University of Wisconsin Milwaukee
UWM Digital Commons

Theses and Dissertations

12-1-2016

Feature-sensitive and Adaptive Image Triangulation: A Super-pixel-based Scheme for Image Segmentation and Mesh Generation

Ming Xu
University of Wisconsin-Milwaukee

Follow this and additional works at: https://dc.uwm.edu/etd

Part of the Computer Sciences Commons

Recommended Citation

Xu, Ming, "Feature-sensitive and Adaptive Image Triangulation: A Super-pixel-based Scheme for Image Segmentation and Mesh Generation" (2016). *Theses and Dissertations*. 1431.
https://dc.uwm.edu/etd/1431

This Dissertation is brought to you for free and open access by UWM Digital Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UWM Digital Commons. For more information, please contact open-access@uwm.edu.

FEATURE-SENSITIVE AND ADAPTIVE IMAGE TRIANGULATION: A SUPER-PIXEL-BASED SCHEME FOR IMAGE SEGMENTATION AND MESH GENERATION

by

Ming Xu

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Engineering

at

The University of Wisconsin–Milwaukee

December 2016

ABSTRACT

FEATURE-SENSITIVE AND ADAPTIVE IMAGE TRIANGULATION: A SUPER-PIXEL-BASED SCHEME FOR IMAGE SEGMENTATION AND MESH GENERATION

by

Ming Xu

The University of Wisconsin–Milwaukee, 2016
Under the Supervision of Professor Zeyun Yu

With increasing utilization of various imaging techniques (such as CT, MRI and PET) in medical fields, it is often in great need to computationally extract the boundaries of objects of interest, a process commonly known as image segmentation. While numerous approaches have been proposed in literature on automatic/semi-automatic image segmentation, most of these approaches are based on image pixels. The number of pixels in an image can be huge, especially for 3D imaging volumes, which renders the pixel-based image segmentation process inevitably slow. On the other hand, 3D mesh generation from imaging data has become important not only for visualization and quantification but more critically for finite element based numerical simulation. Traditionally image-based mesh generation follows such a procedure as: (1) image boundary segmentation, (2) surface mesh generation from segmented boundaries, and (3) volumetric (e.g., tetrahedral) mesh generation from surface meshes. These three majors steps have been commonly treated as separate algorithms/steps and hence image information, once segmented, is not considered any more in mesh generation.

In this thesis, we investigate a super-pixel based scheme that integrates both image segmentation and mesh generation into a single method, making mesh generation truly an image-incorporated approach. Our method, called image content-aware mesh generation, consists of several main steps. First, we generate a set of feature-sensitive, and adaptively distributed points from 2D grayscale images or 3D volumes. A novel image edge enhancement method via randomized shortest paths is introduced to be an optional choice to generate the features' boundary map in mesh node generation step. Second, a Delaunay-triangulation generator (2D) or tetrahedral mesh generator (3D) is then utilized to generate a 2D triangulation or 3D tetrahedral mesh. The generated triangulation (or tetrahedralization) provides an adaptive partitioning of a given image (or volume). Each cluster of pixels within a triangle (or voxels within a tetrahedron) is called a super-pixel, which forms one of the nodes of a graph and adjacent super-pixels give an edge of the graph. A graph-cut method is then applied to the graph to define the boundary between two subsets of the graph, resulting in good boundary segmentations with high quality meshes. Thanks to the significantly reduced number of elements (super-pixels) as compared to that of pixels in an image, the super-pixel based segmentation method has tremendously improved the segmentation speed, making it feasible for real-time feature detection. In addition, the incorporation of image segmentation into mesh generation makes the generated mesh well adapted to image features, a desired property known as feature-preserving mesh generation.

TABLE OF CONTENTS

Abstract……………………………………………………………………………….…....ii

1 Introduction………………………………………………………………………1

1.1 Image Segmentation………………………….………….………………………...3

1.2 2D and 3D Mesh Generation ……….……………………………………….….……...4

1.3 Objectives and Contributions…………………………………………….….….……6

2 Feature-Sensitive and Adaptive Mesh Generation on 2D Images…………...……...9

2.1 Node Generation………………………………………………………….….……..9

2.1.1 Canny Sample Points……………………………………………….….…10

2.1.2 Halftoning Sample Points………………………………………..….…..12

2.1.3 Uniform Sample Points……………………………………………..….…..14

2.2 Mesh Generation via Delaunay Triangulation……………………………….……...14

2.3 Mesh Smoothing………………………………………………………………...15

2.3.1 Image-based CVT Scheme………………………………………..….….16

2.3.2 Image-based ODT Scheme……………………………………………..….18

2.3.3 Edge Flipping Strategy……………………………………………….……18

2.4 Experimental Results…………………………………………………...…....21

3 Super-Pixels Based 2D Image Segmentation……………………….……….….……35

3.1 S-t Graph Cut Method…………………………………………………….……36

3.1.1 Background of Graph…………………………………………………......36

3.1.2 Edges Weight Assignment…………………………………………….....37

3.1.3 Image segmentation based on Min-cut……………………………..…………39

3.2 Experimental results……………………………………………………..….43

4 3D Adaptive Meshes Generation and 3D Image Segmentation……………..……………53

4.1 Nodes Generation……………………………………………………………………54

4.1.1 Canny Sample Nodes……………………………………………..………..54

4.1.2 Surfaces Nodes……………………………………………………..…...56

4.2 Mesh Generation via Delaunay Tetrahedralization…..……..………………………57

4.3 Super-voxels Based 3D Segmentation……………………..………………...…..…58

4.4 Surface Smoothing…………………………………………………………….……..59

4.5 Experimental results………………………………………………………….......59

5 Image Edge Enhancement and Segmentation via Randomized Shortest Paths…..……68

5.1 Seeds Generation……………………………………………………….………70

5.1.1 Initial Seed Selection by Users……………………………………………….…..70

5.1.2 Generating Marching Distance Maps……………………………………...…71

5.1.3 Random Seed Selection………………………………………………………….…73

5.2 Image Edge Enhancement by Shortest Paths………………………………..…..…74

5.3 Edge Detection from the Enhanced Edges……………………………………..…76

5.4 Experimental Results……………………………………………………..……77

6 Conclusion……………………………………………………………………………..84

References………………………………………………………………………….…...85

Appendices 98

Appendix A: Publications 99

Curriculum Vitae 100

LIST OF FIGURES

1.1 (a) MRI scanner system cutaway. (b) MRI scanner gradient magnets. (a) and (b) are from national high magnetic field laboratory. http://www.magnet.fsu.edu ……………………1

1.2 Biomedical image segmentation. www.researchgate.net...2

2.1 The original image of 192*256 pixels. (b) The result image of the canny edge detector, where = 0.7, lower threshod =0.1, and higher threshod = 0.6. The region in red shows an example of curved-boundary or tiny features. The region in blue is an example of straight-boundary features. (c) The adaptively sampled points from (b). The red and blue rectangles show the results of curved-boundary and straight-boundary features respectively. (d) The final result of adaptive node generation. The green points are the canny edge points and the red points are the halftoning and uniform sample points. …………………11

2.2 Initial triangle mesh generated by Delaunay triangulation. ……………………………15

2.3 The one-ring neighborhood of x0. Tj is one of the adjacent triangles in the neighborhood. Cj is the centroid of the Tj . (b) The one-ring neighborhood of x0. xk is one of the neighboring vertices of x0. Skl and Skr are the left and right triangles of edge x0, xk respectively…………………………………………………………………………17

2.4 (a) Initial triangle mesh generated by Delaunay triangulation. (b) A closeup look of (a). (c)Smoothed mesh by the CVT method based on the image features after five iterations. (d) A closeup look of (c). (e) Smoothed mesh by the ODT method based on the image features after five iterations. (f) A closeup look of (e)…………………………………..21

2.5 Result of our method running on a 128*128 heart image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 15 ms. The time for PCA sampling is 12ms (c) The result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 15 ms (d) Result of the image-based ODT method. The time for ODT smoothing is 16ms (e) Result of the image-based CVT method. The time of the CVT smoothing is 16ms (f) A closeup look at the rectangular region in (e). (g) A quantitative measure of the quality of the final triangulations. The minimum angle is 4.57 degree. The maximum angle is 165.96 degree. The minimum size of the triangles is 0.499. The maximum size of the triangles is 32.99.22

2.6 Result of our method running on a 256* 256 knee image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 15 ms. The time for PCA sampling is 31 ms. (c) The result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 15 ms. (d) Result mesh of the image-based ODT method. The time for ODT smoothing is 78 ms. (e) Result mesh of the image-based CVT method. The time of the CVT smoothing is 62 ms (f) A closeup look at the rectangular region in (e)……………………………………………..24

2.7 Result of our method running on a 256*256 cardiac cell image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 46 ms. The time for PCA sampling is 31 ms (c) Result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 15 ms (d) Result of the image-based ODT method. The time for ODT smoothing is 47 ms (e) Result of the image-based CVT method. The time of the CVT smoothing is 47 ms (f) A closeup look at the rectangular region in (e)……………………………………………...27

2.8 Result of our method running on a 1032*912 cardiac cell image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 265 ms. The time for PCA sampling is 437 ms. (c) The result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 140 ms (d) Result of the image-based ODT method. The time for ODT smoothing is 641 ms (e) Result of the image-based CVT method. The time of the CVT smoothing is 562 ms……29

3.1 The weighted graph based on Super-Pixels (2D)………………………………………..35

3.2 Example of S-t graph. Edges drawn by full lines are n-links and edges drawn by dotted lines are t-links…………………………………………………………………………...36

3.3 Example of Search trees S and T. P are passive nodes and A are active nodes. The blank nodes are free nodes……………………………………………………………………...39

3.4 Result of image segmentation on a 128*128 heart image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) The result of super-pixel based image segmentation. The running time for segmentation is 15 ms (d) The result of pixel based image segmentation. The running time for segmentation is 46 ms……………………………………………….43

3.5 Result of image segmentation on a 256*256 heart image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) The result of super-pixel based image segmentation. The running time for segmentation is 16 ms (d) The result of pixel based image segmentation. The running time for segmentation is 203 ms……………………………………………..45

3.6 Result of image segmentation on a 256*256 knee image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) The result of super-pixel based image segmentation. The running time for segmentation is 16 ms (d) The result of pixel based image segmentation. The running time for segmentation is 170 ms……………………………………………….47

3.7 Result of image segmentation on a 1032*912 cell image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) Result image of super-pixel based image segmentation. The running time for segmentation is 250 ms (d) Result image of pixel based image segmentation. The running time for segmentation is 1980 ms……………………………………………49

4.1 (a) The Iso-surface of a 128*128*128 thoracic cavity 3D volume. (b) The slice image of thoracic cavity 3D volume. (c) The slice image of the result of 3D Canny edge detector. Blue rectangular region is the curved-boundary or tiny features. Red rectangular region is the straight-boundary or big features…………………………………………………55

4.2 A slice image of the 3D thoracic cavity volume with the adaptively nodes which are marked in red including both Canny sample nodes and surfaces nodes……56

4.3 (a) Iso-surface of a 128*128*128 sphere 3D volume. (b) The adaptive nodes including both Canny sample nodes and surfaces nodes…………………………………57

4.4 (a) A slice of 128*128*128 thoracic cavity volume mesh generation via Delaunay tetrahedralization. Nodes number= 51625, tetrahedrons number=324577. (b) A closeup look at the rectangular region in (a)…………………………………………………58

4.5 (a) Iso-surface of a 128*128*128 sphere 3D volume. (b) The result of classical voxel-based 3D image segmentation. The running time is 3640. (c) The result of the supervoxel-based 3D image segmentation. The running time of 3D Canny edge detector is 1438ms, The running time of PCA sampling is 62ms. The running time of the generation of the tetrahedral mesh is 78ms. Running time of S-t cut based on the super-voxels is 16ms. Running time of Laplacian smoothing is 78ms. (d) A closeup look at the rectangular region in (c). (e)The clipping result of volume mesh structure. (f) A closeup look at the rectangular region in (e). …………………………………………………………………………..60

4.6 is a 136*121*71 very noisy cell 3D volume. (a) and (b) Two slices of the cross sections of this 3D volume. (c) The iso-surface of this 3D cell volume. (d) Slice of the Canny map.(e) The final picked nodes result. (f) The result of classical voxel-based 3D image segmentation. The running time is 24078ms. (g)The result of the supervoxel-based 3D image segmentation. The running time of 3D Canny edge detector is 500ms, The running time of PCA sampling is 985ms. The running time of the generation of the tetrahedral mesh is 703ms. The running time of S-t cut based on the super-voxel is 594ms. Running time of Laplacian smoothing is 31ms (h) A closeup look at the rectangular region in (g). (i)The clipping result of volume mesh structure. (j) A closeup look at the rectangular region in (i)………………………………………………… ………………………61

4.7 A 128*128*128 thoracic cavity 3D volume. (a) is the iso-surface map of this 3D volume. (b) The final picked nodes result. (c) The result of classical voxel-based 3D image segmentation. The running time is 14422ms. (d)The result of the supervoxel-based 3D image segmentation. The running time of 3D Canny edge detector is 828ms. The running time of PCA sampling is 1422ms. The running time for the generation of the tetrahedral mesh is 1203ms. S-t cut based on the super-voxel is 1031ms. Running time of Laplacian smoothing is 78ms. (e) A closeup look at the rectangular region in (d). (f)The clipping result of volume mesh structure. (g) A closeup look at the rectangular region in (f)……64

5.1 Initial seeds selection on an electron microscopic image of cardiac cells (image size: 177*180 pixels)…………………………………………………………………………..71

5.2 (a) Marching distance map from the user-picked object seeds. (b) Marching distance map from the user-picked background seeds. The seeds are shown in Figure 1. The blue color

means smaller marching distances and by contrast the red color indicates larger marching distances……………………………………………………………………………….72
5.3 The generation of source (in red, representing object) and sink (in blue, representing background) points by using random number generation………………………………74
5.4 The shortest path between one of the source (object) points and one of the sink (background) points. Note the path favors going through the boundary of the object due to the small edge weights on the boundary…………………………………………………75
5.5 Shortest paths between one of the source points (representing the object) and all the sink points (representing the background)……………………………………………………..75
5.6 The edge confidence (EC) map formed by adding one to pixels where a shortest path goes through. ………………………………………………………………………………….76
5.7 Final segmentation result by using the randomized shortest paths, an algorithm based on the Monte Carlo method…………………………………………………………………77
5.8 Illustration of our algorithm on a cardiac cell with 264*235 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The computed edge confidence image by adding all shortest paths together. (f) The final segmentation result…………………………………………………………78
5.9 Illustration of our algorithm on a cardiac cell with 125*145 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The computed edge confidence image by adding all shortest paths together. (f) The final segmentation result…………………………………………………………80
5.10 Illustration of our algorithm on an MRI image of the heart with 128*128 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The final segmentation result of the left ventricle………....82

LIST OF TABLES

2.1 Running time of our mesh generation method on a 128*128 heart image………………..23

2.2 Running time of our mesh generation method on a 256* 256 knee image……………….26

2.3 Running time of our mesh generation method on a 256*256 cardiac cell image…………28

2.4 Running time of our mesh generation method on a 1032*912 cardiac cell image……….33

3.1 Running time comparison between our supervoxel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 128*128 heart image……………….44

3.2 Running time comparison between our supervoxel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 256*256 cell image…………………46

3.3 Running time comparison between our supervoxel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 256*256 knee image………………..48

3.4 Running time comparison between our supervoxel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 1032*912 knee image……………….51

4.1 Running time comparison between our supervoxel-based 3D image segmentation with the classical voxel-based 3D image segmentation…………………………………………66

ACKNOWLEGEMENTS

I am utilizing this opportunity to express my gratitude to everyone who has supported me pursuing my Ph.D. degree in Computer Science. I am thankful for their aspiring guidance, invaluably constructive criticism and intelligent advice in my scientific career.

First, I was grateful to my advisor Prof. Zeyun Yu for recruiting me as one of his Ph.D. students and I was honored to join his biomedical modeling and visualization laboratory. Especially, I thank for his continuous support of my Ph.D. study and related research, for his patience, motivation and immense knowledge. His guidance helped me in all the time of research and writing of this dissertation.

Additionally, I would like to express my genuinely gratitude to Prof. Ichiro Suzuki as well. It was glad being his teaching assistant and assisting his computer graphics class in the past four years. His conscientious attitude to science and teaching inspires my scrupulous attention to pursue scientific paradigm.

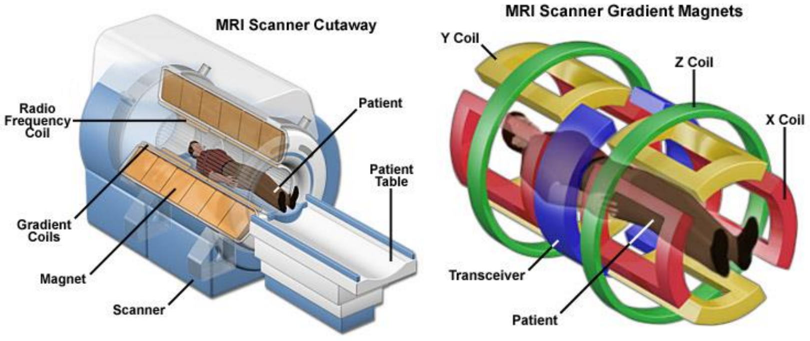
Besides, my earnest thanks also go to the rest of my dissertation committee: Prof. Dexuan Xie, Prof. Jun Zhang and Prof. Roshan D'Souza for their insightful comments and encouragement, but also for the hard questions which incented me to widen my research from various perspectives.

Last but not least, I would like to greatly appreciate my parents' and my girlfriend Yu Lin's spiritually supporting throughout writing my Ph.D. dissertation and in my life in general. Also, Yu Lin's creative and humorous attitude to life and insightful critique impress me that a Ph.D. should not live off the grid and always dig into academic science. There are many brand-new novelties in the world awaiting me to make bold attempts as well.

Chapter 1

Introduction:

Imaging technologies have been widely utilized in many aspects of science and engineering fields. Some popular examples in biomedical applications are computerized tomography (CT), magnetic resonance imaging (MRI), ultrasound (US), single photon emission computed tomography (SPECT), positron emission tomography (PET), and light/electron microscopy (LM/EM) [1, 2]. Figure 1.1 shows the diagrammatic sketch of MRI scanner system.



(a) (b)

Fig 1.1: (a) MRI scanner system cutaway. (b) MRI scanner gradient magnets. (a) and (b) are from national high magnetic field laboratory. http://www.magnet.fsu.edu

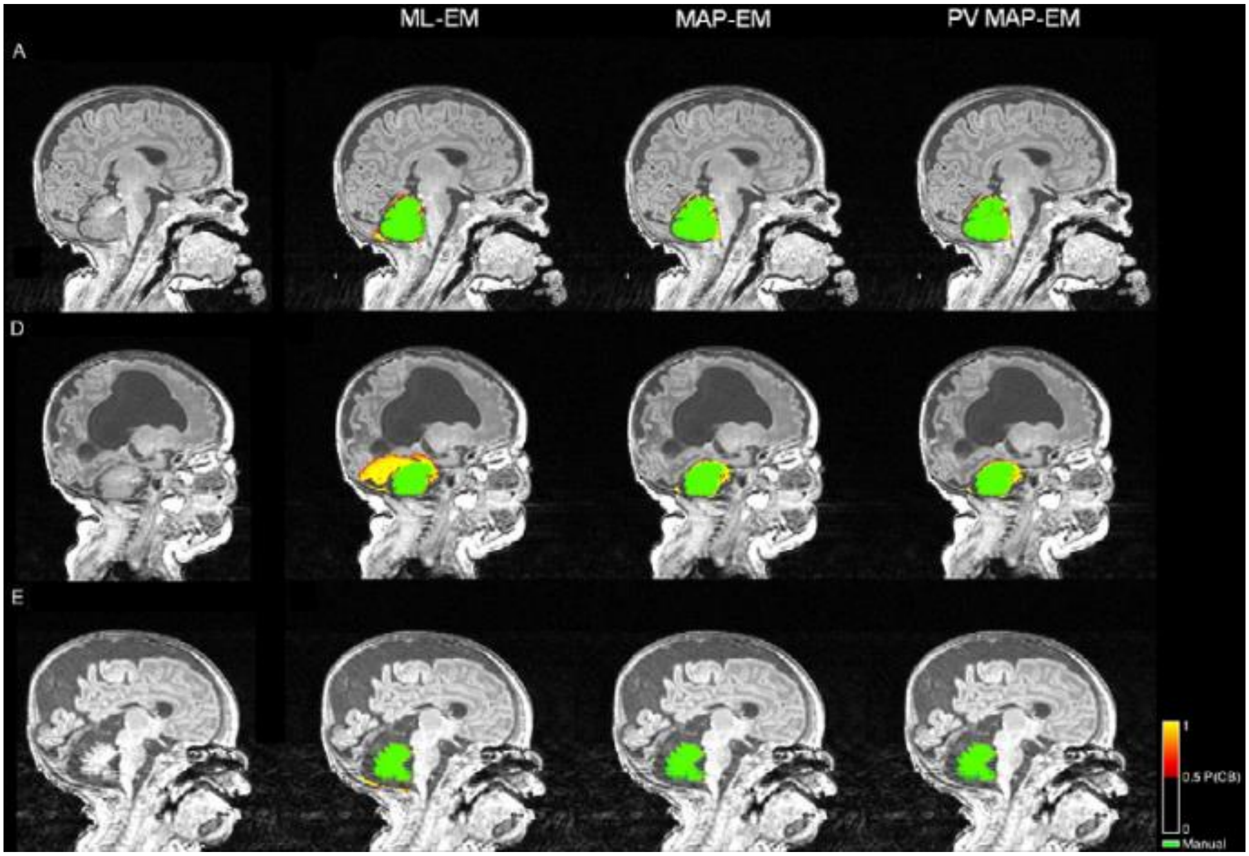


Fig 1.2: Biomedical image segmentation. www.researchgate.net

However, there are some restrictions of medical image segmentation and mesh generation techniques. First, medical imaging data are usually quite noisy and features are often ambiguously presented in images due to the limitations of either imaging devices or reconstruction algorithms. Second, for purpose of doing quick diagnostic analysis for the patients with the 2D or 3D visualization models, finding real-time mesh generation and image segmentation schemes have become progressively vital.

In this chapter, several popular methods of image segmentation and mesh generation will be presented.

1.1 Image Segmentation

Image segmentation as a fundamental problem in image processing and computer vision area which trying to separate objects of interest from surrounding background have received tremendous attention from researchers. Many other tasks, such as shape modeling and feature recognition, rely largely on correct segmentation result. Image segmentation is to partition the pixels or voxels into several clusters. The pixels or the voxels which are in the same cluster share the similar characteristics. With the increasing utilization of imaging technologies in the medical field, it is not uncommon to see a single image or a volume exceeding the size of one Gigabyte or more. The growing availability of high-resolution images has posed challenges on data storage and transmission. The traditional image segmentation methods are based on the pixels. Because the number of the pixels is usually extremely big, the speed of pixel-based image segmentation is typically slow.

Commonly used methods include segmentation based on edge detection [5], region growing and/or region merging [6], active curve/surface motion [7], watershed immersion method [8], and eigenvector analysis [10]. Among the existing computational approaches for segmentation, the deformable contour has drawn a lot of attention in recent years. There are two ways of representing deformable contours: one is parametric (i.e., the well-known snake model) [11] and the other is geometric (i.e., the level set method) [12]. The drawback of the parametric approach is that any topological change of contours would heavily complicate the procedures and data structures being used. By contrast, the level set method can handle topological changes naturally but is computationally more expensive due to the added dimension in the technique. In [6], a variant of the level set method was described to segment features in an image. This method, called the multi-seeded fast marching method, is used in conjunction with seed classification and region growing and merging [6, 13].

Graph-based methods are also very popular in image segmentation field. In these methods, an image is treated as a graph, where each pixel is a node and an edge is created between two adjacent pixels with 4- or 8-connectivity. The first method is based on graph cuts [14, 15, 16], where the goal is to partition the graph into several connected components with certain optimization criteria and each of them corresponds to one segmented feature (or background). This method works in a way similar to data classification – the intra-class difference is minimized while the inter-class difference is maximized. The original graph cut method is computationally expensive because the graph size generated from the original image may be huge, especially for 3D images. To remedy this problem, the concept of "super-pixels" has been introduced to group a set of local, coherent pixels sharing similar texture or brightness into a region (or super-pixel) [6, 23, 24]. It has been estimated that using super-pixels can tremendously reduce both time and memory consumptions [25]. The second graph-based method is the random walker method [17]. This approach also treats the image as a weighted graph but assigns each pixel to a seed (belonging to an object or background) by considering the most likely random walker between the pixel and the seed found. The third widely used graph-based method for image segmentation is the shortest path approach [18]. This approach assigns a pixel to the object if the path from the pixel to the object seed is shorter than paths to the background seed.

1.2 2D and 3D Mesh Generation

Computer visualization and simulations help researchers and doctors to realize the various activities in biomedical systems, ranging from molecular, cellular to organ scales [51, 52, 53].

Many of these biological activities can be formulated as partial differential equations (PDEs), which often do not admit analytical solutions and have to be numerically solved. Among a variety of numerical approaches, the finite element method (FEM) has become popular for solving PDEs. One of the challenges in using the FEM, however, is to generate high-quality meshes on which the simulations are performed. In order to obtain more accurate simulation results, it is also practically important to incorporate realistic biological structures into geometric mesh models [54, 55, 56, 57, 58, 59].

The classic routine to generate the surface or volumetric meshes from given images typically have two steps: (1) from images to surface models. (2) from surface models to triangular meshes or tetrahedral meshes. First step is unknown as image segmentation which is introduced in above section.

From surface models to triangular meshes or tetrahedral meshes which is step (2) typically performed in two steps: nodes generation and Delaunay triangulation of the nodes. Different mesh generation methods are often different from each other in node generation and can be classified into three categories: (1) content-adaptive node generation with various types of local feature measurements [26, 27, 28, 29, 30], (2) greedy (iterative) node insertion starting from a coarse mesh of an image [31, 32, 33, 34, 35], and (3) greedy (iterative) node removal starting from a dense mesh containing all pixels of an image [32, 36, 37, 38]. Most of the approaches mentioned above, however, do not take mesh quality into serious consideration except the method described in [29], where some mesh post-processing strategies are carried out by adding new mesh nodes. It is known that mesh quality can be efficiently improved by moving mesh nodes without adding additional ones [18, 19, 20, 21]. This strategy had been recently used in [22] for image triangulation by optimizing an objective function that incorporates both image intensities and mesh quality. There are two major drawbacks of this approach: low speed due to a numerical scheme adopted to minimize the energy function, and high approximation errors because of the assumption of a constant intensity in each triangle. The Delaunay-based method is very successful in 2D mesh generation but sometimes may not work well in generating high-quality tetrahedral meshes, where some nearly degenerate elements known as slivers often occur [102]. The octree-based method recursively subdivides the domain containing the given mesh until some stopping criteria are reached [69, 70, 71, 72]. In all the above approaches, however, the dihedral angles are not guaranteed in the resulting tetrahedral especially those near the surfaces. More recently, Labelle and Shewchuk [73] presented an iso-Variational tetrahedral meshing, surface stuffing algorithm to generate tetrahedral meshes.

1.3 Objectives and Contributions

How to extract the boundary of features and generate the high-quality mesh from a given 2D or 3D images with a real-time performance has become very popular in image processing. The number of pixels in 2D images or voxels in 3D volumes is typically very huge and image segmentation algorithms are normally global optimization based. Therefore, the speed of image segmentation and mesh generation would be very slow.

The main contributions are what we treat the triangles or tetrahedrons of the feature-sensitive and adaptive mesh on given 2D or 3D images as super-pixels. Our feature-sensitive and adaptive image triangulation is a super-pixel-based scheme for image segmentation and mesh generation to help significantly reducing the time complexity. The output of this scheme is an object of interest with a high-quality mesh.

First, we create an adaptive and feature-sensitive mesh on the overall given 2D or 3D images instead of implementing the image segmentation as the first step in the classic pipeline. During this step, a new image edge enhancement and segmentation via randomized shortest paths method is introduced to be an optional choice of boundary extraction to replace Canny edge detector. The principal components analysis sampling strategy is utilized to calculate the sampling radius so that to make sure the tiny features are captured by small mesh elements and big features are represented by large (sparse) elements.

Second, the generated triangulation (or tetrahedralization) provides an adaptive partitioning of a given image (or volume). Each cluster of pixels within a triangle (or voxels within a tetrahedron) is called a super-pixel. Moreover, a popular graph-cut method is then applied to the graph to extract image features which result in a good boundary segmentation with high quality mesh. Thanks to the significantly reduced number of elements (super-pixels) as compared to that of pixels in an image, the super-pixel based segmentation method has tremendously improved the segmentation speed, making it feasible for real-time feature detection from an image of a decent size.

Finally, the last contribute is that I was chosen to be one of the members to develop toolkit in our lab which names “Bimos” for fast visualization and interpretation on 2D and 3D images and meshes. In this thesis, most of the 2D and 3D meshes are displayed by Bimos.

In chapter 2, I will present a series of algorithms to generate high quality, feature-sensitive, and adaptive meshes from a given grayscale image. Furthermore, this thesis will display some experimental results after the algorithm introduction. In chapter 3, a popular graph cut S-t cut is adopted to cut the graph into two parts, including sink and source which represent object and background. As the chapter 2, a series of the experiment results are shown for comparing the speed of pixel-based segmentation with super-pixel based segmentation in chapter 3 as well. In chapter 4, a procedure of 3D feature-sensitive and adaptive mesh generation based on image segmentation will be discussed. In chapter 5, an image edge enhancement and segmentation via randomized shortest paths method will be presented.

Chapter 2

Feature-Sensitive and Adaptive Mesh Generation on 2D Images

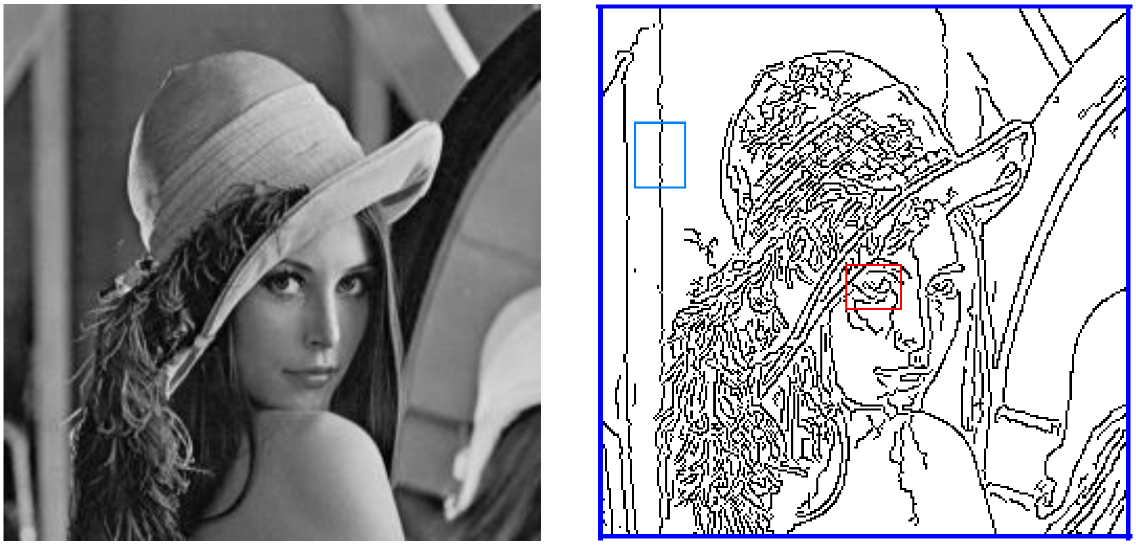
In this chapter, an adaptive and feature-sensitive meshes will be generated to pave on the whole 2D images. The proposed method includes three main steps. Firstly, we generate three kinds of mesh nodes: (1) Canny's points, (2) halftoning points, and (3) uniform points. In the second step, an initial triangular mesh is generated from these points by using the popular open source mesh generators. Finally, the initial mesh is smoothed by modified versions of the conventional centroid Voronoi triangulation (CVT) and optimal Delaunay triangulation (ODT) smoothing methods.

2.1 Node Generation

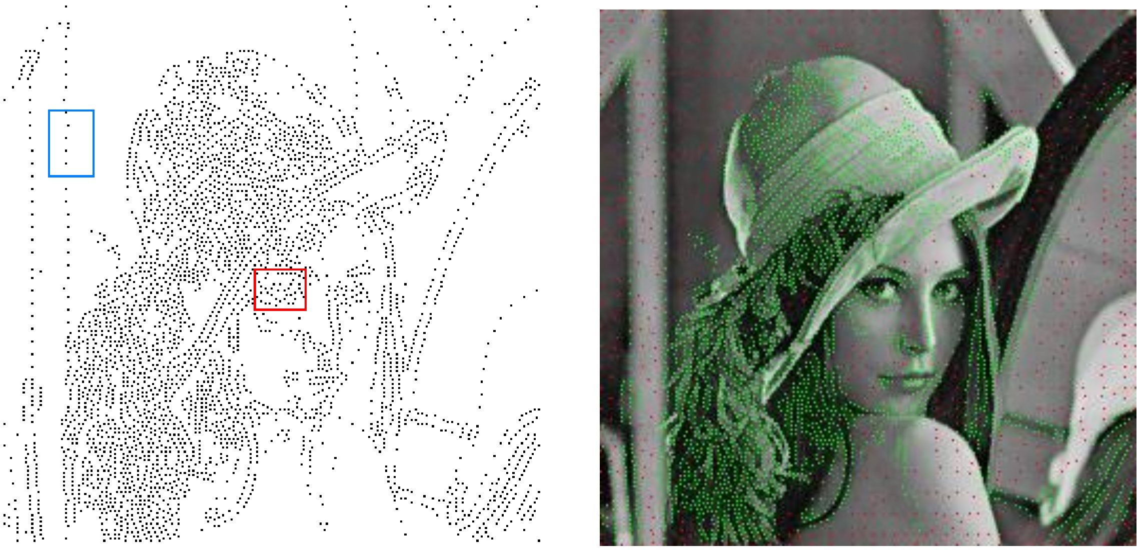
In order to have adaptive and feature-sensitive meshes for given images, we need to generate initial mesh nodes. We would like to generate dense mesh nodes distribution on or near curved-boundary or tiny features and sparse mesh nodes distribution on straight-boundary features or background.

2.1.1 Canny Sample Points

Image edges are important features in an image and need to be preserved in the obtained meshes. Canny edge detector is a well-known method to extract feature boundaries. Figure 2.1(b) displays the result of the Canny edge detector running on an image of 192*256 pixels shown in Figure 2.1(a). While the Canny edge detector can represent the image edges quite faithfully, the edge points are too dense to generate triangular meshes. Our strategy is to implement an adaptive sampling on the Canny edge points. We note that an image usually contains features with high curvature and features with relatively straight edges. An example of curved-boundary and tiny features is displayed in red on Figure 2.1(b). The blue rectangle on the Figure 2.1(b) shows an example of straight-boundary features. Our interest is to generate dense sample points on curved-boundary features and sparse sample points on straight-boundary features.



(a) (b)



(c) (d)

Fig.2.1: (a) The original image of 192*256 pixels. (b) The result image of the canny edge detector, where σ = 0.7, lower threshold =0.1, and higher threshold = 0.6. The region in red shows an example of curved-boundary or tiny features. The region in blue is an example of straight-boundary features. (c) The adaptively sampled points from (b). The red and blue rectangles show the results of curved-boundary and straight-boundary features respectively. (d) The final result of adaptive node generation. The green points are the canny edge points and the red points are the halftoning and uniform sample points.

In our method, we take the curvature information of every Canny's edge point into account and use the principal component analysis (PCA) to determine the sampling density. The PCA method can detect the overall attribute of the neighbors of a certain size by a statistical way, and this method can be easily extended to three dimensional images. We traverse every edge point in Canny map, denoted as $\overline{\mathrm{P}}$. Let all the edge points within $\overline{\mathrm{P}}$'s K×K neighborhood be $\mathrm{P}_1,\mathrm{P}_2,\ldots,\mathrm{P}_n$, where n is the number of neighboring edge points of $\overline{\mathrm{P}}$ in its neighborhood. K is a parameter based on the density of the features in the original image, and is locally calculated for each edge point in the method described below. The covariance matrix is first calculated in the following expression:

$$\sum_{j=1}^{n}(P_j-\overline{P})(P_j-\overline{P})^T \in \mathrm{R}^{2\times 2} \qquad (1)$$

Then we calculate the two eigenvalues of this covariance matrix, denoted by λ_1, λ_2 (assuming $\lambda_1 > \lambda_2$). We decide the sample radius which we call as $R(\overline{P})$ using the following equation:

$$R\left(\overline{P}\right)=\begin{cases}1 & \left(\lambda_1-\lambda_2\right)/\left(\lambda_1+\lambda_2\right)\leq 0.3\\ 3 & 0.3<\left(\lambda_1-\lambda_2\right)/\left(\lambda_1+\lambda_2\right)\leq 0.5\\ 5 & else\end{cases} \tag{2}$$

The current Canny edge point P survives (i.e., being a valid sampling point) if and only if there is no other existing sampling point found within the neighborhood of size K×K, where $K=2\times R(\overline{P})+1$. Figure 2.1(c) shows the sampled points with the PCA strategy. As we have discussed, tiny features and features with high curvature have dense sample points (see the red rectangle for example). On the other hand, big features or features with straight lines have sparsely sampled points (see the blue rectangle for example).

2.1.2 Halftoning Sample Points

The edge points described above only capture the pixels on or near the image edges. In order to have a decent initial mesh, one has to scatter some more points in the non-edge regions of the image. To this aim, we adopt the halftoning sample points based on the approach described in [44]. This method generates the sample points based on the second derivatives of an image, where most of the sample points found are placed near the image features (edges). Below we give a brief summary of this approach. The interested reader can refer to [44] for more details.

The first step of this method is to extract the image feature map by calculating the Hessian matrix described in the following equation [44]:

$$H(x,y)\begin{pmatrix} f_{xx}^{"}(x,y) & f_{xy}^{"}(x,y) \\ f_{xy}^{"}(x,y) & f_{yy}^{"}(x,y) \end{pmatrix} \tag{3}$$

We then calculate the two eigenvalues of the above Hessian matrix:

$$\lambda_{1,2}(x,y)=\frac{1}{2}\left(f_{xx}^{"}(x,y)+f_{yy}^{"}(x,y)\right)\pm\sqrt{\frac{1}{4}\left(f_{xx}^{"}(x,y)-f_{yy}^{"}(x,y)\right)^{2}+\left(f_{xy}^{"}(x,y)\right)^{2}} \tag{4}$$

The image feature map is generated in the following equation (where G(x,y) = $\max\left(\left|\lambda_1(x,y)\right|,\left|\lambda_2(x,y)\right|\right)$):

$$\sigma(x,y)=\left(\frac{G(x,y)}{A}\right)^{\gamma} \tag{5}$$

Where A is the largest value of G(x, y), and γ is a parameter and in this thesis γ = 1.0. The Floyd-Steinberg diffusion algorithm [45] is then used to scan the whole image pixel by pixel and compare each pixel's feature map value with a threshold q as follows[44]:

$$b(x,y)=\begin{cases} 1 & if\,\sigma(x,y)\geq q \\ 0 & otherwise \end{cases} \tag{6}$$

A point (x, y) is chosen as a sample point if b(x, y) = 1. The parameter q determines the number of mesh nodes that can be generated using this method. In our method, q = 0.3. To dynamically update the feature map σ (x, y), this method computes the quantization error e(x, y) by[44]:

$$e(x,y)=\sigma(x,y)-(2q)b(x,y) \tag{7}$$

And the feature map is updated by the diffusion procedure towards the right and down directions[44]:

$$\begin{aligned}
\sigma\left(x, y+1\right) &= \sigma\left(x, y+1\right) + \omega_1 e(x, y) \\
\sigma\left(x+1, y-1\right) &= \sigma\left(x+1, y-1\right) + \omega_2 e(x, y) \\
\sigma\left(x+1, y\right) &= \sigma\left(x+1, y\right) + \omega_3 e(x, y) \\
\sigma\left(x+1, y+1\right) &= \sigma\left(x+1, y+1\right) + \omega_4 e(x, y)
\end{aligned} \tag{8}$$

Where $\omega_1 = 7/16, \omega_2 = 3/16, \omega_3 = 5/16, \omega_4 = 1/16.$

The above updating scheme is applied to every pixel until the bottom of the image is reached. A point (x, y) is said to be a halftoning sample point if the final b(x, y) is non-zero and no Canny's points are found in its 7×7 neighborhood.

2.1.3 Uniform Sample Points

Although the halftoning sample points can cover most non-edge regions of the image, it is possible the no point (either Canny or halftoning) is found in regions of almost constant intensities. We therefore generate some points uniformly to cover the rest of the images where the first two types of sample points are not located. Again, a point (x, y) is said to be a valid uniform sample point if no Canny's or halftoning points are found in its 7×7 neighborhoods. Figure 2.1(d) shows the final result of the sample point generation, where green points are Canny's sample points and red points correspond to the other two types.

2.2 Mesh Generation via Delaunay Triangulation

The sample points found above are used to generate our initial mesh for a given image by using the Delaunay triangulation. In the 2D case, we employed popular open source software [46] in this thesis. Figure 2.2 shows the initial triangle mesh generated by the Delaunay triangulation. As can be seen in Figure 2.2, the quality of the initial triangular mesh is not good enough. Hence a mesh smoothing scheme performing on this initial mesh is necessary, as described below. The smoothed mesh is expected to be not only well shaped and sized but also strictly attached to the features (edges) of the image.

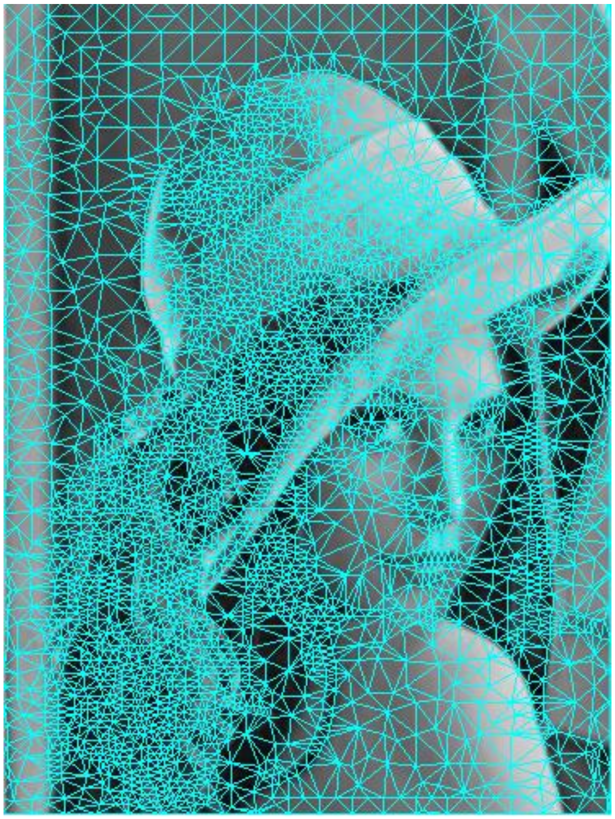


Fig 2.2: Initial triangle mesh generated by Delaunay triangulation.

2.3 Mesh Smoothing

We extend several traditional mesh smoothing methods, including Optimal Delaunay Triangulations (ODT) and Centroid Voronoi Tessellations (CVT) [48, 49], to the image domain, in order to enhance the quality of the mesh. While the traditional mesh smoothing techniques only take the mesh quality into account, we develop two new methods based on the ODT and CVT schemes to incorporate image feature information into mesh quality improvement. Both ODT and CVT are topology-preserving, meaning that they only move mesh nodes without modifying the mesh topology. In the context of image triangulation, we restrict ourselves to moving only halftoning and uniform sample points but keeping Canny's points unchanged because of the feature-preservation purpose.

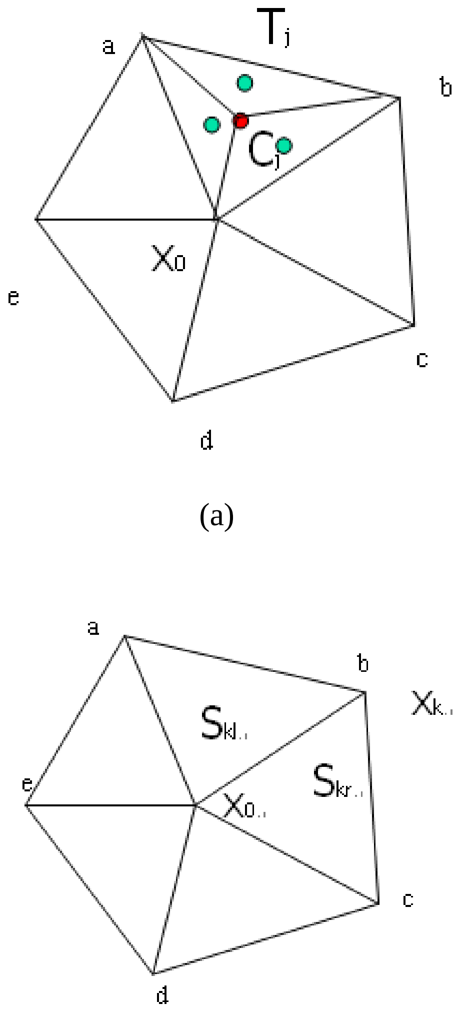
2.3.1 Image-based CVT Scheme

The traditional CVT method traverses every vertex of the triangle mesh, denoted by x_0, and finds its new position as follows:

$$x^* = \frac{1}{|\Omega_0|} \sum_{T_j \in \Omega_0} |T_j| \cdot C_j \tag{9}$$

Where $|\Omega_0|$ is the area of the one-ring neighborhood of x_0, $|T_j|$ is the area of a triangle T_j in the one-ring neighborhood of x_0, and C_j is the centroid of the triangle T_j.

In order to take the image features into account, we add the gradient information of each neighboring triangle of the one-ring neighborhood of x_0 in to the traditional CVT method. Figure 2.3(a) displays how to calculate the gradient information of T_j. First we calculate the centroid of this triangle, called C_j, and its gradient magnitude $g(C_j)$. Secondly, we connect the three vertices of this triangle with C_j yielding three sub-triangles: ΔabC_j, Δbx_0C_j, and ΔC_jx_0a. Thirdly, we calculate every sub-triangle’s centroid and their gradient magnitudes. Then the average of these four gradient values (a red points and three green points in Figure 2.3(a)) is calculated to get the triangle T_j 's gradient information $g(T_j)$.



(a)

(b)

Fig 2.3: (a) The one-ring neighborhood of x0. Tj is one of the adjacent triangles in the neighborhood. Cj is the centroid of the Tj . (b) The one-ring neighborhood of x_0. x_k is one of the neighboring vertices of x_0. S_{kl} and S_{kr} are the left and right triangles of edge x_0, x_k respectively.

The proposed image-based CVT scheme is as follows:

$$x^* = \frac{\sum_{T_j \in \Omega_0} |T_j| \cdot C_j \cdot g(T_j)}{\sum_{T_j \in \Omega_0} |T_j| \cdot g(T_j)} \tag{10}$$

2.3.2 Image-based ODT Scheme

Similar to the CVT scheme, the traditional ODT method is defined based on the one-ring neighborhood of a vertex in the mesh as follows:

$$x^* = x_0 - \frac{1}{2|\Omega_0|} \sum_{x_k \in \Omega_0}^{x_k \neq x_0} (S_{kl} + S_{kr})(x_0 - x_k) \tag{11}$$

Where S_{kl} is the left triangle of edge x_0x_k and S_{kr} is the right triangle of edge x_0x_k (see Figure 2.3(b)).

In the proposed work, we modify the original ODT method by taking the gradient information into account as below:

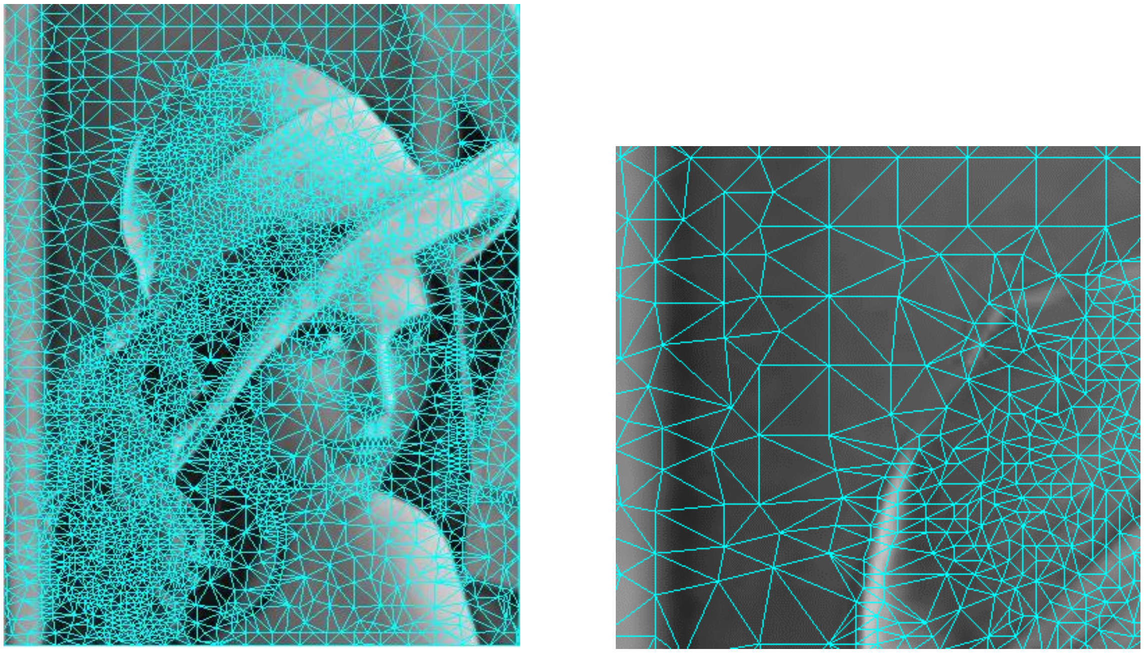
$$x^* = x_0 - \frac{\sum_{x_k \in \Omega_0}^{x_k \neq x_0} (S_{kl} \cdot g(S_{kl}) + S_{kr} \cdot g(S_{kr}))(x_0 - x_k)}{\sum_{x_k \in \Omega_0}^{x_k \neq x_0} (S_{kl} \cdot g(S_{kl}) + S_{kr} \cdot g(S_{kr}))} \tag{12}$$

Where g (S_{kl}) is the averaged gradient magnitude of left triangle of edge x_0x_k and g (S_{kr}) is the averaged gradient magnitude of right triangle of edge x_0x_k.

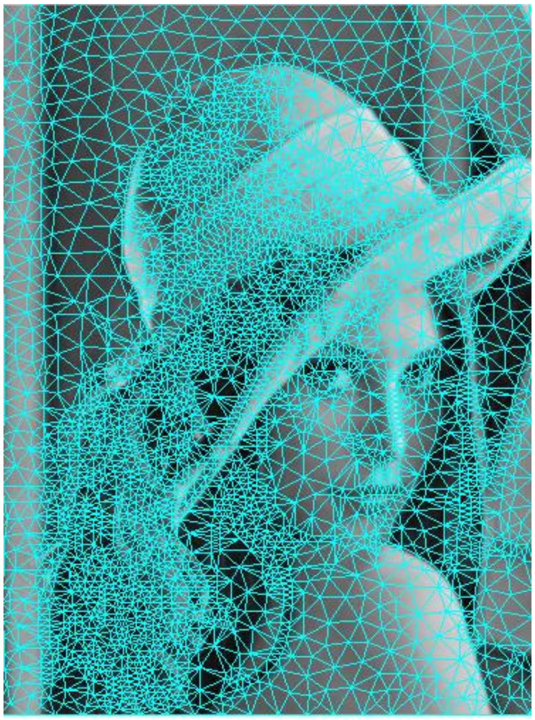
2.3.3 Edge Flipping Strategy

Although the traditional and our modified CVT/ODT methods are intended to move vertices without changing the mesh topology, it is often beneficial to change the mesh topology by using the widely-used edge-flipping technique. In our implementation, we traverse every vertex of the mesh one by one using either image-based CVT or image-based ODT method. The CVT or ODT method and edge-flipping scheme are applied to the mesh alternatively. Typically five iterations of applying ODT/CVT and edge-flipping to the meshes would be sufficient, as demonstrated below in the results.

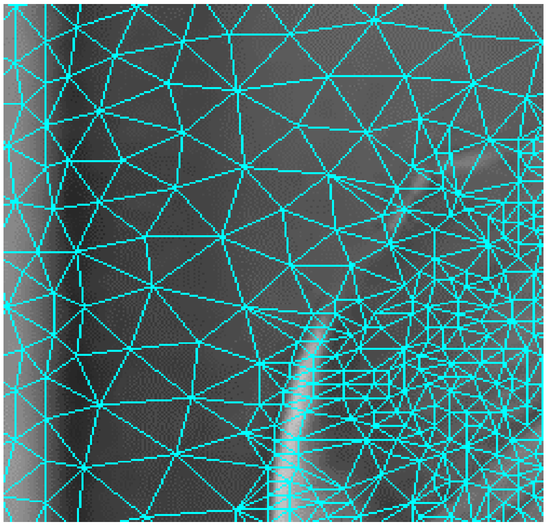
The final results of the image-based CVT and ODT methods are shown in Figure 2.4(c) and (e) respectively, for the initial mesh in Figure 2.4 (a). In both cases, the smoothed meshes have good quality mesh. Moreover, the mesh nodes are better attached along the feature boundary in the image. Also, dense sample points are placed near the image features and sparse sample points are placed in the background or the regions of low-curvature features. From the point of view of the image segmentation, the results retain the fidelity of the boundaries of the objects in Figure 2.4(c) and (e).



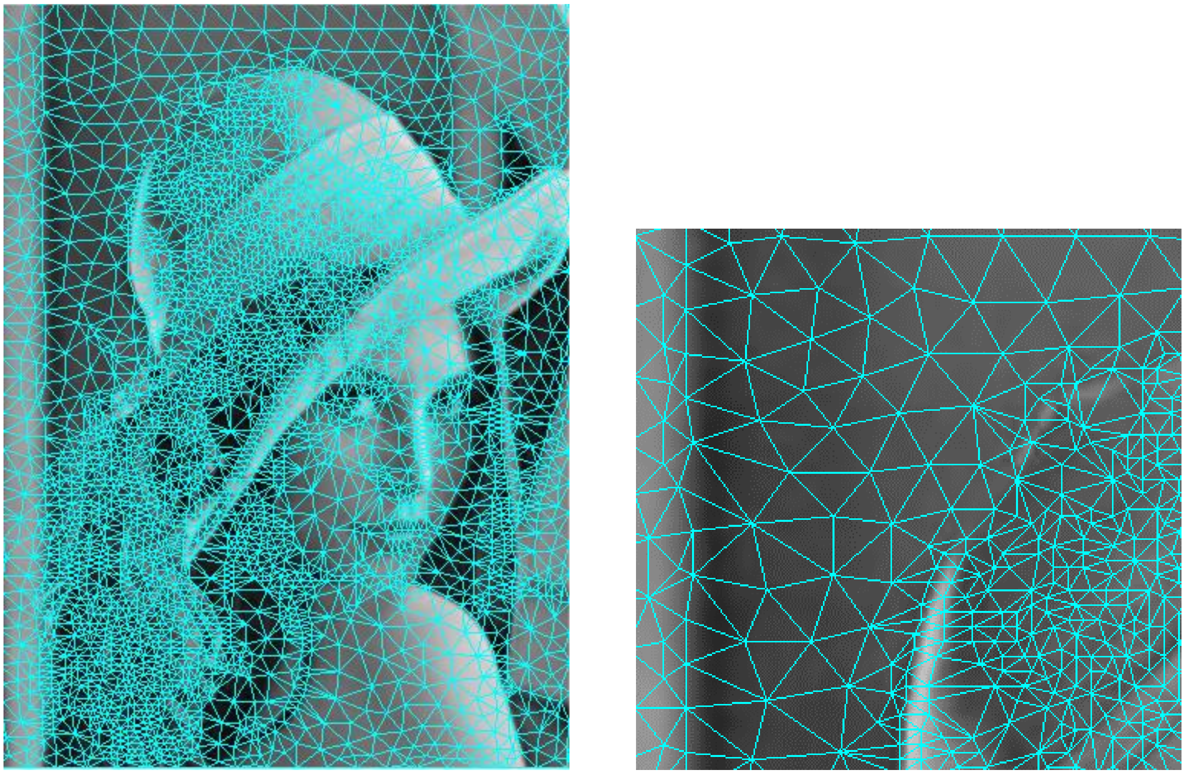
(a) (b)



(c)



(d)

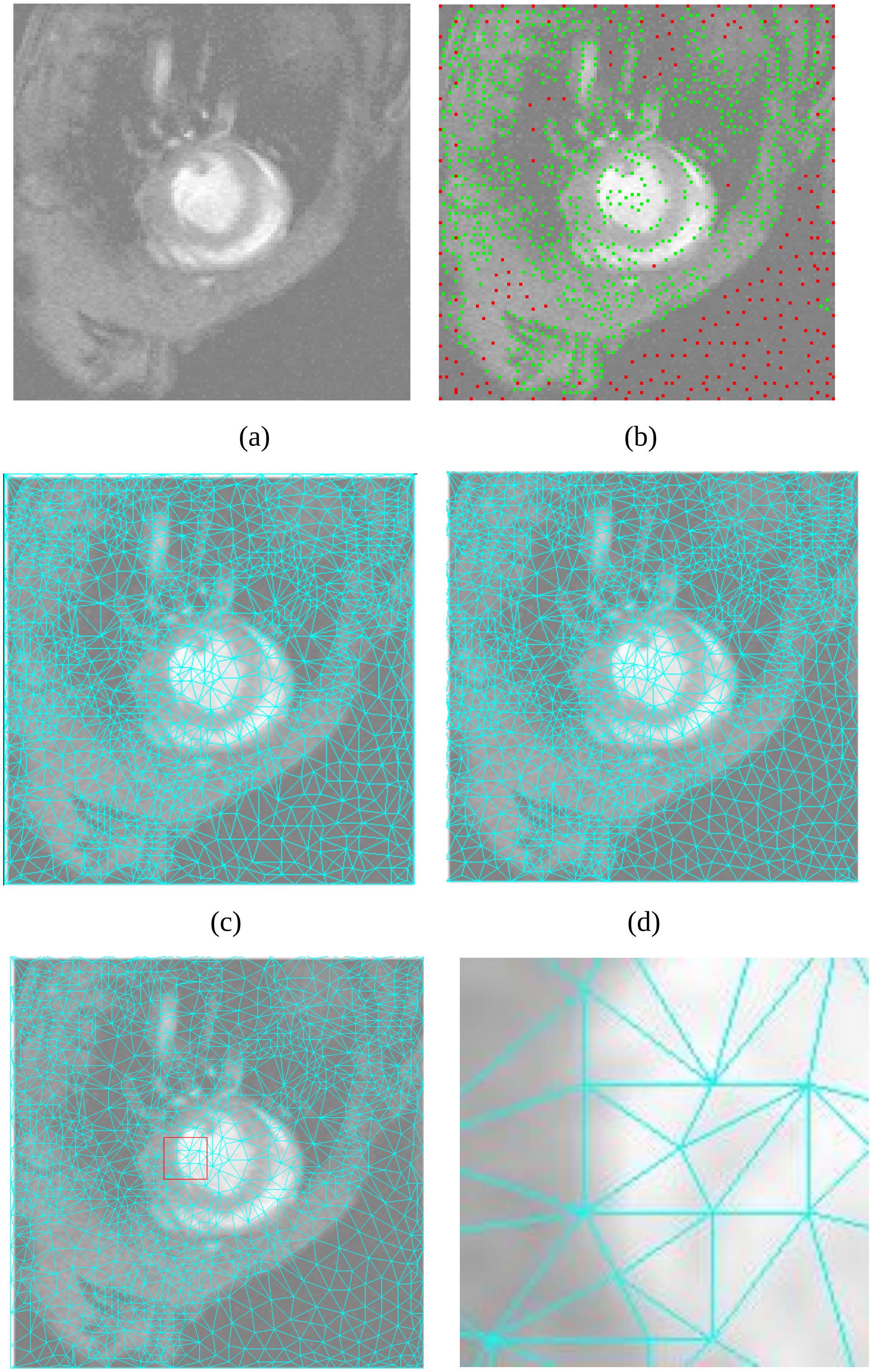


(e) (f)

Fig 2.4: (a) Initial triangle mesh generated by Delaunay triangulation. (b) A closeup look of (a). (c)Smoothed mesh by the CVT method based on the image features after five iterations. (d) A closeup look of (c). (e) Smoothed mesh by the ODT method based on the image features after five iterations. (f) A closeup look of (e).

2.4 Experimental Results

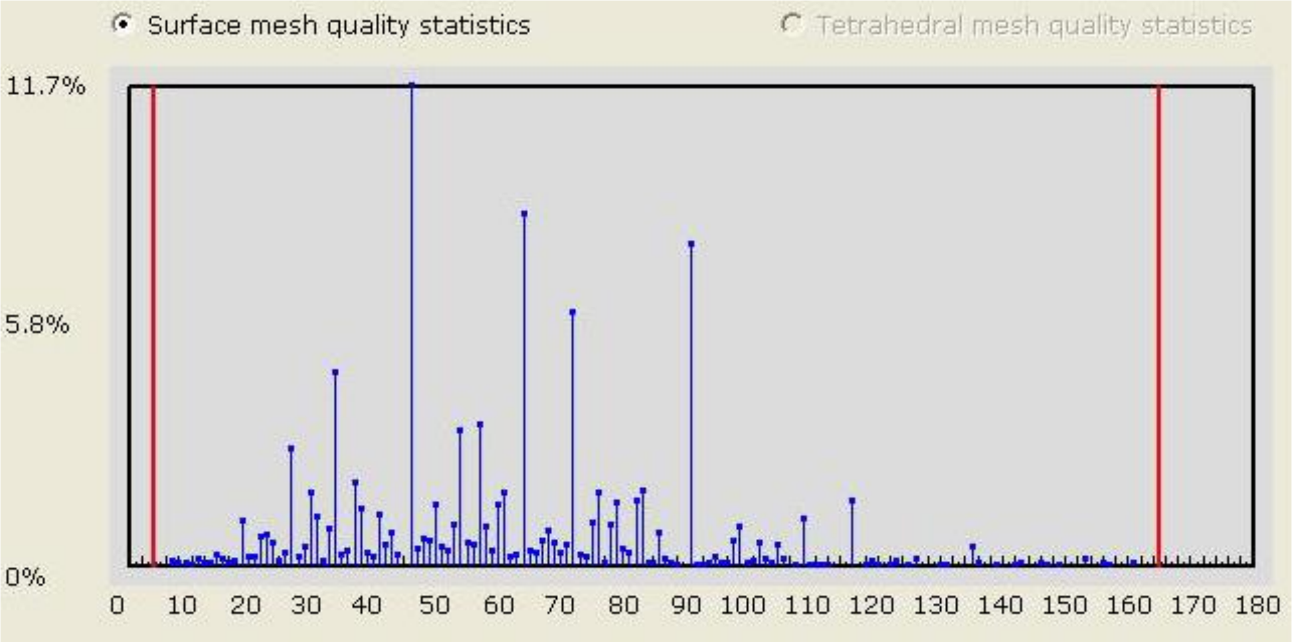
In this section, we provide the results of our method running on some biomedical images. The number of iterations in the mesh smoothing step is set as 5. The code was written in C++ and compiled in Windows Visual Studio 2010.



(a) (b)

(c) (d)

(e) (f)



(g)

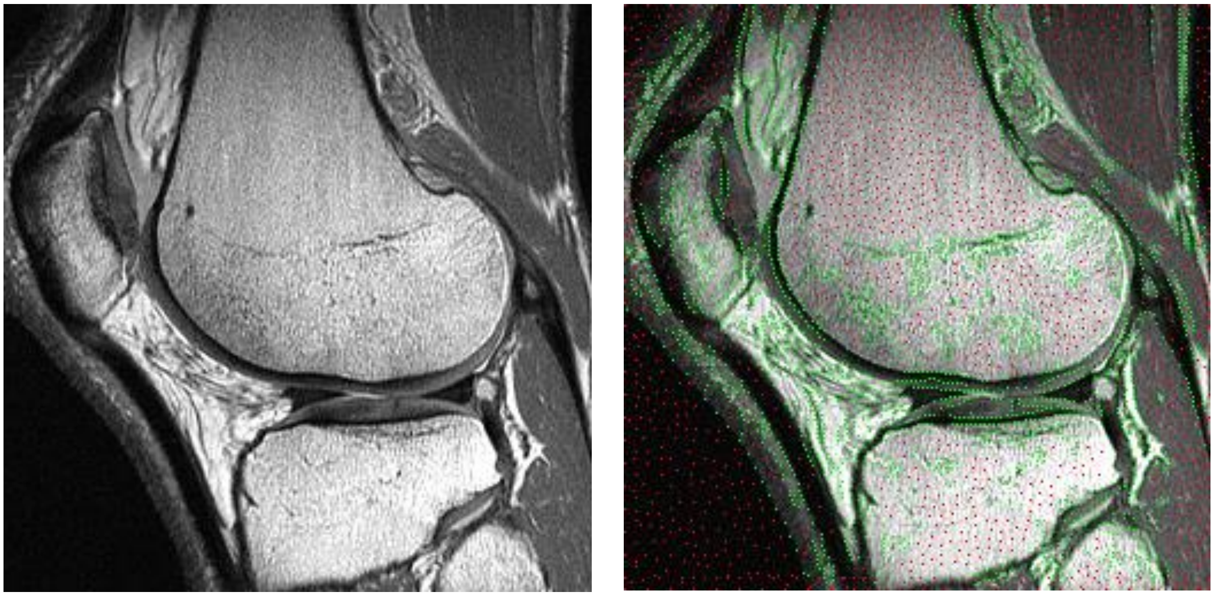
Fig 2.5:Result of our method running on a 128*128 heart image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 15 ms. The time for PCA sampling is 12ms (c) The result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 15 ms (d) Result of the image-based ODT method. The time for ODT smoothing is 16ms (e) Result of the image-based CVT method. The time of the CVT smoothing is 16ms (f) A closeup look at the rectangular region in (e). (g) A quantitative measure of the quality of the final triangulations. The minimum angle is 4.57 degree. The maximum angle is 165.96 degree. The minimum size of the triangles is 0.499. The maximum size of the triangles is 32.99.

Table 2.1: Running time of our mesh generation method on a 128*128 heart image

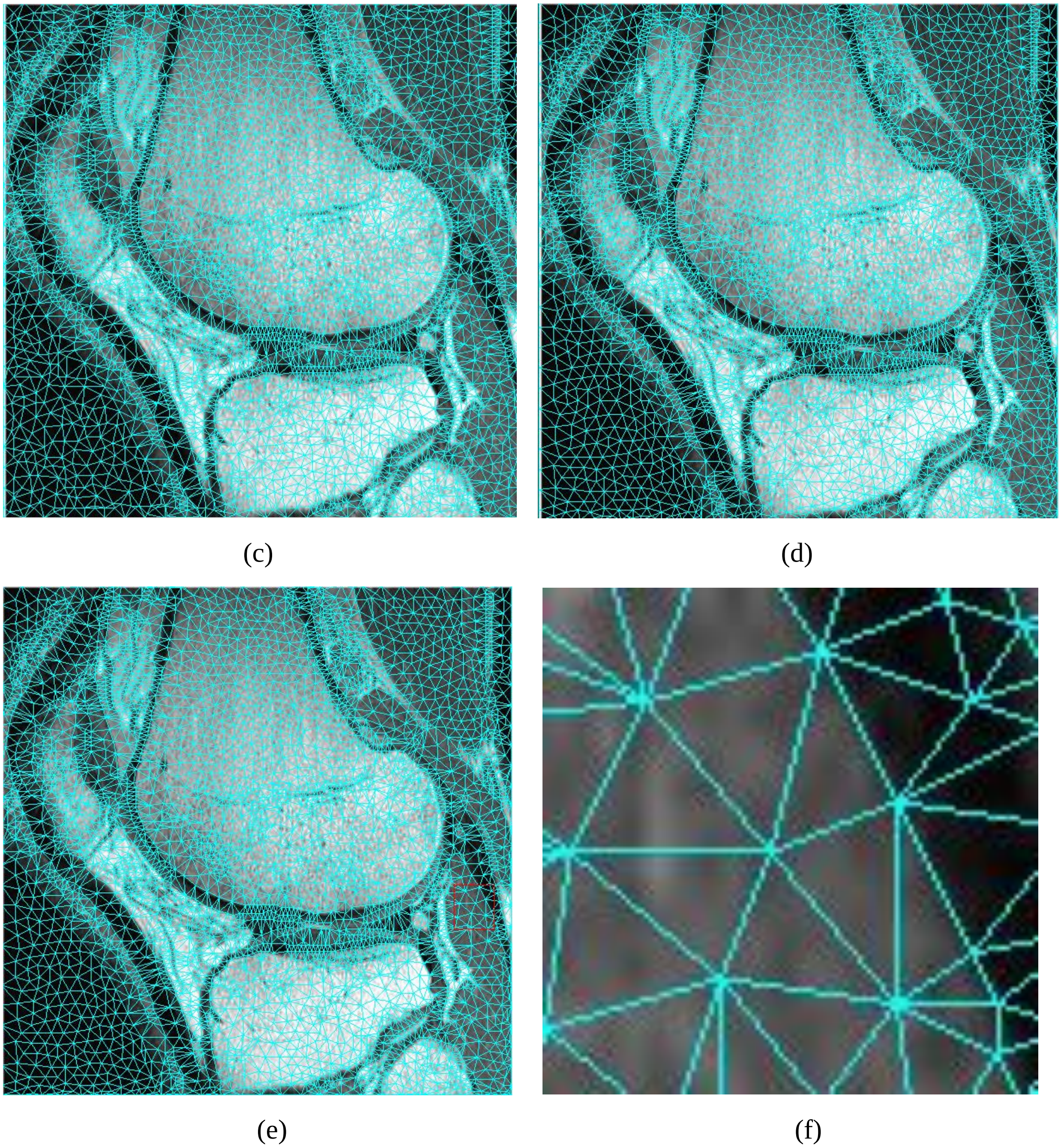
Steps	Running time (ms)
Canny and halftoning nodes	15
PCA sampling	12
Initial mesh generation	15

ODT	16
CVT	16

Figure 2.5 displays the result of our method running on a 128*128 heart image (courtesy of Dr. Andrew McCulloch, UCSD). In (a) we can see that the most important feature in this image is the ventricle in the middle of this image. (b) shows the result of the sample points found. The Canny's sample points (in green) occupy the most area of the left-top of this image. (c) is the result of initial mesh generated by Delaunay triangulation. The mesh smoothing, as shown in (d) and (e), provide better quality of the meshes. Also, the closeup view shows that the triangles are well aligned along the boundary of the ventricle. The time spend in this example listed below the figure shows my method is very fast.



(a) (b)



(c) (d)

(e) (f)

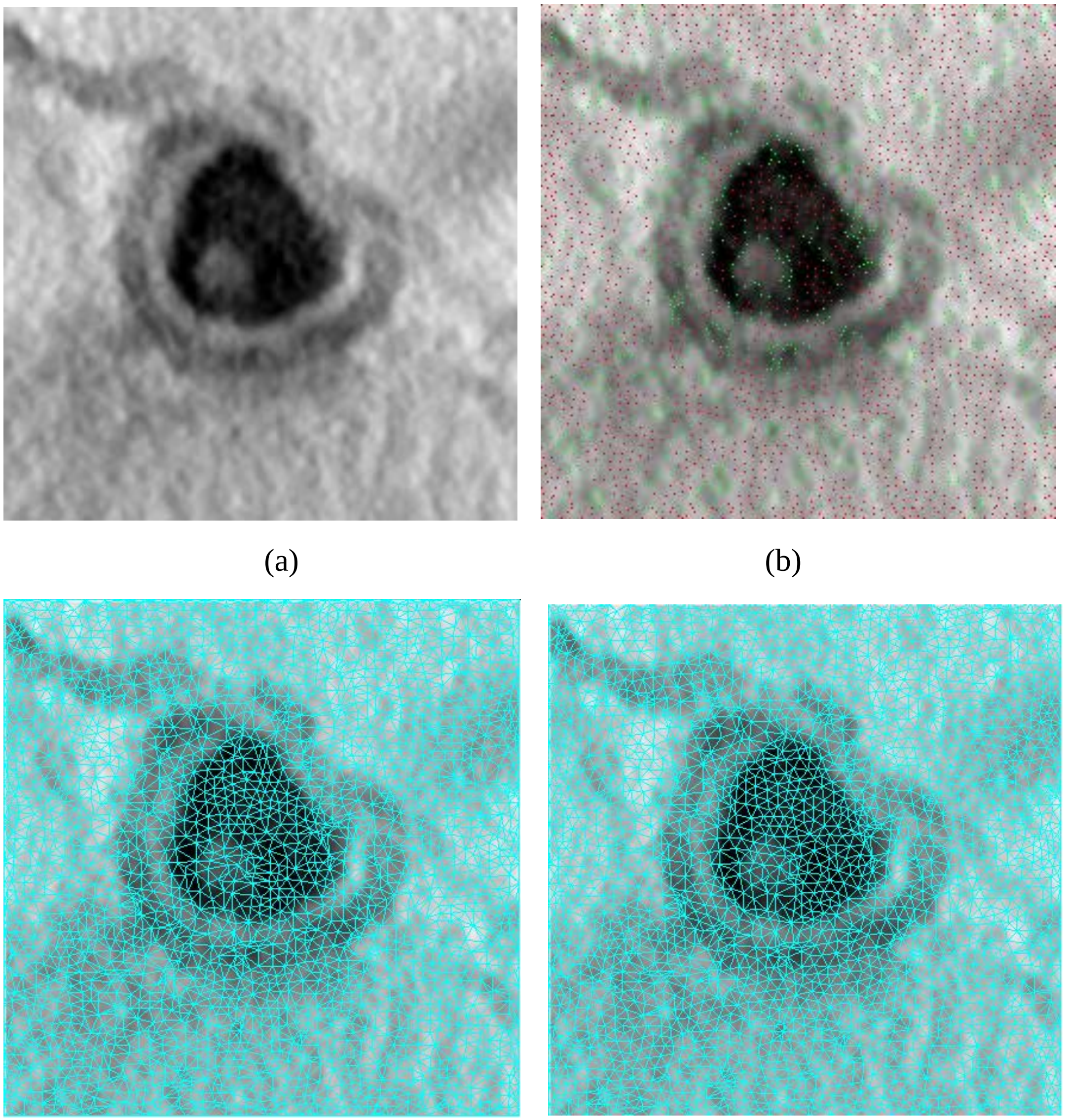
Fig 2.6: Result of our method running on a 256* 256 knee image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 15 ms. The time for PCA sampling is 31 ms. (c) The result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 15 ms. (d) Result mesh of the image-based ODT method. The time for ODT smoothing is 78 ms. (e) Result mesh of the image-based CVT method. The time of the CVT smoothing is 62 ms (f) A closeup look at the rectangular region in (e).

Table 2.2: Running time of our mesh generation method on a 256* 256 knee image.

Steps	Running time (ms)
Canny and halftoning nodes	15
PCA sampling	31
Initial mesh generation	15
ODT	78
CVT	62

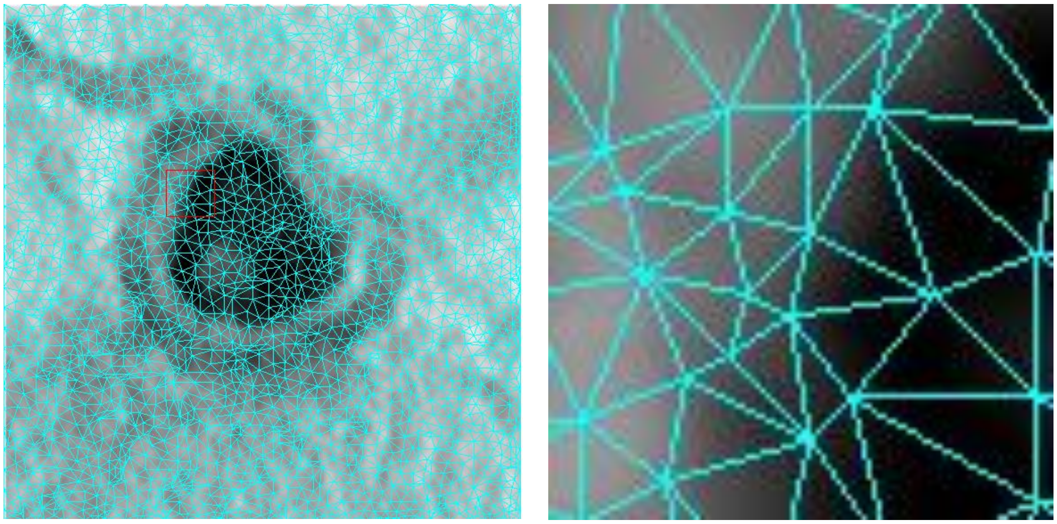
Secondly, we test the proposed method on a 256*256 knee MRI image taken from GE-Healthcare. In Figure 2.6(c), the initial mesh has the same shortcoming as in the initial mesh of the heart image. But this mesh quality is improved by the mesh smoothing, as seen in (d) and (e). A closeup view of the mesh is shown in (f), from which one can see the alignment of the mesh with image features.

Thirdly, in Figure 2.7(a), we show a 256*256 noisy image of partial cardiac cells (courtesy of Dr. Masahiko Hoshijima, UCSD). (b) and (c) show the sample points found and the initial mesh by Delaunay triangulation. The smoothed meshes in (d) and (e) show that the proposed method can generate high quality meshes even for very noisy images.



(a) (b)

(c) (d)



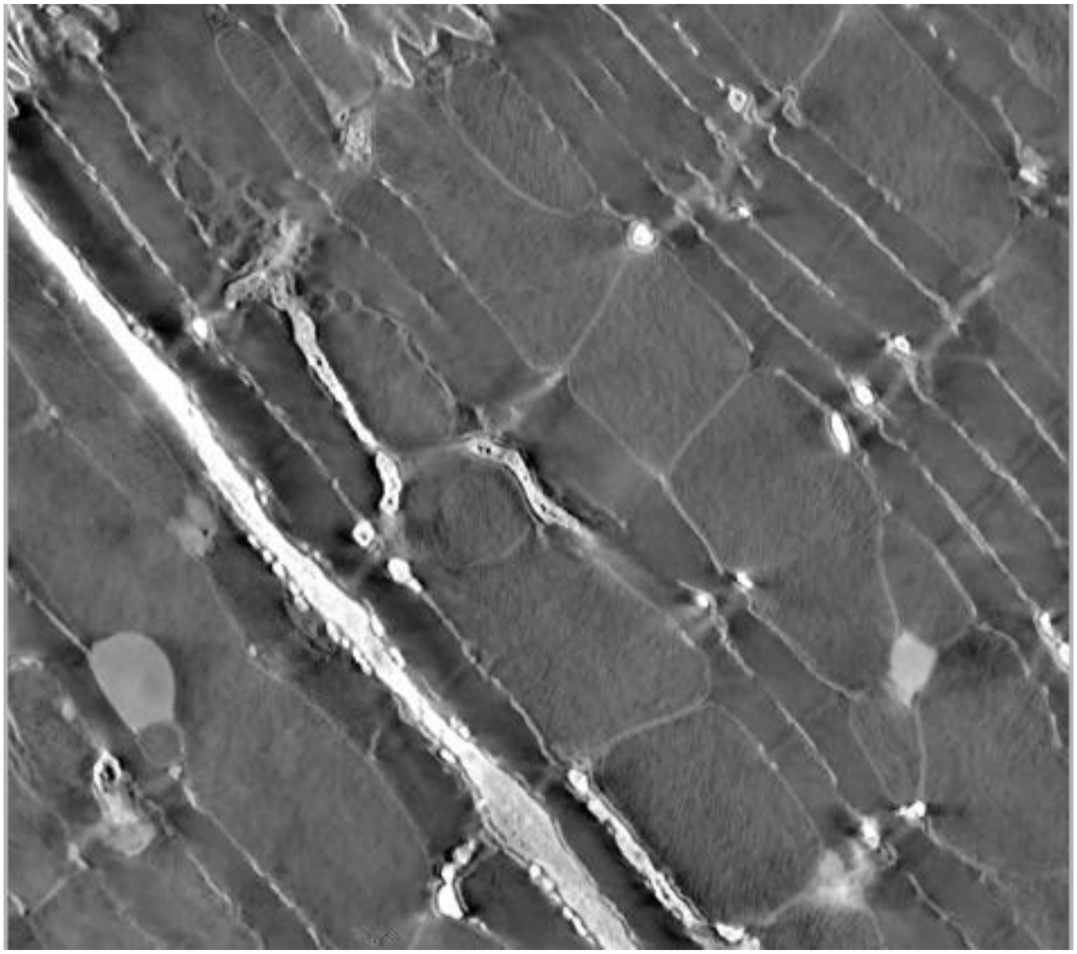
(e) (f)

Fig 2.7: Result of our method running on a 256*256 cardiac cell image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 46 ms. The time for PCA sampling is 31 ms (c) Result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 15 ms (d) Result of the image-based ODT method. The time for ODT smoothing is 47 ms (e) Result of the image-based CVT method. The time of the CVT smoothing is 47 ms (f) A closeup look at the rectangular region in (e).

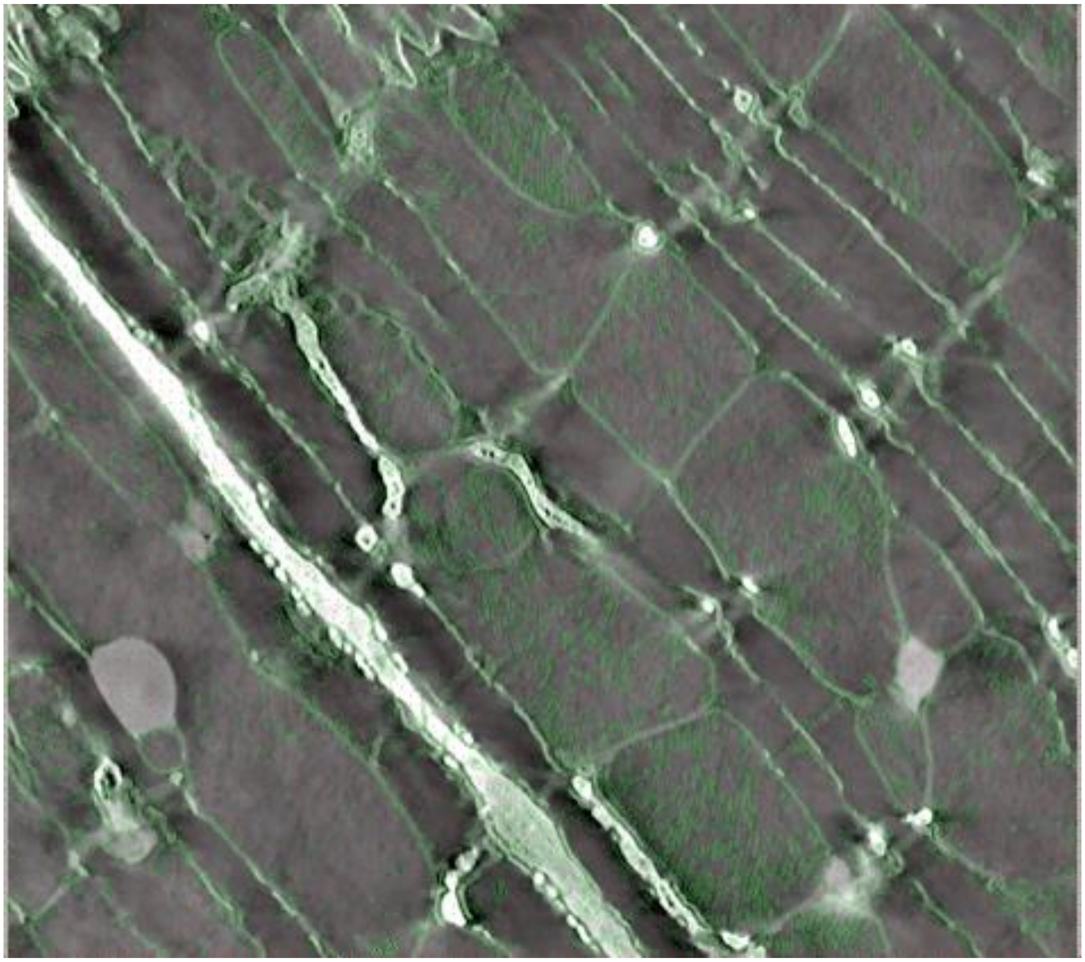
Table 2.3: Running time of our mesh generation method on a 256*256 cardiac cell image.

Steps	Running time (ms)
Canny and halftoning nodes	46
PCA sampling	31
Initial mesh generation	15
ODT	47
CVT	47

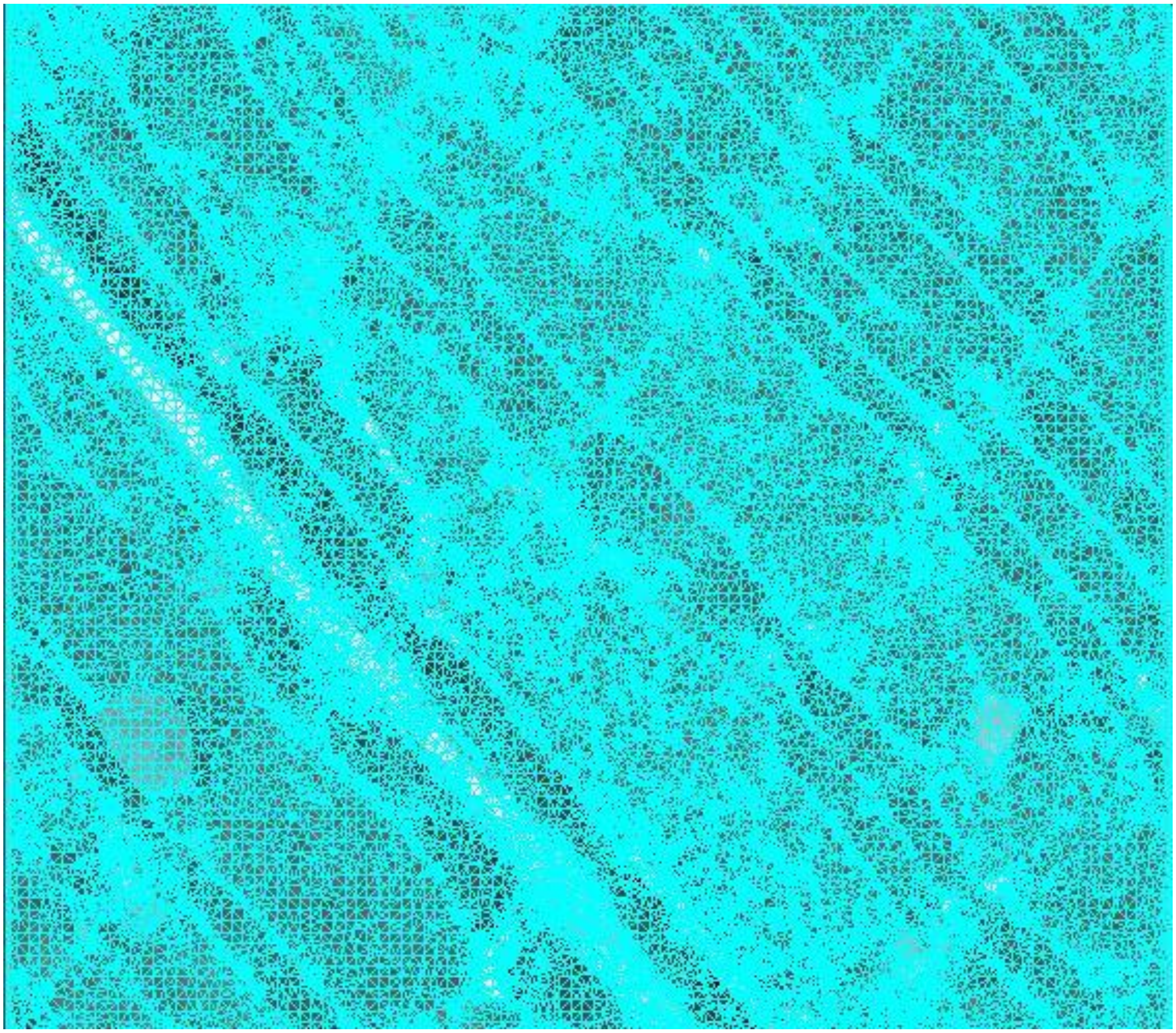
Finally, we implement my method on a big image in Figure 2.8(a) we show a 1032*912 noisy image of cardiac cells. (b) and (c) show the sample points found and the initial mesh by Delaunay triangulation. The smoothed meshes in (d) and (e) show that even deal with a very big noisy image, the proposed method also can generate high quality mesh fast and it take 1404 ms in all.



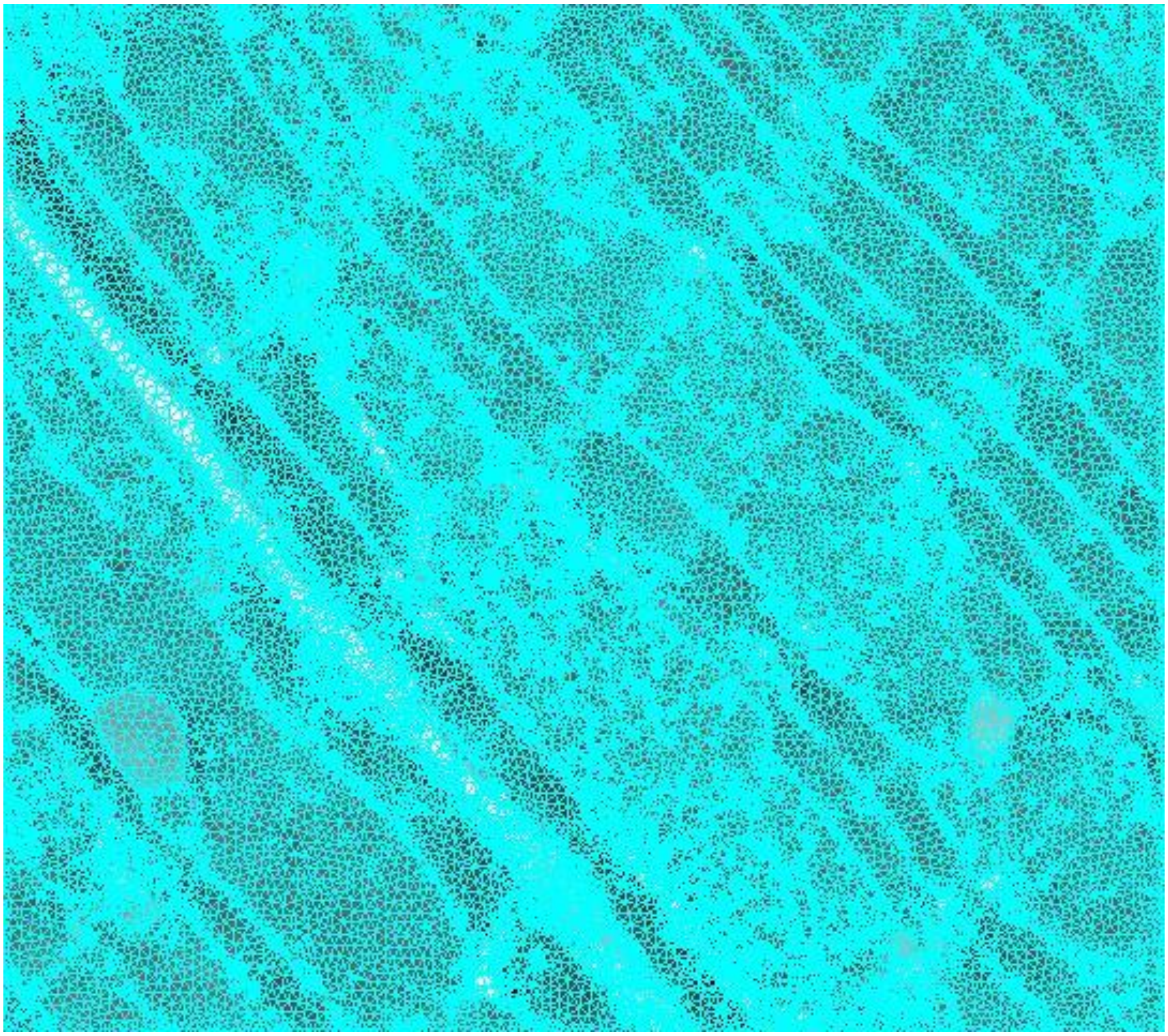
(a)



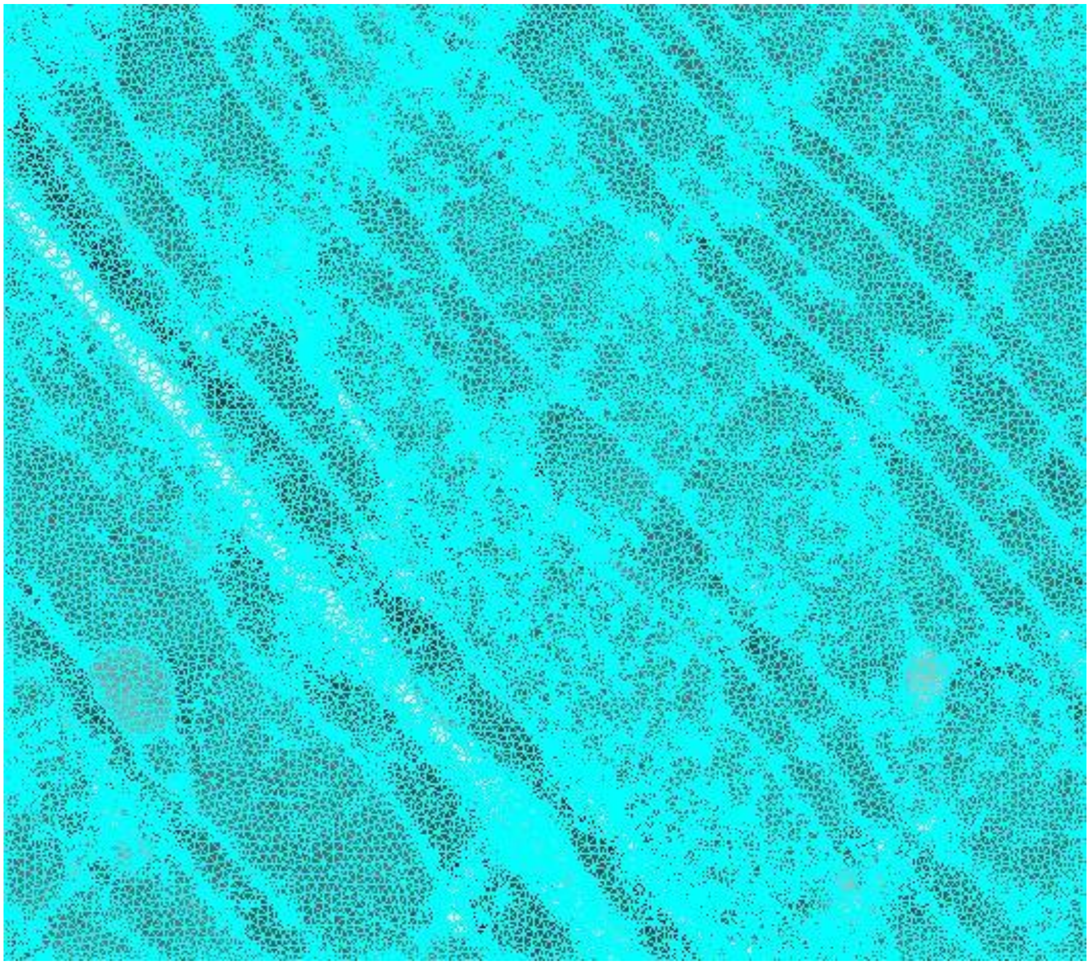
(b)



(c)



(d)



(e)

Fig 2.8: Result of our method running on a 1032*912 cardiac cell image. (a) The original image. (b) The result of the sample point generation. Green points are Canny's sample points. Red points are halftoning and uniform sample points. The time for Canny sample nodes generation is 265 ms. The time for PCA sampling is 437 ms. (c) The result of initial mesh generated by Delaunay triangulation. The time for the generation of initial mesh is 140 ms (d) Result of the image-based ODT method. The time for ODT smoothing is 641 ms (e) Result of the image-based CVT method. The time of the CVT smoothing is 562 ms

Table 2.4: Running time of our mesh generation method on a 1032*912 cardiac cell image.

Steps	Running time (ms)
Canny and halftoning nodes	265
PCA sampling	437

Initial mesh generation	140
ODT	561
CVT	642

I am going to introduce the 3D cases and experiment results with running time information in chapter 4.

Chapter 3

Super-Pixels Based 2D Image Segmentation

In this chapter, every triangle in the triangular mesh generated by the above chapter is treated as a node of a weighted graph. Each edge between two nodes represents the reversed difference between two neighboring nodes. On the other words, the weight of edge between two neighboring nodes is greater when the two nodes' average intensities are more similar. Then a popular method: S-t cut is adopted to do the graph cut on the above weighted graph and generates the final segmentation result.

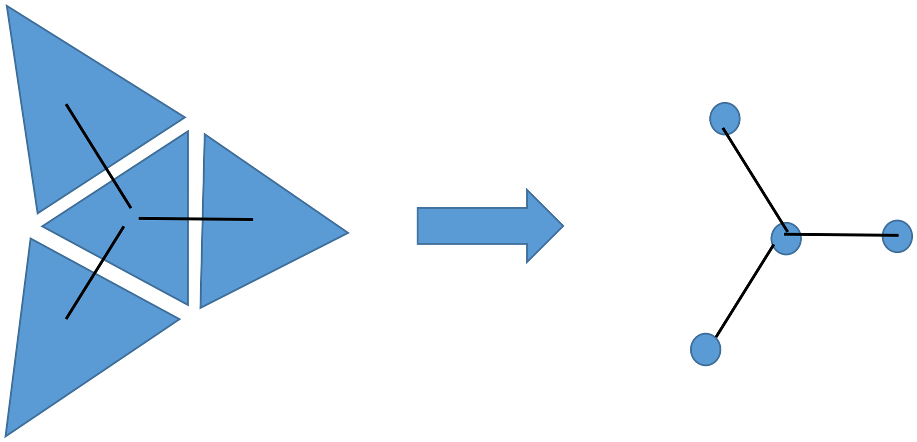


Fig 3.1: The weighted graph generation based on Super-Pixels (2D).

3.1 S-t Graph Cut Method

After connecting the triangles in the triangular mesh generated in last chapter, we need to set the weight for every edge between two super-pixels and generate the segmentation result edges set by min-cut algorithm.

S-t cut is a popular graph cut method which accomplishes global optimization on a specific energy function to generation optimal segmentation results. Below we give a brief summary of this approach. The interested reader is referred to [14] for more details.

3.1.1 Background of Graph

Graph consists of a set of nodes V and a set of edges E. A graph can be represented as G = <V, E>. In addition, S-t cut algorithm adds two terminals which are s (source) node and t (sink) node to represent the object and background in image segmentation.

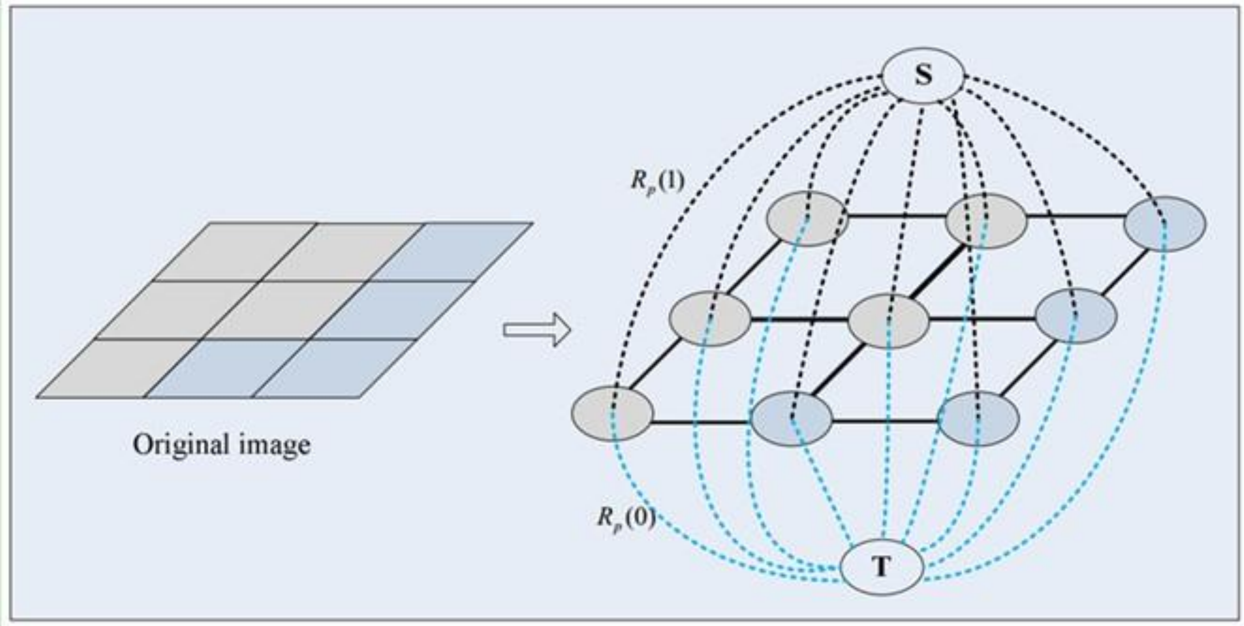


Fig 3.2: courtesy of [14], Example of S-t graph. Edges drawn by full lines are n-links and edges drawn by dotted lines are t-links.

Figure 3.2 shows a pixel-based simple example of S-t graph. The left lattice represents the original image with nine pixels. In order to generate the S-t graph, two terminals s and t are added into the graph. S-t graph has two kinds of edges: n-links and t-links. T-links are the edge which connect pairs of neighboring pixels from the original image. S-t graph connects s and t terminals with all the pixels in the original image which is displayed by the edge drawn by dotted lines in figure 3.2. These edges call t-links in the S-t graph. Next section introduces how to assign the weights to edges including n-links and t-links in S-t graph. [14,105]

3.1.2 Edges Weight Assignment

This section introduces how to assign the weight for every edge including t-link and n-link according to the energy function introduced in [14]. The energy function of the S-t cut method is following equation[14]:

$$E(A) = \lambda \cdot R(A) + B(A) \tag{13}$$

Each A is an assignment to each specific graph node in our method and it can be either "Object" or "Background". R(A) is region properties term and B(A) is the boundary properties term. λ is a parameter which $\lambda > 0$. The formula of region properties term[14]:

$$R(A) = \sum_{p \in P} R_p(A_p) \tag{14}$$

Where:

$$R_p(\text{"}obj\text{"}) = -\ln \Pr(I_p \mid \text{"}obj\text{"}) \tag{15}$$

$$R_p(\text{"}bkg\text{"}) = -\ln \Pr(I_p \mid \text{"}bkg\text{"}) \tag{16}$$

In the above formula, -lnPr() is negative log-likelihoods. In my method, the user should choose the both object and background seeds (The number of the seeds is at least 1) manually. The seeds which are chosen manually are adopted to create two intensities histograms for both object and background. These two intensities histograms are used to calculate the Rp("obj") and Rp("bkg") value.

In this method, two extra terminals s (object) and t (background) are added into the graph and every node in the graph has two edges which connect both s and t. The weight between every node and s is in the following[14]:

$$W_{p,s} = \begin{cases} K & if \quad p \in O \\ 0 & if \quad p \in B \\ \lambda \cdot R_p(''bkg'') & else \end{cases} \tag{17}$$

K is greatest value among the all the edges' weight to ensure that the chosen node manually can be clustered into the type what users indicated. T

The edge weight between every node and s is in the following:

$$W_{p,t} = \begin{cases} K & if \quad p \in B \\ 0 & if \quad p \in O \\ \lambda \cdot R_p(''obj'') & else \end{cases} \tag{18}$$

The formula of boundary properties term:

$$B(A) = \sum_{\{p,q\} \in N} B_{p,q} \cdot \delta_{(A_p, A_q)} \tag{19}$$

$$\delta(A_p, A_q) = \begin{cases} 1 & if \quad A_p \neq A_q \\ 0 & otherwise \end{cases} \qquad (20)$$

B(A) represents the neighboring relationship between two nodes in the graph. In my method, because we use average intensity of the pixels in a triangle to be the intensity of the node in S-t graph, we use Ave_{I_p} and Ave_{I_q} to represent the intensity of a pair of neighboring super-pixels. We use the following equation to calculate $B_{p,q}$.

$$B_{p,q} = \exp(-\frac{(Ave_{I_p} - Ave_{Iq})}{2\sigma^2}) \qquad (19)$$

3.1.3 Image segmentation based on Min-cut

After generating a weighted S-t graph based on the energy function mentioned in last section, a new min-cut algorithm is introduced in this section to separate the graph into two trees [105].

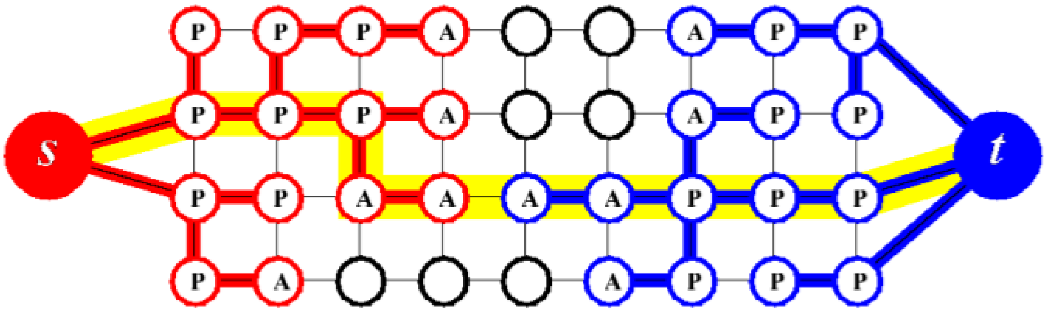


Fig 3.3: courtesy of [105], Example of Search trees S and T. P are passive nodes and A are active nodes. The blank nodes are free nodes.

Figure 3.3 is an example of two search trees min-cut algorithm. This algorithm builds two search threes from S-t graph. At beginning, S search tree and T search tree only have the roots which are s and t node demonstrated in figure 3.3. Both search trees will have internal nodes which are passive nodes and external nodes which are active nodes when growing. The active nodes can grow and connect its neighboring nodes. An augmenting path will be find if two nodes from different trees meet. The algorithm implement the following three steps repeat [105].

1 Growth stage: grow the search trees S ant T to find s → t path.

2 Augmentation stage: augment the found path and break search trees into forest.

3 Adoption stage: trees S and T are restored.

The pseudocode of the Min-cut algorithm is[105]:

```
Initialize: $S = \{s\}, T = \{t\}, A = \{s, t\}, O = \emptyset$
while
    1 grow the search trees S ant T to find s to t path.
    2 if $P = \emptyset$ terminate
    3 augment on P adopt orphans
     End
```

At beginning, S tree only have one node s and T have one node t. Both s and t are active and there are no orphan node in the graph. TREE (p) has three choices: S, T or ∅ (p is free).

Growth stage [105]**:**

```
while $A \neq \emptyset$
    pick an active node $p \in A$
    for every neighbor q such that tree $cap(p \rightarrow q) > 0$
        if $TREE(q) = \emptyset$ then add q to search tree as an active node:
            $TREE(q) = TREE(p), PARENT(q) = p, A = A \cup \{q\}$
        if $TREE(q) \neq \emptyset$ and $TREE(q) \neq TREE(p)$ return $P = PATH(s \rightarrow t)$
    end for
    remove p from A
end while
return $P = \emptyset$
```

In this pseudocode, $cap(p \rightarrow q)$ is residual capacity of either edge (p, q) if TREE(p) = S or edge (q, p) if TREE(p) = T. $PATH(s \rightarrow t)$ is the path connect from S search tree to T search tree. In the growth stage, once an active node p is picked, all the neighboring nodes of p with the edge whose residual capacity is greater than 0 will be added into the search tree. If path is find, this step stops.

Augmentation stage [105]**:**

```
find the bottleneck capacity $\Delta$ on P
update the residual graph by pushing flow $\Delta$ through P
for each edge (p, q) in P that becomes saturated
    if $TREE(p) = TREE(q) = S$ then set $PARENT(q) := \emptyset$ and $O := O \cup \{q\}$
    if TREE(p) = TREE(q) = T then set PARENT(p) := $\emptyset$ and $O := O \cup \{p\}$
end for
```

Firstly, we push the flow through the new found path P. If any edge (p, q) becomes saturated, p or q will be added to the orphan set. In the adoption stage, all the orphans in the orphan set will be processed.

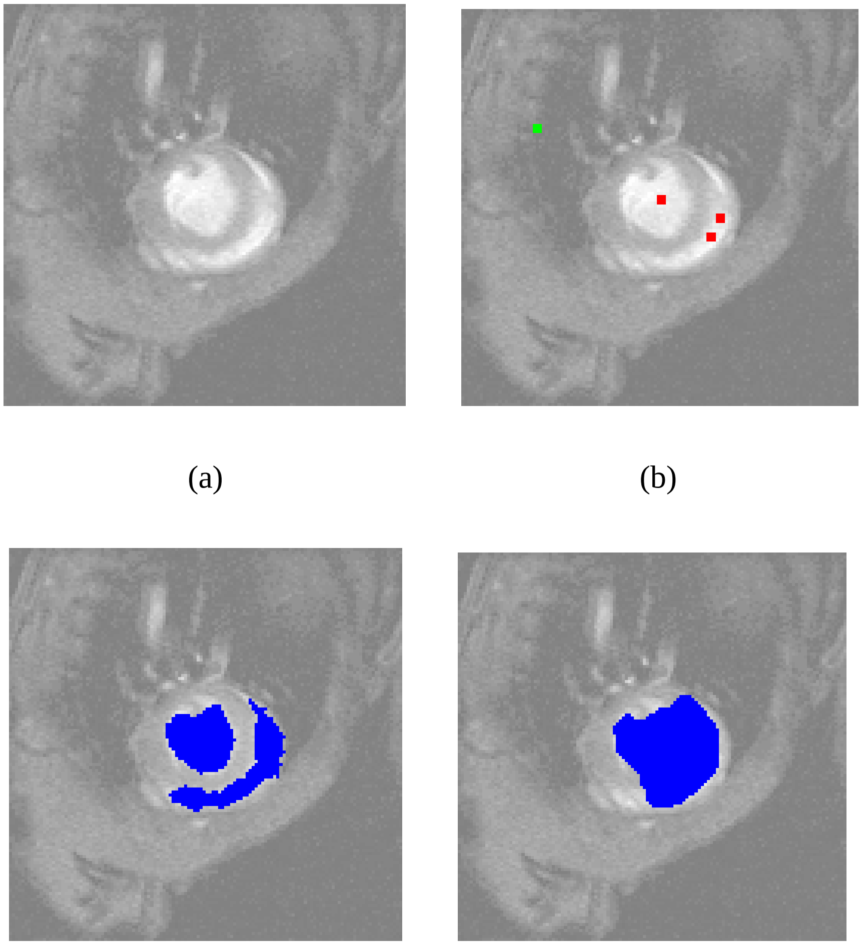
Adoption stage[105]**:**

```
while $O \neq \emptyset$
    pick an orphan node $p \in O$ and remove it from O
    scan all neighbors q of p such that TREE(q) = TREE(p):
        – if tree cap($q \to p$) > 0 add q to the active set A
        – if PARENT(q) = p add q to the set of orphans O and set PARENT(q) = $\emptyset$
    TREE(p) = $\emptyset$, $A = A - \{p\}$
end while
```

In this step, all the orphans are processed. Neighboring nodes of p are traversed to find a new node to be the parent node of p. If p can’t find a valid parent, all p’s children will be added to O and p will become free.

3.2 Experimental Results

In this section, I display the experimental results of the image segmentation by our method on some bio-medical examples. I also list the running time of the method given in this thesis (including the node generation, mesh generation and graph cut) together and compares it with the running time of the pixel-based image segmentation. I am not going to take the running time for mesh smooth into account because the segmentation results by the initial meshes are good enough. The good quality of the mesh has the huge benefits in the following modeling stage. For the sake of speed, the initial mesh helps to achieve the real-time performance.



(a) (b)

(c) (d)

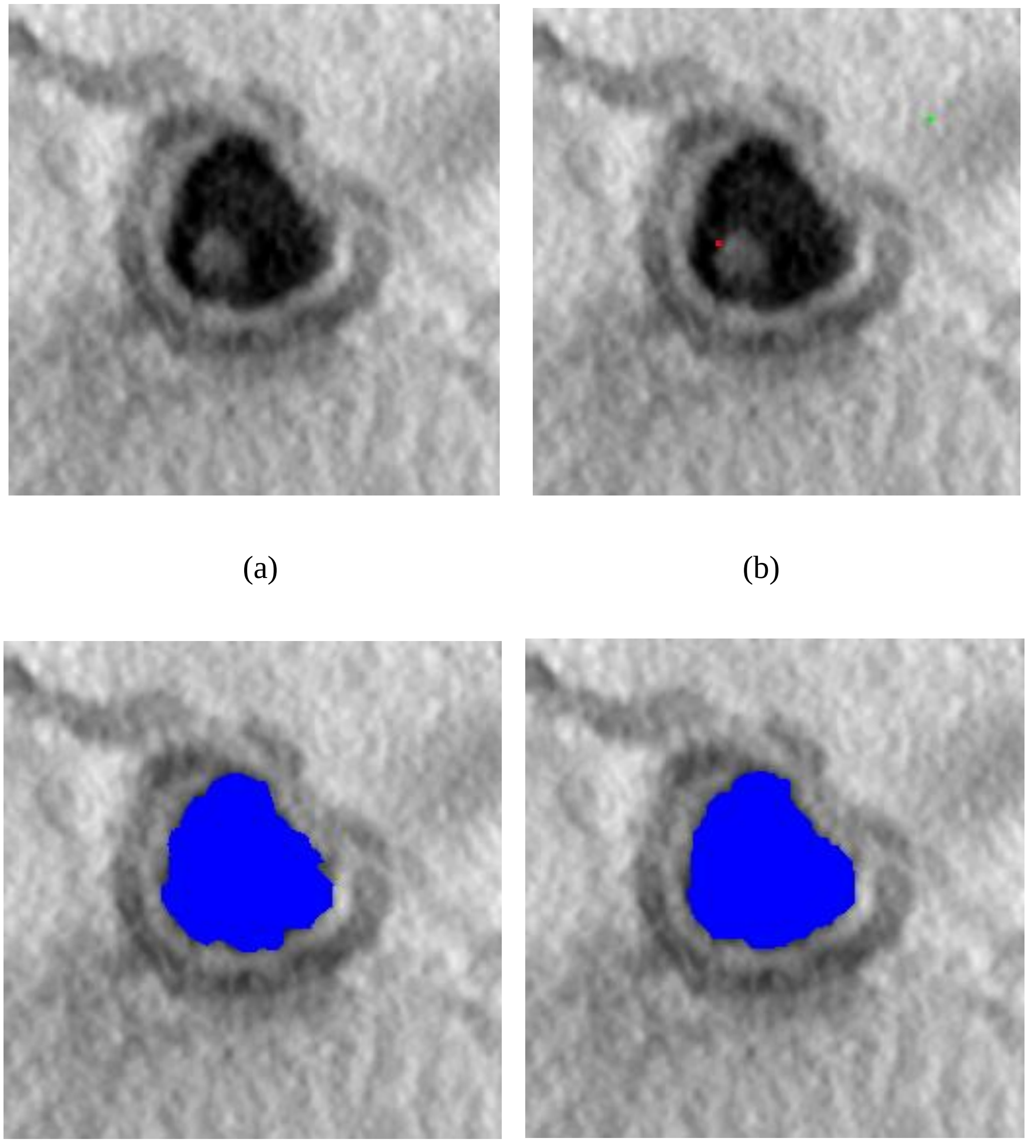
Fig 3.4: Result of image segmentation on a 128*128 heart image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) The result of super-pixel based image segmentation. The running time for segmentation is 15 ms (d) The result of pixel based image segmentation. The running time for segmentation is 46 ms.

Table 3.1: Running time comparison between our superpixel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 128*128 heart image.

	Super-pixel Based	Pixel Based
Size	128*128	128*128
Canny and halftoning nodes (ms)	15	
PCA sampling (ms)	12	
Initial mesh generation (ms)	15	
S-t sut (ms)	15	46
Running time in all (ms)	57	46

Fig 3.4(a) displays a 128*128 original heart image. (b) shows the seeds which chosen by the user manually. We choose three seeds (red) in the features and one seed (green) in the background. (c) is the result of the image segmentation based on our super-pixel method. The graph cut running time is 15 ms. If we take the running time from nodes generation, initial mesh generation and S-t cut into account, the running time is 60 ms. It is a bit longer than the pixel based segmentation which is 46 ms in this image which is displayed in (d) because this image's size is too small. However the quality of the result of the super–pixels based image segmentation is much

better than the pixel-based image segmentation if the number and the position of the seeds are same. In (d), the segmentation result crosses the small gap between the two features which is not as good as the result in (c).



(a)

(b)

(c)

(d)

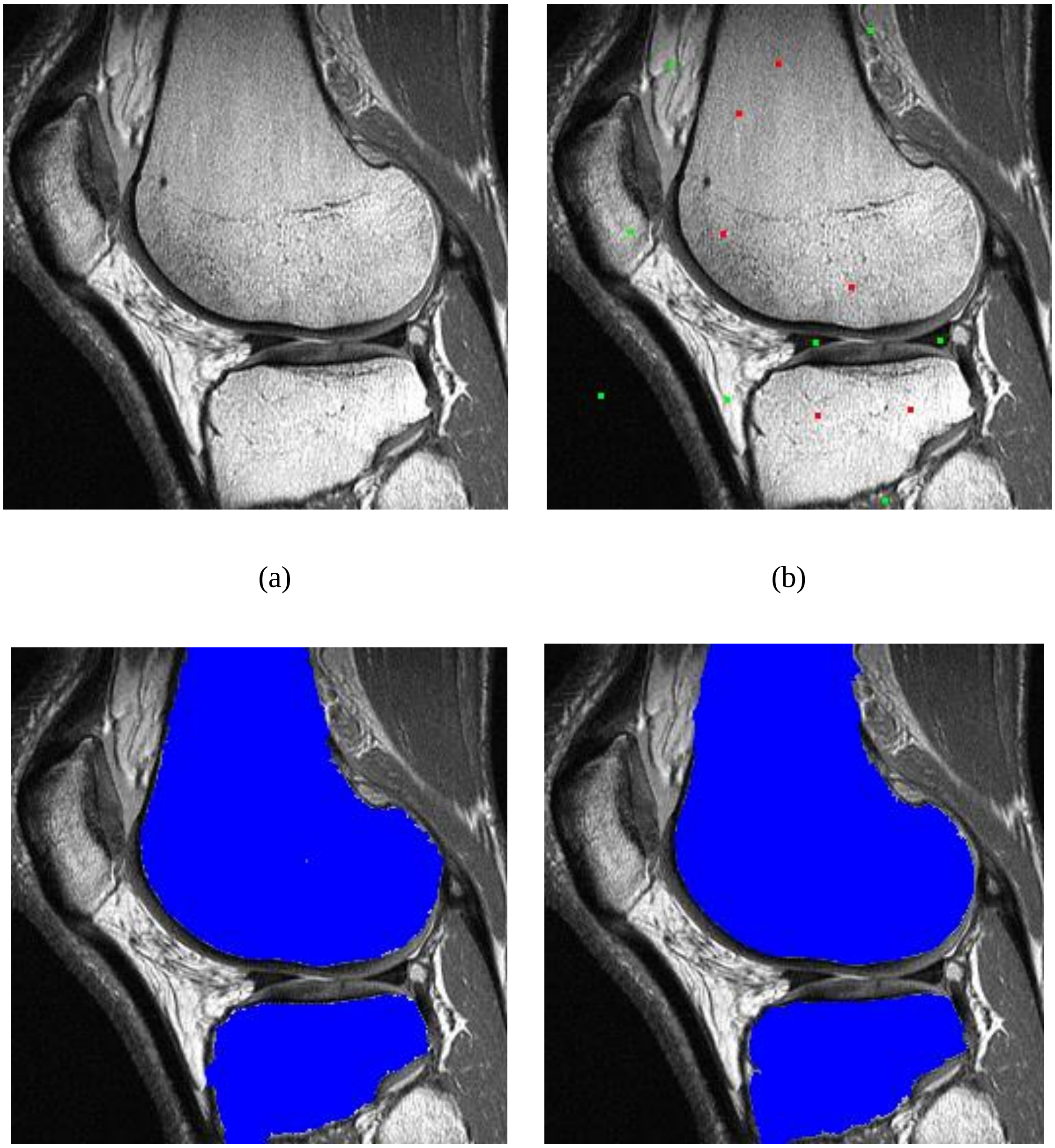
Fig 3.5: Result of image segmentation on a 256*256 heart image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) The

result of super-pixel based image segmentation. The running time for segmentation is 16 ms (d) The result of pixel based image segmentation. The running time for segmentation is 203 ms.

Table 3.2: Running time comparison between our superpixel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 256*256 cell image.

	Super-pixel Based	Pixel Based
Size	256*256	256*256
Canny and halftoning nodes (ms)	46	
PCA sampling (ms)	31	
Initial mesh generation (ms)	15	
S-t sut (ms)	16	203
Running time in all (ms)	108	203

Fig 3.5(a) displays a 256*256 cell image. As we said before, this image is extremely noisy. (b) is the seeds map which chosen by the user manually. We choose only one seed on both object and background. (c) is the result of the image segmentation based on our super-pixel method. The graph cut running time is 16ms. We take the overall running time into account and the running time is 108 ms which is much fast than 203 ms by the pixel-based segmentation. The super-pixels help to smooth the noisy image because of the average intensity strategy. Both of the qualities of the image segmentation results are very good.



(a) (b)

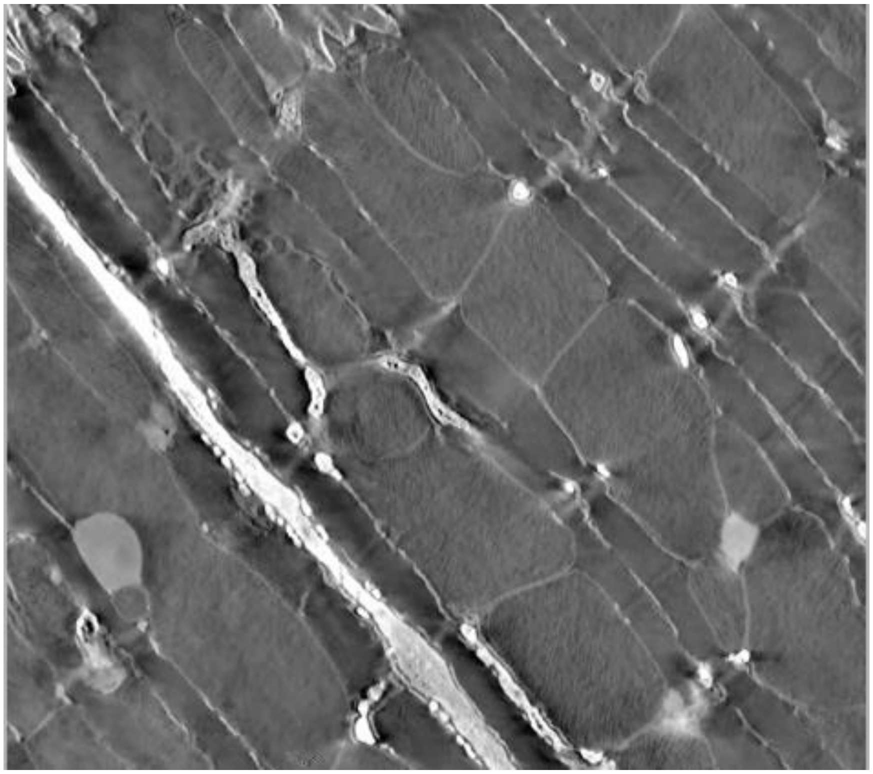
(c) (d)

Fig 3.6: Result of image segmentation on a 256*256 knee image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) The result of super-pixel based image segmentation. The running time for segmentation is 16 ms (d) The result of pixel based image segmentation. The running time for segmentation is 170 ms.

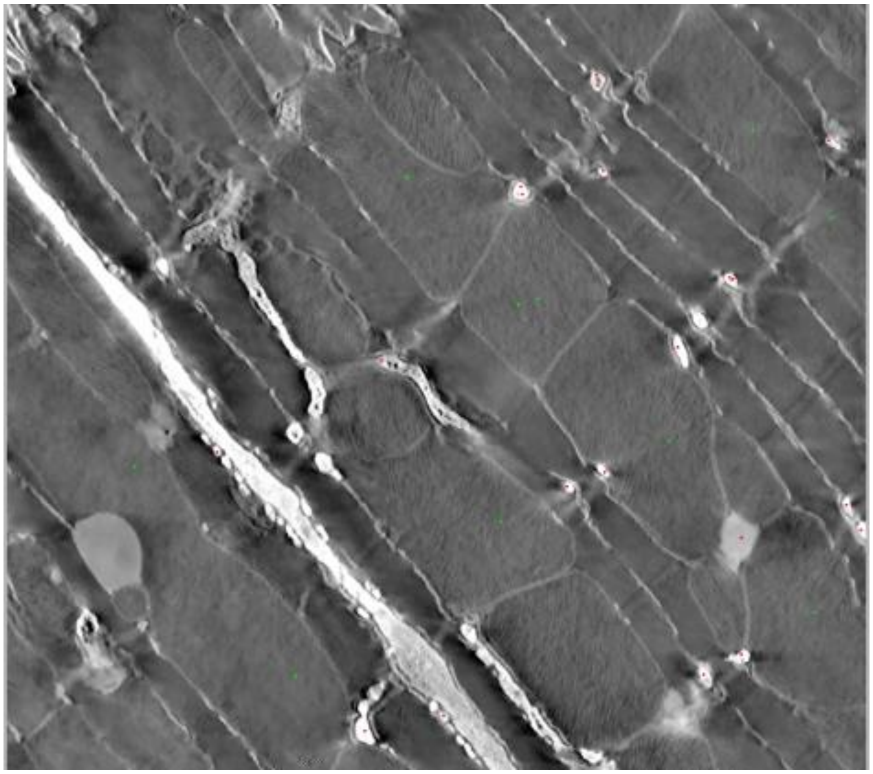
Table 3.3: Running time comparison between our superpixel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 256*256 knee image.

	Super-pixel Based	Pixel Based
Size	256*256	256*256
Canny and halftoning nodes (ms)	15	
PCA sampling (ms)	31	
Initial mesh generation (ms)	15	
S-t sut (ms)	16	170
running time in all (ms)	77	170

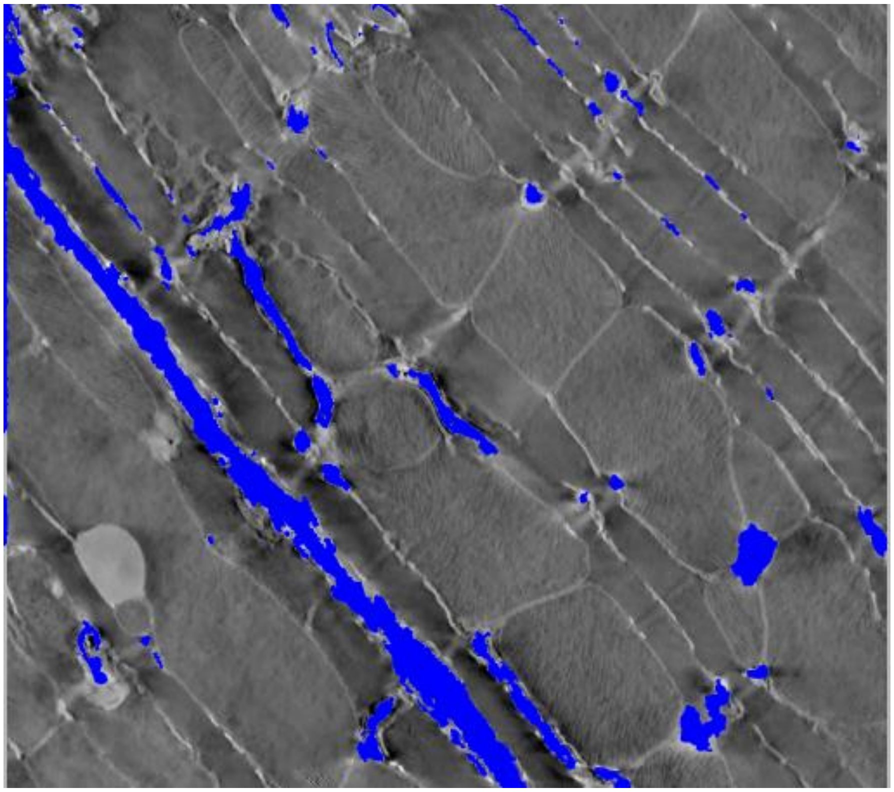
Fig 3.6(a) is a 256*256 knee image. (c) is the result of the image segmentation based on our super-pixel method. The graph cut running time is 16 ms. Running time in all is 77 ms. Comparing with the 170 ms by the pixel-based segmentation, our method is much faster.



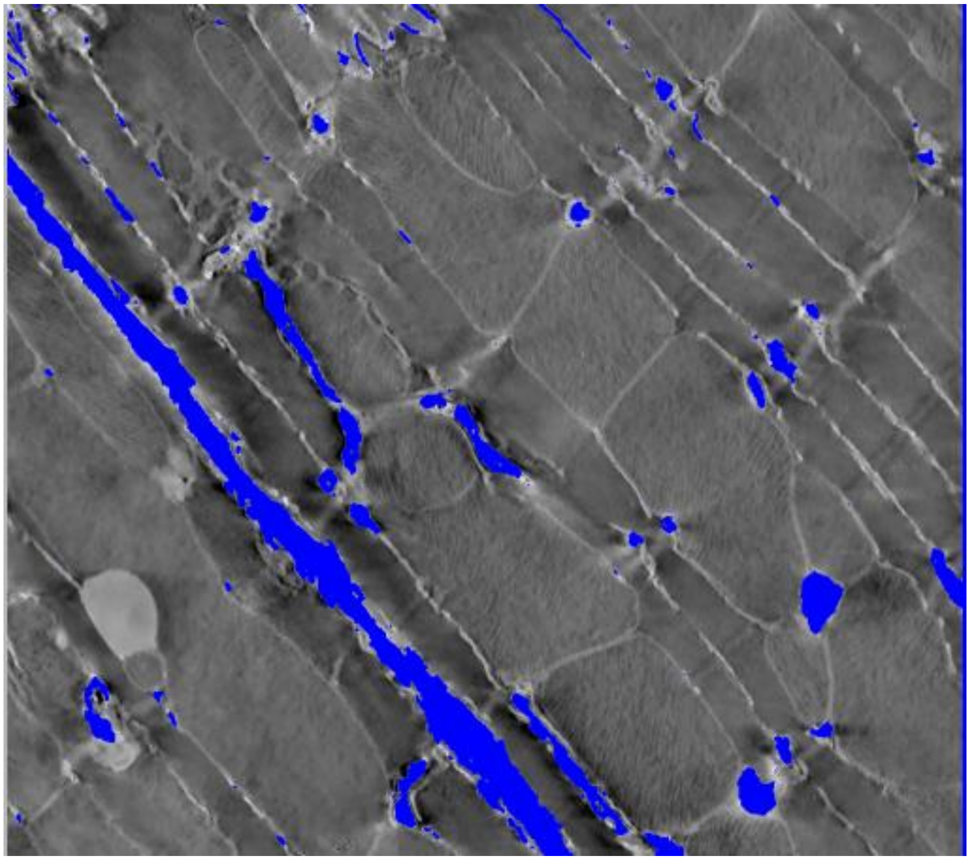
(a)



(b)



(c)



(d)

Fig 3.7: Result of image segmentation on a 1032*912 cell image. (a) The original image. (b) The seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) Result image of super-pixel based image segmentation. The running time for segmentation is 250 ms (d) Result image of pixel based image segmentation. The running time for segmentation is 1980 ms.

Table 3.4: Running time comparison between our superpixel-based 2D image segmentation with the classical pixel-based 2D image segmentation on a 1032*912 knee image.

	Super-pixel Based	Pixel Based
Size	1032*912	1032*912
Canny and halftoning nodes (ms)	265	

PCA sampling (ms)	437	
Initial mesh generation (ms)	140	
S-t sut (ms)	250	1980
Running time in all (ms)	1092	1980

Fig 3.7(a) is a very big 1032*912 cell image. (b) is the seeds which are chosen manually. Red points are the object seeds and green points are background seeds. (c) is the result of the image segmentation based on our super-pixel method. The graph cut running time is 250 ms. The overall running time including node generation, initial mesh generation and S-t cut is 1092 ms. The pixel-based S-t cut image segmentation takes 1980 ms. So the bigger the image is, the more time saving by using our super-pixels based image segmentation.

Chapter 4

Adaptive Meshes Generation and Image Segmentation on 3D Images

In this chapter, a series of algorithms to generate high quality, feature-sensitive, and adaptive tetrahedral mesh are introduced, where the 3D Canny edge detector is utilized to preserve important feature boundaries. Each cluster of voxels within a tetrahedron is called a super-voxel, which is treated as a node in a graph and weighted edges are formed between adjacent tetrahedras (or super-voxels). A graph cut method S-t cut is then applied on the weighted graph to partition the tetrahedral mesh, resulting in a segmentation of image volume.

Same as 2D case, the advantage of this method is not only to implement the 3D image segmentation and feature extraction from the given bio-medical 3D volume. The object which is extracted from the background by the method has already been meshed during this segmentation procedure.

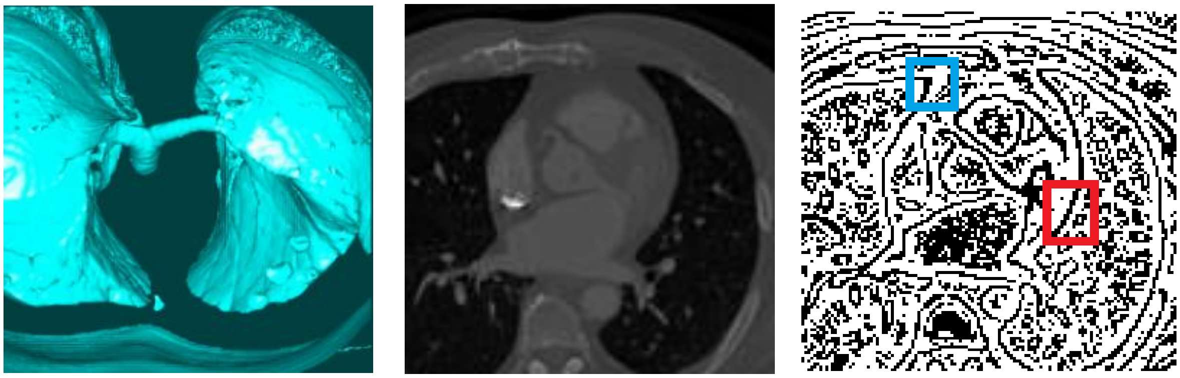
This method generates a bunch of nodes first and we treat these nodes as the tetrahedrons' vertices in the tetrahedrons generation step. For sake of the accurate and adaptive requirement, the most of the nodes should be strictly on the boundary of the features. We utilize 3D Canny edge detector and 3D PCA sampling in order to generate enough but not excessive nodes. Same as the 2D case, the number of these nodes decides overall running time of this method. Our method runs much faster than the classical voxel-based mesh generation and image segmentation method, especially the very noisy cases. One of the very noisy example in the experimental results section shows that the running time of our method is around nine times faster than classical voxel-based method. The speed and real-time performance of this method is outstanding. On the other hand, same as 2D case, the intensities in a super-voxel is averaged which can serves as a noise reduction operator. Our method can deal with even extremely noisy 3D bio-medical volume. We set the smaller sampling radius on curved-boundary or tiny features and the greater sampling radius on straight-boundary or big features. The final experiment results shows that the quality of the both surface and volumetric mesh are very high.

4.1 Nodes Generation

In this step, some voxels from the given 3D volume are picked to be the nodes. Because the goal of this method is to generate an adaptive and feature-sensitive tetrahedral mesh, the quantity and the position of the nodes become very important. The nodes picked in this step should be strictly on the boundary of the features. Moreover, this method generates dense nodes on curved-boundary or tiny features and sparse nodes on straight-boundary or big features likes the 2D case.

4.1.1 Canny Sample Nodes

This method utilizes 3D Canny edge detector. Fig 4.1 is a 128*128*128 thoracic cavity 3D volume and we are only interested in lung.



(a) (b) (c)

Fig 4.1: (a) The Iso-surface of a 128*128*128 thoracic cavity 3D volume. (b) The slice image of thoracic cavity 3D volume. (c) The slice image of the result of 3D Canny edge detector. Blue rectangular region is the curved-boundary or tiny features. Red rectangular region is the straight-boundary or big features.

From Fig 4.1(c), we realize that the marked voxels are strictly on the boundary of the features. Likes 2D case, we can't just treat all the voxels in this result as the nodes because we don't want the result tetrahedrons have very tiny angles. We need to do the sampling on the above result. Moreover, we need to generate dense nodes on the curved-boundary or tiny features which are displayed in the blue rectangular region in (c) and sparse nodes on the straight-boundary or big features which are displayed in the red rectangular region in (c).

We utilize the 3D principal component analysis (PCA) to calculate the sampling radius. When we decide whether we are going to keep the nodes or discard them, we need to take the neighborhoods of every marked node into account. When we visit the arbitrary marked node denoted as $\overline{P}$ in the canny map. We call $\overline{P}$'s K×K×K neighborhoods: P_1, P_2... P_n. n is the number of the neighborhoods of $\overline{P}$. K is a parameter which is decided by the size of the features. We set K=5 for all the examples in this thesis. The covariance matrix is calculated in the following formula [22]:

$$\sum_{j=1}^{n}(P_j - \overline{P})(P_j - \overline{P})^T \in \mathbb{R}^{3\times 3} \tag{22}$$

λ_1, λ_2, λ_3 are three eigenvalues of above covariance matrix. We assume these three eigenvalues follow: $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The sampling radius R($\overline{P}$) is calculated by the following equation:

$$R\left(\overline{P}\right) = \begin{cases} 2 & \left(\lambda_1 - \lambda_3\right) / \left(\lambda_1 + \lambda_3\right) \leq 0.3 \\ 3 & 0.3 < \left(\lambda_1 - \lambda_3\right) / \left(\lambda_1 + \lambda_3\right) \leq 0.5 \\ 4 & else \end{cases} \tag{23}$$

4.1.2 Surfaces Nodes

After the PCA sampling, some extra nodes should be added on the six surfaces of the given 3D volume because this method needs them to generate the tetrahedrons in next step. Fig 4.2 shows a slice image of the final result of nodes generation step.

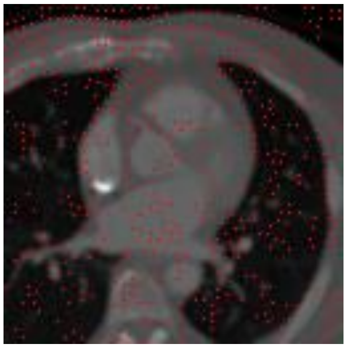
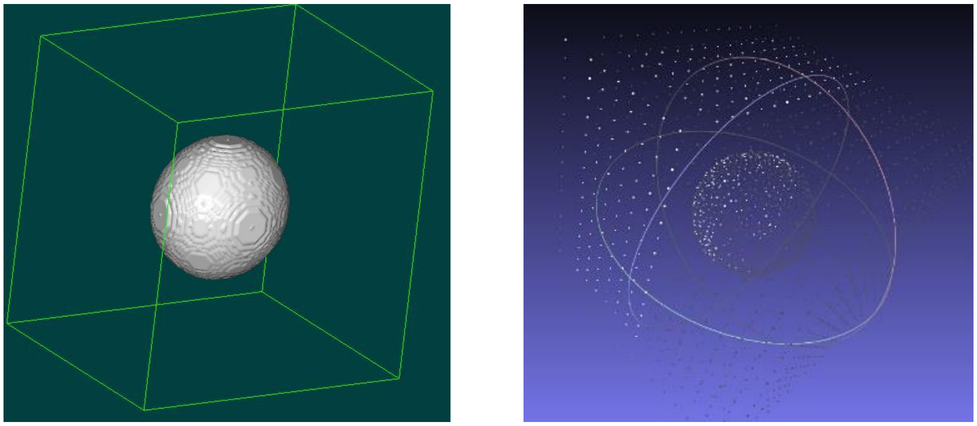


Fig 4.2: A slice image of the 3D thoracic cavity volume with the adaptively nodes which are marked in red including both Canny sample nodes and surfaces nodes.

Fig 4.2 shows that most of the nodes are on the boundary of the lung which we are interested in and some nodes are distributed in the background. Here I provide a simple example which is a single sphere.



(a) (b)

Fig 4.3: (a) Iso-surface of a 128*128*128 sphere 3D volume. (b) The adaptive nodes including both Canny sample nodes and surfaces nodes.

From Fig 4.3, we realize that the all nodes generated by the above step are on the surface of the feature which is a sphere because this sphere example do not have any noise and other features.

4.2 Mesh Generation via Delaunay Tetrahedralization

In this step, we treat the nodes generated from above step as the input. An open source tetrahedral mesh generator is adapted to generate a quality tetrahedral mesh based on the input nodes [50]. Fig 4.4 displays a slice of the Delaunay tetrahedralization result. The number of the input nodes is 51625. After the implementation of the Delaunay tetrahedralization, we generate a tetrahedral mesh whose faces number is 65050 and tetrahedrons number is 324577.

Fig 4.4 shows that the mesh quality is pretty good. This method uses the average of the intensities of all the voxles in a tetrahedron to be the value of this super-voxel. It makes the supervoxel-based method also has a smooth effect on the segmentation. The qualities and shapes of the tetrahedrons are pretty good in Fig 4.4 (b).

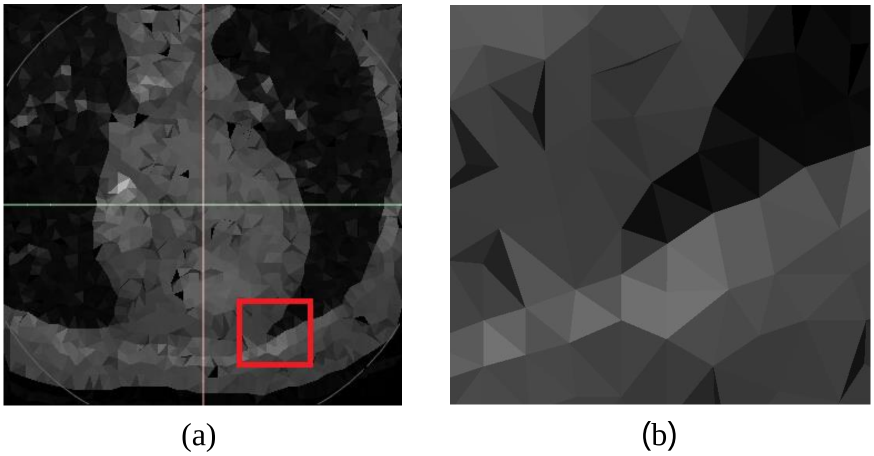

(a) (b)

Fig 4.4: (a) A slice of 128*128*128 thoracic cavity volume mesh generation via Delaunay tetrahedralization. Nodes number= 51625, tetrahedrons number=324577. (b) A closeup look at the rectangular region in (a)

4.3 Super-voxels Based 3D Segmentation on Meshed 3D Images

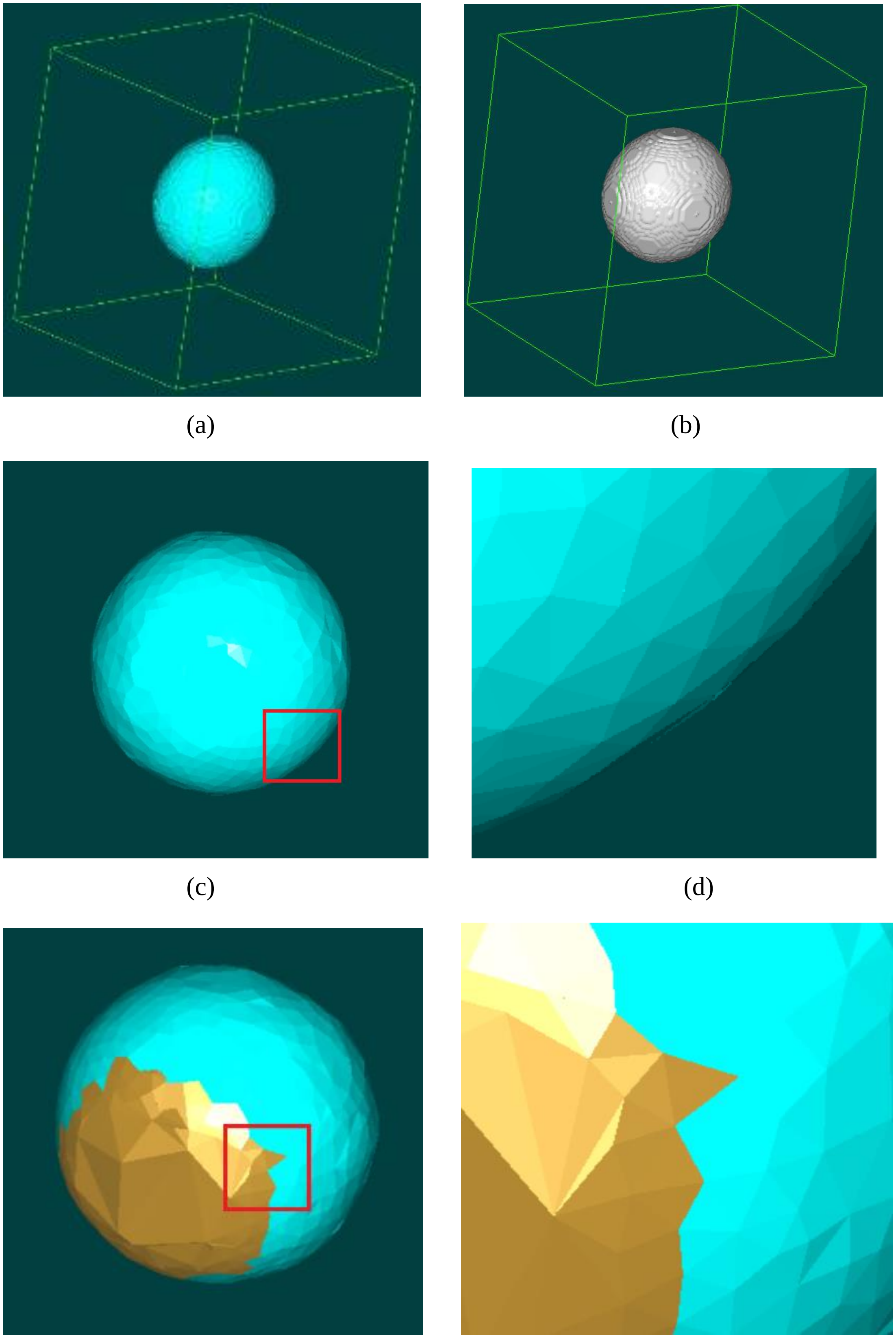
In the 3D segmentation step, similar with 2D case, we treat every tetrahedron as a node in a weighted graph. The value for each node is the averaged intensity of the voxels in this super-voxel. Every node of this graph except the nodes which are on the surfaces of the 3D volume have exactly four neighborhoods. Then a popular global optimization method: S-t cut is adapted to do the graph cut on the above weighted graph [14, 105].

4.4 Surface Smoothing

The surfaces of the results after graph cutting are usually not smooth enough. These 3D cases utilizes the Laplacian smoothing to generate a better visual result. We are going to list the results of several experimental examples in the next section. In this thesis, the iteration number of Laplacian smoothing is 3 and Lambda is 0.5.

4.5 Experimental Results

In this chapter, we lists several experimental results of some 3D bio-medical image volume examples. We also lists the running time of the method in this thesis (including the node generation, mesh generation, graph cut and smoothing) and compares it with classical voxel-based 3D image segmentation method. Compare with the pixel-based image segmentation, the running time of mesh generation is trivial. So in the running time table, I only list the pixel-based image segmentation running time to compare with the running time of the mesh generation method in this thesis. All the results show that our method could fulfill the real-time performance requirement and it also keeps the features accurately.



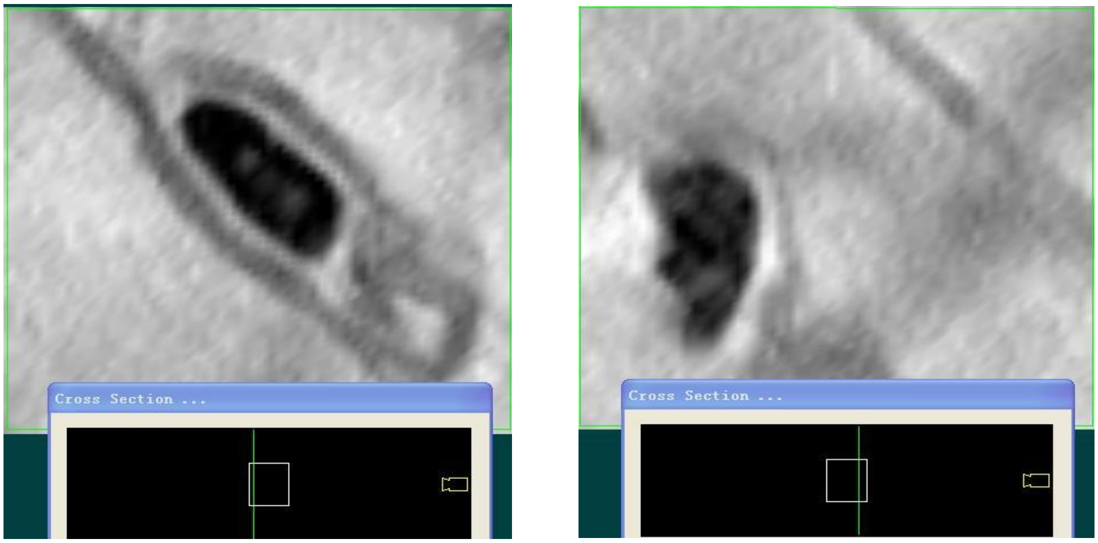
(a) (b)

(c) (d)

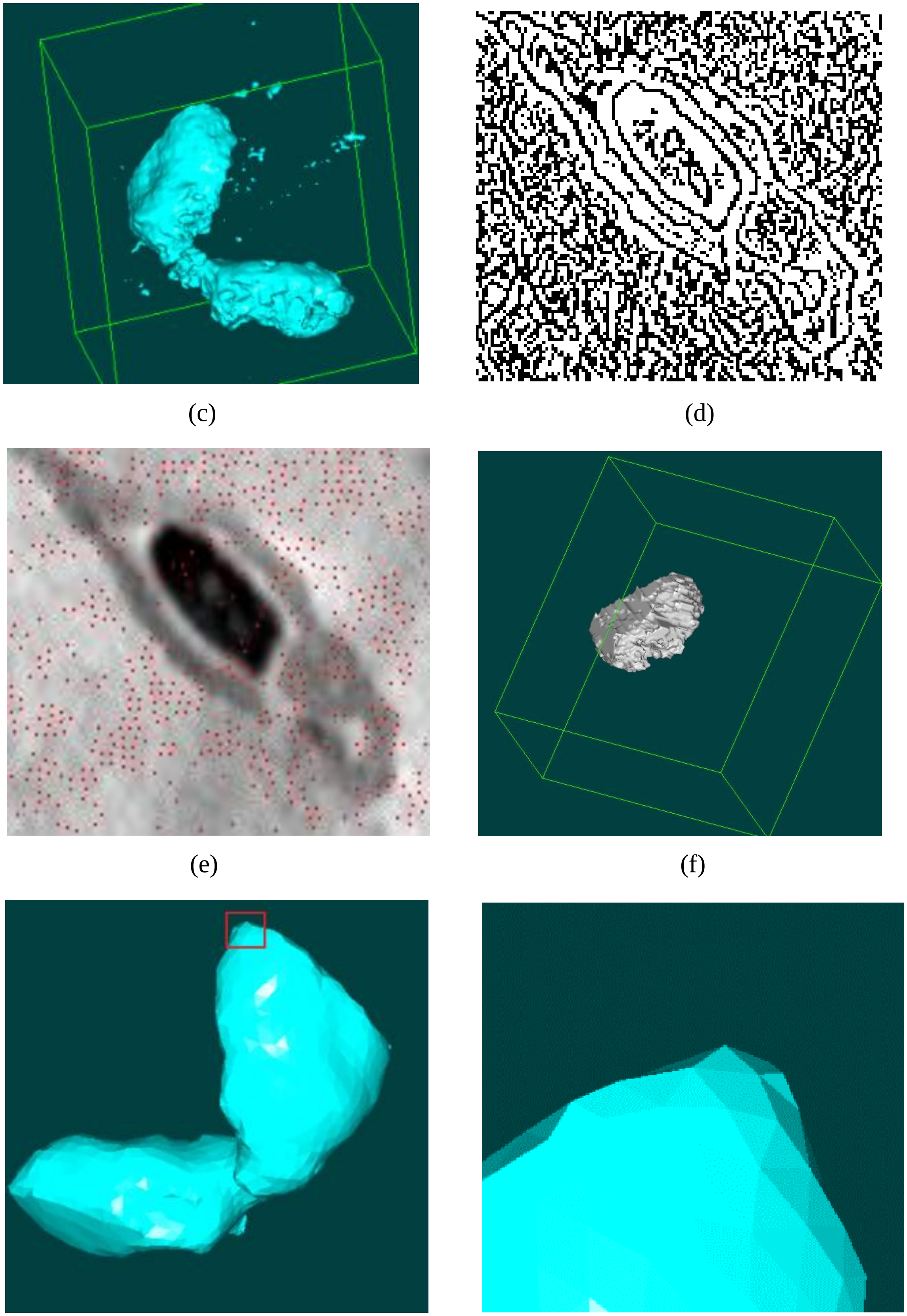
(e) (f)

Fig 4.5: (a) Iso-surface of a 128*128*128 sphere 3D volume. (b) The result of classical voxel-based 3D image segmentation. The running time is 3640. (c) The result of the supervoxel-based 3D image segmentation. The running time of 3D Canny edge detector is 1438 ms, The running time of PCA sampling is 62 ms. The running time of the generation of the tetrahedral mesh is 78 ms. Running time of S-t cut based on the super-voxels is 16 ms. Running time of Laplacian smoothing is 78 ms. (d) A closeup look at the rectangular region in (c). (e)The clipping result of volume mesh structure. (f) A closeup look at the rectangular region in (e).

Fig 4.5 is a very simple example which is a sphere in a 128*128*128 3D volume. This example don’t have any other features in the 3D volume. (b) is the result of the classical voxel-based 3D image segmentation and the surface of it is not smooth enough. The running time of this classical method is 3640ms. (c) is the result of the supervoxel-based 3D segmentation and its surface is very smooth. The PCA sampling helps us a lot to reduce the number of the nodes. There are only 2353 nodes and 10190 tetrahedrons in this example. The overall running time of our method is 1672 ms and it is much faster than the classical voxel-based method. (d), (e) and (f) shows that the quality and shape of the result mesh is very good.



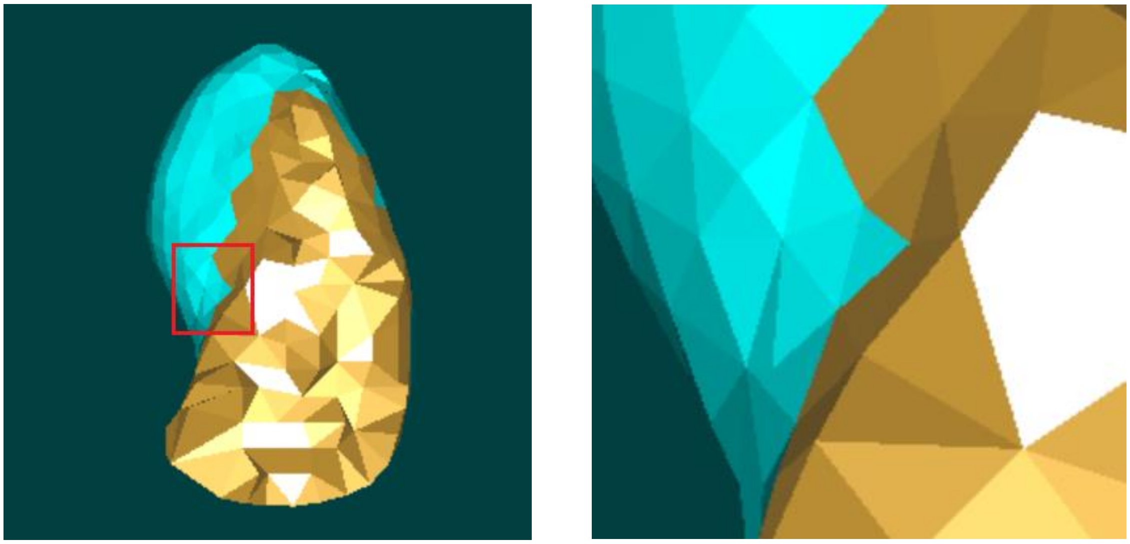
(a) (b)



(c) (d)

(e) (f)

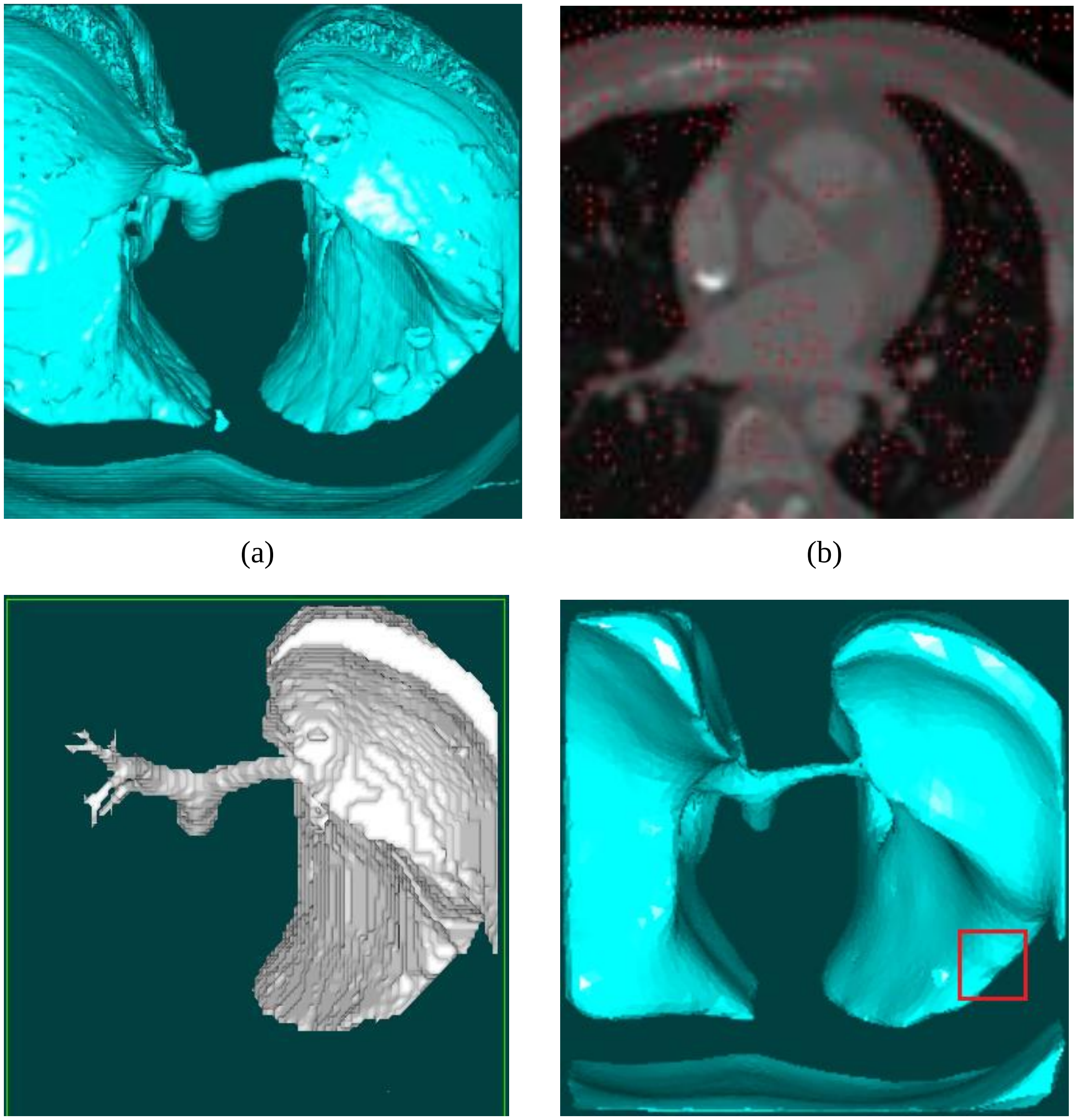
(g) (h)



(i) (j)

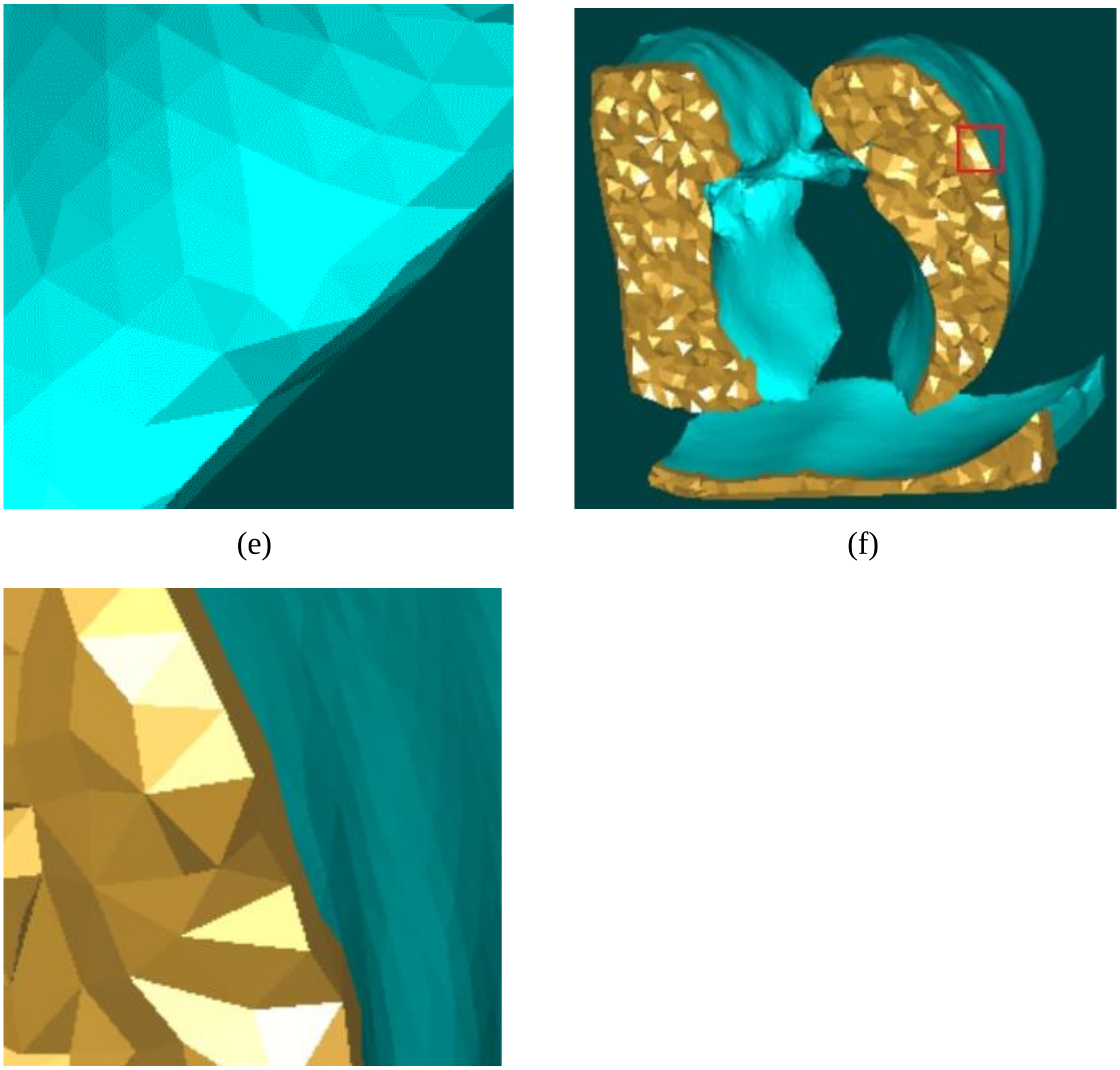
Fig 4.6: is a 136*121*71 very noisy cell 3D volume. (a) and (b) Two slices of the cross sections of this 3D volume. (c) The iso-surface of this 3D cell volume. (d) Slice of the Canny map.(e) The final picked nodes result. (f) The result of classical voxel-based 3D image segmentation. The running time is 24078ms. (g)The result of the supervoxel-based 3D image segmentation. The running time of 3D Canny edge detector is 500ms, The running time of PCA sampling is 985ms. The running time of the generation of the tetrahedral mesh is 703ms. The running time of S-t cut based on the super-voxel is 594ms. Running time of Laplacian smoothing is 31ms (h) A closeup look at the rectangular region in (g). (i)The clipping result of volume mesh structure. (j) A closeup look at the rectangular region in (i).

Fig 4.6 is a very noisy 3D cell volume and our method also works well on this example. Because the size of this cell volume example is much smaller than the previous sphere volume, the speed of the 3D Canny edge detector is much faster than the sphere one. Noisy 3D volume indicates that there are lots of tiny features in this example. As the result, this example generates 29981 nodes in (e) and 187193 tetrahedrons in the mesh. Because of a plenty of nodes, the speed of the PCA sampling, mesh generation and S-t cut is much slower than the sphere one. The overall running time of this example is 2806ms. The classical voxel-based image segmentation is even worse. The running time of it is 24078ms. Our method is around nine times faster than the classical voxel-based segmentation. (g), (h), (i), and (j) shows that the surface of the cell is smooth enough and both volume and surface mesh's quality are great. In this example, our method is much better than the classical voxel-based segmentation in both speed and quality with same seeds given manually in this volume.



(a) (b)

(c) (d)



(e)

(f)

(g)

Fig 4.7: A 128*128*128 thoracic cavity 3D volume. (a) is the iso-surface map of this 3D volume. (b) The final picked nodes result. (c) The result of classical voxel-based 3D image segmentation. The running time is 14422ms. (d)The result of the supervoxel-based 3D image segmentation. The running time of 3D Canny edge detector is 828ms. The running time of PCA sampling is 1422ms. The running time for the generation of the tetrahedral mesh is 1203ms. S-t cut based on the super-voxel is 1031ms. Running time of Laplacian smoothing is 78ms. (e) A closeup look at the rectangular region in (d). (f)The clipping result of volume mesh structure. (g) A closeup look at the rectangular region in (f).

Fig 4.7 is 128*128*128 thoracic cavity 3D volume. Compare with (c) and (d), the result of the super-voxel graph cut method which is introduced in this paper is much better than the classical voxel-based 3D image segmentation. Moreover, the running time of the super-voxel method is

4562ms and the running time of the classical voxel-based 3D image segmentation is 14422ms. Both the speed and quality of supervoxel-based 3D image segmentation method mentioned in this paper is much better than the classical voxel-based one.

Table 4.1: Running time comparison between our supervoxel-based 3D image segmentation methods with the classical voxel-based image segmentation method.

		Sphere	Cells	Thoracic cavity
Number of voxels		2097152	1168376	2097152
Supervoxel-based image segmentation (Method in this thesis)	Number of tetrahedrons	10190	187193	324577
	Running time of Canny(ms)	1438	500	828
	Running time of PCA sampling(ms)	62	985	1422
	Running time of tetrahedral mesh generation(ms)	78	703	1203
	Running time of S-t Cut(ms)	16	594	1031
	Running time of smoothing(ms)	78	31	78
	Running time in all(ms)	1672	2806	4562
Classical voxel-based 3D image segmentation method(ms)		3640	24078	14422

Table 4.1 shows that the speed of supervoxel-based image segmentation method which is much faster than the classical voxel-base image segmentation method in all the cases. Especially when the 3D volume is very noisy which is one of the common attribute of the bio-medical 3D image, the method introduced in this thesis works pretty well. We list the steps from nodes generation, tetrahedral mesh generation, graph cut based on the super-voxels to smoothing.

Moreover, the experimental results also demonstrate that our method can generate very high quality surface and volume meshes on bio-medical 3D volumes

Chapter 5

Image Edge Enhancement and Segmentation via Randomized Shortest Paths

This chapter describes a new method for image edge enhancement and boundary segmentation. Because this method can enhance the edge of the features, it can be used to generate the feature boundary map instead of Canny edge detector in our mesh generation scheme.

Likes many interactive graph-based edge enhancement and segmentation methods, users are asked to provide some foreground (or object) and background seeds. A set of randomly generated points representing the foreground are paired with another set of random points representing the background. The corresponding shortest paths of all the pairs are found and accumulated such that image edges are more likely visited by the shortest paths and thus get enhanced. The final segmentation is obtained from the enhanced edges. Several experiments are provided to demonstrate the effectiveness of the proposed approach.

Our method also treats an input image as a weighted graph and utilizes shortest paths between two pixels. However, our method differs from other approaches in that the source and sink of a path are randomly generated and likely (but not exactly) represent the object and background respectively. Each pixel is associated with an edge confidence (EC) value that is initialized as zero.

Whenever a shortest path is found, all pixels on the path are added by one on their EC value. The final EC map of the whole image gives an enhanced edge map of the original image, and thus the segmentation can be performed by simply running the shortest path twice on the object boundaries. As the number of shortest paths executed affects the quality of the EC map (typically, the more the number of paths, the better the EC map), our method can be thought of as a Monte Carlo method.

Our method is based on the graph constructed from the original image. As most of other graph-based methods, we treat each pixel as a node and the 4-connectivity between pixels as an edge in a graph. First, we define a gradient-dependent value on every pixel in the image is as follows:

$$G(p) = \exp(-0.1 * \|\nabla I\|) \tag{25}$$

where ∇I is the gradient of the input image *I*. Then the edge e_{ij} connecting two vertices p_i and p_j is assigned with the weight as:

$$e_{ij} = \frac{G(p_i) + G(p_j)}{2} \tag{26}$$

With the edge weight defined above, any shortest path between two pixels in the image would try to go through the image edges as the gradient magnitudes are high and hence the edge weights are low.

Different from other graph-based methods, the idea of our method is to use a number of randomized shortest paths to enhance image edges. Every pixel p is associated with a so-called edge confidence (EC) value, which is initialized as zero for every pixel. Whenever a pixel p is visited by one shortest path, the corresponding EC value is accumulated by one. After all the shortest paths are found and applied to the visited pixels, the EC value of a pixel will give a clue of how strong this pixel lies on the image edges, hence producing enhanced edge map of the original image. The key in this process is to determine the end points (source and sink) of the shortest paths, as finding the shortest path between two given points is a standard procedure in graph theory. Below we shall give the details of our algorithm in several steps.

5.1 Seeds Generation

5.1.1 Initial Seed Selection by Users

In this step, users should manually pick some seeds representing foreground (object) and background (non-object) using different colors. Usually the number of the object seeds is less than that of the background seeds. Typically just a few (< 10) object seeds are necessary but they should be distributed evenly inside of the object of interest. The background seeds should be distributed around the object. Figure 5.1 illustrates an example of initial seed selection on an electron microscopic image of cardiac cells. As explained below, these seeds only have high probability to be chosen as the object and background. They may not be used as the sources or sinks of the shortest paths in the segmentation algorithm.

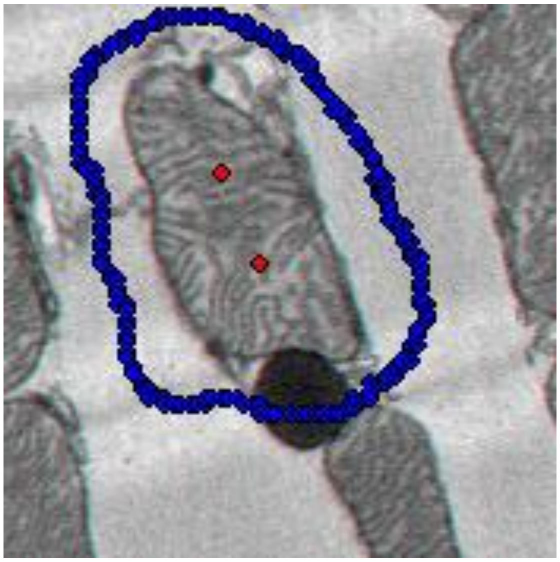


Figure 5.1 Initial seeds selection on an electron microscopic image of cardiac cells (image size: 177*180 pixels).

5.1.2 Generating Marching Distance Maps

The user-selected seeds mentioned above roughly tell us where the foreground and background are in the image. The pairs formed between them, however, may not be statistically sufficient to enhance the edges for accurate segmentation. To add new pairs for the shortest paths, we must know where the end (source or sink) points are and whether they are in foreground or background. The classification of a position (i, j) is achieved by using the marching distance as defined below:

Definition: Given a 2D scalar image *I,* the *marching distance* between two points *A* and *B* in the function domain is defined by [103]:

$$MD_I(A,B) = \min\{ \int_{A \to B} e^{\|\nabla I\|} ds \} \tag{27}$$

where $\int_{A \to B}$ is the integral along a path from A to B. The minimization is conducted over all the paths from A to B. Apparently the marching distance favors a path that goes through low-gradient (or non-edge) regions. In other words, if two points are in the same feature, the marching distance between them should be small.

Given a seed pixel *S* in an image, the marching distance from S to all other pixels is so-called marching distance map, which is similar to the shortest path distance from S to all other pixels in the graph representation of the image. The marching distance map can be efficiently computed by using the fast marching method [104].

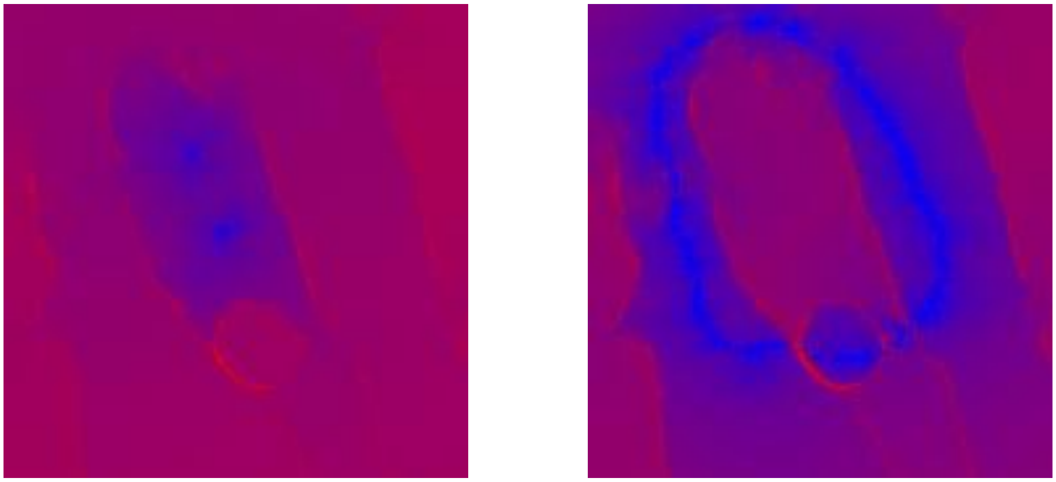

(a) (b)

Figure 5.2: (a) Marching distance map from the user-picked object seeds. (b) Marching distance map from the user-picked background seeds. The seeds are shown in Figure 5.1. The blue color means smaller marching distances and by contrast the red color indicates larger marching distances.

Figure 5.2 shows the marching distance maps by treating the initial object seeds and the initial background seeds as the seeds for the fast marching method [104], where the object and background seeds are picked by a user as shown in Figure 5.2 The colors in both maps are defined as: blue color means small marching distance and red color means larger marching distance. In other words, the blue pixels have higher chance to be classified as objects while red pixels are more likely to be the background in the given image.

5.1.3 Random Seed Selection

The marching distance maps shown in Figure 5.2 are then utilized to randomly generate source and sink points that will be used to form the shortest paths. Basically the value of each pixel in the marching distance map tells us the probability of that pixel being the source or sink point. The lower the value is, the higher the probability would be. Below we briefly explain the steps of generating source points (representing the object).

1. Randomly choose a pixel from the image. We denote this random pixel as P. Let M(P) be the value of P in the marching distance map (see Fig 5.2).
2. We then generate a random number $V(P) \in [0, T]$, where T is a pre-defined threshold. If $M(P) \leq V(P)$, then the pixel P is chosen as a source point (representing the object). Otherwise, ignore P.
3. Repeat (1-2) until sufficient number of source points are chosen.

A similar procedure is applied to generate a set of sink points (representing the background). Figure 5.3 shows an example of source (in red) and sink (in blue) points. Note that because of the random number generation, some source points lie in the background and some sink points lie in the foreground (object) regions. These wrong classification would affect the segmentation. But as we will see below, our final segmentation is a total contribution of all the shortest paths formed by the source and sink points. Just a small number of wrong classification would not make much difference to the final results. To that sense, our method is a Monte Carlo based method.

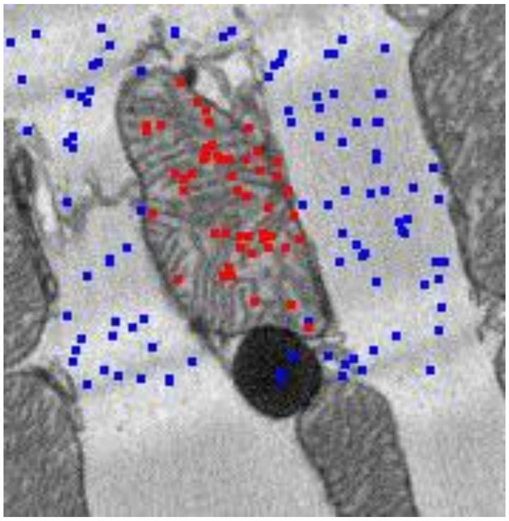


Figure 5.3: The generation of source (in red, representing object) and sink (in blue, representing background) points by using random number generation.

5.2 Image Edge Enhancement by Shortest Paths

The source and sink points generated above (see Figure 5.3 too) are paired and used to find the shortest paths by using the edge weight as defined in Equation (24). For example, if we have n source points and m sink points, then $n \times m$ pairs will be considered and hence $n \times m$ shortest paths will be detected in the graph. The algorithm we use to generate a shortest path is the Shortest Path Faster Algorithm (SPFA).

Figure 5.4 shows the shortest path between one of source (object) point and one of sink (background) points. We can see that this path favors passing through the boundary of the object because according to equation (2), the edge weights are small on the boundary of the object.

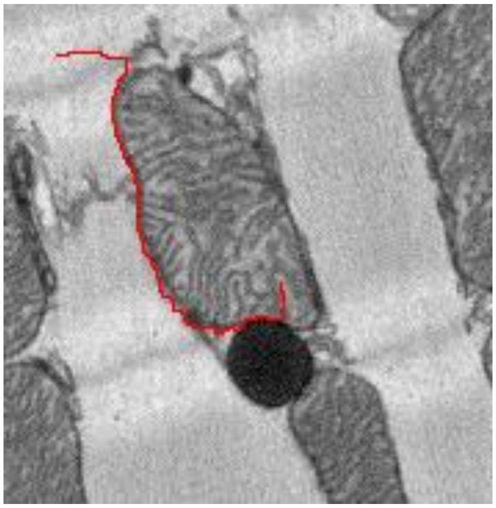


Figure 5.4: The shortest path between one of the source (object) points and one of the sink (background) points. Note the path favors going through the boundary of the object due to the small edge weights on the boundary.

Figure 5.5 shows *m* shortest paths between one of the source points and all the *m* sink points chosen by the random method described above. Although we see many branches on the paths, all paths do prefer to go through the boundary of the object.

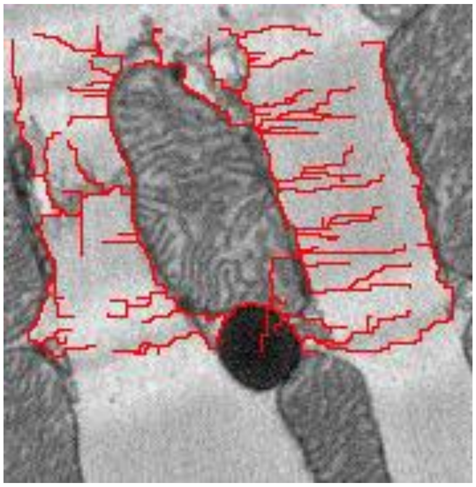


Figure 5.5: Shortest paths between one of the source points (representing the object) and all the sink points (representing the background).

We then create a new image EC(i, j), representing the edge confidence of the pixel (i, j), initialized as zero. For every one of the $n \times m$ shortest path, if a pixel (i, j) is on the path, we add one to its edge confidence value EC(i, j). After all $n \times m$ shortest paths are checked, we will have a gray-scale image EC(i, j), in which a large value indicates a high probability of the pixel being on the object boundary. Figure 5.6 shows such an example after the EC(i, j) image is rescaled to [0, 255]. From this figure we can see that the image edges are enhanced due to the preference of the shortest paths going through the object boundary. It is obvious now that our algorithms works like a Monte Carlo method: the result becomes more and more accurate as the number of samples (in our case , the shortest paths) increases. Also, the more uniform the random numbers are, the better the result would be.

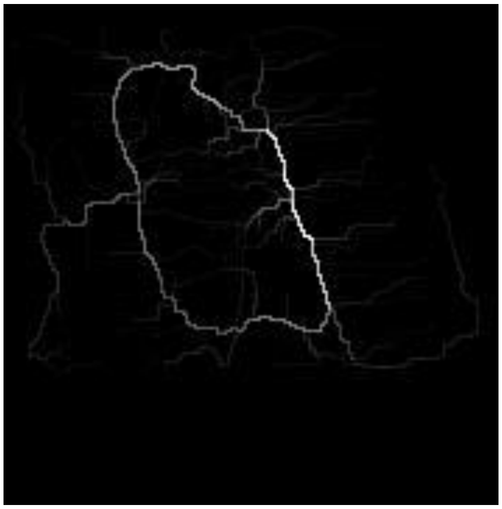


Figure 5.6: The edge confidence (EC) map formed by adding one to pixels where a shortest path goes through.

5.3 Edge Detection from the Enhanced Edges

After the image edges are enhanced by using the randomized shortest paths, we detect the object boundary by running the shortest path one more time. First, we search the edge confidence (EC) image to find out the brightest pixel (denoted by *A*) and we assume that this point be on the boundary of the object. Then we search the neighborhood of this pixel and find out the brightest neighboring pixel (denoted by *B*). Then we form another graph by using the EC image. In this graph, the edge weight is the inverted gray-scale value in the EC image, such that the edges on the object boundary have smaller edge weights. In particular, an edge where one of the end points has zero EC value is assigned the weight infinite. Also, the edge between *A* and *B* mentioned above is also assigned the weight infinite. With these specifications, the shortest path between *A* and *B* in this new graph would result in the segmentation as shown in Figure 5.7 below.

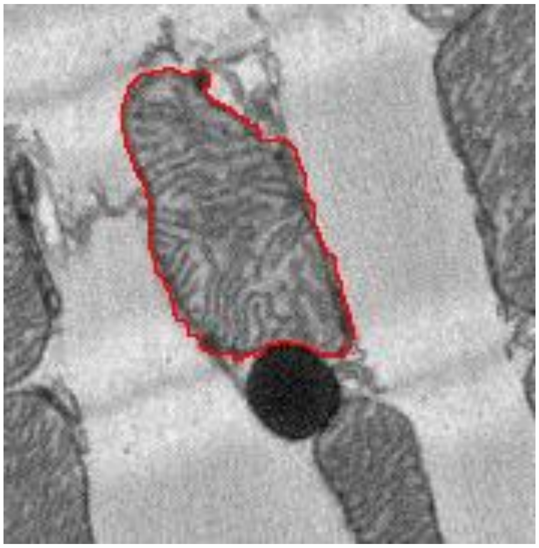
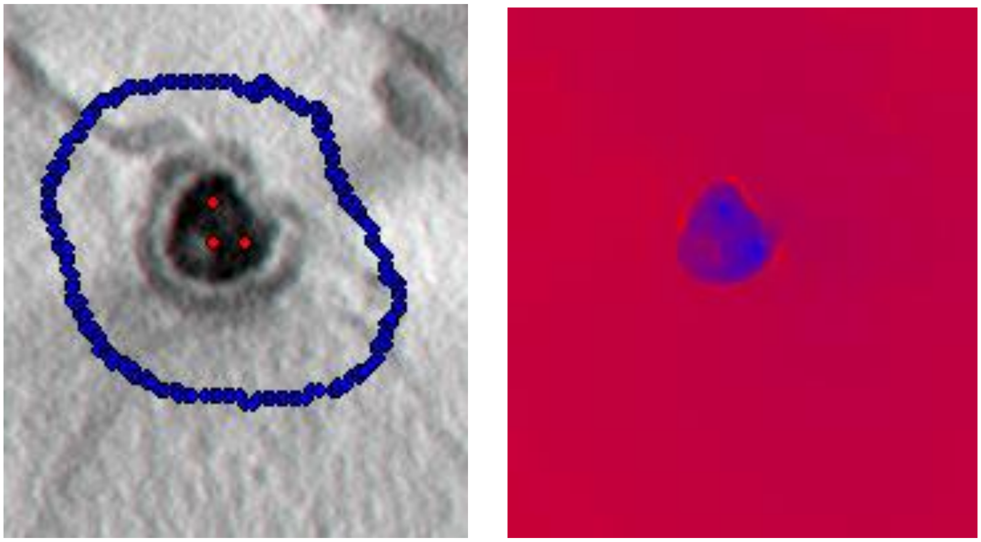


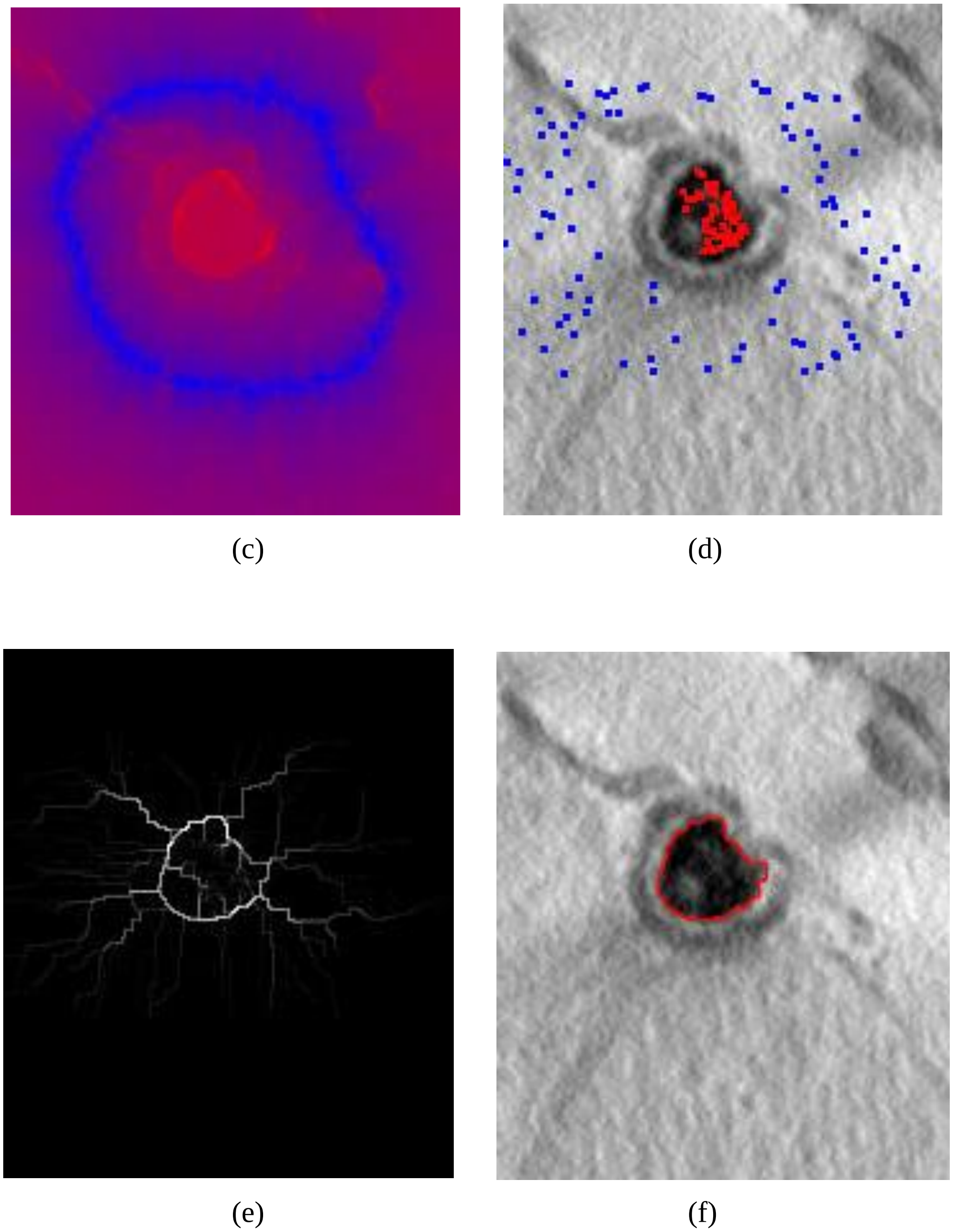
Figure 5.7: Final segmentation result by using the randomized shortest paths, an algorithm based on the Monte Carlo method.

5.4 Experimental Results

In this section we will test the proposed algorithm on several biomedical images to show the effectiveness of our method. Figure 5.8 shows a noisy image of cardiac cell with 264*235 pixels. In Figure 5.8(a) we choose 3 initial object seeds (in red) and a number of initial background seeds (in blue). With these seeds, we generate the marching distance maps for both the object and the background, as shown respectively in Figure 5.8 (b) and (c). These maps are then used to randomly generate the source (in red) and sink (in blue) points, representing the object and background respectively, as seen in Fig 5.8(d). In this example, there are a total of 50 source and 100 sink points. These points are paired and the corresponding 5,000 shortest paths are found in about 1.6 seconds. Figure 5.8(e) and (f) show the rescaled edge confidence image and the final edge detection result of our method.



(a) (b)

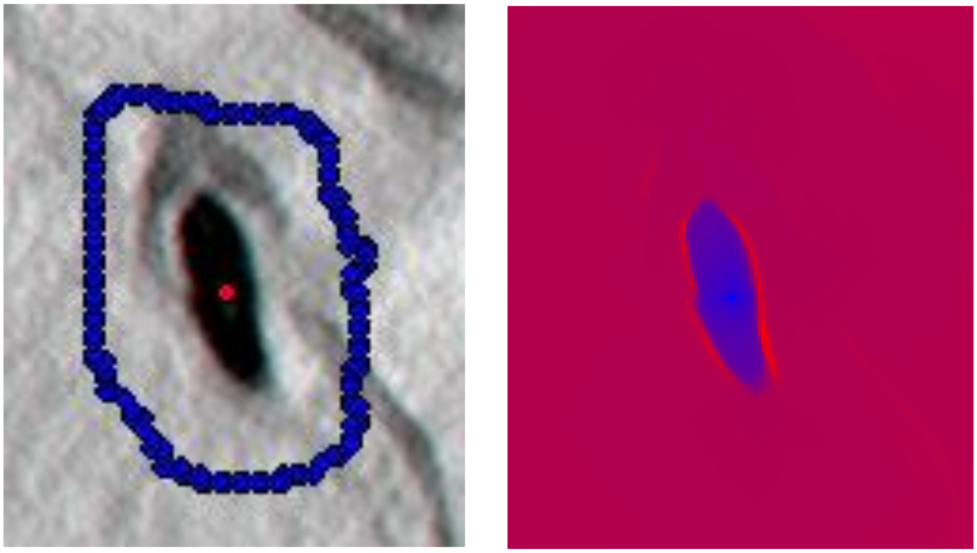


(c) (d)

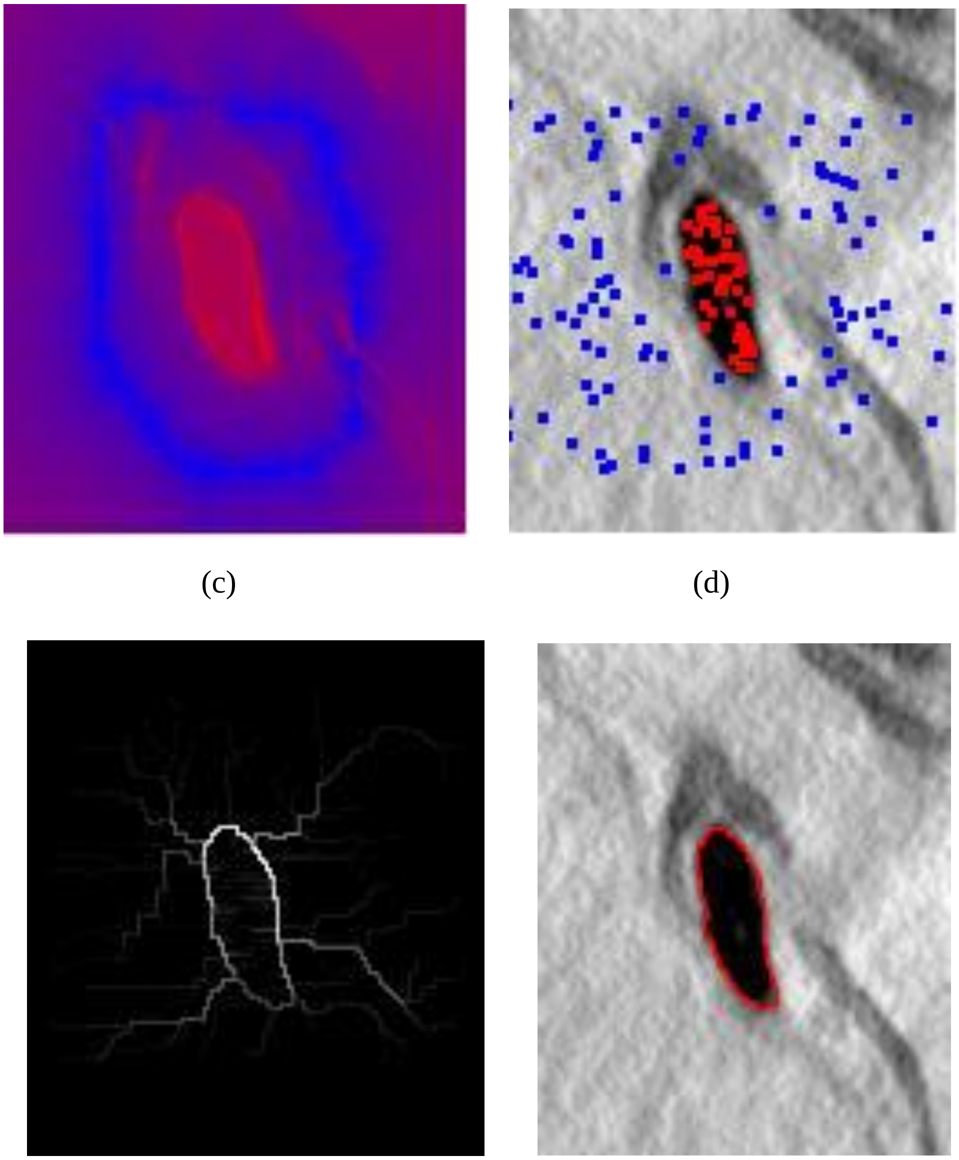
(e) (f)

Figure 5.8 Illustration of our algorithm on a cardiac cell with 264*235 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The computed edge confidence image by adding all shortest paths together. (f) The final edge detection result.

Figure 5.9 shows another example of noisy cardiac cell image with 125*145 pixels. Figure 5.9(a) shows the initial seeds (one object seed and a number of background seeds) picked by the user. Figure 5.9(b) and (c) show the marching distance maps by treating the user-picked seeds as the seeds for the fast marching method. Figure 5.9(d) shows the randomly generated source (red) and sink (blue) points by using the marching distance maps as the probability. The computed edge confidence map that combines all the shortest paths is shown in Figure 5.9(e). Figure 5.9(f) shows that the edge detection result. In this example, we generates 50 source points and 100 sink points, and it takes about 0.64 second to find all the 5,000 shortest paths.



(a) (b)

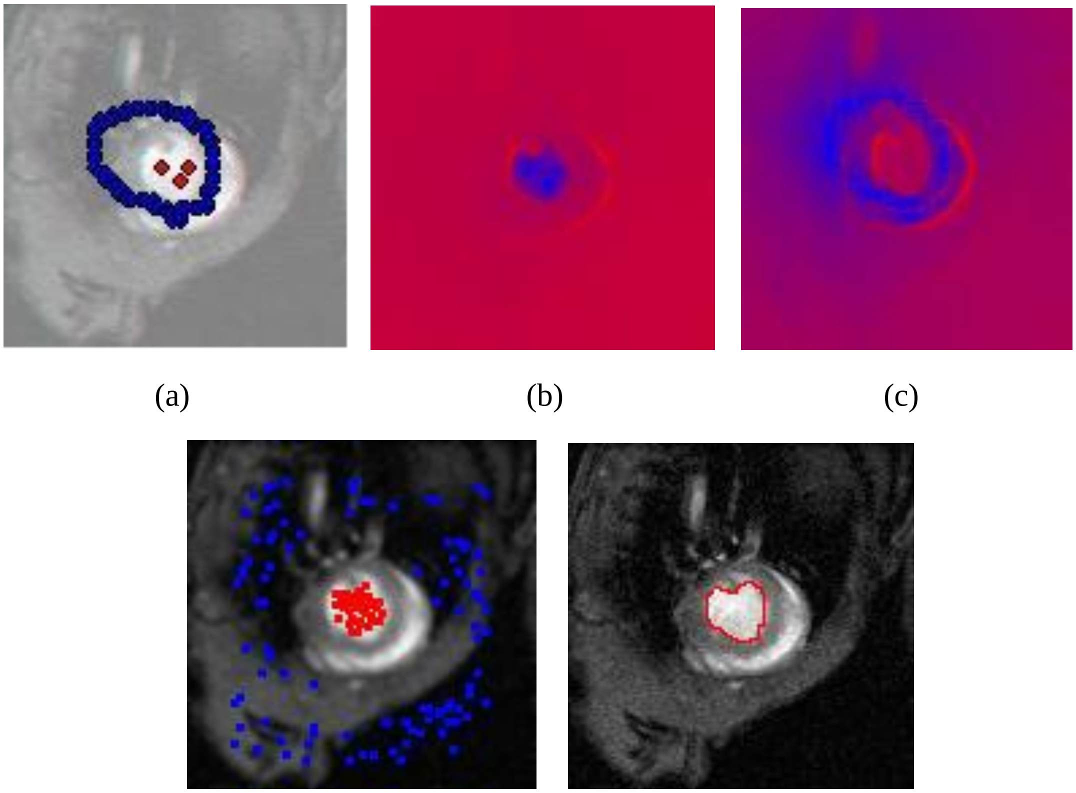


(c) (d)

(e) (f)

Figure 5.9: Illustration of our algorithm on a cardiac cell with 125*145 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The computed edge confidence image by adding all shortest paths together. (f) The final edge detection result.

Figure 5.10 shows how to deal with two adjacent features. The image considered is an MRI image of a mouse's heart. Although the noise is not a big issue in this image, the closely-located features (namely, the left ventricle near the center surrounded by the right ventricle) do impose some difficulties on boundary segmentation using our method. In this example, we randomly generated 50 source (object) points and 100 sink (background) points, and the total time for shortest path detection is 0.29 second.



(a) (b) (c)

(d) (e)

Figure 5.10: Illustration of our algorithm on an MRI image of the heart with 128*128 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The final edge detection result of the left ventricle.

In this chapter, I described a new graph-based algorithm to enhance the edges of a feature in an image by using the shortest paths of random source and sink points. This method can be used to replace the Canny edge detector in our mesh generation scheme. This method likes the traditional Monte Carlo method in that the resulting edge confidence image and hence the final edge detection result become more accurate as the number of random shortest paths increases. Therefore, our approach is very scalable and can be very easily implemented in parallel computing

Chapter 6

Conclusion

2D and 3D mesh generations, and image segmentation are hot research topics in the computer vision and image processing area. How to create the high quality meshes and accurate image segmentation results with a real-time running time become very important. We list a series of the steps from nodes generation, mesh generation to graph cut based on the super-pixels. This algorithm not only results in an accurate image segmentation output, but also generates good quality mesh (both surface and volumetric mesh). Moreover, through the experimental result, the speed of this algorithm is much faster than the speed of classic pipeline and our approach could fulfill the real-time requirement. The experimental results also certificate that our method can generate very high quality surface and volume meshes and accurate image segmentation results on the various bio-medical 2D image and 3D volumes.

References

[1]: Haidekker M.A., Medical Imaging Technology, *SpringerBriefs in Physics*, 2013

[2]: Chandler D. and R.W. Roberson, Bioimaging: Current Concepts In Light & Electron Microscopy, *Jones & Bartlett Learning*, 2008

[3]: E.N. Mortenson and W.A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, pages 349-384, 1998

[4]: C Rother, V Kolmogorov. Grabcut: Interactive foreground extraction using iterated graph cuts . *ACM Transactions on Graphics*, vol. 23, pp. 309–314, 2004

[5]: Gonzalez, R.C. and R.E. Woods, Digital image processing, *Addison-Wesley*, 1992.

[6]: Z. Yu and C. Bajaj, Image Segmentation Using Gradient Vector Diffusion and Region Merging, *in Proceedings of International Conference on Pattern Recognition*, pp. 941-944, 2002.

[7]: R. Malladi, J.A. Sethian, and B.C. Vemuri, Shape modeling with front propagation: A level set approach. IEEE Trans. Pattern Anal. Machine Intell.,. 17(2), pp. 158-175, 1995.

[8]: N. Volkmann, A novel three-dimensional variant of the watershed transform for segmentation of electron density maps. *J Struct Biol*,. 138, pp. 123-129, 2002.

[9]: J. Shi and J. Malik, Normalized cuts and image segmentation. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 731-737, 1997.

[10]: A.S. Frangakis et al., Identification of macromolecular complexes in cryoelectron tomograms of phantom cells. *Proc Natl Acad Sci U S A*, 99(22): pp. 14153-14158, 2002.

[11]: Kass, M., A. Witkin, and D. Terzopoulos, Snakes: active contourmodels. *Int'l J. of Computer Vision*, 1988. 1: p. 321-331.

[12]: Sethian, J.A., Level Set Methods and Fast Marching Methods (2nd edition). 1999: Cambridge University Press.

[13]: Yu, Z. and C. Bajaj, Automatic Ultrastructure Segmentation of Reconstructed Cryo-EM Maps of Icosahedral Viruses. *IEEE Transactions on Image Processing*, 2005. 14(9): p. 1324-1337.

[14]: Y.Boykov, G. Funka-Lea. Graph Cuts and Efficient N-D Image Segmentation, *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109-131, 2006

[15]: A.Delong, Y. Boykov. A Scalable graph-cut algorithm for N-D grids, *Proceeding of Computer Vision and Patten Recognition*, pages 1-8, 2008

[16]: O.Juan, Y. Boykov. Active graph cuts, *Proceeding of Computer Vision and Patten Recognition*, pages 1023-1029, 2006

[17]: L.Grady. Random walks for image segmentation.I*EEE Transactions on Pattern Anal. Machine Intell.*, 28(11):1768–1783, 2006.

[18]: X.Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *Proceedings of International Conference on Computer Vision*, pages 1-8, 2007.

[19]: A.K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Proceedings of International Conference on Computer Vision*, pages 1-8, 2007.

[20]: C.Couprie, L. Najman, L. Grady, and H. Talbot, Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest, In *Proceedings of* International *Conference on Computer Vision*, pages 731-738, 2009.

[21]: Shi,J. and J. Malik, Normalized cuts and image segmentation. Proceedings of IEEE *Conference on Computer Vision and Pattern Recognition* (CVPR), 1997: p. 731-737.

[22]: Shi,J. and J. Malik, Normalized cuts and image segmentation. *IEEE Trans. on Pattern Anal. Machine Intell.*, 2000. 22(8): p. 888-905.

[23]: Stein A., Hoiem D., and Hebert M., Learning to Find Object Boundaries Using Motion Cues. *IEEE International Conference on Computer Vision (ICCV)*, October, 2007. p. 1-8.

[24]: Ren X. and Malik J., Learning a Classification Model for Segmentation . *In Proc. 9th Int. Conf. Comp. Vision*, 2003. p. 10-17.

[25]: Rohkohl, C. and Engel, K.: Efficient Image Segmentation Using Pairwise Pixel Similarites. In *the 29th DAGM Symposium on Pattern Recognition*, F.A. Hamprecht, C. Schnorr, and B. Jahne (Eds.), LNCS 4713, 2007. p. 254–263.

[26]: Ramponi G. and S. Carrato, An adaptive irregular sampling algorithm and its application to image coding, *Image and Vision Computing*, 19(7): 451?60, 2001

[27]: Yang Y., Miles N. Wernick, and Jovan G. Brankov. A fast approach for accurate content-adaptive mesh generation, *IEEE Transactions on Image Processing*, 12(8):866-881, 2003

[28]: Kim T.-S. and W. H. Lee, 3-D MRI and DT-MRI Content-adaptive Finite Element Head Model Generation for Bioelectromagnetic Imaging, *Recent Advances in Biomedical Engineering*, 2009

[29]: Cuadros-Vargas A.J., L. G. Nonato, R. Minghim and T. Etiene, Imesh: An Image Based Quality Mesh Generation Technique, *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*, 2005

[30]: Tu X. and M.D. Adams, Improved Mesh Models of Images Through the Explicit Representation of Discontinuities, *Canadian Journal of Electrical and Computer Engineering*, 36(2): 78 - 86, 2013.

[31]: Garland M., P.S. Heckbert, Fast Polygonal Approximation of Terrains and Height Fields, *CMU-CS-95-181*, 1995

[32]: Adams M.D., A Highly-Effective Incremental/Decremental Delaunay Mesh-Generation Strategy for Image Representation, *Signal Processing*, 93(4): 74964, 2013

[33]: Sarkis M, K. Diepold, Content Adaptive Mesh Representation of Images Using Binary Space Partitions, *IEEE Trans Image Process*., 18(5):1069-79, 2009

[34]: Bougleux, S., Peyre, G., Cohen, L.D., Image Compression with Anisotropic Geodesic Triangulations, *IEEE 12th International Conference Computer Vision*, pages 2343 - 2348, 2009

[35]: Li P., Adams M.D., A Tuned Mesh-Generation Strategy for Image Representation Based on Data-Dependent Triangulation, *IEEE Trans Image Process*., 22(5):2004-2018, 2013

[36]: Demaret L., N. Dyn and A. Iske, Image Compression by Linear Splines over Adaptive Triangulations, *Signal Processing*, 86(7): 1604616, 2006

[37]: L. Demaret and A. Iske, Anisotropic Triangulation Methods in Adaptive Image Approximation, In Approximation Algorithms for Complex Systems, *Springer Proceedings in Mathematics Volume 3*, pp 47-68, 2011

[38]: Adams M.D., A Flexible Content-Adaptive Mesh-Generation Strategy for Image Representation, *IEEE Transactions on Image Processing*, 20(9): 2414 - 2427, 2011

[39]: L. Chen, J. Xu, Optimal Delaunay triangulations, *Journal of Computational Mathematics* 22 (2) (2004) 299–308.

[40]: Chen L., J. Xu, Optimal Delaunay triangulation, *Journal of Computational Mathematics*, 22(2): 299-308, 2004

[41]: Gao Z., Z. Yu, and M. Holst, Quality Tetrahedral Mesh Smoothing via Boundary-Optimized Delaunay Triangulation, *Computer Aided Geometric Design*, 29(9):707-721, 2012

[42]: Gao Z., Z. Yu, and M. Holst, Feature-Preserving Surface Mesh Smoothing via Suboptimal Delaunay Triangulation, *Graphical Models*, 75(1): 23-38, 2013

[43]: Goksel O., and S.E. Salcudean, Image-Based Variational Meshing, *IEEE Transactions on Medical Imaging*, 30(1): 11-21, 2011

[44]: Yang Y., Miles N. Wernick, and Jovan G. Brankov. A fast approach for accurate content adaptive mesh generation, *IEEE Transactions on Image Processing*,12(8):866-881, 2003

[45]: Floyd R. and L. Steinberg, An adaptive algorithm for spatial gray scale, in SID Int. *Symp. Digest of Tech*. Papers, pp. 367, 1975

[46]: Shewchuk J., Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. http://www.cs.cmu.edu/ quake/triangle.html

[47]: Hang Si, A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator http://wias-berlin.de/software/tetgen/

[48]: Chen L., Mesh smoothing schemes based on optimal Delaunay triangulations, *Proceedings of the 13th International Meshing Roundtable*, pages 109-120, 2004

[49]: Chen L., J. Xu, Optimal Delaunay triangulation, *Journal of Computational Mathematics*, 22(2): 299-308, 2004

[50]: Hang Si, A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator http://wias-berlin.de/software/tetgen/

[51]: J.O. Dada, P. Mendes, Multi-scale modelling and simulation in systems biology, *Integrative Biology* 3 (2011) 86–96.

[52]: W.K. Liu, T.R. Lee, A.M. Kopacz, H. Kim, W. Stroberg, H.B.Man, D. Ho, M.K. Kim, J.H. Chung, P. Decuzzi, Multiscale framework for biomedical simulation from molecular dynamics to continuum mechanics, *Journal of the Serbian Society for Computational Mechanics* 5 (2) (2011) 61–80.

[53]: D. Whitley, Analysing molecular surface properties, *Drug Design Strategies: Computational Techniques and Applications*. RSC Drug Discovery, The Royal Society of Chemistry, Cambridge, 2012, pp. 184–209.

[54]: V. d’Otreppe, R. Boman, J.-P. Ponthot, Generating smooth surface meshes from multi-region medical images, *International Journal for Numerical Methods in Biomedical Engineering* 28 (6–7) (2012) 642–660.

[55]: X. Feng, K. Xia, Y. Tong, G.W. Wei, Geometric modeling of subcellular structures, organelles, and multiprotein complexes, *International Journal for Numerical Methods in Biomedical Engineering* 28 (12) (2012) 1198–1223.

[56]: E. Johnson, Y. Zhang, K. Shimada, Estimating an equivalent wall-thickness of a cerebral aneurysm through surface parameterization and a non-linear spring system, *International Journal for Numerical Methods in Biomedical Engineering* 27 (7) (2011) 1054–1072.

[57]: I. Sazonov, P. Nithiarasu, Semi-automatic surface and volume mesh generation for subject-specific biomedical geometries, *International Journal for Numerical Methods in Biomedical Engineering* 28 (1) (2012) 133–157.

[58]: I. Sazonov1, S.Y. Yeo, R.L. Bevan, X. Xie, R. van Loon, P. Nithiarasu, Modelling pipeline for subject-specific arterial blood flow-A review, *International Journal for Numerical Methods in Biomedical Engineering* 27 (12) (2011) 1868–1910.

[59]: T.E. Tezduyar, K. Takizawa, T. Brummer, P.R. Chen, Space-time fluid–structure interaction modeling of patient-specific cerebral aneurysms, *International Journal for Numerical Methods in Biomedical Engineering* 27 (11) (2011) 1665–1710.

[60]: A.B. Albu, T. Beugeling, A morphology-based approach for interslice interpolation of anatomical slices from volumetric images, *IEEE Transactions on Medical Imaging* 55 (8) (2008) 2022–2038.

[61]: S. Valette, J.M. Chassery, R. Prost, Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams, *IEEE Transactions on Visualization and Computer Graphics* 14 (2008) 369–381.

[62]: D.-M. Yan, B. Lévy, Y. Liu, F. Sun, W. Wang, Isotropic remeshing with fast and exact computation of restricted Voronoi diagram, *Computer Graphics Forum* 28 (5) (2009)1445–1454.

[63]: Delaunay, B.: Sur la sphere vide. *Classe des Science Mathematics 16et Naturelle* 7 (1934) 793–800

[64]: Rippa, S.: Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM Journal on Scientific and Statistical Computing* 13 (1992) 1123–1141

[65]: Garland, M., Heckbert, P.: Fast polygonal approximation of terrains and height fields. *Technical Report CMU-CS-95-181*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA (1995)

[66]: P. Alliez, D. Cohen-Steiner, M. Yvinec, M. Desbrun, Variational tetrahedral meshing, *ACM Transactions on Graphics* 24 (3) (2005) 617–625.

[67]: W. Choi, D. Kwak, I. Son, Y. Im, Tetrahedral mesh generation based on advancing front technique and optimization scheme, *International Journal for Numerical Methods in Engineering* 58 (2003) 1857–1872.

[68]: P. Frey, H. Borouchaki, P. George, Delaunay tetrahedralization using an advancing front approach, *Proceedings of the 5th 2International Meshing Roundtable*, 1996, pp. 1–46.

[69]: S.K. Boyd, R. Muller, Smooth surface meshing for automated finite element model generation from 3D image data, *Journal of Biomechanics* 39 (2006) 1287–1295.

[70]: N. Molino, R. Bridson, J. Teran, R. Fedkiw, A crystalline, red green strategy for meshing highly deformable objects with tetrahedra, *Proceedings of the 12th International Meshing Roundtable*, 2003, pp. 3–114.

[71]: M. Yerry, M. Shephard, Automatic three-dimensional mesh generation by the modified-Octree technique, *International Journal for Numerical Methods in Engineering* 20 (11) (1984) 1965–1990.

[72]: Y. Zhang, C. Bajaj, S. Sohn, 3D finite element meshing from imaging data, *Computer Methods in Applied Mechanics and Engineering* 194 (48–49) (2005) 5083–5106.

[73]: F. Labelle, J. Shewchuk, Isosurface stuffing: fast tetrahedral meshes with good dihedral angles, *ACM Transactions on Graphics* 26 (3) (2007), 57, 1–57, 10.

[74]: L.A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, *International Journal for Numerical Methods in Engineering* 40 (21) (1997) 3979 - 4002.

[75]: B.M. Klingner, J.R. Shewchuk, Aggressive tetrahedral mesh improvement, *Proceedings of the 16th International Meshing Roundtable*, 2008, pp. 3–23.

[76]: L.P. Chew, Guaranteed-quality Delaunay meshing in 3D, *Proceedings of the 13th Annual Symposium on Computational Geometry*, 1997, pp. 391–393.

[77]: J.M. Escobar, R. Montenegro, G. Montero, E. Rodríguez, J.M. Gonzáez-Yuste, Smoothing and local refinement techniques for improving tetrahedral mesh quality, *Computers and Structures* 83 (28–30) (2005) 2423–2430.

[78]: D. Nave, N. Chrisochoides, L.P. Chew, Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains, *Computational Geometry: Theory and Applications* 28 (2–3) (2004) 191–215.

[79]: R.E. Bank, R.K. Smith, Mesh smoothing using a posteriori error estimates, *SIAM Journal on Numerical Analysis* 34 (3) (1997) 979–997.

[80]: L.A. Freitag, on combining Laplacian and optimization-based mesh smoothing techniques, *Trends in Unstructured Mesh Generation* (1997) 37–43.

[81]: D.A. Field, Laplacian smoothing and Delaunay triangulations, *Communications in Applied Numerical Methods* 4 (6) (1988) 709–712.

[82]: P. Hansbo, Generalized Laplacian smoothing of unstructured grids, *Communications in Numerical Methods in Engineering* 11 (5) (1995) 455–464.

[83]: L.R. Herrmann, Laplacian-isoparametric grid generation scheme, *Journal of the Engineering Mechanics Division* 102 (5) (1976) 749–907.

[84]: H. Xu, T.S. Newman, An angle-based optimization approach for 2D finite element mesh smoothing, *Finite Elements in Analysis and Design* 42 (2006) 1150–1164.

[85]: Z. Yu, M.J. Holst, J.A. McCammon, High-fidelity geometric modeling for biomedical applications, *Finite Elements in Analysis and Design* 44 (11) (2008) 715–723.

[86]: T. Zhou, K. Shimada, An angle-based approach to two-dimensional mesh smoothing, *Proceedings of the 9th International Meshing Roundtable*, 2000, pp. 373–384.

[87]: Q. Du, D. Wang, Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations, *International Journal for Numerical Methods in Engineering* 56 (2003) 1355–1373.

[88]: Y. Zhang, T.J.R. Hughes, C.L. Bajaj, An automatic 3D mesh generation method for domains with multiple materials, *Computer Methods in Applied Mechanics and Engineering* 199 (5–8) (2010) 405–415.

[89]: L. Rineau, M. Yvinec, A generic software design for Delaunay refinement meshing, *Computational Geometry: Theory and Applications* 38 (2007) 100–110.

[90]: H. Si, K. Gartner, Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations, *Proceedings of the 14th International Meshing Roundtable*, 2005.

[91]: Y. Zhang, C. Bajaj, S. Sohn, 3D finite element meshing from imaging data, *Computer Methods in Applied Mechanics and Engineering* 194 (48–49) (2005) 5083–5106.

[92]: L. Rineau, M. Yvinec, A generic software design for Delaunay refinement meshing, *Computational Geometry: Theory and Applications* 38 (2007) 100–110.

[93]: Wang, J., Liu, G.: On the optimal shape parameters of radial basis functions used for 2d meshless methods. *Computer Methods in Applied Mechanics and Engineering* 191 (2002) 2611–2630

[94]: Casciola, G., Montefusco, L., Morigi, S.: Edge-driven image interpolation using adaptive anisotropic radial basis functions. *Journal of Mathematical Imaging and Vision* 36 (2010) 125–139

[95]: Casciola, G., Lazzaro, D., Montefusco, L., Morigi, S.: Shape preserving surface reconstruction using locally anisotropic RBF interpolants. *Computers & Mathematics with Applications* 51 (2006) 1185–1198

[96]: Casciola, G., Montefusco, L., Morigi, S.: The regularizing properties of anisotropic radial basis functions. *Applied Mathematics and Computation* 190 (2007) 1050–1062

[97]: M. Xu and Z. Yu, "3D Image Segmentation Based on Feature-Sensitive and Adaptive Tetrahedral Meshes", *IEEE International Conference on Image Processing*, Phoenix, 2016.

[98]: M. Xu, Z. Gao, and Z. Yu, "Feature-Sensitive and Adaptive Mesh Generation of Grayscale Images", *The 4th Conference on Computational Modeling of Objects Presented in Images: Fundamentals, Methods and Applications*, Pittsburgh, LNCS 8641, pages 204-215, 2014.

[99]: K. Liu, M. Xu, and Z. Yu, "Feature-preserving Image Restoration from Adaptive Triangular Meshes", *ACCV Workshop on Emerging Topics on Image Restoration and Enhancement*, Singapore, 2014.

[100]: M. Xu, J. Wang, and Z. Yu, "Image Edge Enhancement and Segmentation via Randomized Shortest Paths", *The 5th International Conference on BioMedical Engineering and Informatics*, pages 290-294, China, 2012.

[101]: Z. Yu, M. Xu, and Z. Gao, "Biomedical Image Segmentation via Constrained Graph Cuts and Pre-segmentation", *Proc. of the 33rd Int'l Conf. of IEEE Engineering in Medicine and Biology Society*, pages 5714-5717, Boston, 2011.

[102]: Z. Yu, J. Wang, Z. Gao, M. Xu, and M. Hoshijima, "New Software Developments for Quality Mesh Generation and Optimization from Biomedical Imaging Data", *Computer Methods and Programs in Biomedicine*, 2013.

[103]: Z. Yu and C. Bajaj, *Automatic Ultrastructure Segmentation of Reconstructed Cryo-EM Maps of Icosahedral Viruses.* IEEE Transactions on Image Processing,. 14(9): pp. 1324-1337, 2005.

[104]: R. Malladi and J.A. Sethian. A real-time algorithm for medical shape recovery, in *Proc. of Int'l Conf. on Computer Vision*, pp. 304–310, 1998

[105]: Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision, *IEEE Transactions on PAMI*, Vol. 26, No. 9, pp. 1124-1137, Sept. 2004

[106] Baghaie, Ahmadreza, Roshan M. D'souza, and Zeyun Yu. "Sparse and low rank decomposition based batch image alignment for speckle reduction of retinal OCT images." *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI).* IEEE, 2015.

[107] Baghaie, Ahmadreza, Zeyun Yu, and Roshan M. D'Souza. "State-of-the-art in retinal optical coherence tomography image analysis." *Quantitative imaging in medicine and surgery 5.4 (2015):* 603.

[108] Baghaie, Ahmadreza, and Zeyun Yu. "Structure tensor based image interpolation method." *AEU-international Journal of Electronics and Communications 69.2 (2015):* 515-522.

[109] Baghaie, Ahmadreza, Zeyun Yu, and Roshan M. D'souza. "Fast mesh-based medical image registration." *International Symposium on Visual Computing.* Springer International Publishing, 2014.

[110] Baghaie, Ahmadreza, Roshan M. D'souza, and Zeyun Yu. "Application of Independent Component Analysis techniques in speckle noise reduction of retinal OCT images." *Optik-International Journal for Light and Electron Optics 127.15 (2016):* 5783-5791.

[111] Baghaie, Ahmadreza, Roshan M. D'Souza, and Zeyun Yu. "Dense correspondence and optical flow estimation using gabor, schmid and steerable descriptors*." International Symposium on Visual Computing.* Springer International Publishing, 2015.

[112] Baghaie, Ahmadreza. "Markov Random Field Model-Based Salt and Pepper Noise Removal." *arXiv preprint arXiv*:1609.06341 (2016).

[113] Baghaie, Ahmadreza, and Zeyun Yu. "Curvature-based registration for slice interpolation of medical images." *International Symposium Computational Modeling of Objects Represented in Images*. Springer International Publishing, 2014.

Appendices

Appendix A: Publications

[1] M. Xu and Z. Yu, "3D Image Segmentation Based on Feature-Sensitive and Adaptive Tetrahedral Meshes”, *IEEE International Conference on Image Processing*, Phoenix, 2016.

[2] M. Xu, Z. Gao, and Z. Yu, "Feature-Sensitive and Adaptive Mesh Generation of Grayscale Images*", The 4th Conference on Computational Modeling of Objects Presented in Images: Fundamentals, Methods and Applications*, Pittsburgh, LNCS 8641, pages 204-215, 2014.

[3] K. Liu, M. Xu, and Z. Yu, "Feature-preserving Image Restoration from Adaptive Triangular Meshes", *ACCV Workshop on Emerging Topics on Image Restoration and Enhancement*, Singapore, 2014.

[4] M. Xu, J. Wang, and Z. Yu, "Image Edge Enhancement and Segmentation via Randomized Shortest Paths", *The 5th International Conference on BioMedical Engineering and Informatics*, pages 290-294, China, 2012.

[5] Z. Yu, M. Xu, and Z. Gao, "Biomedical Image Segmentation via Constrained Graph Cuts and Pre-segmentation", *Proc. of the 33rd Int’l Conf. of IEEE Engineering in Medicine and Biology Society*, pages 5714-5717, Boston, 2011.

[6] Z. Yu, J. Wang, Z. Gao, M. Xu, and M. Hoshijima, "New Software Developments for Quality Mesh Generation and Optimization from Biomedical Imaging Data", *Computer Methods and Programs in Biomedicine*, 2013.

Curriculum Vitae

MING XU

xuminguwm@gmail.com

Cell: (414)334–3190

EDUCATION:

- **University of Wisconsin-Milwaukee, Ph.D. candidate**

  Computer Science

  GPA: 3.769/4.0, September 2010 - 2016

  Concentration in biomedical modeling and visualization, 2D and 3D image segmentation, 3D surface and volumetric meshes generation and image processing.

  Minor: Mathematical Science

- **University of Shanghai for Science and Technology, ME**

  Computer Application Technology

  September 2006 - February 2009

- **Fudan University**

  Computer Science and Technology

  September 2001 - July 2005

SKILLS:

- C
- C++
- Java
- OpenGL 4.0+, Shader
- OOP Design, MVC Programming, Git, MFC, SQL, Html, Linux, UI Design (MFC, GLUT), Unit Testing (JUnit), Visual Studio, GCC, Eclipse, LISP

PROFESSIONAL EXPERIENCE:

Biomedical Modeling and Visualization Laboratory (2012.05-2014.10)

University of Wisconsin-Milwaukee

- Software development and maintenance: Bimos (C++, MFC and OpenGL) which is for generating and optimizing surface and volumetric meshes from three-dimensional (3D) biomedical imaging data.

Research Assistant:

Biomedical Modeling and Visualization Laboratory (2010.09-2011.08)

University of Wisconsin-Milwaukee

- Designed and implemented (C++) 2D image segmentation method of the project "Image-based modeling of Ca2+ signaling in ventricular myocytes" funded by National Institutes of Health-National Heart, Lung and Blood Institute.
- Assisted in surface and volumetric meshes generation.

Biomedical Modeling and Visualization Laboratory (2010.09-2016.05)

University of Wisconsin-Milwaukee

- Designed and implemented (C++) SPFA shortest path based 2D image segmentation method.
- Improved and implemented (C++) a feature-preserving anisotropic filter.
- Designed and implemented (C++) multi-object 2D and 3D image segmentation method based on Fast Marching.

ACADEMIC APPOINTMENTS:

Teaching Assistant:

University of Wisconsin-Milwaukee (2011.09-2016.05)

- **CS250: Introductory Computer Programming in Java**
  (Spring 2013, Fall 2013, Spring 2014, Fall 2014, Fall 2015, Spring 2016)

This course seeks to teach its students basic programming skills using a structured high-level language (Java) including basic input and output, variables of the primitive types, control statements, conditionals (if, if-else), iteration (while, for, do-while), user-defined method, arrays, sorting and binary search.

- **CS351: Data Structures and Algorithms in Java**

  (Spring 2014, Spring 2015)

  This course teaches programming in a structured, high-level, object-oriented language. Implementation of data structures and algorithms and their application including ADTs introduction, dynamic array, iterators, linked list, generics, stacks and queues, trees, hashing, graphs and sorting.

- **CS459: Fundamentals of Computer Graphics in OpenGL**

  (Fall 2011, Spring 2012, Fall 2012, Spring 2013, Fall 2013, Fall 2014, Fall 2015)

  This course uses Visual Studio C/C++ and an industry-standard graphics package OpenGL to introduce students the principles of mathematics of geometric transformations, viewing, illumination, splines, and certain implementation algorithms including line-drawing and scan-line fill of convex polygons, clipping algorithm, basic illumination models, surface lighting effects, quadric surfaces, texture mapping and Bezier curves.

ACHIEVEMENTS:

UW-Milwaukee Annual Scholarship	$20,000
2013 Chancellor Award	$2,000
2014 CEAS Dean’s Scholarship	$2,000
2015 CEAS Dean’s Scholarship	$1,000