf{x}_*$ moving along the features of the mesh. Here, we refer to the feature direction at $\mathbf{x}_0$ as the line that passes through $\mathbf{x}_0$ and has the minimal curvature value among all the directions. This line is on the tangent plane, thus the volume is still preserved when $\mathbf{x}_*$ moves along this feature line. The direction of the feature line is found by computing the eigenvalues of the following tensor voting matrix at $\mathbf{x}_0$:

$$M = \sum_{i=1}^{m} S_i \mathbf{n}_i \mathbf{n}_i^T. \tag{5}$$

Here S_i is the area of surface triangle $\Delta\mathbf{x}_0\mathbf{y}_i\mathbf{y}_{i+1}$ and $\mathbf{n}_i = (n_{ix}, n_{iy}, n_{iz})^T$ is the unit normal vector of $\Delta\mathbf{x}_0\mathbf{y}_i\mathbf{y}_{i+1}$. The matrix M is a positive definite matrix and has three orthogonal eigenvectors. The feature line is determined in the following way. Suppose that the three eigenvalues of M are μ_0, μ_1, μ_2 with $\mu_0 \geq \mu_1 \geq \mu_2$ and $\mathbf{e}_0$, $\mathbf{e}_1$, $\mathbf{e}_2$ are the corresponding eigenvectors. If $\mu_0 \gg \mu_1 \approx \mu_2 \approx 0$, then the neighborhood of $\mathbf{x}_0$ corresponds to a planar feature. In this case, the above Algorithm 1 is used to smooth $\mathbf{x}_0$. If $\mu_0 \approx \mu_1 \gg \mu_2 \approx 0$, then $\mathbf{x}_0$ lies on an crease (linear) feature and the direction of the crease is $\mathbf{e}_2$. In this case, the following Algorithm 2 is used to smooth $\mathbf{x}_0$. If $\mu_0 \approx \mu_1 \approx \mu_2 \gg 0$, then $\mathbf{x}_0$ is at a corner which should not be changed during the vertex smoothing process.

Algorithm 2 (B-ODT smoothing with feature preserving)

```
for every crease vertex $\mathbf{x}_0$ do
    a. Set the feature direction at $\mathbf{x}_0$ to be $\mathbf{d} = \mathbf{e}_2/\|\mathbf{e}_2\|$.
    b. Compute the following coefficients:
        i. $A = \frac{1}{4}|\Omega_0| - \frac{1}{60}\sum_{i=1}^{m} \mathbf{d}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{d}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
        ii. $B = \frac{1}{12}\mathbf{d}(\sum_{\tau\in\Omega_*} S_\tau \mathbf{n}_\tau L_\tau) - \frac{1}{60}\sum_{i=1}^{m}(\mathbf{Y}_i^2 + \mathbf{Y}_{i+1}^2 + \mathbf{Y}_i\mathbf{Y}_{i+1})\mathbf{d}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
    c. Compute $\mathbf{x}_*$ as $\mathbf{x}_* = \mathbf{x}_0 + f\mathbf{d}$ with $f = -\frac{B}{2A}$.
```

2.3 Edge-Based B-ODT Algorithm

In practice, the improvement of $\mathbf{x}_0$ is always affected by the configuration of the vertices around $\mathbf{x}_0$. When the vertices around $\mathbf{x}_0$ has good configuration, the quality can be significantly improved. Based on this observation, we presented a modified strategy here to smooth a tetrahedral mesh: smoothing the mesh in an edge-by-edge way. That is, to smooth the two end vertices of each edge recursively. By this way, the angle quality can be improved more than simply performing Algorithm 2. The detail is given in the following algorithm:

Algorithm 3 (eB-ODT smoothing for boundary vertices)

```
for every edge e in the mesh (Let $\mathbf{x}_0$ and $\mathbf{x}_1$ be the two vertices of e), do
    a. Smooth $\mathbf{x}_0$ using Algorithm 1 or Algorithm 2 according to the type of $\mathbf{x}_0$
    b. Smooth $\mathbf{x}_1$ using Algorithm 1 or Algorithm 2 according to the type of $\mathbf{x}_1$
    c. Compute $s$, which is the sum of the movements of $\mathbf{x}_0$ and $\mathbf{x}_1$
    d. If $s \le \varepsilon$, go to next edge
       else, go to step a
```

3 Results

The proposed eB-ODT algorithms were tested on several tetrahedral meshes generated from triangular surface meshes that serve as the boundaries of the domains. For every mesh, the smoothing process shown in Fig. 1 is repeated for 20 times. The mesh smoothing results are summarized in Table 1. The comparisons between the eB-ODT algorithm (Algorithm 3) and several other approaches, including the ODT algorithm, B-ODT algorithm, topology optimization and the Natural ODT algorithm [46], are also provided in Table 1. In Figs. 3–9, the original and smoothed meshes are compared and from the histograms we can see significant improvement of dihedral angles in these meshes.

We compare the smoothing results by using the ODT, B-ODT and eB-ODT algorithms. In Table 1, all the minimum and maximum dihedral angles by using the B-ODT algorithm are better than those by the ODT algorithm and the results by using eB-ODT are better than B-ODT, especially on the Retinal model. Note that the minimum dihedral angle in Retinal model is very small and likely occurs on the boundary of the model. Therefore, the B-ODT and eB-ODT algorithm can perform much better than the original ODT method.

Although the topology optimization is utilized in many mesh smoothing algorithms, this technique alone may not always improve the quality of a mesh. To show this, we smooth all the meshes in Table 1 using only the topology optimization and compare the results with those obtained by using our eB-ODT algorithm. From Table 1, we can see that the ability of improving mesh quality by using topology optimization alone is limited, compared to the eB-ODT algorithm.

The tetrahedral mesh in Fig. 3 is generated by tetrahedralizing randomly-sampled point set on a unit sphere [50]. There are 642 points on the sphere and 87 inner vertices are inserted by the tetrahedralization algorithm. The minimum and maximum

Table 1 Comparisons of dihedral angles using different methods

Model	Algorithm	Vertex number	min	max
Random Sphere	Original mesh	729	5.86°	164.70°
	eB-ODT	767	18.20°	145.56°
	B-ODT	731	15.20°	150.25°
	ODT	729	6.28°	162.46°
	Topology optimization	729	5.86°	164.70°
	NODT	729	6.21°	173.64°
2Cmp	Original mesh	10415	5.57°	163.24°
	eB-ODT	10488	21.68°	145.56°
	B-ODT	10415	18.10°	147.21°
	ODT	10415	11.64°	158.06°
	Topology optimization	10415	5.57°	163.24°
	NODT	10415	10.70°	157.19°
Retinal	Original mesh	14921	1.25°	173.85°
	eB-ODT	15030	19.86°	160.14°
	B-ODT	14948	15.10°	164.58°
	ODT	14921	1.29°	168.13°
	Topology optimization	14921	1.25°	172.09°
	NODT	14921	0.00°	179.99°
RyR	Original mesh	18585	6.19°	170.74°
	eB-ODT	18601	22.57°	143.56°
	B-ODT	18585	18.52°	149.25°
	ODT	18585	10.34°	158.32°
	Topology optimization	18585	6.19°	170.74°
	NODT	18585	7.78°	162.74°
2Torus	Original mesh	4635	5.96°	164.92°
	eB-ODT	4731	21.37°	146.81°
	B-ODT	4656	16.92°	152.05°
	ODT	4635	9.46°	157.53°
	Topology optimization	4635	6.85°	164.75°
	NODT	4635	0.01°	179.98°
FanDisk	Original mesh	9131	6.04°	164.98°
	eB-ODT	9173	20.32°	154.96°
	B-ODT	9162	16.80°	160.53°
	ODT	9131	9.59°	163.53°
	Topology optimization	9131	6.78°	164.98°
	NODT	9131	0.08°	179.86°

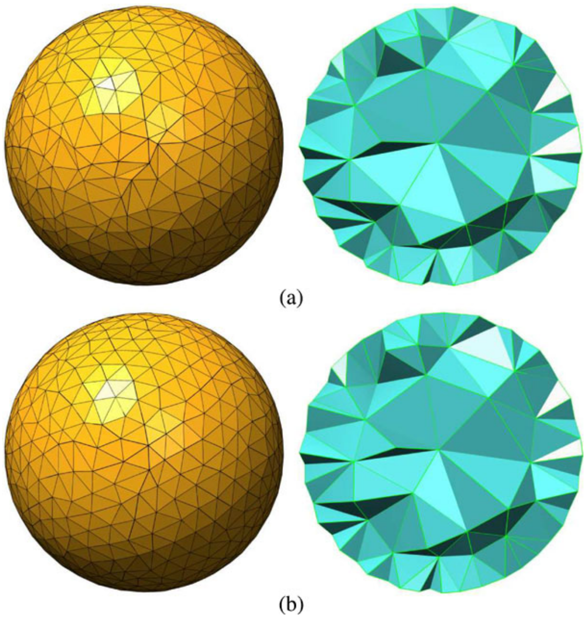


Fig. 3 The original mesh model (**a**) and the smoothed result (**b**). In both meshes, the outer and cross-section views are shown. The minimum dihedral angles of these two meshes are 5.86° and 18.20° respectively, and the maximum dihedral angles are 164.70° and 145.56° respectively

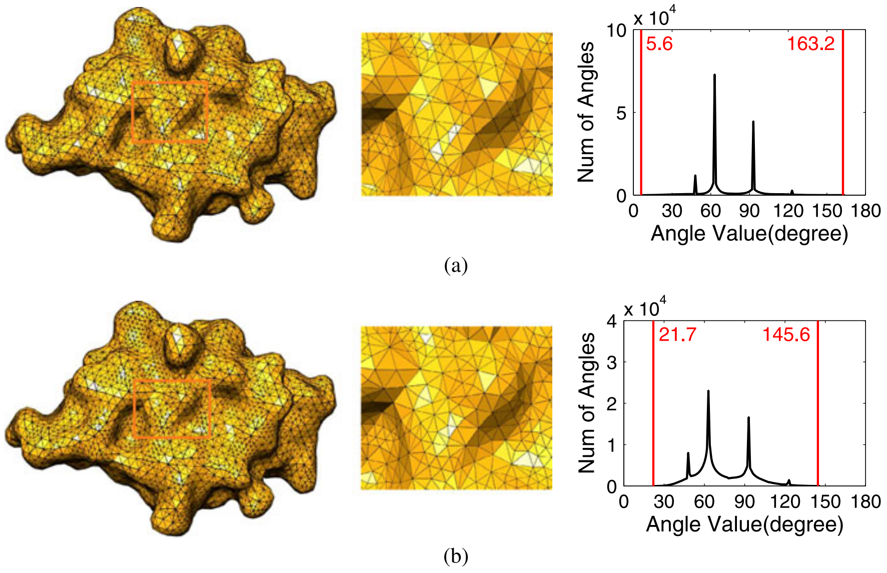
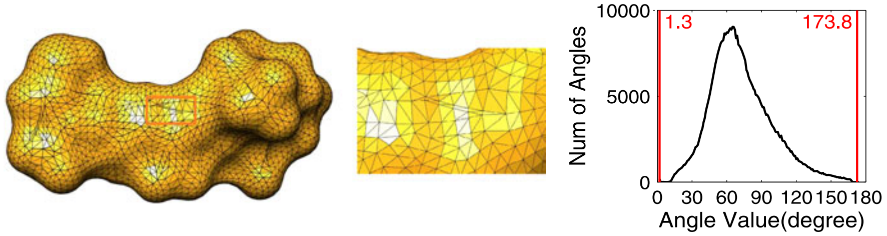
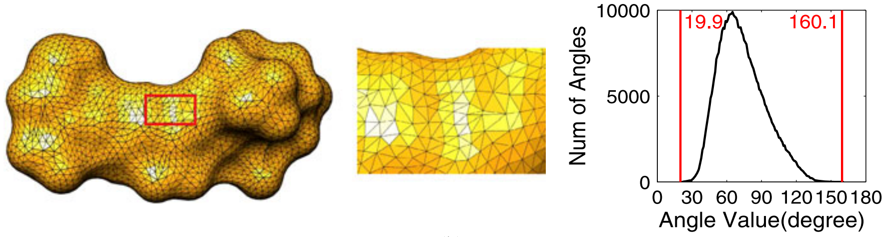


Fig. 4 Original and smoothed 2CMP models. The minimum dihedral angles of these two meshes are 5.57° and 21.68° respectively, and the maximum dihedral angles are 163.24° and 145.56° respectively

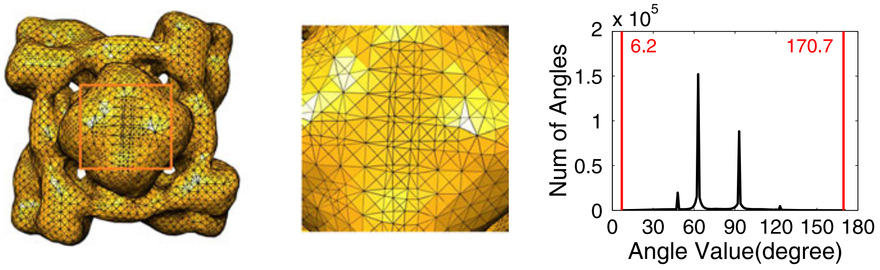


(a)

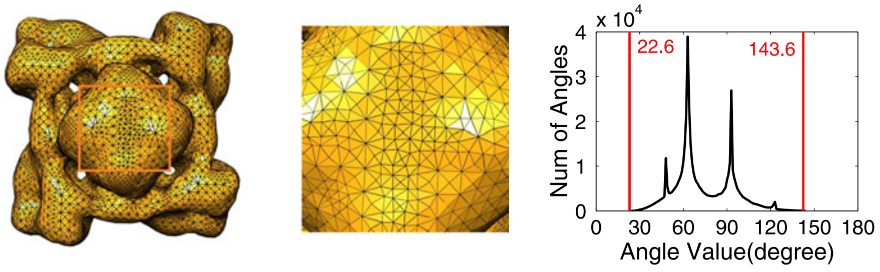


(b)

Fig. 5 Original and smoothed Retinal models. The minimum dihedral angles of these two meshes are 1.25° and 19.86° respectively, and the maximum dihedral angles are 173.85° and 160.14° respectively



(a)



(b)

Fig. 6 Original and smoothed RyR models. The minimum dihedral angles of these two meshes are 6.19° and 22.57° respectively, and the maximum dihedral angles are 170.74° and 143.56° respectively

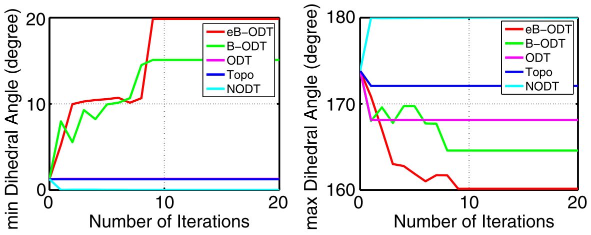


Fig. 7 The convergence of minimum and maximum dihedral angles with respect to the number of iterations on the Retinal model using the eB-ODT algorithm. Note that on the *left* the curves of ODT and topology optimization are almost identical

dihedral angles of this Random Sphere model are 5.86° and 164.70° respectively. After 20 times of running the eB-ODT algorithm, the minimum and maximum dihedral angles are improved to 18.20° and 145.56° respectively. Note that the distribution of the boundary vertices of the smoothed mesh is much more uniform than that of the original mesh, demonstrating that the eB-ODT algorithm can smooth both inner and boundary vertices in a tetrahedral mesh.

The eB-ODT algorithm is also tested on tetrahedral meshes generated from several biomedical molecules: 2CMP molecule in Fig. 4, Retinal molecule in Fig. 5 and Ryanodine receptor (RyR) in Fig. 6. The quality of 2CMP and RyR meshes reaches the best after no more than 10 iterations although all the models in Table 1 are processed 20 times. In Fig. 7, we demonstrate the convergence of minimum and maximum dihedral angles with respect to the number of iterations on the Retinal model using the eB-ODT algorithm.

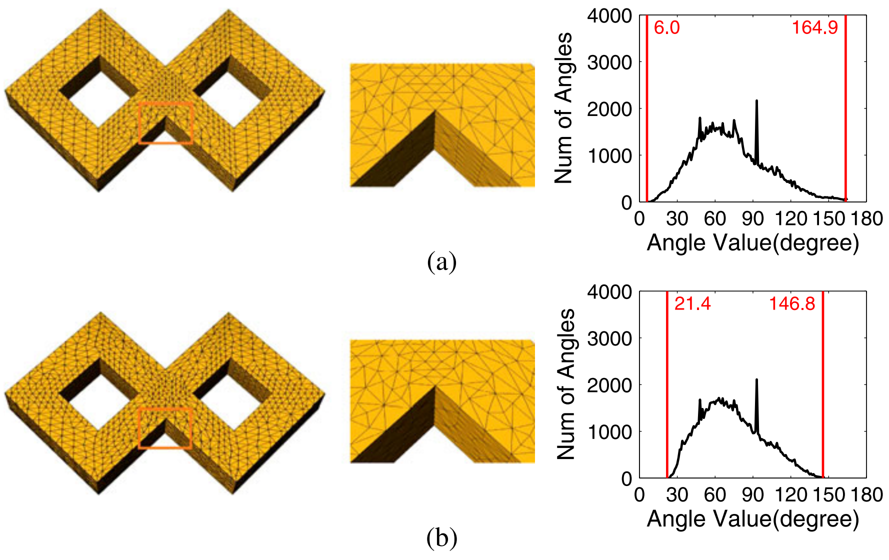


Fig. 8 Original and smoothed 2Torus models. The minimum dihedral angles of these two meshes are 5.96° and 21.37° respectively, and the maximum dihedral angles are 164.92° and 146.81° respectively

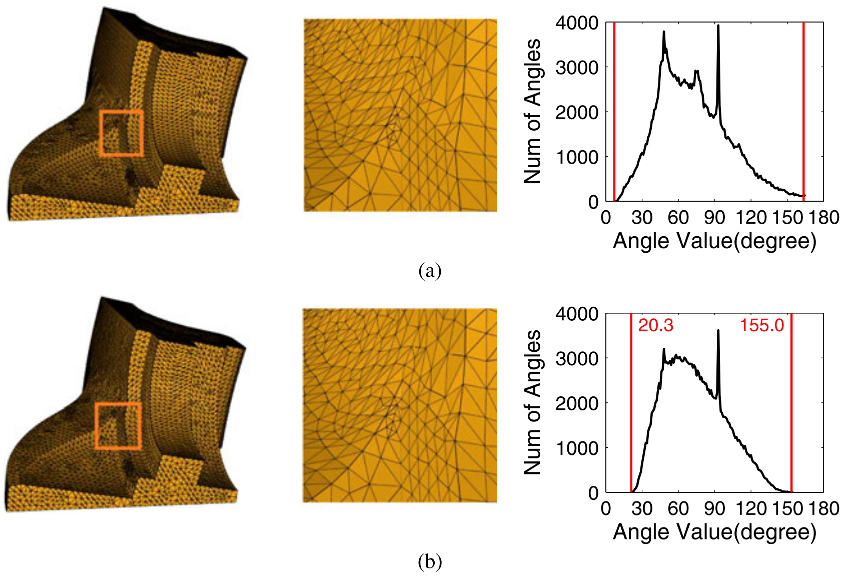


Fig. 9 Original and smoothed FanDisk models. The minimum dihedral angles of these two meshes are 6.04° and 20.32° respectively, and the maximum dihedral angles are 164.98° and 154.96° respectively

Table 2 Relative Hausdorff distance between original and smoothed meshes

Models	Random Sphere	2Cmp	Retinal	RyR	2Torus	FanDisk
Rel. Hausdorff distance	0.40%	1.11%	0.52%	0.77%	0.18%	0.25%

The 2Torus (Fig. 8) and FanDisk (Fig. 9) models show the feature-preserving property of the eB-ODT algorithm. In order to measure the difference between the original and smoothed meshes, we compute the relative Hausdorff distances between the surface meshes of the original and smoothed models, as shown in Table 2. Here, the Hausdorff distance is first computed using the standard definition and then scaled as follows. Let h be the absolute Hausdorff distance between the original and smoothed meshes, and L be the largest side length of the bounding box of the original mesh. The relative Hausdorff distances is defined by $\frac{h}{L}$, which measures the difference of the original and smoothed models relative to the size of the original model. From Table 2 we can see that the relative Hausdorff distances between the original and smoothed models are very small showing that our eB-ODT algorithm preserves the shape of the original models quite well.

The original ODT has also been extended by [46] to 3D tetrahedral mesh smoothing and the method is called Natural ODT (NODT). The NODT method computes the new position of a boundary vertex $\mathbf{x}_0$ in a tetrahedral mesh $\mathcal{T}$ by adding a certain amount of compensation to the weighted centroid of the neighborhood of $\mathbf{x}_0$. The

Table 3 Comparison of running time (20 iterations)

	Random Sphere	2Cmp	Retinal	RyR	2Torus	FanDisk
eB-ODT	305.84 s	5175.6 s	10272.8 s	9086.0 s	2258.4 s	4522.0 s
NODT	11.85 s	159.80 s	323.12 s	291.24 s	64.08 s	124.24 s

compensation is a weighted sum of the normal vectors of the boundary triangles around $\mathbf{x}_0$. Although boundary vertices are considered in the NODT method, the new positions calculated have to be projected onto the boundary of $\mathcal{T}$ to preserve the volume and shape of the original mesh. Therefore, the NODT method does not optimize the positions for boundary vertices. The smoothing results by using the afore-mentioned NODT method are shown in Table 1, where we can see that our eB-ODT algorithm significantly outperforms the NODT method. Sometimes the results obtained by the NODT method are even worse than the original meshes. The running time of eB-ODT and NODT is compared in Table 3.

4 Conclusions

We described a method of simultaneously smoothing both inner and boundary vertices of a tetrahedral mesh under a unified optimization framework. The eB-ODT algorithm presented can preserve sharp features very well and is guaranteed to preserve the volume of the original mesh. For every boundary vertex, the optimal position is computed by solving a linear system. The algorithm is numerically robust and easy to implement because the order of the linear equation system is only degree 2. The experimental results have shown the effectiveness of the proposed method.

Acknowledgements The work described was supported in part by an NIH Award (Number R15HL103497) from the National Heart, Lung, and Blood Institute (NHLBI) and by a subcontract from the National Biomedical Computation Resource (NIH Award Number P41 RR08605). The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsors.

References

1. Djidjev H (2000) Force-directed methods for smoothing unstructured triangular and tetrahedral meshes. In: 9th international meshing roundtable, pp 395–406
2. Phillippe P, Baker T (2001) A comparison of triangle quality measures. In: 10th international meshing roundtable, pp 327–340
3. Ohtake Y, Belyaev A, Bogaevski I (2001) Mesh regularization and adaptive smoothing. Comput Aided Des 33(11):789–800
4. Knupp PM (2002) Algebraic mesh quality metrics. SIAM J Sci Comput 23(1):193–218
5. Knupp PM (2003) Algebraic mesh quality metrics for unstructured initial meshes. Finite Elem Anal Des 39(3):217–241

6. Brewer M, Freitag Diachin L, Knupp P, Leurent T, Melander D (2003) The mesquite mesh quality improvement toolkit. In: 12th international meshing roundtable, pp 239–250
7. Babuska I, Aziz AK (1976) On the angle condition in the finite element method. SIAM J Numer Anal 13(2):214–226
8. Shewchuk JR (2002) What is a good linear element? Interpolation, conditioning, and quality measures. In: 11th international meshing roundtable, pp 115–126
9. Freitag LA, Ollivier-Gooch C (1997) Tetrahedral mesh improvement using swapping and smoothing. Int J Numer Methods Eng 40(21):3979–4002
10. Klingner BM, Shewchuk JR (2008) Aggressive tetrahedral mesh improvement. In: 16th international meshing roundtable. Springer, Berlin, pp 3–23
11. Chew LP (1997) Guaranteed-quality Delaunay meshing in 3D. In: 13th annual symposium on computational geometry, pp 391–393
12. Nave D, Chrisochoides N, Chew LP (2004) Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains. Comput Geom, Theory Appl 28(2–3):191–215
13. Escobar JM, Montenegro R, Montero G, Rodríguez E, Gonzáez-Yuste JM (2005) Smoothing and local refinement techniques for improving tetrahedral mesh quality. Comput Struct 83(28–30):2423–2430
14. Bank RE, Smith RK (1997) Mesh smoothing using a posteriori error estimates. SIAM J Numer Anal 34(3):979–997
15. Freitag LA (1997) On combining Laplacian and optimization-based mesh smoothing techniques. In: Trends in unstructured mesh generation. AMD, vol 220, pp 37–43
16. Canann SA, Tristano JR, Staten ML (1998) An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes. In: 7th international meshing roundtable, pp 419–494
17. Herrmann LR (1976) Laplacian-isoparametric grid generation scheme. J Eng Mech Div 102(5):749–907
18. Field DA (1988) Laplacian smoothing and Delaunay triangulations. Commun Appl Numer Methods 4(6):709–712
19. Hansbo P (1995) Generalized Laplacian smoothing of unstructured grids. Commun Numer Methods Eng 11(5):455–464
20. Zhou T, Shimada K (2000) An angle-based approach to two-dimensional mesh smoothing. In: 9th international meshing roundtable, pp 373–384
21. Xu H, Newman TS (2006) An angle-based optimization approach for 2D finite element mesh smoothing. Finite Elem Anal Des 42(13):1150–1164
22. Yu Z, Holst MJ, McCammon JA (2008) High-fidelity geometric modeling for biomedical applications. Finite Elem Anal Des 44(11):715–723
23. Du Q, Wang D (2003) Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. Int J Numer Methods Eng 56:1355–1373
24. Freitag L, Plassmann P (2001) Local optimization-based untangling algorithms for quadrilateral meshes. In: 10th international meshing roundtable, pp 397–406
25. Li X, Freitag LA, Freitag LA (1999) Optimization-based quadrilateral and hexahedral mesh untangling and smoothing techniques. Technical report, Argonne National Laboratory
26. Knupp PM (2001) Hexahedral and tetrahedral mesh untangling. Eng Comput 17(3):261–268
27. Knupp PM (2000) Hexahedral mesh untangling & algebraic mesh quality metrics. In: 9th international meshing roundtable, pp 173–183
28. Delanaye M, Hirsch C, Kovalev K (2003) Untangling and optimization of unstructured hexahedral meshes. Comput Math Math Phys 43(6):807–814
29. Menédez-Díaz A, González-Nicieza C, Álvarez-Vigil AE (2005) Hexahedral mesh smoothing using a direct method. Comput Geosci 31(4):453–463
30. Vartziotis D, Athanasiadis T, Goudas I, Wipper J (2008) Mesh smoothing using the geometric element transformation method. Comput Methods Appl Mech Eng 197(45–48):3760–3767
31. Vartziotis D, Wipper J, Schwald B (2009) The geometric element transformation method for tetrahedral mesh smoothing. Comput Methods Appl Mech Eng 199(1–4):169–182

32. Vartziotis D, Wipper J (2011) A dual element based geometric element transformation method for all-hexahedral mesh smoothing. Comput Methods Appl Mech Eng 200(9–12):1186–1203
33. Xu K, Cheng Z-Q, Wang Y, Xiong Y, Zhang H (2009) Quality encoding for tetrahedral mesh optimization. Comput Graph 33(3):250–261. IEEE international conference on shape modelling and applications 2009
34. Sirois Y, Dompierre J, Vallet M-G, Guibault F (2010) Hybrid mesh smoothing based on Riemannian metric non-conformity minimization. Finite Elem Anal Des 46(1–2):47–60
35. Parthasarathy V, Kodiyalam S (1991) A constrained optimization approach to finite element mesh smoothing. Finite Elem Anal Des 9(4):309–320
36. Canann SA, Stephenson MB, Blacker T (1993) Optismoothing: an optimization-driven approach to mesh smoothing. Finite Elem Anal Des 13(2–3):185–190
37. Chen CL, Szema KY, Chakravarthy SR (1995) Optimization of unstructured grid. In: 33rd aerospace sciences meeting and exhibit, Reno, NV, January, pp 1–10. AIAA 95-0217
38. Zavattieri PD, Dari EA, Buscaglia GC (1996) Optimization strategies in unstructured mesh generation. Int J Numer Methods Eng 39(12):2055–2071
39. Freitag Diachin L, Knupp P (1999) Tetrahedral element shape optimization via the Jacobian determinant and condition number. In: 8th international meshing roundtable, pp 247–258
40. Knupp PM (2000) Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I—a framework for surface mesh optimization. Int J Numer Methods Eng 48(3):401–420
41. Freitag LA, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Numer Methods Eng 49(1–2):109–125
42. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Numer Methods Eng 53(6):1377–1391
43. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2003) Simultaneous untangling and smoothing of tetrahedral meshes. Comput Methods Appl Mech Eng 192(25):2775–2787
44. Mezentsev A (2004) A generalized graph-theoretic mesh optimization model. In: 13th international meshing roundtable, pp 255–264
45. Chen L, Xu J (2004) Optimal Delaunay triangulations. J Comput Math 22(2):299–308
46. Tournois J, Wormser C, Alliez P, Desbrun M (2009) Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. ACM Trans Graph 28:75–1759
47. Labelle F, Shewchuk JR (2007) Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Trans Graph 26:57–15710
48. Zhang Y, Hughes TJR, Bajaj CL (2010) An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Eng 199(5–8):405–415
49. Molino N, Bridson R, Teram J, Fedkiw R (2003) A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: 12th international meshing roundtable, pp 103–114
50. Si H, Gärtner K, Fuhrmann J (2010) Boundary conforming Delaunay mesh generation. Comput Math Math Phys 50(1):38–53

High-Quality Multi-tissue Mesh Generation for Finite Element Analysis

Panagiotis A. Foteinos and Nikos P. Chrisochoides

Abstract Mesh generation on 3D segmented images is a fundamental step for the construction of realistic biomechanical models. Mesh elements with low or large dihedral angles are undesirable, since they are known to underpin the speed and accuracy of the subsequent finite element analysis. In this paper, we present an algorithm for meshing 3D multi-label images. A notable feature of our method is its ability to produce tetrahedra with very good dihedral angles respecting, at the same time, the interfaces created by two or more adjoining tissues. Our method employs a Delaunay refinement scheme orchestrated by special point rejection strategies which remove poorly shaped elements without deteriorating the representation of the objects' anatomical boundaries. Experimental evaluation on CT and MRI atlases have shown that our algorithm produces watertight meshes consisting of elements of very good quality (all the dihedral angles were between 19 and 150 degrees) which makes our method suitable for finite element simulations.

1 Introduction

Meshing multi-labeled medical images (like those obtained by segmenting MRI or CT images) provides the means for constructing accurate bio-mechanical models for subsequent finite element analysis. Multi-material mesh generation imposes challenges, since it should meet two conflicting requirements: *fidelity* and *quality*.

Fidelity measures the capability of the mesher to preserve the boundaries formed by two or more adjoining tissues. Quality regards the shape of the elements: tetrahedra with small or large dihedral angles (i.e., low quality tetrahedra) result in interpolation errors and in ill-conditioned stiffness matrices undermining in this way the accuracy and speed of the associated finite element analysis [19].

P.A. Foteinos (✉)
Department of Computer Science, College of William and Mary, Williamsburg, VA 23187, USA
e-mail: pfot@cs.wm.edu

N.P. Chrisochoides
Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
e-mail: nikos@cs.odu.edu

The difficulty in mesh generation is that the need to preserve high-curvature creases of the object's surface (i.e., high fidelity) deteriorates the quality of the meshes; on the other hand, the quality of mesh elements should be as high as possible when dealing with isotropic materials [8].

In this paper, we propose a Delaunay meshing algorithm able to respect the interfaces of multi-material domains and produce tetrahedra with very good dihedral angles and radius-edge ratios (and therefore very good aspect ratios), offering at the same time control over the size of the mesh.

1.1 Previous Work

In the literature, there has been work on multi-tissue meshing but the issue of high quality has not been adequately addressed.

Meyer et al. [14] employ a particle-based scheme producing watertight meshes that respect the interfaces formed by adjoining tissues. However, elements with practically zero dihedral angles (slivers) do appear in the final meshes. Furthermore, the execution times reported range from 3 to 12 hours even for small datasets.

Liu et al. [13] compress a body-centered cubic lattice (BCC) using a point-based registration method. The dihedral angles, however, can be as low as 4°. Also, the uniform lattice results in an unnecessary large number of elements in the interior of the objects.

Chentanez et al. [5] model the insertion of needles into soft tissues. The resulting conforming meshes are observed to consist of elements of angles more than 10.3° and less than 160°. It is worth noting that their goal is to represent a 1-dimensional curvilinear object (the needle) as a subset of a single-tissue mesh, which is a goal quite different from ours.

Goksel and Salcudean [9] present a variational meshing technique which combines both meshing and segmentation. They report minimum angles as large as 20°. The synthetic data they used for the evaluation is a sphere, that is, a 2-manifold. Usually, multi-tissue domains consists of complicated geometries, i.e., non-manifold parts which intersect with more than one tissues. These domains impose challenges to any meshing technique and are the focus of this work.

Zhang et al. [20] develop an octree-based meshing algorithm. Although edge-contraction and smoothing schemes are employed for quality improvement, the authors do not report the dihedral angles observed in their meshes.

Hu et al. [11] and Hartmann and Kruggel [10] develop uniform meshes for multi-material domains achieving dihedral angles more than 10°, albeit without good fidelity: their meshes suffer from the "staircase" effect.

Based on previous work on single material Delaunay surface [2] and volume meshing [15], Pons et al. [16] present a meshing algorithm for multi-tissue domains. Recently, Boltcheva et al. [3] extend the work of Pons et al. [16], so that 0- and 1-junctions are preserved in the final meshes. Both these methods apply *sliver exudation* [4] in order to improve the quality of the mesh. Edelsbrunner and

Guoy [7], however, have shown that in most cases sliver exudation does not remove all poor tetrahedra: elements with dihedral angles less than 5° survive. Indeed, Pons et al. [16] and Boltcheva et al. [3] report dihedral angles as low as 4°.

1.2 Our Contribution

In this paper, we present a Delaunay refinement algorithm for meshing multi-tissue medical data so that the boundaries between neighboring tissues are conforming. It works directly on segmented images meshing both the surface and the volume of the tissues.

A notable feature of our method is its ability to produce tetrahedra with very good dihedral angles: in all the experiments on synthetic and real images we ran, our algorithm produces watertight meshes consisting of tetrahedra with dihedral angles larger than 19° and smaller than 150°.

The technique we employ for quality improvement is inspired by the work of Shewchuk [18]. Therein, poor tetrahedra are eliminated by inserting the center of their circumball, giving priority to tetrahedra with larger radius-edge ratio. Shewchuk, however, meshes input domains bounded by polyhedral surfaces. In this paper, we extend this technique to deal with multi-tissue domains bounded by curved surfaces. The main difficulty is that vertices near the surface might be inserted during quality improvement. This fact in turn hurts fidelity: edges that cross interfaces or holes appear. To overcome this problem, we propose special *point rejection strategies*. They improve the quality of elements preventing the insertion of points near the surface; rather, carefully chosen points are inserted precisely on the surface. This allows to achieve both good quality and good fidelity meshes.

The rest of the paper is organized as follows: Sect. 2 outlines the concept of Delaunay refinement. The multi-tissue capability of our algorithm and the point rejection strategies are described in Sect. 3. Lastly, Sect. 4 presents results on CT and MR multi-label images and Sect. 5 concludes the paper.

2 Background

Delaunay meshes have been shown to successfully approximate the surface of both manifold and non-manifold surfaces [2], due to the properties of the *restricted Delaunay triangulations*, first introduced by Amenta and Bern [1].

Let $V \subset \mathcal{R}^3$ be a set of vertices and $\mathcal{D}(V)$ their Delaunay triangulation. Any Delaunay triangulation satisfies the *empty ball* property: the circumscribing open ball (a.k.a. *circumball*) of each tetrahedron in $\mathcal{D}(V)$ does not contain any vertex.

The *Voronoi point* of a tetrahedron $t \in \mathcal{D}(V)$ is defined as the center (a.k.a. *circumcenter*) of t's circumball. The *Voronoi edge* of a triangle $f \in \mathcal{D}(V)$ is the segment containing those points of $\mathcal{R}^3$ such that (a) they are equidistant from f's vertices and (b) they are closer to f's vertices than to any other vertex in V.

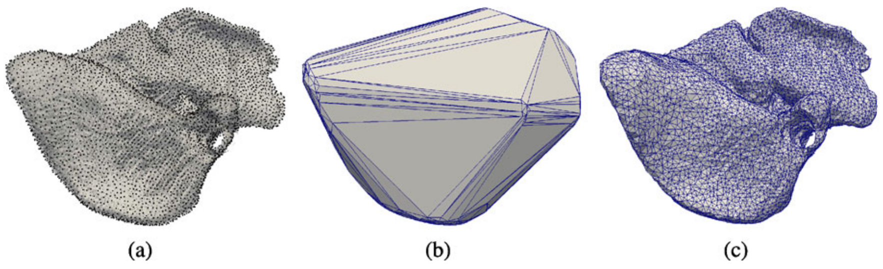


Fig. 1 (**a**) Sample set V of a liver's surface. (**b**) The Delaunay triangulation $\mathcal{D}(V)$ of the samples. (**c**) The restricted triangulation $\mathcal{D}_{|\mathcal{O}}(V)$

Let $\mathcal{O}$ be the multi-label domain to be meshed. We denote $\mathcal{O}$'s surface with $\partial\mathcal{O}$. The *restriction* of $\mathcal{D}(V)$ to $\mathcal{O}$ (denoted with $\mathcal{D}_{|\mathcal{O}}(V)$) is defined as the set of tetrahedra in the triangulation whose circumcenter lies inside $\mathcal{O}$.

It can be shown [2, 15] that if V samples $\partial\mathcal{O}$ sufficiently densely, then the set of boundary triangles (a.k.a. *restricted facets*) of $\mathcal{D}_{|\mathcal{O}}(V)$ is a good approximation of $\partial\mathcal{O}$ in a both topological and geometric sense. The approximation guarantees hold as long as $\partial\mathcal{O}$ does not have sharp corners. This is a reasonable assumption, since biological tissues do not exhibit sharp features on their surface. See Fig. 1 for a single-tissue example. The same idea extends to more than one tissues as well.

As an interesting consequence of the way $\mathcal{D}_{|\mathcal{O}}(V)$ is defined, only the Voronoi edges of the restricted facets intersect the surface $\partial\mathcal{O}$, a property that we will exploit in Sect. 3 to improve quality.

3 Our Method

The input of our algorithm is an image $\mathcal{I}$ containing the multi-material object $\mathcal{O}$. Image $\mathcal{I}$ can be seen as a function $f : \mathcal{R}^3 \mapsto \{0, 1, 2, \ldots, n\}$, such that $f(p)$ is the label that point $p \in \mathcal{R}^3$ belongs to. More precisely, $f(p)$ is the label of the voxel that p lies in. Usually, a label of 0 denotes voxels outside $\mathcal{O}$.

Points on the surface $\partial\mathcal{O}$ of object $\mathcal{O}$ are classified as those points lying in a voxel of label i which is incident to at least one other voxel of label j, such that $i < j$. In this way, surface $\partial\mathcal{O}$ contains not only the portions of the image that separate $\mathcal{O}$ from the background, but it also contains the interfaces that separate any adjoining tissues. The goal is to recover $\partial\mathcal{O}$ and mesh the volume (induced by $\partial\mathcal{O}$) at the same time.

Our algorithm first creates a box by inserting its 8 corners. The box contains $\mathcal{O}$ such that the (shortest) distance between the box and $\partial\mathcal{O}$ is larger than $2\delta\sqrt{2}$. Parameter δ is the only parameter that the users have to specify. This parameter determines how densely $\partial\mathcal{O}$ will be sampled: lower values indicate a denser sampling which in turn implies a better surface approximation. Notice that the calculation of the corners of the box is a quite trivial task, since it requires just one image traversal.

Next, the Delaunay triangulation of these corners is computed. This triangulation is the initial mesh (consisting of 12 tetrahedra) where the actual refinement starts from.

The refinement is governed by 2 steps, namely, *mesh conformity* and *point rejection quality improvement*. Upon termination, the tetrahedra whose circumcenter belongs to label i constitute the mesh representing the i^{th} tissue. Below, we outline each step separately.

3.1 Mesh Conformity

As noted in Sect. 2, vertices on $\partial\mathcal{O}$ have to be inserted in order for the mesh boundary (i.e., triangles incident to 2 or more tetrahedra of different labels) to be a good approximation of $\partial\mathcal{O}$. For this reason, we keep track of the tetrahedra whose circumball $\mathcal{B}$ intersects the surface $\partial\mathcal{O}$. We call such elements *intersecting* tetrahedra.

Suppose that an intersecting tetrahedron t is found. We compute the closest surface point—say p—to the center c of t's circumball $\mathcal{B}$. To facilitate the computation of such a point, we make use of an image Euclidean distance transformation [6]. If p is not closer than δ to any other surface vertex (already inserted in the mesh), then p is inserted (see Fig. 2(a)). Otherwise, and if the radius of $\mathcal{B}$ is larger than 2δ, c is inserted instead (see Fig. 2(b)). In this way, we can show that this step does not cause the insertion of infinite number of vertices and therefore, termination is not compromised.

For the same reason, we also require that no vertex is ever inserted outside the box. When the circumcenter c of an intersecting tetrahedron is chosen for insertion, however, c might lie outside the box. To prevent such cases, c is rejected and its projection on the box is inserted instead. See Figs. 2(c) and 2(d) for a couple of examples.

At the end of this step, all the vertices that do not lie on $\partial\mathcal{O}$ are deleted from the triangulation. At this moment, the restricted facets of the mesh are a good approximation of $\partial\mathcal{O}$, because the vertices remained in the triangulation form a dense sample of $\partial\mathcal{O}$ (see Sect. 2). Also, we can show that no 2 vertices are closer than δ and this is why δ controls the size of the mesh.

3.2 Point Rejection Quality Improvement

Our algorithm keeps track of poor tetrahedra, i.e., tetrahedra with small or large dihedral angles. Poor tetrahedra are eliminated by inserting their circumcenter. Priority is given to the tetrahedra with higher radius-edge ratio as in [18]. The radius-edge ratio of a tetrahedron t is defined as the length of t's circumball radius divided by the length of t's shortest edge.

Problems arise, however, when the circumcenter of a poor tetrahedron (about to be eliminated) lies close to the surface. If this is the case, the restricted facets in the

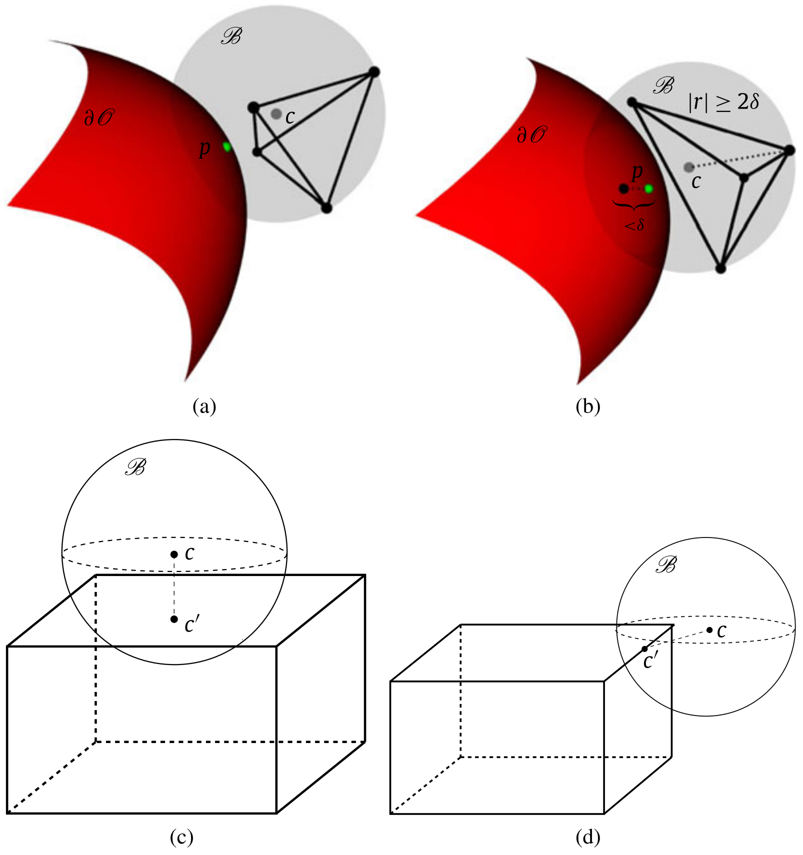


Fig. 2 (**a**) The closest surface point p to circumcenter c is inserted, (**b**) c is inserted but p is not, (**c**)–(**d**) c does not lie inside the box and therefore it is not inserted. Its projection c' is inserted and the vertices closer than δ to c' are deleted from the mesh

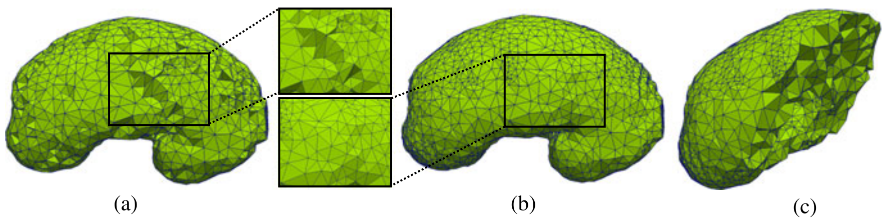


Fig. 3 Meshes for a kidney. All dihedral angles are between 19° and 150°. (**a**) No extra care has been taken to preserve fidelity and holes appear. (**b**) The point rejection strategies prevented the creation of holes. (**c**) A cross section of the mesh in (**b**)

triangulation are not any more a good approximation of $\partial\mathcal{O}$. See Fig. 3(a) for an example: the boundary facets have vertices that do not lie precisely on the surface.

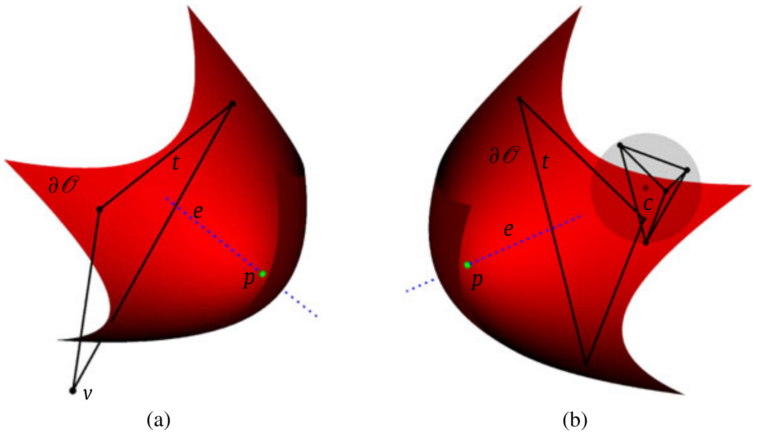


Fig. 4 The point rejection strategies. (**a**) t is an illegal facet. (**b**) t is a legal facet

To overcome this issue, we propose special *point rejection strategies*. Their goal is to make sure that all poor tetrahedra are eliminated without inserting points close to the surface.

Our algorithm first tries to convert *illegal* facets to *legal* ones. We define legal facets to be those restricted facets whose thee vertices lie precisely on $\partial\mathcal{O}$. Conversely, a restricted facet with at least one vertex not lying on $\partial\mathcal{O}$ is called an illegal facet.

Let t be an illegal facet and e its Voronoi edge (see Fig. 4(a) for an illustration). Recall that e has to intersect $\partial\mathcal{O}$ (see Sect. 2) at a point p. Any vertex v of t which does not lie precisely on $\partial\mathcal{O}$ is deleted from the triangulation, while point p is inserted. Note that since only non-surface vertices are deleted from the triangulation and since p is inserted on $\partial\mathcal{O}$, this step does not introduce an infinite loop: points that are inserted are never deleted.

In addition, the algorithm tries to keep in the Delaunay triangulation as many legal facets as possible. Let c be the circumcenter of a poor tetrahedron considered for insertion. If the insertion of c eliminates a legal facet t (see Fig. 4(b)), then c is not inserted. Instead, a point p on the intersection of $\partial\mathcal{O}$ and t's Voronoi edge e is inserted.

Figures 3(b) and 3(c) show how our algorithm meshed a kidney; observe that now the boundary facets have vertices lying precisely on $\partial\mathcal{O}$. In the next section, we will demonstrate that our point rejection strategies work also very well on multi-material domains.

4 Results

We ran our experiments on a 64 bit machine equipped with a 2.80 GHz quad-core Intel i7 processor and 8 GB of memory. Our algorithm was built on top of the *Com-*

putational Geometry Algorithms Library (CGAL, http://www.cgal.org). We used the *Insight Toolkit* (ITK, http://www.itk.org) for image processing. Lastly, the *Visualization Toolkit* (VTK, http://www.vtk.org) rendered the meshes.

Figure 5(a) illustrates the output mesh obtained for a segmented CT image taken from IRCAD (http://www.ircad.fr). Similarly, Fig. 5(b) depicts the output mesh obtained for the MR Brodmann atlas (http://www.sph.sc.edu/comd/rorden/mricro.html). Observe that the mesh elements are of excellent quality. Although we do not give guarantees on the minimum and maximum angles achieved by our method, we observed that the point rejection strategies are able to remove elements with angles less than 19° and more than 150° (in any image input we tried) without creating an edge smaller than $\frac{\delta}{10}$. It would be interesting to theoretically investigate why elements of worse quality are eliminated so easily without introducing very small edges. We leave that exploration as a future work. See the columns "ircad" and "brodmann" of Table 1 for some statistical results.

The last row of Table 1 shows the largest tetrahedron aspect ratio. Aspect ratio is defined as the ratio of a tetrahedron's circumradius to its inradius. The reported aspect ratio is normalized such that the best aspect ratio equals 1. Therefore, the aspect ratio ranges from 1 to $+\infty$. A high aspect ratio is an indication of bad quality.

Lastly, we show that the size of the mesh can be controlled directly by parameter δ. For this reason, we ran our mesher on the same CT image (obtained by IRCAD), but this time we set the value of δ at 8. That is, we double the value of δ used to obtain the mesh of Fig. 5(a). See Fig. 5(c) for an illustration and the last column of Table 1 for some statistical results. Observe that the number of elements, the number of vertices, and the execution time are greatly reduced, in the expense of worse fidelity. This is an expected trade-off: the fewer elements a mesh has, the less likely it is to represent complex surface creases accurately.

To evaluate our method, we compare it with CGAL (http://www.cgal.org). A comparison with other popular meshing techniques like Tetgen (http://tetgen.berlios.de/) or Netgen (http://www.hpfem.jku.at/netgen/) is omitted in this paper, because they do not operate directly on images. Rather, they require that the surface is already meshed as a piecewise linear complex. In contrast, both our algorithm and CGAL mesh the surface and the volume at the same time.

We run CGAL on the ircad CT image and report the achieved quality. We set CGAL's sizing parameters to values that gave output meshes with size similar to the size of our mesh depicted in Fig. 5(a). Furthermore, we set the quality parameters to their best theoretical values as described in [15]. Quantitative results for CGAL's output mesh are shown in Table 2. Compare it with the first column of Table 1. Observe that the quality of the CGAL mesh is lower than ours in terms of dihedral angles and aspect ratios. Another popular quality metric is the minimum scaled Jacobian value [12, 17]. It ranges from -1 to 1 with 1 being the best value. A negative value means that some elements are inverted. Both our algorithm and CGAL report positive scaled Jacobian values. In fact, the minimum Jacobian value achieved by our algorithm is 30 times larger than that achieved by CGAL. In terms of absolute numbers, we feel that the Jacobian values of our method is low, an issue we are looking into as future work.

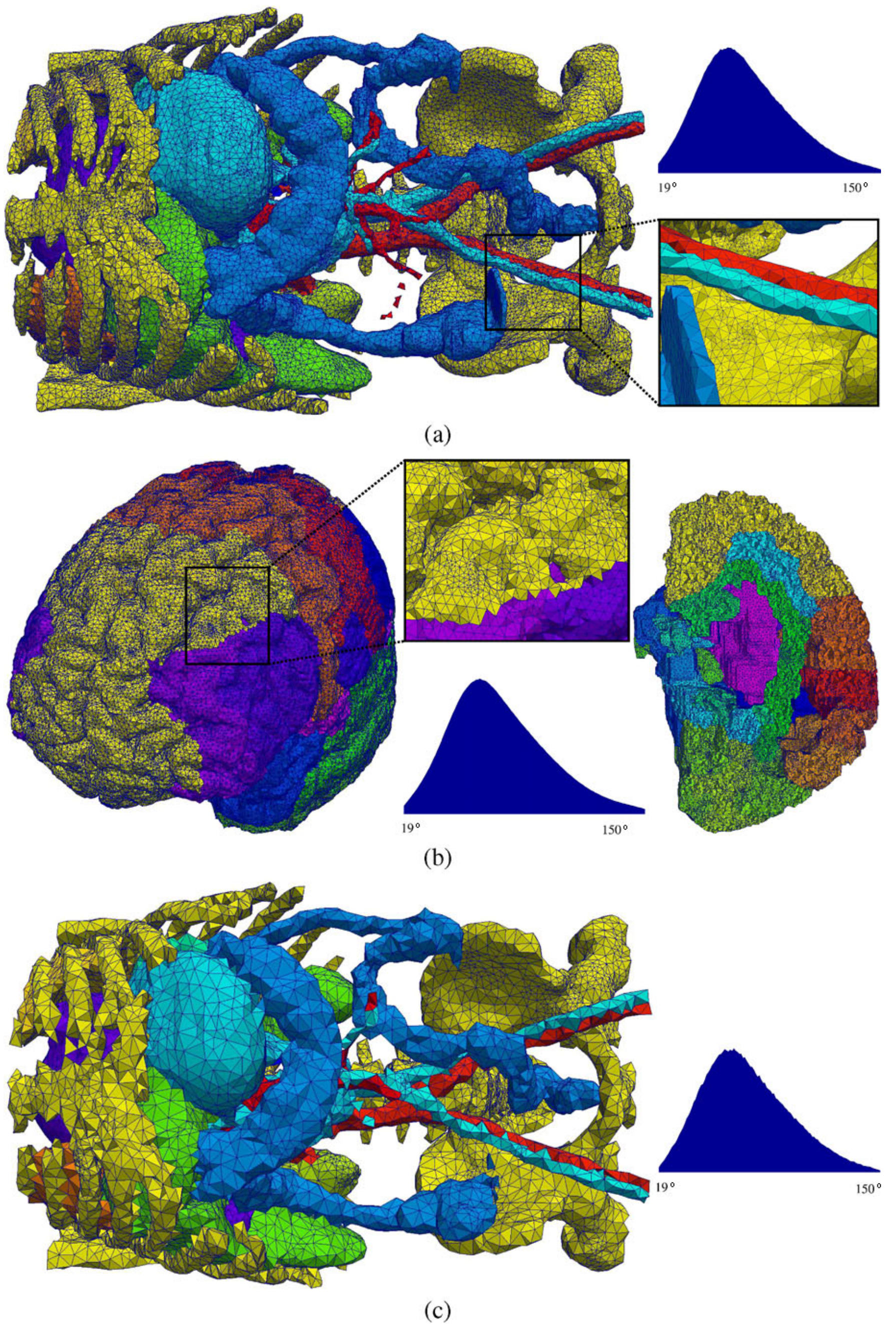


Fig. 5 Whole meshes, zoomed views, cross sections, and distributions of the dihedral angles for (**a**) the CT multi-label image and (**b**) the MR brain atlas. In (**c**), we show a coarser mesh on the same input image that was used in (**a**)

Table 1 Information about the images used for the evaluation, the chosen value for parameter δ, and some quantitative results of the final meshes produced by our algorithm

Experiment	ircad	brodmann	ircad(coarse)
Image size	$512 \times 512 \times 219$	$181 \times 217 \times 181$	$512 \times 512 \times 219$
Image resolution (mm)	$0.961 \times 0.961 \times 2.4$	$1 \times 1 \times 1$	$0.961 \times 0.961 \times 2.4$
#Labels	20	41	20
δ (mm)	4	2	8
Time (sec)	421	1,066	96
#Vertices	139,740	473,994	41,097
#Tetrahedra	783,445	2,575,220	173,575
Dihedral angles (degrees)	19–150	19–150	19–150
Max. aspect ratio (normalized)	4.67	6.22	4.55

Table 2 Quantitative results of the final mesh produced by CGAL

Experiment	ircad
#Vertices	156,902
#Tetrahedra	756,462
Dihedral angles (degrees)	3–174
Max. aspect ratio (normalized)	16,823

5 Conclusions and Future Work

In conclusion, we have shown that Delaunay refinement techniques are able to mesh multi-material domains with tetrahedra of very good angles, which makes our method suitable for subsequent finite element analysis. The point rejection strategies proposed in this work maintain mesh conformity and high quality offering, at the same time, control over the mesh size.

Note that surface patches of high curvature need to be meshed with more elements than patches that are not sharp. In its current state, our method meshes the surfaces uniformly. In the future, we plan to extend our method to produce graded triangular surfaces and, therefore, smaller meshes.

Acknowledgements We thank the anonymous referees for their comments and insight that helped improve the presentation and completeness of the paper. This work is supported in part by NSF grants: CCF-1139864, CCF-1136538, and CSI-1136536 and by the John Simon Guggenheim Foundation and the Richard T. Cheng Endowment.

References

1. Amenta N, Bern M (1998) Surface reconstruction by Voronoi filtering. In: SCG '98: proceedings of the fourteenth annual symposium on computational geometry. ACM, New York, pp 39–48

2. Boissonnat JD, Oudot S (2005) Provably good sampling and meshing of surfaces. Graph Models 67(5):405–451
3. Boltcheva D, Yvinec Y, Boissonnat J-D (2009) Mesh generation from 3D multi-material images. In: Medical image computing and computer-assisted intervention. Springer, Berlin, pp 283–290
4. Cheng SW, Dey TK, Edelsbrunner H, Facello MA, Teng SH (2000) Sliver exudation. J ACM 47(5):883–904
5. Chentanez N, Alterovitz R, Ritchie D, Cho L, Hauser KK, Goldberg K, Shewchuk JR, O'Brien JF (2009) Interactive simulation of surgical needle insertion and steering. In: Proceedings of ACM SIGGRAPH 2009, article 88
6. Danielsson PE (1980) Euclidean distance mapping. Comput Graph Image Process 14:227–248
7. Edelsbrunner H, Guoy D (2002) An experimental study of sliver exudation. Eng Comput 18:229–240
8. Goksel O, Salcudean SE (2009) High-quality model generation for finite element simulation of tissue deformation. In: 12th international conference on medical image computing and computer-assisted intervention (MICCAI). MICCAI '09. Springer, Berlin, pp 248–256
9. Goksel O, Salcudean SE (2011) Image-based variational meshing. IEEE Trans Med Imaging 30(1):11–21. doi:10.1109/TMI.2010.2055884
10. Hartmann U, Kruggel F (1998) A fast algorithm for generating large tetrahedral 3D finite element meshes from magnetic resonance tomograms. In: Proceedings of the IEEE workshop on biomedical image analysis. WBIA. IEEE Comput Soc, Washington, pp 184–192
11. Hu P, Chen H, Wu W, Heng P-A (2010) Multi-tissue tetrahedral mesh generation from medical images. In: International conference on bioinformatics and biomedical engineering (iCBBE), pp 1–4
12. Knupp PM (2000) Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II—a framework for volume mesh optimization and the condition number of the Jacobian matrix. Int J Numer Methods Eng 48(8):1165–1185
13. Liu Y, Foteinos P, Chernikov A, Chrisochoides N (2010) Multi-tissue mesh generation for brain images. In: Proceedings of the 19th international meshing roundtable, Chattanooga, TN, USA
14. Meyer M, Whitaker R, Kirby RM, Ledergerber C, Pfister H (2008) Particle-based sampling and meshing of surfaces in multimaterial volumes. IEEE Trans Vis Comput Graph 14(6):1372–1379
15. Oudot S, Rineau L, Yvinec M (2005) Meshing volumes bounded by smooth surfaces. In: Proceedings of the international meshing roundtable, San Diego, CA, USA. Springer, Berlin, pp 203–219
16. Pons JP, Ségonne F, Boissonnat JD, Rineau L, Yvinec M, Keriven R (2007) High-quality consistent meshing of multi-label datasets. In: Information processing in medical imaging, pp 198–210
17. Shepherd JF, Tuttle CJ, Silva CT, Zhang Y (2006) Quality improvement and feature capture in hexahedral meshes. SCI institute technical report, University of Utah
18. Shewchuk JR (1998) Tetrahedral mesh generation by Delaunay refinement. In: Proceedings of the 14th ACM symposium on computational geometry, Minneapolis, MN, pp 86–95
19. Shewchuk JR (2002) What is a good linear element?—Interpolation, conditioning, and quality measures. In: Proceedings of the 11th international meshing roundtable, Sandia National Laboratories, pp 115–126
20. Zhang Y, Hughes TJR, Bajaj CL (2010) An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Eng 199(5–8):405–415

Construction of Models and Meshes of Heterogeneous Material Microstructures from Image Data

Ottmar Klaas, Mark W. Beall, and Mark S. Shephard

Abstract This chapter presents a set of procedures that start from image data to construct a non-manifold geometric model that supports the effective generation of meshes with the types of mesh configurations and gradations needed for efficient simulations. The types of operations needed to process the image information before and during the creation of the non-manifold geometric domains are outlined, with emphasis on those methods that are most appropriate for the analysis of materials system's behavior.

1 Introduction

One of the most important scales quantifying the behavior of materials is the mesoscale at which the mechanics of grain interfaces, voids and inclusions can be modeled. Often neglected in the development of mesoscale simulation technologies are the tools needed to support the accurate definition of the heterogeneous mesoscale geometry and automatic generation of meshes for the accurate prediction of the critical solution parameters. The accurate representation of the mesoscale geometry requires a statistically accurate representation of the grains, interfaces, voids, and inclusions as they exist in the as processed material systems.

Imaging technologies, such as X-ray computed microtomography (XCMT), have continued to develop to the point that they can provide a voxel level description of many important materials systems with sufficient resolution to construct the needed mesoscale geometries.

O. Klaas (✉) · M.W. Beall
Simmetrix Inc., 10 Executive Park Drive, Clifton Park, NY 12065, USA
e-mail: oklaas@simmetrix.com

M.W. Beall
e-mail: mbeall@simmetrix.com

M.S. Shephard
Rensselaer Polytechnic Institute, Troy, NY 12180, USA
e-mail: shephard@rpi.edu

This paper presents a set of components that take 3D voxel data, originating from either a statistically generated 3D description of a polycrystal material or a set of image slices, as input and, in combination with knowledge of the material systems involved and simulation methods to be applied, construct proper geometric representations and associated simulation discretizations (meshes) suitable for use in mesoscale analyzes. The methods used in the generation of these representations and discretizations include (i) tools to deal with irregularities at the level of the voxel data set (elimination of physically impossible voxel constellations and small features introduced by scanning or segmentation errors) (ii) tools to treat quantization artifacts at the level of the geometric model, and (iii) methods to enforce periodic boundaries when desired. With a geometric model as one of the outcomes of the presented process, the user has the ability to set mesh control such that the meshes generated support the simulation requirements in terms of parameters to be predicted and accuracy required.

Examples are presented to show the effectiveness of the presented methods. They start from data represented as a series of images representing a 3D volume, or data sets generated by an electron back scatter diffraction method.

2 Image Data Input

The continued advancement of image techniques is providing "geometric information" in terms of discrete gray scale levels over a uniform grid of voxels. The use of such information in many medical related areas is common place with its use in combination with various simulation techniques increasing at a dramatic rate. Another area of application where image data is beginning to be used on a more regular basis is the quantification of as processed materials.

2.1 3D Voxel Using XCMT

Commercially available imaging systems such as X-ray computed microtomography (XCMT) [6] systems have advanced to the point that they can provide accurate microstructural information for many materials (for examples see e.g. [5]). The output of the imaging system is a set of image slices that, put together provide a set of voxels with image intensity data associated with each voxel. The first step in the process of constructing the geometric model is to convert that image data into voxel sets where each voxel is labeled such that voxels with similar characteristics receive the same label. The labeled voxel set is called a segmentation of the original data set. The characteristics to consider vary depending on the application. In medical imaging different distinct tissues may have the same gray-level appearance and thus the gray level alone is not a sufficient indicator to assign a voxel label. On the other hand, due to deficiencies of the imaging process, materials that ought to be identical

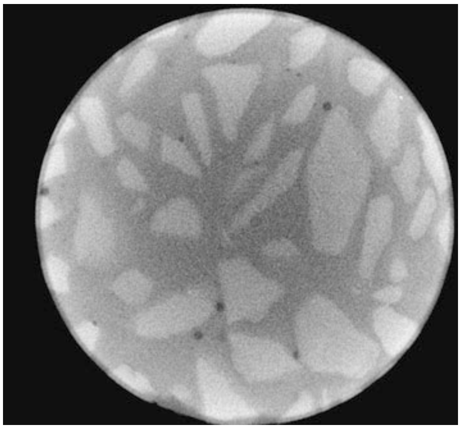


Fig. 1 Image slice of a XCMT dataset of a concrete specimen

can be represented with different gray levels. This effect is often seen by comparing the gray level of objects in the center of the image with the gray level of objects closer to the boundary of the image.

There are a number of algorithmic approaches, and associated software, to carry out the segmentation process [3]. The open source Insight Segmentation and Registration Toolkit (ITK) [12] includes a wide variety of these algorithms and is used in this work. 3DSlicer is a graphical front end to a subset of the ITK algorithms [8, 22, 23, 31].

The basic segmentation algorithms are thresholding procedures, which group gray values into buckets based on a threshold value and their location in the histogram. Edge detection algorithms try to find connected voxels forming regions by locating rapid changes in the image. Region growing methods [30] start from a set of seed points that are iteratively grown by comparing the points identified with the neighboring voxels and deciding whether they are to be included into the region or not based on information gathered from already segmented areas.

The complexity of the algorithms and the parameters to control the behavior of those algorithms that need to be applied is a function of the degree of contrast of the constituent materials to be detected as well as the level and type of noise present in the image data. When noise is present, this image needs to be preprocessed with noise filtering algorithms before it can be handed to the segmentation procedures. Again there are a wide variety of noise filters available, each with different characteristics and thus they have to be appropriately paired with the segmentation method to be used. As an example, many noise filters will by their very nature smear out distinct sharp jumps in gray level values, thus making their choice inadequate for segmentation algorithms that operate on detecting such features (e.g. edge detection segmentation procedures). Other noise filters, e.g. the so called anisotropic diffusion filter, are designed to reduce noise while preserving the features that edge detection algorithms rely on for the segmentation process.

Figure 1 shows an example of image deficiencies that the segmentation process has to contend with. The image shows significant randomly distributed noise, imaging artifacts presenting themselves as concentric circles, and a very visible change

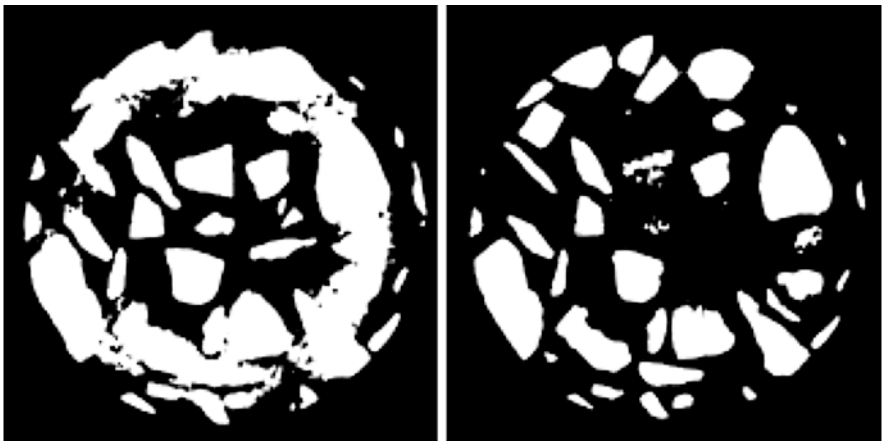
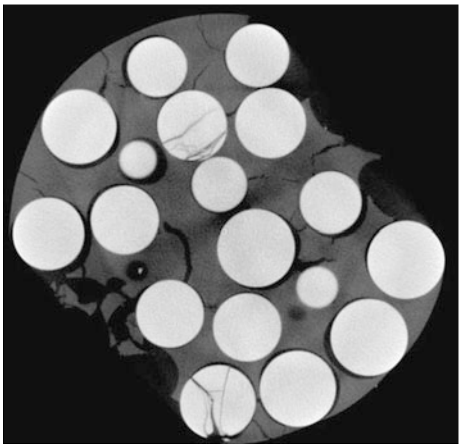


Fig. 2 Unacceptable segmentation results obtained using tresholding

Fig. 3 Image slice of Large Glass beads suspended in hydroxyl-terminated polybutadiene



in the average gray level toward the boundary of the circular image domain. Trying to simply apply global tresholding to segment this data leads to inadequate results, capturing either the center area or the boundary area (Fig. 2 left and right, respectively) correctly, leaving some particles undetected or detecting the matrix material as particles as well.

Note that in the case of imaging of inanimate objects many of the problems may be overcome as the imaging process can use higher doses of radiation without having to worry about harming the object. Figure 3 shows an image slice of glass beads suspended in hydroxyl-terminated polybutadiene. One can see that the noise level is reduced and there are less artifacts from the imaging process (no visual concentric circles). However, there are still some issues with varying gray levels throughout the image, causing a fully automatic segmentation process to miss some parts of the glass beads on the boundary (see Fig. 4 at the bottom right). The Simple Region

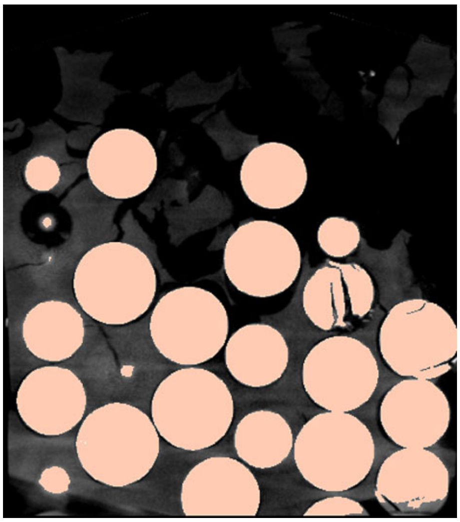


Fig. 4 Segmentation of Large Glass Beads

Growing algorithm, which is part of 3DSlicer was used for the segmentation in this example.

For imaging done on living subjects there are a wide variety of issues that typically make it harder to create images that are optimal for the segmentation process. Movement of the subject (e.g. due to breathing) can cause image artifacts and distortions. Some scanning methods rely on radiation (e.g. CT scans), and while for imaging purposes a high dose of radiation would be beneficial, the negative health effects require lower radiation doses that cause substantial noise and artifacts in the created images.

2.2 2D Slices Plus Statistical Processing

Electron back-scatter diffraction (EBSD) is a highly accurate method to provide a spatially resolved orientation map of a sample surface [1]. Its primary use has been focused on two-dimensional surfaces [1, 24], since the accuracy of the method is quite high when applied to a properly prepared surface. Efforts on methods to directly construct 3D representations are under consideration [17], but are not currently of high accuracy.

One method that has been developed for the automatic construction of fully three-dimensional representations is to combine statistical methods with a limited set of two-dimensional sections to construct statistically equivalent microstructures [17, 24]. The most well known tool for this type of construction is the Microstructure builder [19]. The input to the Microstructure builder consists of grain size and shape data as obtained from orthogonal images. The output is a 3-D voxel structure that matches the size and shape statistics provided as input. Microstructure builder is currently to create microstructures for (i) single-phase, equiaxised and non-equiaxised microstructures, (ii) single-phase, variable grain shape (with certain limits), (iii) orientation distribution matching only, or both orientation and misorientation matching and (iv) two-phase with either high volume fraction of second phase particle or with smaller particles.

3 Construction of Geometric Model

There are a number of potential methods to go from the segmented voxel data to meshes that could be used in the simulation. However, these methods yield a fine uniform mesh that is over resolved in many areas while still providing what is typically a poor geometric representation of the material phases, which can violate known properties of the given material system. The alternative, used in this work, is to convert the image data into a multi-material non-manifold boundary representation [34] that accounts for both the specifics of the given image data and known properties of the material system.

Given the finite size of the image voxels, the limited level of contrast produced, and the presence of noise in the results, any geometric representation produced from material microstructural image data is an approximation of the constituents present. The level of fidelity of the resulting representation with respect to the operations of interest, in this case the evaluation of mechanical behavior, can be further enhanced by accounting for known geometric properties of the constituents in the heterogeneous material system. Examples of properties of the material system can include: The material components (crystals or inclusions) are of a minimum size. The material components faces are flat with sharp edges or are rounded and are surrounded by filler. The interface between components will not be of dimension lower than two, or one. When known, accounting for such properties, which are material system dependent, can yield statistically more accurate geometric representations of these heterogeneous systems.

Taking account of knowledge of the accuracy limitation of the imaging modality in terms of voxel resolution relative to feature size, contrast level and/or consistency, is also important. When detected features are smaller than meaningful for the given resolution, their inclusion in the geometric model should be questioned, and in the case where those features are not consistent with known properties they should be eliminated.

3.1 Voxel Data Processing

Voxel input data produced by the methods above can present situations that lead to topological inconsistencies that, in turn, prevent construction of non-manifold solid models that are physically consistent with the properties of the microstructure. The typical example is a situation where two voxels representing the same grain touch in

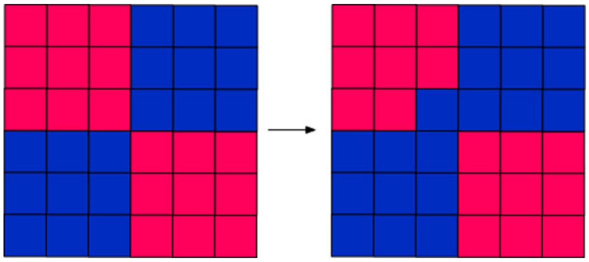


Fig. 5 Elimination of corner singularity in voxel data set

such a manner that at a single common vertex or a single common edge is created between two components. Figure 5 illustrates the situation in 2D, where the voxels of the same material (indicated by the colors) touch through a single common vertex at the center. Although physically implausible with actual materials, these conditions can occur in datasets because they are a quantized representation. To eliminate these situations one of the local voxels is reassigned a different material label to correct it. For common vertices, the logic uses voting to select which voxel to reassign and what grain to reassign it to based on the neighboring voxels. The same approach is used for common-edge conditions, with a generalization allowing any one of the four voxels connected to the common edge to be selected for reassignment. In some cases, the reassignment can create another topological inconsistency in the dataset, so a multiple-pass algorithm is used to iteratively remove all of these conditions. Because these conditions are sparsely distributed, originate from inaccurate processing of the original data to begin with, and involve single-voxel modifications only, the impact on final model construction is negligible.

The process just described is the last to be executed before the non-manifold model is constructed. This is as important as other processes run on the voxel data set (e.g., small object removal, erosion/dilation) have the potential to introduce situations that would yield the generation of a non-realistic non-manifold model from the voxel data set.

As an example of the importance of processing a dataset, a synthetic volume with 5269 grains [10] was processed. In that dataset, which contains slightly more than 10 million voxels, the procedures above corrected 150,000 physically implausible conditions in order to create a valid geometric model. In some places, the voxel data exhibits a check board pattern, indicating that the quantization level was very close to the Nyquist limit for reliable reconstruction.

Another significant issue is the underlying quantization of the voxel data. In some cases, datasets contain very small individual grains on the order of 10 or 100 total voxels. Those voxel groups could have been generated for example because changes in gray level of the input image were large enough to cause the segmentation process to assume a different material at that location. Those small voxel groups lead to difficulties downstream as they are turned into small geometric objects in the non-manifold model. That not only leads to a highly refined mesh were none is needed, due to the small size of the geometric objects that need to be resolved, but often self-intersections occur in the resulting discretization. This leads to a discussion of what constitutes an acceptable quantization level and some evolution in knowledge from recent experiences about the problem of representing microstructures with voxel data. When characterizing a specific grain boundary plane, the error can be readily quantified, and it appears that in 2-D at least ten voxels are required to capture a straight-line segment and maintain acceptable confidence that the feature is captured at the appropriate angle. If this holds for 3-D as well, a given flat grain interface should have a diameter of at least ten voxels, or an area of at least 79 square voxels. Extending this thinking to the volume, measured in terms of square voxels, we begin to lose confidence for a "spherical grain" with a diameter of less than ten voxels which translates to a volume of 523 voxels. A more liberal threshold of seven voxels/linear boundary section has also been suggested, which translates to a spherical volume of 180 voxels.

Recognizing that there is some minimum resolution at which grain geometry can no longer be accurately captured, an algorithm was developed that first filteres the data to detect grains with sub-minimal volume followed by voxel reassignment by neighbor voting. For larger objects made up of hundreds of voxels, however, voxel reassignment can only be applied at the object boundaries, where there are one or more "external" voxel neighbors to use for voting. So the process works iteratively, reassigning voxels on the object boundary each pass, thereby making the object smaller and smaller, until the object is removed. The software takes a threshold value as input to specify the size of objects, in voxels, that are removed by this processing.

While the small object removal process can eliminate artifacts that represent themselves as disconnected groups of a small number of voxels making up the object, it will not remove objects that are small (on the order of one or two voxels) in one or two dimensions, but connected to a larger object. These objects significantly influence mesh generation since their small size requires extremely fine meshes so they can be resolved appropriately without causing mesh self-intersections. As they are often artifacts of the imaging and segmentation process, they don't represent real data that needs to be resolved, and thus it is advisable to eliminate them before the non-manifold model is constructed. One method that can successfully be used for this purpose is an erosion/dilation procedure. Erosion/dilation algorithms belong to a set of algorithms developed as part of mathematical morphology [9, 32]. Morphology operations change the input voxel data set by applying a structuring element, effectively changing the shape of objects in the underlying voxel data set. In the case of erosion/dilation the erosion operation deletes voxels on the boundary of objects, while the dilation operation adds voxels. Together they have the effect to leave the boundary of larger smooth objects mostly untouched, while small outliers of one or two voxels thickness will be eliminated. The strength of the operation can be controlled by the shape and size of the structuring elements as well as how many passes the algorithms takes through the data set.

While originally developed for binary images, we have extended the algorithm to work on voxel data sets of multiple materials. Similar to the process of eliminating physically implausible situations from the voxel data set, for multi-material situations decisions have to be made during erosion to determine what material label to give to the voxel to be eroded, and similarly during dilation. A careful implementation of the neighbor voxel voting process can make sure that a data set without any small artifacts does not get disturbed in the areas where several materials meet.

3.2 Construction of Non-manifold Model Topology

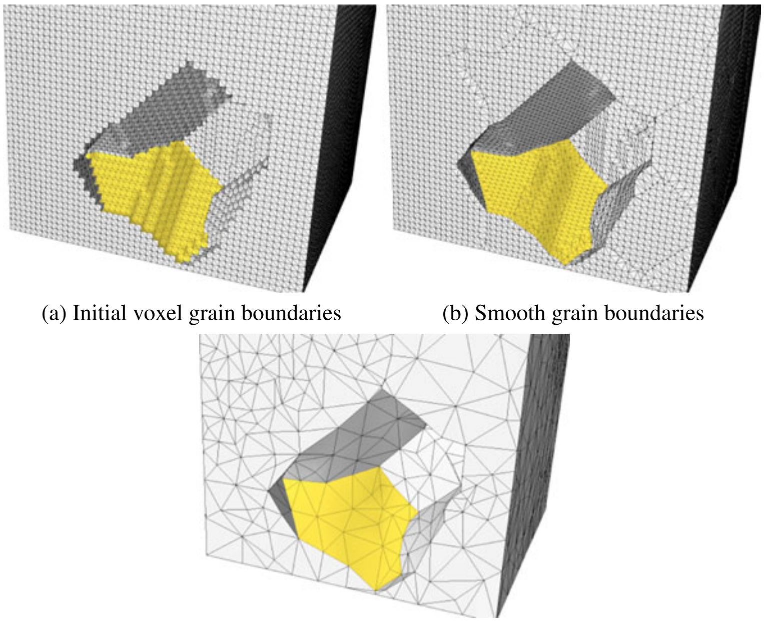
The most basic of the procedures to convert segmented voxel data to a form useful for the definition of a non-manifold geometric model is a marching cube type of procedure, originally defined for the extraction of a single triangulated isosurface [18]. Recent versions of this class of method are better able to account for information from neighboring voxels/octants to produce more controlled triangulations [13, 26] and to account for non-manifold multimaterial interaction [35]. A drawback of the marching cube type method is that the triangulations are at the resolution of the voxel/sampling size, which is typically higher than needed for a non-sampled representation, thus methods that reduce the size and complexity of these triangulations are typically applied [7, 11]. In the present work, an initial triangulation is created based on voxel level operations like those in [13, 26, 35] however, mesh coarsening is not preformed for this mesh until after the grain topology is determined and the surface triangulation is smoothed to get an accurate description of the geometry.

The process of defining the non-manifold boundary representation for a surface triangulation is focused on processes that group sets of triangles into faces and applying feature detection methods to define edges and vertices that bound faces [14, 21].

Due to the voxel nature of the initial geometry information the surface geometries created contain quantization artifacts on the scale of the individual voxels. Thus some form of surface smoothing method is needed to create more realistic shapes of the faces. Conventional data-smoothing methods, however, are not well suited to removing the quantization artifacts found in voxel datasets. Laplacian smoothing, for example, is only effective when the perturbations to be removed obey a smoothly varying statistical distribution. Volume-preserving smoothing filters [20] have been developed for medical imaging applications, however, such filters distort the surface geometry at grain/grain boundaries. Although overall grain volume is important, preserving the fidelity of the grain interfaces is much more important to obtaining the most accurate simulation results.

Therefore, a new algorithm for data smoothing designed specifically to remove quantization artifacts and recover the underlying surface geometry was developed. The algorithm is applied on a geometric face-by-face basis such that for each face the following steps are performed: (i) Calculate the surface normal at each mesh face based on the normal vector of its neighboring mesh faces. (ii) Smooth the surface normals to obtain the normals of the desired surface. (iii) Iteratively adjust the mesh vertex positions to create a surface matching the smoothed surface normals.

Figure 6 shows an example of this process. Image (a) shows the triangulation before normal smoothing and triangulation adjustment, while image (b) shows the



(a) Initial voxel grain boundaries (b) Smooth grain boundaries

(c) Resulting surface mesh after coarsening

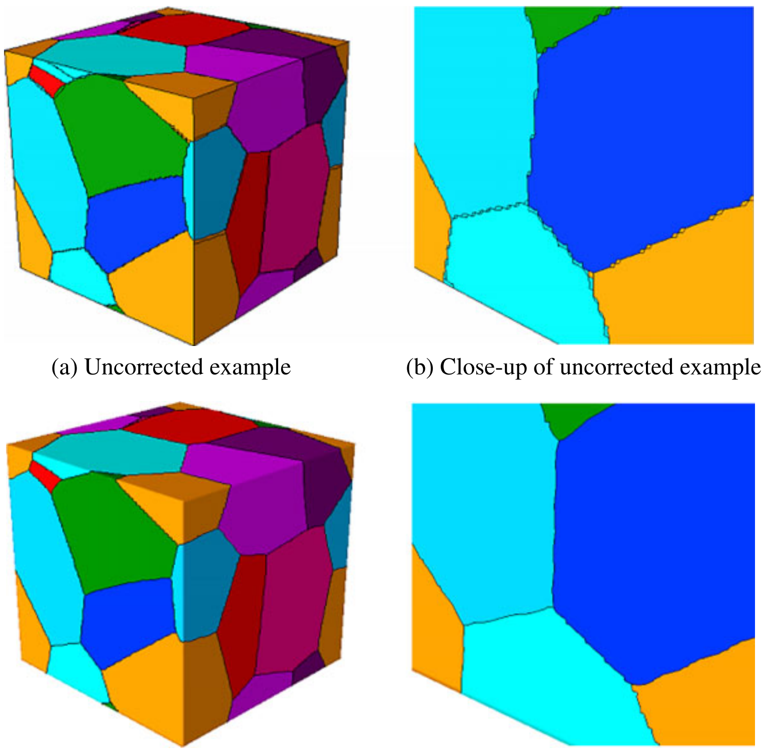
Fig. 6 Geometry smoothing and mesh coarsening

triangulation after smoothing and triangulation adjustment. It is the surface mesh in image (b) that is then coarsened to the mesh to be used for simulations, shown in image (c) using general mesh modification operations [16].

3.3 Non-manifold Models for Periodic Representative Volumes

Tools, such as the Microstructure builder [19], that construct microstructural geometries using limited numbers of image slices, often support features to define periodic representative volume elements (RVEs). The use of such periodic RVEs is advantageous when the microstructure analysis is to calculate material properties that will be used in a macro-scale simulation. The use of periodic RVEs requires that the geometric model have matching geometry and topology on opposing sides.

When there is periodicity, the voxel description of the microstructure is such that moving to the right from the rightmost voxel you end up in a voxel that is equivalent to the leftmost voxel (as expected, this means that there is effectively a copy of the RVE repeated in space). However, this means that the boundaries of the entire RVE are not periodic in the sense that they can be a voxel off which leads to the introduction on one voxel thin sliver faces at the intersection of the RVE boundary with a face between two grains. The top two images of Fig. 7 show an example of


(a) Uncorrected example (b) Close-up of uncorrected example

(c) Corrected example (d) Close-up of corrected example

Fig. 7 Elimination of boundary artifacts in RVE

these artifacts. In downstream processes, these small model entities lead to reduced mesh quality and higher computational load in finite element simulation.

To eliminate this problem a procedure was developed that, in effect, splits the first voxel plane in half and wraps it around, so as to place the periodic surface in the middle of this voxel plane. Doing this removes grain-grain boundaries spanning the outside boundary, thereby removing these one voxel misalignments. The bottom two images of Fig. 7 show the results using this procedure.

Small features can still occur at the outside/periodic boundary, depending on how the boundary interacts with the grains. Since for periodic RVEs the location of the boundary is arbitrary, a preprocessing step is carried out that allows the positions of the periodic unit cell boundaries to be adjusted so that the least number of small features are created. This step leads to improved quality in the final mesh and reduces the computational size of the subsequent finite element simulation.

For domains that are periodic the final step in non-manifold model construction is to identify which grain corresponds to each region in the model, so that suitable material models and boundary conditions can be applied. Because the input data set is periodic, each grain can intersect the boundary of the RVE and therefore end

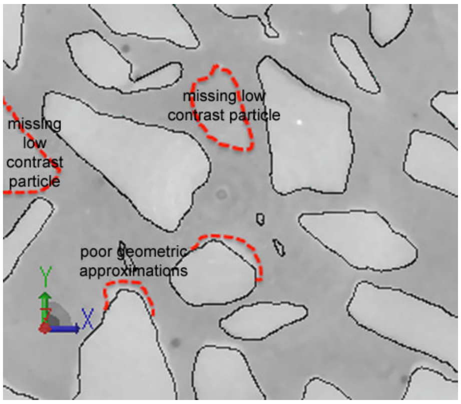

missing low contrast particle
missing low contrast particle
poor geometric approximations
Y
X

Fig. 8 Example of the ability to overlay geometry on an image to identify problems and verify the results

up with portions in multiple parts of the model. These parts need to be properly matched so the same material properties are assigned to each portion of the grain. For example, a simple test case was produced by Microstructure builder comprising 14 grains. The corresponding non-manifold model produced contains 84 model regions, or an average of six model regions per grain.

The approach used to track grain assignments updates the model conversion software to tag each face (triangle) in the initial tessellation with the id numbers of the grains on either side. After the model is constructed, the software transfers these grain numbers from the mesh faces to the corresponding model faces. For model regions intersecting the RVE boundary, the grain number can be obtained from the data attached to its model faces. For internal regions, a multiple pass algorithm was implemented to derive the grain number from the data previously transferred to the model faces plus already assigned regions. Using the 14-grain, 84-region example mentioned above, all model regions were resolved in two passes, with 78 regions resolved on the first pass and the remaining 6 on the second pass.

3.4 User Interface Functions to Support Image to Geometry Operations

Even with the level of effort that has gone into the development of good algorithms for the automatic construction of the non-manifold geometric model, there are cases where the image data is not of sufficient quality for the fully automatic execution of this process. Thus a user interface is available for the user to compare image data to the extracted non-manifold model data. One specific interface function allows the user to overlay the geometry along planes representing each slice onto the original image data. Figure 8 shows an example of this functionality on a slice of data that has been partially segmented using some preliminary settings. One can see where particles are not included in the geometry and where the geometry does and does not match well with the image. In cases like that in Fig. 8, one has to ability to alter the settings used by the segmentation procedures and can have them re-executed. While somewhat similar functionalities exists at the level of the segmentation tools to compare the segmentation with the image data, the tool here displays the intersection of the constructed geometric model with a plane at the location of the image slice. This opens up the possibility to make changes at the level of the geometry, for example through dragging control points for the surface until the geometry matches with the underlying image. Given a 3D visualization of the surface these operations can conceivably be done in three dimensional space, clearly superior to the manual editing of individual voxels for individual slices that segmentation tools provide.

Additional functions allow one to perform local geometric modifications to the geometry by introducing additional geometric features. Boolean operations (intersections, union) are supported such that the final geometric model is a valid non-manifold model that includes the feature and can be meshed. Such tools can be applied where there are a small number of modifications needed, or where one or several items that are not part of the scan need to be inserted to evaluate its impact during simulation. Examples are individual parts like implants, or simple "what if" scenarios where the user might want to introduce e.g. a certain amount of void or other particles. Procedures to determine various geometric properties are also available. Since all geometric entities representing the particles have a known relation to the segmentation of the original scan, geometric quantities like determination of the surface area and/or volume of selected particles or voids are readily available.

4 Mesh Generation

Once the geometry is fully defined, generation of a mesh requires determination of appropriate mesh control information to ensure that the initial mesh has mesh sizes, configurations, and gradations appropriate for the execution of the simulations, and mesh generation algorithms to automatically create a mesh that matches the specified mesh control information.

Although there is reasonable a priori geometry-based knowledge of areas where meshes should be refined (e.g., material faces and sharp geometric features), it is often desirable to be able to adapt the mesh during the analysis process based on simulation results information obtained from the analysis steps performed to that point. Thus the mesh adaptivity procedures need to include mesh refinement and coarsening that can be driven by mesh discretization error control and can account for the influence of geometric changes including large deformations, fracture or fragmentation, or evolving contact.

The automatic mesh generator used has been designed to generate valid meshes for general non-manifold objects [27, 29] which includes multi-material geometric models with voids acting through a direct interface with the native system using the abstraction of topological entities and their adjacencies, and the modeling system libraries to support the geometric interrogations needed by the automatic mesh generator [4, 27]. The automatic mesh generation [27, 29], and adaptive mesh modification procedures [28] have been integrated with the geometric modeling kernels of

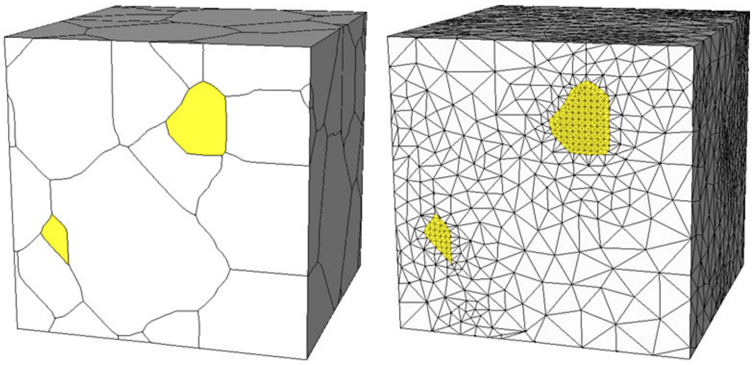


Fig. 9 Model and mesh of a polycrystal with finer mesh in selected crystals

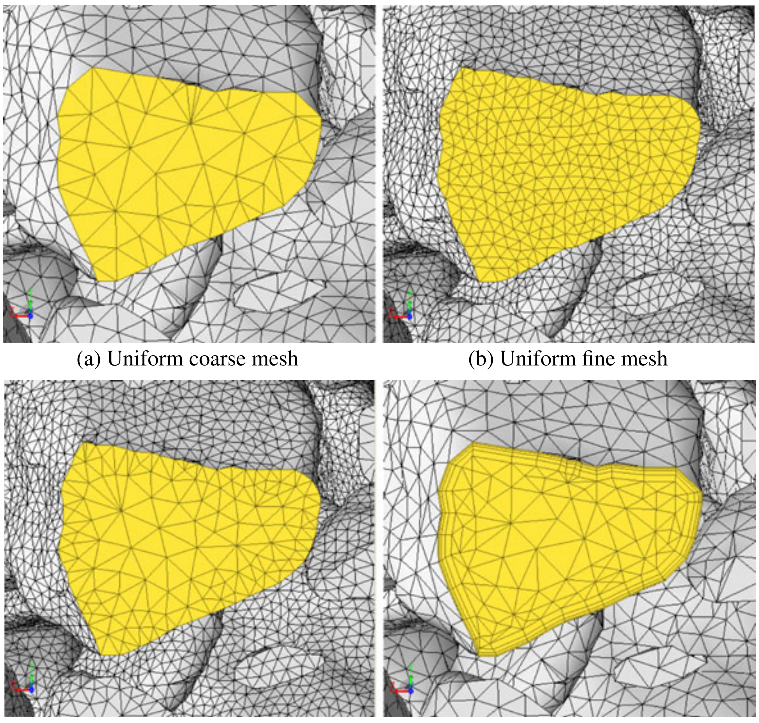
Siemens's NX, Dassault's CATIA V5, and PTC's Pro/Engineer CAD systems, and Spatial's ACIS, Siemens's Parasolid and PTC's Granite modeling kernels which are the basis for the majority of other CAD systems. The mesh generation and adaptation procedures also support discrete model representations such as mesh models for large deformation analyzes including multiple materials and fracture and classified voxel data for use in meshing microstructures.

A broad range of mesh control functions can be invoked for the generation of graded meshes where the mesh can be refined at critical geometric features and material interfaces and coarsened elsewhere. For example, Fig. 9 shows a mesh in which two particular grains are refined and the remaining mesh graded appropriately around those areas. The rate of gradation in mesh size can also be controlled.

Figure 10 shows some examples of the various types of mesh control available. These are close-up images of the concrete dataset showing where one of the aggregate pieces meets the boundary of the domain. These images show a range of mesh sizes applied to the entire domain and the interior faces as well as using boundary layer meshing to provide a graded mesh along the aggregate boundary. Anisotropic unstructured mesh gradation is supported when an anisotropic mesh metric field is specified. The definition of automatic procedures to set either gradations, boundary layers, anisotropic metrics or any combination of mesh specification at the material interfaces is easily supported by linking that information to the material interfaces in the non-manifold geometric model [29]. The mesh generators also include the ability to create periodic meshes from voxel data that is non-periodic. This can be used with procedures such as those in reference [15] to use non-periodic microstructural representations rather than requiring periodic unit cells.

Anisotropic mesh adaptation to account for discretization errors [16, 28] including procedures to maintain boundary layers [25], and/or evolving geometry [33] is based on local mesh modification of the model.

Figure 11 shows the overall workflow for an adaptive finite element analysis incorporating the described technologies. Note that besides the initial step of creating a discrete model from the voxel data input, the remaining part of the analysis workflow is identical to a process where the input is a CAD model. In both cases the


(a) Uniform coarse mesh (b) Uniform fine mesh

(c) Mesh graded with in the aggregate (d) Layered mesh at aggregate boundary

Fig. 10 Example mesh gradation for concrete data

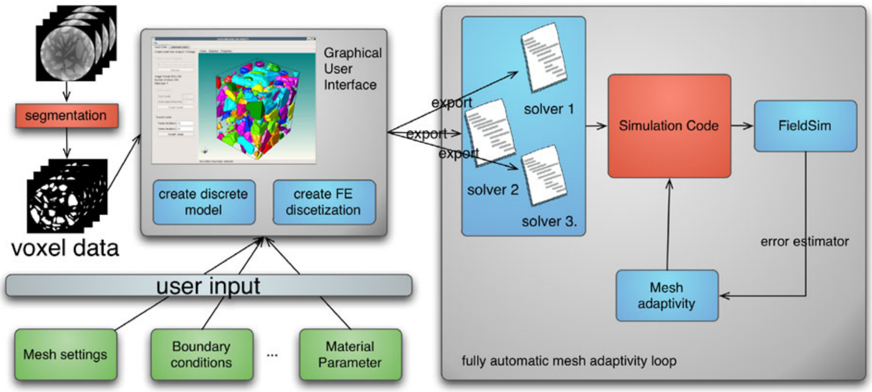


Fig. 11 Overall workflow

user will attribute the geometric model to apply non-geometric attribute information (meshing parameters, boundary conditions, material information etc.), export the problem setup to a Finite Element solver of their choice and perform simu-

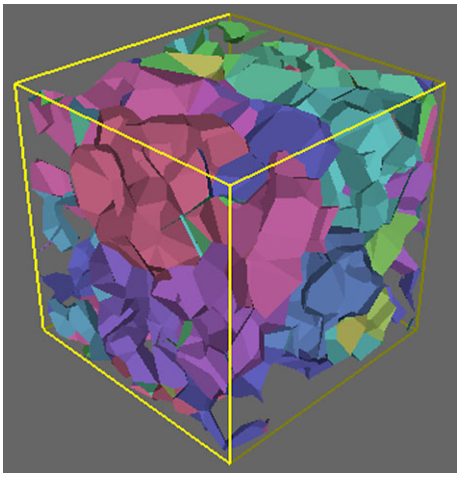


Fig. 12 Grain data set produced by Microstructure Builder

lations followed by error estimation and the construction of an adaptively refined mesh. That loop continues until the desired solution accuracy is obtained. In the work flow shown in Fig. 11, all steps within the blue boxes indicate Simmetrix technology, green boxes indicate user input, and red boxes indicate third party software technologies.

5 Results

The first example demonstrates the model and mesh generation starting from a Microstructure Builder dataset consisting of 25 grains. Figure 12 shows the dataset after the grains constructed by the Microstructure Builder were clipped to a cubical domain. Figure 13 shows the model and a mesh that was generated from the model. One geometric face on the boundary of the model and the corresponding mesh were

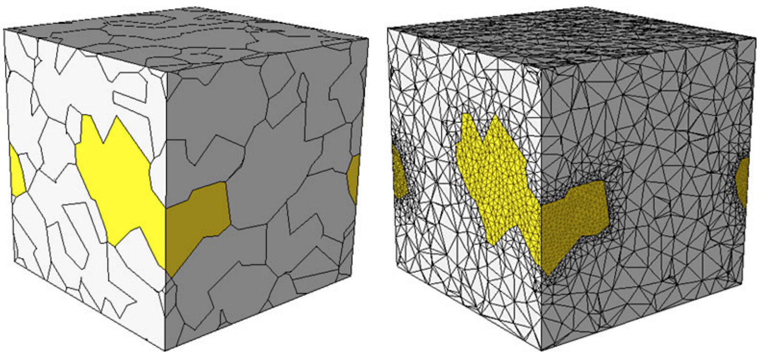


Fig. 13 Model and mesh generated from data set from Microstructure Builder

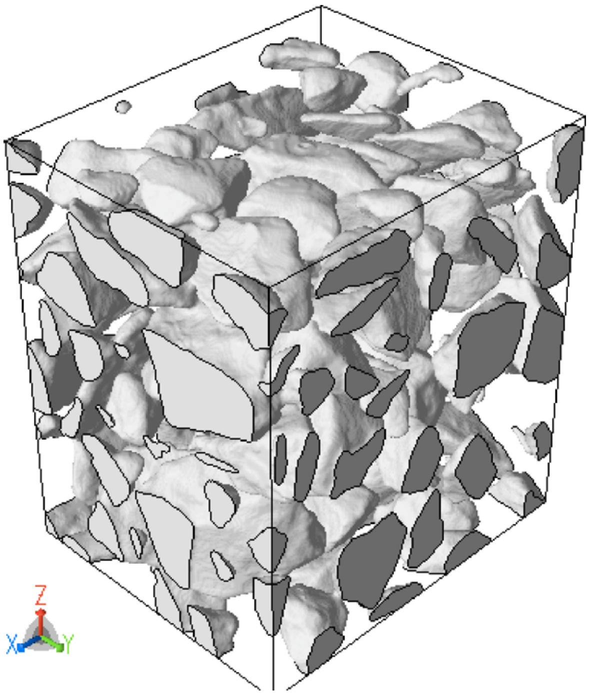


Fig. 14 Non-manifold model of image data set of concrete

highlighted to make it easier to see the periodicity condition across the boundary of the RVE. The Microstructure Builder generates periodic grain structures, and thus the periodicity is maintained in the model and meshes generated.

The second example starts from segmented data stored in an Analyze 7.5 file format. Analyze 7.5 is a file format commonly used to store MRI data [2]. For this example, the original data was given as a set of 678 slices stored as 16 bit TIFF files of 943 $\times$ 943 pixels resolution produced by a 3D scan of a cylindrical block of concrete. Figure 1 shows one slice of the input data set, from which a cubical domain was cut out from the center to avoid irregularities of the imaging data at the boundaries of the cylinder that was scanned. The segmentation of the data was performed using 3DSlicer [31]. Figure 14 shows the non-manifold model constructed for the cubical domain. At this stage no further processing has been done on the original voxel data set. In a next step the user removes the quantization artifacts using the discrete model smoothing operation described earlier by selecting the desired number of smoothing iterations (see previous section), and then proceeds to generate the finite element mesh (Fig. 15). It should be noted that the meshes shown for all examples in this paper are the surface meshes of otherwise full 3D tetrahedral meshes of the cubical domains. However, for the purpose of visualization only the surface mesh of the various geometric regions is shown.

The next image dataset was obtained from an XCMT scan of an aluminum foam sample. This dataset consists of 845 TIFF image files, each 950 $\times$ 950 pixels. An initial review shows that the gray-level histogram for this data has two distinct regions,

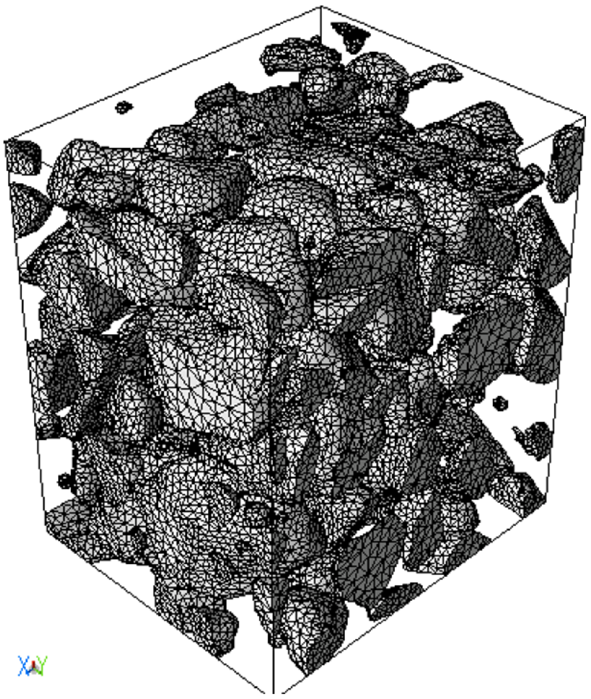


Fig. 15 Mesh for concrete image data

so that segmentation could be carried out using a single threshold value. Visual inspection also shows that this dataset has very high surface complexity relative to the sampling (voxel) size. Because of this the segmentation was directly used without additional processing which would have likely eliminated some of the small features, which in the case of such sharp image data do in fact exist. The model produced was a 100 $\times$ 100 $\times$ 100 subset of the aluminum foam data. Figure 16 shows the surface mesh generated including a close-up showing the level of detail contained in the model.

The last data set is a set of partially shattered glass beads enclosed in hydroxyl-terminated polybutadiene. The data set contained a total of 990 jpg images, each 954 $\times$ 915 pixels. It was downsampled by a factor of two to a final data set consisting of 495 images, each 477 $\times$ 458 pixels.

As it can be seen in Fig. 3 the contrast is very high between the glass beads and the material they were embedded in. For this example only the glass beads were segmented and it was assumed they are encapsulated in a homogenous material.

Given the high contrast of the pictures, the Simple Region Growing segmentation method available in 3DSlicer [31] was used to create the segmentation.

Figures 17 and 18 show the discrete model that was constructed from the segmentation. Note that the colors in those pictures indicate the separate regions that were detected, not the different materials found in the voxel data set (the glass beads were all segmented with just one label).

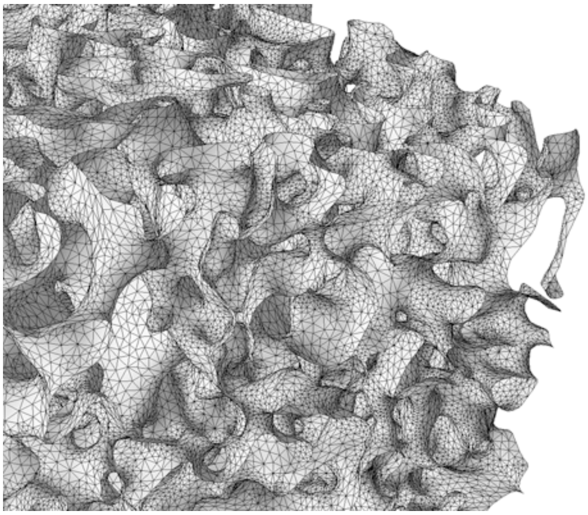


Fig. 16 Surface mesh detail for the aluminum foam image data set

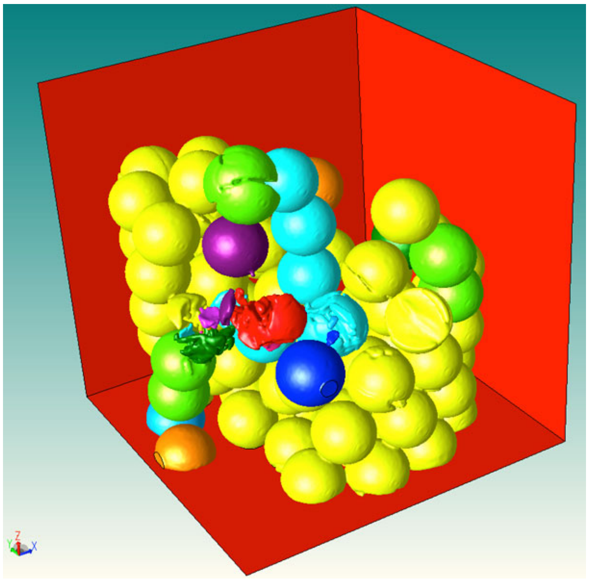


Fig. 17 Discrete model created from Glass Beads image data set

The discrete model was created by first eliminating small objects (any disconnected object with less than 180 voxels in volume), followed by an erosion/dilation step to eliminate small artifacts that are the size of a voxel in one or two dimensions. The discrete smoothing procedure described earlier was applied to the resulting discrete model to eliminate the voxel artifacts. At this point in the process the discrete model can be saved and, for the most part, treated as any other geometric model coming from CAD systems or geometric kernels for the purpose of creating simulation inputs.

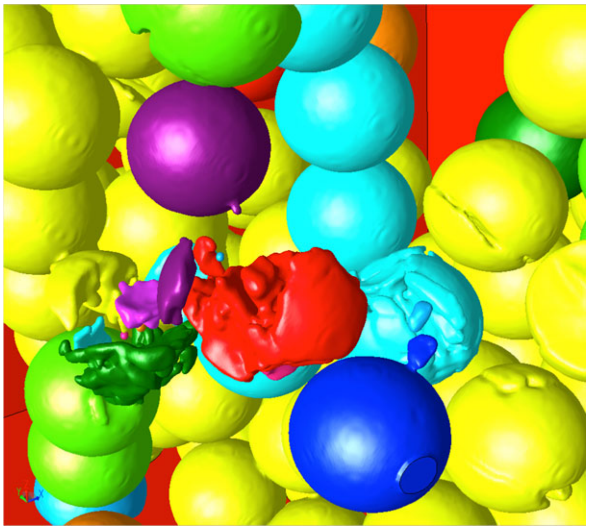


Fig. 18 Close up of discrete model create from Glass Beads image data set

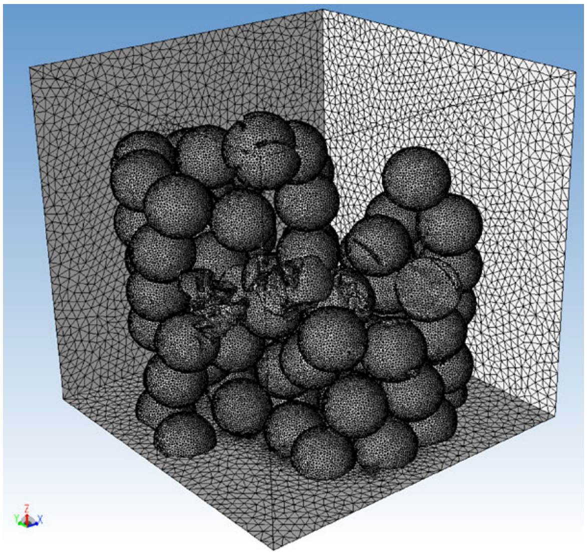


Fig. 19 Mesh for Large Glass Beads image data set

The mesh created from the discrete model is shown in Figs. 19 and 20. Meshing attributes were chosen such that local mesh refinement are applied on the interface between the glass beads and the material they are embedded in. The difference can be seen in Fig. 19 when the mesh of the glass beads is compared to the mesh on the outer faces of the enclosing cube.

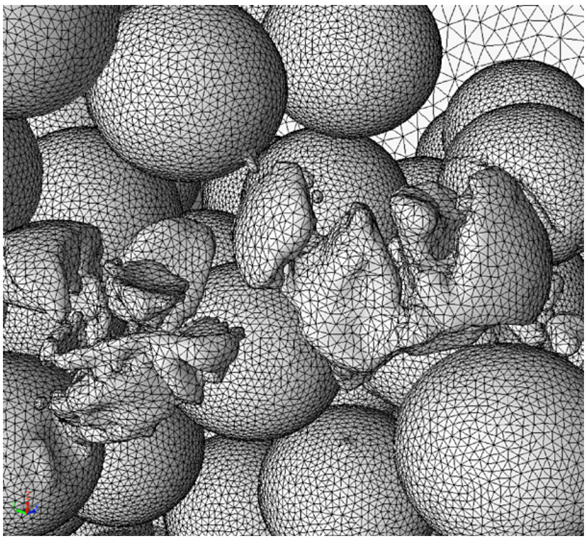


Fig. 20 Close up of mesh for Large Glass Beads image data set

6 Closing Remarks

This paper has presented a set of procedures for creating simulation ready unstructured meshes from image data of the microstructure of material systems. Given the image data for a given microstructure the steps in the process include:

- Performing the segmentation of the image data to identify the material regions.
- Constructing a simulation appropriate non-manifold geometric model, including the application of automated procedures to correct the non-manifold model as needed to properly represent the material system and to make the model more appropriate for the creation of meshes to be used in simulations. There are also options for the user to interact with the non-manifold model and original image data to compare them and modify the non-manifold model if desired.
- Automatic and/or user controlled specification of mesh control information as desired for the simulations to be performed. This is followed by fully automatic mesh generation.
- Execution of the desired simulation, which can include the application of adaptive mesh control (not covered in the current paper).

The process of constructing a proper non-manifold geometric model from the image data is a critical technical step in this process. Image data segmentation and the interactions of the resulting data with the procedures to construct the non-manifold geometric model is the most complex step in the process and the one that can benefit the most form the continued development of improved methods.

Acknowledgements The research reported in this document was performed in connection with contract W911QX-06-C-0040 with the U.S. Army Research Laboratory and contract FA8651-11-C-0200 with the Air Force Research Laboratory (AFRL).

References

1. Adams BL, Wright SI, Kunze K (1993) Orientation imaging—the emergence of a new microscopy. Metall Trans A 24:819–831
2. Analyze file format. http://www.grahamwideman.com/gw/brain/analyze/formatdoc.htm
3. Bankman I (ed) (2008) Handbook of medical image processing and analysis, 2nd edn. Elsevier, Amsterdam
4. Beall MW, Walsh J, Shephard MS (2004) A comparison of techniques for geometry access related to mesh generation. Eng Comput 20(3):210–221
5. Data set retrieved from http://www.lungcanceralliance.org
6. Flannery BP, Deckman HW, Roberge WG, D'Amico KL (1987) Three-dimensional X-ray microtomography. Science 237(4821):1439–1444
7. Garland M, Heckbert P (1997) Surface simplification using quadric error metrics. In: Proceedings of ACM SIGGRAPH, pp 209–216
8. Gering D, Nabavi A, Kikinis R, Grimson W, Hata N, Everett P, Jolesz F, Wells W (1999) An integrated visualization system for surgical planning and guidance using image fusion and interventional imaging. In: International conference on medical image computing and computer assisted intervention, pp 809–819
9. Gil JY, Kimmel R (2002) Efficient dilation, erosion, opening, and closing algorithms. IEEE Trans Pattern Anal Mach Intell 24(12):1606–1617
10. Groeber M, Ghosh S, Uchic MD, Dimidukc DM (2008) A framework for automated analysis and simulation of 3D polycrystalline microstructures. Part 1: statistical characterization. Acta Mater 56(6):1257–1273
11. Hoppe H (1996) Progressive meshes. In: Proceedings of ACM SIGGRAPH, pp 99–108
12. Ibanez L, Schroeder W, Ng L, Cates J (2005) The ITK software guide: the insight segmentation and registration toolkit (version 1.4). Kitware, Inc., Clifton Park
13. Ju T, Losasso F, Schaefer S, Warren J (2002) Dual contouring of hermite data. In: Proceedings of SIGGRAPH, pp 339–346
14. Krysl P, Ortiz M (2001) Extraction of boundary representation from surface triangulations. Int J Numer Methods Eng 50:1737–1758
15. Kumar RS, Wang AJ, McDowell DL (2006) Effects of microstructure variability on intrinsic fatigue resistance of nickel-base superalloys—a computational mechanics approach. Int J Fract 137:173–210
16. Li X, Shephard MS, Beall MW (2005) 3-D anisotropic mesh adaptation by mesh modifications. Comput Methods Appl Mech Eng 194(48–49):4915–4950
17. Lin FX, Godfrey A, Juul Jensen D, Winther G (2010) 3D EBSD characterization of deformation structures in commercial purity aluminum. Mater Charact 61:1203–1210
18. Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of SIGGRAPH, pp 163–169
19. Microstructural Builder web page. http://code.google.com/p/mbuilder
20. Moore RH, Rohrer GS, Saigal S (2009) Reconstruction and simplification of high-quality multiple-region models from planar sections. Eng Comput 25(3):221–235
21. Owen SJ, White DR (2003) Mesh-based geometry. Int J Numer Methods Eng 58:375–395
22. Pieper S, Halle M, Kikinis R (2004) 3D Slicer. In: Proceedings of the 1st IEEE international symposium on biomedical imaging: from nano to macro, vol 1, pp 632–635
23. Pieper S, Lorensen B, Schroeder W, Kikinis R (2006) The NA-MIC Kit: ITK, VTK, pipelines, grids and 3D slicer as an open platform for the medical image computing community. In: Proceedings of the 3rd IEEE international symposium on biomedical imaging: from nano to macro, vol 1, pp 698–701
24. Rollett AD, Lee S-B, Campman R, Rohrer GS (2007) Three-dimensional characterization of microstructure by electron back-scatter diffraction. Annu Rev Mater Res 37:627–658
25. Sahni O, Jansen KE, Shephard MS, Taylor CA, Beall MW (2008) Adaptive boundary layer meshing for viscous flow simulations. Eng Comput 24(3):267–285

26. Schaefer S, Ju T, Warren J (2007) Manifold dual contouring. IEEE Trans Vis Comput Graph 13(3):610–619
27. Shephard MS (2000) Meshing environment for geometry-based analysis. Int J Numer Methods Eng 47(1–3):169–190
28. Shephard MS, Flaherty JE, Jansen KE, Li X, Luo X-J, Chevaugeon N, Remacle J-F, Beall MW, O'Bara RM (2005) Adaptive mesh generation for curved domains. Appl Numer Math 52(2–3):251–271
29. Simmetrix, Inc. http://www.simmetrix.com/
30. Simple Region Growing webpage. http://www.slicer.org/slicerWiki/index.php/Modules:Simple_Region_Growing-Documentation-3.4
31. Slicer web page. http://www.slicer.org
32. Verdu-Monedero R, Angulo J, Serra J (2011) Anisotropic morphological filters with spatially-variant structuring elements based on image-dependent gradient fields. IEEE Trans Image Process 20(1):200–212
33. Wan J, Kocak S, Shephard MS (2005) Automated adaptive 3-D forming simulation process. Eng Comput 21(1):47–75
34. Weiler K (1988) The radial edge structure: a topological representation for non-manifold geometric modeling. In: Geometric modeling for CAD applications, pp 3–36
35. Zhang Y, Hughes TJR, Bajaj CL (2010) An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Eng 199(5–8):405–415

Quality Improvement of Segmented Hexahedral Meshes Using Geometric Flows

Juelin Leng, Guoliang Xu, Yongjie Zhang, and Jin Qian

Abstract This paper presents a new quality improvement algorithm for segmented quadrilateral/hexahedral meshes which are generated from multiple materials. The proposed algorithm combines mesh pillowing, curve and surface fairing driven by geometric flows, and optimization-based mesh regularization. The pillowing technique for quadrilateral/hexahedral meshes is utilized to eliminate doublets with two or more edges/faces located on boundary curves/surfaces. The non-manifold boundary for multiple materials is divided into several surface patches with common curves. Then curve vertices, surface vertices, and interior vertices are optimized via different strategies. Various geometric flows for surface smoothing are compared and discussed as well. Finally, the proposed algorithm is applied to three mesh datasets, the resulting quadrilateral meshes are well smoothed with volume and feature preserved, and hexahedral meshes have desirable Jacobians.

1 Introduction

In many applications such as computer graphics, finite element analysis and numerical simulations, 3D objects are usually discretized as polygonal meshes, typically tetrahedral and hexahedral meshes. For example, in biomedical modeling and material property analysis, 3D objects are often partitioned into multiple domains

J. Leng · Y. Zhang (✉) · J. Qian
Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: jessicaz@andrew.cmu.edu

J. Qian
e-mail: jq@andrew.cmu.edu

J. Leng · G. Xu
ICMSEC, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100190, China

J. Leng
e-mail: lengjl@lsec.cc.ac.cn

G. Xu
e-mail: xuguo@lsec.cc.ac.cn

according to different physical/chemical attributes, or material properties. In computational simulations, quadrilateral and hexahedral meshes are often preferred [9]. For a segmented domain, the hexahedral mesh for each component composes the whole hexahedral mesh with conforming quadrilateral meshes on common surfaces. The union of boundary meshes for all components forms a non-manifold quadrilateral mesh. Compared to traditional mesh improvement problem for single-material domains, the problem is much more challenging for segmented meshes.

In this paper, we will focus on quality improvement of quadrilateral/hexahedral meshes for multiple materials. The pillowing technique for quadrilateral/hexahedral meshes is utilized to remove doublets. Then hexahedral mesh vertices are categorized into four types: fixed vertices, curve vertices, surface vertices, and interior vertices; while quadrilateral meshes only contain the first three types. Curve vertices and surface vertices are modified along the tangent directions to regularize the non-manifold boundary mesh. Moreover, geometric flows are applied for curve fairing and surface smoothing, which relocate curve and surface vertices along the normal directions. We will apply four typical geometric flows for surface smoothing, and discuss their effectivity of evolving the surface. Then the best feature-preserving geometric flow will be selected in our quality improvement algorithm. Interior vertices in hexahedral meshes are relocated via an optimization-based method. Finally, the proposed algorithms are validated on three application examples.

The rest of this paper is organized as follows. Section 2 reviews related previous work. We describe the quality improvement problem of quadrilateral/hexahedral meshes for multiple materials in Sect. 3. Section 4 presents algorithms and implementation details for quality improvement. We give several experimental results in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Previous Work

Hexahedral Mesh Generation The existing methods for unstructured hexahedral mesh generation can be grouped into four categories [15]: grid-based [18, 21, 22], medial surface [13], plastering [3], and whisker weaving [19]. The grid-based approaches generate a three dimensional grid of hexahedral elements in the interior of the domain. Grid-based methods are robust, but tend to generate poor quality elements near the boundary. Medial surface methods divide the whole domain into map-meshable regions by a set of medial surfaces, and then a series of templates are utilized to fill those regions. Plastering methods start with the boundaries, new hexahedra are attached to the meshing front until the volume is completely meshed. Whisker weaving builds the combinatorial dual of the mesh, then the dual mesh is converted into the primal mesh, and finally embedded into the given domain.

For multiple materials, mesh generation is a much more challenge problem. In [24], an octree-based isocontouring method [21, 22] was extended to multiple-material regions. However, the generated hexahedral meshes always have poorly shaped elements near the boundary.

Quality Improvement for Quadrilateral/Hexahedral Meshes The pillowing technique [14] was proposed to remove the cases that two neighboring elements share two edges/faces by inserting new vertices. Relocating mesh vertices is another popular approach to improve mesh quality. Laplacian smoothing [6] which relocates vertices to the arithmetic average of its neighboring vertices is simple and inexpensive, but it does not guarantee an improvement of the mesh quality and also results in degraded or inverted elements. Therefore, a number of optimization-based methods [8, 10, 11] were developed to improve the mesh quality by optimizing an objective function which reflects the element quality. Most of these improvement approaches were designed for manifold meshes. Due to the complexity of segmented meshes, quality improvement for segmented meshes is much more challenging.

Surface Smoothing Using Geometric Flows Geometric flows have been successfully used for surface modeling and designing because they are good at controlling geometric shape evolution. In the process of surface evolution, the geometric partial differential equations (PDEs) are discretized on a given mesh. On the other hand, geometric flows can also be used to fair zigzag meshes. In [4], an approach was described to fair meshes with rough features using diffusion and curvature flows. Surface diffusion flow and averaged mean curvature flow were used to smooth surface meshes in [16, 23] and [12], respectively.

In our previous paper [12], we have proposed an geometric flow-based method for quality improvement of segmented tetrahedral meshes. The experimental results demonstrate the proposed method is effective. The generalization of improvement approach from tetrahedral meshes to hexahedral meshes is not straightforward, since a hexahedron has higher flexibility to become extremely distorted. In this paper, we will focus on quality improvement of quadrilateral/hexahedral meshes.

3 Problem Description and Preparation

In this section, we first provide the problem description of quality improvement for quadrilateral/hexahedral meshes, and then classify mesh vertices into four groups. Before introducing our mesh improvement algorithms, we should select proper quality metrics.

3.1 Problem Description

For a given mesh, quality improvement aims to make each element of the mesh has an optimal shape. A segmented hexahedral mesh for multiple regions is composed of several separated sub-meshes for each component with conforming boundaries. The union of component boundaries forms a complicated non-manifold quadrilateral mesh. Due to the complexity of segmented meshes, quality improvement is much

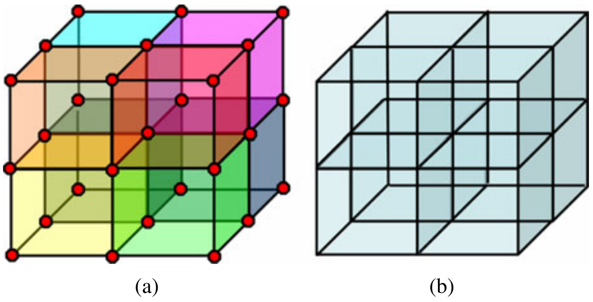


Fig. 1 An illustration example of a multi-material domain. (**a**) is a cube consists of eight components, which are marked by different colors. (**b**) is the non-manifold lattice boundary made up of eight component boundaries

more intractable than the traditional improvement problem for single-material regions. The quadrilateral/hexahedral mesh for a segmented domain is referred as high quality, if all the elements are well shaped, and the boundary surfaces are smooth. In this paper, we intend to develop a novel geometric flow-based approach to optimize hexahedral elements in each component, and improve non-manifold quadrilateral boundary meshes simultaneously.

To simplify the non-manifold boundary, we divide the whole boundary into several surface patches sharing common boundary curves with each other. Here, the common surface shared by any two components is referred as a boundary surface patch, and the exterior boundary of each component is regarded as a boundary surface patch as well. The common curve of any two surfaces is defined as a boundary curve. As shown in Fig. 1, the cube is composed of eight small cubes representing different materials. The common faces shared by any pair of neighboring cubes are called surface patches, and black lines with red end points are boundary curves. Therefore, the boundary smoothing problem is converted to fairing and regularizing curves and surface patches.

Due to the complexity of meshes for multiple regions, we categorize mesh vertices into the following four groups:

Interior vertices: Vertices inside one volumetric component.

Surface vertices: Manifold vertices on boundary surface patches, which can move along the normal direction to smooth the surface, and can also move along the tangent direction to improve the Jacobian.

Curve vertices: Vertices located on boundary curves, which can only move along the tangent direction.

Fixed vertices: End points of boundary curves and other non-manifold vertices, which are fixed during the mesh improvement process.

Then we will handle various vertices using different algorithms. For quadrilateral meshes, there are only three types of vertices: surface vertices, curve vertices, and fixed vertices.

3.2 Quality Metrics

Various quantities have been used to measure the shape or quality of a hexahedron. Here, we choose the determinant and the condition number of Jacobian matrix [7] as quality metrics for hexahedral meshes.

Let H be a hexahedron with eight vertices $\mathbf{x}_{ijk}$ $(i, j, k = 0, 1)$, then the hexahedron can be represented as a trilinear parametric volume defined on a unit cube,

$$\mathbf{x}(u, v, w) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} u^i(1-u)^{1-i}v^j(1-v)^{1-j}w^k(1-w)^{1-k}\mathbf{x}_{ijk}. \tag{1}$$

The Jacobian matrix

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

describes the linear transformation from the ideal shape (unit cube) to hexahedron H. If the determinant of the Jacobian matrix at all the eight vertices are positive, then the hexahedron is valid, otherwise, the hexahedron is regarded as inverted. We call the determinant of Jacobian matrix as *Jacobian*, and the determinant of the column-normalized Jacobian matrix as the *scaled Jacobian*.

The condition number of the Jacobian matrix is defined as $\kappa(J) = \frac{1}{3}\|J\|_F \times \|J^{-1}\|_F$, where $\|J\|_F = [\mathrm{tr}(J^T J)]^{1/2}$ denotes the Frobenius norm. It is easy to derive that $\kappa(J) = \frac{1}{3}\sqrt{\sum_{i,j}(\sigma_i/\sigma_j)^2} \geq 1$ is a metric with respect to the singular values $\{\sigma_i\}_{i=1}^3$ of the Jacobian matrix. The condition number reaches minimum iff $\sigma_1 = \sigma_2 = \sigma_3$.

For a quadrilateral $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$, we define the following metric

$$J(\mathbf{x}_j) = \det(\mathbf{x}_{j+1} - \mathbf{x}_j, \mathbf{x}_{j+3} - \mathbf{x}_j, \mathbf{n}_j)$$

named the *Jacobian* for each vertex, where "det" denotes determinant, the subscript of $\mathbf{x}_j$ is in module of 4, and $\mathbf{n}_j$ is the unit normal vector at vertex $\mathbf{x}_i$. The corresponding *scaled Jacobian* is $\det(\frac{\mathbf{x}_{j+1}-\mathbf{x}_j}{\|\mathbf{x}_{j+1}-\mathbf{x}_j\|}, \frac{\mathbf{x}_{j+3}-\mathbf{x}_j}{\|\mathbf{x}_{j+3}-\mathbf{x}_j\|}, \mathbf{n}_j)$.

4 Quality Improvement Algorithm and Implementation

Our quality improvement algorithm is composed of four parts: pillowing, boundary curve fairing and regularization, boundary surface fairing and regularization, and volume mesh optimization. The pillowing technique is used to remove the hexahedra with two or more faces on the boundary surface. To fair a curve/surface mesh, the vertices are relocated along its normal direction to make the curve/surface as smooth as possible. To regularize a curve/surface mesh, we intend to modify the

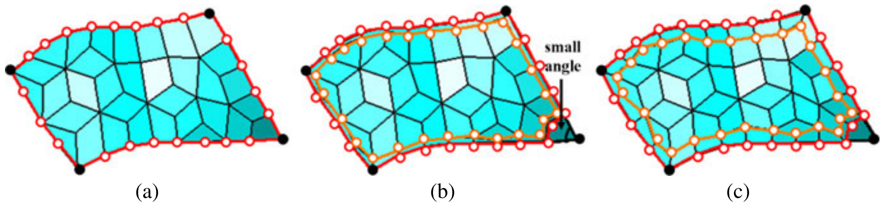


Fig. 2 Procedure of the pillowing technique applied on a surface patch. (**a**) is a given surface patch, *red curves with black end points* are boundary curves; (**b**) shows the shrink set (*red*) and a pillowed layer (*orange*); and (**c**) the inserted layer is dragged inside using the regularization technique

vertices along the tangent direction such that each quadrilateral element becomes similar to a square.

4.1 The Pillowing Technique

In a quadrilateral mesh, a doublet occurs when two elements share two edges. For non-manifold boundary, if a quadrilateral has more than two edges located on a boundary curve, we regard it as a doublet. Doublets will result in poor quality elements, and one effective method is to change the connectivity of doublet vertices. The pillowing technique can be used to improve the mesh quality by inserting one layer around the boundary curves [14]. Since the whole boundary has been divided into several manifold surface patches, mesh pillowing can be operated on each surface patch independently.

Figure 2 shows the pillowing procedure for a surface patch. First, we set the whole surface patch as a shrink set. If there is a quadrilateral with two edges forming a small angle on boundary curves (Fig. 2(b)), it would be excluded from the shrink set. The shrink set boundary is the outer layer. Second, we create a parallel layer which is a shrinkage of the outer layer. Vertex connections in the shrink set with respect to the outer layer vertices are replaced with the corresponding newly added vertices. Then, each newly added vertices is connected to its corresponding vertex on the outer layer to fill the gap between the two layers. Finally, we utilize the regularization technique introduced later to drag the inserted layer inside so as to improve the mesh quality.

For hexahedral meshes, the pillowing technique is also a popular approach to remove doublets so that any two elements have at most one common face. The pillowing idea can be generalized to eliminate the doublet that one hexahedron has two or more faces on the mesh boundary. In segmented hexahedral meshes, this kind of doublet is much more common and results in low quality elements.

We apply the pillowing algorithm (Algorithm 1) to pillow each component of the segmented meshes. Since one component shares boundary with other components, the shrink set does not shrink actually. We insert inner vertices on the pillowed layer without interfering the common boundary mesh. Figure 3 shows a simple illustration

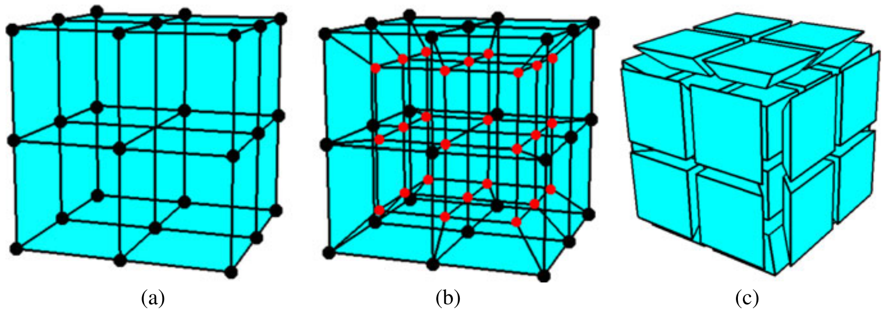


Fig. 3 An illustration of pillowing a cube component. (**a**) A hexahedral mesh of a cube; (**b**) the pillowed mesh, the *black vertices* are on the outer layer, and the *red vertices* are newly added; and (**c**) hexahedral elements after pillowing

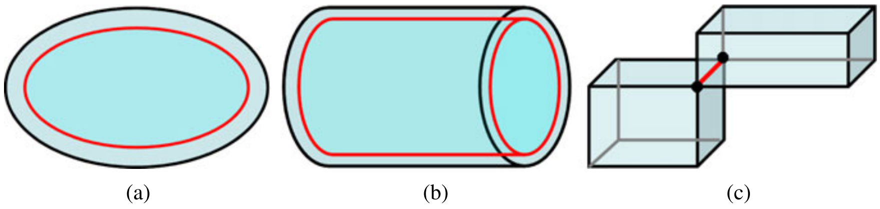


Fig. 4 (**a**) The outer layer (*black*) and the pillowed layer (*red*) of a closed component; (**b**) the outer layer (*black* one) and the pillowed layer (*red*) of an open component; and (**c**) hexahedra surrounding the non-manifold edge (*red*) should be eliminated from the shrink set

of hexahedral mesh pillowing. It can be seen that all the hexahedra have at most one face on the boundary after pillowing.

Algorithm 1 (Pillowing one component of the segmented hexahedral meshes)

1. Find the shrink set and the outer layer of the component.
   a. Set the whole component as the shrink set;
   b. For a closed component (see Fig. 4(a)), the outer layer is just the shrink set boundary; for an open component (see Fig. 4(b)), the outer layer is open as well; and
   c. Find out non-manifold boundary edges (see Fig. 4(c)), and eliminate all the hexahedra surrounding these edges from the shrink set.
2. Mesh pillowing.
   a. Create a copy of the outer layer, which is the pillowed layer. Shrink the pillowed layer along the normal direction toward the interior of the component;
   b. Loop for each hexahedron contained in the shrink set, replace the outer layer vertices by the corresponding vertices on the pillowed layer. Hence, there forms a gap between the two layers; and

   c. Fill the gap by connecting each pair of opposite vertices on the two layers, and obtain several new hexahedra sandwiched between the outer layer and the pillowed layer.
3. Update the data information such as vertex type, face neighbor, and hexahedron neighbor.

4.2 Curve Smoothing Driven by Curve Diffusion Flow

Let $[\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_n]$ be a boundary curve with two fixed end points $\mathbf{x}_0$ and $\mathbf{x}_n$. To fair the curve, we introduce a temporal variable t, and evolve the curve along the normal direction at a speed with respect to curvature. Simply choosing the curvature as the speed can fair the curve but can not preserve shape features. Here, we construct a shape-preserving curve diffusion flow to evolve the curve,

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = -\big[(\Delta\boldsymbol{\kappa}_i)^T\mathbf{n}_i\big]\mathbf{n}_i, \quad i = 1,\ldots,n-1, \tag{2}$$

where

$$\boldsymbol{\kappa}_i = \frac{\mathbf{t}_{i+1}-\mathbf{t}_i}{s_i}, \qquad \mathbf{n}_i = \frac{\boldsymbol{\kappa}_i}{\|\boldsymbol{\kappa}_i\|}, \tag{3}$$

$$s_i = \frac{\|\mathbf{x}_i-\mathbf{x}_{i-1}\| + \|\mathbf{x}_i-\mathbf{x}_{i+1}\|}{2}, \qquad \mathbf{t}_i = \frac{\mathbf{x}_i-\mathbf{x}_{i-1}}{\|\mathbf{x}_i-\mathbf{x}_{i-1}\|}, \tag{4}$$

and Δ is the Laplace operator. $\mathbf{n}_i$ is a discretization of the normal direction at vertex $\mathbf{x}_i$, and $\|\boldsymbol{\kappa}_i\|$ is the corresponding curvature.

Equation (2) can be solved using the explicit Euler scheme

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} - \tau\big[(\Delta\boldsymbol{\kappa}_i)^T\mathbf{n}_i\big]\mathbf{n}_i, \quad i = 1,\ldots,n-1, \tag{5}$$

where τ is a temporal step-size, $\mathbf{x}_i^{(0)} = \mathbf{x}_i$, and $\mathbf{x}_0^{(k)} = \mathbf{x}_0^{(k+1)} = \mathbf{x}_0$, $\mathbf{x}_n^{(k)} = \mathbf{x}_n^{(k+1)} = \mathbf{x}_n$. $\boldsymbol{\kappa}_i$ and $\mathbf{n}_i$ are defined in Eq. (3) by taking $\mathbf{x}_i = \mathbf{x}_i^{(k)}$, $i = 1,\ldots,n-1$. $\Delta\boldsymbol{\kappa}_i$ is discretized as $(\frac{\boldsymbol{\kappa}_{i+1}-\boldsymbol{\kappa}_i}{\|\mathbf{x}_{i+1}-\mathbf{x}_i\|} - \frac{\boldsymbol{\kappa}_i-\boldsymbol{\kappa}_{i-1}}{\|\mathbf{x}_i-\mathbf{x}_{i-1}\|})/s_i$, with $i = 1,\ldots,n-1$, $\boldsymbol{\kappa}_0$ and $\boldsymbol{\kappa}_n$ are taken as zero vectors.

4.3 Curve Regularization

The boundary curve $[\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_n]$ is referred as regular if vertices are uniformly distributed on the curve. Therefore, it can be regularized by minimizing the following energy functional

$$\mathcal{E}(\mathcal{C}) = \frac{1}{2}\sum_{i=1}^{n}\big(\|\mathbf{x}_i-\mathbf{x}_{i-1}\| - h\big)^2, \tag{6}$$

where $h = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{x}_{i-1}\|$ is the averaged length of each two neighboring vertices. At each free vertex $\mathbf{x}_i$ of the curve, we vary $\mathbf{x}_i$ as $\mathbf{x}_i \to \mathbf{x}_i + \varepsilon_i \Phi_i$, $\Phi_i \in \mathbb{R}^3$, $i = 1, \ldots, n-1$. Then we obtain the first-order variation

$$\begin{aligned}\delta(\mathcal{E}, \Phi_i) &= \left.\frac{\partial \mathcal{E}(\mathcal{C}, \varepsilon_i)}{\partial \varepsilon_i}\right|_{\varepsilon_i=0} \\ &= \big(\|\mathbf{x}_{i+1} - \mathbf{x}_i\| - h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i+1})}{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|} + \big(\|\mathbf{x}_i - \mathbf{x}_{i-1}\| - h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i-1})}{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|}.\end{aligned}$$

To keep the curve shape, Φ_i is chosen as $\mathbf{e}_i$ which is the unit tangential direction at $\mathbf{x}_i$, then a set of L^2-gradient flows are derived as

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} + \delta(\mathcal{E}, \mathbf{e}_i)\mathbf{e}_i = \mathbf{0}, \tag{7}$$

$i = 1, \ldots, n-1$. The discretization of Eq. (7) can be written as

$$\begin{aligned}&\frac{\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}}{\tau} + \big(\|\mathbf{x}_{i+1}^{(k)} - \mathbf{x}_i^{(k)}\| - h\big)\frac{\mathbf{e}_i\mathbf{e}_i^T(\mathbf{x}_i^{(k)} - \mathbf{x}_{i+1}^{(k)})}{\|\mathbf{x}_i^{(k)} - \mathbf{x}_{i+1}^{(k)}\|} \\ &+ \big(\|\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k)}\| - h\big)\frac{\mathbf{e}_i\mathbf{e}_i^T(\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k)})}{\|\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k)}\|} = 0.\end{aligned} \tag{8}$$

The initial value is chosen as $\mathbf{x}_i^{(0)} = \mathbf{x}_i$. Each $\mathbf{e}_i$ is calculated as the unit tangent direction of a fitting quadratic curve with respect to $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$.

4.4 Surface Smoothing Using Various Geometric Flows

Geometric flows have been successfully used in surface modeling since they are inherently good at controlling geometric shape evolution. Let S_0 be a piece of compact orientable surface in $\mathbb{R}^3$ with the boundary denoted as Γ. Introducing the temporal variable t, the surface evolution can be formularized as

$$\frac{\partial \mathbf{x}(t)}{\partial t} = V_n(\mathbf{x})\mathbf{n}(\mathbf{x}), \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{9}$$

where $\mathbf{x}(t)$ is surface point located on $S(t)$, $V_n(\mathbf{x})$ denotes the normal velocity on $S(t)$ at $\mathbf{x}$, and $\mathbf{n}(\mathbf{x})$ stands for the unit normal. Since the velocity $V_n(\mathbf{x})$ usually represents several geometric quantities which reflect geometric properties of the evolving surface, Eq. (9) is referred as a geometric flow.

Various geometric flows can be constructed to meet different application requirements by choosing an appropriate normal velocity $V_n(\mathbf{x})$. Curvature is an important descriptor reflecting the flexibility of surface, hence geometric PDEs are basically expressed by curvatures. The most common used geometric flows include Mean Curvature Flow (MCF), Averaged Mean Curvature Flow (AMCF), Surface Diffusion Flow (SDF) and Willmore Flow (WF). MCF can be used to get the minimum surface with respect to fixed boundary. AMCF and SDF are volume-preserving during the evolution. At first, we would like to introduce definitions of these four well-used geometric flows [17, 20].

Definition 1 (Mean curvature flow (MCF))

$$\frac{\partial \mathbf{x}}{\partial t} = 2\mathbf{H}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{10}$$

where **H** denotes the mean curvature vector. MCF is an area-reducing flow, which can be used to get the minimum surface with respect to a fixed boundary.

Definition 2 (Averaged mean curvature flow (AMCF))

$$\frac{\partial \mathbf{x}}{\partial t} = \big[H - h(t)\big]\mathbf{n}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{11}$$

where

$$h(t) = \int_S H \,\mathrm{d}A \Big/ \int_S \mathrm{d}A.$$

Since $h(t)$ is the average of the mean curvature H on the whole surface, hence Eq. (11) is called the averaged mean curvature flow [5].

Definition 3 (Surface diffusion flow (SDF))

$$\frac{\partial \mathbf{x}}{\partial t} = -2\Delta_s H\mathbf{n}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{12}$$

where Δ_s is the Laplace–Beltrami operator. It is an area-reducing and volume-preserving flow which can be used for noise removing in surface design.

Definition 4 (Willmore flow (WF))

$$\frac{\partial \mathbf{x}}{\partial t} = -\big[\Delta_s H + 2H\big(H^2 - K\big)\big]\mathbf{n}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{13}$$

where H and K are the mean curvature and Gaussian curvature, respectively. WF has been investigated and used widely in computational geometry and other fields. Suppose the initial surface is a sphere, WF can keep the sphere without evolution.

The following theorems [20] describe area variation and volume variation of the evolving surface, respectively.

Theorem 1 *Let $V(t)$ denote the (directional) volume of the region enclosed by $S(0)$ and $S(t)$ (see Fig. 5 for a 2D curve case). Then we have*

$$\frac{dV(t)}{dt} = \int_{S(t)} V_n(\mathbf{x})\,\mathrm{d}A. \tag{14}$$

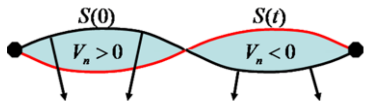


Fig. 5 The directional area between the curves $S(0)$ and $S(t)$. The area of the region with normal velocity $V_n > 0$ (or $V_n < 0$)

Taking $V_n = H(t) - h(t)$, where $h(t) = \int_{S(t)} H \, \mathrm{d}A / \int_{S(t)} \mathrm{d}A$, then we have

$$\frac{dV(t)}{\mathrm{d}t} = \int_{S(t)} \big(H(\mathbf{x}) - h(t)\big) \,\mathrm{d}t = \int_{S(t)} H \,\mathrm{d}A - h(t) \int_{S(t)} \mathrm{d}A = 0.$$

Hence AMCF is volume-preserving. Similarly, with $V_n = -2\Delta_s H$,

$$\frac{\mathrm{d}}{\mathrm{d}t} V(t) = -\frac{2}{3} \int \mathrm{div}_s(\nabla_s H) \,\mathrm{d}A = \frac{2}{3} \int (\nabla_s H)^{\mathrm{T}} \nabla_s(1) \,\mathrm{d}A = 0,$$

where div_s and ∇_s are the tangential divergence operator and the tangential gradient operator, respectively. Thus, SDF is volume-preserving as well.

Theorem 2 *Let $A(t)$ be the are $S(t)$, then we have*

$$\frac{\mathrm{d}A(t)}{\mathrm{d}t} = -\int_{S(t)} V_n(\mathbf{x})^{\mathrm{T}} \mathbf{H} \,\mathrm{d}A. \tag{15}$$

For MCF, we have

$$\frac{\mathrm{d}A(t)}{\mathrm{d}t} = -2 \int_{S(t)} H^2 \,\mathrm{d}A < 0,$$

which means the surface area keeps reducing until the mean curvature $H = 0$ all over the surface. Hence, the steady solution depends upon the fixed boundary curves, while the enclosed surface will shrink to a point. SDF is another area-reducing flow, unlike MCF, SDF decreases the surface area until H is constant. WF is a gradient flow corresponding to the Willmore energy [1, 2]

$$E(S) = \int_S H^2 \,\mathrm{d}A,$$

which evolves the surface S by decreasing the Willmore energy at the steepest direction. The Willmore energy is a scale invariant. For any sphere, the Willmore energy is 4π, and the sphere is a global minimum for an enclosed surface.

All the above four geometric flows will be applied for surface fairing. Since the geometric flows evolve surface within a pre-defined range, the initial features will not be destroyed seriously. Let S be a quadrilateral surface patch and $\{\mathbf{x}_i\}_{i=1}^N$ be its free vertex set. For a vertex $\mathbf{x}_i$ with valence $2n_i$, $N(i) = \{i_1, i_2, \ldots, i_{n_i}, i'_1, i'_2, \ldots, i'_{n_i}\}$ denotes the index set of the first-ring neighbors of $\mathbf{x}_i$. Geometric PDEs are solved on the quadrilateral mesh S using an explicit discretization method, where the discrete approximation of the mean curvature vector, mean curvature, Gaussian curvature, and surface normal are required. These approximations can be obtained from the quadratic fitting surface with respect to $\mathbf{x}_i$ and its first-ring neighbors [20].

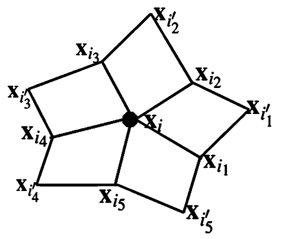


Fig. 6 The first ring neighborhood of $\mathbf{x}_i$

Discretization of Geometric PDEs An Euler explicit discrete scheme

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}}{\tau}$$

is used in the temporal direction. In the averaged mean curvature flow, $h(t)$ can be discretized as $h(t) = \int_{S(t)} H \, \mathrm{d}A / \int_{S(t)} \mathrm{d}A = \sum_{i=1}^{N} [H(\mathbf{x}_i) A_{S(t)}(\mathbf{x}_i)] / A(S(t))$, $A(S(t))$ is the total area of the quadrilateral mesh $S(t)$, $A_{S(t)}(\mathbf{x}_i)$ is one fourth of the first ring neighbor area surrounding vertex $\mathbf{x}_i$, where the first ring neighborhood of $\mathbf{x}_i$ is shown in Fig. 6.

Next, we compute the mean curvature $H(\mathbf{x}_i)$, the Gaussian curvature $K(\mathbf{x}_i)$, and the Laplace–Beltrami operator $\varDelta_s$ at vertex $\mathbf{x}_i$ on quadrilateral meshes. Suppose the vertex $\mathbf{x}_i$ has a valence of n, and its neighboring vertices are $\mathbf{x}_{i_j}$ $(i_j \in N(i))$. First, we fit $\mathbf{x}_i$ and its neighboring vertices to a quadric surface in the local coordinate system via the algorithm proposed in [20]. The basis function is chosen as

$$\{B_l(u,v)\}_{l=0}^{5} = \left\{1, u, v, \frac{1}{2}u^2, uv, \frac{1}{2}v^2\right\},$$

then the problem is to determine coefficients $\mathbf{c}_l \in \mathbb{R}^3$ for the parametric-form fitted surface $\mathbf{x}(u,v) := \sum_{l=0}^{5} \mathbf{c}_l B_l(u,v)$, such that

$$\sum_{l=0}^{5} \mathbf{c}_l B_l(\mathbf{q}_k) = \mathbf{x}_{i_k}, \quad k = 0, \ldots, n$$

in the least square sense. Here i_0 is denoted as i, and $\mathbf{q}_k$ is the local coordinate of $\mathbf{x}_{i_k}$ on the tangent plane of $\mathbf{x}_i$. After determining $\{\mathbf{c}_l\}_{l=0}^{5}$, it is easy to compute $\mathbf{x}_u$, $\mathbf{x}_v$, $g_{11} = \langle \mathbf{x}_u, \mathbf{x}_u \rangle$, $g_{12} = \langle \mathbf{x}_u, \mathbf{x}_v \rangle$, $g_{22} = \langle \mathbf{x}_v, \mathbf{x}_v \rangle$, $g = g_{11} g_{22} - g_{12} g_{12}$, $b_{11} = \langle \mathbf{x}_{uu}, \mathbf{n} \rangle$, $b_{12} = \langle \mathbf{x}_{uv}, \mathbf{n} \rangle$, $b_{22} = \langle \mathbf{x}_{vv}, \mathbf{n} \rangle$, $g_{\alpha\beta\gamma} = \langle \mathbf{x}_{u^\alpha}, \mathbf{x}_{u^\beta u^\gamma} \rangle$ $(\alpha, \beta, \gamma = 1, 2)$, and $\mathbf{x}_{u^\alpha u^\beta} = \frac{\partial^2 \mathbf{x}}{\partial u^\alpha \, \partial u^\beta}$ $(\alpha, \beta = 1, 2)$.

Using the approximate equation given in [20], we can calculate the mean curvature vector, the Gaussian curvature, and the Laplace–Beltrami operator as follows.

$$H(\mathbf{x}_i)\mathbf{n} = \mathbf{H}(\mathbf{x}_i) = \frac{1}{2} \varDelta_s \mathbf{x}_i \approx \frac{1}{2} \sum_{j=0}^{n} w_{ij}^{\varDelta} \mathbf{x}_{i_j},$$

where we use the superscript "$\varDelta$" to denote the approximation coefficient for the Laplacian–Beltrami operator $\varDelta_s$.

$$\begin{aligned}
w_{i,j}^{\Delta} &= g_u^{\Delta} c_1^{(j)} + g_v^{\Delta} c_2^{(j)} + g_{uu}^{\Delta} c_3^{(j)} + g_{uv}^{\Delta} c_4^{(j)} + g_{vv}^{\Delta} c_5^{(j)}, \\
g_u^{\Delta} &= -\big[g_{11}(g_{22}g_{122} - g_{12}g_{222}) + 2g_{12}(g_{12}g_{212} - g_{22}g_{112}) \\
&\quad + g_{22}(g_{22}g_{111} - g_{12}g_{211})\big]/g^2, \\
g_v^{\Delta} &= -\big[g_{11}(g_{11}g_{222} - g_{12}g_{122}) + 2g_{12}(g_{12}g_{112} - g_{11}g_{212}) \\
&\quad + g_{22}(g_{11}g_{211} - g_{12}g_{111})\big]/g^2, \\
g_{uu}^{\Delta} &= \frac{g_{22}}{g}, \qquad g_{uv}^{\Delta} = -\frac{2g_{12}}{g}, \qquad g_{vv}^{\Delta} = \frac{g_{11}}{g},
\end{aligned}$$

and $c_l^{(j)}$ $(l = 1, \ldots, 5,\ j = 0, \ldots, n)$ is the $(l+1, j+1)$-th element of $\mathbf{C}$.

For the Gaussian curvature, we have

$$K(\mathbf{x}_i)\mathbf{n} = \mathbf{K}(\mathbf{x}_i) = \frac{1}{2}\Box\mathbf{x}_i \approx \frac{1}{2}\sum_{j=1}^{n} w_{ij}^{\Box}\mathbf{x}_{i_j},$$

where “□” denotes the Giaquinta–Hildebrandt operator,

$$\begin{aligned}
w_{i,j}^{\Box} &= g_u^{\Box} c_1^{(j)} + g_v^{\Box} c_2^{(j)} + g_{uu}^{\Box} c_{11}^{(j)} + g_{uv}^{\Box} c_{12}^{(j)} + g_{vv}^{\Box} c_{22}^{(j)}, \\
g_u^{\Box} &= -\big[b_{11}(g_{22}g_{122} - g_{12}g_{222}) + 2b_{12}(g_{12}g_{212} - g_{22}g_{112}) \\
&\quad + b_{22}(g_{22}g_{111} - g_{12}g_{211})\big]/g^2, \\
g_v^{\Box} &= -\big[b_{11}(g_{11}g_{222} - g_{12}g_{122}) + 2b_{12}(g_{12}g_{112} - g_{11}g_{212}) \\
&\quad + b_{22}(g_{11}g_{211} - g_{12}g_{111})\big]/g^2, \\
g_{uu}^{\Box} &= \frac{b_{22}}{g}, \qquad g_{uv}^{\Box} = -\frac{2b_{12}}{g}, \quad \text{and} \quad g_{vv}^{\Box} = \frac{b_{11}}{g}.
\end{aligned}$$

4.5 Regularization of Boundary Quadrilateral Mesh

Generally, a quadrilateral mesh is referred as regular if its vertices are equally distributed, and each quadrilateral is close to a square. For a given quadrilateral surface patch $\mathcal{S}$, we use the following energy functional to describe its regularity,

$$\mathcal{E}(\mathcal{S}) = \frac{1}{2}\sum_{i=1}^{N}\big(E_1(\mathbf{x}_i) + \lambda_1 E_2(\mathbf{x}_i) + \lambda_2 E_3(\mathbf{x}_i)\big), \tag{16}$$

where

$$\begin{aligned}
E_1(\mathbf{x}_i) &= \sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big)^2, \\
E_2(\mathbf{x}_i) &= \sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i'_j} - \mathbf{x}_i\| - \sqrt{2}h\big)^2,
\end{aligned} \tag{17}$$

$$E_3(\mathbf{x}_i) = \sum_{j=1}^{n_i}\big(\det(\mathbf{x}_{i_j} - \mathbf{x}_i, \mathbf{x}_{i_{j+1}} - \mathbf{x}_i, \mathbf{n}_i) - J_i\big)^2.$$

In Eq. (16), $\{\mathbf{x}_i\}_{i=1}^N$ are free vertices of surface $\mathcal{S}$. For the vertex $\mathbf{x}_i$ with the quadrilateral valence of n_i, let $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{n_i}}$ be the neighboring vertices connected with $\mathbf{x}_i$, and $\mathbf{x}_{i'_1}, \ldots, \mathbf{x}_{i'_{n_i}}$ be the opposite vertices of $\mathbf{x}_i$ in the quadrilateral $[\mathbf{x}_i\mathbf{x}_{i_j}\mathbf{x}_{i'_j}\mathbf{x}_{i_{j+1}}]$ $(j = 1, \ldots, n_i)$, $n_i + 1 \triangleq 1$. Figure 6 illustrates the case with $n_i = 5$. We intend to regularize the quadrilateral mesh by minimizing the energy functional (16) which is the combination of three terms:

(1) Obviously, $\sum_{i=1}^N E_1(\mathbf{x}_i)$ is globally minimized when the distance between each pair of neighboring vertices equals to h, where $h = \sqrt{A_m}$, and A_m is the average area of all the quadrilaterals.
(2) $\sum_{i=1}^N E_2(\mathbf{x}_i)$ is used to force the diagonals of each quadrilateral as long as $\sqrt{2}h$, so as to avoid the existence of slender elements.
(3) In the third term $E_3(\mathbf{x}_i)$, J_i stands for the averaged Jacobian with respect to $\mathbf{x}_i$, and $\mathbf{n}_i$ is the unit normal direction at $\mathbf{x}_i$.

At each free vertex $\mathbf{x}_i$, we vary $\mathbf{x}_i$ as $\mathbf{x}_i \to \mathbf{x}_i + \varepsilon_i\Phi_i$, where $\Phi_i \in \mathbb{R}^3$, $i = 1, \ldots, N$. It is easy to derive the following first order variation form

$$\begin{aligned}\delta\big(\mathcal{E}(\mathcal{S}), \Phi_i\big) = &\sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i_j})}{\|\mathbf{x}_i - \mathbf{x}_{i_j}\|}\\ &+ \lambda_1\sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i'_j} - \mathbf{x}_i\| - \sqrt{2}h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i'_j})}{\|\mathbf{x}_i - \mathbf{x}_{i'_j}\|}\\ &+ \lambda_2\sum_{j=1}^{n_i}\big(\det(\mathbf{x}_{i_j} - \mathbf{x}_i, \mathbf{x}_{i_{j+1}} - \mathbf{x}_i, \mathbf{n}_i) - J_i\big)\det(\Phi_i, \mathbf{x}_{i_j} - \mathbf{x}_{i_{j+1}}, \mathbf{n}_i).\end{aligned}$$

To preserve the surface shape, all the free vertices are forced to move on its tangential plane. Let $\mathbf{e}_i^{(1)}$ and $\mathbf{e}_i^{(2)}$ be two unit orthogonal tangential directions at $\mathbf{x}_i$, we construct the following two sets of L^2-gradient flows from the first-order variations,

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} + \delta\big(\mathcal{E}(\mathcal{S}), \mathbf{e}_i^{(l)}\big)\mathbf{e}_i^{(l)} = 0, \quad l = 1, 2. \tag{18}$$

An explicit Euler scheme is applied to solve the L^2-gradient flows with unknown $\mathbf{x}_i$, $i = 1, \ldots, N$.

Remark 1 In the energy functional (16), h is global. In practice, the local $h_i = \sqrt{A_i}$ can be used for each $\mathbf{x}_i$ as well, where A_i is the average area of quadrilaterals surrounding $\mathbf{x}_i$. During the iteration process, either the global h or the local h_i should be updated.

4.6 Hexahedral Mesh Optimization

For hexahedral mesh optimization, the above algorithms can be used to improve boundary quadrilateral meshes. Here, we introduce three approaches to optimize the shape of hexahedra elements by modifying the interior vertices.

4.6.1 Local Optimization

During the process of curve fairing and surface smoothing, elements nearby curves and surfaces maybe inverted. Here, we use a simple and fast local optimization approach proposed in [8] to untangle the hexahedral mesh. The vertex with negative Jacobian is relocated such that

$$\max \min_{j=1,\ldots,n_i} Jacobian_j(\mathbf{x}_i), \tag{19}$$

where $Jacobian_j(\mathbf{x}_i)$ denotes the Jacobian of $\mathbf{x}_i$ with respect to its j-th neighboring hexahedron, n_i is the vertex valence of $\mathbf{x}_i$. The optimization problem (19) is a linear programming problem which can be solved by the simplex method.

4.6.2 Global Optimization

Suppose $\{\mathbf{x}_i\}_{i=1}^{N}$ is the set of all interior vertices in a hexahedral mesh, for each $\mathbf{x}_i$, n_i, n_i', and n_i'' are the vertex valence, quadrilateral valence, and hexahedral valence, respectively. To optimize the whole quality of the hexahedral mesh, we minimize the following energy functional,

$$\mathcal{E}(\mathcal{H}) = \frac{1}{2}\sum_{i=1}^{N} E_1(\mathbf{x}_i) + \lambda \sum_{i=1}^{N} E_2(\mathbf{x}_i), \tag{20}$$

where

$$\begin{aligned} E_1(\mathbf{x}_i) = &\sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big)^2 + \sum_{j=1}^{n_i'}\big(\|\mathbf{x}_{i_j'} - \mathbf{x}_i\| - \sqrt{2}h\big)^2 \\ &+ \sum_{j=1}^{n_i''}\big(\|\mathbf{x}_{i_j''} - \mathbf{x}_i\| - \sqrt{3}h\big)^2, \end{aligned} \tag{21}$$

$$E_2(\mathbf{x}_i) = \sum_{j=1}^{n_i''}\big(\det(\mathbf{x}_{i_{j_1}} - \mathbf{x}_i, \mathbf{x}_{i_{j_2}} - \mathbf{x}_i, \mathbf{x}_{i_{j_3}} - \mathbf{x}_i) - J_i\big)^2.$$

In Eq. (21), $h = \sqrt[3]{A}$, and A is the average volume of hexahedra in one component. J_i stands for the averaged Jacobian with respect to $\mathbf{x}_i$. $\{\mathbf{x}_{i_j}\}_{j=1}^{n_i}$ is the neighboring vertices connected with $\mathbf{x}_i$, $\{\mathbf{x}_{i_j'}\}_{j=1}^{n_i'}$ are opposite vertices of each neighboring

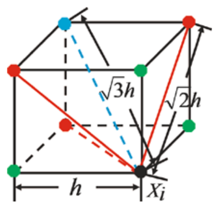


Fig. 7 Neighboring vertices of $\mathbf{x}_i$ in a hexahedron. *Green points* are neighboring vertices connected with $\mathbf{x}_i$; *red points* are opposite vertices of neighboring quadrilaterals; the *blue point* is the diagonal vertex of $\mathbf{x}_i$ in the hexahedron. Distances between neighboring vertices and $\mathbf{x}_i$ are expected to be h, $\sqrt{2}h$, and $\sqrt{3}h$, respectively

quadrilateral, and $\{\mathbf{x}_{i''_j}\}_{j=1}^{n''_i}$ are opposite vertices of each neighboring hexahedron. $\det(\mathbf{x}_{i_{j_1}} - \mathbf{x}_i, \mathbf{x}_{i_{j_2}} - \mathbf{x}_i, \mathbf{x}_{i_{j_3}} - \mathbf{x}_i)$ is the Jacobian of $\mathbf{x}_i$ with respect to its j-th neighboring hexahedron, and $\mathbf{x}_{i_{j_1}}, \mathbf{x}_{i_{j_2}}, \mathbf{x}_{i_{j_3}}$ are the three neighboring vertices connected with $\mathbf{x}_i$ in the hexahedron.

The first term of the energy functional attempts to make the vertex distance in each hexahedron satisfy the relationship as shown in Fig. 7. The second term intends to make the Jacobians of $\mathbf{x}_i$ equal to the averaged Jacobian J_i. We can derive the first order variation of the energy functional (20) as follows:

$$\begin{aligned}\delta\big(\mathcal{E}(\mathcal{H}), \Phi_i\big) = {} & \sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i_j})}{\|\mathbf{x}_i - \mathbf{x}_{i_j}\|} \\ & + \sum_{j=1}^{n'_i}\big(\|\mathbf{x}_{i'_j} - \mathbf{x}_i\| - \sqrt{2}h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i'_j})}{\|\mathbf{x}_i - \mathbf{x}_{i'_j}\|} \\ & + \sum_{j=1}^{n''_i}\big(\|\mathbf{x}_{i''_j} - \mathbf{x}_i\| - \sqrt{3}h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i''_j})}{\|\mathbf{x}_i - \mathbf{x}_{i''_j}\|} \\ & - \lambda\sum_{j=1}^{n''_i}\big(\det(\mathbf{x}_{i_{j_1}} - \mathbf{x}_i, \mathbf{x}_{i_{j_2}} - \mathbf{x}_i, \mathbf{x}_{i_{j_3}} - \mathbf{x}_i) - J_i\big) \\ & \times \Phi_i^T\big((\mathbf{x}_{i_{j_2}} - \mathbf{x}_i) \times (\mathbf{x}_{i_{j_3}} - \mathbf{x}_i)\big) + \big((\mathbf{x}_{i_{j_3}} - \mathbf{x}_i) \times (\mathbf{x}_{i_{j_1}} - \mathbf{x}_i)\big) \\ & + \big((\mathbf{x}_{i_{j_1}} - \mathbf{x}_i) \times (\mathbf{x}_{i_{j_2}} - \mathbf{x}_i)\big).\end{aligned}$$

Then we move each interior vertices using the L^2-gradient flow:

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} + \sum_{l=1}^{3}\delta\big(\mathcal{E}(\mathcal{S}), \mathbf{e}^{(l)}\big)\mathbf{e}^{(l)} = 0. \tag{22}$$

Where $\mathbf{e}^{(1)} = (1, 0, 0)^T$, $\mathbf{e}^{(2)} = (0, 1, 0)^T$, $\mathbf{e}^{(3)} = (0, 0, 1)^T$. The equation is solved using explicit Euler scheme with unknown interior vertices.

The global optimization method has the advantage of improving the whole mesh quality, but it cannot guarantee all the hexahedra are valid. Thus, we combine the global optimization with the local optimization in our mesh improvement algorithm.

4.6.3 Further Improvement

Generally, after local and global optimization, we can get high quality hexahedral meshes with no degraded or inverted elements. However, for the meshes with complicated boundaries, there still exists several negative Jacobians for boundary vertices, since the above two optimization approaches only optimize the interior vertex Jacobians. Hence, we intend to modify neighboring interior vertices of those boundary vertices to eliminate negative Jacobians.

The concrete procedure has three steps. First, we loop for all the hexahedra and compute eight Jacobians for each vertex. Second, loop for each vertex, if the vertex has any Jacobian less than a given threshold, then move its neighboring interior vertices along the gradient direction of Jacobian to increase the Jacobian. Third, gradually increase the threshold, and repeat the previous two steps.

5 Application Examples and Discussion

In this section, we choose one biological dataset and two microstructure datasets to demonstrate the effectiveness of the proposed quality improvement method. For each of dataset, two meshes were generated by an octree-based method [24]: one is the boundary quadrilateral mesh, and the other is the hexahedral mesh. In the following, we will show the improvement results for these meshes.

5.1 Surface Smoothing Using Various Geometric Flows

In Sect. 4.4, we introduced four typical geometric flows: mean curvature flow (MCF), averaged mean curvature flow (AMCF), surface diffusion flow (SDF), and Willmore flow (WF). These geometric flows have their own specific properties, and can be used in different applications. In some practical applications, it is prerequested that the object volume, the boundary area, or the shape features should be preserved. All the four geometric flows can be applied for surface smoothing. To compare the smoothing effects, we validate the four geometric flows on a biological mesh dataset named ATcpnα, which is a chaperonin subunit of an archaea Acidianus tengchongensis strain S5T.

As shown in Fig. 8(a) and Fig. 9(a), the original quadrilateral mesh consists of 103,746 vertices and 104,366 quadrilaterals. Before surface smoothing, vertices are modified along the tangent directions to get a relatively regular mesh, since geometric PDEs discretized on irregular meshes always result in numerical error or

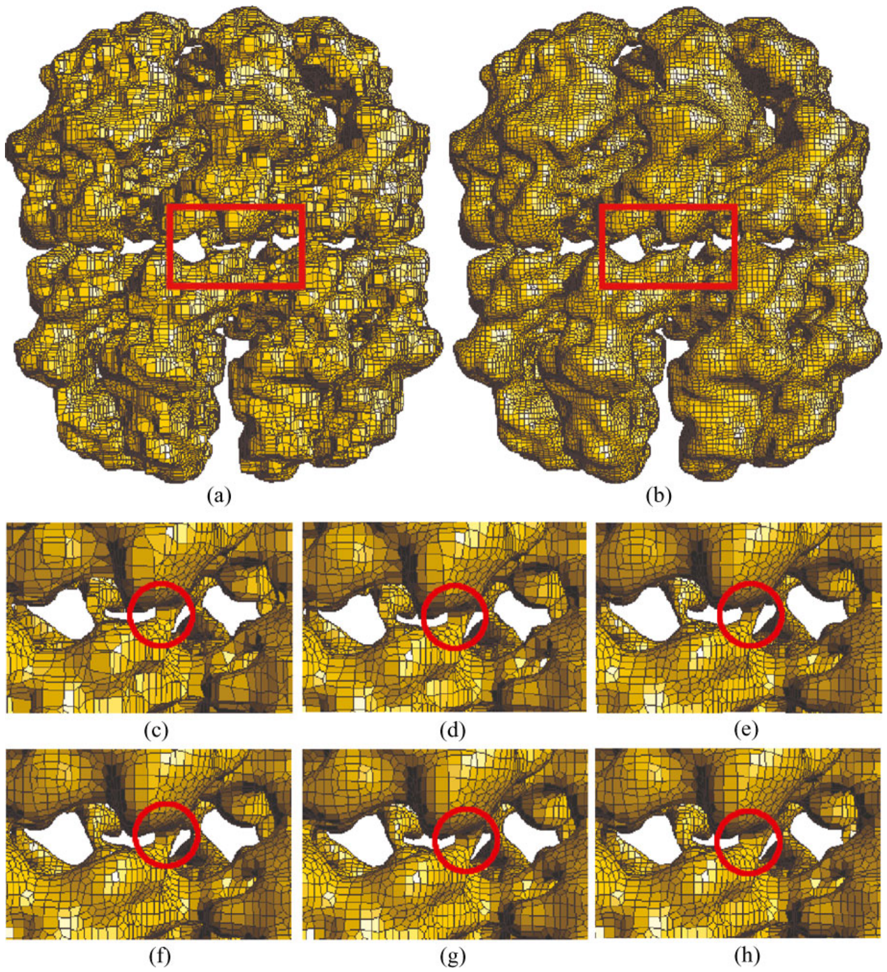


Fig. 8 Quadrilateral mesh of ATcpnα (side view). (**a**) The original mesh; (**b**) the smoothed mesh using surface diffusion flow; (**c**) the enlargement for the red window in (**a**); (**d**) the regularized mesh; and (**e**)–(**h**) are smoothed results using MCF, AMCF, SDF and WF, respectively

even divergence. By minimizing the energy functional (16), we obtain a regularized mesh with well-shaped elements, and the statistics of Jocabians are given in Table 1. Then, MCF, AMCF, SDF, and WF are applied to denoise the regularized but bumpy quadrilateral mesh.

For these four geometric flows, we choose the same temporal step size and iteration number. The smoothing process has 400 iterations, and vertices are re-regularized along the tangent directions for every 100 iterations. The total area of quadrilateral meshes for each iterative step is plotted in Fig. 10. The quality statistics of mesh smoothing by various geometric flows are given in Table 1. Moreover, the volume enclosed by the meshes are calculated to investigate the volume-preserving

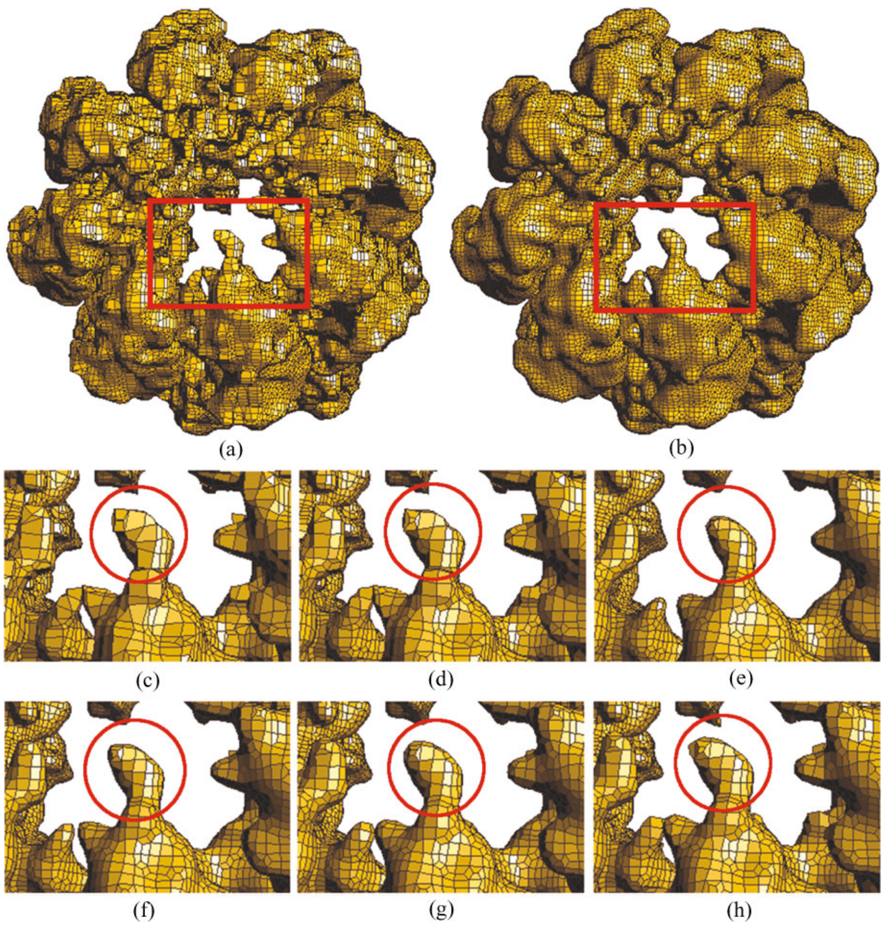


Fig. 9 Quadrilateral mesh of ATcpnα (top view). (**a**) The original mesh; (**b**) the smoothed mesh using surface diffusion flow; (**c**) the enlargement for the red window in (**a**); (**d**) the regularized mesh; and (**e**)–(**h**) are smoothed results using MCF, AMCF, SDF and WF, respectively

Table 1 Mesh quality comparison for using different geometric flows

Mesh	Jacobian		Number of Jacobian						Volume
	Worst	Best	Negative	0.0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8–1.0	
Original	−0.9798	1.0000	1,214	2,175	7,242	19,922	62,442	324,469	143309.9
Regularized	0.0324	1.0000	0	147	1,288	6,626	47,657	361,746	143357.0
MCF	−0.3490	1.0000	9	165	1,244	5,000	35,989	375,057	140378.1 (2.07%)
AMCF	0.0105	1.0000	0	18	815	4,082	34,459	378,090	143344.6 (0.01%)
SDF	0.0561	1.0000	0	81	735	3,841	32,497	380,310	143690.9 (0.23%)
WF	0.0767	1.0000	0	87	706	3,993	33,891	378,787	143393.0 (0.02%)

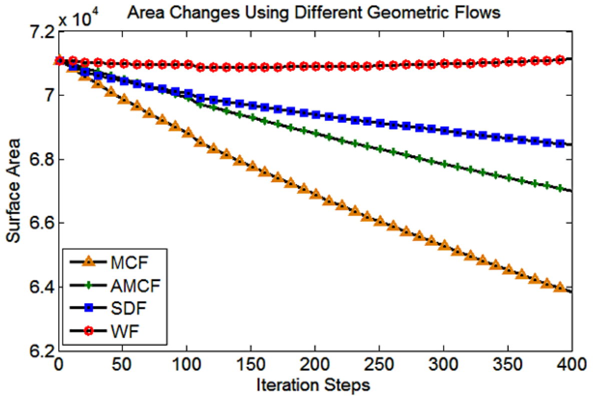


Fig. 10 Surface area changes during the evolution driven by four geometric flows

property of those geometric flows. Except for MCF, the other three geometric flows keep the volume well (the volume change is within 0.3%).

The MCF aims to evolve the surface along the normal direction at the speed of the mean curvature, which is simple and intuitive. From the definition of MCF (10), the evolution stops when $H = 0$ all over the surface. Therefore, the enclosed surface will shrink to a point eventually. Moreover, MCF can be used to get the minimum surface according to the given boundary curve. In Fig. 10, it is clear that the MCF reduces the surface area at the fastest speed among the four flows. As shown in Fig. 8(e) and Fig. 9(e), the bumpy surface can be well-smoothed using the MCF, but also along with the inevitable shrinkage.

AMCF and SDF are two volume-preserving flows. AMCF intends to equalize the mean curvature all over the surface, which seems unreasonable for a complicated surface. As a fourth order geometric flow, SDF takes account of the 1-ring and 2-ring neighbor vertices, and intends to make the mean curvature vary gradually. In Fig. 10, it can be seen that, AMCF decreases the surface area slower than MCF but faster than SDF.

WF has the property of driving a surface to a sphere, no matter how small the neck is, and the terminate sphere radius depends on the initial surface. Figure 8(h) shows the tiny expansion of thin necks. Theoretically, WF is not area-preserving and volume-preserving. However, in this example, WF almost keeps the surface area (see Fig. 10) and the enclosed volume (see Table 1).

Comparing the resulting meshes in Fig. 8 and Fig. 9, we discovered that the SDF preserves surface feature better than the other three flows. In the process of mesh smoothing, the area reduction is reasonable since the given mesh is bumpy. In the following application examples, we choose the SDF to evolve the boundary surfaces.

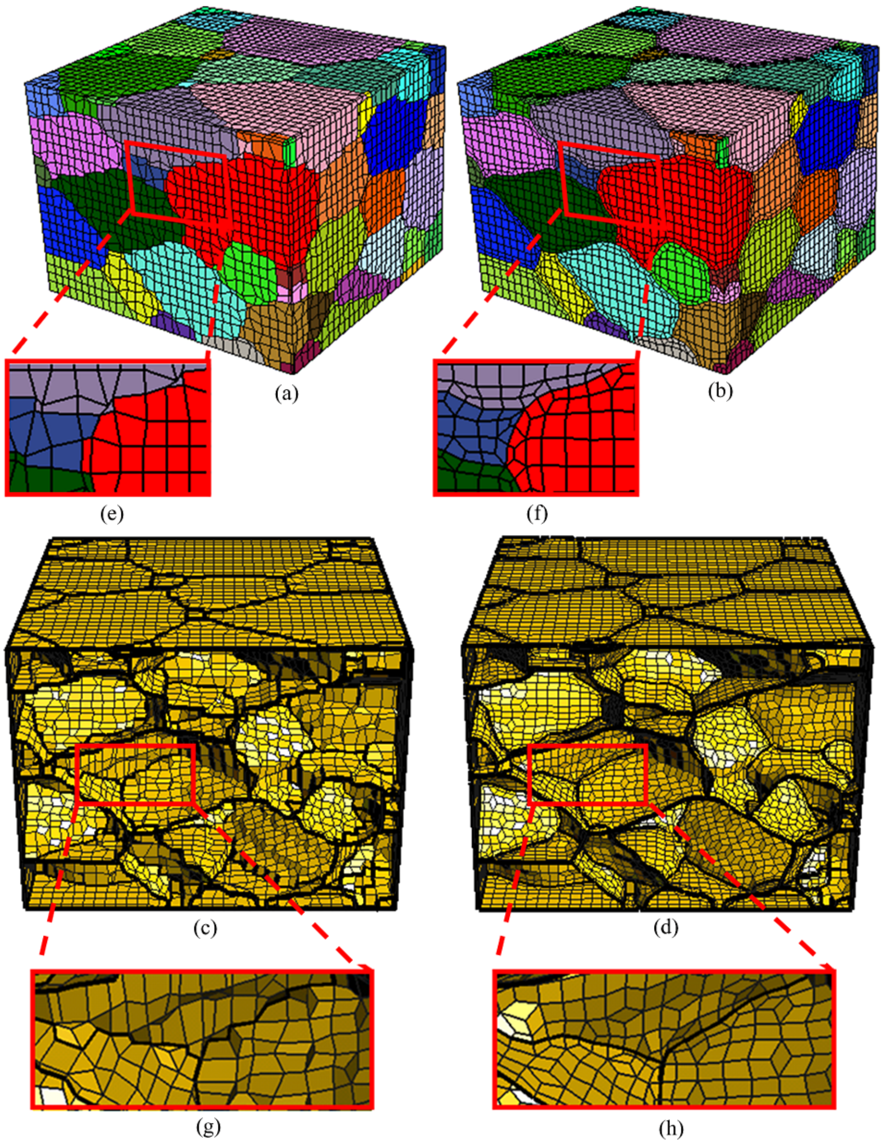


Fig. 11 92-grain microstructure. (**a**) The exterior of the original mesh; (**c**) the exterior of the improved mesh; (**d**) the interior of the original mesh; (**f**) the interior of the improved mesh; and (**e**)–(**h**) show the enlargement of *red windows* in (**a**)–(**d**), respectively

5.2 Quality Improvement for Quadrilateral Meshes

The proposed approach is then applied to two titanium alloy microstructure datasets. The two datasets are composed of 92 grains (see Fig. 11) and 52 grains (see Fig. 12),

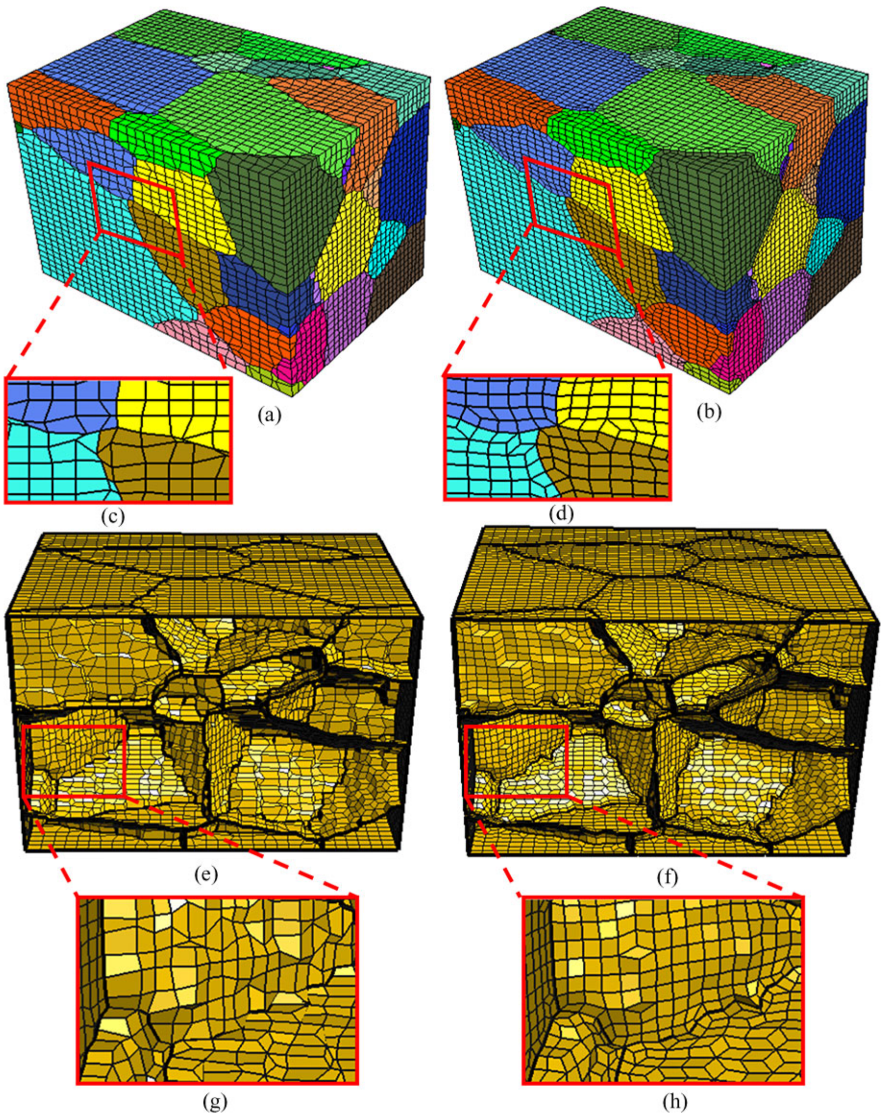


Fig. 12 52-grain microstructure. (**a**) The exterior of the original mesh; (**c**) the exterior of the improved mesh; (**d**) the interior of the original mesh; (**f**) the interior of the improved mesh; and (**e**)–(**h**) show the enlargement of *red windows* in (**a**)–(**d**), respectively

respectively. The union of all the grain boundaries forms a non-manifold boundary. The given quadrilateral meshes of the two non-manifold boundaries are given in Fig. 11(a) and Fig. 12(a). There are a great number of poorly-shaped quadrilaterals in the original meshes. Mesh vertices are irregularly distributed, the boundary curves and surfaces are bumpy.

Table 2 Quality comparison of quadrilateral meshes before and after improvement

	Mesh	Mesh size (vertex, quad)	Jacobian		Number of Jacobian					
			Worst	Best	Negative	0.0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8–1.0
92-grain	Original	(13,690, 15,459)	−0.8711	1.0000	151	336	965	2,855	7,966	49,561
	Improved	(24,258, 26,027)	0.1052	1.0000	0	21	323	2,426	9,963	91,375
52-grain	Original	(13,511, 14,738)	−0.6129	1.0000	150	397	1,082	2,520	7,778	47,023
	Improved	(20,823, 22,050)	0.1221	1.0000	0	26	258	2,053	11,461	74,402

To improve the quadrilateral meshes, we first divide the mesh into several manifold surface patches and boundary curves. Since the outline of the two data volumes is a cuboid, we treat the eight corners as fixed vertices, and cuboid edges as boundary curves. Then the algorithms presented in Sects. 4.2–4.5 are applied to the boundary curves and surface patches. Since there are several poor quality quadrilaterals with more than two edges on the boundary curve, the pillowing technique is used to eliminate these cases by inserting some vertices.

After quality improvement, we obtain remarkable optimized quadrilateral meshes. Fig. 11 and Fig. 12 show the contrast between meshes before and after quality improvement. It is obvious that both curves and surfaces in the improved meshes are smooth. Moreover, the vertices are uniformly distributed with no poorly shaped quadrilaterals. Table 2 lists the statistics of the scaled Jacobian for the two meshes. As shown in the table, there are a great number of negative Jacobians in the original mesh. Our improvement method makes all the Jacobian greater than 0.1, the overall mesh quality is significantly upgraded with the number of good elements (Jacobian > 0.6) increased and the number of poor elements (Jacobian < 0.4) reduced.

5.3 *Quality Improvement for Hexahedral Meshes*

We further validate the proposed improvement method on the hexahedral meshes of the three datasets: ATcpnα, 92-grain, and 52-grain titanium alloy microstructure. Approaches proposed in Sects. 4.2–4.5 are applied to smooth and regularize boundary meshes. Since there are several hexahedra with more than one face on the boundary, mesh pillowing should be implemented. Then, the local improvement method is used to modify the vertices near the boundary surface, which can eliminate most negative Jacobians. The whole mesh quality is improved by minimizing the energy functional (20). Finally, further optimization is implemented to eliminate the Jacobians less than the threshold 0.1.

The mesh quality statistics before and after improvement are listed in Table 3, which shows the significant improvement of the mesh quality. There are thousands of negative Jacobians in the original meshes, while the improved meshes are high quality, either scaled Jacobians or condition numbers are desirable. The cross sections for the three meshes are shown in Figs. 13, 14, 15. It can be seen that the

Table 3 Quality comparison of hexahedral meshes before and after improvement

	Mesh	Mesh size (vertex, quad)	Jacobian		Number of Jacobian				Condition number	
			Worst	Best	Negative	0.0–0.2	0.2–0.6	0.6–1.0	Min	Max
ATcpnα	Original	(196,042, 141,979)	−0.9337	1.0000	45,306	41,747	78,117	970,662	1.0000	4.9e6
	Improved	(299,916, 246,277)	0.0375	1.0000	0	221	190,355	1,779,640	1.0000	328.3
92-grain	Original	(27,720, 25,024)	−0.7993	1.0000	1,783	3,215	14,723	180,473	1.0004	1.2e5
	Improved	(49,072, 44,994)	0.1000	1.0000	0	1,885	47,896	310,171	1.0000	815.6
52-grain	Original	(32,768, 29,791)	−0.6861	1.0000	2,178	4,204	15,157	216,789	1.0000	2.6e15
	Improved	(50,756, 46,695)	0.1002	1.0000	0	226	24,842	348,492	1.0017	1.6e4

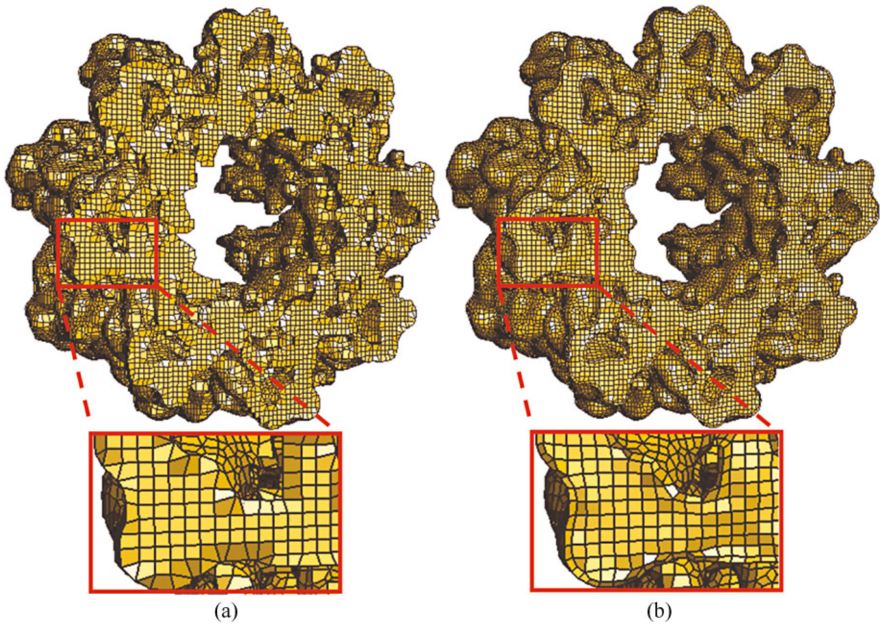


Fig. 13 Cross sections for ATcpnα hexahedral meshes. (**a**) The original mesh; and (**b**) the improved mesh

newly added vertices by the pillowing technique are distributed regularly after mesh improvement.

6 Conclusion

We have developed a series of algorithms to improve the mesh quality of quadrilateral/hexahedral meshes for segmented multiple regions. Our proposed method combines the pillowing technique, geometric flow method, and optimization-based

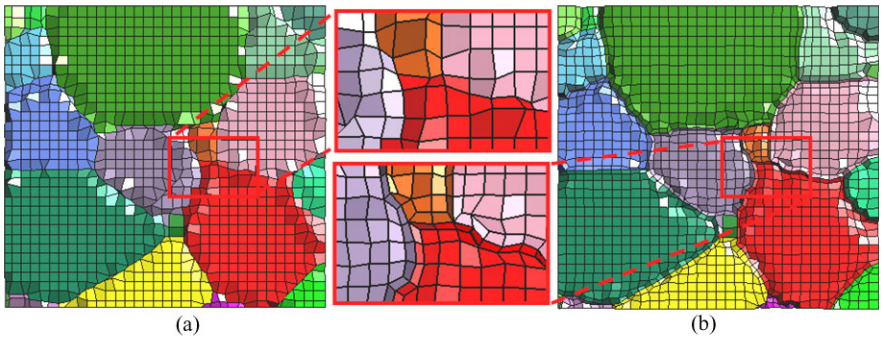


Fig. 14 Cross sections of the 92-grain hexahedral meshes. (**a**) The original mesh; and (**b**) the improved mesh

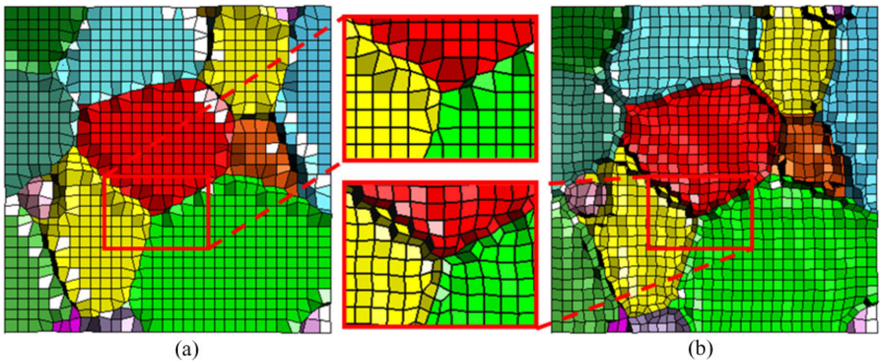


Fig. 15 Cross sections of the 52-grain hexahedral meshes. (**a**) The original mesh; and (**b**) the improved mesh

approaches. The pillowing technique is applied to eliminate the cases that two or more edges/faces of one quadrilateral/hexahedron are located on a boundary curve/surface. Driven by geometric flows, vertices located on boundary curves and boundary surfaces move along the normal direction to remove the zigzag and bumpiness. Energy functionals, which are minimized using L^2-gradient flows, are constructed to regularly distribute vertices and improve vertex Jacobians.

We compared the surface smoothing effects of four typical geometric flows, and utilized the surface diffusion flow, which is feature-preserving and volume-preserving, to smooth surfaces in our quality improvement algorithm. Finally, we validated the proposed method on three application examples. The experimental results and quality statistics results demonstrate the remarkable improvement efficiency of our method. The improved quadrilateral/hexahedral meshes have high quality and the shape feature of boundary curves/surfaces are well preserved.

Acknowledgements The work in this paper was done when J. Leng was a visiting student at the Department of Mechanical Engineering, Carnegie Mellon University. J. Leng, Y. Zhang and

J. Qian were supported in part by Y. Zhang's ONR-YIP award N00014-10-1-0698, an ONR grant N00014-08-1-0653, an AFOSR grant FA9550-11-1-0346, and a NSF/DoD-MRSEC seed grant. J. Leng and G. Xu were supported in part by NSFC key project under the grant 10990013 and Funds for Creative Research Groups of China (grant No. 11021101).

References

1. Bobenko E, Schröder P, Caltech T (2005) Discrete Willmore flow. In: Eurographics symposium on geometry processing, pp 101–110
2. Bryant R (1984) A duality theorem for Willmore surfaces. J Differ Geom 20:23–53
3. Canann A (1991) Plastering and optismoothing: new approaches to automated, 3D hexahedral mesh generation and mesh smoothing. PhD dissertation, Brigham Young University, Provo, UT
4. Desbrun M, Meyer M, Schröder P, Barr AH (1999) Implicit fairing of irregular meshes using diffusion and curvature flow. In: SIGGRAPH'99 proceedings of the 26th annual conference on computer graphics and interactive techniques, Los Angeles, USA, pp 317–324
5. Escher J, Simonett G (1998) The volume preserving mean curvature flow near spheres. Proc Am Math Soc 126(9):2789–2796
6. Field D (1988) Laplacian smoothing and Delaunay triangulation. Commun Appl Numer Methods 4:709–712
7. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Numer Methods Eng 53:1377–1391
8. Freitag L, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Numer Methods Eng 49:109–125
9. Ito Y, Shih A, Soni B (2009) Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. Int J Numer Methods Eng 77:1809–1833
10. Knupp P (2000) Hexahedral mesh untangling and algebraic mesh quality metrics. In: Proceedings of 9th international meshing roundtable, pp 173–183
11. Knupp P (2001) Hexahedral and tetrahedral mesh untangling. Eng Comput 17:261–268
12. Leng J, Zhang Y, Xu G (2011) A novel geometric flow-driven approach for quality improvement of segmented tetrahedral meshes. In: Proceedings of the 20th international meshing roundtable, pp 327–344
13. Li T, McKeag R, Armstrong C (1995) Hexahedral meshing using midpoint subdivision and integer programming. Comput Methods Appl Mech Eng 124:171–193
14. Mitchell S, Tautges T (1995) Pillowing doublets: refining a mesh to ensure that faces share at most one edge. In: Proceedings of the 4th international meshing roundtable, pp 231–240
15. Owen S (1998) A survey of unstructured mesh generation technology. In: Proceedings of the 7th international meshing roundtable, pp 239–267
16. Qian J, Zhang Y, Wang W, Lewis A, Siddiq Qidwai M, Geltmacher A (2010) Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. Int J Numer Methods Eng 82:1406–1423
17. Sapiro G (2001) Geometric partial differential equations and image analysis. Cambridge University Press, Cambridge
18. Schneiders R (1996) A grid-based algorithm for the generation of hexahedral element meshes. Eng Comput 12:168–177
19. Tautges T, Blacker T, Mitchell S (1996) The whisker-weaving algorithm: a connectivity based method for constructing all-hexahedral finite element meshes. Int J Numer Methods Eng 39:3327–3349
20. Xu G (2008) Geometric partial differential equation methods in computational geometry. Science Press of China, Beijing
21. Zhang Y, Bajaj C (2006) Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Comput Methods Appl Mech Eng 195:942–960

22. Zhang Y, Bajaj C, Sohn B (2005) 3D finite element meshing from imaging data. Comput Methods Appl Mech Eng 194:5083–5106
23. Zhang Y, Bajaj C, Xu G (2005) Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In: Proceedings of the 14th international meshing roundtable, pp 449–468
24. Zhang Y, Hughes T, Bajaj C (2010) An automatic 3D mesh generation method for domains with multiple material. Comput Methods Appl Mech Eng 199:405–415

Patient-Specific Model Generation and Simulation for Pre-operative Surgical Guidance for Pulmonary Embolism Treatment

Shankar P. Sastry, Jibum Kim, Suzanne M. Shontz, Brent A. Craven, Frank C. Lynch, Keefe B. Manning, and Thap Panitanarak

Abstract Pulmonary embolism (PE) is a potentially-fatal disease in which blood clots (i.e., emboli) break free from the deep veins in the body and migrate to the lungs. In order to prevent PE, anticoagulation therapy is often used; however, for some patients, it is contraindicated. For such patients, a mechanical filter, namely an inferior vena cava (IVC) filter, is inserted into the IVC to capture and prevent emboli from reaching the lungs. There are numerous IVC filter designs, and it is not well understood which particular IVC filter geometry will result in the best clinical outcome for a given patient. Patient-specific computational fluid dynamic (CFD) simulations may be used to aid physicians in IVC filter selection and placement. In particular, such computational simulations may be used to determine the capability of various IVC filters in various positions to capture emboli, while not creating additional emboli or significantly altering the flow of blood in the IVC. In this paper, we propose a computational pipeline that can be used to generate patient-specific geometric models and computational meshes of the IVC and IVC filter for

S.P. Sastry (✉) · J. Kim · S.M. Shontz · K.B. Manning · T. Panitanarak
The Pennsylvania State University, University Park, PA 16802, USA
e-mail: sps210@cse.psu.edu

J. Kim
e-mail: jzk164@cse.psu.edu

S.M. Shontz
e-mail: shontz@cse.psu.edu

K.B. Manning
e-mail: kbm10@psu.edu

T. Panitanarak
e-mail: txp214@cse.psu.edu

B.A. Craven
Penn State Applied Research Laboratory, University Park, PA 16801, USA
e-mail: bac207@psu.edu

F.C. Lynch
Penn State Milton S. Hershey Medical Center, Hershey, PA 17033, USA
e-mail: flynch@hmc.psu.edu

various IVC anatomies based on the patient's computer tomography (CT) images. Our pipeline involves several steps including image processing, geometric model construction, surface and volume mesh generation, and CFD simulation. We then use our patient-specific meshes of the IVC and IVC filter in CFD simulations of blood flow, whereby we demonstrate the potential utility of this approach for optimized, patient-specific IVC filter selection and placement for improved prevention of PE. The novelty of our approach lies in the use of a superelastic mesh warping technique to virtually place the surface mesh of the IVC filter (which was created via computer-aided design modeling) inside the surface mesh of the patient-specific IVC, reconstructed from clinical CT data. We also employ a linear elastic mesh warping technique to simulate the deformation of the IVC when the IVC filter is placed inside of it.

1 Introduction

Venous thromboembolic disease is a process that begins with blood clot formation in the legs known as deep vein thrombosis (DVT). If left untreated these blood clots can progress and eventually migrate to the lungs, causing a pulmonary embolism (PE). The progression of venous thromboembolic disease from DVT to PE is often asymptomatic, but once PE occurs, it is associated with high morbidity and mortality. Over 10% of patients with acute PE die in the first hour. If not adequately treated, the overall mortality of PE is over 30%. The incidence of PE in the United States is estimated to be over 600,000 cases per year. Approximately 200,000 deaths are attributed to PE in the United States annually [4].

Both the occurrence of DVT and its progression to PE can be prevented with the use of anticoagulants, medications that prevent blood from clotting. However, anticoagulation therapy carries with it the risk of bleeding complications. There are patients who cannot receive anticoagulation therapy because of comorbid factors that put them at increased risk for bleeding complications or for whom anticoagulation has failed to prevent the occurrence or progression of DVT. As an alternative to anticoagulation therapy, these patients often undergo implantation of a device known as an inferior vena cava filter (IVC filter) (Fig. 1).

IVC filters are mechanical devices that are placed into the IVC using percutaneous catheter based techniques. The IVC filter functions as a mechanical barrier that can capture large clots and prevent them from reaching the lungs. Most DVT occurs in the legs and pelvis. Since the inferior vena cava (IVC) is the common venous pathway that drains the majority of blood from the lower half of the body, it makes sense to place a barrier there since it is the common route that any blood clot must traverse in order to reach the lungs.

While clinical studies have shown that IVC filters are effective in preventing PE, especially during the period shortly after they have been placed [16], PE still occurs in up to 5% of the cases, despite the presence of an IVC filter [36]. Experiments designed to study the factors that determine the clot trapping efficiency of IVC filters have largely focused on device design. Little attention has been given to other

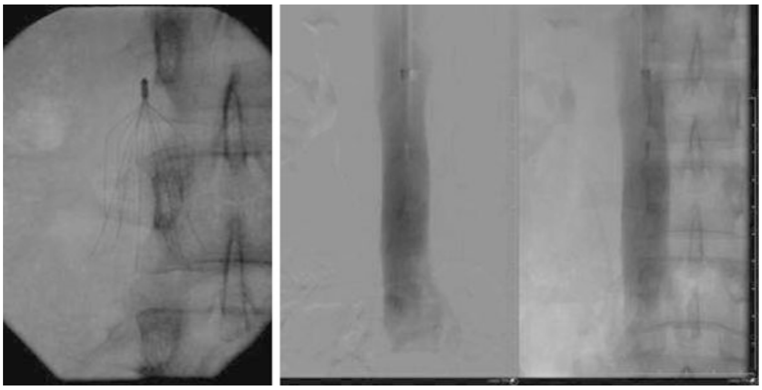

(a) (b)

Fig. 1 X-ray image (*left*) and digital subtraction angiography (*right*) of a G2 filter (Bard Peripheral, Phoenix, AZ) in the inferior vena cava

factors that might affect IVC filter effectiveness, which include the size and shape of the cava, anatomic variations of the IVC and its tributaries, the site of IVC filter implantation, and the effect of respiratory variation in caval size and blood flow.

The IVC is the dominant venous structure in the abdomen. It is generally oval in shape with a major axis measuring between 14 and 33 mm. It is formed as the confluence of the right and left common iliac veins at approximately the level of the fourth lumbar vertebral body. The largest tributaries are the single left and right renal veins that typically enter the IVC at the level of the first lumbar segment. Typically, IVC filters are placed in the infrarenal IVC, that segment of the IVC between the confluence of the iliac veins and the insertion of the renal veins. Filters are placed here so that clots captured in the filter do not propagate back into the renal veins and cause renal failure.

A high degree of anatomic variability is seen in the human infrarenal IVC. Structural anatomic variants of the infrarenal IVC are common and have been well described in the literature [82] (Fig. 2). Duplication of the infrarenal IVC occurs 1–3% of the time. In this case, the second IVC drains into the left renal vein. While the two infrarenal IVCs may be equal in size, the variant represents a spectrum in caval anatomy. In the extreme case, the right infrarenal IVC is entirely absent leaving only a left-sided IVC. This occurs in 0.2–0.5% of patients.

Anatomic variations of the renal veins also greatly impact IVC anatomy and therefore affect IVC filter placement and function [82]. While duplication of the right renal vein has a relatively minor impact on caval anatomy, duplication of the left renal vein usually results in one renal vein that passes anterior to the aorta and one that passes posterior to the aorta. This circumaortic left renal vein occurs in up to 9% of patients, and in its extreme form, which occurs in approximately 3% of patients, only the retroaortic left renal vein is present. These variants are significant because the retroaortic vein inserts into the IVC much more caudally, greatly shortening the effective length of the infrarenal IVC (Fig. 2).

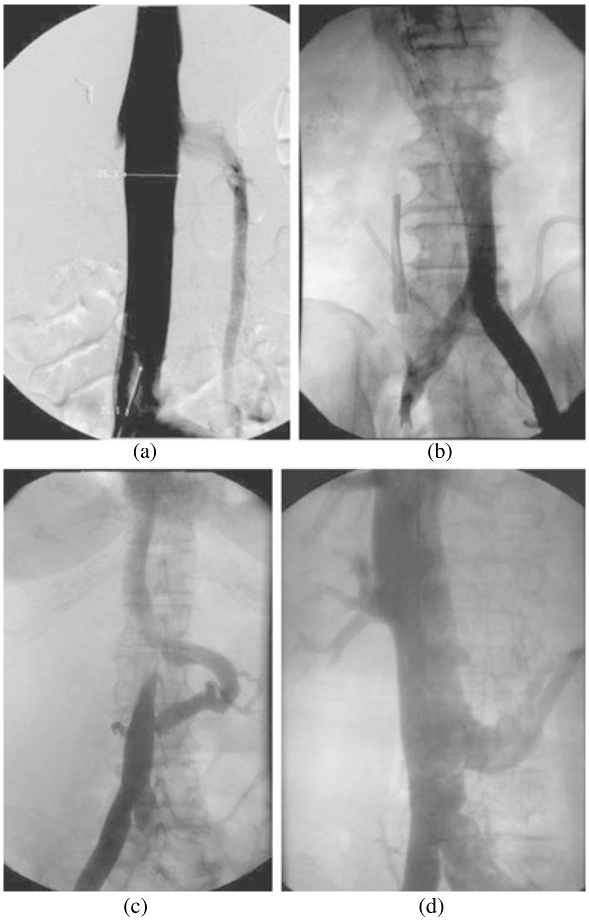


Fig. 2 Common IVC/renal vein variants: (**a**) duplicated IVC, (**b**) left sided IVC, (**c**) circumaortic left renal vein, (**d**) retroaortic left renal vein

These variations in infrarenal IVC size and anatomy may significantly impact IVC filter function. IVC filters are generally designed with self expanding filter elements that become constrained against the caval wall. This "one size fits all" approach leads to instances where the clot trapping elements of the filter are more closely aligned in cavae of smaller cross sectional area (Fig. 3). Variations in branch anatomy may lead to different flow patterns that significantly affect how a clot is presented to the IVC filter, potentially influencing the chances that it is captured. Abnormalities of the spine may lead to curvature or angulation of the infrarenal IVC. Even in cases of "usual" caval size and anatomy, the location of filter implantation (high or low in the IVC) may impact its ability to capture clots. Blood flow through a filter as well as its ability to capture clots also changes as clots accumulate in the filter.

Historically, information regarding the clot trapping effectiveness of IVC filters has been derived from in vivo animal studies [8, 31, 34, 52] and increasingly from in vitro modeling [12, 34, 52, 70]. However, information derived by these means

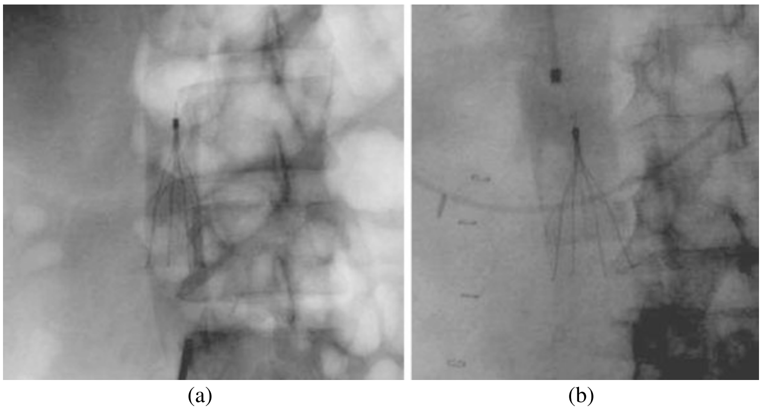


Fig. 3 Celect IVC filters (Cook, Bloomington, IN) placed in a 15 mm diameter cava (*left*) and a 26 mm diameter cava (*right*). In the smaller diameter cava, the filter elements are more closely compressed, potentially leading to a higher clot trapping efficiency

is limited by several factors. Animal studies typically involve a small number of subjects and are performed in species where IVC anatomy and size only approximate that of humans. Real time evaluation of blood flow characteristics in animal models is difficult or impossible given that methods to measure flow characteristics tend to be optically based, requiring an optically clear caval wall and blood. Evaluation of clot trapping in live animal models usually requires the creation and use of radiopaque thrombi, which must be observed using radiographic techniques.

In vitro experiments are typically performed by placing the filter in a Silastic or Plexiglas tube, which is then perfused with a medium that approximates the fluid characteristics of blood. These models are limited since they are based on an idealized IVC that lacks the biomechanical properties, anatomic variation, respiratory variation in size, and flow disturbances from branch vessels that are seen in actual human IVCs. Until recently, evaluation of flow characteristics and clot trapping efficiency has been based on subjective observations, such as observations made when artificially created thrombi are introduced into the flow model.

Given the sheer number of combinations of the factors discussed, exhaustive modeling of all possibilities by in vivo and in vitro techniques is impractical. However, computational modeling has been one approach to overcome this and other limitations of in vivo and in vitro experimentation. Several papers have been written to date concerning computational fluid dynamic (CFD) studies of the flow characteristics of various combinations of the IVC, IVC filter, and blood clots [54, 67–69, 71, 72, 78]. An important limitation of the majority of these studies is the simplistic and unrealistic geometric modeling of the IVC with the exception of [54]. In their paper, Moore et al. employ a realistic, compliant model to represent the IVC, its branching vessels (including the iliac and renal veins), and lumbar curvature based on an average of the normal IVC anatomy of a patient. In addition, none of these studies examined the impact of anatomic IVC variant on the flow results. Simplistic, idealized geometric models of the blood clots were all used in all of these

CFD studies. In contrast, accurate models of the IVC filters are used in each of the studies with the exception of [54], which studied only the realistic IVC model in conjunction with blood clots.

Accurate anatomic models of normal and variant IVCs combined with accurate models of IVC filters can be used for CFD studies, providing a tool that can then be used to make predictions about the flow characteristics, biological response, and clot trapping ability of a given IVC filter design across a wide range of anatomic and physiologic variables. Predictions about differences in clot trapping ability of various filter designs, deployment locations, and deployment configurations might also be derived with the technique. Finally, CFD studies on anatomically correct models could also prove useful in the development of novel IVC filter designs.

The rest of the paper is organized as follows. In Sect. 2, we describe the state-of-the-art techniques for patient-specific mesh generation for use in CFD simulations. In Sect. 3, we describe our previous attempts at generating accurate geometric models of the IVC and the IVC filter using only computer tomography (CT) images as input. In Sect. 4, we describe our computational pipeline for generating geometric models and computational meshes of the IVC and IVC filter and performing CFD simulations of blood flow. In Sect. 5, we present our results on generation of patient-specific meshes for various IVC anatomies. We also present our results from the CFD simulations of blood flow in which use our patient-specific meshes. Conclusions and some directions for future research are presented in Sect. 6.

2 Generation of Patient-Specific Meshes for CFD Simulations

Generation of patient-specific meshes for CFD simulations from medical images (e.g., CT, MRI, etc.) usually follow a computational pipeline approach (e.g., [7, 9, 37]). The computational pipeline typically involves many of the following steps [15]: (1) segmentation of the medical image; (2) modeling of invisible structures (if applicable); (3) surface mesh generation; (4) volume mesh generation; (5) optimization of the mesh. The main criteria for computational meshes generated via this pipeline to satisfy are: mesh validity, geometric accuracy, smoothness and resolution control, and adequate mesh quality. Below we summarize state-of-the-art techniques for Steps 1 and 3–5 in the above pipeline, as these steps are the most relevant to our proposed research.

2.1 Image Segmentation

Region growing methods [28] and level-set methods [57] are two popular approaches for segmentation of medical images [51]. Region growing methods require an initial seed to be manually chosen and extract a region connected to the seed by growing until predefined criteria (such as intensity information or edges in images) [28] are met; however, they are sensitive to noise. Level-set methods are techniques for delineating region boundaries using closed parametric curves (or surfaces) that deform under the influence of a PDE; however, they require initial input. Region-growing methods can be combined with level-set methods to obtain initial and improved medical image segmentations, respectively (e.g., [27]).

2.2 Surface Mesh Generation

The classical approach for generating computational meshes from segmented medical images has been to perform a surface interpolation between the contours describing the segmented volume [21, 53, 56]. Several researchers have used the marching cubes algorithm [41] or one of its improvements [13, 14, 24, 29, 32] to create an initial surface mesh from the 3D segmented data set. The marching cube method is fast and relatively simple, but may not generate a topologically-consistent surface mesh. This limitation was removed in the regularized marching tetrahedron (RMT) algorithm [74].

Virtual implantation of a medical device into a patient anatomy has also been used to create patient-specific geometric models of implanted stents [26]. In this technique, a series of Boolean operations are performed in order to implant the stent geometry into the patient anatomy, yielding an embedded surface.

2.3 Volume Mesh Generation

Numerous researchers have developed techniques for generating computational volume meshes for use in biomedical simulations. Typically, finite element or finite volume simulations are performed using tetrahedral [15, 23, 33, 73, 80] or hexahedral volume meshes [58, 76, 81]; although hybrid meshes [20, 47, 79] represent a tradeoff between the higher accuracy of hexahedral meshes and the ease of generation of tetrahedral meshes.

Dynamic mesh generation is the problem of maintaining a mesh for a geometric domain whose shape changes, e.g., as a function of time. Various techniques, such as mesh warping or adaptivity can be used to update the mesh as the domain deforms. Mesh warping (or morphing) is the process of determining a one-to-one transformation which maps the original 3D mesh to a target domain specified by its boundary surface. Several biomedical mesh warping algorithms have been developed (e.g., [5, 6, 38, 46, 60, 62–65, 81]). Typically the mesh topology is maintained throughout the mesh warping process; however, moving mesh techniques employing adaptive mesh refinement [49, 50] have been used when topological changes occur in the domain.

2.4 Mesh Optimization

Recent research has shown the importance of performing mesh quality improvement before solving PDEs in order to: improve the condition number of the linear systems being solved [59], reduce the time to solution [22], and increase the accuracy of the partial differential equation (PDE) solution. Therefore, mesh optimization methods are often used to obtain high-quality biomedical meshes [30, 40, 75] for numerical modeling and simulation. Only recently have mesh optimization techniques been designed for improving the quality of hybrid biomedical meshes. Dyedov et al. [20] recently introduced a variational approach for smoothing prismatic boundary-layer meshes based on improving triangle shape and edge orthogonality in prism elements; their smoothing approach is based upon their scaled aspect ratio prism mesh quality metric, which is one of only two such metrics in the literature [17].

2.5 Motivation for Current Study

Despite all of the research that has been performed to date in image processing and dynamic mesh generation, there is no computational pipeline or algorithm for generation of anatomically-accurate, patient-specific computational meshes of an IVC filter implanted in the IVC for use in CFD simulations of blood flow. Thus, in this chapter, we focus on the development of such a computational pipeline.

3 Patient-Specific Geometric Modeling of an IVC Filter Implanted in the IVC

A high-resolution geometric model of an IVC filter is necessary for generation of an accurate geometric model of an IVC filter implanted in an anatomically-accurate, patient-specific IVC model. The use of such a model will permit accurate CFD simulations of blood flow on the IVC models. Thus, although a completely image-based technique for geometric modeling of the IVC, its surrounding veins, and the IVC filter may seem desirable, in this section, we illuminate the difficulties experienced in trying to generate such a high-resolution model solely from patient CT images.

An example of an IVC filter is shown in Fig. 4(a). The challenge in generating a high-resolution model of such an IVC filter lies in extracting it from relatively low-resolution clinical CT images. This is particularly difficult for the following reasons: (1) the IVC filter is composed of extremely thin nitinol wires (approximately 0.2 mm in diameter); (2) a radiologist typically acquires a low-resolution CT scan of

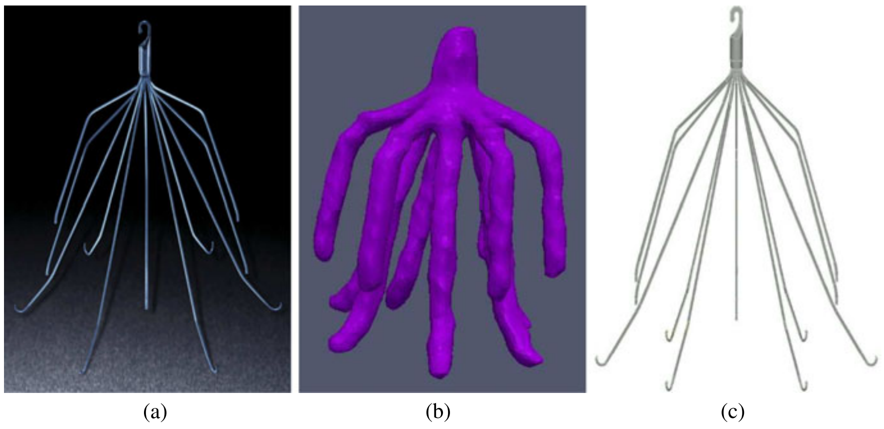


Fig. 4 An initial, unsuccessful attempt at creating a high-resolution IVC filter models from CT data alone, illustrating the shortcomings of this approach: (**a**) G2 Express IVC filter (Bard Peripheral, Tempe, AZ), (**b**) low-resolution IVC filter model (created from a low-resolution patient CT scan), which would result in a severely-obstructed IVC if virtually implanted, (**c**) high-resolution IVC filter model (created via CAD)

the patient after filter implantation to minimize radiation exposure; (3) beam hardening artifacts distorts the wires in the CT images. Consequently, using the IVC filter extracted from patient CT images results in a low-resolution IVC filter model (Fig. 4(b)), which is unrealistic in size and shape (i.e., it is too thick, and the geometry is underresolved), and leads to unrealistic obstruction of the IVC in the blood flow simulation.

Initial, unsuccessful attempts at creating high-resolution models of the IVC filter have included: (1) segmentation of higher-resolution CT images of an idealized model (i.e., one constructed for performing in vitro experiments), (2) modification of the low-resolution IVC model using surface offsetting methods, and (3) skeletonization of the low-resolution IVC filter model. Segmentation of the higher-resolution CT images resulted in an IVC filter model which was somewhat improved in terms of size but was still too thick, as the IVC filter arms/legs were still inadequately resolved. Modification of the original low-resolution IVC filter model by surface offsetting techniques led to a thinner, but still unrealistic IVC filter model in terms of the geometry. The skeletonization approach also led to a thin IVC filter model with unrealistic geometry. Thus, more sophisticated techniques are needed for creation of a high-resolution IVC filter model to be virtually implanted in a patient's IVC. Consequently, to obtain a high-resolution model of the IVC filter, we propose a virtual filter implantation technique, whereby a high-resolution IVC filter model created using computer-aided design (CAD) (Fig. 4(c)) is implanted into the IVC using mesh warping techniques.

4 A Combined CAD- and CT Image-Based Model Creation Technique for CFD Simulations of Blood Flow in Patient-Specific Models

In this section, we describe our computational pipeline for generation of patient-specific geometric models and computational meshes of the IVC and IVC filter, suitable for use in CFD simulations of blood flow.

Our computational pipeline consists of several steps. The process begins with *acquisition* and then *segmentation* of a patient's CT images in order to obtain a geometric model of the patient's IVC and its adjacent veins such as the renal and iliac veins. A surface mesh is then generated on the geometry using the marching cubes algorithm. The *surface mesh generation* step is followed by *surface reconstruction* and *smoothing* resulting in an optimized surface such as is required for subsequent volume mesh generation. A geometric model (i.e., a CAD model) of the IVC filter is then *generated* and *virtually placed* in the IVC at an appropriate location specified by a vascular surgeon or at the location observed in the CT images. Superelastic and linear elastic constitutive laws are used to simulate the deformations of the IVC filter and the IVC, respectively. In particular, the IVC surface mesh and IVC filter volume mesh are warped using these constitutive laws. A *volume mesh* with properties suitable for CFD simulations, i.e., with high-quality boundary layer elements, is then generated on this anatomically-accurate, patient-specific geometry. The final step is to perform *CFD simulations* of the blood flow using the high-fidelity volume mesh of the IVC and IVC filter.

We now describe each step in our computational pipeline in more detail.

4.1 Image Acquisition

The first step in our computational pipeline is image acquisition. After obtaining Institutional Review Board exemption, two sets of CT images representing different IVC anatomies were obtained from a retrospective review of patient records. In particular, the CT images obtained for use in this study represent the left and retroaortic IVC anatomies. The CT image data sets have 0.87 mm/pixel in-plane resolution and a 1.5 mm out-of-plane resolution. In addition, we designed an idealized IVC model based on a normal IVC anatomy. A data set of high-resolution CT images were obtained using a multi-slice spiral CT scanner for our idealized IVC model. The image data set has a 0.44 mm/pixel in-plane resolution and a 0.6 mm out-of-plane resolution.

4.2 Image Segmentation

The second step of our pipeline is segmentation of the patient's CT images; this is done in order to determine the geometry of the patient's IVC. In our study, image segmentation is a semi-automatic process in which we mark the regions corresponding to IVC and its adjacent veins in each 2D CT image in order to obtain an accurate geometry. The regions are then extracted using one of the algorithms described below, which are implemented in Amira [3], an image segmentation software package.

4.2.1 Region Growing

The region growing technique [28] takes a medical image and seeds as input, where each seed denotes a region that needs to be segmented. The seeds are progressively grown from one pixel to the neighboring pixels based on the pixel intensity and its distance from the seed pixel.

4.2.2 Gradient-Based Region Growing

The gradient-based region growing technique, as implemented in Amira, is similar to the region growing technique. The difference is in the stopping criteria used to terminate the progressive growth of the regions. In the former case, termination was based on the intensity of the pixels. In the latter case, it is based on the magnitude of the gradient of the pixel intensity [42]. The gradient of the pixel intensity is given by

$$G = \sqrt{I_x^2 + I_y^2},$$

where I_x and I_y are the partial derivatives of the image intensity with respect to the horizontal and the vertical direction, respectively.

4.2.3 Intelligent Scissors

The intelligent scissors algorithm [44] constructs an underlying graph for an image in which the pixels are interpreted as nodes and adjacent pixels are connected by edges. As the gradient of the magnitude of the pixel intensity is an excellent edge indicator for medical image segmentation, the weight of the edges is a function of the gradient of the magnitude of the pixel intensity between two adjacent pixels. In order to enable the shortest path between each pair of pixels to correspond to edges in the images, the gradients are scaled so that a high gradient results in lower edge cost. The intelligent scissors algorithm finds the shortest path between a set of user-chosen pixels using Dijkstra's algorithm. An ordered set of pixels whose shortest paths from one pixel to the next one form a closed loop successfully completes the segmentation of an image. This technique is used to segment patient images when the other techniques failed due to image noise.

4.3 Surface Mesh Generation

After the geometry of the IVC and its adjacent veins are extracted using one of the above segmentation algorithms, a surface mesh of their geometry is generated using the marching cubes algorithm in Amira. The surface mesh which is generated by this algorithm is often not of good quality and hence often needs to be smoothed before being used for further computational analysis. If the surface mesh is overrefined, it may also need to be coarsened before smoothing.

4.3.1 Marching Cubes

The marching cubes algorithm [24, 29, 41] constructs a triangular surface mesh from segmented images by evaluating the pixels from multiple images and determining the optimal triangulation that best describes the isosurface obtained from the segmented images. The pixels from multiple images form imaginary cubes, where each pixel represents each vertex of the cubes. Each pixel is deemed either inside the region of interest (i.e., inside either the IVC or its adjacent veins) or outside the region of interest. For a set of eight pixels forming a cube, this results in $2^8 = 256$ possible combinations for the cube. Due to symmetry, the number of combinations can be reduced to around 15. Every cube is evaluated to find the combination to which the configuration reduces, and a topologically-consistent triangulation is constructed based on the configuration to obtain a triangular surface mesh.

4.3.2 Poisson Surface Reconstruction

For the idealized IVC model, the surface mesh of the IVC and the adjacent veins obtained from the marching cube algorithm was overrefined. Thus, the mesh was be simplified before using it for further analysis. For this purpose, we use the Poisson surface reconstruction approach [35], which has been implemented in Meshlab [43]. Here, an indicator function is constructed, whose value is 1 inside the surface or 0 outside the surface. Since this function is discontinuous, the indicator function is convolved with a smoothing function, and the gradient of the resulting function, $\mathbf{V}$, is computed. In order to reconstruct the surface using the gradient, a function, χ, is computed such that it gradient is equal to the vector field, i.e.,

$$\nabla\chi = \mathbf{V},$$

where ∇ is a gradient operator. Applying the divergence operator on both sides, we obtain

$$\Delta\chi = \nabla\cdot\mathbf{V}.$$

In order to solve the Poisson system, an adaptive octree mesh of the surface mesh is constructed such that there is a greater resolution near the original surface mesh. After solving the Poisson equation with Neumann boundary conditions, a coarser surface mesh is constructed using the solution.

4.3.3 Surface Mesh Smoothing

The image segmentation and surface reconstruction steps are usually affected by image noise. Construction of volume meshes using noisy surface meshes results in failure of the mesh generator or in poor quality meshes for CFD simulations. Thus, the surface meshes must be smoothed before they are used to generate a volume mesh of the IVC and IVC filter. Because the naive implementation of the Laplacian smoothing algorithm results in the reduction of the volume of the models, we use the Humphrey's Classes (HC) Laplacian smoothing algorithm [77] implemented in Meshlab to smooth our surface meshes. Unlike the Laplacian smoothing algorithm, the HP-Laplacian smoothing algorithm preserves the volume of the surface meshes during smoothing [77].

We briefly summarize the HP-Laplacian smoothing algorithm as follows. First, we define some notation. Let the original mesh vertices be denoted by vector **o**, and let o_i denote the coordinates of the i^{th} vertex in the mesh. Let vector **q** denote the mesh vertex positions prior to the beginning of the current iteration. Now define vector **p** to be the new vertex positions after the current iteration. In the HC-Laplacian algorithm, the modified vertices, represented by vector **p**, are first computed by Laplacian smoothing. They are then pushed back toward their original positions, vector **o**, or their positions from the previous iteration, i.e., vector **q**, by taking the average of the differences between their original positions and their positions in their previous iteration as shown:

$$b_i = p_i - \big(\alpha o_i - (1-\alpha)q_i\big), \quad \text{i.e., by}$$

$$d_i = \beta b_i - \frac{1-\beta}{|\operatorname{adj}(i)|} \sum_{j \in \operatorname{adj}(i)} b_j,$$

where $\operatorname{adj}(i)$ is the set of vertices adjacent to vertex i and d_i is the vector by which the vertex p_i is pushed back. In the final step of the algorithm, the final positions,

$$p_i^{\text{final}} = p_i - d_i,$$

are computed.

4.4 Generation of a Geometric Model of the IVC Filter

The next step in our pipeline is to generate a geometric model of the IVC filter. We obtained a Computer-Aided Design (CAD) model of the G2 Express IVC filter, which was generated using the Siemens NX 6 software package [61], from Rick Schraf and Todd Fetterolf for use in our study.

4.5 Virtual IVC Filter Placement

The subsequent step in the pipeline is to virtually place the geometric model of the IVC filter (i.e., the CAD model) inside the IVC by simulating the bending of the filter arms and legs which occur during the IVC filter insertion procedure using finite element modeling and a superelastic constitutive law. We simulate the resulting deformation of the IVC based on a linear elastic constitutive law. We now describe the two-step process of setting up the forces to compress the filter and then releasing the forces until the filter arms and legs touch the walls of the IVC, which then slightly deforms the IVC.

4.5.1 Superelasticity-Based Mesh Warping

The arms and legs of the IVC filter are made from nitinol wires; nitinol is a superelastic material [19]. Nitinol is a shape memory alloy [18], i.e., it can revert back its original shape after undergoing large deformations. Such materials are also used to manufacture stents, which are used to dilate constricted blood vessels. Nitinol also has applications in orthodontics [48].

In order to simulate the mechanical behavior of the IVC filter during the IVC filter insertion procedure, the undeformed filter is first moved to a location recommended by a physician or to the location observed in the patient's CT images. Since the IVC's diameter is smaller than the IVC filter's diameter, in our numerical model, the IVC filter wires protrude out of the IVC surface mesh after this step has been performed. Cylindrically-inward forces are then applied on each of the arms and legs of the IVC filter in order to compress it so that it completely fits inside the IVC. In the next step, the forces are reduced until the IVC filter legs become close to the IVC walls.

This two-step process is necessary because the stress versus strain curve for the loading and unloading of nitinol follow different paths. During the IVC filter insertion procedure, the IVC filter is first compressed and is then fed into a thin tube called a deployment sheath. The sheath as well as the IVC filter are then inserted into the body either through the groin or the neck and are guided into place in the IVC. The IVC filter is then released at the desired location inside the IVC. Our two-step process mimics the loading and unloading processes acting on the IVC filter during the procedure. The forces on the filter legs in the IVC are used to deform the IVC.

Abaqus [1] was used to perform the finite element modeling for virtual filter placement of the IVC filter surface mesh inside the IVC. Numerical models for nitinol and the constitutive law for superelasticity which we employ are implemented in Abaqus. Nitinol's superelastic behavior is implemented in the Nitinol Umat subroutine in Abaqus [66]. The model is based on the uniaxial behavior of a thin nitinol wire. Table 1 shows the values of the parameters used as input to the Umat subroutine in Abaqus. These values were obtained from [11]. The deformation of the IVC filter is modeled using a superelastic constitutive law [55].

Table 1 Values of the parameters for the superelastic material properties of nitinol [55]

Parameter	Description	Value
E_A	Austenite elasticity	35877 MPa
ν_A	Austenite Poisson's ratio	0.33
E_M	Martensite elasticity	24462 MPa
ν_B	Martensite Poisson's ratio	0.33
e^L	Transformation strain	0.0555
$(\partial\sigma/\partial T)_L$	Loading	6.7
σ_L^S	Start of transformation loading	489 MPa
σ_L^E	End of transformation loading	572 MPa
T_0	Temperature	37°C
$(\partial\sigma/\partial T)_U$	Unloading	6.7
σ_U^S	Start of transformation unloading	230 MPa
σ_U^E	End of transformation unloading	147 MPa

4.5.2 Linear Elasticity-Based Mesh Warping

It has been observed in some patient images that the circular or elliptical cross-section of the IVC deforms into a hexagonal cross-section upon insertion of the IVC filter due to the forces applied by the deformed filter. Once the IVC filter mesh is virtually warped to be completely inside the IVC, the IVC is deformed based on the forces acting on the filter. The forces on the IVC filter are cylindrically inward, and, thus, the forces on the IVC are of the same magnitude but are cylindrically outward. The vein surface vertices on which the forces are applied are chosen such that they are close to the filter vertices with which they are most likely to come in contact. Once the deformation of the vertices is obtained, we use the final vertex positions to create the volume mesh for the CFD simulations of blood flow.

Abaqus was also used for numerical modeling of the deformation of the IVC. For modeling the IVC's linear elastic behavior, we used a value of $E = 2.6 * 10^6$ dyn/cm^2 for the Young's modulus of the IVC. This value corresponds to the Young's modulus of an artery [25], which we used since its value has not yet been determined for a vein.

A surface interaction module is present in Abaqus in which it is possible to detect and impose conditions on contact interaction between the surfaces of the IVC filter and the IVC. We have found that the numerical solvers in Abaqus fail to converge when this module is used for our specific application. Because the module also failed to provide an accurate solution for modeling force interaction between a stent and a blood vessel in [11], we modified our computational technique so that the IVC filter legs come close to the IVC wall but do not come in contact with it. This was necessary in order to obtain convergence of the solvers. However, the CFD results should not be significantly affected by this modification because the distance between the IVC filter legs and the IVC wall is minimal.

4.6 Volume Mesh Generation

The next step in the computational pipeline is generation of the volume mesh using the surface meshes obtained from the above steps. We generate a volume mesh outside of the IVC filter and inside the walls of the IVC and its surrounding veins. As blood cannot flow into the surface of the filter wall, a volume mesh of the IVC filter is not needed for our CFD analysis (however; for fluid-structure interaction problems, a volume mesh of the filter may be necessary). The AFLR3 software package [2] allows us to generate such meshes using the advancing front technique with local reconnection, which we explain below.

4.6.1 Advancing Front Technique

The advancing front technique [39] generates a volume mesh from a valid surface mesh of the computational domain. A front, which is defined as a set of 2D elements, propagates from the boundary of the computational domain (i.e., its surface mesh) toward its interior. As the front propagates, volume elements are generated and are added to the volume mesh. AFLR3 generates prismatic elements on the boundary and pyramidal and tetrahedral elements elsewhere in the domain. The distance by which the front propagates inward from the surface mesh is controlled by setting the initial distance and increasing it geometrically. Boundary layers are also generated in the mesh near the IVC walls in order to obtain physically-accurate CFD simulations of the blood flow in these areas.

4.7 Computational Fluid Dynamics

Given a high-fidelity volume mesh, the final step in the computational pipeline is the simulation of blood flow in the patient-specific IVC model using CFD. Depending on the problem of interest, steady or unsteady, laminar or turbulent calculations may be carried out. In this study, the open-source computational continuum mechanics library OpenFOAM [45] was utilized to simulate steady, laminar blood flow through the IVC. Blood was assumed to behave as a Newtonian fluid with a kinematic viscosity of 4.4 cSt. The flow rate in each renal and iliac vein was specified as 0.75 L/min and 0.6 L/min, respectively, yielding infrarenal and suprarenal IVC flow rates of 1.2 L/min and 2.7 L/min, respectively [10].

The semi-implicit method for pressure linked equations (SIMPLE) algorithm was used to solve the incompressible continuity and Navier–Stokes equations, i.e.,

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}. \end{aligned}$$

Iterative convergence of the SIMPLE solver was guaranteed by forcing the so-

Algorithm 1 The computational pipeline to generate patient-specific volume meshes of the IVC and IVC filter from patient CT images and a CAD model of the IVC filter and to perform CFD simulations of blood flow on them

```
1: Acquire patient CT images
2: Segment each patient CT image using one of the following image segmentation
   techniques:
   • Region growing
   • Gradient-based region growing
   • Intelligent scissors
3: Generate the surface mesh using the marching cubes algorithm.
4: if the surface is overrefined then
5:     Coarsen the mesh using the Poisson surface reconstruction algorithm.
6: end if
7: Smooth the mesh using the HC-Laplacian smoothing algorithm.
8: Generate a geometric model (e.g., a CAD model) of the IVC filter
9: Virtually place the geometric model of the IVC filter in the smooth surface mesh
   of the IVC and deform the legs and arms of the IVC filter by simulating the IVC
   filter insertion procedure using a superelastic constitutive law for the IVC filter
   and a linear elastic constitutive law for the IVC.
10: Generate a volume mesh with boundary layer elements using the advancing
    front algorithm.
11: Perform CFD simulations of the blood flow.
```

lution residuals to be less than approximately 10^{-4}. Additionally, various solution variables were monitored throughout the simulation to ensure convergence of the computed result. Computations were performed on 120 processors of a high-performance parallel computer cluster at Penn State, each simulation requiring approximately 6 h of wall clock time to obtain a converged solution.

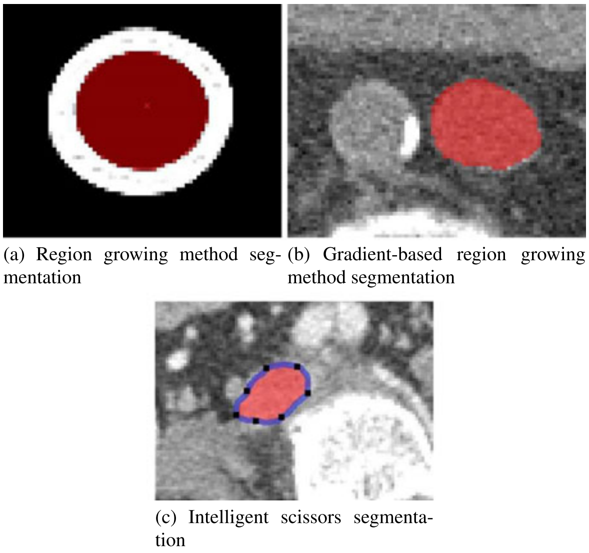
Our computational pipeline is summarized in Algorithm 1.

5 Results from Our Computational Pipeline

In the previous section, we described our computational pipeline for generation of high-fidelity, patient-specific volume meshes of the IVC anatomy and the IVC filter and CFD simulation of blood flow on these meshes. In this section, we discuss the results we obtained from each step in the pipeline.

5.1 Image Segmentation

We use the region growing technique for segmentation of our high-quality idealized IVC model CT images. Because the images had very little noise, the edge detection



(a) Region growing method segmentation

(b) Gradient-based region growing method segmentation

(c) Intelligent scissors segmentation

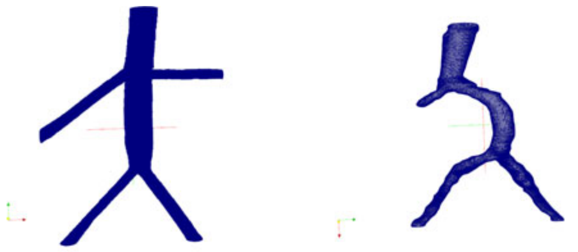
Fig. 5 Image segmentation of various patient CT images using the region growing method, gradient-based region growing method, and the intelligent scissors method: (**a**) idealized IVC model, (**b**) left IVC model, and (**c**) retroaortic IVC model

process was rather easy. Figure 5(a) shows the segmentation of the one of the CT images of the idealized model using this technique.

As patients cannot be subjected to high-level radiation, the patient CT images we obtained were of lower resolution. Figure 5(b) shows the segmentation of the one of the patient CT images from the left IVC model using the gradient-based region growing technique. Figure 5(c) shows the segmentation of another patient CT image from the retroaortic model using the intelligent scissors technique. The blue border shows the path traced by the algorithm, and the black square dots are the pixels using which the relevant shortest paths were found. Because the patient images we obtained were not taken at high resolution, we were only able to identify and segment the renal veins and not all of their branching vessels in the left IVC patient model. Similarly, we were only able to identify and segment one renal vein in the retroaortic IVC patient model. The renal veins are upstream of the IVC filter, and, thus, the results from our CFD simulations of blood flow using these volume meshes should not be significantly affected due to the absence of one renal vein or the branching vessels from each IVC model.

5.2 *Surface Mesh Generation*

After segmentation of the images, we generate a surface mesh of the IVC model using the marching cubes algorithm. Figure 6 shows the surface meshes generated for the three models, i.e., the idealized, left, and retroaortic IVC models generated using the marching cubes algorithm.



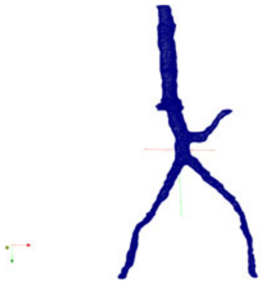


Fig. 6 Surface meshes of the three IVC models on which the blood flow simulation were carried out: (**a**) idealized IVC model, (**b**) left IVC model, and (**c**) retroaortic IVC model

The idealized IVC mesh is overrefined, and hence it is reconstructed using Poisson's reconstruction technique and smoothed using the HC-Laplacian smoothing algorithm. The meshes of the left and retroaortic IVC models are also smoothed using this algorithm. Figures 7(a) and (b) show the overrefined mesh of the idealized model and the reconstructed and smoothed mesh of the same model, respectively. Figures 8(a) and (b) show the unsmoothed and final smoothed meshes of the retroaortic IVC model, respectively.

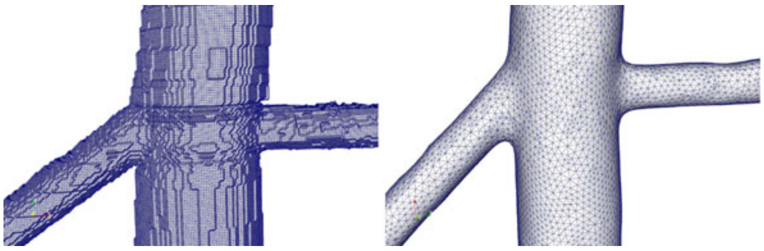
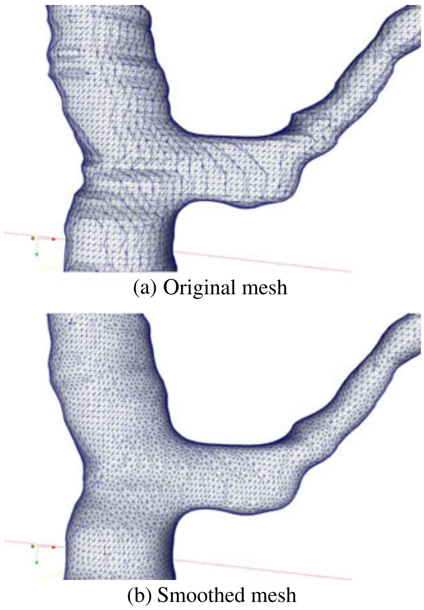


Fig. 7 Meshes generated from reconstruction and smoothing: (**a**) the original mesh of the idealized IVC model constructed using the marching cubes algorithm; (**b**) the mesh reconstructed using the Poisson equation-based algorithm and smoothed using the HC-Laplacian algorithm


(a) Original mesh

(b) Smoothed mesh

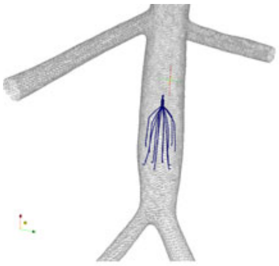
Fig. 8 (**a**) The original mesh of the retroaortic IVC model constructed using the marching cubes algorithm. (**b**) The mesh smoothed using the HC-Laplacian algorithm

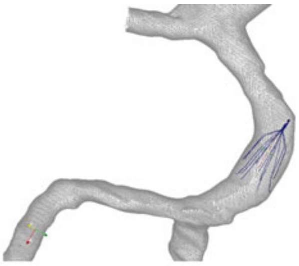
5.3 Virtual IVC Filter Placement

Virtual placement of the IVC filter was performed as described in Sect. 4.5. Figure 9 shows the deformed filter and the deformed IVC surface meshes. Since arteries are less elastic than veins, we did not see a major deformation in the IVC due to the forces applied by the IVC filter.

5.4 Volume Mesh Generation

In each of our volume meshes, we generated 10 boundary layers composed of prismatic elements. The initial boundary layer thickness is set to 0.00005 meters, and the geometric ratio by which the thickness increases is set to 1.1. After the boundary layer elements are constructed, tetrahedral and pyramidal elements are generated in the interior of the domain and added to the volume mesh. The quality of the elements are then improved using local reconnection (i.e., edge swaps and face swaps) subject to min-max type quality criteria in which the quality of the worst element improved by targeted local reconnection operations. A cut-away view of each of the volume meshes for the three IVC models are shown in Fig. 10.


(a) Idealized IVC mode


(b) Left IVC model

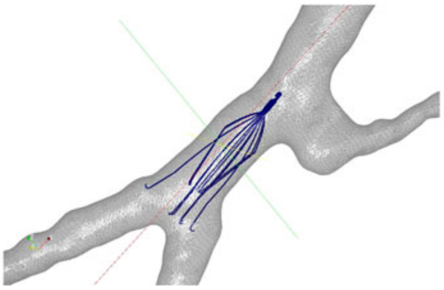
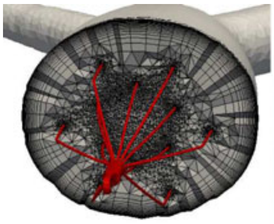
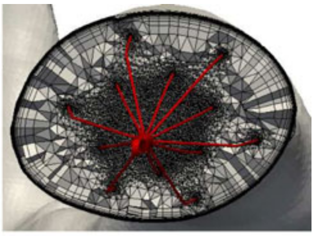

(c) Retroaortic IVC model

Fig. 9 Surface meshes of the three IVC models with the deformed filter inside: (**a**) idealized IVC model, (**b**) left IVC model, and (**c**) retroaortic IVC model. The IVC filter undergoes a superelastic deformation, and the IVC undergoes a linear elastic deformation during the IVC filter insertion procedure, which we simulate using a finite element method


(a) Idealized model


(b) Left IVC model

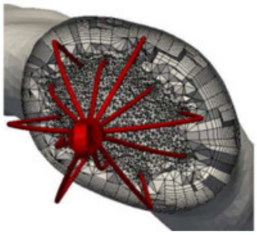
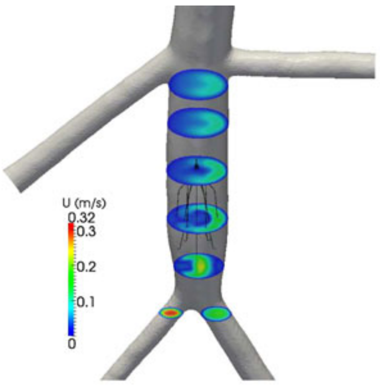

(c) Retroaortic IVC model

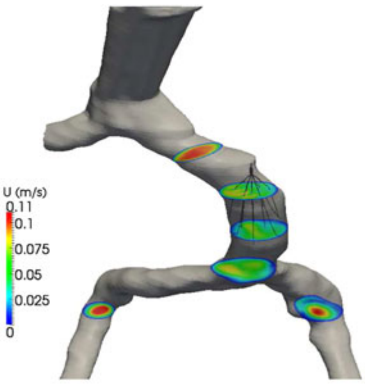
Fig. 10 Cut-away views of the volume meshes on the (**a**) idealized, (**b**) left, and (**c**) retroaortic IVC models showing the IVC filter and boundary layer

5.5 Computational Fluid Dynamics

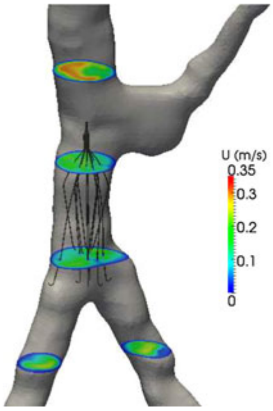
As shown in Fig. 11, blood flow in the IVC is quite complex, consisting of vortical structures downstream of the iliac veins, where mixing of the inflow occurs. Based on the present CFD results, for steady flow at physiologically-realistic flow rates,



(a) Idealized model



(b) Left IVC model



(c) Retroaortic IVC model

Fig. 11 Transverse contours of velocity magnitude extracted from CFD simulation results in the (**a**) idealized, (**b**) left, and (**c**) retroaortic IVC models

blood flow in the anatomically-accurate models is less disturbed, consisting of less separated flow, at the level of the IVC filter compared to the idealized model. Such differences in flow patterns may significantly affect clot capture and optimal IVC filter location. Additionally, since such disparate flow patterns likely occur in different patient anatomies, this suggests the need for patient-specific modeling of blood flow in the IVC for optimized filter selection and placement.

6 Conclusions and Future Work

We have proposed a computational pipeline approach for generation of patient-specific geometric models and computational meshes of the IVC and IVC filter based on patient CT images and a CAD model of the IVC filter. Our computational pipeline generates anatomically-correct geometric models of the IVC and its surrounding veins by processing patient CT images. In particular, the 2D CT images are segmented and are formed into a 3D geometric model using the marching cubes algorithm. A high-quality surface mesh is then generated on the IVC model. Mesh smoothing is then used to improve the quality of the surface mesh. A geometric model (represented as a surface mesh) of the IVC filter is then created. The IVC filter model is then virtually placed inside the IVC model using a superelastic mesh warping algorithm which simulates the compression and expansion of the IVC filter arms and legs which are composed of nitinol. The corresponding deformation of the IVC is simulated using a linear elastic mesh warping algorithm. The inlet and outlet surfaces of the IVC are rebuilt to be planar and perpendicular to the blood flow as necessary. A volume mesh of the IVC and IVC filter for the left, retroaortic, and idealized IVC models is then generated and is then generated and is used in CFD simulations of blood flow, whereby we illustrate the potential utility of this approach for optimized, patient-specific IVC filter selection and placement for improved treatment of PE.

Our computational pipeline approach is the first semi-automatic technique for generation of patient-specific models and computational meshes of the IVC and IVC filter based on patient CT images of the IVC and surrounding veins and a CAD model of the IVC filter. The novelty in our approach lies in the use of a superelastic mesh warping method to perform virtual implantation of the IVC filter in the IVC. Other virtual implantation techniques, such as those used for implantation of stents, have been solely geometric, as opposed to geometric and physical, in nature. However, the use of physics-based mesh warping techniques allows us to simulate more closely the IVC filter insertion procedure.

We are also the first group to have investigated the effect that patient anatomy has on the performance of an IVC filter. In particular, we investigated the performance of the G2 Express filter in the left IVC, retroaortic IVC, and normal IVC patient anatomies. Earlier work by other researchers has focused on the performance of various IVC filters in the normal IVC anatomy.

The strength of our computational pipeline lies in its ability to be used to investigate the performance of other IVC filters in various IVC anatomies. Because our approach is semi-automatic, several patient-specific geometric models and computational meshes can be generated with less effort by researchers. Previous techniques for generating these involved manual insertion of the IVC filter into the IVC and were more time consuming. Another advantage of our computational pipeline is its flexibility which allows for it to be used to generate patient-specific geometric models and computational meshes for other implanted medical devices composed of nitinol, such as stents or orthodontics.

Future work will focus on automation of the virtual implantation aspect of the computational pipeline, as well as further studies involving patient-specific geometric models and computational meshes of the IVC and IVC filter.

Acknowledgements The authors would like to thank Rick Schraf and Todd Fetterolf for creating the CAD model of the IVC filter. They would also like to thank Katerina Papoulia, of the University of Waterloo, for helpful discussions on the superelasticity computations. The work of the first three authors was supported in part by NSF CAREER Award OCI-1054459. The work of the first, third, fifth, and sixth authors was supported in part by an Institute for Cyberscience grant from The Pennsylvania State University. The work of the third and fifth authors was also supported in part by a grant from the Grace Woodward Foundation at The Pennsylvania State University. The work of the third author was also supported in part by NSF grant CNS-0720749. The work of the seventh author was supported by a Royal Thai Government Scholarship.

References

1. Abaqus 6.9. http://www.simulia.com/products/abaqus_fea.html
2. Aflr3. http://www.simcenter.msstate.edu/software.php
3. Amira 4.1.2. http://www.amira.com/. Mercury Computer Systems
4. Anderson F et al (1991) A population-based perspective of the hospital incidence and case-fatality rates of deep-vein thrombosis and pulmonary embolism: the Worcester DVT study. Arch Intern Med 151:933–938
5. Bah MT, Nair PB, Browne M (2009) Mesh morphing for finite element analysis of implant positioning in cementless total hip replacement. Med Eng Phys 31:1235–1243
6. Baldwin MA, Langenderfer JE, Rullkoetter PJ, Laz PJ (2010) Development of subject-specific and statistical shape models of the knee using an efficient segmentation and mesh-morphing approach. Comput Methods Programs Biomed 97:232–240
7. Barber DC, Oubel E, Frangi AF, Hose DR (2007) Efficient computational fluid dynamics mesh generation by image registration. Med Image Anal 11:648–662
8. Brountzos EN, Kaufman JA, Venbrux AC, Brown PR, Harry J, Kinst TF, Klenshinski S, Ravenscroft AC (2003) A new optional vena cava filter: retrieval at 12 weeks in an animal model. J Vasc Interv Radiol 14:763–772
9. Cebral J, Lohner R (1999) From medical images to CFD meshes. In: Proc. of the 8th international meshing roundtable
10. Cheng CP, Herfkens RJ, Taylor CA (2003) Inferior vena caval hemodynamics quantified in vivo at rest and during cycling exercise using magnetic resonance imaging. Am J Physiol, Heart Circ Physiol 284(4):1161–1167
11. Conti M (2007) Finite element analysis of self-expanding braided wirestent. Master's thesis, Ghent University, Belgium
12. Couch GG, Johnston KW, Ojha M (2000) An in vitro comparison of the hemodynamics of two inferior vena cava filters. J Vasc Surg 31:539–549
13. Craven B, Neuberger T, Paterson E, Webb A, Josephson E, Morrison E, Settles G (2007) Reconstruction and morphometric analysis of the nasal airway of the dog (*Canis familiaris*) and implications regarding olfactory airflow. Anat Rec 290:1325–1340
14. Craven B, Paterson E, Settles G, Lawson M (2009) Development and verification of a high-fidelity computational fluid dynamics model of canine nasal airflow. J Biomech Eng 131:1–11
15. de Putter S, Laffargue F, Breeuwer M, van de Vosse FN, Gerritsen FA (2006) Computational mesh generation for vascular structures with deformable surfaces. Int J Comput Assisted Radiol Surg 1:39–49
16. Decousus H et al (1998) A clinical trial of vena caval filters in the prevention of pulmonary embolism in patients with proximal deep-vein thrombosis. N Engl J Med 338:409–415
17. Dheeravongkit A, Shimada K (2007) Inverse adaptation of a hex-dominant mesh for large deformation finite element analysis. Comput Aided Des 39:427–438
18. Duerig TW (1995) Present and future applications of shape memory and superelastic material. In: Proc. of material for smart systems, vol 360. Material Reasarch Society, Warrendale, pp 497–506
19. Duerig TW, Pelton AR, Stockel D (1996) The use of superelasticity in medicine. In: Fachzeitschrift fur Handle Wirtschaft, Technik und Wissenschaft, Sonderdruck aus Heft, vol 9/96. Metall Verlag-Huthig GanbH, Pforzheim, pp 569–574

20. Dyedov V, Einstein DR, Jiao X, Kuprat AP, Carsona JP, del Pin F (2009) Variational generation of prismatic boundary-layer meshes for biomedical computing. Int J Numer Methods Eng 79:907–945
21. Fillinger M, Raghavan M, Marra S, Cronenwett J, Kennedy F (2002) In vivo analysis of mechanical wall stress and abdominal aortic aneurysm risk. J Vasc Surg 26:589–597
22. Freitag L, Ollivier-Gooch C (2000) A cost/benefit analysis for simplicial mesh improvement techniques as measured by solution efficiency. Int J Comput Geom Appl 10:361–382
23. Frey P (2004) Generation and adaptation of computational surface meshes from discrete anatomical data. Int J Numer Methods Eng 60:1049–1074
24. Frey PJ, Sarter B, Gautherie M (1994) Fully automatic mesh generation for 3D domains based upon voxel sets. Int J Numer Methods Eng 37:2735–2753
25. Greenfield JC Jr, Griggs DM Jr (1963) Relation between pressure and diameter in main pulmonary artery of man. J Appl Physiol 18:557–559
26. Gundert TJ, Shadden SC, Williams AR, Koo B-K, Feinstein JA, LaDisa JF Jr (2011) A rapid and computationally inexpensive method to virtually implant current and next-generation stents into subject-specific computational fluid dynamics models. Ann Biomed Eng 39(5):1423–1437
27. Hao J, Shen Y (2006) Region growing within level set framework: 3-D image segmentation. In: Proc. of the 6th world congress on intelligent control and automation (WCICA 2006), pp 10352–10355
28. Haralick RM, Shapiro LG (1985) Image segmentation techniques. Comput Vis Graph Image Process 29:100–132
29. Hartmann E (1998) A marching method for the triangulation of surfaces. Vis Comput 14:95–108
30. Hilton A, Illingworth J (1997) Marching triangles: Delaunay implicit surface triangulation. Technical report, University of Surrey
31. Hoekstra A, Hoogeveen Y, Elstrodt JM, Tiebosch AT (2003) Vena cava filter behavior and endovascular response: an experimental in vivo study. Cardiovasc Interv Radiol 26:222–226
32. Holmes W, Cotton R, Xuan VB, Rygg A, Craven B, Abel R, Slack R, Cox J (2011) Three-dimensional structure of the nasal passageway of a hagfish and its implications for olfaction. Anat Rec 294(6):1045–1056
33. Ito Y, Shum PC, Shih AM, Soni BK, Nakahashi K (2006) Robust generation of high-quality unstructured meshes on realistic biomedical geometry. Int J Numer Methods Eng 65:943–973
34. Katsamouris AA, Waltman AC, Delichatsios MA, Athanasoulis CA (1988) Inferior vena cava filters: in vitro comparison of clot trapping and flow dynamics. J Radiol 166:361–366
35. Kazhdan M, Bolitho M, Hoppe H (2006) Poisson surface reconstruction. In: Proc. of the fourth eurographics symposium on geometry processing. SGP. Eurographics Association, Aire-la-Ville, pp 61–70
36. Kinney TB (2003) Update on inferior vena cava filters. J Vasc Interv Radiol 14:425–440
37. Kunz RF, Haworth DC, Porzio DP, Kriete A (2009) Progress towards a medical image through CFD analysis toolkit for respiratory assessment on a clinical timescale. In: Proc. of 2009 international symposium on biomedical imaging: from nano to macro, pp 382–385
38. Liu Y, D'Arceuil H, He J, Duggan M, Gonzalez G, Pryor J, de Crespigny A (2006) A nonlinear mesh-warping technique for correcting brain deformation after stroke. Magn Reson Imaging 24:1069–1075
39. Löhner R (1996) Progress in grid generation via the advancing front technique. Eng Comput 12:186–210
40. Lohner R (1996) Regridding surface triangulations. J Comput Phys 126:1–10
41. Lorensen W, Cline H (1987) Marching cubes: a high resolution 3D surface construction method. Comput Graph 21:163–169
42. Mercury Computer Systems (2009) Amira reference guide
43. Meshlab v1.3.1. http://meshlab.sourceforge.net/
44. Mortensen E, Barrett W (1995) Intelligent scissors for image composition. In: Proc. of the 22nd annual conference on computer graphics and interactive techniques. SIGGRAPH. ACM,

New York, pp 191–198
45. OpenFOAM website. http://www.openfoam.com
46. Park J, Shontz SM, Drapaca CS (2012) A combined level set/mesh warping algorithm for tracking brain and cerebrospinal fluid evolution in hydrocephalic patients. In: Image-based geometric modeling and mesh generation. Springer, Dordrecht
47. Peiró J, Giordana S, Griffith C, Sherwin S (2002) High-order algorithms for vascular flow modelling. Int J Numer Methods Fluids 40:137–151
48. Pelton AR, Stockel D, Duerig TW (1999) Medical uses of nitinol. In: Proc. of the international symposium on shape memory materials. Material science forum, vols 327–328, pp 63–70
49. Persson P (2004) Mesh generation for implicit geometries. PhD thesis, MIT
50. Persson P (2006) Mesh size functions for implicit geometries and PDE-based gradient limiting. Eng Comput 22:95–109
51. Pham DL, Xu C, Prince JL (2000) Current methods in medical image segmentation. Annu Rev Biomed Eng 2:315–337
52. Qian Z, Yasul K, Nazarian GK, Vlodaver Z, Hunter DW, Castenda-Zuniga WR, Amplatz K (1994) In vitro and in vivo experimental evaluation of a new vena cava filter. J Vasc Interv Radiol 5:513–518
53. Quatember B, Muhlthaler H (2003) Generation of CFD meshes from biplane angiograms: an example of image-based mesh generation and simulation. Appl Numer Math 46:379–397
54. Rahbar E, Mori D, Moore JE Jr (2011) Three-dimensional analysis of flow disturbances in inferior vena cava filters. J Vasc Interv Radiol 22:835–842
55. Rogers C, Schief WK, Chow KW (2007) A novel class of model constitutive laws in nonlinear elasticity: construction via Loewner theory. Theor Math Phys 152:1030–1042
56. Schmidt J, Johnson C, Eason J, McLeod R (1994) Applications of automatic mesh generation and adaptive methods in computational medicine. In: Modeling, mesh generation, and adaptive numerical methods for partial differential equations. Springer, Berlin
57. Sethian JA (1999) Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge Monographs on Applied and Computational Mathematics
58. Shepherd JF, Johnson CR (2009) Hexahedral mesh generation for biomedical models in SCIRun. Eng Comput 25(1):97–114
59. Shewchuk J (2002) What is a good linear element? Interpolation, conditioning, and quality measures. In: Proc. of the 11th international meshing roundtable, pp 115–126
60. Shontz SM, Vavasis SA (2003) A mesh warping algorithm based on weighted Laplacian smoothing. In: Proc. of the 12th international meshing roundtable, pp 147–158
61. Siemens NX 6. http://www.plm.automation.siemens.com/en_us/products/nx/
62. Sigal IA, Whyne CM (2010) Mesh morphing and response surface analysis: quantifying sensitivity of vertebral mechanical behavior. Ann Biomed Eng 38:41–56
63. Sigal IA, Hardisty MR, Whyne CM (2008) Mesh-morphing algorithms for specimen-specific finite element modeling. J Biomech 41:1381–1389
64. Sigal IA, Yang H, Roberts MD, Downs JC (2010) Morphing methods to parameterize specimen specific finite element model geometries. J Biomech 43:254–262
65. Silver-Thorn MB (1991) Predication and experimental verification of residual limb-prosthetic socket interface pressures for below-knee amputees. PhD thesis, Northwestern University
66. SIMULIA (2010) Abaqus/cae user manual
67. Singer MA, Wang SL (2011) Modeling blood flow in a tilted inferior vena cava filter: does tilt adversely affect hemodynamics? J Vasc Interv Radiol 22:229–235
68. Singer MA, Henshaw WD, Wang SL (2009) Computational modeling of blood flow in the TrapEase inferior vena cava filter. J Vasc Interv Radiol 20:799–805
69. Singer MA, Wang SL, Diachin DP (2010) Design optimization of vena cava filters: an application to dual filtration devices. J Biomech Eng 132:101006
70. Sion M, Rabkin DJ, Kleshinski S, Kim D, Ransil BJ (1993) Comparative evaluation of clinically available inferior vena cava filters with and in vitro physiologic simulation of the vena cava. J Radiol 189:769–774

71. Stewart SF, Robinson RA, Nelson RA, Malinauskas RA (2008) Effects of thrombosed vena cava filters on blood flow: flow visualization and numerical modeling. Ann Biomed Eng 36:1764–1781
72. Swaminathan TN, Hu HH, Patel AA (2006) Numerical analysis of the hemodynamics and embolus capture of a Greenfield vena cava filter. J Biomech Eng 128:360–370
73. Szczerba D, McGregor R, Szekely G (2007) High quality surface mesh generation for multi-physics bio-medical simulations. In: Proc. of the 2007 international conference on computational science, vol 4487. Springer, Berlin, pp 906–913
74. Treece GM, Prager RW, Gee AH (1999) Regularised marching tetrahedra: improved iso-surface extraction. Comput Graph 23:593–598
75. Ulrich D, van Rietbergen B, Weinans H, Ruegsegger P (1998) Finite element analysis of trabecular bone structure: a comparison of image-based meshing techniques. J Biomech 31:1187–1192
76. Verma CS, Fischer PF, Lee SE, Loth F (2005) An all-hex meshing strategy for bifurcation geometries in vascular flow simulation. In: Proc. of the 14th international meshing roundtable, Sandia National Laboratories, pp 11–14
77. Vollmer R, Mencl R, Müller H (1999) Improved Laplacian smoothing of noisy surface meshes. Comput Graph Forum 18(3):131–138
78. Wang SL, Singer MA (2010) Toward an optimal position for inferior vena cava filters: computational modeling of the impact of renal vein inflow with Celect and TrapEase filter. J Vasc Interv Radiol 21:367–374
79. Yamakawa S, Shimada K (2008) Converting a tetrahedral mesh to a prism-tetrahedral hybrid mesh for FEM accuracy and efficiency. In: Proc. of the 2008 ACM symposium on solid and physical modeling, pp 287–294
80. Yu Z, Holst MJ, McCammon JA (2008) High-fidelity geometric modeling for biomedical applications. Finite Elem Anal Des 44:715–723
81. Zachariah SG, Sanders JE, Turkiyyah GM (1996) Automated hexahedral mesh generation from biomedical image data: applications in limb prosthetics. IEEE Trans Rehabil Eng 4:91–102
82. Zhang L, Yang G, Shen W, Qi J (2007) Spectrum of the inferor vena cava: MDCT findings. Abdom Imaging 32:495–503

Computational Techniques for Analysis of Shape and Kinematics of Biological Structures

Jia Wu and John C. Brigham

Abstract This chapter presents state-of-the-art methods for statistical shape analysis of biological structures obtained from sets of medical images. In particular, emphasis is placed on the techniques necessary to parameterize and then decompose a given set of 3D surfaces extracted from medical images. Shape representation methods such as medial representation (i.e., skeletonization) and harmonic topological mapping are presented as tools for parameterizing a given set of surfaces so that they can be quantitatively compared. Then, methods for statistical decomposition including the proper orthogonal decomposition (also called principal component analysis) and independent component analysis are shown for the decomposition of the set of parameterized shapes into the set of fundamental shape features that can be applied to cluster the shapes and build classifiers with the potential for relating shape characteristics to pathology. An example analysis is presented which applies these techniques to parameterize and decompose the shape change of two human right ventricles from cardiac CT scans throughout the cardiac cycle. Advantages, disadvantages, and the relevance of the described methods in clinical medical image applications will be addressed throughout the chapter.

1 Introduction

There are a multitude of pathologies that significantly and adversely affect human health, which are also known to induce noticeable changes in the shape and/or mechanical behavior of certain organs or other biological structures [4, 17, 18, 25, 29, 32]. For example, as shown in Fig. 1, pulmonary hypertension (PH) has been found to significantly alter both the shape and mechanical function of the right heart [8, 28, 29]. Developments in medical imaging methods and associated image processing techniques have gone a long way to aid physicians in both observing the na-

J. Wu (✉) · J.C. Brigham
Swanson School of Engineering, University of Pittsburgh, Pittsburgh, PA 15260, USA
e-mail: jiw59@pitt.edu

J.C. Brigham
e-mail: brigham@pitt.edu

Y.(J.) Zhang (ed.), *Image-Based Geometric Modeling and Mesh Generation*,
Lecture Notes in Computational Vision and Biomechanics 3,
DOI 10.1007/978-94-007-4255-0_13, © Springer Science+Business Media Dordrecht 2013

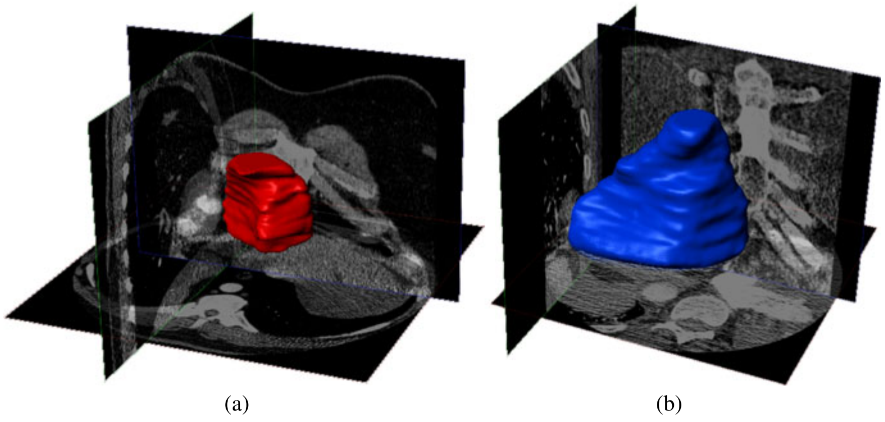


Fig. 1 Right ventricle endocardial surfaces segmented from ECG-gated cardiac CT scans for (**a**) a patient with normal hemodynamics and (**b**) a patient with pulmonary hypertension

ture of such shape changes and identifying diagnostic relationships between shape change and a particular pathology. However, there are situations where although a shape change is known to occur and the nature of this change is suspected to be linked with pathology, clinical scientists have been so far unable to find measurable quantitative features that discriminate various states in the pathology and/or treatment outcomes. For example, while PH is observed to dramatically change the size and shape of the right ventricle, features of this shape change have yet to be identified to accurately predict the likelihood of right ventricle failure that is often fatal.

There have been several efforts for various applications to choose specific (either local or global) features of shape a priori to quantitatively understand organ function and/or pathology. For example, the work by Vuille et al. and Lorenz et al. studied the left ventricle globally with respect to its bulk volume and mass [20, 33]. Alternatively, examples of local organ features include the work by Sacks et al. [26] that characterized right ventricle free wall geometry through curvature measures and Simon et al. [28] that analyzed PH's effect on the right ventricle with respect to regional wall thickness. However, there is typically no more statistical or physical reasoning for referencing the chosen organ features other than that they are intuitive, observable, and/or comparable among individuals, and the consistency and predictive capabilities of these metrics may therefore suffer. Efforts are made even more challenging when the shape of the anatomical structure of interest does not conform to standard geometric definitions, limiting the ability to make consistent quantitative comparisons among individuals and thus observe patterns within a set.

Some recent work has taken a more generalized approach to analyzing and contrasting the shape of a collection of biological structures from medical imaging data (see [14] and the references therein for examples of such approaches). What is particularly notable about these efforts is that at their foundation they attempt to develop a generalized mathematical description (i.e., parameterization) of the organ shapes.

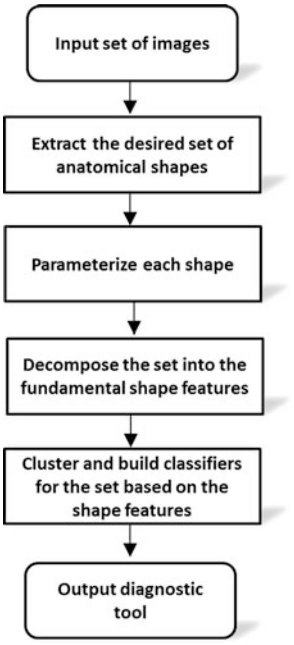

Input set of images
Extract the desired set of anatomical shapes
Parameterize each shape
Decompose the set into the fundamental shape features
Cluster and build classifiers for the set based on the shape features
Output diagnostic tool

Fig. 2 Framework for 3D statistical shape analysis methods

Once parameterized such that each shape is uniquely comparable to one another, pattern recognition approaches can be applied to optimally decompose, cluster, and build classifiers from a collection of shapes, with no a priori selection of the nature or location of the most significant discerning features. By not preselecting the features of interest, but rather allowing them to be naturally extracted from the data, the hope is that more statistically significant and physically meaningful features and classifications can be obtained than would be possible otherwise.

This chapter presents several modern approaches to analyze organ shape described by 3D surfaces through quantitative parameterizations and statistical pattern recognition techniques. Section 2 outlines a standard structure for this statistical shape analysis process as well as the details of several common parameterization and pattern recognition methods. Section 3 presents an example analysis and decomposition of human right ventricles from cardiac CT scans, which is followed by thoughts on potential future directions for this area of research.

2 Methods

The overall structure for the approach to statistical shape analysis considered herein is shown in Fig. 2. Provided a collection (either multiple time frames, several different patients, or both) of 3D medical image stacks, the first step is to extract the 3D surfaces that describe the anatomical structures of interest from each stack, which typically takes the form of a set of Cartesian point clouds and their connectivity along the surface. Note that in general even if all surfaces were properly registered (which is also not a trivial act), other than any given fiducial points, there is no inherent relationship between the points on any two surfaces within a population to allow for quantitative comparison in this state (e.g., the difference shape cannot be calculated and one shape cannot be projected onto another). There exist several examples of work that attempt to directly relate a collection of points on several surfaces through anatomical and/or geometric feature tracking, minimizing the average distance between corresponding points, or through various other similar metrics (e.g., [19, 21, 23]). However, relating surfaces directly in such ways is generally only applicable to sets of shapes with relatively small changes, which typically implies the shapes compared can only correspond to a single individual's biological structure (e.g., one patient's vessel at several instances during the cardiac cycle). As such, the second critical step is to parameterize each surface with respect to a common domain so that any two surfaces, whether selected from a single patient at different times or across a population, may be quantitatively compared. Next, the collection of parameterized surfaces can be decomposed to identify the fundamental shape features, and lastly those features can be used to cluster the shapes into significant categories and/or build classifiers for diagnostic purposes.

The focus of this chapter is on the analysis occurring after the given collection of medical images has been segmented and initially processed (e.g., smoothed to remove image noise and high-frequency segmentation error). Therefore, it is assumed for the following discussion that a collection of continuous, three-dimensional (3D), non-overlapping surfaces describing anatomical structures is provided, with the k^{th} surface domain labeled as $\Omega_k \subset \Re^3$, and all surfaces are defined in some consistent global coordinate system with spatial coordinates $\mathbf{x}$. Provided with such a dataset, the key components of the analysis approach presented are to uniquely map each surface to a unified reference state (i.e., parameterize) and then apply suitable pattern recognition techniques to decompose, cluster, and build classifiers. In particular, the focus herein will be on parameterization and decomposition, and approaches to accomplish these steps are detailed in the following subsections.

2.1 Parameterization

There are a variety of methods that have been implemented for the quantitative comparison of 3D shapes with different benefits and shortcomings (see [10] for an overview of several approaches to parameterize surfaces). However, while parameterization is an inherently ill-posed problem and there are many ways to perform such a mapping, of critical importance is that the mapping consistently provides unique and one-to-one representations of each surface. In other words, the parameters of the chosen mapping process should be unique and clearly defined for each surface, and once mapping is completed each point on a surface must have one unique corresponding point on every other surface and vice versa. An additional issue for clinical applications in medical imaging is that the parameterization should

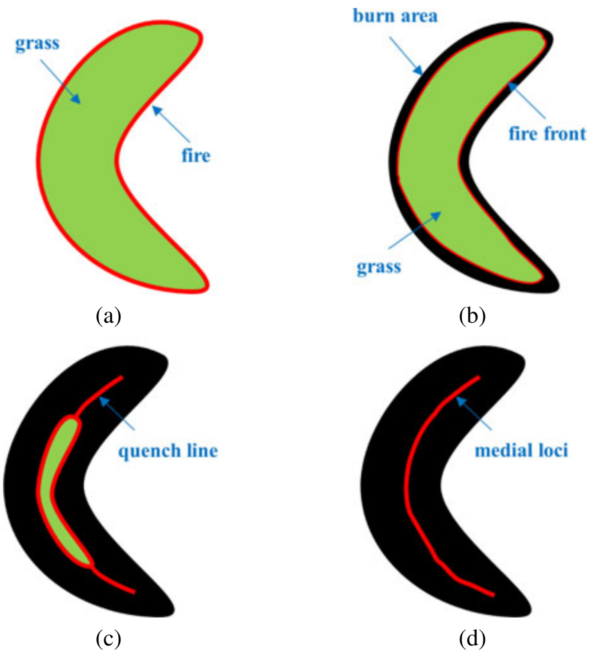


Fig. 3 Schematic of the grassfire analogy to define the medial axis of a shape showing (**a**) the initial shape as a grass field, (**b**) the early stages of the grass burning, (**c**) the fire beginning to quench, and (**d**) the final medial axis

ideally require minimal user input (e.g., anatomically defined reference points). This constraint is mainly due to current limitations on image tagging and implanting markers in a clinical setting leading to few possible fiducial points in many biological structures. Two very different parameterization approaches that are actively used for statistical analysis of shapes are the medial representation and harmonic mapping, and these methods will be focused on here.

2.1.1 Medial Representation

Medial representation is a widely used object representation method that has been successfully applied in several cases to biological structure shape analysis [34]. The purpose of this method is to define a unique collection of points within the object known as medial loci, which combine to form the medial axis or "skeletonization" of the object's surface. An intuitive way to picture these medial loci of an object and how they could be attained is through the grassfire burning analogy [5]. One can imagine the volume enclosed by the object's surface as a field of grass with uniform density, and then the entire surface of the object is instantaneously and simultaneously ignited with fire as shown in Fig. 3a. The front of the fire will propagate inward at uniform speed as shown in Fig. 3b until the front propagating from multiple directions meets at some point(s), which then causes the fire to quench at the meeting point as shown in Fig. 3c. Finally, when the entire fire is extinguished the collection of all points where fronts met and quenched is the collection of medial loci (Fig. 3d), and every point on the object surface is associated with a locus and a "burning time" that is simply the distance from the locus to the boundary point. Then the medial axis combined with the corresponding distance between the medial axis and the shape boundary (i.e., radius function) is sufficient to uniquely define the original surface. Once each shape has been skeletonized, the medial axes can be compared directly to analyze the shape differences, such as through the relative length and curvature of the axis segments. Alternatively, the medial axis can be used to register the shapes with respect to one another, and then the radius functions describing each shape can be used to quantitatively compare each shape continuously or point by point.

This approach is readily applicable to compare shapes with large variations and across populations. In addition, the medial axis approach is particularly unique to other shape analysis methods in that it allows for direct comparison of points throughout the entire volume of each shape, not only the surface points. However, conceptually this method tends to be best used with shapes that are cylindrical in nature, such as vessels, or generally shapes with a somewhat consistent skeletonization. Furthermore, obtaining the medial loci for any given shape is not a trivial problem in practice, and there are a large number of formal algorithms that have been developed to extract the medial axis of objects, including shocks of boundary evolution, Voronoi skeletons, grayscale, core tracking, distance transform, and so on [27].

As an example, the shocks of boundary evolution method is in essence an attempt to numerically simulate the grassfire burning analogy [31]. For the simplest possible cases, the approach starts by assuming that the boundary of interest is described by the vector $\mathbf{S}$ and the volume enclosed by the surface (i.e., inner grass) is isotropic and homogeneous (meaning the fire will flow at a constant rate in all directions). Then the evolution of the fire's front can be described by the rate equation

$$\frac{\partial \mathbf{C}(t)}{\partial t} = -\kappa \mathbf{N}, \tag{1}$$

where $\mathbf{C}(t)$ is the front of the fire at time t, $\mathbf{N}$ is the unit outward normal vector to $\mathbf{C}(t)$, κ is the flow speed of the fire, and the initial condition for the front corresponds to the original boundary such that $\mathbf{C}(0) = \mathbf{S}$ (note that time is an artificial parameter here, used to track the front evolution throughout the volume). Equation (1) can easily be solved incrementally, and then all that is necessary is to track the evolution of the front and cease the evolution of points when their position coincides with any other part of the front.

There also exist several methods that are not directly based on the grassfire analogy, such as the methods derived from Voronoi skeletons [22]. One such approach for a given surface begins by sampling the surface at n locations as shown in Fig. 4a (note that Fig. 4a shows a very small number of locations for illustrative purposes). Next, the domain is divided into n subvolumes based on the n points selected by drawing the midperpendicular surfaces for each pair of points so that each subvolume contains exactly one boundary point and all spatial locations within a subvolume are closer to the boundary point contained than any other point in the set

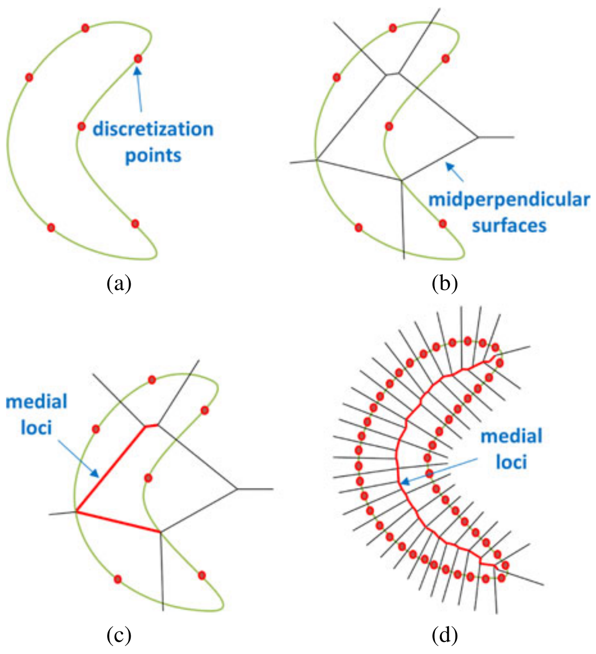


Fig. 4 Schematic of the Voronoi diagram method to obtain the medial axis of a shape showing (**a**) six boundary points sampled, (**b**) the midperpendicular surfaces for the six points, (**c**) the medial axis for the six points, and (**d**) the medial axis corresponding to a larger number of boundary points

(Fig. 4b). The medial axis is then defined by all finite segments of the midperpendicular surfaces that are wholly contained within the original surface (Fig. 4c). As the number of points selected from the boundary increases, the representation approaches the "true" medial axis (Fig. 4d), and in the limit matches the definition of the medial axis exactly.

2.1.2 Harmonic Mapping

Harmonic mapping to the unit sphere is an alternate parameterization approach that has seen considerable recent development for parameterization of 3D closed surfaces for statistical shape analysis. Harmonic mapping approaches have been shown to produce unique, one-to-one, and non-overlapping surface representations, but a key constraint of these approaches is that the shapes of interest must be closed genus-0 topologies. However, the genus-0 description can be a suitable approximation for many biological structures such as regions of the brain and heart, among others. If the shapes are suitable, some variation of harmonic mapping can be applied to map each shape to the surface of a unit sphere (i.e., transform the Cartesian coordinates to the spherical latitude and longitude) from which they can be quantitatively compared to one another. These techniques have been shown to be applicable to a wide variety of shapes and have been coupled with several decomposition and pattern recognition techniques [3, 6, 11, 14].

By definition, a harmonic mapping (i.e., change in coordinates) is simply a parameterization that satisfies Laplace's equation for each new parameter. As such, choosing the new parameters to be the spherical coordinates, ϕ (longitude) and θ (latitude), the spherical coordinates for each point on the given surface (i.e., location on the surface of the unit sphere) can be determined from the solution of the following differential equations

$$\begin{aligned} \nabla^2\theta(\mathbf{x}) &= 0 \quad \text{in } \Omega_k, \\ \nabla^2\phi(\mathbf{x}) &= 0 \quad \text{in } \Omega_k. \end{aligned} \tag{2}$$

Defining sufficient boundary conditions on ϕ and θ is then all that is necessary to uniquely determine a value of latitude and longitude on the unit sphere for every point on the original surface Ω_k. Once mapped, any given surface (defined by all $\mathbf{x} \in \Omega_k$) can be described continuously with respect to a common domain as

$$\mathbf{x} = \mathbf{x}(\theta, \phi) \quad \text{in } \theta \in [0, \pi], \phi \in [0, 2\pi]. \tag{3}$$

Therefore, the mapped shapes can be quantitatively compared continuously over the entire surfaces in terms of the spherical coordinates to assess variations, identify patterns, etc.

One perceived downside to directly applying the harmonic mapping is that the surface may be distorted in the mapped domain as a majority of the surface may have ϕ and/or θ values within a relatively small range, which may degrade the subsequent analysis. Therefore, a common practice is to add an additional relaxation step in the mapping process [12]. In the cases that define the original surface as a mesh, this relaxation typically seeks to optimize the mapping so that the elements in the mapped domain are as regularly distributed as possible (e.g., so that element internal angles and area are nearly constant in the range of ϕ and θ). Alternatively, the authors have developed and implemented a direct (i.e., no iterative optimization) two-step process for a consistent harmonic mapping that is completely deterministic and requires minimal user-defined referential data.

The harmonic mapping approach developed by the authors begins by solving Eq. (2) using a variation of the techniques shown in [6, 24, 35] to the define and implement the necessary boundary conditions for the spherical coordinates ϕ and θ in a way that requires a minimal amount of referential data and is not significantly dependent upon the surface mesh dimensions. For this approach each surface must have at minimum two reference points and a reference line: a point defined as the north pole (Γ_n), a point defined as the south pole (Γ_s), and a continuous non-overlapping line defined as the dateline (Γ_d) that connects the two poles. In practice, the references should be related to some identifiable component of the anatomy for consistent comparisons between surfaces. It should also be noted that the dateline has a nonunique mapped value as it represents where the longitude passes 360°. As such, the surface is cut along the dateline to create two overlapping, but independent boundaries: an east dateline (Γ_d) and west dateline (Γ_d^o). Thus, the values of the spherical coordinates along the references will be assigned as

$$\begin{aligned} \theta &= 0 \quad \text{on } \Gamma_n, \qquad & \theta &= \pi \quad \text{on } \Gamma_s, \\ \phi &= 0 \quad \text{on } \Gamma_d, \qquad & \phi &= 2\pi \quad \text{on } \Gamma_d^o. \end{aligned} \tag{4}$$

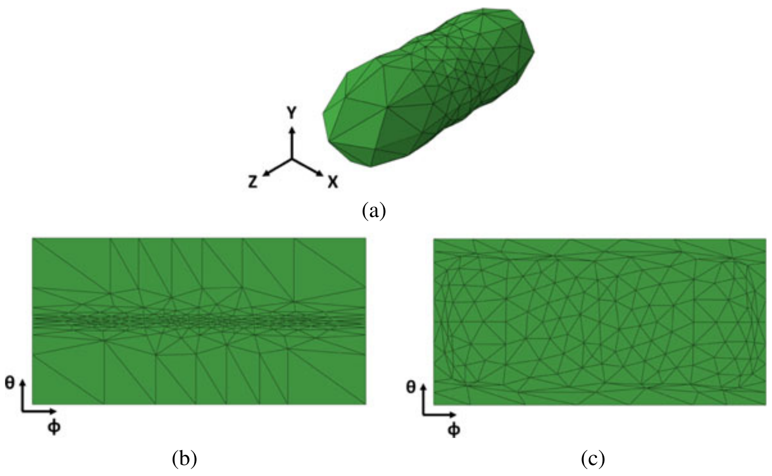


Fig. 5 Example of (**a**) a 3D genus-0 shape, (**b**) the results of the initial harmonic parameterization, and (**c**) the results of the secondary harmonic parameterization

The reference choices are entirely at the discretion of the user, and additional anatomical information can be easily added beyond the minimal data requirement through intermediate datelines or other similar boundary conditions. In addition, since the poles of a sphere have nonunique values for longitude, to solve for the longitudinal mapping, a region around each pole must be removed in the domain for the boundary value problem of ϕ in Eq. (2). This can typically be done by removing all elements connected to the poles when solving for ϕ. This is the only directly mesh-dependent step in the mapping process being presented. However, several tests have shown that the mapping results are not sensitive to changes in the overall mesh size, particularly the region eliminated for the longitudinal mapping. The resulting boundary value problems described by Eqs. (2) and (4) can then be solved through any preferred approach (e.g., finite element or boundary element method) for the spherical coordinates of each surface point. Lastly, the region removed for the longitudinal mapping is replaced in the mapped domain by connecting all of the northmost and southmost points to the pole and adding a vertex with the latitude value of the corresponding pole and longitude value of the connecting points.

As discussed, the results from this initial parameterization may not be ideal for further statistical analysis. As shown in the example in Fig. 5, the vertices (i.e., nodes) of the original surface mesh are relatively uniformly distributed over the original surface (Fig. 5a), but after mapping most of the vertices are concentrated in the middle third of the latitude domain (Fig. 5b). While not necessarily an issue in all circumstances, depending on the subsequent statistical analysis used, details may be lost due to the concentration of the surface information over a relatively small portion of the referential domain. To compensate for these potential difficulties (if necessary) a secondary mapping is performed to more uniformly distribute the mapped vertices over the domain of the spherical coordinates.

To improve the parameterization it can first be recognized that the harmonic parameterization of a perfect sphere defines the relationship between a uniformly distributed mesh over the unit sphere domain (the same domain as the mapped domain) and the distortion caused by the mapping. As such, the assumption can be made that the inverse of the harmonic mapping of the sphere will in some sense bring any distorted mapped mesh "closer" to a uniformly distributed mesh over the unit sphere domain. To apply this concept the initial parameterization procedure is performed for a unit sphere as the surface (Ω) to produce the mapping

$$\mathbf{X}(\theta_u, \phi_u) = \mathbf{x}(\theta, \phi), \tag{5}$$

where

$$\mathbf{X}(\theta_u, \phi_u) = \begin{bmatrix} \sin(\theta_u)\cos(\phi_u) \\ \sin(\theta_u)\sin(\phi_u) \\ \cos(\theta_u) \end{bmatrix}, \tag{6}$$

and θ_u and ϕ_u are the latitude and longitude of the unit sphere without mapping, respectively. Equations (5) and (6) can then be rearranged to produce the relationship between distorted and undistorted spherical coordinates as

$$\begin{aligned} \theta_u &= \theta_u(\theta, \phi), \\ \phi_u &= \phi_u(\theta, \phi). \end{aligned} \tag{7}$$

To apply this secondary parameterization to a given surface, it is then only necessary to substitute the results from the initial parameterization (Eq. (2)) into Eq. (7) to obtain the parameterization of the given surface in terms of θ_u and ϕ_u. Figure 5c shows an example of a parameterized surface after the secondary parameterization, with the final mesh being clearly more uniformly distributed than the mesh after only the initial parameterization. Of particular significance is that since both the initial and secondary mappings are unique and one-to-one, the final mapping between the original surface and the unit sphere ($\mathbf{x}(\theta_u, \phi_u)$ for Ω_k) will also be unique and one-to-one.

2.2 Decomposition

Once a set of n biological shapes is converted to a set of shape functions (i.e., parameterized), the application of a decomposition strategy to determine and rank the fundamental shape components that exist within the set is relatively straightforward. In general, most techniques to decompose a dataset can be formulated as identifying the m basis functions (i.e., modes) $\{\mathbf{v}_i(\theta, \phi)\}_{i=1}^{m}$ that are optimal in some sense for representing the given set of data. For a linear form, each surface is approximated as a combination of these global shape modes as

$$\mathbf{x}_k(\theta, \phi) \approx \overline{\mathbf{x}}(\theta, \phi) + \sum_{i=1}^{m} a_{ki}\mathbf{v}_i(\theta, \phi), \quad \text{for } k = 1, 2, \ldots, n, \tag{8}$$

where a_{ki} is the coefficient that best approximates the k^{th} shape with the i^{th} mode, and $\bar{\mathbf{x}}(\theta, \phi)$ accounts for a possible translation in the dataset. Depending on any prior knowledge of the nature of the dataset and the objective of the analysis, the translation function $\bar{\mathbf{x}}(\theta, \phi)$ can be taken as the null shape (i.e., zero), the mean shape (i.e., the average of the dataset), or some reference shape so that the analysis corresponds to change in shape (i.e., pseudo-deformation), which will all lead to different results and different physical interpretations. There are also a variety of decomposition procedures to identify the optimal set of modes with respect to Eq. (8), with varying benefits and shortcomings. Two of the relatively popular such techniques will be discussed in more detail in the following: proper orthogonal decomposition and independent component analysis.

2.2.1 Proper Orthogonal Decomposition

Proper orthogonal decomposition (POD), depending on context, is often interchangeably referred to as principal component analysis or the Karhunen–Loeve transform, but is differentiated here as POD since it will be formulated in the space of continuous functions. POD has been used successfully in a variety of pattern recognition and reduced-order modeling applications [1, 2, 7], and it is particularly beneficial to the shape analysis problem as it is well known to be tolerant to noise, and the continuous function formulation allows for a set of shape functions to be analyzed without further processing regardless of mesh conformity (e.g., varying mesh density and/or node distribution throughout the set).

The main feature of POD is that it defines the optimal basis $\{\mathbf{v}_i(\theta, \phi)\}_{i=1}^m$ as that which minimizes the average of the L_2-norm of the difference (i.e., mean squared error) between each shape function (referred to as a snapshot) and the best approximation of the snapshot leading to the following optimization problem

$$\min_{\{\mathbf{v}_i(\theta,\phi)\}_{i=1}^m} \left\langle \left\| \mathbf{x}_k(\theta, \phi) - \mathbf{x}_k^*(\theta, \phi) \right\|_{L_2}^2 \right\rangle, \tag{9}$$

where

$$\langle x_k \rangle = \frac{1}{n} \sum_{k=1}^{n} x_k, \tag{10}$$

and $\mathbf{x}_k^*(\theta, \phi)$ is the best approximation to $\mathbf{x}_k(\theta, \phi)$ through Eq. (8), which can be obtained using the projection operator. Lastly, through several manipulations including applying the method of snapshots, the optimal modes can be solved deterministically through the following eigenvalue problem (see [1] and the references therein for details)

$$\frac{1}{n} \sum_{k=1}^{n} A_{jk} C_k^{(i)} = \lambda^{(i)} C_j^{(i)}, \tag{11}$$

where

$$A_{jk} = \int_0^{2\pi} \int_0^{\pi} (\mathbf{x}_j - \overline{\mathbf{x}}) \cdot (\mathbf{x}_k - \overline{\mathbf{x}})\, d\theta\, d\phi, \tag{12}$$

and

$$C_k^{(i)} = \int_0^{2\pi} \int_0^{\pi} (\mathbf{x}_k - \overline{\mathbf{x}}) \cdot \mathbf{v}_i\, d\theta\, d\phi. \tag{13}$$

The n-dimensional eigenvalue problem can be solved to obtain at most n modes as

$$\mathbf{v}_i(\theta, \phi) = \frac{1}{\lambda^{(i)} n} \sum_{k=1}^{n} \big(\mathbf{x}_k(\theta, \phi) - \overline{\mathbf{x}}(\theta, \phi)\big) C_k^{(i)}. \tag{14}$$

An important note is that the corresponding eigenvalues ($\lambda^{(i)}$) relate to the relative importance of each mode, with the larger eigenvalues corresponding to modes that are more significantly representative of the dataset. As an added benefit, the lowest eigenvalues (i.e., lowest energy modes) are typically associated with noise, allowing for spurious components to be identified and ultimately removed. In practice, the modes themselves could provide a physical understanding of the fundamental aspects of shape and kinematics of a given biological structure. For classification purposes the highest energy modes are typically chosen (one rule of thumb used is to choose enough modes to capture 99% of the total eigenvalue sum of the set), and the coefficients of these modes for each surface in the set, which are obtained by projecting the surface onto each mode, are used to define the surfaces and build classifiers.

2.2.2 Independent Component Analysis

From a statistical point of view, since the POD modes are orthonormal, the linear transformation to these orthonormal bases decorrelates the data (i.e., removes the second-order dependence between the variables) if the translation function $\overline{\mathbf{x}}(\theta, \phi)$ is taken to be the mean shape. Independent Component Analysis (ICA) can be considered as an extension of POD as it considers not only these second-order statistics, but higher-order statistics as well. ICA does not constrain the modes to be orthogonal; rather, the criteria for the basis functions (independent components) is that they be as statistically independent as possible.

Similarly to POD, it is assumed that the observed data is a linear combination of these components (Eq. (8)). There are many approaches which attempt to solve this problem, the most popular of which are based on maximizing the non-Gaussianity of the independent components (motivated by the Central Limit Theorem). The Central Limit Theorem states that given n independent random variables, the probability distribution of the sum of these random variables approaches a normal distribution under certain general conditions as $n \to \infty$. Note that these random variables need not be identically distributed. To summarize how the Central Limit Theorem is used, first let

$$\mathbf{X} = \mathbf{A}\mathbf{S}, \tag{15}$$

where $\mathbf{X}$, $\mathbf{A}$, and $\mathbf{S}$ are matrices containing the observed data ($\{\mathbf{x}_i(\theta, \phi)\}_{i=1}^{n}$), the weights ($a_{ki}$) (which dictate how the independent components are mixed to produce the observed data), and the independent components ($\{\mathbf{v}_i(\theta, \phi)\}_{i=1}^{m}$), respectively. $\mathbf{A}$ is also referred to as the "mixing" matrix. Further, a linear combination of the observed data yields

$$\mathbf{y} = \mathbf{w}^T\mathbf{X} = \mathbf{w}^T\mathbf{A}\mathbf{S} = \mathbf{b}^T\mathbf{S}. \tag{16}$$

Therefore, $\mathbf{y}$ is a linear combination of the independent components $\mathbf{S}$. According to the Central Limit Theorem, since the sum of as few as two independent random variables is more Gaussian than one independent random variable, it follows that $\mathbf{y}$ is least Gaussian when it equals one of the independent components. Therefore, finding the vector $\mathbf{w}$ that maximizes the non-Gaussianity of $\mathbf{y}$ should yield a $\mathbf{b}$ that has only one nonzero component, thus recovering one of the independent components. Two commonly used quantitative measures of non-Gaussianity are kurtosis and negentropy. There are also other approaches to solving the ICA problem, e.g., minimization of the mutual information between the components or maximum likelihood estimation. More details on these approaches and others can be found in [16].

3 Example

To show a potential clinical application for the methods discussed, this example shows the mapping and decomposition of two human right ventricle endocardial surfaces (RVES) throughout the cardiac cycle. Sets of clinically obtained cardiac ECG-gated computed tomography images were taken from a study relating to pathological changes in the right heart due to pulmonary hypertension. Both patients were symptomatic, however one patient had normal hemodynamics while the other was classified as having pulmonary hypertension [29]. Since the data was obtained from humans, there were of course no implanted markers, and only a select number of anatomical locations can be identified from the images making this an ideal example to show the unique capabilities of the shape analysis approach. Each patient had nine time frames captured during the cardiac cycle and all time frames were analyzed, leading to a total of 18 shapes to map and decompose. It should be acknowledged that this dataset is nowhere near sufficient to draw any legitimate conclusions about heart function and its relationship to pathology, but serves only as an academic example to contrast the nature of potential shape analysis results.

The RVES of both patients at each time frame was manually segmented and smoothed using a standard recursive and discrete Gaussian filter within the commercial medical image processing software Simpleware [15] to remove the high-frequency surface oscillations that occur from the segmentation of stacked images.

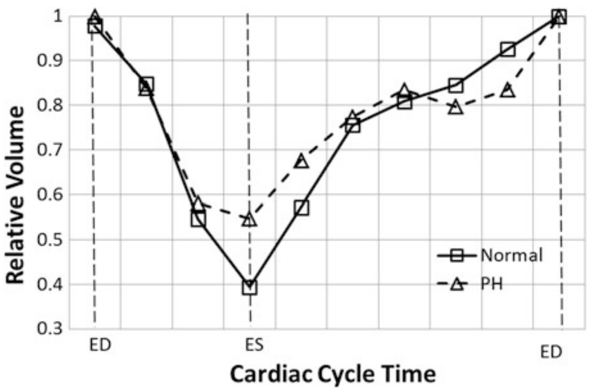


Fig. 6 Relative volume enclosed by the RVES during the cardiac cycle for a patient with normal hemodynamics (*Normal*) and a patient with pulmonary hypertension (*PH*). End diastole is indicated by (*ED*) and end systole is indicated by (*ES*)

For a basis of comparison, the volume enclosed by each RVES was calculated and normalized by the volume at end diastole. Figure 6 shows the relative volume of the two patients' RVES over the cardiac cycle. It should be noted that the total enclosed volume is one of the only few unambiguous metrics that can be directly obtained from the segmented surfaces. As can be seen, the volume for both patients follows a similar path, with volume dropping relatively quickly during ejection, whereas the filling appears to follow two stages: a relatively fast volume increase followed by a more gradual filling. More importantly, while there are clearly similarities and differences between the two patients, these differences are relatively subtle and it is difficult to identify a specific portion of the volume curves that would differentiate the individuals.

To analyze the shapes of the RVES throughout the cardiac cycle, the method of harmonic parameterization described in Sect. 2.1.2 was applied to map each shape to the unit sphere referential domain, and then POD (Sect. 2.2.1) was used to decompose each RVES into its fundamental shape components and the coefficients of those components as the shapes change during the cardiac cycle. In addition, to show the consistency of the shape components found relative to the dataset size, the sets of RVES were mapped and decomposed for each patient individually (i.e., each set of 9) and then combined (i.e., 18 total).

The mapping reference points (i.e., poles and dateline) were chosen as the anterior border between the free wall and the septum with endpoints at the pulmonary valve and the apex. These points were chosen as they can be consistently and accurately identified for each individual at each time frame. Figure 7 shows the RVES shape at end diastole with the pulmonic valve (PV), tricuspid valve (TV), free wall (FW), and septal wall (SW) labeled, as well as the reference points and the distribution of latitude (θ) and longitude (ϕ) over the surface after the initial parameterization. After the segmented surfaces were topologically mapped, the difference between each shape function and the corresponding patient's end systole shape function was calculated, and these functions (i.e., the change in shape or pseudo-displacement fields) were used as the snapshot functions for decomposition. In other words, the translation shape ($\bar{\mathbf{x}}(\theta, \phi)$ in (8)) was taken to be this patient's end systole shape. By converting the data to relative change in shape, there is no need to

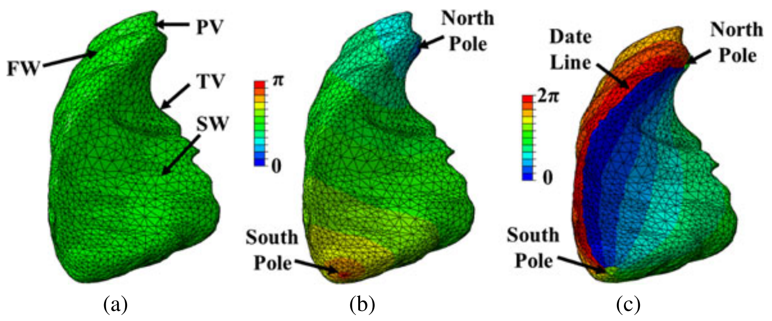
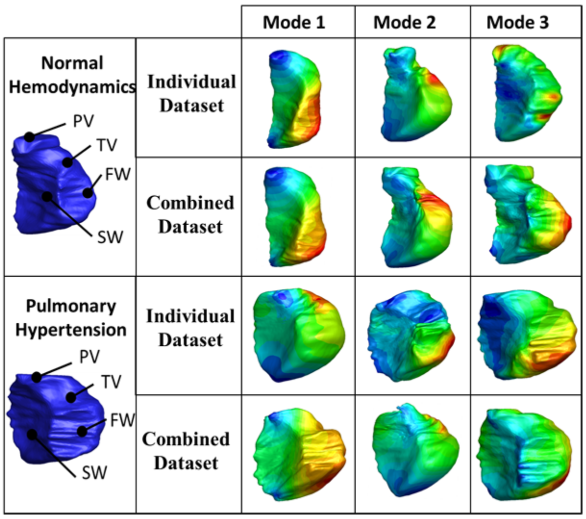


Fig. 7 Example of the RVES shape at end diastole with (**a**) the anatomical features labeled, (**b**) the latitude distribution, and (**c**) the longitude distribution after the initial harmonic parameterization

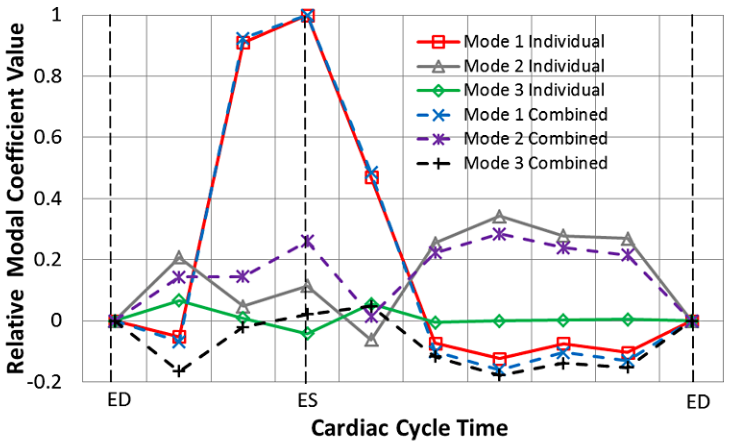
Fig. 8 First three POD modes for the two RVES plotted with respect to the end systole shape with the patient datasets analyzed both individually and combined



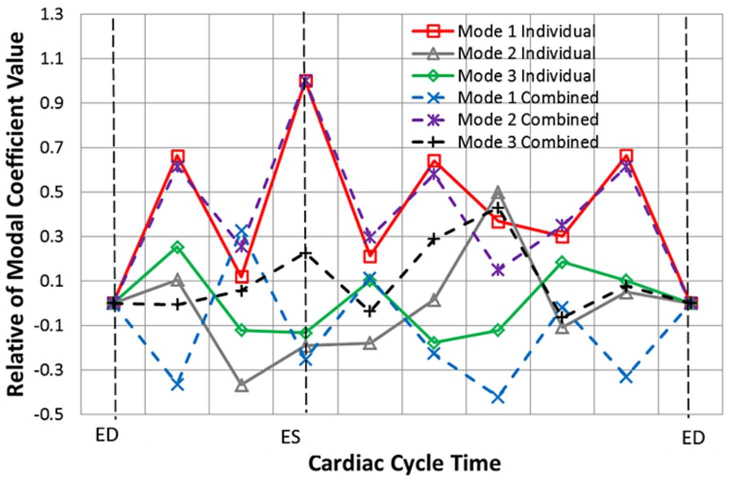
register the shapes for further analysis, preventing a potentially time consuming and uncertain additional step.

Figure 8 shows the first three modes (i.e., fundamental components of the shape change) ranked from highest energy in the set to lowest graphically that were obtained by the decomposition of each patient individually and then combined. These three modes captured more than 99% of the total energy of the datasets, and for display purposes the modes were plotted individually for each patient using their respective end systole coefficients. Similarly, Fig. 9 shows the coefficients for each set of modes over the cardiac cycle for the two patients.

It can be observed that the first mode for the patient with normal hemodynamics is highly consistent, as can be seen from both Figs. 8 and 9, regardless of whether the analysis included only that patient or both patients combined. The second mode of the normal patient also remains consistent (to a slightly lesser extent than the first mode) in appearance and with respect to the coefficients over the cardiac cycle. Of



(a)



(b)

Fig. 9 Relative value of the first three POD modal coefficients defining the shape change of the RVES during the cardiac cycle with the patient datasets analyzed both individually and combined for (**a**) a patient with normal hemodynamics and (**b**) a patient with pulmonary hypertension. End diastole is indicated by (*ED*) and end systole is indicated by (*ES*)

particular interest is that the first mode for the patient with pulmonary hypertension (PH patient) when analyzed individually corresponds to the second mode for that patient from the combined analysis, and therefore corresponds to the second mode for the normal patient. What is significant is that this finding of the consistency of the modes, particularly the second mode from the combined analysis which occurs in both patients when analyzed individually as well, implies that the analysis was able to uncover an intrinsic physiological feature from the right ventricle, not just a statistic (e.g., average shape) of the dataset that would change depending on the amount and composition of the data. In addition, the modal coefficients over the cardiac cycle can be seen to be highly unique for each individual. Whereas the volume change over the cardiac cycle (Fig. 6) was nearly indistinguishable between patients, the shape change (especially with respect to the second mode) is completely distinct between individuals even for though the features are shared. As such, although the number of patients is far too small to draw significant conclusions, especially regarding the pathology, these results would imply that the shape analysis method is capable of deriving an intrinsic biometric relating to the right ventricle, that in the very least uniquely identifies individuals and could potentially distinguish pathological stages.

While it is never possible to say exactly how much data would be necessary to obtain the most significant features that unequivocally describe the desired relationships in a given dataset, the obvious next step to this analysis process of the human RVES would be to expand the dataset by nearly an order of magnitude at least. Given a sufficiently large dataset, the decomposition results could be used to identify the most consistently shared modes throughout patients, regardless of pathology, and each patient would be described by their associated coefficients for these shared modes. Then, supervised or unsupervised data clustering (e.g., k-means or hierarchical clustering [9]) could be used to identify any relevant groupings of the patients as they relate to the pathology and/or any other relevant patient statistics. Lastly, the modal descriptions of the patients and their associated groupings could be applied to build classifiers (e.g., linear discriminant analysis or support vector machines [9]) that could be used for future diagnostic purposes, to determine for instance the pathological classification of a new patient to estimate prognosis and/or the effects of treatment.

4 Future Directions

The techniques presented in this chapter have all shown promise for use in the statistical analysis of the shape of biological structures. However, above and beyond the obvious need for clinical validation, further advancements are still required before these methods will be sufficient for clinical practice. For instance, there is substantial need to enhance the automation of the entire process, particularly with regard to the initial steps of segmenting the desired biological structure through to parameterizing each shape. It is imperative to clinical use that the procedure from the point of imaging the patient to obtaining the diagnostic information is at least semi-automated to a point where a reasonably trained technician could complete the process consistently accurately and efficiently. In addition, future work will likely seek to employ developments in the state-of-the-art in medical imaging, such as tagged MRI [13], that will improve the number of fiducial points that can be acquired and therefore the accuracy of the parameterizations. Similarly, for applications to deformable structures such as the heart, these methods could also incorporate mechanical modeling (e.g., fluid-structure interaction [30]) to better predict the "expected" shape change in time, thereby further improving the consistency and adding important physical interpretation to the shape analysis.

References

1. Aquino W (2007) An object-oriented framework for reduced-order models using proper orthogonal decomposition (POD). Comput Methods Appl Mech Eng 196(41–44):4375–4390. http://linkinghub.elsevier.com/retrieve/pii/S0045782507002058
2. Aquino W, Brigham JC, Earls CJ, Sukumar N (2009) Generalized finite element method using proper orthogonal decomposition. Int J Numer Methods Eng 79(7):887–906. http://doi.wiley.com/10.1002/nme.2604
3. Bansal R, Staib LH, Xu D, Zhu H, Peterson BS (2007) Statistical analyses of brain surfaces using Gaussian random fields on 2-D manifolds. IEEE Trans Med Imaging 26(1):46–57. http://www.ncbi.nlm.nih.gov/pubmed/17243583
4. Barker JL, Garden AS, Ang KK, O'Daniel JC, Wang H, Court LE, Morrison WH, Rosenthal DI, Chao KSC, Tucker SL et al (2004) Quantification of volumetric and geometric changes occurring during fractionated radiotherapy for head-and-neck cancer using an integrated ct/linear accelerator system. Int J Radiat Oncol Biol Phys 59(4):960–970. http://www.ncbi.nlm.nih.gov/pubmed/15234029
5. Blum H (1967) A transformation for extracting new descriptors of shape. In: Models for the perception of speech and visual form, pp 362–380. http://pageperso.lif.univ-mrs.fr/~edouard.thiel/rech/1967-blum.pdf
6. Brechbuhler C, Gerig G, Kobler O (1995) Parameterization of closed surfaces for 3D shape description. Comput Vis Image Underst 61:154–170
7. Brigham JC, Aquino W (2009) Inverse viscoelastic material characterization using POD reduced-order modeling in acoustic-structure interaction. Comput Methods Appl Mech Eng 198(9–12):893–903. http://dx.doi.org/10.1016/j.cma.2008.10.018
8. Chin KM, Kim NHS, Rubin LJ (2005) The right ventricle in pulmonary hypertension. Coron Artery Dis 16(1):13–18. http://www.ncbi.nlm.nih.gov/pubmed/15654194
9. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York
10. Floater MS, Hormann K (2005) Surface parameterization: a tutorial and survey. In: Advances in multiresolution for geometric modelling, pp 157–186
11. Gerig G, Styner M, Jones D, Weinberger D, Lieberman J (2001) Shape analysis of brain ventricles using spharm. In: Proceedings IEEE workshop on mathematical methods in biomedical image analysis MMBIA 2001, pp 171–178. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=991731
12. Gotsman C, Gu X, Sheffer A (2003) Fundamentals of spherical parameterization for 3D meshes. ACM Trans Graph 22(3):358. http://portal.acm.org/citation.cfm?doid=882262.882276
13. Haber I, Metaxas DN, Axel L (2000) Three-dimensional motion reconstruction and analysis of the right ventricle using tagged MRI. Med Image Anal 4(4):335–355. http://www.ncbi.nlm.nih.gov/pubmed/11154021
14. Heimann T, Meinzer H-P (2009) Statistical shape models for 3D medical image segmentation: a review. Med Image Anal 13(4):543–563. http://www.ncbi.nlm.nih.gov/pubmed/19525140
15. http://www.simpleware.com/
16. Hyvärinen A, Oja E (2000) Independent component analysis: algorithms and applications. Neural Netw 13(4–5):411–430
17. Jagroop I, Clatworthy I, Lewin J, Mikhailidis D (2000) Shape change in human platelets: measurement with a channelyzer and visualisation by electron microscopy. Platelets 11(1):28–32. http://discovery.ucl.ac.uk/1312114/
18. Kim SH, Lee J-M, Kim H-P, Jang DP, Shin Y-W, Ha TH, Kim J-J, Kim IY, Kwon JS, Kim SI (2005) Asymmetry analysis of deformable hippocampal model using the principal component in schizophrenia. Hum Brain Mapp 25(4):361–369. http://www.ncbi.nlm.nih.gov/pubmed/15852383

19. Lorenz C (2000) Generation of point-based 3D statistical shape models for anatomical objects. Comput Vis Image Underst 77(2):175–191. http://linkinghub.elsevier.com/retrieve/pii/S1077314299908147
20. Lorenz CH, Walker ES, Morgan VL, Klein SS, Graham TP (1999) Normal human right and left ventricular mass, systolic function, and gender differences by cine magnetic resonance imaging. J Cardiovasc Magn Reson 1(1):7–21
21. O'Dell WG, Moore CC, Hunter WC, Zerhouni EA, McVeigh ER (1995) Three-dimensional myocardial deformations: calculation with displacement field fitting to tagged MR images. Radiology 195(3):829–835. http://radiology.rsna.org/content/195/3/829.abstract
22. Ogniewicz R (1995) Hierarchic Voronoi skeletons. Pattern Recognit 28(3):343–359. http://linkinghub.elsevier.com/retrieve/pii/003132039400105U
23. Planitz B, Maeder A, Williams J (2005) The correspondence framework for 3D surface matching algorithms. Comput Vis Image Underst 97(3):347–383. http://linkinghub.elsevier.com/retrieve/pii/S1077314204001195
24. Quicken M, Brechbuhler C, Hug J, Blattmann H, Szekely G (2000) Parameterization of closed surfaces for parametric surface description. In: Proceedings IEEE conference on computer vision and pattern recognition CVPR 2000, vol 1, pp 354–360. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=855840
25. Riggs BL, Melton LJ III, Robb RA, Camp JJ, Atkinson EJ, Peterson JM, Rouleau PA, McCollough CH, Bouxsein ML, Khosla S (2004) Population-based study of age and sex differences in bone volumetric density, size, geometry, and structure at different skeletal sites. J Bone Miner Res 19(12):1945–1954. http://www.ncbi.nlm.nih.gov/pubmed/15537436
26. Sacks MS, Chuong CJ, Templeton GH, Peshock R (1993) In vivo 3-D reconstruction and geometric characterization of the right ventricular free wall. Ann Biomed Eng 21(3):263–275. http://www.ncbi.nlm.nih.gov/pubmed/8328726
27. Siddiqi K, Pizer SM (2008) Medial representations: mathematics, algorithms and applications. Springer, Berlin
28. Simon MA, Deible C, Mathier MA, Lacomis J, Goitein O, Shroff SG, Pinsky MR (2009) Phenotyping the right ventricle in patients with pulmonary hypertension. Clin Transl Sci 2(4):294–299
29. Simonneau G, Robbins IM, Beghetti M, Channick RN, Delcroix M, Denton CP, Elliott CG, Gaine SP, Gladwin MT, Jing Z-C et al (2009) Updated clinical classification of pulmonary hypertension. J Am Coll Cardiol 54(1):S43–S54. http://www.ncbi.nlm.nih.gov/pubmed/19555858
30. Tang D, Yang C, Geva T, Del Nido PJ (2008) Patient-specific MRI-based 3D FSI rv/lv/patch models for pulmonary valve replacement surgery and patch optimization. J Biomech Eng 130(4):041010
31. Telea A, Wijk JJV (2002) An augmented fast marching method for computing skeletons and centerlines. In: Proceedings of the symposium on data visualisation, pp 251–259. http://portal.acm.org/citation.cfm?id=509782
32. Thompson PM, Hayashi KM, de Zubicaray GI, Janke AL, Rose SE, Semple J, Hong MS, Herman DH, Gravano D, Doddrell DM, Toga AW (2004) Mapping hippocampal and ventricular change in Alzheimer disease. NeuroImage 22(4):1754–1766. http://www.sciencedirect.com/science/article/pii/S105381190400196X
33. Vuille C, Weyman AE (1994) Left ventricle I: general considerations, assessment of chamber size and function. In: Principle and practice of echocardiography, 2nd edn. Lea and Febiger, Philadelphia, pp 575–624
34. Yushkevich PA, Zhang H, Gee JC (2006) Continuous medial representation for anatomical structures. IEEE Trans Med Imaging 25(12):1547–1564
35. Zayer R, Rossl C, Seidel HP (2006) Curvilinear spherical parameterization. In: IEEE international conference on shape modeling and applications 2006 SMI06, pp 11. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1631195

Finite Element Modeling of Biomolecular Systems in Ionic Solution

Benzhuo Lu

Abstract An accurate finite element method is introduced to solve the two most commonly used continuum models in computational biophysics: Poisson–Boltzmann (PB) equation and Poisson–Nernst–Planck (PNP) equations. They describe equilibrium and non-equilibrium (with diffusion existed) properties of ionic liquid, respectively. Both models involve two domains (solvent and solute) with distributed singular permanent charges inside biomolecules (solute domain) and a dielectric jump at the interface between solvent and solute. A stable regularization scheme is described to remove the singular component of the electrostatic potential induced by the permanent charges inside biomolecules, and regular, well-posed PB/PNP equations are formulated. The interface conditions for electric potential are also explicitly enforced to be satisfied. An inexact-Newton method is used to solve the nonlinear elliptic PB equation and the coupled steady-state PNP equations; while an Adams–Bashforth–Crank–Nicolson method is devised for time integration for the unsteady electrodiffusion. The numerical methods are shown to be accurate and stable by various tests of real biomolecular electrostatic and diffusion problems.

1 Introduction

All biomolecules in cell are solvated in ionic solution which supplies an essential environment to molecular activities. These activities are generally involved in multi-scale processes. Explicit molecular dynamics (MD) or Monte Carlo (MC) simulations that includes all the solute and solvent particles are known to be limited in size and time scales of simulated systems. To overcome the shortage, implicit simulation approaches were developed to significantly reduce the degree of freedom of the system by treating the solvent as a continuum medium. The continuum models focus on the average properties of solvent through a solution of partial differential equation(s), and is therefore computationally more efficient. Furthermore, continuum

B. Lu (✉)
State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
e-mail: bzlu@lsec.cc.ac.cn

model can conveniently include different types of physical interactions/processes and bridge different temporal-spatial scales, e.g. by coupling electrostatics with diffusion convection, or/and elasticity, the Navier–Stokes equations and so on. These features have made the continuum model very appealing and useful. The Poisson–Boltzmann equation (PBE) and the Poisson–Nernst–Planck equations (PNPEs) are the two most studied and established continuum models in computational molecular biology. The former is usually used for equilibrium simulation of molecular electrostatic solvation effects, and the later for non-equilibrium simulation of ionic diffusion processes interacting with biomolecular systems.

Efficiency and accuracy are two central issues in applying the PBE/PNPEs to biophysical modeling. First, a typical macromolecule may consist of a few to hundreds of thousand atoms (point charges in the PBE), which significantly challenges the current computer memory and speed. Secondly, in order to incorporate the PB electrostatics (on the fly) in a typical MD, MC, or Brownian dynamics (BD) simulation that could involve tens of millions of steps to get converged statistical results, a single solution of the PBE has to be completed within no more than a few tenths of a second on a modern workstation to meet the total wall-clock time constraint. Based on this estimation, the current solvers are still, e.g., about one to two orders of magnitude slower [56]. Thirdly, a similar demand of efficiency lies in virtual high throughput screening in drug discovery from many candidate structures and different conformations. This screening is usually based on free energy calculations (e.g., binding affinity) to an accuracy of a few kcal/mol. However, these free energies normally result from the cancellation of energies of several orders of magnitude larger terms such as electrostatic energies. This demand poses another numerical challenge for electrostatic computations with the PBE.

Finite element method (FEM) is an efficient and powerful numerical method for solution of nonlinear elliptic equation(s). Adaptive mesh refinement is a mature strategy developed in FEM to control the accuracy and efficiency of the solution. While a complicated situation in both PB and PNP models is that the solvated biomolecular systems are usually modeled by dielectrically distinct regions with singular charges distributed in the molecular region. Specific strategies are needed in FEM framework to accurately treat the singular charges and the dielectric jump at molecular boundary.

In this chapter, the two models and related methodologies will be briefly reviewed. We apply a stable regularization scheme to remove the singular component of the electrostatic potential induced by the permanent charges inside biomolecules, and formulate a regular, well-posed PB equation. The interface conditions can be explicitly enforced in the solution through using boundary conforming meshes in the FEM simulations. Then, a corresponding FEM algorithm is given. Similar regularization scheme and interface condition treatment are applied to PNP system. An inexact-Newton method is used to solve the nonlinear elliptic PBE or the coupled PNP equations for steady problems; while an Adams–Bashforth–Crank–Nicolson method is devised for time integration for the unsteady electrodiffusion. The numerical methods are shown to be accurate and stable by various test problems, and are applicable to real large-scale biophysical electrostatics and electrodiffusion problems.

A mesh is required in finite element method. Molecular mesh generation is a very technical and challenging task for practical FEM simulation of biomolecular systems. The topic is briefly discussed in Sect. 4. Interested readers are refereed to chapter “Surface Triangular Mesh and Volume Tetrahedral Mesh Generations for Biomolecular Modeling” on biomolecular meshing of this book.

The rest of the chapter is organized as follows. The PB and PNP models and their FEM treatments are introduced in Sects. 2 and 3, respectively. Each section contains a brief history of the equation(s) and related methodologies, descriptions of the regularization scheme, the numerical strategies and properties for the equation(s). Numerical examples for real biomolecular electrostatics and diffusion problems are given in Sect. 5. The chapter ends with a summary in Sect. 6.

2 PB Model

Poisson–Boltzmann (PB) theory has been a well-established model in a broad range of scientific research areas. In eletrochemistry, it is known as Gouy–Chapman (GC) theory [15, 37]; in solution chemistry, it is known as Debye–Hückel theory [24]; in colloid chemistry, it is known as the Derjaguin–Landau–Verwey–Overbeek (DLVO) theory [25, 79]; and in biophysics, it is known as PB theory [23, 47]. The Poisson–Boltzmann equation (PBE) represents a typical implicit solvent model, and provides a simplified continuum description of the discrete particle (e.g., water, ion, and/or protein molecule) distributions in solution. In particular, the PBE describes the electrostatic interaction and ionic density distributions of a solvated system at the equilibrium state. Since the first application of the PBE in a biomolecular system [80], a large amount of literatures and many solution techniques have been produced in this area and directed to studies of diverse biological processes.

A number of review papers can be found that focus on the physical fundamentals [66, 71], brief history [30], the methodology and applications in biomolecular modeling [3, 6], the methodological developments in both PB and the related generalized Born models [51]. A more recent review [58] focused on the numerical aspects of PB methodology covering several major numerical methods. This chapter will present detailed techniques in use of finite element approach.

Solvated biomolecular systems are usually modeled by dielectrically distinct regions with singular charges distributed in the molecular region. Systems without singular charges or dielectric discontinuities are usually found in simplified models with planar or cylindrical boundary geometries in electrochemistry and biopolymer science, and can be regarded as a special case of the systems in this investigation. Figure 1 schematically illustrates a solvated biomolecular system occupying a domain Ω with a smooth boundary $\partial\Omega$. The solute (molecule) region is represented by Ω_m and the solvent region by Ω_s. The dielectric interface Γ is defined by the molecular surface, which can be defined as the solvent-excluded surface, solvent-accessible surface, Gaussian surface [81], or some other appropriately de-

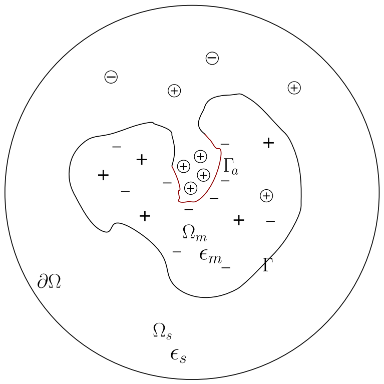


Fig. 1 2-D schematic illustration of the computational domain modeling a solvated biomolecular system. The biomolecular (solute) region is Ω_m with dielectric constant ε_m and the aqueous solution (solvent) is domain Ω_s with dielectric constant ε_s. The molecular surface is $\Gamma = \bar{\Omega}_s \cap \bar{\Omega}_m$. The *circles with plus or minus sign inside* represent the diffusive charged particles which move only in Ω_s. The singular charges inside molecules are signified by *plus* or *minus sign* in Ω_m. The active reaction center $\Gamma_a \subset \Gamma$ is also highlighted in *red* where a different boundary condition may be applied in the PNP model

fined solvent-molecular interface. n is the unit normal vector at Γ, pointing from Ω_m to Ω_s. The nonlinear Poisson–Boltzmann equation in Ω reads

$$-\nabla \cdot (\varepsilon \nabla u) - \lambda \sum_{j=1}^{K} c_j q_j e^{-\beta q_j u} = \sum_{i=1}^{N} q_i \delta(x - x_i), \quad x \in \Omega, \tag{1}$$

where ε is a spatial-dependent dielectric coefficient, the characteristic function $\lambda = 0$ in Ω_m (impenetrable to ions) and 1 in Ω_s, c_j is the bulk density of mobile ion species j with charge q_j, $\beta = 1/kT$, k is the Boltzmann constant, T is the absolute temperature, q_i is the singular charge located at x_i within solute region. For symmetric 1 : 1 salt (the bulk densities of cation and anion need to be equal, $C_+ = C_- = C$, to satisfy the neutrality condition), to simplify the presentation we use

$$-\nabla \cdot (\varepsilon \nabla u) + \lambda \kappa^2 \sinh(u) = \rho^f, \quad x \in \Omega, \tag{2}$$

and

$$-\nabla \cdot (\varepsilon \nabla u) + \lambda \kappa^2 u = \rho^f, \quad x \in \Omega, \tag{3}$$

for the linearized Poisson–Boltzmann equation in case of weak electrostatic potential, where $\kappa^2 = 2\beta e^2 C$ absorbing the related parameters (e is the elementary charge), and $u \to \beta e u$ and $\rho^f \to \beta e \rho^f$ are the scaled electrostatic potential and singular charge distribution, respectively. Note that $\kappa = 0$ in Ω_m because the mobile ions only present in the solvent region Ω_s. An additional region called the Stern layer might be present in some Poisson–Boltzmann models. This Stern layer is part of the solvent but is not penetrable for the mobile ions so $\kappa = 0$ there. The transition from the low-dielectric solute region to the high-dielectric solvent region is usually modeled to be abrupt, which gives rise to a dielectric interface Γ. This interface is usually identified as the molecular surface. There are two conditions on Γ needed to be satisfied from the dielectric theory:

$$u_m = u_s, \qquad \varepsilon_m \frac{\partial u_m}{\partial n} = \varepsilon_s \frac{\partial u_s}{\partial n}, \qquad x \in \Gamma. \tag{4}$$

These conditions are explicitly used in boundary integral equations based approaches, but may not be exactly satisfied in other approaches such as the traditional finite difference methods, or finite element methods without special interface treatment. An approximated Dirichlet boundary condition is normally imposed on $\partial\Omega$. The dielectric permittivity is usually assumed to be a piecewise constant with $\varepsilon = \varepsilon_m \varepsilon_0$ in Ω_m and $\varepsilon = \varepsilon_s \varepsilon_0$ in Ω_s, where ε_0 is the dielectric constant of vacuum. This is indispensable to the regularization schemes to be introduced later. The internal dielectric interface separating the molecules and solvent regions is defined to be the molecular surface, but other definitions of dielectric interface might apply also. Typical values of ε_m and ε_s are 2 and 80, respectively. The singular charge distribution within biomolecules, discontinuous dielectric constant, exponential nonlinearity at strong potential, and the highly irregular molecular surface constitute the most prominent features of the Poisson–Boltzmann equation.

2.1 Regularization Schemes of the Poisson–Boltzmann Equation

The presence of the singular charge distribution in the PBE indicates that its solution is not continuous and does not belong to $H^1(\Omega)$ [16], which directly challenges the solution theory of standard finite difference methods, finite volume methods or finite element methods for the PBE. In many finite difference or finite element solvers of the Poisson–Boltzmann equation, the singular charges are distributed onto the grid points near the singular charges by using polynomial interpolations. These approximations work well for electrostatic solvation energy ΔG_{ele} calculations. The solvation energy is defined as

$$\Delta G_{ele} = G_{sys} - G_{ref}, \tag{5}$$

where G_{sys} is the electrostatic free energy of the biomolecular system in the solvated state and the G_{ref} is the electrostatic free energy of the system assuming it is in space of uniform dielectric constant ε_m and without mobile ions. By using a finite difference method, finite volume method or a finite element method, the PBE is solved twice with corresponding parameters for G_{sys} and G_{ref}, respectively. Linear interpolation or higher order polynomial interpolation are usually used in these numerical methods for approximating the singular charge distribution. Although the potentials from these two calculations might suffer from large error near the singular charges, it is believed that this error would cancel in computing the ΔG_{ele} via Eq. (5) if the same mesh and charge interpolation are used in these two solutions of the PBE. This treatment is widely applied in computational chemistry and is somehow validated by many numerical experiments [28, 29, 35, 36, 61, 74, 86, 89]. However, the quality of the potential near the molecular surface is actually critically dependent on the specific treatment of the singular charges [32]. If the gradient of the electrostatic potential is needed, such as force calculation on atoms in the MD simulation, or electric field calculation at the boundary in the diffusion-reaction simulation by PNP equations studied in next section, more rigorous treatments of the singular charges are needed.

The regularization schemes aim at removing the singular component of the potential from the equation such that the remaining component has higher regularity and thus is solvable by using general numerical methods. The straightforward decomposition [36] considers the singular Coulomb potential u^s of all singular charges

$$-\varepsilon_m \Delta u^s = \rho^f \quad \text{in } \Omega \tag{6}$$

The corresponding regular potential component u^r is then found by subtracting Eq. (6) from Eq. (2) to be

$$-\nabla \cdot \big(\varepsilon \nabla u^r\big) + \lambda \kappa^2 \sinh\big(u^r + u^s\big) = 0 \quad \text{in } \Omega, \tag{7}$$

The singular component ϕ^s should also be subtracted from the interface conditions (4), generating the following interface conditions for Eq. (8):

$$u_s^r - u_m^r = 0, \qquad \varepsilon_s \frac{\partial u_s^r}{\partial n} - \varepsilon_m \frac{\partial u_m^r}{\partial n} = (\varepsilon_m - \varepsilon_s) \frac{\partial u^s}{\partial n}, \quad x \in \Gamma. \tag{8}$$

This approach is applied to solve the Poisson–Boltzmann equation by Zhou et al. [89] to completely remove the self-energy so that the equation need not to be solved twice for computing the electrostatic energy. Another slightly different decomposition but leading to quite different numerical strategies using a similar equation as (6) but with varying dielectric were proposed in a hybrid finite difference/boundary element method [12] and a hybrid finite element/boundary element method [57, 87] for solving the nonlinear PBE. These two methodologies take the advantage of boundary element method to conveniently handle the singular point charges and also leads to stable and accurate numerical solution. The removal of the singular potential makes it possible for the first time to analyze the Poisson–Boltzmann equation rigorously in Sobolev spaces [16]. However, it is found that the first scheme suffers a numerical instability that will lead to a substantial error in FEM numerical solution of the full potential [45]. This is because that the total potential ϕ is relatively weak while the singular potential ϕ^s and the regular potential are both strong. In particular, the regular potential in Ω_s is larger than the total potential ϕ by $\varepsilon_s/\varepsilon_m \approx 40$ times. Consequently, when the numerical solution of ϕ^h is added to the analytical solution of ϕ^s to get the total potential, the relative numerical error will be amplified by about 40 times. For this reason we will apply a stable decomposition in this FEM study. This decomposition is first introduced by Chern et al. for solving the PBE with an interface method [19], and is implemented later in finite different method [32] and finite element method [58, 59].

We define the singular component u^s to be the restriction on Ω_m of the solution of

$$-\varepsilon_m \Delta \phi^s(x) = \rho^f(x), \quad x \in \mathbb{R}^3, \tag{9}$$

and the harmonic component $u^h(x)$ to be the solution of a Laplace equation:

$$\begin{aligned} -\Delta u^h(x) &= 0, \quad x \in \Omega_m, \\ u^h(x) &= -u^s(x), \quad x \in \Gamma. \end{aligned} \tag{10}$$

It is seen that $u^s(x)$ can be given analytically by the sum of Coulomb potentials. This $u^s(x)$ is then used to compute the boundary condition for $u^h(x)$, the latter is to be solved numerically from Eq. (10), for which we use a finite element method in this study. Subtracting these two components from Eq. (2) we get the governing equation for the regular component $u^r(x)$:

$$-\nabla \cdot \big(\varepsilon \nabla u^r(x)\big) + \lambda \kappa^2 \sinh\big(u^r(x)\big) = 0, \quad x \in \Omega, \tag{11}$$

and the interface conditions

$$u_s^r - u_m^r = 0, \qquad \varepsilon_s \frac{\partial u_s^r}{\partial n} - \varepsilon_m \frac{\partial u_m^r}{\partial n} = \varepsilon_m \frac{\partial (u^s + u^h)}{\partial n}, \quad x \in \Gamma. \tag{12}$$

It is worth noting that there is no decomposition of the potential in the solvent region, thus $\phi(x) = \phi^r(x)$ in Ω_s. There is no decomposition in Ω_s in the second scheme, and thus the numerical solution of ϕ^r in Ω_s does not suffer the instability [45].

2.2 *Finite Element Methods*

The adaptive finite element method developed by Holst et al. in [4, 16, 41, 44] tackled some of the numerical issues of the Poisson–Boltzmann equation. This method uses the piecewise-linear finite element and a well-defined error indicator for driving the local mesh refinement [16, 41]. The nonlinear Poisson–Boltzmann equation is solved using Newton-AMG iterations [42, 43, 46]. After discretization by either finite difference or finite element techniques, the inexact Newton-AMG approach results in linear memory and computational complexity solution of the nonlinear algebraic equations produced by finite difference, finite volume, or finite element discretization methods. In the case of adaptivity, non-standard variations of multigrid solvers must be used to preserve both linear memory and linear computational complexity; see [2, 45] for a detailed discussion.

Instead of using the Newton-AMG iterations for the nonlinear PBE, the finite element method of Shestakov et. al [75] uses Newton–Krylov iterations for the nonlinearity. The applications of this finite element method have not been extended from colloid systems with rather simple geometry to biomolecular systems with complicated dielectric interfaces. A mortar finite element discretization was also introduced recently by Xie et al. for numerical solutions of the PBE, which explicitly computed dielectric interface so that the interface conditions are satisfied naturally [82].

Though the new regularization scheme [19] and inclusion of molecular surface have been practically used for PB solution for real biomolecule [32, 58, 59], the analysis of a convergent adaptive finite element method was only made recently [45]. With this scheme, the accuracy of the potential near the molecular surface is substantially improved, becoming comparable to that of the interface Poisson–Boltzmann solvers [45, 86]. The finite element method advanced by Cortis et al. [21] makes use of the similar Galerkin formulation but lack a treatment of the nonlinear Poisson–Boltzmann equation. Moreover, there is no enforcement of the interface conditions on the molecular surface so the results of this method agree well with those of DelPhi. A recently proposed discontinuous Galerkin method for elliptic interface problems [38] might also be customized for solving the Poisson–Boltzmann equation provided a good description of the molecular surface.

Now we describe the FEM computational algorithm with the new regularization scheme for 3D molecular simulations. To consider the finite element solution of the PBE (11) (Eq. (10) is a simpler and special case), we define the solution space

$$H := \left\{u \in H_0^1(\Omega)\right\} \tag{13}$$

and its finite dimensional subspace

$$S := \left\{u \in P_1(\Omega)\right\}, \tag{14}$$

where P_1 is the space consisting of piecewise linear tetrahedral finite elements. Functions in the space

$$H_0^1 = \left\{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\right\},$$

satisfy the Dirichlet boundary condition on the exterior boundary $\partial\Omega$. We assume that the finite elements are regular and quasi-uniform. The weak formulation of the problem now is:

Find $u = \in S$ such that

$$\left\langle F(u), v\right\rangle = 0 \quad \forall v \in S. \tag{15}$$

Here the nonlinear mapping $F : H \mapsto H^*$ and $\langle\cdot,\cdot\rangle$ is the standard duality paring between the dual space H^* and H. Specifically, the nonlinear weak form $\langle F(u), v\rangle$ is defined to be

$$\left\langle F(u), v\right\rangle = (\varepsilon\nabla u, \nabla v) + \left(\lambda\kappa^2 \sinh u, v\right) + \langle p, v\rangle_\Gamma, \tag{16}$$

where

$$p = \varepsilon_m \frac{\partial(u^s + u^h)}{\partial n}$$

is the jump in electric displacement defined in Eq. (12), $\langle\cdot,\cdot\rangle_\Gamma$ denotes the L_2 inner product defined on the interface Γ, and the L_2 scalar inner product over the domain Ω is denoted by $(\cdot,\cdot)$. It is worth noting that the interface integral $\langle\cdot,\cdot\rangle_\Gamma$ is conveniently and directly evaluated in FEM by using a boundary conforming mesh (Γ is a collection of some faces of the tetrahedral mesh). This type of meshes, as generated by TMSmesh [17] are used in all of our FEM simulations. To solve the nonlinear problem (15) we employ the damped inexact-Newton method [41] which necessitates the Gâteaux derivative $DF(u)$ defined by the bilinear form

$$\begin{aligned}\big\langle DF(u)w, v\big\rangle &= \frac{d}{d\tau}\big\langle F(u+\tau w), v\big\rangle\Big|_{\tau=0} \\ &= (\varepsilon\nabla w, \nabla v) + \big(\lambda\kappa^2 w\cosh u, v\big)\end{aligned} \tag{17}$$

With these well-defined operators the complete algorithm can be given as follows:

Algorithm 1

```
• Choose the initial approximation $u$, the nonlinear tolerance $\varepsilon$, the residual $r$ in
  approximately solving the linear system, and the damping factor $c$.
• Do until $|\langle F(u), v\rangle| < \varepsilon$
    1. Solve the correction $w$ from $\langle DF(u)w, v\rangle = -\langle F(u), v\rangle + r$.
    2. $u \Leftarrow u + cw$.
```

A constant damping parameter $c = 1$ is chosen in this study. We note here that the step in the algorithm to solve the correction w leads to a linear system to be solved. Denoting the solution $u(x)$ by its expansion in the test function space, i.e., $u(x) = \sum_j a_j v_j(x)$, the weak form (16) essentially produces two matrices: a stiff matrix A associated with the product $\varepsilon\nabla u \cdot \nabla v$ and the mass matrix M associated with the product $\lambda\kappa^2 w(\cosh u)v$. The solution of $w(x)$ (correction of $u(x)$ at each Newton iteration step) from the bilinear form (17) is therefore equivalent to the solution of a linear algebraic system

$$(A + M)\mathbf{a} = -\mathbf{f}, \tag{18}$$

where unknown vector $\mathbf{a} = \{a_j\}$ is the expansion coefficients of $w(x)$, and vector $\mathbf{f}$ is $\langle F(u), v\rangle$ for all given test functions v. The system of equations implied by Eq. (16) and the linearization Eq. (17) are then solved by a FEM software package like FETK [40] or PHG [85].

3 PNP Model

Under non-equilibrium condition(s), net ionic fluxes are produced in solution, to which case the PB model does not apply. The diffusive fluxes and the relevant electrostatic interactions in ionic solution are described as electrodiffusion. When small charged molecules are approximated as diffusive ions, the electrodiffusion framework can also be adopted to study their transportation and/or diffusion-reaction processes. Electrodiffusion is a rate-limiting step in numerous biological processes, such as ligand-enzyme binding, protein-protein diffusive encounter. An example is neurotransmission within synapses between adjacent nerve cells [9]. The kinetic properties of these processes are mostly governed by the multi-scale electrodiffusion of charged molecules in aqueous solution with various ionic concentrations, molecular charges and complicated solvent-solute interface geometries. The continuum model is more straightforward and efficient to determine the kinetics than discrete particle simulations. Furthermore, continuum electrodiffusion models can be readily modified to incorporate other types of physical interactions, such as varying molecular conformation or flow convection, by coupling with elasticity equation or the Navier–Stokes equations. These appealing features have made the continuum electrodiffusion models very useful not only for the quantitative analysis of the biological ion channels [27, 33], substrate-enzyme diffusion-reactions [54, 76, 77], and cellular electrophysiology [62, 63], but also for investigating ion-separation membranes in non-biological applications [72] and the transport of electrons and holes in semiconductors [49].

The Poisson–Nernst–Planck equations are commonly used to describe the electrodiffusion of mobile ions and charged substrates, all modeled as diffusive particles with vanishing size, in solvated biomolecular systems. Here the electrostatic potential is induced by the mobile ions, charged substrates, and the fixed charges carried by biomolecules. The system setup is similar to the PB case (see Fig. 1). The diffusive particles (ions and substrates) are distributed in Ω_s. Charged substrates might react with the biomolecules on a part of the molecular surface Γ_a, for which a suitable boundary condition for the diffusion equations of the particles is needed. On the non-reactive molecular surface $\Gamma \setminus \Gamma_a$ appropriate boundary condition is needed to model the vanishing macroscopic flux. In a typical solvated biomolecular system there are multiple species of ions and substrates; each species may have its own boundary condition on molecular surface. We assume that the exterior boundary $\partial\Omega$ is connected to a particle reservoir maintained at constant concentrations, and hence a Dirichlet boundary condition for particle concentration can be applied. Compared to the pure diffusion [78], or the Nernst–Planck equation (also called Smoluchowski equation) [77] which characterizes diffusional drift by a given fixed potential, the Poisson–Nernst–Planck model is able to generate a self-consistent, full electrostatic potential and the non-equilibrium densities of ions/substrates [57, 87]. Similar to PBE, the PNP equations for describing the electrodiffusion around the biomolecules modeled at atomistic level also have the two features: presence of singular permanent charges and highly irregular surfaces not penetrable to diffusive particles.

Mathematical analysis of the Poisson–Nernst–Planck equations have been developed long after the introduction of the equation by Nernst and Planck [65, 67]. The existence and stability for the solutions of the steady PNP equations are established by Jerome [48] in studying the steady Van Roostbroeck model for electron flows in semiconductors, via a delicate construction of a Schauder fixed point mapping. Although this mapping is not shown to be contractive, an alternative pseudo-monotone mapping is constructed which guarantees the convergence of the Galerkin approximations of the equations. It noted that the permanent charges in this study are located in the same domain as that in the diffusion process, and are assumed to be in L^∞ which ensures the $H^1 \cap L^\infty$ regularity of the electrostatic potential and the charge densities. Existence and long time behavior of the unsteady PNP equations were studied in [10]. The analysis and computation of the PNP equations can be further simplified by reducing the 3-D system to 1-D models. Singular perturbation methods and asymptotic analysis can then be applied to study the solution properties of these simplified 1-D equations. For example, 1-D steady PNP equations for modeling physiological channels are investigated in [8, 53] in the absence of permanent charges by using various singular perturbation theories. The effects of the permanent charges are considered in [1, 26], where the permanent charge density is vanishing in the reservoirs at the two ends of the channel and is constant at the center of the channel. The piecewise constant form of the permanent charges implies that the electrostatic potential and ionic densities are still differentiable. The reduction of the dimensionality greatly simplifies the mathematical analysis of the electrodiffusion systems, and the results provide useful guide lines for the analysis of the corresponding fully 3-D systems at some limit cases. As a trade-off they are generally unable to reproduce the diffusion and reaction processes that critically depend on the geometry of the system and complicated boundary conditions.

In contrast to the limited amount of work on the mathematical analysis of the PNP equations for biophysical applications, numerical computations with the PNP and the PNP-like systems have been widely conducted by computational physicists and biophysicists. Finite difference methods are particularly popular due to the simplicity in their implementation, and have been applied to a large extent to 1-D or 3-D ion conduction characteristics of biological ion channels or other transmembrane pores [11, 14, 20, 27, 52]. The lattice nature of the finite difference method makes it difficult to model the highly irregular surface of the ion channel or the active sites of the enzymes. This difficulty can be readily overcome by using finite element methods, which have been well developed for simulating semiconductor devices [31, 50] and were recently introduced to simulate the electrodiffusion with realistic molecular structures [76, 77]. In many of the PNP solvers developed thus far such as [11, 14, 52] the electrostatic part is solved by using well-established finite difference or finite element Poisson–Boltzmann solvers [5, 13, 34]. These PB solvers use polynomial interpolations to approximate the singular charges. As described in last section, the treatments do not supply an electric field of high fidelity at molecular boundary to the Nernst–Planck equation. A similar decomposition scheme to that used in the PB equation will be adopted for the PNP equations.

The objective of this section is to present the regularized PNP equations with singular permanent charges, and to develop finite element methods for them with realistic biomolecular structures. A symmetric transformation of PNP will be mentioned. We will show that the electrostatic potential that couples the Nernst–Planck equation is indeed the regular component. Therefore the framework established in [48] for general L_2 permanent charges could be utilized to show the well-posedness of the regularized PNP system. An inexact-Newton method will be used to solve the nonlinear differential equations. Since the Poisson–Nernst–Planck equations can be derived from the first variations of a free energy functional, the Newton-like methods can produce a convergent solution that corresponds to the minimization of the free energy.

3.1 PNP Equations

The continuum PNP equations can be derived via different routes. They can be obtained from the microscopic model of Langevin trajectories in the limit of large damping and absence of correlations of different ionic trajectories [64, 73], or from the variations of the free energy functional that includes the electrostatic free energy and the ideal component of the chemical potential [33]. The former gives the PNP model a sound theoretical basis while the latter provides a flexible framework to include more physical interactions, most prominently the correlations among particles with finite sizes, into the continuum model. In this chapter, we are concentrated in the development of numerical techniques for the standard nonlinear PNP equations, i.e., we treat all diffusive particles, including mobile ions and charged substrates, as particles with vanishing size. This is a reasonable assumption in case that solution is dilute and the characteristic dimension of space for diffusion is much larger than the particle size.

We obtain the PNP equations by coupling the Nernst–Planck equation

$$\frac{\partial \rho_i}{\partial t} = \nabla \cdot D_i(\nabla \rho_i + \beta q_i \rho_i \nabla \phi), \quad x \in \Omega_s, 1 \le i \le n, \tag{19}$$

and the electrostatic Poisson equation with interface $\Gamma = \bar{\Omega}_s \cap \bar{\Omega}_m$:

$$-\nabla \cdot (\varepsilon \nabla \phi) - \lambda \sum_i q_i \rho_i = \rho^f, \quad x \in \Omega, \tag{20}$$

where $\rho_i(x,t)$ is the concentration of the i-th species particles carrying charge q_i, $D_i(x)$ is the spatial-dependent diffusion coefficient, and ϕ is the electrostatic potential. The interface conditions for PE is similar to that for PBE. If the mobile charge density $\rho_i(x)$ in Eq. (20) is assumed to follow the Boltzmann distribution, the equation converts to the nonlinear Poisson–Boltzmann equation. The readers are referred to [58] for discussions on the derivation and relations of these equations. The time-dependence of the electrostatic potential is seen from the appearance of time-dependent particle concentrations in Eq. (20).

Because the singular charge $\rho^f(x)$ poses the same numerical issue to the Poisson equation as to the PBE, a similar potential decomposition as described in the PB model (the second scheme) is adopted here to achieve a stable FEM solution for the Poisson equation. The singular and harmonic components follow the equation and boundary condition as in (9) and (10). The governing equation for the regular component $\phi^r(x)$:

$$-\nabla \cdot \big(\varepsilon \nabla \phi^r(x,t)\big) - \lambda \sum_i q_i \rho_i(x,t) = 0, \quad r \in \Omega, \tag{21}$$

and the interface conditions

$$\phi^r_s - \phi^r_m = 0, \qquad \varepsilon_s \frac{\partial \phi^r_s}{\partial n} - \varepsilon_m \frac{\partial \phi^r_m}{\partial n} = \varepsilon_m \frac{\partial(\phi^s + \phi^h)}{\partial n}, \quad x \in \Gamma. \tag{22}$$

It is worth noting that there is no decomposition of the potential in the solvent region, thus $\phi(x) = \phi^r(x)$ in Ω_s. Hence the final regularized Poisson–Nernst–Planck equations consist of the regularized Poisson equation (21) and

$$\frac{\partial \rho_i(x)}{\partial t} = \nabla \cdot D_i(x)\big(\nabla \rho_i(x) + \beta q_i \rho_i(x) \nabla \phi^r(x)\big), \quad x \in \Omega_s. \tag{23}$$

To simplify the presentation we use ϕ to denote the electrostatic potential coupled with the Nernst–Planck equation, but keep in mind that the singular and harmonic components are to be added to get the full potential inside molecules.

The singular and harmonic components only need to be solved one time a priori the coupled solutions of the regularized PNP equations. Indeed, it is the regular potential in solvent region that couples the Nernst–Planck equation and the regular Poisson equation. The singular and harmonic components serve only for providing a fixed interface conditions for solving the regular component, which varies with the ionic concentrations.

We apply the following boundary conditions for the PNP equations. The approximate Debye law is used to compute the value of $\phi^r = \phi$ on the exterior boundary $\partial\Omega$:

$$\phi(x) = \sum_j \frac{q_j e^{-|x-x_j|/\lambda_d}}{\varepsilon_s |x - x_j|},$$

where λ_d being the Debye length computed from the bulk concentrations of all species of charged particles. For all species of particles ρ_i on $\partial\Omega$ is given by its bulk concentration. A zero macroscopic normal flux

$$D_i(\nabla \rho_i + \beta q_i \rho_i \nabla \phi) \cdot n = 0$$

is prescribed on the non-reactive molecular surface $\Gamma \setminus \Gamma_a$ with outer normal vector n for all species. For particles that react with the molecule on the surface Γ_a we apply the homogeneous Dirichlet boundary condition, i.e., $\rho_i = 0$. This models the fact that the diffusion time scale is much larger than the reactive time scale, and that in the solution there is a sufficient large number of solute molecules which are able to hydrolyze all substrates that migrate to the reaction centers of solute molecules. The non-zero flux on the reactive surface makes the particle concentrations described by PNP differ fundamentally from the Boltzmann distribution, which can be reproduced if the macroscopic flux is vanishing everywhere [72].

3.2 Finite Element Algorithms

The numerical methods are focused at some major aspects of the PNP model: the nonlinearity of the system due to the drift term; the coupling between Poisson and NP equations for both steady and unsteady diffusions.

3.2.1 Steady-State Diffusion

We first consider the finite element solution of the steady state PNP equations (21), (23). To this end we define the solution space

$$H := \left\{(\phi, \rho) \in H_0^1(\Omega) \times H_{0_1}^1(\Omega_s) \times \cdots \times H_{0_n}^1(\Omega_s)\right\} \tag{24}$$

and its finite dimensional subspace

$$S := \left\{(\phi, \rho) \in P_1(\Omega) \times \left(P_1(\Omega_s)\right)^n\right\}, \tag{25}$$

where the vector $\rho = \{\rho_j\}_{j=1}^n$, and P_1 is the space consisting of piecewise linear tetrahedral finite elements. Functions in the space

$$H_{0_i}^1 = \left\{v \in H^1(\Omega_s) : v = 0 \text{ on } \partial\Omega,\, v = 0 \text{ on } \Gamma_{D_i}\right\}$$

satisfy the Dirichlet boundary condition on the exterior boundary $\partial\Omega$ and the essential or Dirichlet boundary condition on the molecular surface Γ if there is one. We assume that the finite elements are regular and quasi-uniform. The weak formulation of the problem now is:

Find $u = (\phi, \rho) \in S$ *such that*

$$\left\langle F(u), v\right\rangle = 0, \quad \forall v = (\psi, \eta) \in S. \tag{26}$$

Here the nonlinear mapping $F : H \mapsto H^*$ and $\langle \cdot, \cdot \rangle$ is the standard duality paring between the dual space H^* and H. Specifically, the nonlinear weak form $\langle F(u), v\rangle$ is defined to be

$$\left\langle F(u), v\right\rangle = \begin{bmatrix} (\varepsilon\nabla\phi, \nabla\psi) - (\lambda \sum_i q_i\rho_i, \psi) + \langle p, \psi\rangle_\Gamma \\ (D_i\nabla\rho_i, \nabla\eta_i) + (D_i\beta q_i\rho_i\nabla\phi, \nabla\eta_i) \end{bmatrix}, \tag{27}$$

where

$$p = \varepsilon_m \frac{\partial(\phi^s + \phi^h)}{\partial n}$$

is the jump in electric displacement defined in Eq. (22), $\langle \cdot, \cdot\rangle_\Gamma$ denotes the L_2 inner product defined on the interface Γ, and the L_2 scalar inner product over the domain Ω or Ω_s is denoted by $(\cdot, \cdot)$. To solve the nonlinear problem (26) we employ the damped inexact-Newton method [41] which necessitates the Gâteaux derivative $DF(u)$ defined by the bilinear form

$$\begin{aligned}\left\langle DF(u)w, v\right\rangle &= \frac{d}{d\tau}\left\langle F(u + \tau w), v\right\rangle\Big|_{\tau=0} \\ &= \begin{bmatrix} (\varepsilon\nabla\varphi, \nabla\psi) - (\lambda \sum_i q_i\zeta_i, \psi) \\ (D_i\nabla\zeta_i, \nabla\eta_i) + D_i\beta q_i(\rho_i\nabla\varphi + \zeta_i\nabla\phi, \nabla\eta_i) \end{bmatrix}\end{aligned} \tag{28}$$

where $w = (\varphi, \zeta)$. With these well-defined operators the complete algorithm can be given as follows:

Algorithm 2

```
• Choose the initial approximation $u = (\phi, \rho)$, the nonlinear tolerance $\varepsilon$, the residual $r$ in approximately solving the linear system, and the damping factor $c$.
• Do until $|\langle F(u), v\rangle| < \varepsilon$
    1. Solve the correction $w$ from $\langle DF(u)w, v\rangle = -\langle F(u), v\rangle + r$.
    2. $u \Leftarrow u + cw$.
```

A constant damping parameter $c = 1$ is chosen in this study, with which the convergence is reached in less than 20 steps in all simulations.

It is noted here that the above algorithm solves the steady-state PNP equations as a whole system. A commonly used approach is also to solve the NPEs and PE separately. That means iteration is needed between NPEs and PE until the solutions are self-consistently converged. A standard Gummel iteration proceeds as following: given any initial solution function ϕ^0 (or ρ^0), solve the NP equations Eq. (23) in steady state (or the PE (21)) to get a solution ρ^0 (ϕ^0), then solve the PE (NPEs) with these ρ^0 (ϕ^0) to get an updated solution ϕ^1 (ρ^1), and with ϕ^1 (ρ^1) get an updated solution of NPEs ρ^2 (ϕ^2 of the PE), continue this iteration until approaching a converged solution (ρ, ϕ) of the PE and the NPEs. It is found that the standard Gummel iteration converges slowly, and may diverge in some circumstances. A γ-iteration procedure for the iteration between the NP and PE as used in our former PNP solution [55, 57] appears helpful in assisting convergence of solution for the PNP system. When obtained a solution (ρ_n, ϕ_n) of the PNP equations at the n-th step during the iterations between solutions of the PE and NPEs, we modify them for use in next iteration step by a γ-relaxation

$$\rho_i^n = \gamma \rho_i^n + (1-\gamma)\rho_i^{n-1}, \tag{29}$$

$$\phi^n = \gamma \phi^n + (1-\gamma)\phi^{n-1}. \tag{30}$$

It is found that usually under-relaxation, i.e. $\gamma < 1$ is helpful or necessary for large-sized PNP system, while over-relaxation does not help the convergence.

3.2.2 Unsteady-State Diffusion

For time-dependent problems the elliptic equation for the electrostatic potential and parabolic equations for the particle concentrations are solved sequentially. The weak form of the unsteady Nernst–Planck equation for i-th species of particle is

$$\langle F(\rho_i), v\rangle = \int_{\Omega_s} \Big[D_i \big(\nabla \rho_i + \beta q_i \rho_i \nabla \phi^r\big) \cdot \nabla v + \frac{\partial \rho_i}{\partial t} v \Big] dx, \quad \forall v \in H^1_{0_i}(\Omega_s). \tag{31}$$

Various schemes can be used for the time discretization of this equation. For example, Prohl and Schmuck proposed convergent schemes based on different types of fixed-point mappings [68]. Due to the nonlinearity of the equation, the application of these and high order methods such as a third-order Runge–Kutta method or its combination with the exponential time differencing (ETD) method [22, 60] demands solving the electrostatic potential multiple times in each step of time evolution. To reduce the computational cost and maintain the stability, we adopt the Crank–Nicolson method for the time discretization. This gives rise to the following semi-discrete equation at $t_{n+1/2}$ for $n > 0$:

$$\langle F(\rho_i^{n+1/2}), v\rangle = \int_{\Omega_s} \left[D_i \left(\nabla \frac{\rho_i^{n+1} + \rho_i^n}{2} + \beta q_i \frac{\rho_i^{n+1} + \rho_i^n}{2} \nabla \phi^{n+1/2} \right) \cdot \nabla v \right.$$
$$\left. + \frac{\rho_i^{n+1} - \rho_i^n}{\Delta t} v \right] dx \tag{32}$$

for a constant time increment Δt. Here the electrostatic potential $\phi^{n+1/2}$ is solved from the Poisson equation (21) with particle concentrations at $t_{n+1/2}$ computed with an Adams–Bashforth scheme

$$-\nabla \cdot \left(\varepsilon \nabla \phi^{n+1/2}\right) - \lambda \sum_i q_i \frac{3\rho_i^n - \rho_i^{n-1}}{2} = 0. \tag{33}$$

We then use the inexact-Newton approach presented above to solve ρ_i^{n+1} from the equation

$$\langle F(\rho_i^{n+1/2}), v\rangle = 0. \tag{34}$$

To this end we need the Gâteaux derivative $DF(\rho_i^{n+1})$, which is now defined by

$$\langle DF(\rho_i^{n+1/2})w, v\rangle = \frac{d}{d\tau} \langle F(\rho_i^{n+1/2} + \tau w), v\rangle \Big|_{\tau=0}$$
$$= \int_{\Omega_s} \left[\frac{1}{2} D_i \left(\nabla w \cdot \nabla v + \beta q_i w \nabla \phi^{n+1/2} \right) + \frac{w}{\Delta t} v \right] dx, \tag{35}$$

where $w \in H^1_{0_i}$. The solutions of (34)–(35) follow Algorithm 2 with residual $r = 0$. Since Eq. (32) is linear in ρ_i^{n+1}, only one solution of w is needed for an arbitrary initial guess of ρ_i^{n+1} at each time step. We note that a similar Adams–Bashforth–Crank–Nicolson (ABCN) method was used for solving the Navier–Stokes equations and ensuring divergence-free velocity field [69, 84]. The extrapolation of source term at t_n, t_{n-1} in Eq. (33) is similar to the construction of the pressure Poisson equation at $t_{n+1/2}$ in those studies.

3.2.3 A Symmetric Transform of the Electro-Diffusion Equations

We here introduce a commonly used transformation to the NP equations, which might be useful in future PNP-like simulations in biomolecular systems. It is known that by introducing the Slotboom variables

$$\bar{D}_i = D_i e^{-\beta q_i \phi}, \qquad \bar{\rho}_i = \rho_i e^{\beta q_i \phi}, \tag{36}$$

the Nernst–Planck equation can be transformed to be

$$\frac{\partial(\bar{\rho}_i e^{-\beta q_i \phi})}{\partial t} = \nabla \cdot (\bar{D} \nabla \bar{\rho}). \tag{37}$$

These transformations, frequently used in solving the PNP equations for semiconductor device simulations [7, 49], hence give rise to a symmetric, uniformly elliptic operator in case of a fixed potential. The application of transformations (36) to the electrostatic Poisson equation (21) will lead to

$$-\nabla \cdot (\varepsilon \nabla \phi) - \lambda \sum_i q_i \bar{\rho}_i e^{-\beta q_i \phi} = 0. \tag{38}$$

While the Eq. (38) appears identical to the nonlinear Poisson–Boltzmann equation, the actual particle concentrations, nevertheless, do not follow the Boltzmann distribution if there is a non-zero macroscopic flux inside the domain or on the boundary.

We also consider the finite element solution of transformed PNP equations (37), (38). For which the solution $u = (\phi, \bar{\rho})$ contains the transformed particle concentrations and nonlinear weak form $\langle F(u), v \rangle$ is given by

$$\big\langle F(u), v \big\rangle = \begin{bmatrix} (\varepsilon \nabla \phi, \nabla \psi) - (\lambda \sum_i q_i \bar{\rho}_i e^{-\beta q_i \phi}, \psi) + \langle p, \psi \rangle_\Gamma \\ (\bar{D}_i \nabla \bar{\rho}_i, \nabla \bar{\eta}_i) \end{bmatrix}, \tag{39}$$

where $v = (\psi, \bar{\eta})$. Accordingly, the bilinear form now is

$$\begin{aligned} \big\langle DF(u)w, v \big\rangle &= \frac{d}{d\tau} \big\langle F(u + \tau w), v \big\rangle \Big|_{\tau=0} \\ &= \begin{bmatrix} (\varepsilon \nabla \varphi, \nabla \psi) - (\lambda \sum_i (q_i \bar{\zeta}_i - \beta q_i^2 \bar{\rho}_i \varphi) e^{-\beta q_i \phi}, \psi) \\ (\bar{D}_i (\nabla \bar{\zeta}_i - \beta q_i \varphi \nabla \bar{\rho}_i), \nabla \bar{\eta}_i) \end{bmatrix} \end{aligned} \tag{40}$$

where $w = (\varphi, \bar{\zeta})$. The complete algorithm for solving the transformed PNP equations is the same as Algorithm 2 but with $\langle F(u), v \rangle$ and $\langle DF(u)w, v \rangle$ defined by (39) and (40), respectively. It is worth noting that the operator $DF(u)w$ defining the linearized equation for solving correction variable w is not symmetric regardless of the transformation due to the nonlinearity of the PNP model.

It is worth noting that the Slotboom variables are associated with the weighted inner product in many finite element approximations of semiconductor NP equations [31], for which exponential fitting techniques are usually used to obtain numerical solutions free of non-physical spurious oscillations. Although the solutions in our numerical experiments and biophysical applications presented below do not show significant non-physical oscillations, these methods can be adopted if needed. Our previous work [59] analyzed the condition number of the transformed NP equations (37) and shew that the Slotboom variables (36) can lead to quick growth of the condition number either due to large molecular permanent charge(s) or due to large difference in potential near molecular surface. However, the partial charge carried by any atom in real biomolecule is generally smaller than 2 folds of the elementary charge. Besides, solving the non-linear equation (38), instead of the linear form (21), at each step results in improved solution for the potential, especially when the density solutions of NPEs from last step deviate largely from the correct ones during the iteration. This can actually make the solution of the PNP equations converged within less iteration steps for real protein systems. The numerical properties of the Slotboom transformation or similar transform in other system [55] and its application to solution of nonlinear coupled systems need further systematic exploration. As an illustration of the usage of FEM, we shall still use the primitive formulation in this book.

3.3 PB Model as a Special Case of PNP Model

Physically, the equilibrium state is a special case of non-equilibrium state when no flux existed for any ionic species. This implies that the PB results can be obtained from the PNP system. The mathematical procedure corresponds to a relaxation of the total energy of the solvated solute-ions system.

To this end, we can numerically solve the PNPEs (either steady state or non-steady state, but non-steady state needs finally reach steady state) using the similar boundary conditions as in the usual solution of the PBE for ϕ, such as $\phi = 0$ or the Debye–Huckel approximation, at the outer boundary $\partial\Omega$, and using the ionic bulk densities as boundary conditions for ρ_i. In addition, we use a reflective condition for each ion species in the molecular interface Γ (no Γ_a for PB calculation) to enforce zero-flux across the interface

$$J(r)_i = 0, \quad r \in \Gamma.$$

Then, the solution leads to the PB results. The reason is as following: We know that the steady state PNP system has only one solution [55], and we also know that the solution of zero-flux-everywhere $J_i = 0$ (equilibrium) is a solution of the PNP system (see Eq. (19)) satisfying the interface condition, which is corresponding to the special case of the PB model. The equilibrium distribution with zero-flux condition

$$J_i = D_i(r)\big(\nabla\rho_i(r,t) + \beta\rho_i(r,t)q_i\nabla\phi(r)\big) = 0$$

can be seen equivalent to the Boltzmann distribution condition

$$\rho_i \sim e^{-\beta q_i \phi}.$$

Therefore, the PNP solution obtained from above procedure with zero-flux conditions at Γ must satisfy the zero-flux condition everywhere. This indicates that the solution of PNP is exactly the solution of the PBE. The equivalence is numerically proven true in our previous work [57], where it was shown that PBE and PNPE have essentially the same results despite a small numerical error. This fact leads to an indirect approach to solve the PB model, which sometimes shows indispensable advantage to treat certain difficult modified PB models [55].

4 Finite Element Implementation and Mesh Generation

As described above, FEM is convenient to treat the nonlinearity and complex geometries. When qualified mesh generation is available, FEMs can achieve good performance in the accuracy and memory demands.

The numerical implementation of Algorithm 1 for solving Eq. (15) for PBE, and of Algorithm 2 for solving Eq. (26), Eq. (34) and the Poisson equation (33) for the PNPEs are carried out using FETK, an expandable collection of the adaptive finite element method (AFEM) software libraries [40]. Standard linear finite element spaces and Galerkin approximation are adopted in these solutions. Recently we also improved the algorithm stability and developed a parallel solver for these equations by using the parallel AFEM software package PHG [85]. The work will be reported [83].

A volumeric mesh is prerequisite to FEM calculations. How to stably, efficiently generate a molecular mesh with correct representation of the irregular and complex molecular boundary is a challenging task in the area of mathematical continuum modeling of biomolecular systems. A mesh generation tool chain described in our former work [57] only works for not big biomolecular systems. We recently developed a new technique and software TMSmesh [17] for molecular surface meshing for general larger systems. And based on this, a tool chain can be setup for volume mesh generation. Interested readers are referred to chapter "Surface Triangular Mesh and Volume Tetrahedral Mesh Generations for Biomolecular Modeling" on molecular mesh generation of the book. It is worth noting that for PNP system, the Poisson equation and the NP equations are solved in different domains, and usually only one file of the mesh in the entire Ω and conforming to Γ is necessary for input to the code. One way to tackle this issue as in [59] is that the mesh of $\bar{\Omega}_s$ is extracted by a subprogram embedded in the solver when solving the NP equations. Another way is to solve the NP equations in the entire domain Ω, but with special numerical treatments in domain Ω_m [83].

Combined with our new mesh generation tool TMSmesh [17], the FEM solver can serve as a standalone, complete computational tool for modeling protein/DNA systems in ionic solution.

5 Numerical Experiments and Biophysical Applications

Because PB results can be generated from the more general PNP model, and the main numerical properties of the PB solution are similar to that of the PNP solution due to similar FEM schemes applied to treat the singular charges and interface conditions, in this section we will mainly focus on numerical experiments and applications of the PNP model.

Table 1 Accuracy of the numerical solutions for Eq. (41)

h_{max}	3.277	1.821	0.965	0.574	0.297
L_2	2.872(-3)	9.747(-4)	2.908(-4)	1.152(-4)	3.271(-4)
Order		1.84	1.90	1.78	1.91
k_r	1.373(11)	1.806(11)	2.149(11)	2.378(11)	2.519(11)

5.1 Steady-State Diffusion: Numerical Accuracy

Due to the intrinsic nonlinearity of the equation, the analytical solutions for the steady-state PNP equations are not available in general, even for the simplest problems such as the electrodiffusion in the spherical annulus exterior to a charged sphere. Here we choose two examples to examine the accuracy of our algorithm. The first example is to solve the Nernst–Planck system for the concentration of a single species at a given potential $\phi(r) = Q/(\varepsilon_s r)$ in a spherical annulus:

$$-\frac{1}{r^2}\frac{d}{dr}\left(r^2 D\left(\frac{d\rho}{dr} - \beta\rho q\frac{Q}{\varepsilon_s r^2}\right)\right) = 0, \quad r_1 < r < r_2$$
$$\rho(r_1) = 0, \qquad \rho(r_2) = \rho_0, \tag{41}$$

where ρ_0 is the bulk concentration. Note here we are applying a reactive boundary condition on the whole sphere $r = r_1$. The analytical solution for Eq. (41) is

$$\rho(r) = \rho_0 \frac{e^{-\beta w(r)} - e^{-\beta w(r_1)}}{e^{-\beta w(r_2)} - e^{-\beta w(r_1)}}, \quad \text{where } w(r) = \frac{qQ}{\varepsilon_s r}. \tag{42}$$

The reactive rate constant k_r is then computed from the flux $J(r)$ on the reactive surface via

$$-k_r \equiv \frac{\int_{S_A} J(r)\,ds(r)}{\rho_0} = \frac{4\pi r_1^2 J(r_1)}{\rho_0} = 4\pi D w(r_1) r_1 \frac{-e^{-\beta w(r_1)}}{e^{-\beta w(r_2)} - e^{-\beta w(r_1)}} \tag{43}$$

where S_A is the reactive surface. In this case we choose $r_1 = 1$, $r_2 = 40$, $\varepsilon_s = 78\varepsilon_0$, $\rho_0 = 50$ mM, $D = 78000$ Å/μs, $q = -1$, $Q = 1$, and thus the exact $k_r = 2.5315 \times 10^{11}$ M^{-1} min^{-1}. Table 1 lists the relative L_2 errors of the computed particle concentration, the asymptotic order of error reduction and the reaction rate constants. These results demonstrate that our finite element method is convergent for this problem, with an asymptotic rate of convergence close to 2 as anticipated for a linear finite element method. It is also noticed that the errors in the computed reactive rate constant are large for all the mesh sizes considered. This is related to the very large gradient of concentration close the reactive surface, as seen in Fig. 2 where the exact and computed concentration profiles are plotted. Physically, this large gradient is induced by the electrostatic attraction of the negatively charged particles to the positively charged sphere. In this study we use finite element meshes refined toward the molecular surface to improve the local numerical resolution. Other higher order methods can also be introduced to this problem to resolve this large gradient and improve the numerical accuracy.

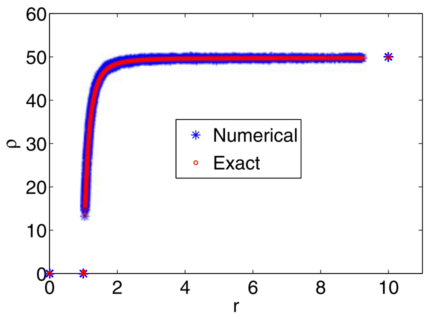


Fig. 2 The exact and computed concentration profiles for the Nernst–Planck equation in the spherical annulus $1 < r < 40$ for a given potential. The x-axis is truncated at $r = 10$ in the illustration. $h_{max} = 3.277$

The second example is to solve the full steady-state PNP equations for two species of particles, one carries charge -1 and the other has charge $+1$, in the same spherical annulus as in the last example. We prescribe the flux $J(r) = 0$ for both species on the unit sphere, and the particle concentrations on the exterior boundary are set to be the respective bulk concentrations. The macroscopic flux of either species of particles is therefore zero everywhere in the domain, and thus the PNP model shall produce the nonlinear PBE and the particle concentrations shall follow the Boltzmann distribution. This criterion is used to examine the numerical solutions of the PNP equations. We would note that there is no analytical solution of the potential available for the nonlinear PBE. Rather, we will compare the computed concentration profiles of the PNP equations and those predicted by using the Boltzmann distribution and the computed electrostatic potential. In particular, let the numerical solutions of the potential and the particle concentration be ϕ and ρ, and the exact solutions of them be $\hat{\phi}$ and $\hat{\rho}$, respectively. Let the particle concentration computed from the solved potential ϕ be $\tilde{\rho}$, then the error we are measuring is $\rho - \tilde{\rho}$. It follows that for any Sobolev norm $\|\cdot\|$ we have

$$\begin{aligned}
\|\rho - \tilde{\rho}\| &\le \|\rho - \hat{\rho}\| + \|\hat{\rho} - \tilde{\rho}\| \\
&= \|\rho - \hat{\rho}\| + \left\|\rho_0 e^{-q\beta\hat{\phi}} - \rho_0 e^{-q\beta\phi}\right\| \\
&\le \|\rho - \hat{\rho}\| + \left\|\rho_0 e^{-q\beta\hat{\phi}}\right\|_\infty \left\|e^{-q\beta(\phi-\hat{\phi})} - 1\right\| \\
&\approx \|\rho - \hat{\rho}\| + \left\|\rho_0 e^{-q\beta\hat{\phi}}\right\|_\infty \left\|q\beta(\phi - \hat{\phi})\right\| \\
&= \|\rho - \hat{\rho}\| + C\|\phi - \hat{\phi}\|,
\end{aligned} \tag{44}$$

where the constant C is independent of the numerical methods. This estimate suggests that the error we are measuring has the same rate of convergence as the error of solutions of the PNP equations. Table 2 shows that the rate with respect to L_2 norm is about 1, which is close to the one predicted for the linear elliptic interface

Table 2 L_2 errors between the computed particle concentrations and those predicted by the Boltzmann distribution

h_{max}	3.277	1.821	0.965	0.574	0.297
$L_2(\rho)$	1.715(-2)	9.437(-3)	5.095(-3)	2.726(-3)	1.280(-3)

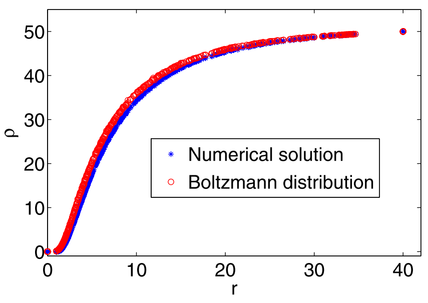


Fig. 3 Computed concentration profiles and the Boltzmann distribution for particles with $q = 1$. $h_{max} = 3.277$

problems in [18]. Figure 3 plots the computed particle concentration and that predicted by the Boltzmann distribution. The flattening of the profile close to $r = 1$ indicates the vanishing concentration due to the electrostatic repulsion and the vanishing macroscopic flux as prescribed by the boundary condition.

5.2 *Accuracy for Solving the Unsteady-State Diffusion*

To examine the accuracy of the time integration method we design a problem that has the essential features of the PNP and admits an analytical solution:

$$-\nabla \cdot (\varepsilon_s \nabla \phi) = q\rho + f(r), \tag{45}$$

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (D\nabla \rho + \beta q \rho \nabla \phi) + g(r). \tag{46}$$

This equation is solved in the spherical annulus $1 \le r \le 4$. The analytical solutions for ϕ and ρ are prescribed to be

$$\phi = \frac{r^2}{\varepsilon_s} e^{-\delta t}, \tag{47}$$

$$\rho = \rho e^{-\beta q r^2/\varepsilon_s} e^{-\delta t}. \tag{48}$$

These two analytical solutions determine the functions $f(r)$, $g(r)$ and the Dirichlet boundary conditions for both equations. A very fine mesh with 40859 unknowns is used to ensure that the error due to the time discretization is dominant in the numerical approximation. The equations are integrated to $t = 200$ with various time increments Δt and fixed parameter $\delta = 0.01$. The relative L_2 errors are collected in Table 3, which features a convergence of approximately second-order for both variables. This agrees with the convergence of the ABCN scheme applied for solving the Navier–Stokes equations [69]. It is worth noting that here we are using large time increments in time integration; the convergence properties we observed in this study agree with the theoretical analysis [39] which proves that the ABCN for time-dependent Navier–Stokes equations is almost unconditionally stable.

Table 3 Numerical error and asymptotic order of convergence for time integration

Δt	e_ϕ	Order	e_ρ	Order
2	5.33(-3)		1.16(-2)	
1	1.47(-3)	1.86	3.65(-3)	1.67
0.5	3.86(-4)	1.93	9.33(-4)	1.97
0.25	1.02(-4)	1.92	2.52(-4)	1.89

5.3 *Biophysical Applications: Diffusion-Reaction Study of AChE-ACh System*

Finally we apply the regularized PNP solver to compute the reaction rate constant of neurotransmitter acetylcholine (ACh) at the reaction center of the enzyme acetylcholinesterase (AChE). The electrodiffusion reaction for the same system has been studied by using the Smoluchowski equation [77], in which the electric field is fixed and approximated by a PB solution. This approximation agrees with the underlying assumption of the well-known Debye–Hückel limiting law (DHL) describing the ionic screening effect to reaction rate constant. The DHL for AChE-ACh system is [70]:

$$k_{\text{on}} = \left(k_{\text{on}}^0 - k_{\text{on}}^H\right) 10^{-1.18|z_E z_I|\sqrt{I}} + k_{\text{on}}^H,$$

where k_{on}, k_{on}^0, and k_{on}^H are second-order association rate constants at the specified ionic strength I, zero ionic strength, and infinite ionic strength, respectively. z_E and z_I are the charges of the enzyme and substrate involved in the interaction. With the same assumption, Song et al.'s numerical results recover the DHL. The more complete PNP model also takes into account the charged substrate influence to the electric field around the enzyme, therefore leads to improved rate prediction. Here, we will show by using a more sophisticated PNP model the reaction rate coefficient obviously depends not only on the ionic strength, but also on the substrate concentration itself [54, 57, 87]. We treat the ACh molecules as particles with $+1$ charge. The computation domain is chosen to be a ball with a radius 400 Å centered at the geometric center of the AChE molecule. We consider two species of background "spectator" ions (non-reactive), one is cation with $+1$ charge and the other is anion with -1 charge. The boundary conditions for these two species of particles are therefore $J_i(r) = 0$ on the whole surface of AChE. The reaction center of the AChE is signified in Fig. 4 in red where $\rho_i = 0$ is set for ACh as the reactive boundary conditions, and on the rest surface the $J_i(r) = 0$ is prescribed. Suppose that C_+ and C_- are the bulk concentrations of cation and anion respectively, and that C_{subs} is the bulk concentration of substrate ACh. These bulk concentrations are used as the outer boundary conditions of the diffusion domain in solving the NP equations. Therefore, to make a closer connection with physiology, it is reasonable to consider a neutrality condition of the bulk solution in this work as $C_+ + C_{\text{subs}} - C_- = 0$. The same mesh as that in [87] is used in this study. The electrostatic potential on the surface of AChE is shown in Fig. 4 along with the surface mesh and a close view of

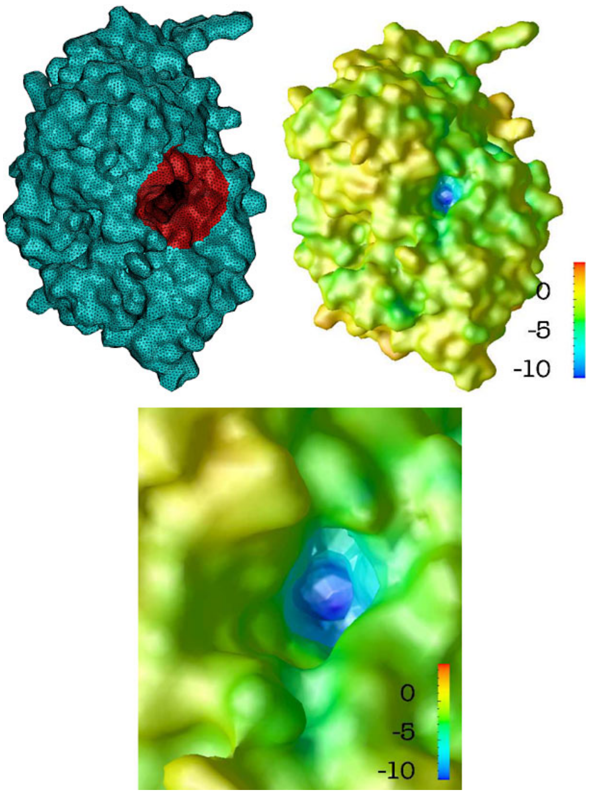
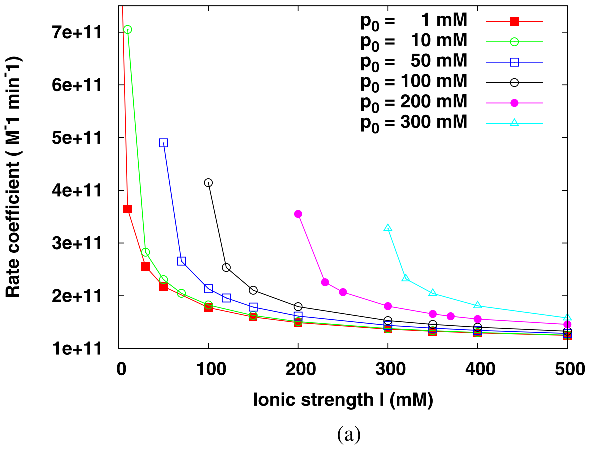


Fig. 4 The discretized molecular surface of AChE with the region around the reaction center colored red (*left*); The electrostatic potential on the surface (*middle*) and the surface potential around the reaction center (*right*). Ionic strength = 50 mM

the potential around the reaction center. The surface potential is smooth overall and the negative potential near the reaction center is well reproduced.

The reaction rate coefficient is shown as a function of ionic strength (= "spectator" + bulk substrate) for different prescribed substrate concentrations in Fig. 5(a) and as a function of bulk substrate concentration for different prescribed ionic strengths in Fig. 5(b). The results show that the reaction rate coefficients strongly depend on both ionic strength and substrate concentration. At very low substrate concentration, e.g., 1 mM or less, the results show asymptotic agreement with the DHL (see red line in Fig. 5(a)). The find also agrees with the continuum model when the substrate density is not coupled into the full electric field [57, 77, 87]). However, at moderate concentrations of the substrate, the curves are shifted. A general trend is observed: the rate coefficient increases as the bulk/distant concentration of substrate increases for a fixed overall ionic strength. For instance, for a fixed ionic strength of 300 mM ($C_+ + C_{\text{subs}} = 300$ mM), the rate coefficient is 1.36×10^{11} M^{-1} min^{-1} for $C_{\text{subs}} = 1$ mM and is increased to 3.28×10^{11} M^{-1} min^{-1} for $C_{\text{subs}} = 300$ mM. The physical origins of the observed behavior can be explained as follows. If substrate concentration is not considered, as in most previous work based on the DHL, the concentration of the counter ion of the enzyme, i.e., C_+ here, is equal to the concentration of the co-ion, i.e., $C_+ = C_-$. The counter ions are attracted and concentrated



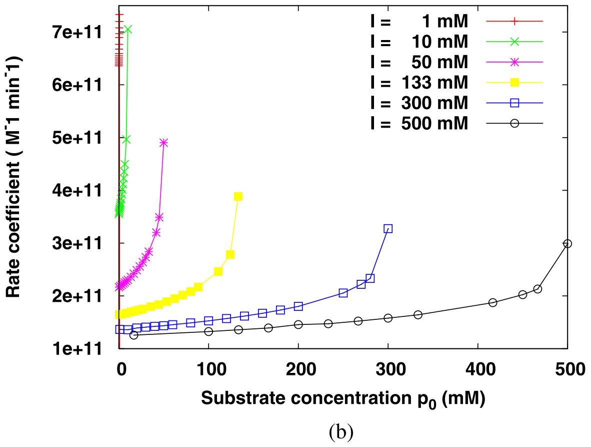


Fig. 5 Reaction rate coefficients for ACh-AChE reaction system. p_0 is bulk substrate concentration (C_{subs}); I is total ionic strength (spectator ions plus substrate)

around the negatively charged active site, which serves to screen the Coulomb interaction between ACh molecules and AChE, hence slowing the association. When C_{subs} is considered in the PNP model, to maintain the same ionic strength, C_+ needs to be reduced by C_{subs} compared with that in the familiar Debye–Hückel theory. This leads to a thinner counter-ion atmosphere around the active site, and it can not be compensated by the additional substrate (ACh) density that is relatively low due to reactant depletion that results from the absorbing boundary condition. In other words, in the resulting non-equilibrium state, the sum of counter-ion density and ACh density near the active site is lower than that obtained with the Boltzmann distribution for a $+1$e particle. The consequences are a reduced overall screening effect and thereby an enhanced reaction rate.

The ionic atmosphere always screens the electrostatic interactions, and hence reduces the rate coefficients. At very high ionic strength, due to strong ionic screening effects, the electrostatic interactions become very weak. This is close to the pure diffusion case, and all the rate constants for different substrate concentrations are close to the pure diffusion-reaction rate constant.

The phenomena observed in above rate coefficient predictions are expected to be general for attractive substrate-enzyme systems.

6 Conclusions

A finite element method is described for solving the PBE and PNP equations with permanent atomic charges within molecular region. The electrostatic PB or Poisson equation is regularized by analytically removing the singular component of the electrostatic potential from the numerical solution. A harmonic component is defined inside biomolecules to partially compensate the removed singular component such that the remaining electrostatic component is continuous on the molecular surface. This remaining regular component is governed by an elliptic interface problem, with interface conditions computed from the singular and the harmonic components. For PNP system, it is shown that the diffusion in the solvent region is completely drifted by the regular component, which gives rise to regularized Poisson–Nernst–Planck equations. An inexact-Newton method was used to solve the regular PBE and the regular steady-state PNP systems. For unsteady diffusion a second-order Adams–Bashforth–Crank–Nicolson method is proposed for time integration. Various test problems to examine the accuracy and the stability of the proposed 3D finite element methods and time integration scheme.

In the application to simulations of the electro-diffusion controlled reaction processes, we find that the DHL only applies to very dilute situations. Our numerical results show that for electrostatically steered diffusion-controlled reaction processes, the rate coefficients strongly depend on both ionic strength and substrate concentration. In particular, at the same ionic strength, the current model predicts that increasing substrate concentration results in significant increase in rate coefficients for the attractive substrate-enzyme systems in case the product concentration can be ignored (the product effects is not considered in current model).

We also show that the non-linear PB model is a special case of the PNP model, and can be implicitly achieved through the solution of the PNP model by appropriately controlling the boundary/interface conditions. By taking such an advantage, a recent work [55] shew that a more complicated, non-uniform ionic size-modified PB model can be numerically achieved through solution of a size-modified PNP model. This indicates that PNP-like model seems a powerful framework to achieve extended PB or PNP models beyond the current mean field approximation. Because all of those models, in addition to possessing all the features such as permanent charges, irregular interface and so on as aforementioned, are intrinsically strong non-linear, and may be coupled, finite element method can serve as a powerful tool for numerical simulation of these models. The other type of nonlinear models, such as a coupled elastic equation and a Poisson equation describing the elastic deformation of a protein-membrane interacting system was also effectively solved using a finite element method [88].

The accurate and stable FEM scheme can also achieve high efficiency with contemporary developments in adaptive, multi-level multi-grid, and parallelization techniques in FEM area. Some FEM soft packages, such as FETK [40] that uses AMG technique, PHG [85] that is parallelized, are also available. This makes it a promising numerical method for some future applications to such as supermolecular energy/mechanics analysis, ion-channel simulation, molecular conformation sampling, and multi-scale multi-physics modeling of other molecular/cellular activities. Finally, the current FEM PB/PNP solvers, combining with our new mesh generation tool TMSmesh [17], can be standalone and complete computational tools for modeling protein/DNA systems in ionic solution.

Acknowledgements The author was supported by the State Key Laboratory of Scientific/Engineering Computing, the National Center for Mathematics and Interdisciplinary Sciences, Chinese Academy of Sciences, and the China NSF (NSFC10971218, NSFC11001257).

References

1. Abaid N, Eisenberg RS, Liu W (2008) Asymptotic expansions of I-V relations via a Poisson–Nernst–Planck system. SIAM J Appl Dyn Syst 7(4):1507–1526
2. Aksoylu B, Holst M (2006) Optimality of multilevel preconditioners for local mesh refinement in three dimensions. SIAM J Numer Anal 44(3):1005–1025
3. Baker NA (2005) Biomolecular applications of Poisson–Boltzmann methods. In: Lipkowitz KB, Larter R, Cundari TR (eds) Reviews in computational chemistry, vol 21. Wiley, Hoboken, pp 349–379
4. Baker NA, Holst M, Wang F (2000) Adaptive multilevel finite element solution of the Poisson–Boltzmann equation II: refinement schemes based on solvent accessible surfaces. J Comput Chem 21:1343–1352
5. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. Proc Natl Acad Sci USA 98:10037–10041
6. Baker NA, Bashford D, Case DA (2006) Implicit solvent electrostatics in biomolecular simulation. In: Leimkuhler B, Chipot C, Elber R, Laaksonen A, Mark A, Schlick T, Schutte C, Skeel R (eds) New algorithms for macromolecular simulation. Springer, Berlin, pp 263–295
7. Bank RE, Rose DJ, Fichtner W (1983) Numerical methods for semiconductor device simulation. SIAM J Sci Stat Comput 4:416–435
8. Barcilon V, Chen DP, Eisenberg RS, Jerome JW (1997) Qualitative properties of steady-state Poisson–Nernst–Planck systems: perturbation and simulation study. SIAM J Appl Math 57(3):631–648
9. Berg OG, von Hippel PH (1985) Diffusion-controlled macromolecular interactions. Annu Rev Biophys Biophys Chem 14:131–160
10. Biler P, Hebisch W, Nadzieja T (1994) The Debye system: existence and large time behavior of solutions. Nonlinear Anal 23:1189–1209
11. Bolintineanu DS, Sayyed-Ahmad A, Davis HT, Kaznessis YN (2009) Poisson–Nernst–Planck models of nonequilibrium ion electrodiffusion through a protegrin transmembrane pore. PLoS Comput Biol 5(1):e1000277
12. Boschitsch AH, Fenley MO (2004) Hybrid boundary element and finite difference method for solving the nonlinear Poisson–Boltzmann equation. J Comput Chem 25(7):935–955
13. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M (1983) Charmm: a program for macromolecular energy, minimization, and dynamics calculations. J Comput Chem 4:187–217
14. Cardenas AE, Coalson RD, Kurnikova MG (2000) Three-dimensional Poisson–Nernst–Planck theory studies: influence of membrane electrostatics on gramicidin a channel conductance. Biophys J 79(1):80–93

15. Chapman DL (1913) A contribution to the theory of electrocapollarity. Philos Mag 25:475–481
16. Chen L, Holst M, Xu J (2007) The finite element approximation of the nonlinear Poisson–Boltzmann equation. SIAM J Numer Anal 45(6):2298–2320
17. Chen MX, Lu BZ (2011) TMSmesh: a robust method for molecular surface mesh generation using a trace technique. J Chem Theory Comput 7(1):203–212
18. Chen Z, Zou J (1998) Finite element methods and their convergence for elliptic and parabolic interface problems. Numer Math 79(2):175–202
19. Chern IL, Liu JG, Wang WC (2003) Accurate evaluation of electrostatics for macromolecules in solution. Methods Appl Anal 10:309–328
20. Cohen H, Cooley JW (1965) The numerical solution of the time-dependent Nernst–Planck equations. Biophys J 5:145–162
21. Cortis CM, Friesner RA (1997) Numerical solution of the Poisson–Boltzmann equation using tetrahedral finite-element meshes. J Comput Chem 18(13):1591–1608
22. Cox SM, Matthews PC (2002) Exponential time differencing for stiff systems. J Comput Phys 176(2):430–455. doi:10.1006/jcph.2002.6995
23. Davis ME, McCammon JA (1990) Electrostatics in biomolecular structure and dynamics. Chem Rev 90(3):509–521
24. Debye P, Huckel E (1923) Zur theorie der elektrolyte. Phys Z 24:185–206
25. Derjaguin B, Landau L (1941) Theory of the stability of strongly charged lyophobic sols and the adhesion of strongly charged particles in solutions of electrolytes. Acta Physicochim (USSR) 14:633–662
26. Eisenberg B, Liu W (2007) Poisson–Nernst–Planck systems for ion channels with permanent charges. SIAM J Math Anal 38(6):1932–1966
27. Eisenberg R, Chen DP (1993) Poisson–Nernst–Planck (PNP) theory of an open ionic channel. Biophys J 64(2):A22
28. Feig M, Brooks CL (2004) Recent advances in the development and application of implicit solvent models in biomolecule simulations. Curr Opin Struct Biol 14(2):217–224
29. Feig M, Onufriev A, Lee MS, Im W, Case DA, Brooks CL (2004) Performance comparison of generalized Born and Poisson methods in the calculation of electrostatic solvation energies for protein structures. J Comput Chem 25(2):265–284
30. Fogolari F, Brigo A, Molinari H (2002) The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology. J Mol Recognit 15(6):377–392
31. Gatti E, Micheletti S, Sacco R (1998) A new Galerkin framework for the drift-diffusion equation in semiconductors. East-West J Numer Math 6:101–135
32. Geng WH, Yu SN, Wei GW (2007) Treatment of charge singularities in implicit solvent models. J Chem Phys 127(11):114106
33. Gillespie D, Nonner W, Eisenberg RS (2002) Coupling Poisson–Nernst–Planck and density functional theory to calculate ion flux. J Phys, Condens Matter 14(46):12129–12145
34. Gilson MK, Honig BH (1987) Calculation of electrostatic potentials in an enzyme active-site. Nature 330(6143):84–86
35. Gilson MK, Sharp KA, Honig BH (1988) Calculating the electrostatic potential of molecules in solution—method and error assessment. J Comput Chem 9(4):327–335
36. Gilson MK, Davis ME, Luty BA, McCammon JA (1993) Computation of electrostatic forces on solvated molecules using the Poisson–Boltzmann equation. J Phys Chem 97(14):3591–3600
37. Gouy G (1910) Constitution of the electric charge at the surface of an electrolyte. J Phys 9:457–468
38. Guyomarch G, Lee CO (2004) A discontinuous Galerkin method for elliptic interface problems with applications to electroporation. Technical report, Korea Advanced Institute of Science and Technology
39. He Y, Sun W (2007) Stability and convergence of the Crank–Nicolson/Adams–Bashforth scheme for the time-dependent Navier–Stokes equations. SIAM J Numer Anal 45(2):837–869

40. Holst M Finite element toolkit. http://www.fetk.org/
41. Holst M (2001) Adaptive numerical treatment of elliptic systems on manifolds. Adv Comput Math 15(1–4):139–191
42. Holst M, Saied F (1995) Numerical solution of the nonlinear Poisson–Boltzmann equation: developing more robust and efficient methods. J Comput Chem 16:337–364
43. Holst M, Saied F (1993) Multigrid solution of the Poisson–Boltzmann equation. J Comput Chem 14(1):105–113
44. Holst M, Baker NA, Wang F (2000) Adaptive multilevel finite element solution of the Poisson–Boltzmann equation I: algorithms and examples. J Comput Chem 21:1319–1342
45. Holst M, McCammon JA, Yu Z, Zhou YC, Zhu Y (2011) Adaptive finite element modeling techniques for the Poisson–Boltzmann equation. Commun Comput Phys 11:179–214
46. Holst MJ (1993) Multilevel methods for the Poisson–Boltzmann equation. PhD thesis, University of Illinois at Urbana-Champaign
47. Honig B, Nicholls A (1995) Classical electrostatics in biology and chemistry. Science 268(5214):1144–1149
48. Jerome JW (1985) Consistency of semiconductor modeling: an existence/stability analysis for the stationary van Boosbroeck system. SIAM J Appl Math 45:565–590
49. Jerome JW (1996) Analysis of charge transport: a mathematical study of semiconductor devices. Springer, Berlin
50. Jerome JW, Kerkhoven T (1991) A finite element approximation theory for the drift diffusion semiconductor model. SIAM J Numer Anal 28(2):403–422
51. Koehl P (2006) Electrostatics calculations: latest methodological advances. Curr Opin Struct Biol 16(2):142–151
52. Kurnikova MG, Coalson RD, Graf P, Nitzan A (1999) A lattice relaxation algorithm for 3D Poisson–Nernst–Planck theory with application to ion transport through the gramicidin a channel. Biophys J 76(2):642–656
53. Liu WS (2005) Geometric singular perturbation approach to steady-state Poisson–Nernst–Planck systems. SIAM J Appl Math 65(3):754–766
54. Lu BZ, McCammon JA (2010) Kinetics of diffusion-controlled enzymatic reactions with charged substrates. PMC Biophys 3(1):1
55. Lu BZ, Zhou YC (2011) Poisson–Nernst–Planck equations for simulating biomolecular diffusion-reaction processes II: size effects on ionic distributions and diffusion-reaction rates. Biophys J 100(10):2475–2485
56. Lu BZ, Cheng XL, McCammon JA (2007) "New-version-fast-multipole method" accelerated electrostatic calculations in biomolecular systems. J Comput Phys 226(2):1348–1366
57. Lu BZ, Zhou YC, Huber GA, Bond SD, Holst MJ, McCammon JA (2007) Electrodiffusion: a continuum modeling framework for biomolecular systems with realistic spatiotemporal resolution. J Chem Phys 127(13):135102
58. Lu BZ, Zhou YC, Holst M, McCammon JA (2008) Recent progress in numerical solution of the Poisson–Boltzmann equation for biophysical applications. Commun Comput Phys 3(5):973–1009
59. Lu BZ, Holst MJ, McCammon JA, Zhou YC (2010) Poisson–Nernst–Planck equations for simulating biomolecular diffusion-reaction processes I: finite element solutions. J Comput Phys 229(19):6979–6994
60. Lu T, Cai W (2008) A Fourier spectral-discontinuous Galerkin method for time-dependent 3-D Schrodinger–Poisson equations with discontinuous potentials. J Comput Appl Math 220:588–614
61. Luty BA, Davis ME, McCammon JA (1992) Solving the finite-difference nonlinear Poisson–Boltzmann equation. J Comput Chem 13(9):1114–1118
62. Mori Y, Jerome JW, Peskin CS (2007) A three-dimensional model of cellular electrical activity. Bull Inst Math Acad Sin 2:367–390
63. Mori Y, Fishman GI, Peskin CS (2008) Ephaptic conduction in a cardiac strand model with 3D electrodiffusion. Proc Natl Acad Sci USA 105:6463–6468

64. Nadler B, Schuss Z, Singer A, Eisenberg RS (2004) Ionic diffusion through confined geometries: from Langevin equations to partial differential equations. J Phys, Condens Matter 16(22):S2153–S2165
65. Nernst W (1889) Die elektromotorische wirksamkeit der ionen. Z Phys Chem 4:129
66. Orozco M, Luque F (2000) Theoretical methods for the description of the solvent effect in biomolecular systems. Chem Rev 100(11):4187–4226
67. Planck M (1890) Über die erregung von electricität und wärme in electrolyten. Ann Phys Chem 39:161
68. Prohl A, Schmuck M (2009) Convergent discretizations for the Nernst–Planck–Poisson system. Numer Math 111(4):591–630. doi:10.1007/s00211-008-0194-2
69. Quere PL, de Roquefort TA (1982) Computation of natural convection in two-dimension cavities with Chebyshev polynomials. J Chem Phys 57:210–228
70. Radic Z, Quinn DM, McCammon JA, Taylor P (1997) Electrostatic influence on the kinetics of ligand binding to acetylcholinesterase—distinctions between active center ligands and fasciculin. J Biol Chem 272(37):23265–23277
71. Roux B, Simonson T (1999) Implicit solvent models. Biophys Chem 78(1–2):1–20
72. Rubinstein I (1990) Electro-diffusion of ions. SIAM, Philadelphia
73. Schuss Z, Nadler B, Eisenberg RS (2001) Derivation of Poisson and Nernst–Planck equations in a bath and channel from a molecular model. Phys Rev E 64(3):036116
74. Sharp K, Honig B (1989) Lattice models of electrostatic interactions—the finite-difference Poisson–Boltzmann method
75. Shestakov AI, Milovich JL, Noy A (2002) Solution of the nonlinear Poisson–Boltzmann equation using pseudo-transient continuation and the finite element method. J Colloid Interface Sci 247(1):62–79
76. Song YH, Zhang YJ, Bajaj CL, Baker NA (2004) Continuum diffusion reaction rate calculations of wild-type and mutant mouse acetylcholinesterase: adaptive finite element analysis. Biophys J 87(3):1558–1566
77. Song YH, Zhang YJ, Shen TY, Bajaj CL, McCammon JA, Baker NA (2004) Finite element solution of the steady-state Smoluchowski equation for rate constant calculations. Biophys J 86(4):2017–2029
78. Tai KS, Bond SD, Macmillan HR, Baker NA, Holst MJ, McCammon JA (2003) Finite element simulations of acetylcholine diffusion in neuromuscular junctions. Biophys J 84(4):2234–2241
79. Verwey EJW, Overbeek JTG (1948) Theory of the stability of lyophobic colloids. Elsevier, Amsterdam
80. Warwicker J, Watson HC (1982) Calculation of the electric-potential in the active-site cleft due to alpha-helix dipoles. J Mol Biol 157(4):671–679
81. Weiser J, Shenkin PS, Still WC (1999) Optimization of Gaussian surface calculations and extension to solvent-accessible surface areas. J Comput Chem 20:688–703
82. Xie D, Zhou S (2007) A new minimization protocol for solving nonlinear Poisson–Boltzmann mortar finite element equation. BIT Numer Math 47(4):853–871
83. Xie Y, Cheng J, Lu BZ, Zhang LB Parallel adaptive finite element algorithms for solving the coupled electro-diffusion equations (submitted)
84. Yang SY, Zhou YC, Wei GW (2002) Comparison of the Discrete Singular Convolution algorithm and the Fourier pseudospectral method for solving partial differential equations. Comput Phys Commun 143:113–135
85. Zhang L (2007) PHG: parallel hierarchical grid. http://lsec.cc.ac.cn/phg/
86. Zhou YC, Feig M, Wei GW (2007) Highly accurate biomolecular electrostatics in continuum dielectric environments. J Comput Chem 29:87–97
87. Zhou YC, Lu BZ, Huber GA, Holst MJ, McCammon JA (2008) Continuum simulations of acetylcholine consumption by acetylcholinesterase—a Poisson–Nernst–Planck approach. J Phys Chem B 112(2):270–275
88. Zhou YC, Lu BZ, Gorfe AA (2010) Continuum electromechanical modeling of protein-membrane interactions. Phys Rev E 82(4):041923

89. Zhou ZX, Payne P, Vasquez M, Kuhn N, Levitt M (1996) Finite-difference solution of the Poisson–Boltzmann equation: complete elimination of self-energy. J Comput Chem 17(11):1344–1351