3D Segmentation of Soft Tissues by Flipping-free Mesh Deformation

PhD Thesis
Submitted to School of Computing

by

Ding Feng (HT040297J)

supervised by

Dr. Leow Wee Kheng (Associate Professor)

School of Computing
National University of Singapore

October 2010

Dedication

To my father Beiping Ding,
my mother Pingping Wu,
my wife Wenxian Yang,
and my daughter Simeng Ding.

Acknowledgement

I would like to give my sincere thankfulness to my supervisor A/Prof. Leow Wee Kheng for his extremely patient and professional guidance and continuous encouragement throughout my PhD study, as well as his invaluable comments on my research and this thesis.

At the same time, I would like to express my gratitude to Dr. Terence Sim and Dr. Ng Teck Khim. Their lectures inspired my interest in computer vision.

I am very grateful to Prof. Chua Tat Seng, Prof. Mohan Kankanhalli and Dr. Terence Sim for their constructive comments on my GRP and thesis proposal. I would also like to thank Dr. Michael S. Brown for his invaluable comments on my research.

I would like to thank Dr. Wang Shih-Chang, Dr. Sudhakar Venkatesh and Dr. Borys from Department of Diagnostic Radiology (DDR) of National University of Singapore (NUS) and National University Hospital (NUH), Dr. Tian Qi and Dr. Zhou Jiayin from Institute for Infocomm Research (I^2R) and Dr. Howe Tet Sen from Singapore General Hospital (SGH) for their invaluable comments on my research.

I would like to thank Chen Ying, Wang Ruixuan, Zhang Sheng, Miao Xiaoping, Piyush Kanti Bhunre, Saurabh Garg, Zhang Xiaopeng, Sheng Chang, Li Hao, Ehsan Rehman and all the other lab mates and friends. I enjoyed the precious moments staying with them.

I appreciate all the staff members in School of Computing and DDR in NUS for their continuous support.

Finally, I am eternally indebted to my family members for their love and support, which words cannot describe.

Abstract

Medical image segmentation has been a very hot research topic over many years. In general, it is a highly challenging problem. Medical images usually have inhomogeneous voxel intensities. Boundaries of target objects may be indistinct in some regions. The shapes of the target objects can be very complex in 3D, and they may have large variance across different patients. Moreover, medical volume images usually contain 50 to 100 million voxels per data set, which is very challenging for a segmentation algorithm. Many existing segmentation algorithms are often plagued by the problems mentioned above. They tend to produce undesired segmentation results. Many of them resort to a global shape constraint, which enable the segmentation result to resemble a normal shape in such low contrast regions. This strategy succeeds when the shapes of the target objects are regular, i.e., close to the normal shape. However, shapes of soft organs are highly variable across different patients. They are in general very difficult to be modelled statistically even with a large number of training samples because the shape variations have huge number of degrees of freedom. With limited number of training samples, they usually cannot achieve accurate results when segmenting such very different shapes.

This thesis presents a novel approach to the segmentation of soft tissues in 3D volume images. The proposed approach uses a specially designed 3D quadrilateral mesh to explicitly represent and segment an object, which is much more efficient compared to voxel-based segmentation algorithms. Segmentation is achieved by evolving the mesh to register to the desired object boundary. The mesh evolution-based segmentation is significantly more efficient than volumetric approaches. The proposed algorithm does not require any shape constraints, and is flexible for segmenting target organs with large shape variations among patients.

Test results on using the single-object segmentation algorithm to segment various abdominal organs show that the proposed algorithm achieved higher accuracy than other segmentation algorithms such as snake, level set and graph-cut in segmenting individual organs. It is also more time efficient.

The proposed approach can be extended to segmenting multiple organs simultaneously. As the meshes for different organs constraint each other, the proposed approach is free from the over-segmentation problem. It has no leaking problem and is more noise

resilient.

Test results on the multiple-object segmentation algorithm demonstrate that it is able to segment multiple objects simultaneously and to improve the segmentation accuracy by overcoming the leakage problem that may happen in single-object segmentation.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

Segmentation is a very crucial and the most fundamental stage in medical imaging work flow, which may include other stages such as quantification, visualization and simulation. It is not only because segmentation is the very first stage in the work flow, but also because the accuracy of segmentation will affect greatly the accuracy of subsequent stages. Therefore, research about medical image segmentation is of particular importance.

In general, segmentation of medical images is a very difficult and challenging task. As shown in a 2D abdominal CT slice (Fig. 1.1), pixel or voxel intensities are often inhomogeneous even within the target object (blue dashed box). This suggests that intensity values of the target object can not be modelled easily. Object boundaries at some locations may be indistinct (red solid boxes). Such inhomogeneous regions and indistinct boundaries either prevent these algorithms from capturing fully the target organs or cause the algorithms to leak out of the target region.

The problem is even more challenging for segmenting 3D soft tissues in 3D volume images, because soft tissues in 3D have complex shapes in general. Many soft tissues including brain, liver, kidney etc., contain deeply concave part. Moreover, the 3D shapes of soft tissues may vary greatly from patient to patient.

Many existing segmentation algorithms are based on local features. These algorithms are fast and easy to use, but are prone to over-segmentation. An over-segmented result produced by the watershed algorithm is shown in Fig. 1.2(b), where colored regions

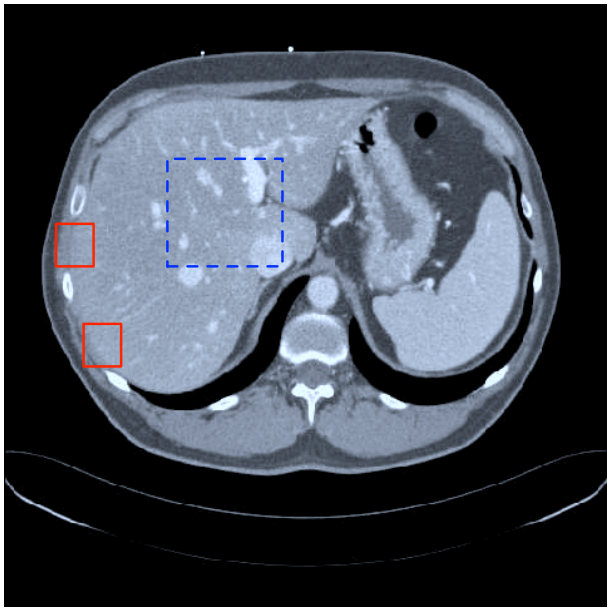


Figure 1.1: Medical image characteristics. Blue dashed box: pixel intensities inside the target object (liver) are highly inhomogeneous. Red solid boxes: object boundaries in some locations are indistinct.

represent different image segments.

In comparison, deformable model-based algorithms usually have one model for one target object. If these algorithms are topology-preserving, the segmentation result will contain a single region only. Therefore, they are free from over-segmentation. These models, if not deformed properly, will not stop at the boundary location of the target object. If the model infiltrates into the neighboring objects, the *leakage* problem occurs. This can be illustrated by Fig. 1.3, where the segmented liver boundary (solid red curve) infiltrates into the kidney region.

Many existing image segmentation methods segment only one single target object. These methods include region growing, classification and active contours, active shape and appearance models, etc. These methods are usually very efficient in terms of temporal complexity. However, these segmentation methods have several intrinsic limitations which may have problems in segmentation of complex medical images. Firstly, they usually assume that the region to be segmented is homogeneous inside and inhomogeneous at its boundary. However, this assumption is often invalid for many complex medical

(a) (b)

Figure 1.2: Result of the watershed algorithm. Over-segmentation is clearly visible. (a) The input image. (b) The segmentation result (from `http://www.itk.org/HTML/WatershedSegmentationExample.html`).

images. As discussed above, Fig. 1.1 shows that the region inside the liver is highly inhomogeneous, and some boundaries between liver and neighboring organs are almost indistinct. Such inhomogeneous regions and indistinct boundaries either prevent these algorithms from capturing the target organs fully or cause the algorithm to leak out. Secondly, some methods such as active shape models rely on shape priors to aid segmentation. Nevertheless, these methods require a large number of training samples. This is often impractical for anatomical structures such as soft tissues since their shapes are highly variable.

Some image segmentation methods such as thresholding, graph cut, etc., can segment multiple regions at the same time. Segmentation results of these segmentation methods may contain multiple regions. However, these regions are of the same properties. They have either similar intensity distributions or similar texture patterns. On the contrary, different target organs may not exhibit similar properties which can be handled by these algorithms.

Some segmentation methods deal with multiple objects at the same time. A fraction of them segment multiple objects one by one. Each object is in fact segmented using the same algorithm such as region growing, classification and active contours, etc. Such multiple-object segmentation algorithms are natural transitions from the single-object segmentation algorithms. They are easy to be implemented and intuitive. However, inter-object relationships are not taken into account during segmentation, such that seg-

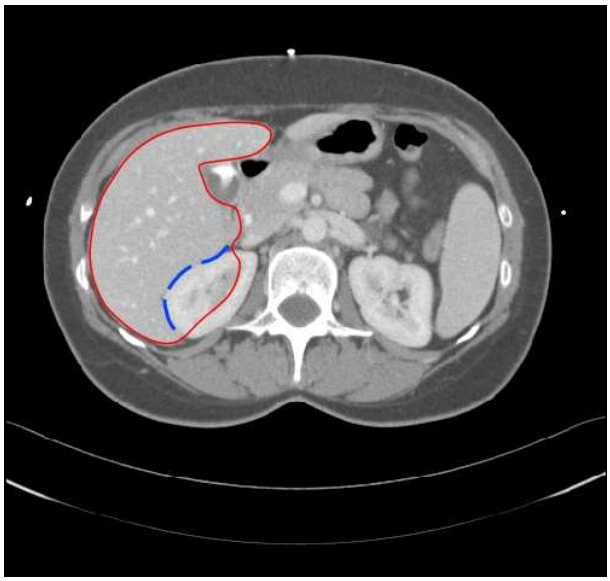


Figure 1.3: Leakage problem. The segmented liver boundary (solid red curve) leaks out of its real boundary (dashed blue curve) into the kidney region.

mentation for each object is not well constrained. In comparison, some methods consider multiple-object segmentation as a whole. All the target objects are segmented simultaneously, and overlap between different objects are discouraged during the segmentation process. These methods try to solve the medical image segmentation problem in a global perspective. These methods are in general more promising to solve the complex medical image segmentation problem because more information is utilized to constrain segmentation. Existing segmentation methods often impose certain shape priors on each target object. They also require a large number of training samples and are difficult to segment objects with highly variable shapes.

There also exist several commercial systems for medical image segmentation, for instance, PathFinder (Fig. 1.4) and IntraSense (Fig. 1.5) for liver segmentation. Detailed algorithms and source code for most commercial systems are not available. From the users' point of view, they are very similar to the region growing algorithm with multiple initial seeds. They are fast and can be implemented easily. The initial segmentation results are usually crude, and require manual touch-ups. The touch-up stage usually includes adding more seeds to increase the target region, or remove some regions that do not belong to the target. Such a stage sometimes takes a considerable amount of time for

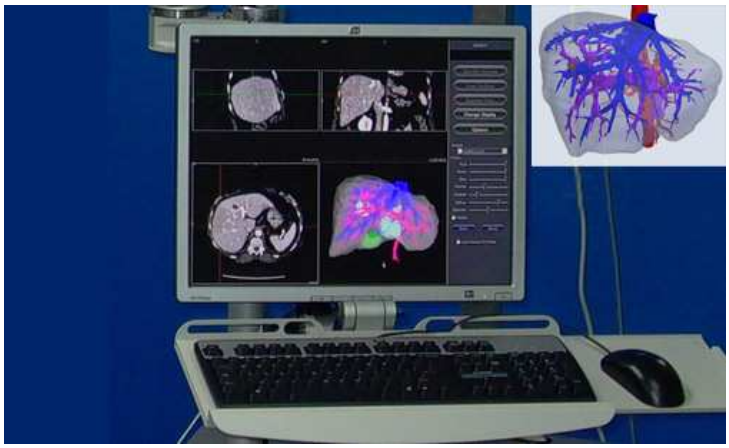


Figure 1.4: PathFinder. Image from `http://www.pathsurg.com`.

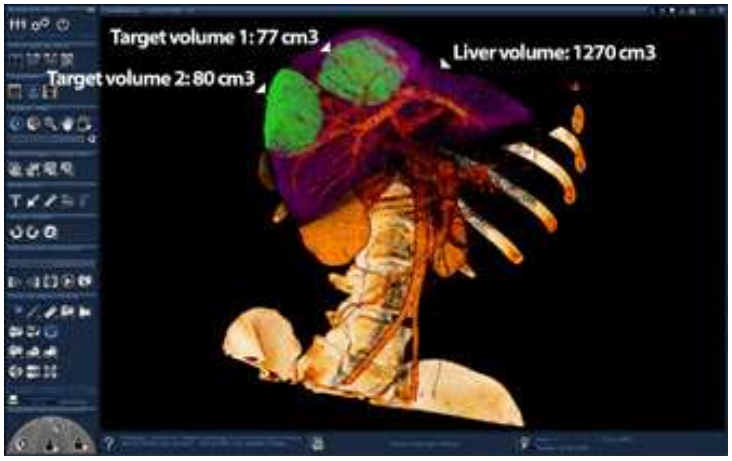


Figure 1.5: IntraSense Myrian software. Image from `http://www.intrasense.fr`.

users to get an accurate final result. Open source medical image segmentation systems, e.g., ITK-SNAP [YPCH$^+$06], provides its users with a semi-automatic segmentation environment based on the level set method. Its level set implementation will be used as a comparison to the proposed algorithm.

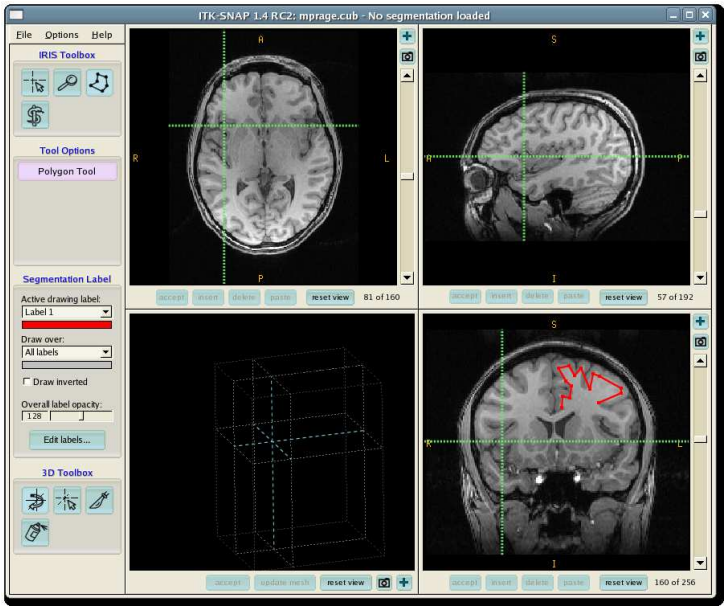


Figure 1.6: ITK-SNAP. Image from `http://www.itksnap.org`.

1.2 Thesis Objectives

To overcome the limitations of existing segmentation methods, this thesis presents a novel approach to the segmentation of soft tissues in 3D volume images. The proposed approach uses a 3D mesh to explicitly represent and segment an object, which is much more efficient compared to voxel-based segmentation algorithms. Segmentation is achieved by evolving the mesh to register to the desired object boundary. The mesh evolution-based segmentation is significantly more efficient than volumetric approaches. The proposed algorithm does not require any shape constraints, and is flexible for segmenting target organs with large shape variations among patients. In addition, the proposed approach can be extended to segmenting multiple organs simultaneously. As the meshes for different organs constraint each other, the proposed approach is free from the over-segmentation problem. It has no leaking problem and is more noise resilient.

The major contributions of this research include the following:

- Developed an efficient flipping-free mesh deformation algorithm based on Laplacian mesh deformation.

- Applied the mesh deformation algorithm to efficiently segment soft organs in medical volume images. The algorithm can be applied to the segmentation of soft organs of various shapes.
- Extended the segmentation algorithm to segmenting multiple target objects simultaneously. The algorithm constructs a deformation band for each target object so that inter-object mesh intersection can be avoided. It prevents the segmented region of one object leaking into another.

1.3 Thesis Organization

To clarify the intrinsic problem of 3D mesh deformation, Chapter 2 presents some traditional mesh editing and deformation algorithms and possible problems during mesh deformation. In Chapter 3, a detailed review of existing medical image segmentation algorithms is presented. The strength and weakness of these methods are discussed. To overcome the weakness of existing medical image segmentation algorithms, this thesis presents a novel 3D segmentation algorithm using flipping-free mesh deformation. Chapter 4 presents the flipping-free mesh deformation algorithm based on a specially designed quadrilateral mesh model. This quadrilateral mesh facilitates the detection and avoidance of flippings during mesh deformation. Chapter 5 presents the 3D segmentation algorithm based on the flipping-free mesh deformation algorithm. The algorithm can be applied to segmenting soft organs of various shapes. The segmentation algorithm is extended in Chapter 6 to segment multiple soft organs in volume images simultaneously. Chapter 7 concludes this thesis and discusses possible future works for the current algorithm.

Chapter 2

Mesh Editing and Deformation

Mesh deformation is an important component of the proposed segmentation method. In computer graphics, 3D mesh is manipulated by mesh editing algorithms to change its shape, resulting in mesh deformation. This chapter reviews existing 3D mesh editing methods and a tricky issue relating to mesh deformation, i.e., self-intersection of mesh (Section 2.2).

2.1 Generic Mesh Editing Methods

Many mesh editing methods have been proposed in the computer graphics community, among which free-form deformation-based methods and differential geometry-based methods are the most widely adopted due to their efficiency and ease of use in 3D object modelling.

2.1.1 Free-form Deformation

Free-form deformation (FFD) [SP86] deforms a 3D object by altering its underlying 3D space enclosing the object. The 3D space is sub-divided into parallelpiped regions. The vertices of these regions function as control points. The deformation of mesh is specified by displacing the control points to some new locations. The deformed mesh vertices are then computed based on a trivariate tensor product of Bernstein polynomial. FFD can work with surface mesh of any degrees, and is in general easy to use. However,

the deformation is based on moving the control points that are usually not on the mesh surface. This makes complex deformation of mesh vertices difficult.

In order to ease this problem, and make FFD method more intuitive, direct manipulation of FFD (DMFFD) [HHK92] is proposed. In contrast, DMFFD deforms a 3D mesh by moving its mesh vertices directly. This is done by representing displacements of control points by the displacements of mesh vertices using pseudo-inverse matrices. Such representations allow natural manipulation of mesh itself during modelling.

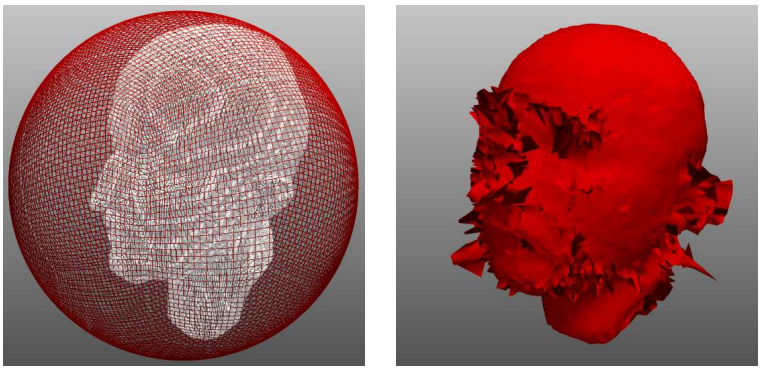
2.1.2 Differential Geometry Methods

Apart from FFD-based methods, differential geometry mesh editing methods such as Laplacian-based [SLCO+04] and Poisson-based methods [YZX+04] are also quite popular. Laplacian-based methods deform a target mesh by displacing some of its vertices to the designated locations, and try to keep the geometry properties for the rest of the vertices. By operating on the mesh vertices directly, the deformation is efficient and intuitive. Poisson-based methods deform a mesh by setting some boundary conditions of a target region and manipulating the gradient field inside the region. These methods are often used to combine 2 meshes into a new one, in which the first mesh provides a boundary condition and the second mesh provides a gradient field within the corresponding boundary.

2.2 Self-intersection of 3D Mesh

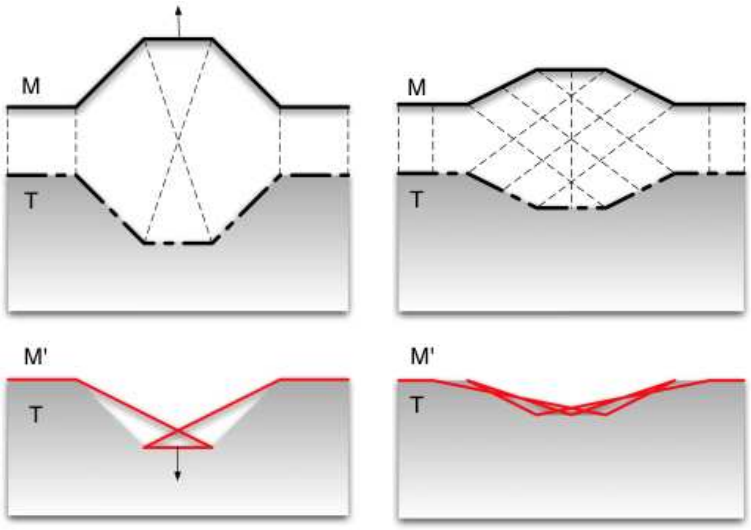
Self-intersection problem may happen if a mesh is not deformed properly. Fig. 2.2 shows a registration of a spherical mesh to a 3D volume (a) resulting in a deformed mesh with self-intersection problem (b). The deformation is done by displacing vertices naively, i.e., moving vertices directly to their designated locations.

A closer look at this problem reveals that it is caused by displacing two neighboring vertices along "opposite" directions. This can be demonstrated using a 2D case as shown in Fig. 2.2(a). The vertices on mesh model M have their estimated corresponding point on the surface of target object T. Displacing vertices directly based on these correspondences results in surface normal flipping of the deformed mesh M' (red). A more complicated situation which involves more vertices is illustrated in Fig. 2.2(b).



(a) (b)

Figure 2.1: Self-intersection problem. (a) Initialization for a binary volume image. (b) Surface normal flippings occur due to the self-intersection problem.



(a) (b)

Figure 2.2: Flip of surface normals (arrows). It may occur when deforming a mesh surface (M) towards the surface (dash dotted line) of a target volume object (T) according to the estimated vertex displacement directions (dashed lines). (a) A flip caused by two neighboring vertices. (b) Multiple flips caused by multiple neighboring vertices.

2.3 Handling of Self-intersection Problem

Below we discuss the segmentation methods based on mesh deformation and focus on their strategies for handling self-intersections. There are two general approaches to handle self-intersection of 3D mesh: (1) detection and resolution of self-intersection and (2) avoidance of self-intersection.

2.3.1 Detection and Resolution of Self-intersection

Some existing segmentation frameworks detect the self-intersections actively but solve them using simple and straightforward methods. The T-snake model [MT99] discretizes the space underlying the mesh into grids and detects self-intersections after deformation by tracking the status of the underlying grid points. It resolves self-intersections by cancelling such an deformation and exerting repulsion forces for the mesh vertices. The method in [LM98] imposes proximity conditions between mesh vertices. By displacing mesh with a very small step size, violation of proximity conditions is detected, and the model is remeshed to remove self-intersections. The methods in [DM01, JSC04, ZBH07] detect self-intersections based on collision detection and resolve them by remeshing. In general, collision detection and remeshing are computationally expensive, and they contribute to most of the computational costs of the algorithms.

2.3.2 Avoidance of Self-intersections

Under Free-Form Deformation (FFD) [SP86], self-intersections can be avoided by imposing injectivity condition [HF98, CL00] on the deformation function. The injectivity condition confines the displacements of FFD control points within regions that do not incur self-intersections. For segmentation, directly displacing mesh vertices, as in Directly Manipulated FFD [HHK92], is preferred so that the mesh surfaces can be accurately aligned to the target boundaries. Unfortunately, it is nontrivial to derive the injectivity condition of mesh vertices from that of the control points. Moreover, the injectivity condition limits the displacements of control points to short ranges, resulting in very slow convergence.

Another approach is to compute a diffeomorphic deformation function [KAB+05,

ZRA+08]. As a diffeomorphic function and its inverse are one-to-one and smooth, the topology of the mesh model will be kept. As a result, self-intersections are avoided. However, computation of the diffeomorphic function is expensive, especially when it is applied to the segmentation of complex and noisy 3D volume images.

Observing that the flipping problem can be circumvented for 2D deforming contour (polygon) by imposing certain constraints, we propose a special quadrilateral mesh where contours can be easily defined. Based on the regularity of this mesh, the surface flipping problem can be solved.

Chapter 3

Related Work

In this chapter, a detailed review of existing medical image segmentation methods is given. There exist many possible criteria for categorizing these segmentation methods. This thesis focuses on (a) the interaction mode used and (b) the types of models those algorithms relying on.

The review begins with a discussion of the user interaction mode of existing segmentation methods. i.e., manual (Section 3.1.1), interactive (Section 3.1.2), semi-automatic (Section 3.1.3) and automatic (Section 3.1.4). Then based on the types of models used, existing segmentation algorithms are categorized into (1) local feature-based (Section 3.2.1), (2) deformable model-based (Section 3.2.2) and (3) atlas-based algorithms (Section 3.2.3). Both advantages and disadvantages of these methods are discussed.

3.1 User Interaction Mode

In general, based on how many human labors are involved, existing segmentation methods can be categorized into manual, interactive, semi-automatic and automatic methods. Manual segmentation methods require full human labors, whereas automatic segmentation methods require none.

3.1.1 Manual Segmentation Methods

Manual segmentation methods require doctors to either draw the contours or paint the regions of the corresponding tissues on computer screens, completely by hand. Manual segmentation was used to quantify soft and rigid tissues in dual energy x-ray images [BAA09]. Manual segmentation was also used in [JAA+95, CMA+98, CMB+98] for radiotherapy planning of prostate cancer. Results of manual segmentation are usually considered the most accurate, and are often used as the ground truth data to evaluate other segmentation methods. However, manual segmentation methods are very time consuming. Some researchers reported that manual segmentation of series of 1500–2000 images of 512×512 pixels usually takes two to four hours [SCK+03]. Besides, different users often give different segmentations of the same image (inter-observer variability), and a single user may give different segmentations of the same image at different times (intra-observer variability). An inter-observer variability of 14-22% measured in disagreement ratio was reported [KWJK98].

3.1.2 Interactive Segmentation Methods

Interactive segmentation methods require doctors to give user input interactively during segmentation process. If doctors are not satisfied with current segmentation results, they can give new initializations or parameter values based on previous segmentation results. These methods can re-compute the results accordingly until an accurate enough result is produced. These methods are usually very time efficient, and are able to provide fast feedback to users. They are widely adopted clinically. Accuracy of interactive segmentation methods depends heavily on how much interaction is involved. In general, more accurate segmentation results can be obtained as more interaction is given. [LMT99] introduce hard constraints on the snake algorithm. interactively place seed points along the boundary of the object of interest. [LMT99] manual markup representing foreground and background of liver tumor, then perform segmentation using graph-cut. If the results are not satisfying, the user can adjust the manual markup and re-compute segmentation.

Commercial products like PathFinder and IntraSense also segment objects of interest interactively. They usually incorporate manual touch-up stages, so that users have chance to modify the results when segmentation is not carried out ideally. The touch-up stage

usually includes adding more seeds to increase the target region, or mark up some regions that are not part of the target.

3.1.3 Semi-automatic Segmentation Methods

Semi-automatic segmentation methods [SS04, AB94, YYJH+92, PT01, vGBvR08, KWT88, XP98, Set99a] require doctors to provide certain degree of initialization, and properly set parameter values. Semi-automatic methods are similar to interactive methods since human labors are involved. Compared to interactive methods, semi-automatic methods do not require users to provide further input and re-compute the results. Both interactive methods and semi-automatic methods are trade offs between fully manual and fully automatic methods.

3.1.4 Automatic Segmentation Methods

Automatic segmentation methods [GW01, GT95, HAHR08, BL79, BMGNJM+97, FLC91, XXE+08, TB92, HKR+08, MTA+08, TT07] require no user input. a computer program does segmentation fully automatically, without any user input. Therefore, segmentation results of automatic methods are not affected by the users, and the results are repeatable. Automatic methods can also save human labors. However, automatic segmentation methods still have several difficulties which hinders their clinical usage. First, large intensity variance of same target tissue across different patients may happen due to (1) different image acquisition machine, (2) diverse tissue properties across patients and (3) different stages of diseases. Second, large shape variance of the same target tissue across different patients. The shape variance roots from either normal shape variance of different patients or deformed shapes due to diseases and operations. Thirdly, amount of image noise is varied. These difficulties usually cause the automatic segmentation methods not as robust.

3.1.5 Summary

All these types of interactions have their pros and cons. Manual segmentation methods are accurate, but require user input, which is time consuming. In comparison, automatic

segmentation approaches are not affected by individual users. However, they are generally not robust enough, and thus cannot be adopted for clinical use at the present stage.

As a trade-off between manual segmentation and fully-automatic segmentation, interactive and semi-automatic methods require minimum user input, thus reducing the inter- and intra-observer variabilities.

As a matter of fact, for some of the algorithms, these types of interactions may be interchangeable depend on the their implementations. For example, if a robust initialization methods can be given, semi-automatic/interactive methods may be converted into fully automatic methods. If users are not satisfied with the results of fully automatic methods, they can manually touch up the results, which in fact converts the automatic methods into interactive/semi-automatic.

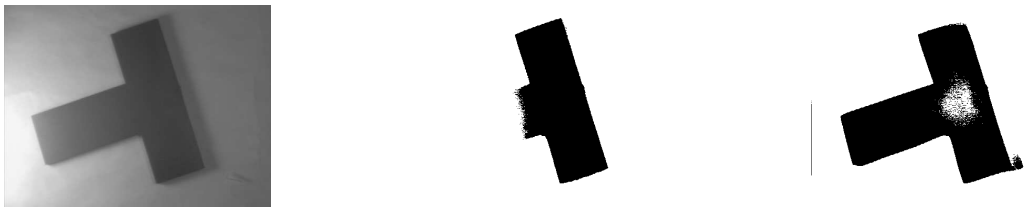
This thesis presents a semi-automatic segmentation algorithm which simply requires its user to place an initial sphere model inside the target object. It can handle large intensity and shape variance. It is possible to be converted into fully automatic if such initialization can be carried out robustly and automatically.

3.2 Model Type

Based on the types of model used, existing medical image segmentation methods can be categorized into model-less, local feature-based (Section 3.2.1), deformable model-based (Section 3.2.2) and atlas-based (Section 3.2.3).

3.2.1 Local Feature-based (No Model)

Model-less methods, as the name implies, do not rely on any model. These methods generally make use of local features. They can be further classified into the following sub-categories [PXP98, Rog00]: *thresholding*, *edge-based*, *region-based*, *graph-based* and *classification-based*.



(a) (b) (c)

Figure 3.1: Adaptive thresholding. (a) Input image with strong illumination gradient. (b) Result of global thresholding at $t = 80$. (c) Adaptive thresholding using 140×140 window (from `http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm`).

Thresholding

Thresholding [SS04] is one of the most basic segmentation techniques. Given an image I, thresholding method tries to find a threshold t such that pixels with intensity values greater than or equal to t are categorized into one group, and the rest of the pixels into the other group. Thresholding requires that the intensity of the image has a bimodal distribution, and performs well on simple images with such a distribution. However, most of the medical images do not have bimodal intensity distribution. In this case, thresholding cannot correctly partition the images into various anatomical structures.

Uneven illumination is another factor that affects the performance of thresholding. Adaptive thresholding [GW01] handles this problem by subdividing an image into multiple sub-images and applying different thresholds on the sub-images (Figure 3.1). The problem with adaptive thresholding is how to subdivide the image and how to estimate the threshold for each sub-image.

In general, thresholding algorithms do not consider the spatial relationship between pixels. Moreover, the segmentation result is quite sensitive to noise. Thresholding alone is seldom used for medical image segmentation. Instead, it usually functions as an image pre-processing step as in [GT95].

Edge-based

Edge-based segmentation algorithms use edge detectors to find object boundaries in the image. Traditional *Sobel* edge detector [GW01] uses a pair of 3×3 convolution kernels

to compute the first order derivatives (gradients) along the x- and y-directions of the 2-D image. Instead of computing first order derivatives, the *Laplacian* computes the second order derivatives of the image. Usually, the Laplacian is not applied directly on the image since it is sensitive to noise. It is often combined with a Gaussian smoothing kernel, which is then referred to as the Laplacian of Gaussian (LoG) function. Bomans *et al.* [BHTR90] used a 3-D extension of the LoG to segment brain structures in 3-D MR images. Goshtasby and Turner [GT95] used this operator to extract the ventricular endocardial boundary in cardiac MR images. Similarly, 3D Log-Gabor filter bank was used to extract ridge features which correspond to tissue/bone interface in 3D ultrasound image [HAHR08].

More advanced edge detectors have been proposed in the computer vision literature. *Canny* edge detector [Can86] uses a double-thresholding technique. A higher threshold t_1 is used to detect edges with strict criterion, and a lower threshold t_2 is used to generate a map that helps to link the edges detected in the former step. Harris proposed a combined corner and edge detector known as the *Harris* detector [HS88], which finds the edges based on the eigenvalues of the Hessian matrix.

Edge-based image segmentation algorithms are sensitive to noise and tend to find edges that are irrelevant to the real boundary of the object. Moreover, the edges extracted by edge-based algorithms are disjoint and cannot completely represent the boundary of an object. Additional processing is needed to connect them to form closed and connected object regions.

Region-based

Typical region-based segmentation algorithms include *region growing* and *watershed*.

A. Region Growing. The region growing algorithm begins with selecting n seed pixels. The seed pixel can be selected either manually or by certain automatic procedures, e.g., the *converging square* algorithm [OS83] as applied in [AB94]. The converging square algorithm recursively decomposes an $n \times n$ square image into four $(n-1) \times (n-1)$ square images and continues with the one with maximum intensity density. This procedure is repeated until a single point remains. After the seed pixels are selected, each seed pixel

i is regarded as a region A_i, $i \in \{1, 2, \ldots, n\}$. The region growing algorithm then adds neighboring pixels to the regions with similar image features, thereby growing the regions.

The choice of homogeneity criterion is crucial for the success of this algorithm. A homogeneity criterion proposed by Adams and Bischof [AB94] is the difference between the pixel intensity and the mean intensity of the region. Yu *et al.* [YYJH+92] proposed to use the weighted sum of gradient and the contrast between the region and the pixel as the homogeneity criterion. Pohle and Toennies [PT01] proposed an adaptive region growing algorithm that incorporates a homogeneity learning process instead of using a fixed criterion. Ginneken *et al.* [vGBvR08] labelled airway trees from thoracic CT images using region growing.

Region growing algorithms are fast, but may produce undesired segments if the images contain much noise. Furthermore, region-based algorithms will segment objects with inhomogeneous region into multiple sub-regions, resulting in over-segmentation.

B. Watershed. The watershed algorithm is another region-based image segmentation approach originally proposed by Beucher and Lantuéjoul [BL79]. It is a popular segmentation method coming from the field of mathematical morphology. According to Serra [Ser82], the watershed algorithm can be intuitively thought of as a landscape or topographic relief that is flooded by water. The height of the landscape at each point represents the pixel's intensity. Watersheds are the dividing lines of the catchment basins of rain falling over the regions. The input of the watershed transform is the gradient of the image, so that the catchment basin boundaries are located at high gradient points [RM01].

The watershed transform has good properties that make it useful for many image segmentation applications. It is simple and intuitive. It can also be parallelized [RM01], and always produces a complete division of the image. However, it has several major drawbacks. It can result in over-segmentation (Figure 3.2) because each local minimum, regardless of the size of the region, will form its own catchment basin. It is also sensitivity to noise. Moreover, watershed algorithm does not perform well at detecting thin structures and structures with low signal-to-noise ratio [GMA+04].

To improve the performance of the watershed algorithm, Najman and Schmitt [NS96]

(a) (b)

Figure 3.2: Result of the watershed algorithm. Over-segmentation is clearly visible. (a) The input image. (b) The segmentation result (from `http://www.itk.org/HTML/WatershedSegmentationExample.html`).

proposed to use morphological operations to reduce over-segmentation. Grau *et al.* [GMA+04] encoded prior information into the algorithm. Part of its cost function is changed from the gradient between two pixels to the difference of posterior probabilities of having an edge between two pixels given their intensities as the prior information. Wegner *et al.* [WHOF96] proposed to perform a second watershed transform on the mosaic image generated by the first watershed transform to reduce over-segmentation.

Graph-based

Graph-based approach is relatively new in the area of image segmentation. The common theme underlying this approach is the formation of a weighted graph, where each vertex corresponds to a pixel or a region and each edge is weighted with respect to the similarity between neighboring pixels or regions. A graph $G = (V, E)$ can be partitioned into two disjoint sets A and B, where $A \cup B = V$ and $A \cap B = \emptyset$, by removing edges between them. Graph-based algorithms try to minimize certain cost functions, such as a *cut*,

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{3.2.1}$$

where $w(u, v)$ is the edge weight between u and v.

Wu and Leahy proposed the *minimum cut* in [WL93]. A graph is partitioned into k sub-graphs such that the maximum cut across the subgroups is minimized. However, based on this cutting criterion, their algorithm tends to cut the graph into small sets of

nodes because the value of Eq. (3.2.1) is, to some extent, proportional to the size of the sub-graphs. To avoid this bias, Shi and Malik [SM00] proposed the *normalized cut* with a new cost function *Ncut*,

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{3.2.2}$$

where $assoc(X,V) = \sum_{u \in X, t \in V} w(u,t)$ is the total connection from nodes in X to all nodes in the graph. In [WS03], Wang and Siskind further improved the graph cut algorithm, and proposed a new cost function for general image segmentation, namely *Ratio Cut*. This scheme finds the minimal ratio of the corresponding sums of two different weights associated with edges along the cut boundary in an undirected graph:

$$Rcut(A,B) = \frac{c_1(A,B)}{c_2(A,B)} \tag{3.2.3}$$

where $c_1(A,B)$ is the first boundary cost that measures the homogeneity of A and B, and $c_2(A,B)$ is the second boundary cost that measures the number of links between A and B. A polynomial-time algorithm is also proposed.

Boykov and Jolly [BJ01] used graph cuts for interactive organ segmentation, e.g., bone removal from abdominal CT images. Their segmentation is initialized with some manual "clicks" and "strokes" on object regions and backgrounds (Figure 3.3). These clicks and strokes are regarded as seed points, which provide hard constraints for the segmentation, and intensity distributions for the object and the background. This information is later integrated into the proposed graph cut cost function, which is minimized during segmentation.

Zheng *et al.* [ZBE+07] proposed to refine the manual coarse segmentation of breast tumor in MR images using the graph-cut algorithm.

Wels *et al.* [WCA+08] applied probabilistic boosting trees [Tu05] to compute the probability of a voxel as foreground or background. The computed probability was then used in the graph-cut algorithm to segment brain tumors.

Compared to region-based segmentation algorithms, graph-based segmentation algorithms tend to find the global optimal solutions, while region-based algorithms are based on greedy search. Since graph-based algorithms try to find the global optimum, they are computationally expensive.

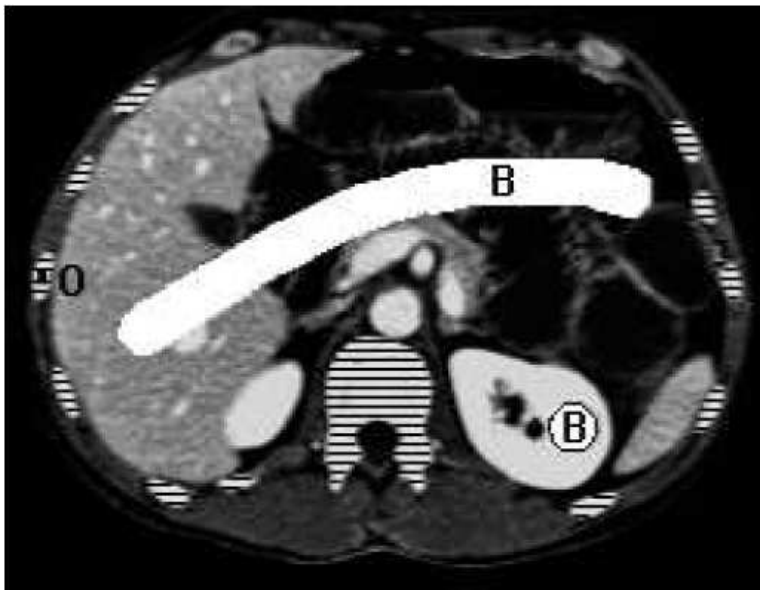


Figure 3.3: Bone removal in a CT image using interactive graph cut [BJ01]. The regions marked by "O" and "B" are manually initialized as object and background respectively. Bone segments are marked by horizontal lines.

The performance of the graph cut method is quite good for images whose foreground and background intensities are well separable, but often unsatisfactory when the foreground and the background share similar color distributions. Another limit for the graph cut based method lies in its underlying assumption that an object's shape is best described by the shape with smallest boundary length, which does not hold for sophisticated shapes in medical images.

Classification-based

Ren and Malik proposed to train a classifier to separate "good segmentation" from "bad segmentation" [RM03]. The criteria used for classification include texture similarity, brightness similarity, contour energy and curvilinear continuity, etc. A pre-processing step which groups pixels into "super-pixels" is used to reduce the size of the problem, which adopts the normalized cut [SM00]. For classification, human segmented natural images are used as positive examples, while negative examples are constructed by randomly matching human segmentations and the images. Based on the trained classifier, the

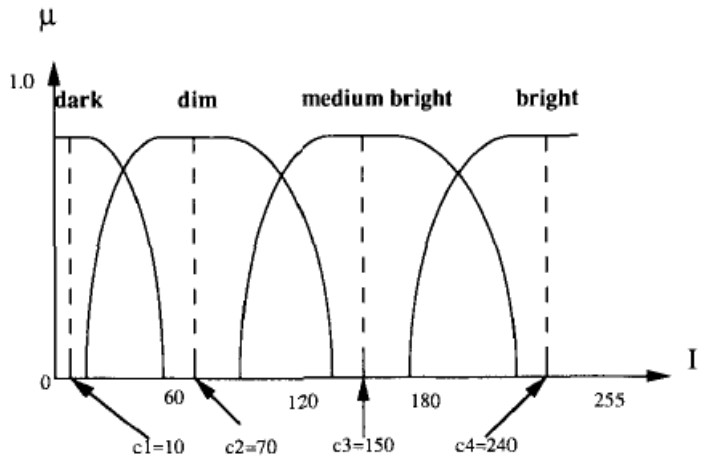


Figure 3.4: Fuzzy membership functions for linguistic descriptions *dark*, *dim*, *medium-bright* and *bright* [FLC91], c1–c4 are the intensity values at which the respective membership function reaches its maximum.

algorithm groups "super-pixels" into segments.

Fuzzy reasoning methods are proposed to detect the cardiac boundary automatically [BMGNJM+97, FLC91]. These methods begin with the application of the Laplacian-of-Gaussian to obtain the zero-crossings of the image. High-level knowledge is usually represented in linguistic form. For example, intensities are described as "dark", "dim", "medium bright" or "bright". Fuzzy sets are developed based on the fuzzy membership functions of these linguistic categories (Figure 3.4). The fuzzy membership function is set empirically to describe the range of possible intensity values. A rough boundary region is then obtained from fuzzy reasoning, where a search operation is employed to obtain the final boundary. Fuzzy C-means was reported to be used for parenchyma segmentation in breast MR images [XXE+08].

Toulson and Boyce proposed to use a back-propagation neural network in image segmentation [TB92]. The neural network is trained on the set of manually segmented samples. Segmentation is performed on a pixel basis. The inputs to the neural network are the class membership probabilities of the pixels from a neighborhood around the pixel being classified. Therefore, contextual rules can be learned and spatial consistency of the segmentation can be improved.

Mixture of Gaussians of voxel intensities was estimated by Habas *et al.* [HKR+08],

where each Gaussian represented an anatomy class in brain. Segmentation was obtained by maximizing the posterior probability of a tissue class given the voxel intensity.

A single strong classifier is usually hard to be learned. To counter this problem, boosting algorithms combine multiple weak classifiers into a strong one. AdaBoost was used to segment sub-cortical structures in brain images [MTA+08]. However, AdaBoost needs to pick a large number of weak classifiers, and is therefore computational expensive. In addition, the order of the features selected which may correspond to high level semantics is not respected. Re-weighting procedure may alter the classification results that are correct in the earlier stages. Probabilistic Boosting Trees (PBT) [Tu05] was proposed to tackle these problems. It contains a standard AdaBoost classifier in each of its node. PBT was used in [TT07] to get an initial rough segmentation of brain structures. The inputs of the classifiers are sub-volumes of the images.

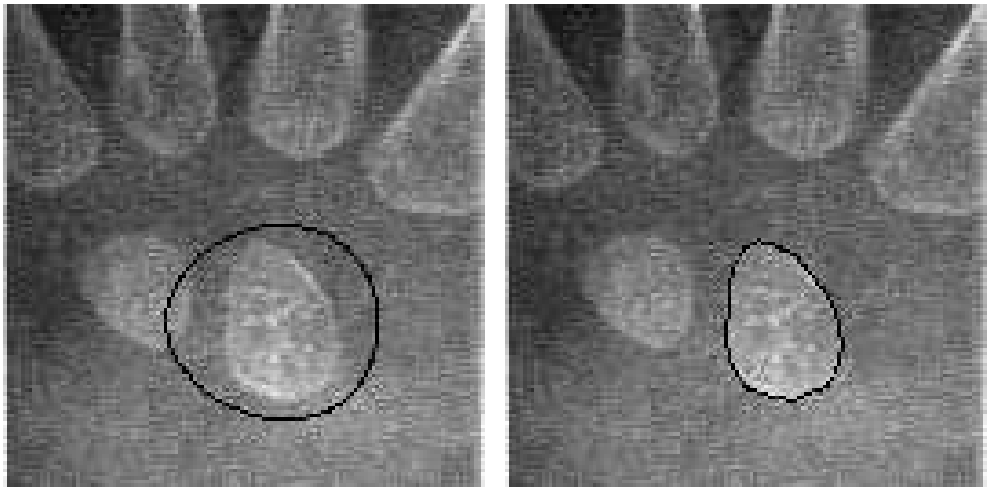
Classification-based segmentation algorithm requires training. The training parameters are usually set in a trial-and-error manner, which is subjective. The accuracy of this algorithm largely depends on the selected training samples. In addition, classification-based segmentation algorithm is more tedious to use.

3.2.2 Deformable Model-based

Deformable model-based segmentation methods are quite popular recently, because such methods are able to change the shape of the model relatively easily, and thus can handle shape variability of the target organ. They can also incorporate domain information (sometimes training shapes) to help handle shape variability if the variability is not very significant. Many deformable model-based algorithms have been proposed, the most important of which are discussed below.

Active Contour Models (Snake)

The *snake* model was first proposed by Kass *et al.* [KWT88]. It is a controlled continuity spline which can deform to match any shape under the influence of two kinds of forces. The internal spline force imposes a piecewise smoothness constraint, while the external force attracts the snake to the salient image features such as lines, edges and terminations.



(a) (b)

Figure 3.5: Snake segmentation of bone [AM00]. (a) The initial contour (black curve). (b) Segmentation result (black curve) (from `http://www.cvc.uab.es/~petia/dmcourse.htm`).

The snake algorithm iteratively deforms the model and finds the configuration with the minimum total energy, which hopefully corresponds to the best fit of the snake to the object contour in the image (Figure 3.5).

Atkins and Mackiewich [AM00] used the active contour for brain segmentation. The input image is smoothed, and then an initial mask that determines the brain boundary is obtained by thresholding. Finally, segmentation is performed by the snake model.

Snake is a good model in edge detection, shape modeling, segmentation and motion tracking, because it forms a smooth contour that corresponds to the region boundary. However, it has two intrinsic problems. First, its result is often sensitive to the initial configuration of the snake. Second, it cannot converge well to concave parts of the regions.

An analysis of the snake model shows that its image force, usually composed of image intensity gradient, exists in a narrow region near the convex part of the object boundary. A snake that falls in a region without image forces cannot be pulled towards the object boundary. Snake with Gradient Vector Flow (GVF), proposed by Xu and Prince [XP98], partially solved this problem by pre-computing the diffusion of the gradient vectors (gradient vector flow) on the edge map (Figure 3.6). As a result, image forces exist even near

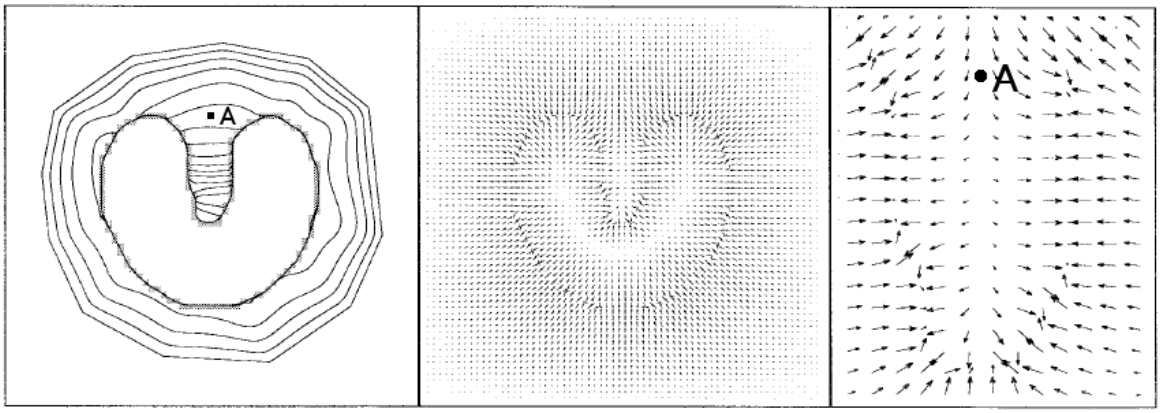


Figure 3.6: Gradient vector flow [XP98]. Left: deformation of snake with GVF forces. Middle: GVF external forces. Right: close-up within the boundary concavity.

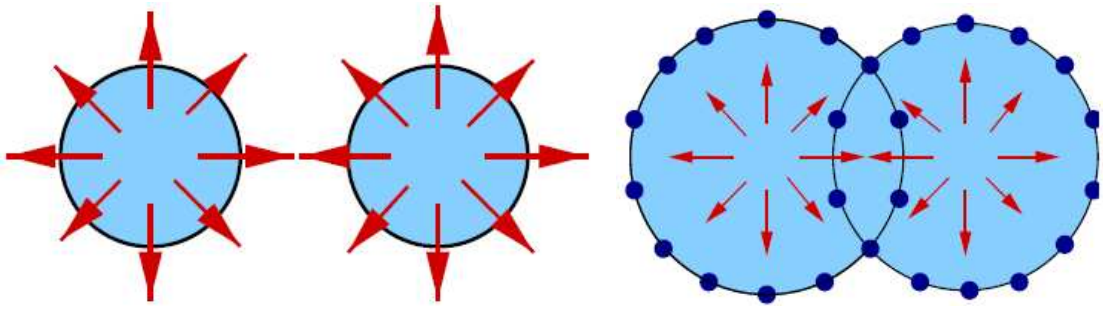
concave regions, which can pull the snake towards the desired object boundary. GVF is less sensitive to the initial configuration of the contour than the original snake model. However, it still requires a good initialization and can still be attracted to undesired locations by noise.

Level Set

Snake-based deformable model cannot handle evolving object contours that require topological changes. For example, when two evolving contours merge into one (Figure 3.7), algorithms that represent the contours by connected points need to remove the points inside the merged region. This is computationally expensive, especially in 3-D.

Sethian proposed a *level set* [Set99a] algorithm to solve this problem by embedding the contour in a higher dimensional surface called the level set function. The contour is exactly the intersection between the level set function and the x-y plane, and corresponds to the boundary of the object to be segmented. For 2-D contour, the level set function $z = \phi(x, y, t = 0)$ is represented as a 3-D surface, and is initialized by the signed distance from point (x, y) to the contour in the x-y plane. The desired object contour is the zero level set of the level set function, i.e., $\phi(x, y, t) = 0$.

The evolution of the contour is propelled by force F, which may depend on many factors such as local geometric information and properties of the contour. Once the

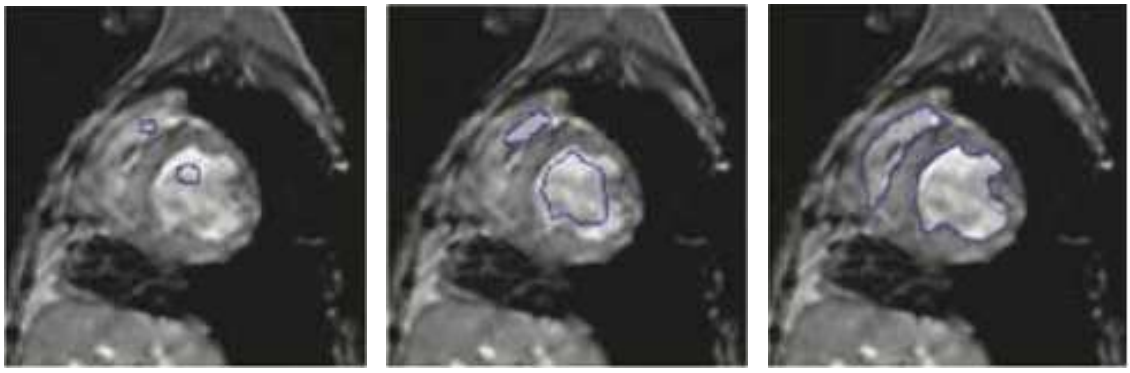


(a) (b)

Figure 3.7: Merging of contours [Set97]. (a) Two initially separate contours. (b) Two contours are merged together.

level set function is constructed, the evolution of the interface (contour) can be easily computed. The level set function actually represents all possible states of the evolution of the contour, which cannot be constructed in advance. To solve this chicken-and-egg problem, instead of constructing the whole level set function directly, the evolving zero level set is computed iteratively based on the force F. The iterative algorithm needs to update only the level set function values near the current object boundary. This leads to the narrow band method [Set99a]. If the contour propagates only in one direction, the fast marching algorithm [Set99a] can be used.

The level set method is widely adopted in the literature of medical image segmentation, and a number of improvements have been proposed. To restrict the evolution of the zero level set, Yang *et al.* proposed a Level Set Distribution Model (LSDM) [YSD04], which is similar to the Point Distribution Model (PDM) described in the next section. The segmentation process is shown in Figure 3.8. Pluempitiwiriyawej *et al.* proposed to segment 2D cardiac MR images using the level set method [PMWH05]. Their method incorporates stochastic region information (which is similar to that in [CV01]) and edge information (largest magnitude of the gradient) with an ellipse shape prior. The length of the contour is also taken into account to keep contour smoothness. The contour of the heart is represented implicitly such that the energy function can be minimized using the level set framework. Li *et al.* [LHD+08] applied the level set framework to perform segmentation and intensity bias correction for MR images. Level set method was also adopted to segment vertebrae from thoracic CT images [SLA08]. The average of several



(a) (b) (c)

Figure 3.8: Level set segmentation of heart image [YSD04]. (a) Initial contour (blue curve). (b) Evolving contour. (c) Final contour.

registered manually segmented vertebra is used as the initial front. Blockers were derived based on detected ridges and valleys to prevent leakage of the front. To deal with the noise problem, Droske *et al.* proposed to incorporate curvature terms in the velocity function [DMRS01]. They also proposed an adaptive grid to speed up the fast marching algorithm.

Image segmentation problem can be formulated as an energy minimization problem as the Mumford-Shah model [MS89]. The energy function in the Mumford-Shah model contains 3 terms: (1) the difference between the approximation function and the original image, (2) sum of squared gradient magnitude of the approximation function inside regions, and (3) length of the region boundary. Term (2) and (3) function as regularization terms. Term (2) assumes the piecewise smooth image model. When term (2) is omitted, the piecewise constant image model is adopted. In general, it is difficult to minimize such a functional since the problem is not convex and the dimensionality of the possible region boundary is high.

Chan and Vese [CV01] proposed to solve the Mumford-Shah model in a reduced form, i.e., the piecewise constant approximation function was used. The function takes two values, either the average intensity inside or the average intensity outside the region. The region boundary was represented implicitly as a level set function. Euler Lagrange equation can be derived from the energy functional, and the level set function can be solved by finite difference method.

The major advantage of the level set approach is that the level set function remains a single function while the zero level set may change topology, break, merge and form sharp corners [MSV95]. However, it generally cannot maintain shape information, and is sensitive to noise.

Fast Marching Method

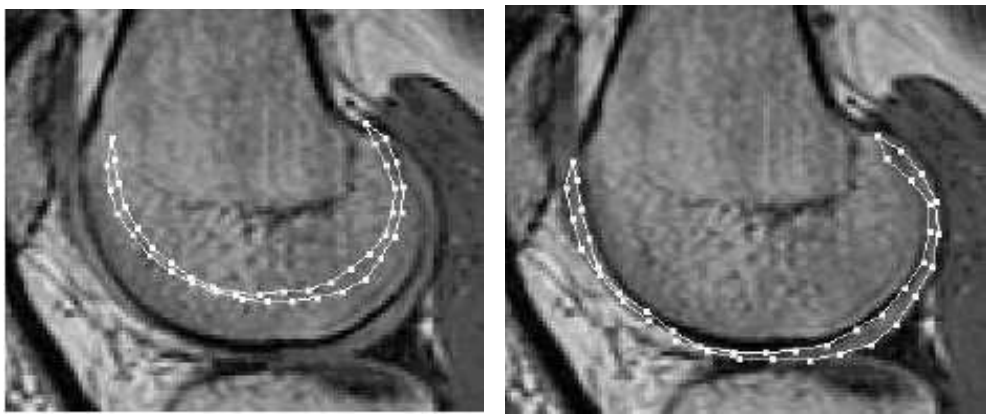
Fast marching method [Set99b] formulates deformation problem as boundary propagation. Given an initial boundary, the fast marching method compute the arrival time of the boundary on each point in the image. The arrival time is computed based on a speed function derived from the image characteristics at each image point. The iso-curves (surfaces) at a specific time produces the deformed boundary, i.e., segmentation results.

To use the fast marching method for image segmentation purpose, the user needs to provide foreground and background strokes in order to compute the speed function. It was used for fast natural image segmentation [BS09]. It was also used for soft segmentation of medical volume images [DLCL09].

Fast marching method is time efficient. It can achieve good segmentation results if the foreground and background of the image has large intensity/color difference. Therefore, it can obtain good segmentation results for natural images. However, medical images are in gray-scale, and some boundaries of the target tissues may be of low contrast. Therefore, the fast marching method used on medical images directly for segmentation is not as efficient. The fast marching method cannot preserve the topology of the target, resulting in undesired segmentations.

Active Shape/Appearance Model

Many objects in medical images, such as organs, cells and other biological structures, have a tendency towards some average shape. When a collection of shapes of the same organ is available, standard statistical analysis can be applied. The *active shape model* (ASM) [CHTJ94] and the *active appearance model* (AAM) [CET98], both proposed by Cootes *et al.*, are widely used when a set of training samples are available. ASM is also know as Statistical Shape Model (SSM).



(a) (b)

Figure 3.9: Segmentation of cartilage by active shape model [CT04]. (a) Initial contour (white curve). (b) Resultant contour after 14 iterations.

In ASM, a training shape is usually represented by a $2n$-dimensional vector $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$ containing coordinates of points on the shape. The shape vector corresponds to a point in a high-dimensional ($2n$-D) space called the eigen space. Thus, ASM is also known as the Point Distribution Model (PDM). The training samples form a point cloud in the eigen space. ASM applies Principal Component Analysis (PCA) on this point cloud to identify the eigenvectors (eigenshapes) that describe the point cloud. An arbitrary shape can be represented by linear combination of these eigenshapes with different coefficients and a model can be deformed by changing these coefficients. More specifically, an initial guess can be randomly generated [HTC92]. Then, an optimization algorithm such as the genetic algorithm or direct searching in the eigen space can be used to find the optimal solution. Segmentation of cartilage on MR images is shown in Figure 3.9. In comparison, AAM incorporates not only shape but also gray level information, and improves the robustness of ASM.

Based on Cootes' model, Wang and Staib [WS98] applied the smoothness covariance matrix to make the neighboring points on the shape correlated, i.e., the neighboring points on the shape are more likely to move together. To bias the search process in a certain range, a Bayesian formulation based on prior knowledge is proposed. The prior of shape and pose parameters is modeled as a zero-mean Gaussian distribution. Similar work was done by Gleason *et al.* [GSSA$^+$02] in detecting kidney disease in CT images.

Shi and Shen [SS08] observed that the test samples may be similar to only sub-population of the training samples Therefore, hierarchical SSM was proposed. In the training stage, training samples were clustered by similarity to form shape models representing sub-population. In the testing stage, the input sample was first deformed according to the global SSM, and then deformed according to the most similar sub-population.

Multi-level statistical shape model [OSS+07, OYH+08] has also been proposed. It is able to increase the variability of the model when the number of training samples is small. However, the model needs to be partitioned hierarchically into patches. Overlapping between neighboring patches during deformation has to be taken into account. These introduce complexities to the algorithm.

Fripp *et al.* [FCWO07] used statistical shape model to segment femoral head in order to extract the cartilage. Direct segmentation of the cartilage is hard due to its very thin structure.

Volumetric Active Appearance Model was adopted to segment brain structures [BCT+08]. A global AAM is used to get an initial segmentation. Individual AAMs are then used to refine segmentation of each structure. Similar samples were clustered together to represent sub-population shape models.

The advantage of ASM and AAM is that the shape can be deformed in a more controllable way compared to snake and level set methods. One of the disadvantages is that they require a lot of training samples to build a point distribution in the high-dimensional eigen space. An eigen space with a small number of eigenshapes may not be able to generate the desired shape, while an eigen space with a large number of eigenshapes may incur high complexity in finding the optimal solution.

3.2.3 Atlas-based

Atlas-based algorithms are very promising for automated segmentation of healthy organs. However, it is very difficult to encode information of pathological organs, which may exhibit large variance in shapes and intensities.

An atlas is a model that contains domain information of anatomical parts. In practice, an atlas is usually obtained by manually segmenting and labeling one or a set of n-

dimensional images containing the same anatomical parts. The basic idea of atlas-based segmentation is to register the atlas to the input images and label the input images according to the atlas.

Broadly speaking, existing methods use either a non-probabilistic atlas [ANWD99, BMC+05, CP00, CKRP98, CP04, CCMT01, CPB+04, DHT+99, DLW05, DHT05, GDMW04, HPMD99, LVSOMR02, LAGBA05, RSOLV+02, SL97, SD00, SHD01, SLZ+04, VJYL03] or a probabilistic atlas [GMA+04, CHT+03, LVRMSO03, LVSOE+04, PBM03, PGLG05, SPvG05, SCK+03]. In the probabilistic case, the probability of a voxel belonging to certain tissue type [GMA+04, PBM03] or probabilistic shape in active shape model are used. The representations of probability are often Gaussian [GMA+04, PBM03].

Atlas-based segmentation methods consist of two major stages, namely *global alignment* and *local refinement*. These two stages are discussed in more detail in the following sections.

Global Alignment

The purpose of global alignment is to align the position, scale and rotation of the atlas to the input image. "Global" means that each part of the atlas undergoes exactly the same transformation, which includes scaling, rotation and translation.

Several transformation types, such as similarity transformation and affine transformation, are frequently applied. Affine transformation [BMC+05, CCMT01, CPB+04, DLW05, DHT05, LVSOMR02, LAGBA05, VJYL03, LVRMSO03, LVSOE+04] has the highest Degrees of Freedom (DoF). It captures rotation, scaling, translation and shearing. Similarity transformation [CKRP98, DHT+99, HPMD99, HPS+99] includes rotation, scaling and translation. These transformations are all linear transformations and have very low computational complexity. Non-linear transformations (usually low-order polynomial) may also be used [DHT05], since they can capture more variation of the atlas, thus making the global alignment more accurate. At the same time, their computational complexity is relatively low. Rigid transformation is seldom used since it only captures rotation and translation.

The transformation can be performed either manually or automatically. Semi-automatic methods are also proposed. Gansor *et al.* applied a manual approach which manipulates

a grid to match the atlas and input images [GDMW04]. Semi-automatic schemes normally include a common procedure, which is manually selecting landmark points, and thus, establishing correspondence between the atlas and the input images. The rest of this section reviews the automatic methods.

Aboutanos *et al.* [ANWD99] applied morphological operations to segment the surface of brains. Their algorithm erodes an initial model with a 2-D circular disk, which has 11 pixels in diameter. The algorithm is claimed to guarantee the placement of the model inside the cortical area. However, this algorithm assumes that the organ in the atlas and in the target image are very similar in position. It is very specific and cannot be easily generalized. Especially when the initial model and the organ in the target image are quite different, this algorithm will fail inevitably.

For the rest of the automatic methods, we further classify them into two groups. One group actively searches for correspondence, and the other group does not search for correspondence explicitly.

Iterative Closest Point (ICP) and optical flow-based algorithms belong to the first group. When performing registration, for each (sampled) point on the atlas, ICP iteratively searches for the nearest point on the target as a possible correspondence, solves a transformation matrix, and updates, i.e., transforms the atlas, until the sum-squared difference between the atlas and the target is minimized. ICP can be regarded as a geometric method. Optical flow-based algorithm, on the other hand, is a photometric method. It borrows the idea from tracking, and treats the atlas and target image as neighboring frames in a temporal motion sequence. In each iteration, the algorithm uses optical flow to search for the correspondence between the atlas image and the target image, and computes the displacement of each point in the image. A Gaussian filtering step is often applied to smooth the displacements.

Optimization-based algorithms belong to the second group. The main framework of optimization-based algorithms is to formulate a similarity or dissimilarity function between the atlas and the target, and apply optimization algorithm to maximize or minimize that function. The similarity or dissimilarity functions proposed can be sum of squared differences between the intensities of corresponding voxels [CKRP98, VJYL03, HPS+99], chamfer distance [CCMT01, CPB+04], correlation ratio [BMC+05, LAGBA05], or mutual

information [DHT+99, DHT05, HPMD99, LVSOMR02, LVRMSO03, LVSOE+04]. The optimization algorithms applied include gradient descent, Levenberg-Marquardt, simulated annealing, etc.

Local Refinement

The purpose of the local refinement is to align the atlas and the target as accurately as possible. "Local" means that different parts of the atlas may undergo different transformations. Since accuracy is the main objective, the methods used in this stage have to focus on the details of the atlas and the target image. Therefore, they are highly complex. Local deformation and pixel classification are two major approaches used for the local refinement stage.

A. Local Deformation Local deformation deforms the atlas locally and fits it to the target image accurately. A number of methods have been proposed to solve the local deformation problem. Some methods actively search for or guess the corresponding points in the target image. Thirion [Thi98] named such methods as attraction models, since the model is attracted by image features and deformed to match the target image.

Ding *et al.* proposed an Iterative Corresponding Point algorithm [DLW05], which belongs to the attraction model. The proposed method is similar to the original ICP. Apart from that, it uses intensity difference distribution (IDD) along the contour to find possible correspondence. It computes IDD of each point along the contours in the atlas, and searches for corresponding point in a small neighborhood in the target image that has the most similar IDD. Once the correspondence is established, an affine transformation matrix is computed to transform the atlas contours. The process discussed above is repeated until convergence. Finally, the GVF snake is applied to extract accurate object boundaries.

Diffusion-based algorithms such as demons algorithm [Thi98] is very popular, and it is used in [GMA+04, BMC+05, CKRP98, CCMT01, CPB+04, DHT+99, HPMD99, HPS+99]. In the demons algorithm, the deformable model or image diffuses through the fixed target image by the action of the effectors called demons located on the object boundaries in the target image. Demons act locally to pull the deformable models towards

the target image.

For the attraction method, the points located on the model contour are attracted by the closest points on the target contour. On the other hand, for the diffusion method, the model is pulled into the target by the action of the demons located on the target contour. Demons algorithm has several variations based on the selection of the demons' positions, the types of deformations, and the forces of the demons. In the most popular variation, all pixels in the target image are selected as the demons. For each demon, a displacement is computed by the optical flow algorithm. A Gaussian filter is then applied to obtain a smooth displacement field. The above process is iterated until the final displacement field is obtained, which represents the deformation of the model.

PASHA [CBD+03] proposed by Cachier *et al.* is used in [LAGBA05]. PASHA algorithm incorporates both geometric features and intensity features. The energy function of PASHA contains three components: intensity similarity, geometric distance, and a regularization term. The intensity similarity term measures the local correlation between the points in the source image and those in the target image. The geometric distance measures the disagreement between the estimated point correspondence and the computed deformation. The regularization term imposes smoothness constraints. A gradient descent algorithm is used to optimize the energy function and, in the process, estimate the point correspondence, and compute the deformation. Large displacement vectors are not favored. The displacement field produced by PASHA is smoother than that produced by the demons algorithm.

Some local deformation methods are based on the standard optimization algorithm. Aboutanos *et al.* [ANWD99] proposed a cost function that combines intensity, morphology, gradient, deviation from previous contour and smoothness. They used an optimization algorithm to minimize the cost function and transform the initial contour to fit the brain.

Standard deformable models are also used in the local refinement stage such as snake and its modified versions [DLW05, SL97, CHT+03], level set [DHT05, VJYL03]. For example, Ding *et al.* [DLW05] applied snake with GVF algorithm for final contour refinement. Duay *et al.* [DHT05] applied level set algorithm to perform local deformation after global alignment.

B. Pixel Classification Pixel classification method separates the pixels into several groups, and each corresponds to an anatomical part. It is usually performed in probabilistic atlas-based segmentation. Classification is based on maximizing a posterior probability of a pixel belonging to a particular anatomical part. The features used in pixel classification is usually intensity and position information of the pixel. Park *et al.* [PBM03] proposed to classify pixels using a Bayesian framework. Pixels are classified into 5 groups: liver, right kidney, left kidney, spinal cord and "none of the above". The atlas is constructed from manually segmented organs from 32 registered CT slices. The intensity value of each pixel in the atlas image corresponds to the probability that it belongs to certain organ. The cost function to be maximized contains a Maximum A Posteriori (MAP) formulation that estimates the classes of the pixels that best explain the given input image. It also includes a Markov Random Field (MRF) regularization term, which penalizes dissimilar adjacent labels.

Lorenzo *et al.* [LVSOMR02, LVRMSO03, LVSOE$^+$04] proposed a classification algorithm to segment heart images. The intensity distribution of each class that corresponds to an organ is modeled by a Gaussian distribution. The mean and variance of the distribution are computed by Expectation Maximization (EM) algorithm based on the training samples. The classification is also followed by a MRF regularization process similar to [PBM03].

Prastawa *et al.* [PGLG05] used Minimum Covariance Determinants (MCD) [RD99] to generate robust mean and variance of uni-modal intensity distribution, and then classified pixels in a Bayesian framework. The MCD estimator computes the mean and covariance that have the smallest determinant of covariance.

3.3 Summary

In terms of the interaction modes of current medical image segmentation methods, they can be categorized into manual, interactive, semi-automatic and fully automatic. Manual segmentation methods are accurate, but time-consuming. They have relatively large intra- and inter-observer variabilities. Fully automatic segmentation methods do not rely on human labor, and can produce repeatable results. However, they are not robust enough for clinical use at current stage. As a trade off, semi-automatic or interactive

segmentation methods incorporate minimal user input, and are able to produce clinically acceptable results.

In terms of the types of model used by current medical image segmentation methods, they can be categorized into model-less, deformable model-based and atlas-based. Model-less methods such as thresholding, region-based, edge-based, graph-based and classification-based methods make use of local features of the image. They are very time efficient, and can obtain good results dealing with simple natural images, i.e., the foreground and background of the images have high intensity contrast, or the foreground and background regions are homogeneous. They cannot perform well on medical images which in general have none of these properties. Deformable model-based methods, in contrast, formulated the segmentation problem as contour/surface deformation. These methods make use of more global information, e.g., geometric properties of the contour/surface to control the segmentation. Therefore, they tend to be more robust to inhomogeneity of the target regions. These methods have relatively higher computational complexities.

Model-less and deformable model-based methods focus more on segmentation of a single target object. Their segmentation results may leak into irrelevant regions (leak problem) or are not able to represent complex shapes once constrained by statistical shape priors. Some model-less methods such as thresholding may produce results with multiple regions, but these methods cannot give semantically meaningful interpretation of the segmented regions.

In contrast, atlas-based methods segment multiple objects at the same time. These methods are more robust than those segmenting only one object, since the atlas can provide many information to guide the segmentation process, such as relative position, orientation, size, intensity, etc, between these objects. However, many atlas-based still rely statistical shape priors of individual object to constrain their shapes. Therefore, they are not able to segment complex object with large shape variances using limited training samples.

This thesis presents a semi-automatic segmentation algorithm, which includes minimal user input. The segmentation results are less likely affected by the users. It is robust to image noise and time efficient because it uses a mesh-based deformable model. The proposed method can segment objects with complex shapes since it does not reply on

any statistical shape priors. It can segment multiple objects simultaneously without overlapped regions, thus reducing the leakage problem.

Chapter 4

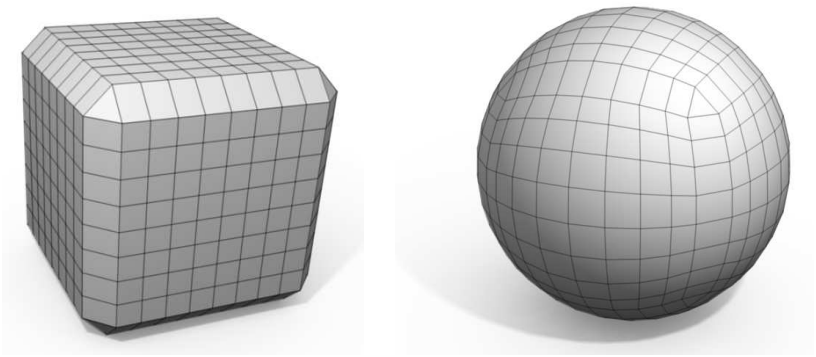
Flipping-free Mesh Deformation

This chapter presents a flipping-free mesh deformation algorithm based on a specially designed quadrilateral mesh model (Section 4.1). This quadrilateral mesh facilitates the detection and avoidance of flippings during mesh deformation (Section 4.2). Extensive experiments show that the algorithm can register the mesh to complex volume data without self-intersection.

4.1 3D Quadrilateral Mesh

The 3D quadrilateral mesh M is initially defined on a cube whose sides are aligned with the x-, y- and z-axis (Fig. 4.1(a)).

- The mesh is defined by 3 groups G_{xy}, G_{yz}, and G_{zx} of closed contours. Each group consists of mutually non-intersecting closed contours that are parallel to the xy-, yz- or zx-plane respectively. The contours in a group are *orthogonal* to those in the other groups.
- Each mesh vertex u_i in M is an intersection of two contours, each from a different group.
- Each mesh vertex u_i in M has exactly 4 connected neighboring vertices.
- Each cell has four edges, except for those at the 8 corners. Each corner cell has only 3 edges.



(a) (b)

Figure 4.1: 3D quadrilateral mesh. (a) The cubical quadrilateral mesh is projected onto a spherical surface to generate (b) a spherical quadrilateral mesh.

The cubical quadrilateral mesh can be mapped to a spherical quadrilateral mesh (Fig. 4.1(b)) by projecting its mesh vertices onto a concentric spherical surface along the radius direction. In fact, any convex quadrilateral mesh can be generated without flipping from the cubical or spherical mesh by projecting its mesh vertices along appropriately defined directions. For ease of explanation, closed contours in different groups are still referred to as orthogonal to each other although they are not necessarily physically orthogonal.

The essence of such a construction is that quadrilateral mesh allows a linear ordering of the mesh vertices to be defined along any closed contour. Linear ordering of vertices on a closed contour helps to simplify the detection and avoidance of possible flippings, which will be discussed in the next section. In contrast, resolution of flippings in a triangular mesh is much more complex because there is no general way to define linear orderings of its vertices.

The quadrilateral mesh is regular in the sense that it is pole-free. In contrast, the UV sphere (Fig. 4.2) that is widely adopted by the computer graphics community, contains two poles where many mesh edges intersect. The existence of poles may cause difficulties and complications in the algorithm.

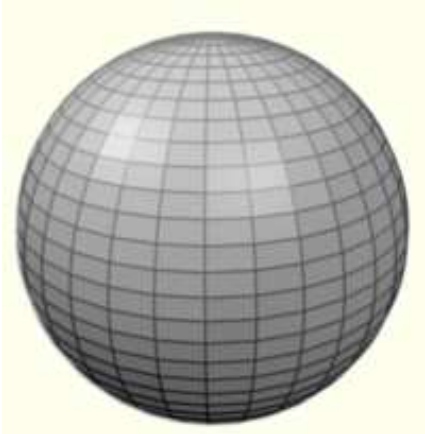


Figure 4.2: UV sphere. Multiple edges intersects at the north and south poles (image from `http://www.rab3d.com/rab3d/tutorial/608/tutorial_608-3.html`).

4.2 Flipping-free Quadrilateral Mesh Deformation

4.2.1 Algorithm Overview

To facilitate explaining the ideas, a list of symbol definition is given. These symbols are used throughout the thesis.

- $I = I(\mathbf{x})$: an input volume image, where $I(\mathbf{x})$ denotes the intensity of the voxel at location $\mathbf{x}$,
- $M = \{\mathbf{u}_i\}$: a 3D mesh, where $\mathbf{u}_i = [x_i, y_i, z_i]^T$ denotes the 3D coordinates of the vertex i of M.
- M': a 3D mesh representing the extracted surface of the segmented organ.

For ease of explanation, this section presents the flipping-free mesh deformation algorithm as an algorithm that registers the quadrilateral mesh M to the known surface of a 3D volumetric object T. In each iteration, the algorithm searches for possible correspondences between mesh vertices and object surface points over long distances. The detected correspondences are refined before mesh deformation to avoid flipping. Application of the mesh deformation algorithm to segmentation will be discussed in Chapter 5.

Algorithm 1 Flipping-free mesh deformation algorithm.

Mesh Initialization.
Repeat until convergence:
  Search for correspondence (Section 4.2.2).
  For each contour in each group
    Detect possible flippings (Section 4.2.3).
    Avoid possible flippings (Section 4.2.4).
  Perform mesh deformation (Section 4.2.5).

4.2.2 Correspondence Search

This stage searches for the correspondence between the model M and the target T. For each vertex $\mathbf{u}_i$ on M, the algorithm searches along the projection line $P(\mathbf{u}_i)$ for a possible corresponding point $\mathbf{v}_i$ on the surface of T. The direction of $P(\mathbf{u}_i)$ can be defined as the surface normal at $\mathbf{u}_i$. The vector $\mathbf{v}_i - \mathbf{u}_i$ is the *displacement vector* of $\mathbf{u}_i$. If the algorithm cannot find any correspondence for $\mathbf{u}_i$, $\mathbf{u}_i$ is labeled as a *solitary* vertex; otherwise, $\mathbf{u}_i$ is labeled as a *non-solitary* vertex.

Note that the mesh M can be initialized either completely inside or completely outside T. The search directions are therefore changed accordingly. The correspondence search stage may find different number of corresponding points on the surface of T along the search directions and within the search distance. As shown in Fig. 4.3, within the search distance, vertex a has no corresponding point, vertex b and c have 1 and 2 corresponding points respectively.

4.2.3 Flip Detection

The flipping of a mesh cell after mesh deformation is characterized by the flipping of at least one of its edges. Therefore, surface flipping can be identified by detecting edge flipping. Let $\mathbf{u}_i$ and $\mathbf{u}_j$ denote two non-solitary neighbors on a closed contour, and $\mathbf{v}_i$ and $\mathbf{v}_j$ denote their corresponding points on the target object. Then, edge flipping occurs when the orientations of the edges $\mathbf{u}_i - \mathbf{u}_j$ and $\mathbf{v}_i - \mathbf{v}_j$ differ significantly:

$$\frac{\mathbf{u}_i - \mathbf{u}_j}{\|\mathbf{u}_i - \mathbf{u}_j\|} \cdot \frac{\mathbf{v}_i - \mathbf{v}_j}{\|\mathbf{v}_i - \mathbf{v}_j\|} \leq \tau, \tag{4.2.1}$$

where $\tau \in [0, 1)$ is a predefined threshold. The vertices $\mathbf{u}_i$ and $\mathbf{u}_j$ that form a flipping

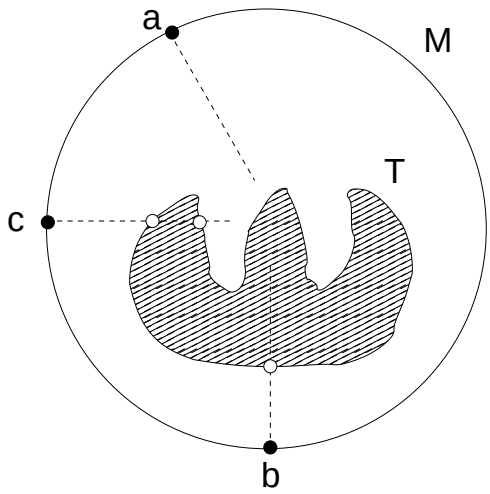
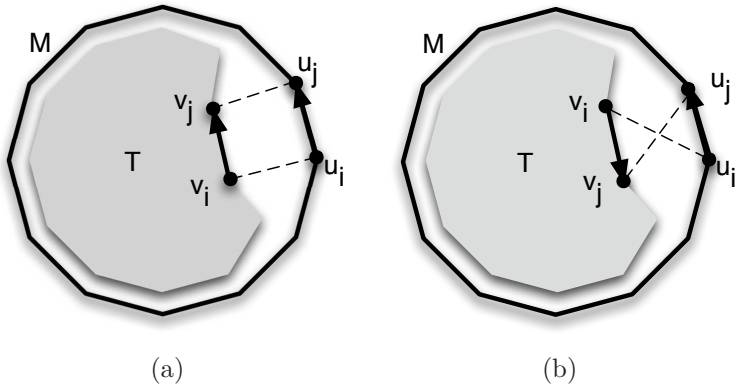


Figure 4.3: Search for correspondence. The mesh M is initialized completely outside the target T. Within the search distance, vertex a has no corresponding point, vertex b and c have 1 and 2 corresponding points respectively.



(a) (b)

Figure 4.4: Flip detection. Flip of the normal may be characterized by abrupt change of the orientation of the edge before ($\mathbf{u}_i\mathbf{u}_j$) and after ($\mathbf{v}_i\mathbf{v}_j$) the deformation. (a) No abrupt edge orientation change, no flipping. (b) Abrupt edge orientation change, flipping.

edge are labeled as *flipping* vertices; otherwise, *non-flipping* vertices.

Each vertex $\mathbf{u}_i$ is an intersection of two orthogonal closed contours on M. Therefore, each $\mathbf{u}_i$ will undergo flip detection along both closed contours when the algorithm iterates. It is labelled as a flipping vertex if it forms a flipping edge along any one of the two contours.

4.2.4 Flip Avoidance

The basic idea of flip avoidance is to discard the point correspondences of all flipping vertices. However, this may prevent the mesh model converging to concave target such as shown in Fig. 4.5. Let $\mathbf{u}_i$, $\mathbf{u}_{i+1}, \ldots, \mathbf{u}_n$ denote a consecutive sequence of the flipping vertices on a closed contour, excluding solitary vertices, such that $\mathbf{u}_{i-1}$ and $\mathbf{u}_{n+1}$ are non-flipping. The method identifies the middle flipping vertex $\mathbf{u}_m$ of the sequence, labels it as non-flipping, and labels the other flipping vertices as solitary, i.e., discarding their correspondences. As shown in Fig. 4.5, $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ are a consecutive sequence of flipping vertices. Therefore, $\mathbf{u}_2$ as the middle vertex, is labeled non-flipping. $\mathbf{u}_1$ and $\mathbf{u}_3$ are labeled solitary. Note that labeling $\mathbf{u}_m$ alone as non-flipping does not cause flippings because these solitary vertices will be displaced in such a way that their local surface shapes are preserved (Section 4.2.5). After repeating this process for every closed contour, only non-flipping vertices have point correspondences. The rest of the vertices are deformed by local geometric constraints. Thereafter, deforming the mesh according to these correspondences does not result in flipping.

This approach can handle flipping self-intersection but cannot handle non-flipping self-intersection or folding. This can be demonstrated in Fig. 4.6, where under some extreme cases, non-flipping self-intersections occur. To alleviate this problem, the displacement vectors of non-flipping vertices are propagated to neighboring solitary vertices, turning them into non-flipping vertices, by iterative local averaging of displacement vectors:

$$\tilde{\mathbf{v}}'_i = \mathbf{v}_i + \frac{1}{N+1}\left[\mathbf{v}'_i - \mathbf{v}_i + \sum_{v_j \in \mathcal{N}(\mathbf{v}_i)} (\mathbf{v}'_j - \mathbf{v}_j)\right], \tag{4.2.2}$$

where $\mathbf{v}_i$ is the corresponding point of $\mathbf{u}_i$, and $\mathbf{v}'_i$ is the corresponding point after diffusion.

This process is analogous to the diffusion of gradient vectors in gradient vector flow

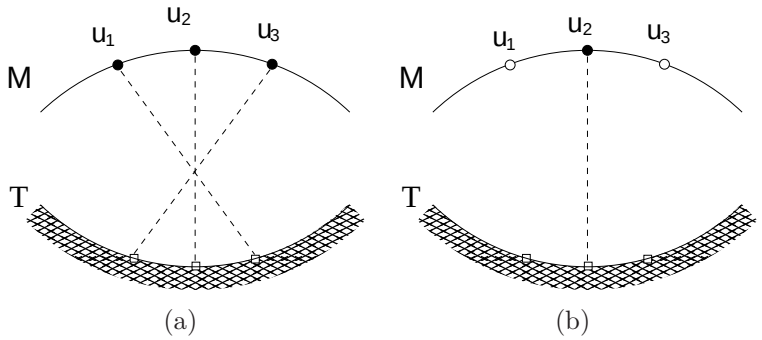


Figure 4.5: Flip avoidance. (a) $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ are a consecutive sequence of flipping vertices. (b) To avoid flipping, $\mathbf{u}_2$ as the middle vertex, is labeled non-flipping; $\mathbf{u}_1$ and $\mathbf{u}_3$ are labeled solitary.

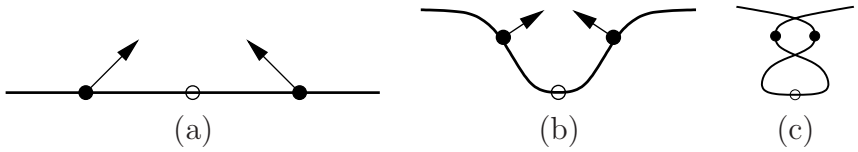
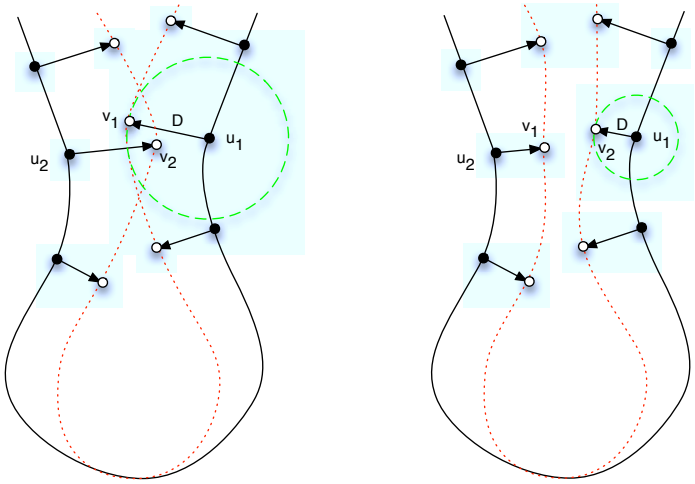


Figure 4.6: Folding problem. (a) Displacing non-flipping vertices (dots) around solitary vertices (circle) may cause (b) folding of the mesh, and in the extreme case, (c) non-flipping self-intersection.

for the snake algorithm [XP98]. It also smoothens the variation of displacement vectors among neighboring non-flipping vertices, thus improving noise resilience.

To further prevent non-flipping self-intersection, for each vertex $\mathbf{u}_i$, the algorithm searches for any neighboring vertex and its corresponding point within a certain distance $\mathcal{D}$ (Fig. 4.7). $\mathcal{D}$ is equal to $\|\mathbf{v}_i - \mathbf{u}_i\|$. If any vertex or its corresponding point within $\mathcal{D}$ is found, e.g., in Fig. 4.7, $\|\mathbf{v}_2 - \mathbf{u}_1\| < \mathcal{D}$, the two displacement vectors $\mathbf{v}_1 - \mathbf{u}_1$ and $\mathbf{v}_2 - \mathbf{u}_2$ are used to examine whether possible non-flipping self-intersection will happen.

Noticing that such self-intersection can be identified by large orientation difference of the two vectors, the same criteria as Eqn. 4.2.1 is used to test any possibility. Deformation according to the displacement vector $\mathbf{v}_1 - \mathbf{u}_1$ and $\mathbf{v}_2 - \mathbf{u}_2$ will cause the original mesh surface (black solid line) to form non-flipping self-intersection (red dotted line). Based on the test, if the displacement vectors may cause non-flipping self-intersection, the lengths of both displacement vectors are reduced by half iteratively until no intersection is going



(a) (b)

Figure 4.7: Non-flipping self-intersection. (a) Deformation according to the displacement vector $\mathbf{v}_1 - \mathbf{u}_1$ and $\mathbf{v}_2 - \mathbf{u}_2$ will cause the original mesh surface (black solid line) to form non-flipping self-intersection (red dotted line). The algorithm identifies that $\mathbf{v}_2$ is within the search sphere (green dashed line), and $\mathbf{v}_1 - \mathbf{u}_1$ and $\mathbf{v}_2 - \mathbf{u}_2$ have large orientation difference. (b) Reducing the lengths of both displacement vectors prevents the intersection.

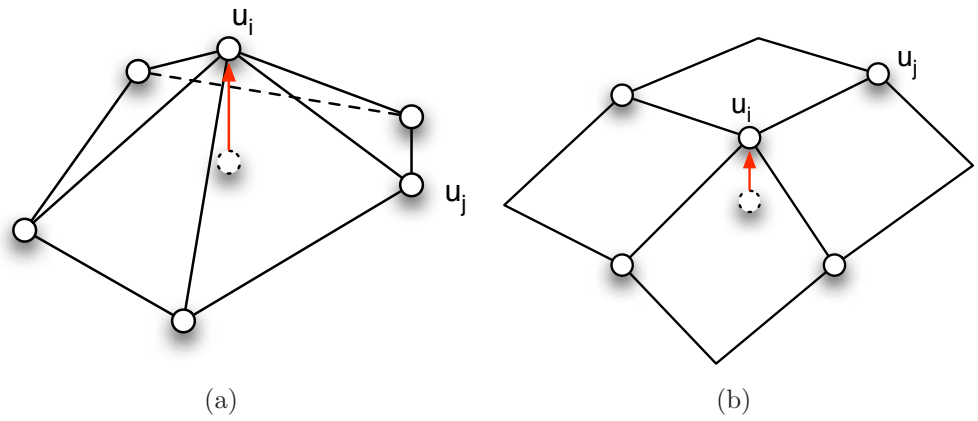
to happen (Fig. 4.7(b)).

4.2.5 Laplacian Deformation

The Laplacian method [SLCO+04] is adopted for mesh deformation because it is very efficient, easy to use, and easy to incorporate geometric constraints. During deformation, non-flipping vertices are displaced towards their target locations, which are regarded as *positional constraints*. The other mesh vertices are displaced according to geometric constraints including *Laplacians preservation* and *uniform vertex distribution*. The deformation problem is then formulated as minimizing the energy

$$E = \lambda_L E_L + \lambda_p E_p + \lambda_u E_u \tag{4.2.3}$$

where E_L, E_p and E_u are energies for imposing Laplacian preservation, positional constraint and uniform vertex distribution respectively. Weighting parameters λ_L, λ_p and λ_u are chosen empirically.



(a) (b)

Figure 4.8: Laplacian operator. Laplacian of (a) triangular mesh and (b) quadrilateral mesh. Solid circles: mesh vertices. Dashed circle: centroid of the neighboring vertices $\mathbf{u}_j$ of $\mathbf{u}_i$. Red arrow: Laplacian $L(\mathbf{u}_i)$ of $\mathbf{u}_i$.

The Laplacian preservation energy E_L is defined as:

$$E_L = \sum_i \|L(\mathbf{u}_i) - L(\mathbf{u}'_i)\|_2^2, \tag{4.2.4}$$

where $\mathbf{u}'_i$ denotes the position of vertex $\mathbf{u}_i$ after deformation. The Laplacian operator L is defined as

$$L(\mathbf{u}_i) = \mathbf{u}_i - \frac{1}{|\mathcal{N}(\mathbf{u}_i)|} \sum_{\mathbf{u}_j \in \mathcal{N}(\mathbf{u}_i)} \mathbf{u}_j, \tag{4.2.5}$$

where $\mathcal{N}(\mathbf{u}_i)$ denotes the connected neighboring vertices of $\mathbf{u}_i$ and $|\mathcal{N}(\mathbf{u}_i)|$ denotes the number of the neighboring vertices. $L(\mathbf{u}_i)$ encodes the local shape information of $\mathbf{u}_i$. Fig. 4.8 shows the Laplacian of $\mathbf{u}_i$ in a triangular mesh (a) and a quadrilateral mesh (b). Minimizing E_L results in local shape preservation.

The positional energy E_p imposes the positional constraint, and is defined as:

$$E_p = \sum_i \frac{d_i}{\bar{d}} \|\mathbf{u}'_i - \mathbf{p}_i\|_2^2, \tag{4.2.6}$$

where $\mathbf{p}_i$ denotes the target position of $\mathbf{u}_i$, d_i is the distance from $\mathbf{u}_i$ to $\mathbf{p}_i$, $\bar{d}$ is the average of d_i for $\mathbf{u}_i$ of M. Minimizing E_p displaces mesh vertices towards their designated locations. In addition, positional constraints are weighted according to the distance between $\mathbf{u}_i$ and $\mathbf{p}_i$, i.e., $d_i/\bar{d}$. The further the corresponding point, the larger the weight.

This weighting scheme ensures that the model can deform towards the target surface quickly.

The uniform vertex distribution constraint can be formulated as minimizing the energy E_u:

$$E_u = \sum_{C} \sum_{\mathbf{u}_i, \mathbf{u}_j \in C}^{\mathbf{u}_j \in \mathcal{N}(i)} \|(\mathbf{u}'_i - \mathbf{u}'_j) - \frac{\bar{l}}{l_{ij}}(\mathbf{u}'_i - \mathbf{u}'_j)\|_2^2, \tag{4.2.7}$$

where $\bar{l}$ is the average distance between $\mathbf{u}'_i$ and $\mathbf{u}'_j$ estimated based on the length of a closed contour consisting of the corresponding points. l_{ij} is the current edge length between $\mathbf{u}_i$ and $\mathbf{u}_j$ on a contour C. If l_{ij} is equal to $\bar{l}$, i.e., the current edge length l_{ij} between vertex $\mathbf{u}_i$ and $\mathbf{u}_j$ is equal to the estimated average edge length $\bar{l}$, the term $\|(\mathbf{u}'_i - \mathbf{u}'_j) - (\mathbf{u}'_i - \mathbf{u}'_j)\bar{l}/l_{ij}\|_2^2$ goes to zero. Otherwise, the larger the difference between l_{ij} and $\bar{l}$, the larger the term $\|(\mathbf{u}'_i - \mathbf{u}'_j) - (\mathbf{u}'_i - \mathbf{u}'_j)\bar{l}/l_{ij}\|_2^2$. Therefore, minimizing E_u enables the vertices to distribute more evenly over the mesh surface.

When the mesh vertices are close to the target surface, the positional constraints become small (Eqn. 4.2.6). In comparison, the constraint of uniform vertex distribution becomes more significant, and it minimizes the difference between the length l_{ij} of a mesh edge and the average length $\bar{l}$. Thus, the mesh vertices can displace along directions tangential to the mesh surface and distribute more evenly.

In the following, we demonstrate how the matrices in the system of equations are constructed and how they are solved. Consider part of the mesh surface as shown in Fig. 4.9, which labels 5 vertices as $\mathbf{u}_1$ to $\mathbf{u}_5$, in which $\mathbf{u}_2$ to $\mathbf{u}_5$ are 4 neighboring vertices of $\mathbf{u}_1$.

In order to compute preservation of Laplacians, current Laplacians need to be computed, which is obtained by multiplication of the adjacency matrix $\mathbf{A}$ and current Cartesian coordinates $\mathbf{x}$ of all the vertices. The adjacency matrix $\mathbf{A}$ representing neighboring conditions is constructed as follows:

$$\mathbf{A}_{n\times n} = \begin{pmatrix} \lambda_L & -\lambda_L/4 & -\lambda_L/4 & -\lambda_L/4 & -\lambda_L/4 & 0 & \dots & 0 \\ & & & & & & & \\ & & & & & & & \\ \vdots & & & & & & & \ddots \end{pmatrix} \tag{4.2.8}$$

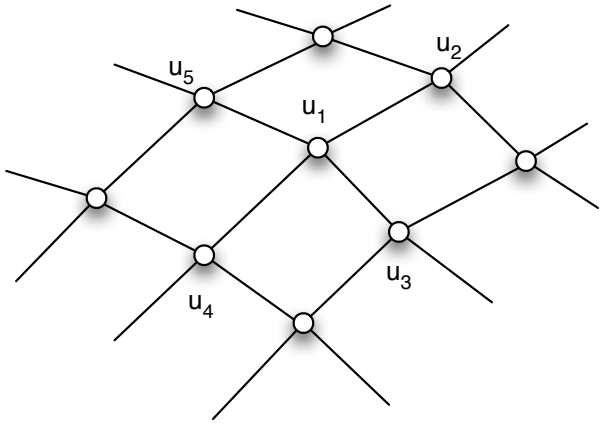


Figure 4.9: Example mesh configuration. $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$ and $\mathbf{v}_5$ are neighboring vertices of $\mathbf{v}_1$.

Row i of $\mathbf{A}$ corresponds to vertex $\mathbf{u}_i$, and column j in row i corresponds to the coefficient of $\mathbf{u}_j$ that contributes to computing $L(\mathbf{u}_i)$. These coefficients are set based on Eqn. 4.2.5.

Row i of $\mathbf{x}$ contains the Cartesian coordinates of $\mathbf{u}_i$.

$$\mathbf{x}_{n\times 3} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \mathbf{u}_3^T \\ \mathbf{u}_4^T \\ \mathbf{u}_5^T \\ \vdots \\ \mathbf{u}_n^T \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{pmatrix} \tag{4.2.9}$$

Therefore, given $\mathbf{A}$ and $\mathbf{x}$, the Laplacian coordinates $\mathbf{l}$ of mesh vertices can be computed as:

$$\mathbf{l}_{n\times 3} = \mathbf{A}\mathbf{x} \tag{4.2.10}$$

The purpose is to solve the Cartesian coordinates $\mathbf{x}'$ of vertices after deformation. Base on the current Laplacian coordinate $\mathbf{l}$, the system of equations can be written as:

$$\mathbf{A}\mathbf{x}' = \mathbf{l} \tag{4.2.11}$$

Suppose that m_1 vertices have their estimated correspondences. These correspondences function as the positional constraints. To incorporate the positional constraints

into the system, i.e., minimizing $\lambda_p E_p$, the following equation needs to be solved:

$$\mathbf{B}\mathbf{x}' = \mathbf{p}, \tag{4.2.12}$$

where

$$\mathbf{B}_{m_1 \times n} = \begin{pmatrix} \lambda_p p_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \lambda_p p_2 & 0 & 0 & \dots & 0 \\ \vdots & & & & & \end{pmatrix}, \tag{4.2.13}$$

and

$$\mathbf{p}_{m_1 \times 3} = \begin{pmatrix} \lambda_p x'_1 & \lambda_p y'_1 & \lambda_p z'_1 \\ \lambda_p x'_2 & \lambda_p y'_2 & \lambda_p z'_2 \\ & \dots & \end{pmatrix} \tag{4.2.14}$$

To incorporate vertex uniform distribution constraints, i.e., minimizing $\lambda_u E_u$, the following equation needs to be solved:

$$\mathbf{C}\mathbf{x}' = \mathbf{d}, \tag{4.2.15}$$

where

$$\mathbf{C}_{m_2 \times n} = \begin{pmatrix} \lambda_u & -\lambda_u \frac{\bar{l}}{l_{12}} & 0 & 0 & \dots & 0 \\ 0 & \lambda_u & -\lambda_u \frac{\bar{l}}{l_{23}} & 0 & \dots & 0 \\ \vdots & & & & & \end{pmatrix}, \tag{4.2.16}$$

and

$$\mathbf{d}_{m_2 \times 3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ & \dots & \end{pmatrix} \tag{4.2.17}$$

To minimize the total energy E (Eqn. 4.2.3), the entire system of equations to be solved is:

$$\begin{pmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{pmatrix} \mathbf{x}' = \begin{pmatrix} \mathbf{l} \\ \mathbf{p} \\ \mathbf{d} \end{pmatrix}. \tag{4.2.18}$$

For ease of explanation, let

$$\mathbf{H} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{pmatrix}, \tag{4.2.19}$$

and

$$\mathbf{b} = \begin{pmatrix} \mathbf{l} \\ \mathbf{p} \\ \mathbf{d} \end{pmatrix}. \tag{4.2.20}$$

Therefore, the entire system is written as

$$\mathbf{H}\mathbf{x}' = \mathbf{b}. \tag{4.2.21}$$

The usual approach to solve the above equation is to use pseudo inverse,

$$\mathbf{x}' = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{b}. \tag{4.2.22}$$

However, evaluating $(\mathbf{H}^T\mathbf{H})^{-1}$ is a prohibitive operation since $\mathbf{H}^T\mathbf{H}$ is a huge matrix.

Fortunately, $\mathbf{H}^T\mathbf{H}$ is a sparse matrix, the system can be solved with the help of Cholesky factorization [HTVF92] of $\mathbf{H}$, which is very efficient. Cholesky factorization factorizes $\mathbf{H}^T\mathbf{H}$ into the multiplication of a lower triangular matrix and its transpose:

$$\mathbf{H}^T\mathbf{H} = \mathbf{L}\mathbf{L}^T. \tag{4.2.23}$$

Cholesky factorization can be computed very efficiently by the Taucs solver (`http://www.tau.ac.il/~stoledo/taucs/`).

The system of equations can be re-written as

$$\mathbf{L}\mathbf{L}^T\mathbf{x}' = \mathbf{H}^T\mathbf{b}. \tag{4.2.24}$$

Noticing that $\mathbf{L}^T$ is a triangle matrix, Let

$$\mathbf{y} = \mathbf{L}^T\mathbf{x}'. \tag{4.2.25}$$

$\mathbf{y}$ can be simply solved by back substitution.

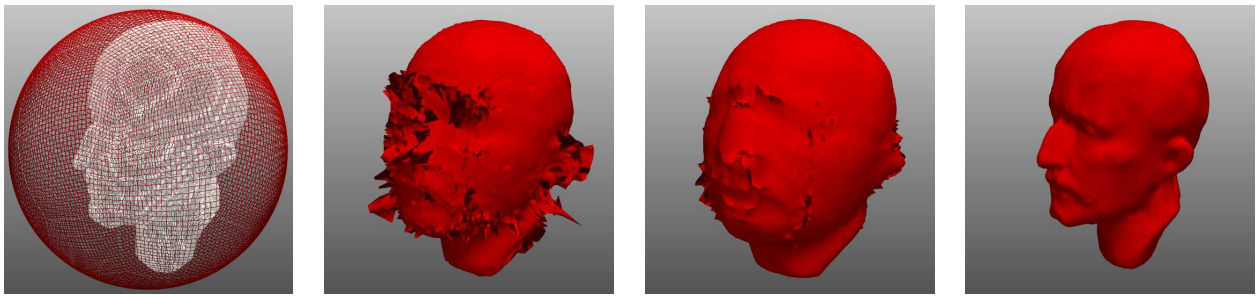
Substitute $\mathbf{y}$ into Eqn. 4.2.24,

$$\mathbf{L}\mathbf{y} = \mathbf{H}^T\mathbf{b}. \tag{4.2.26}$$

Again, $\mathbf{L}$ is a triangle matrix. Therefore, $\mathbf{x}'$ can be solved by performing back substitution again. The non-linear total energy E is minimized using Gauss-Newton iteration.

4.3 Experiments and Results

Experiments were carried out to test the algorithm's capabilities of flip avoidance (Section 4.3.1), convergence to deeply concave object (Section 4.3.2), uniform vertex distribution (Section 4.3.2) and time complexity (Section. 4.3.4).



(a) (b) (c) (d)

Figure 4.10: Registration results of a naive method and the proposed method. (a) Initialization for a binary volume image. (b) Naive method with $\lambda = 0.3$ and 4 iterations. (c) Naive method with $\lambda = 0.01$ and 100 iterations. In (b, c), surface flipping causes discontinuities. (d) No flipping occurs using the proposed algorithm.

4.3.1 Flip Avoidance

The objective of this experiment is to demonstrate the algorithm's capabilities to avoid flipping when registering the spherical mesh to the surface of a binary volume object.

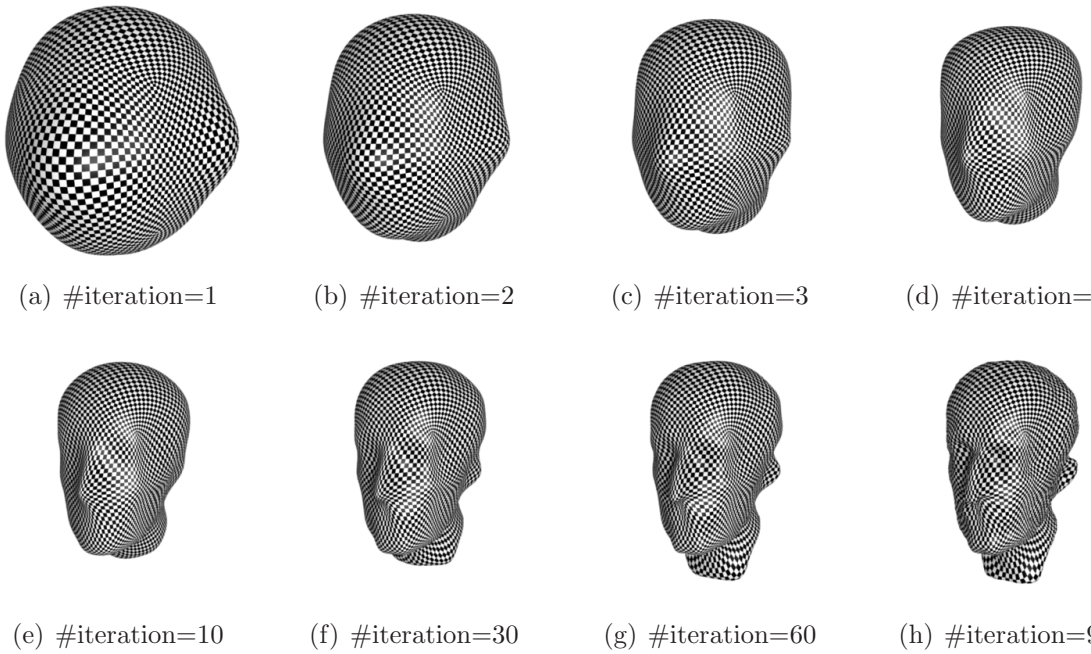
The target volume object is a human head (Maxplanck), which contains multiple convex and concave parts on this surface. The resolution of the volume image is $256 \times 256 \times 256$. The initial sphere was placed outside of the object (Fig. 4.10(a)).

The spherical mesh was deformed first using a naive method, i.e., moving each vertex to its corresponding position directly without flip avoidance. The naive algorithm was stopped once flippings of the surface normals are clearly visible. Test results show that even with a very small deformation step size, surface discontinuities caused by flipping occur (Fig. 4.10(b, c)).

The deformed mesh surface was examined qualitatively for any flippings of surface normals. In comparison, with the same initialization, the mesh was deformed using the naive deformation method.

In comparison, the spherical mesh was deformed using the proposed flipping-free algorithm. The algorithm was stopped until all the vertices are fit on to the surface. i.e., the algorithm converges.

Test result shows that registration using the proposed algorithm is flipping-free (Fig. 4.10(d)).



(a) #iteration=1 (b) #iteration=2 (c) #iteration=3 (d) #iteration=5

(e) #iteration=10 (f) #iteration=30 (g) #iteration=60 (h) #iteration=99

Figure 4.11: Registration of the quadrilateral mesh to the maxplanck volume. (a)–(g) Intermediate deformation process. (h) Final result.

To demonstrate that the proposed algorithm is free from flipping during deformation, several intermediate deformation results are shown in Fig. 4.11. For better illustration purpose, the meshes are applied with checker board pattern. Fig. 4.12 also shows that the registration error decreases rapidly as the algorithm iterates. The results show that the proposed algorithm can avoid flippings during deformation.

4.3.2 Convergence to Deeply Concave Objects

The objective of this experiment is to test the algorithm's capabilities to handle concave target object and to demonstrate its robustness under different parameter values, e.g., deformation step size (α), mesh resolution (r), iteration number (n), position constraint weight (λ_p) and Laplacian constraint weight (λ_L). The mesh resolution is defined as the number of contours (n) in one of the three groups that construct the mesh, i.e., the number of vertices in the mesh is $6 \cdot n^2$ if each group has exactly the same number of contours.

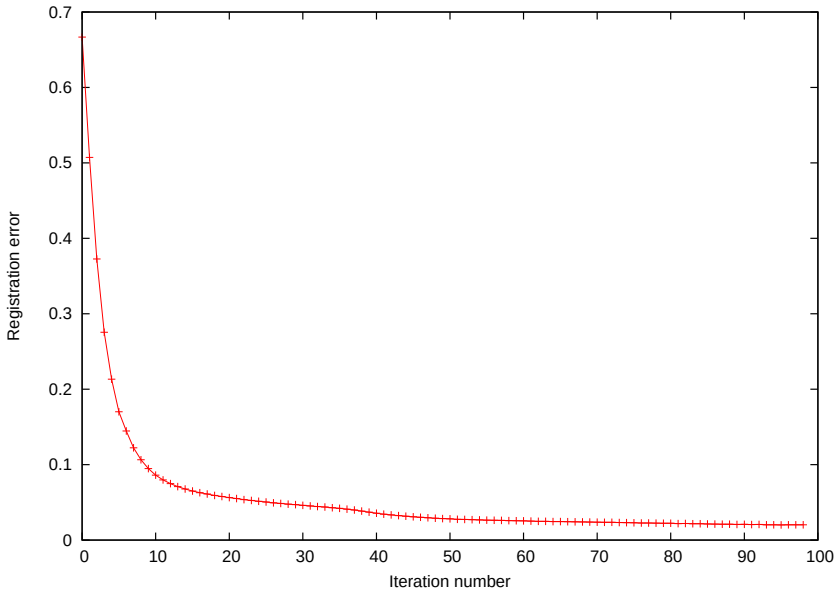


Figure 4.12: Registration error over iterations: registration of the quadrilateral mesh to the maxplanck volume. The error drops sharply in the first a few iterations, and then gradually converges.

The target image is a binary cup volume (Fig. 4.13) which has a deep concave part. The resolution of the volume image is $256 \times 256 \times 256$. The initial spherical mesh was placed outside of the cup volume manually.

The quality of convergence is measured using the registration error. The lower the registration error, the higher the quality of convergence. The registration error (E_r) is measured based on the overlap between the ground truth volume (A) and the registration results (B):

$$E_r = 1 - \frac{|A \cap B|}{|A| + |B|}, \tag{4.3.1}$$

where $|\cdot|$ represents the volume of $\cdot$. E_r varies between 0 and 1. In order to compute the volume of the deformed mesh after registration, the mesh was converted to another binary volume image.

Robustness to Mesh Resolution and Deformation Step Size Changes

The registration was carried out multiple times using various combinations of different mesh resolution r and deformation step size α, with $r \in [21, 29]$ and $\alpha \in [0.1, 0.9]$. In all

Figure 4.13: Cup volume.

the test cases, the algorithm was executed a fixed number of iterations (99).

between the registration error and the mesh resolution (r) and deformation step size (α).

The results, i.e., the registration errors are plotted as a 3D surface (Fig. 4.15), where the light orange area corresponds to higher registration errors and the dark blue area corresponds to lower registration errors. As can be observed in this figure, the mesh models with lower resolution tend to produce larger registration errors compared to those with higher resolution. This can be attributed to mesh models not having enough vertex to represent the target volume. This figure also demonstrates that a very small deformation step size, i.e., $\alpha = 0.1$ or $\alpha = 0.2$, results in larger registration errors. It seems that the algorithm needs a few more iterations to converge using small α. The algorithm using a mesh with larger r and adopting a larger α generally yields smaller registration errors (the dark-blue flat region in the figure). The lowest error plotted in Fig. 4.15 is about 0.03. This error can be generated from converting the mesh to a discretely represented volume image. The plotted error surface is smooth, which suggests the stability of the proposed algorithm even if the resolution n and deformation step size r are changed in a broad range. Fig. 4.16 shows some intermediate deformation process of the mesh. Test results also show that the proposed algorithm can handle concave objects properly.

Fig. 4.17 shows the convergence curves/surfaces under different deformation step size.

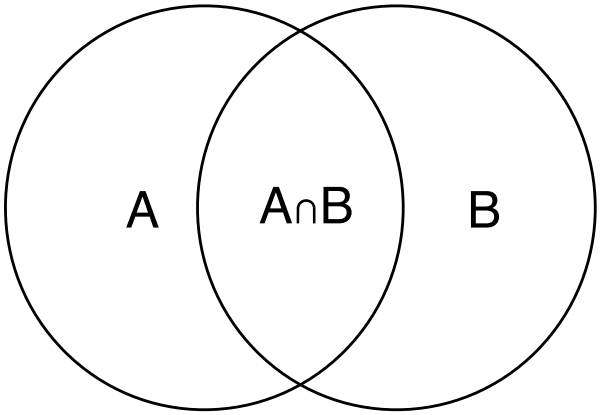


Figure 4.14: Error measure. The registration error E_r is measured based on the overlap between the ground truth (A) and the registration result (B). $E_r = 1 - \frac{|A \cap B|}{|A|+|B|}$.

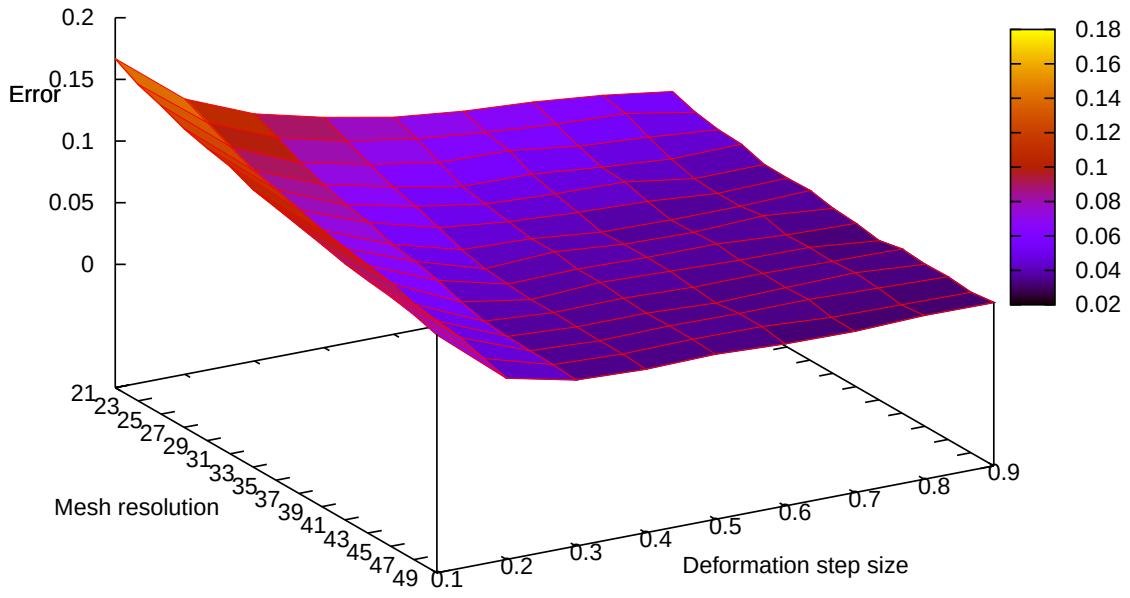
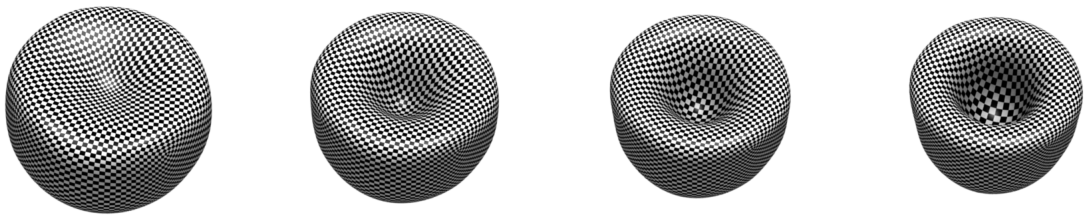
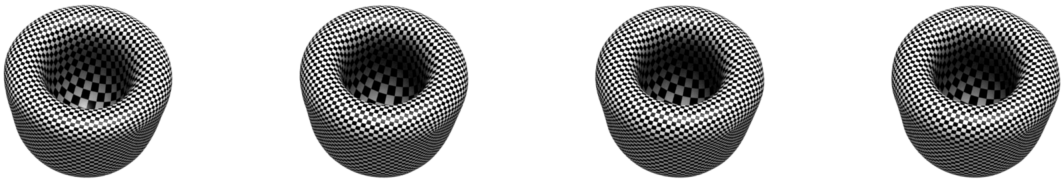


Figure 4.15: Robustness to mesh resolution and deformation step size changes when Registering a spherical quadrilateral mesh to the cup volume. The dark-blue region represents lower error. The errors are plotted regarding to different mesh resolution and deformation step size.



(a) #iteration=1 (b) #iteration=2 (c) #iteration=3 (d) #iteration=5



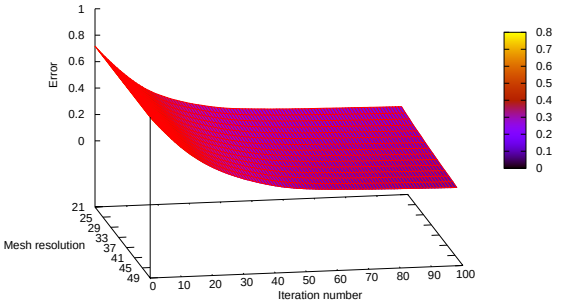
(e) #iteration=10 (f) #iteration=30 (g) #iteration=60 (h) #iteration=99

Figure 4.16: Registration of the quadrilateral mesh to a cup volume. (a)–(g) Intermediate deformation process. (h) Final result.

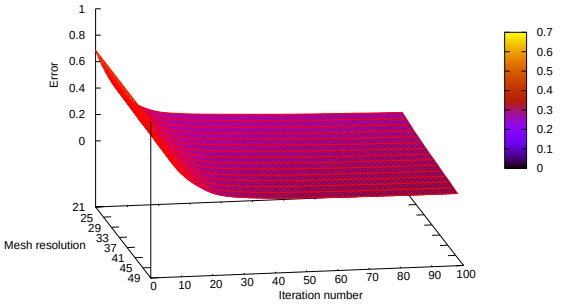
To facilitate viewing the convergence data, the figure is composed of 9 plots, each corresponding to a fixed deformation step size varying from 0.1 to 0.9. As can be observed in the figure, firstly, in all the plots, the errors decrease as the iteration number increases. This phenomenon demonstrates that the proposed algorithm can converge well. Secondly, Fig. 4.17(a) shows a smoothly descending surface. In contrast, Fig. 4.17(i) shows a rapidly descending surface. This suggests that the algorithm can converge faster if a larger α is chosen. Thirdly, the mesh resolution has relatively small impact on the converge property, but we still find that mesh models with higher resolution converges faster. These observations give some clues for the user to apply better parameter values. A higher mesh resolution and a larger deformation step size seem to help the algorithm converge faster and achieve lower errors.

Robustness to Weight of Positional Constraint and Laplacian constraint

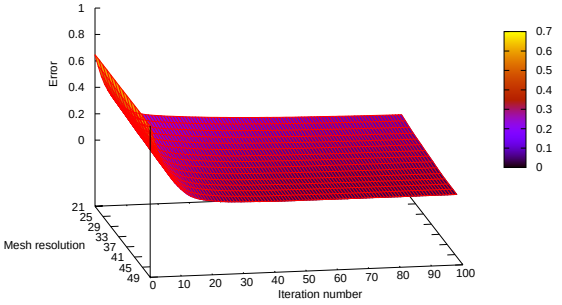
This test is to show that the proposed algorithm is not sensitive to the weight of the positional constraint (λ_p) and the weight of the positional constraint (λ_L).



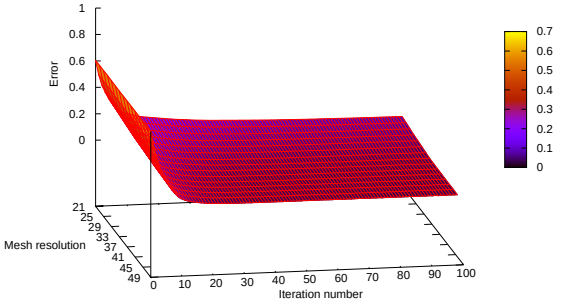
(a) $\alpha = 0.1$



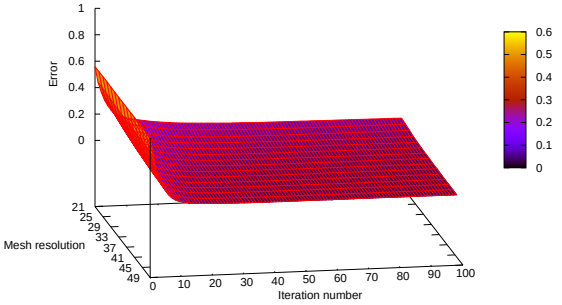
(b) $\alpha = 0.2$



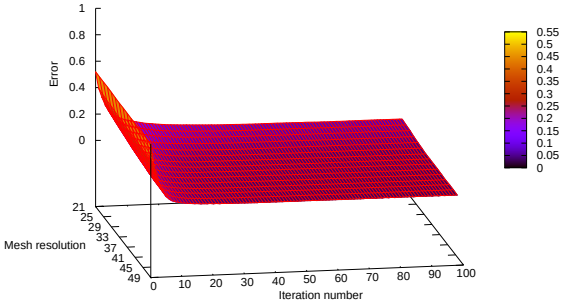
(c) $\alpha = 0.3$



(d) $\alpha = 0.4$

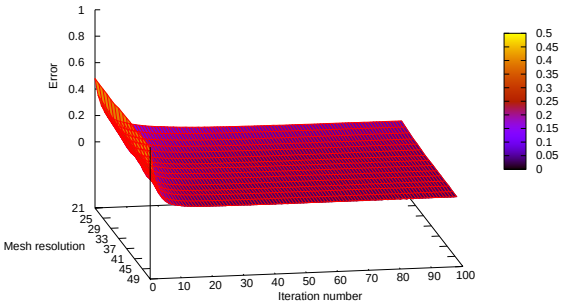


(e) $\alpha = 0.5$

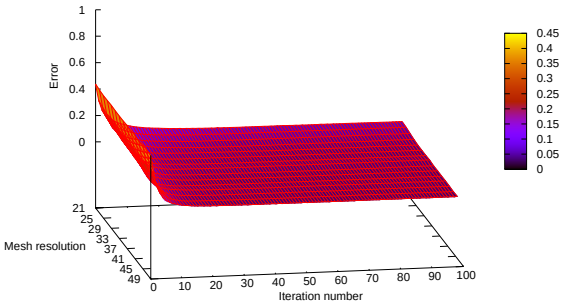


(f) $\alpha = 0.6$

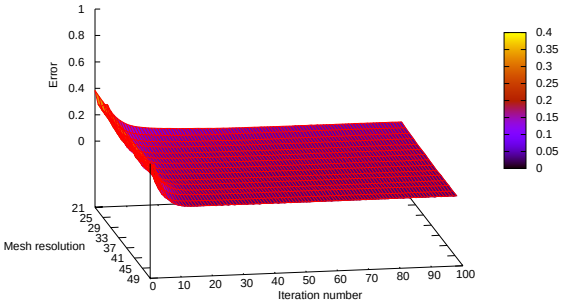
Figure 4.17: Robustness to deformation step size (α) changes. α varies from 0.1 to 0.9. The larger the α, the faster the algorithm converges. (a–i) Plot of registration errors with respect to different mesh resolution and iteration number.



(g) $\alpha = 0.7$



(h) $\alpha = 0.8$



(i) $\alpha = 0.9$

Figure 4.17: Continued.

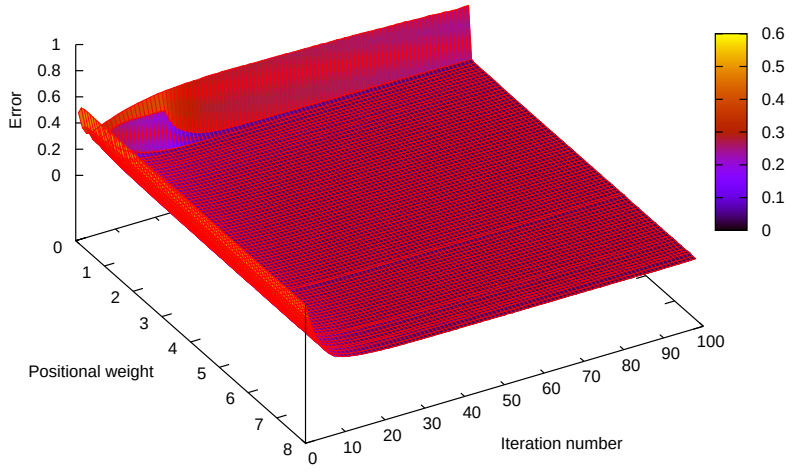


Figure 4.18: Convergence with different positional weights. For most weight values, the algorithm can converge successfully. However, it fails to converge when the weight is 0.1, which is too small. Larger positional weight results in faster convergence speed.

The mesh resolution was fixed at $r = 39$, the step size $\alpha = 0.5$, and the Laplacian weight $\lambda_L = 4$. λ_p was varied from 0.1 to 8. For each λ_p, the registration error in each deformation is plotted as shown in Fig. 4.18. This figure demonstrates that for most of the λ_p values, the deformation algorithm can converge well. For larger λ_p, the algorithm converges a bit faster. However, when $\lambda_p = 0.1$, the algorithm fails to converge simply because the positional constraints is too small and they are compromised by other constraints such as the Laplacian constraints. When $\lambda_p = 0.2$, the algorithm has some problem to converge at the first few iterations, but converges successfully finally. This test suggests the user to select a weight value that is not too small.

This test is to show that the proposed algorithm is not sensitive to the weight of the positional constraint (λ_L). The mesh resolution is fixed at $r = 39$, the step size $\alpha = 0.5$, and the Laplacian weight $\lambda_p = 4$. λ_L was varied from 0.1 to 8. For each λ_L, the registration error in each deformation is plotted as shown in Fig. 4.19. This figure demonstrates that for all λ_L, the deformation algorithm can converge well. A smaller λ_L results in slightly faster convergence of the algorithm.

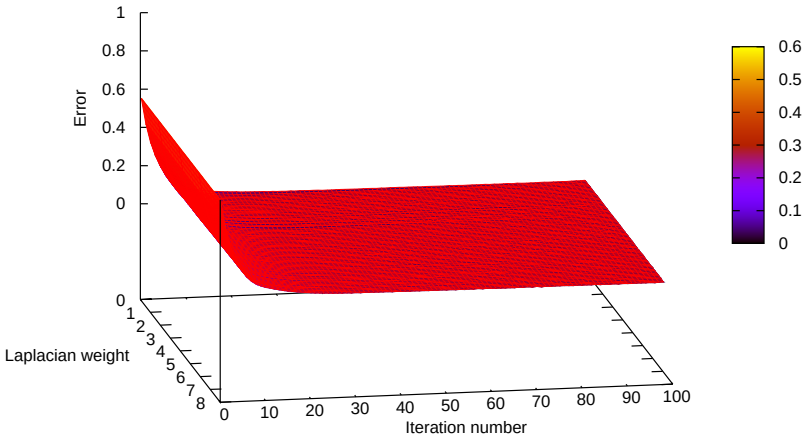


Figure 4.19: Convergence with different Laplacian weights. The algorithm can converge successfully at all weight values. Smaller Laplacian weight results in slightly faster convergence speed.

4.3.3 Uniform Vertex Distribution

The objective of this test is to investigate the uniform vertex distribution property of the algorithm with different mesh resolutions and deformation step sizes. The input volume image is the cup volume since it contains both convex and deeply concave part. The uniformity is measure by the variance of all the edge lengths in the deformed mesh. The smaller the variance, the more uniform the vertices are distributed. The test is also to verify the effectiveness of the weight of uniform vertex distribution. The target image is the binary cup volume.

The proposed algorithm was executed with different values of mesh resolution and deformation step size. The results are evaluated using volume overlap error.

In Fig. 4.20, the variance is plotted as a function of mesh resolution and deformation step size. As can be observed, lower mesh resolution corresponds to higher edge length variance. This can be interpreted by the lower resolution mesh having longer average edge length, since the size of the registration target is fixed. Another observation is that either a small or a large deformation step size results in a larger edge length variance. The

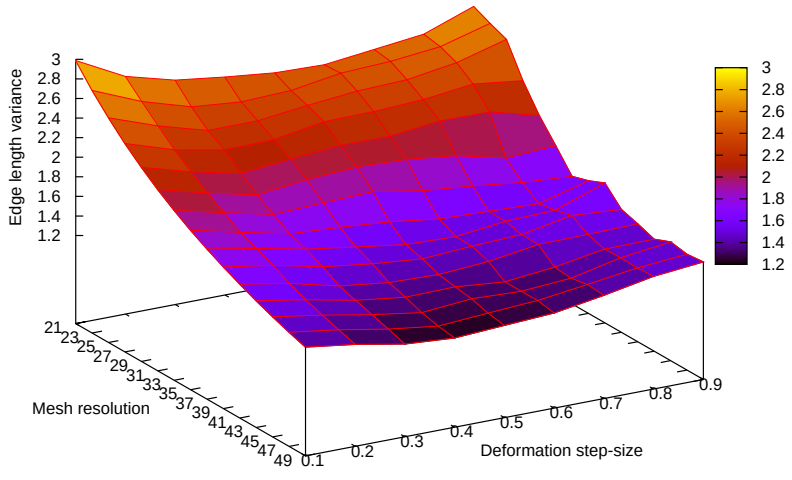


Figure 4.20: Variance of the edge lengths in the deformed mesh. The variance is measured with respect to different mesh resolution and deformation step size.

non-uniformity produced by a small step size can attribute to sub-optimal convergence without enough deformation iterations. The non-uniformity produced by a large step size can be explained by the fast convergence process such that the vertex redistribution function has less impact. Therefore, as shown in the figure, a deformation step size in the middle (from about 0.4 to 0.6) works better.

The algorithm was executed multiple times with increasing values of the uniform vertex distribution weight. The results were evaluated by computing the variance of all edge lengths after deformation. Results show that the variance of edge length is in a descending fashion which suggests a larger weight resulting in more uniformly distributed vertices (Fig. 4.21).

4.3.4 Time Complexity

This experiment is to investigate the time complexity of the proposed algorithm with respect to different mesh resolution. The test data is again the binary cup volume. The execution time was measured on a 2GHz Mac Pro computer.

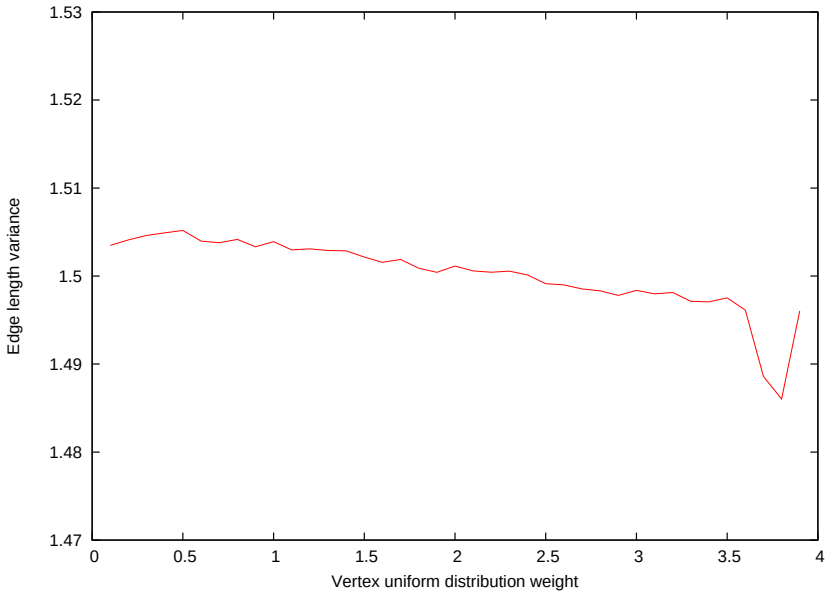


Figure 4.21: Edge length variance with different vertex distribution weights. The variance of edge length is in a descending fashion which suggests a larger weight resulting in more uniformly distributed vertices.

The algorithm was executed in fixed number of iterations (99) for different mesh resolution and deformation step size. At each particular mesh resolution, the average execution time and the variance across difference deformation step size are shown in the form of error bars (red). The dashed green curve is a cubic smoothened running across the averages. The execution time is 36.9 ± 0.2 seconds when $n = 21$, and 261.8 ± 1.5 seconds when $n = 49$.

As shown in the error bars in Fig. 4.22, the deformation step size r does not affect the execution time since the variances at all occasions are very small. The execution time is proportional to the mesh resolution. Within this particular range of r, the time complexity of the algorithm is close to linear.

4.4 Summary

This chapter presented a 3D flipping-free mesh deformation algorithm. The algorithm is based on a quadrilateral mesh which consists of 3 groups of contours orthogonal to each

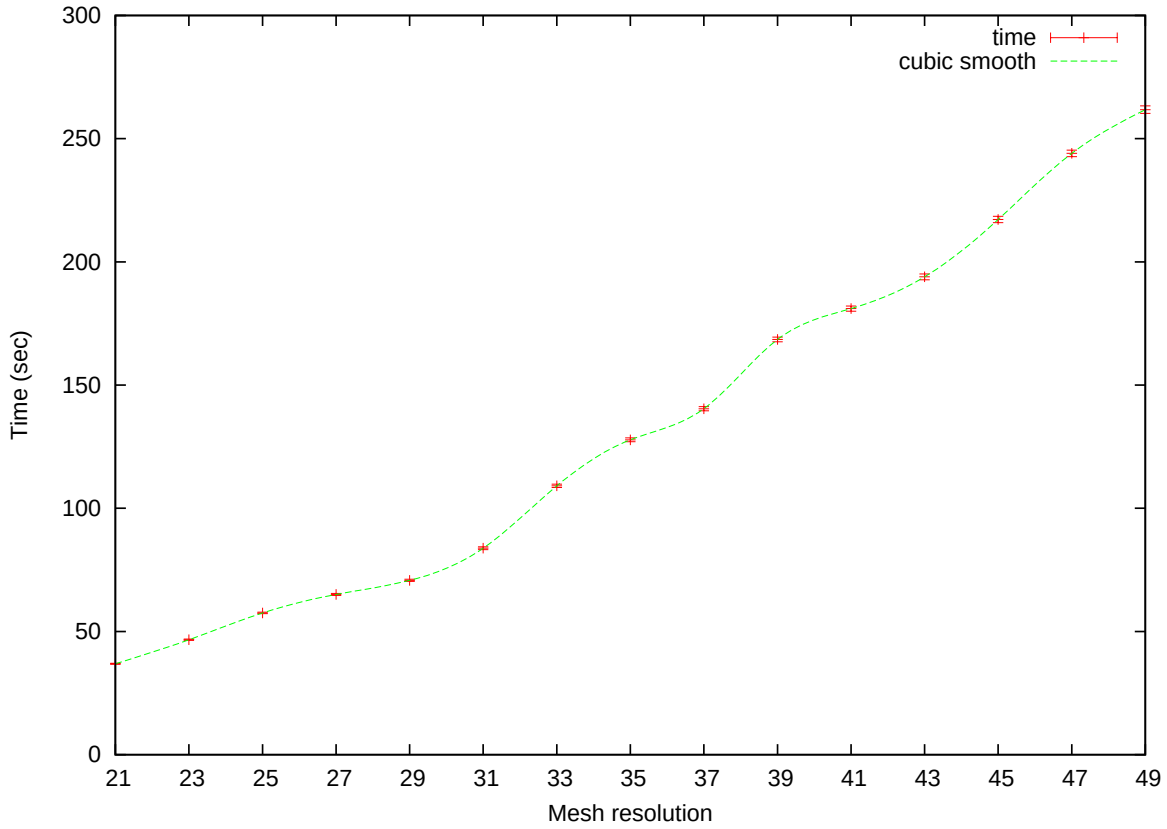


Figure 4.22: Execution time. Each bar represents the average execution time and variance across different deformation step size at that particular mesh resolution. The dashed green curve is a cubic smoothened running across the averages. The time complexity of the algorithm is close to linear with respect to changes of mesh resolution.

other. The advantage of such a mesh over triangular mesh is the natural existence of contours which facilitate solving the flipping problem. In comparison, it is difficult for one to define contours on a triangular mesh to include all the vertices. The advantage of such a mesh over the UV-spherical mesh is that it has no poles, i.e., each vertex has exactly 4 neighbors. In comparison, the UV-spherical mesh has two poles, each has significantly more neighbors than the rest of the vertices do.

The proposed algorithm iteratively deforms the mesh by performing the following stages: (1) searching correspondence for each vertex, (2) detecting and avoiding possible flippings, (3) handling of non-flipping self-intersection and (4) performing mesh deformation based on the Laplacian mesh deformation framework. The correspondences are found by searching along certain directions (e.g., surface normal directions), and some of the mesh vertices may not have correspondences. The Laplacian deformation framework can displace the mesh vertices with correspondence using positional constraints, and deform the rest of vertices using geometric constraints (Laplacian preservation). The estimated correspondences my cause flipping and non-flipping self-intersection problems after deformation. Stage (2) and (3) try to prevent these problems before each deformation iteration.

The experiment is focused on deforming a quadrilateral spherical mesh to make it registered to binary volume images. Experimental results show that the proposed algorithm can avoid the flipping problem effectively. It can successfully converge to concave targets. It is stable and efficient regarding to the change of the parameters. These merits make it possible to be applied to medical volume image segmentation, which is elaborated in the next two chapters.

Chapter 5

Segmentation of Single Object

This chapter presents an algorithm for segmenting single object based on the 3D mesh deformation described in chapter 4. This algorithm begins with user initialization of a single mesh model (Section 5.1)., followed by extraction of features in volume images (Section 5.2). Next, the mesh deformation algorithm described in Chapter 4 is executed to register the mesh to the extracted features. In this case, the correspondence search algorithm is revised to search for corresponding feature points in the volume image (Section 5.3). These correspondences are diffused to smoothen the displacement vector fields and propagate neighboring displacement vectors to vertices without correspondence. The proposed segmentation algorithm is tested extensively regarding to its convergence, accuracy and efficiency, and is used to segment various 3D anatomical structures with significantly different shapes (Section 5.4).

5.1 Mesh Initialization

A specific GUI (Fig. 5.1) was designed to facilitate the initialization process, and to ensure the spherical mesh is placed inside the target object. This is done by positioning the 3D spherical mesh in the 3D view, and check whether the corresponding 2D contours in axial, sagittal and coronal views are completely inside the corresponding 2D regions. The size of the initial sphere should be reasonably large to contain sufficient numbers of pixels for the feature extraction stage. Depending on the the segmentation target, the initial sphere can be initialized completely outside the target (like the one in Chapter 4)

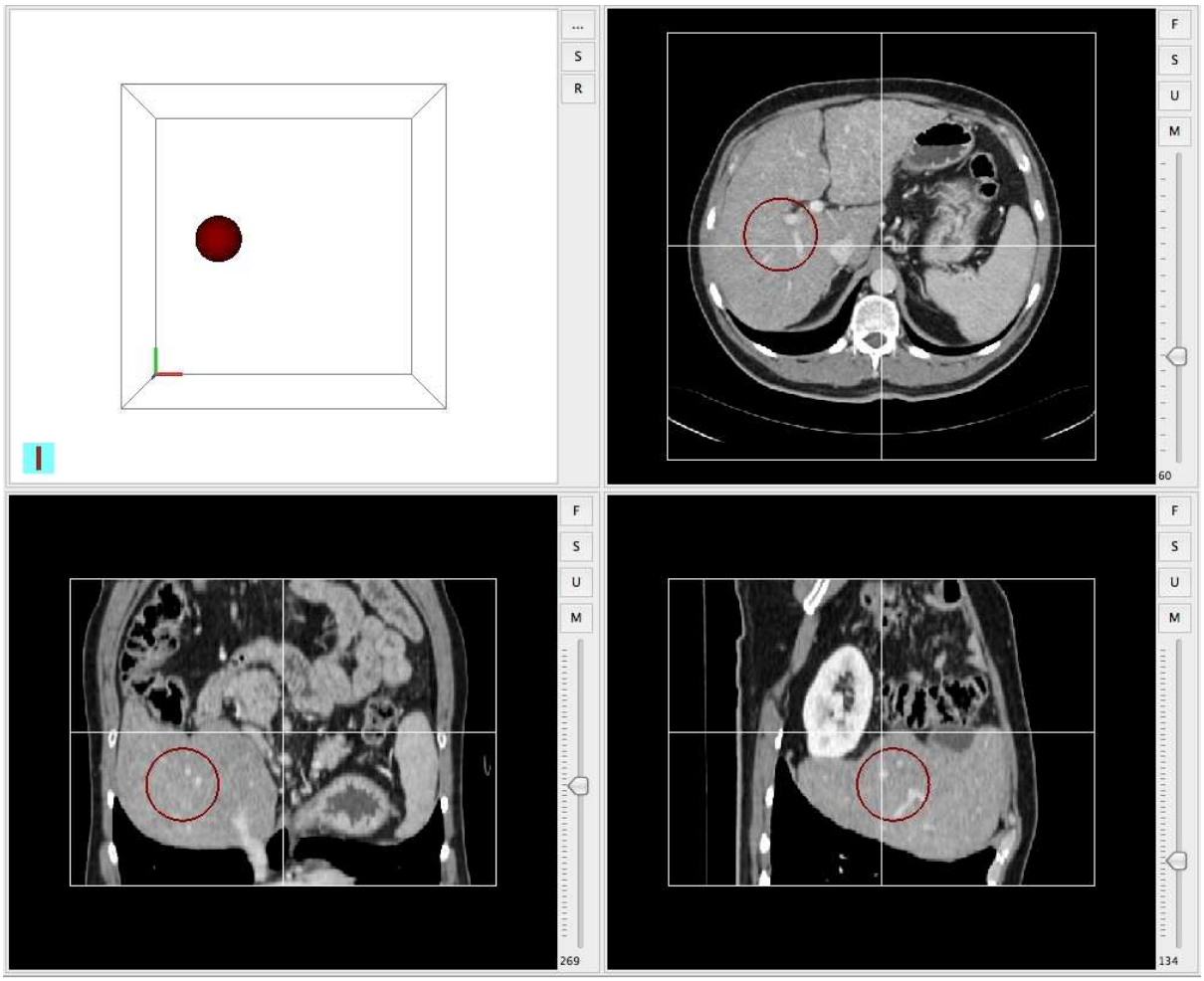


Figure 5.1: Mesh initialization. The spherical mesh (dark red) is positioned in the 3D view, and check whether the corresponding 2D contours (dark red) in axial, sagittal and coronal views are within the corresponding 2D regions.

as well.

5.2 Image Feature Extraction

This stage extracts image features from the volume image for segmentation after image noise is reduced by image smoothing process.

5.2.1 Image Smoothing

A Gaussian filter can be used to reduce image noise, and it is quite efficient. However, it blurs image edges and introduces inaccuracy in the localization of the object's boundary in the filtered images. Instead, anisotropic filtering [PM90] is applied to the input image I to reduce noise.

The purpose of using anisotropic filtering is to reduce the image noise and at the same time to keep the significant features (such as edges) in the image. The anisotropic diffusion equation is as follows:

$$\frac{\partial I}{\partial t} = \mathrm{div}(c\nabla I) = c\Delta I + \nabla c \cdot \nabla I. \tag{5.2.1}$$

where ΔI represents the Laplacian of image I, and c is the conductance term. c determines the diffusion speed of image intensities, and it can be defined in such a way that the diffusion process is stopped at image edges. It is defined as

$$c(\|\nabla I\|) = \frac{1}{1 + (\frac{\|\nabla I\|}{\kappa})^2}, \tag{5.2.2}$$

where κ is a constant that represents the sensitivity to image edges. The larger the image gradient magnitude $\|\nabla I\|$, the smaller the $c(|\nabla I\|)$, the slower is the diffusion speed. Therefore, the diffusion process is slow at the vicinity of edges which usually have large image gradient magnitude.

5.2.2 Intensity Statistics Estimation

For most of the medical images, the intensities of the target organs are inhomogeneous due to various tissue properties of the organ and the noise introduced in the image acquisition stage. Some of the boundaries between neighboring organs may be indistinct if they have similar tissue properties. These problems prevent the proposed algorithm from finding correspondence directly. Instead, some features are extracted for modelling the target organs.

The proposed algorithm can work with any feature including, but not restricted to, intensity, gradient, edge, texture, etc. The mixture consisting of m Gaussians (i.e., parametrically) is chosen to model the intensities because of its compactness and robustness.

The Gaussian Mixture Model (GMM) is written as follows.

$$g(x) = \sum_{i=1}^{m} a_i f_i(x), \tag{5.2.3}$$

where x is the voxel intensity, a_i are coefficients, such that $\sum_i a_i = 1$, and $f_i(x)$ are Gaussian distributions with parameters (μ_i, σ_i). The number of Gaussians is determined by the input images and the target organs. Parameters $\boldsymbol{\theta}_i = (a_i, \mu_i, \sigma_i)$ are estimated by Expectation Maximization (EM) [DLR77].

The number of Gaussians is determined automatically by an adaptive binning algorithm [LL03]. Given the estimated class radius and class separation, the adaptive binning algorithm tries to group a new intensity value into its closest cluster if they are close enough (less than radius R) to each other. The algorithm will create a new cluster if the intensity value are too far (greater than separation distance S) from any existing cluster. The unclustered intensity values will be re-considered in subsequent iterations. The adaptive binning algorithm guarantees that each cluster has a maximum radius and different clusters can be separated by a certain distance. It automatically determines the optimal number of clusters (Algorithm 2). This property is highly desirable since the number of clusters are usually hard to determine based on the user initialization.

Algorithm 2 Adaptive binning.

Repeat:
for each intensity value p **do**
  Find the nearest cluster k to p,
  if no cluster is found or distance $d_{kp} \geq S$ **then**
    create a new cluster with p;
  else if $d_{kp} \leq R$ **then**
    Add p to cluster k
  end if
end for

for each cluster i **do**
  if cluster i has at least N_m intensity values **then**
    update centroid c_i of cluster i;
  else
    remove cluster i.
  end if
end for

Given observed of voxel intensities $x_i (i = 1, \ldots, n)$), introducing an new variable z_{ij} (unobservable data), the EM algorithm determines the maximum likelihood estimate of the parameter a_i, μ_i, and σ_i by iterating the E-step and the M-step:

- E-step: At iteration k, compute the expected value of z_{ij} based on the current estimation of the parameters (or initialization). For $i = 1, \ldots, m$, $j = 1, \ldots, n$,

$$z_{ij}^{(k)} = \frac{a_i^{(k)} f_i(x_j; \boldsymbol{\theta}_i^{(k)})}{g(x_j; \boldsymbol{\theta}^{(k)})} \tag{5.2.4}$$

- M-step: Re-estimate the parameter vector by maximizing the likelihood estimate.

$$a_i^{(k+1)} = \frac{1}{n} \sum_{j=1}^{n} z_{ij}^{(k)} \tag{5.2.5}$$

$$\mu_i^{(k+1)} = \frac{\sum_{j=1}^{n} z_{ij}^{(k)} x_i}{\sum_{j=1}^{n} z_{ij}^{(k)}} \tag{5.2.6}$$

$$\sigma_i^{(k+1)} = \frac{\sum_{j=1}^{n} z_{ij}^{(k)} (x_i - \mu_i^{(k+1)})^2}{\sum_{j=1}^{n} z_{ij}^{(k)}} \tag{5.2.7}$$

The initialization ($\boldsymbol{\theta}^{(0)}$) of the EM algorithm is given by the results of Adaptive binning algorithm. An example of GMM modeling of the input intensity values is shown in Fig. 5.8(b). Based on the estimated intensity statistics, the correspondence search stage can be carried out for each mesh vertex.

5.3 Correspondence Search

This stage estimates the correspondence between the model M and the target T. For each vertex $\mathbf{u}_i$ on M, the algorithm searches along the projection line $P(\mathbf{u}_i)$ for a possible corresponding point $\mathbf{v}_i$ on T within a search range R_s. The direction of $P(\mathbf{u}_i)$ is defined as the mesh surface normal at $\mathbf{u}_i$. R_s is pre-defined according to the size of the target organ. The larger the target organ, the larger the R_s. Since the algorithm is expected to converge at the vicinity of the surface of T, R_s can be reduced during iterations as the deformed mesh is close to the target surface. The point $\mathbf{v}_i$ is the intersection of $P(\mathbf{u}_i)$ and the face of a voxel in T that contains the nearest feature point. Nearest feature point

is chosen because it is more likely to be on the surface of T than feature points further away. Each $\mathbf{v}_i$ serves as a target location for $\mathbf{u}_i$. In general, $P(\mathbf{u}_i)$ may be defined along other appropriate directions.

Assuming that the intensities of the foreground voxels are described by g and while background voxels cannot, the problem of finding correspondence $\mathbf{v}_i$ for $\mathbf{u}_i$ along $P(\mathbf{u}_i)$ is formulated as finding the first voxel that its subsequent voxels along P cannot be described by g consistently. Therefore, the equation describing this can be formulated as finding the minimum j such that

$$\sum_{i=j,j<R}^{j+N} h(\mathbf{u}_i) = 0, \tag{5.3.1}$$

$$h(\mathbf{u}_i) = \begin{cases} 0 & g(x_i) < \Gamma \\ 1 & \text{otherwise} \end{cases}, \tag{5.3.2}$$

where x_i is the intensity of the voxel containing $\mathbf{u}_i$ and Γ is a pre-defined threshold.

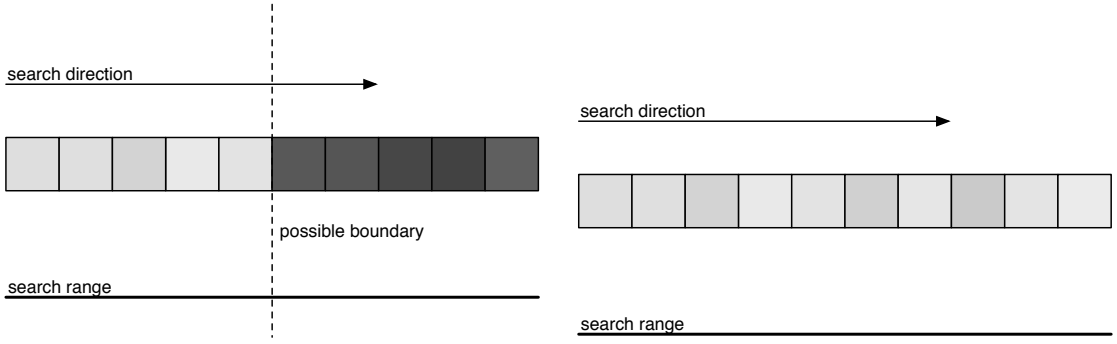
This equation can be illustrated by Fig. 5.2, where the algorithm searches for correspondence for each vertex (red dot) on the mesh (red contour) in the image (Fig. 5.2(a)). As can be seen in Fig. 5.2(b), the $\mathbf{u}_j$ that satisfies Eqn. (5.3.2) is likely on the boundary of the target object, since $(N+1)$ consecutive voxels starting from position $(j+1)$ along the search direction $P(\mathbf{u}_i)$ all have low probabilities of belonging to the foreground. Fig. 5.2(c) shows that the search algorithm cannot find any corresponding boundary voxel within the search range. A larger N can be used for images which are more noisy, but may have the chance to miss the target boundary if the neighboring organ has a thin structure as shown in Fig. 5.2(c). In our current implementation, $N = 3$. If such a j cannot be found within the search range R due to either indistinct object boundary or image noise, $\mathbf{u}_i$ is labeled as a *solitary* vertex.

In this way, the algorithm can search possible correspondence for each vertex over a long range, which makes it more noise resilient.

After correspondence search, most mesh vertices have corresponding points, but solitary vertices do not have any. Moreover, some displacement vectors may cross due to erroneous estimation caused by image noise.

To solve the problems mentioned above, diffusion of correspondence is proposed to propagate displacement vectors to vertices that have no correspondence. Diffusion of

(a)



(b) (c)

Figure 5.2: Correspondence search. (a) For each vertex (red dot) on the mesh (red contour), the algorithm searches for its correspondence within R_s. (b,c) zoom in view of the voxels along the yellow and green search direction lines. (b) Found correspondence; (c) Not found.

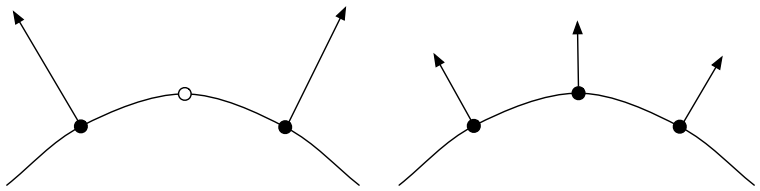


Figure 5.3: Diffusion of correspondence. (a) The solitary vertex (empty circle) has no correspondence before diffusion. (b) All vertices have correspondence after diffusion from non-solitary vertices (solid circle).

correspondence over the mesh surfaces can be formulated as a problem similar to gradient vector flow (GVF) [XP98] over a 2D image space. For efficiency, a simple vector average is used. For each vertex that has a correspondence, recompute its estimated correspondence as:

$$\tilde{\mathbf{v}}'_i = \mathbf{v}_i + \frac{1}{N+1}\left[\mathbf{v}'_i - \mathbf{v}_i + \sum_{v_j \in \mathcal{N}(\mathbf{v}_i)} (\mathbf{v}'_j - \mathbf{v}_j)\right], \tag{5.3.3}$$

where $\mathbf{v}_j$ are neighboring vertices of $\mathbf{v}_i$ which have correspondence, and N is the number of such $\mathbf{v}_j$. Fig. 5.3 illustrates that diffusion helps to create correspondence for the solitary vertex (empty circle), which has no correspondence previously. Diffusion of correspondence also helps to smoothen the displacement vectors such that they have less chance to introduce flipping during deformation. This is because neighboring displacement vectors tend to point to a locally average direction rather than cross with each other..

Once the problem of correspondence is solved, the rest of the segmentation algorithm is just the same as the flipping-free deformation algorithm described in Chapter 4.

5.4 Experiments and Discussions

To verify the strength of the segmentation algorithm in terms of convergence (Section 5.4.1), accuracy and efficiency (Section 5.4.2), comprehensive tests were conducted. The experiments were mainly carried out on globular objects such as liver and spleen. Applicability of the single-object segmentation algorithm to tubular objects is also discussed (Section 5.4.3). Results of the segmentation algorithm were compared with those

of other algorithms. These algorithms includes GVF snake [XP98], level set method as implemented in ITK-SNAP (`www.itksnap.org`) and graph cut [RKB04] in terms of noise resilience, accuracy and efficiency, since they are widely used in the medical image segmentation field and represent the state of the art. Ideally, it would be better to compare segmentation performance with more advanced variants of the level set method. However, these variants are not readily available for comparison. Implementing them is beyond the scope of this thesis. Only 2D version of the graph cut algorithm was tested, because the 3D version can not fit in the physical memory constraints (4GB) of a 32 bit computer. To further verify its capability to handle different target object, the segmentation algorithm was used to extract the abdominal walls (Section 5.4.4). All experiments were performed on an Intel Core 2 Duo 2.33 GHz computer with 4GB memory.

5.4.1 Convergence

This experiment was designed to test the convergence of the segmentation algorithm in handling real medical images. The single-object segmentation algorithm was performed to segment the liver from on one abdominal CT scan for 60 iterations. The average surface distances between the deformed mesh and the surface of the ground truth data were plotted during these iterations.

Fig. 5.4 shows that the distance decreases rapidly as the algorithm iterates, and the distance drops close to 0 when the algorithm finishes. This suggests that the segmentation algorithm can converge when performing segmentation in real medical volume images.

5.4.2 Accuracy and Efficiency

This experiment was designed to test the accuracy and efficiency of the single-object segmentation algorithm. The test data contains 8 abdominal CT scans with slice thickness varied from 1mm to 3mm. They were obtained from MICCAI liver segmentation data set (`http://www.silver07.org`). The target objects included in this experiment are liver and kidney.

The single-object segmentation algorithm and level set were applied to the whole CT volume and initialized with spheres of the same size at the same location (Fig. 5.5(a)).

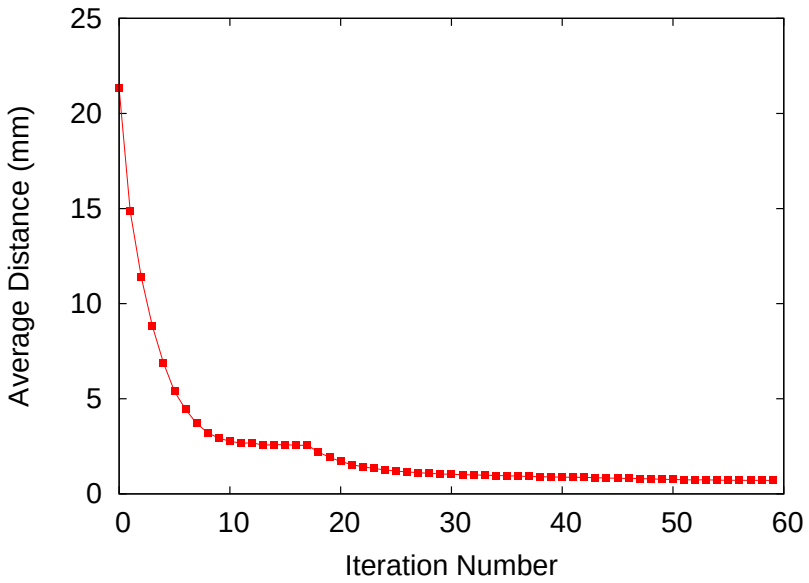


Figure 5.4: Convergence curve. The average estimated distance from model vertices to target points decreases as the algorithm iterates.

GVF snake was applied on a single CT slice because of its 2D nature. It was initialized with a 2D cross-section of the sphere in the slice. Graph cut was applied to a single CT slice because its 3D implementation ran out of memory given the input volume images. It was initialized by manual markups representing foreground and background pixels (Fig. 5.5(i)). The level set algorithm was stopped immediately by the user when the liver regions were fully segmented.

Segmentation was first performed on liver. Test results of the single-object segmentation algorithm, the level set algorithm and the graph cut algorithm were evaluated both qualitatively and quantitatively. Average surface distance and volume overlap error were measured between the segmented results and the ground truth to test segmentation accuracy. Segmentation time of the entire volume and per slice time were measured to test the efficiency of the algorithms.

Figure 5.5(f, g) show two views of the segmented liver which has a complex shape using the single-object segmentation algorithm.

As shown in Fig. 5.5(h) and (d), indistinct boundaries caused severe leakage problem for both level set and graph cut. The noise problem prevented the GVF snake contour

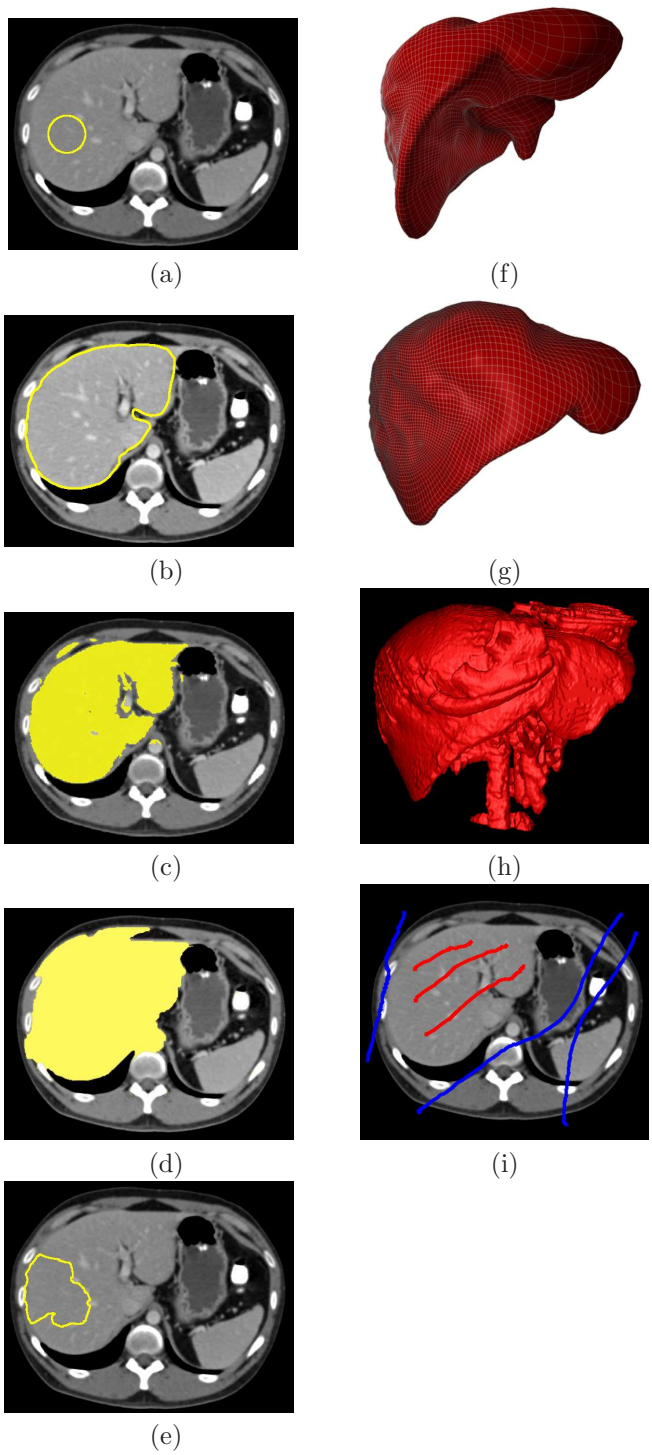


Figure 5.5: Comparison of segmentation algorithms. (a) Initialization for the single-object segmentation algorithm, level set and GVF snake. (b, f, g) Segmentation results of the single-object segmentation algorithm, (c, h) level set, (d) graph cut, (e) GVF snake. (i) Initialization for graph cut: (red) foreground and (blue) background markups. Best viewed in color.

Table 5.1: Comparison of level set algorithm (LS), graph cut (GC) and the single-object segmentation algorithm. V: ground truth volume. D: average symmetric distance. VO: volume (area for graph cut) overlap. K: number of iterations. T: execution time. T′: execution time per slice.

	V (ml)	Algorithm	D (mm)	VO	K	T (sec)	T′ (sec)
liver	1754.37± 387.57	LS	10.28±3.25	(72.98± 8.12)%	1051± 138	476.29± 116.27	3.58± 2.46
		GC	10.49	81.50%	23	17.30	17.30
		proposed	**1.99± 0.41**	**(88.7± 3.36)%**	43± 6	54.47± 7.38	**0.42± 0.26**
spleen	367.89± 196.43	LS	4.43± 3.59	(76.10± 15.06)%	519± 75	110.38± 43.60	0.71± 0.27
		GC	**0.82**	**96.00%**	13	11.56	11.56
		proposed	1.00± 0.27	(88.87± 2.98)%	42± 11	17.84± 4.50	**0.14± 0.11**
left brachi-ocephalic vein	6.05	LS	0.38	81.20%	387	53.04	0.53
		GC	1.36	81.20%	14	10.99	10.99
		proposed	**0.35**	**83.00%**	72	26.01	**0.26**

from converging to the target boundary (Fig. 5.5(e)). In comparison, our algorithm has less leakage thanks to the geometric constraints. Segmentation accuracy in terms of average symmetric distance and volume overlap was computed for level set and our algorithm (Table 5.1). Our algorithm achieved better accuracy with shorter average symmetric distance and larger volume overlap. As shown in Table 5.1, the segmented livers from level set had an average 72.98±8.12% volume overlap with and 10.28±3.25mm average surface distance to their respective ground truth. In contrast, those of ours have $89.97 \pm 1.52\%$ and 1.69 ± 0.40mm on average. The much lower variance achieved by our algorithm also indicates it is more stable. Graph cut has a area overlap of 81.50% and an average surface distance of 10.49mm on one slice. With regard to efficiency, the level set algorithm took 1051 iterations in 476.29 seconds on average to segment the whole liver. In contrast, our algorithm took only 43 iterations in 54.47 seconds on average to segment the liver. Compared to graph cut, our algorithm also took much less time on per slice basis. Note that the level set algorithm implemented in ITK-SNAP automatically used two threads for computation in our PC, whereas graph cut and our algorithm used one

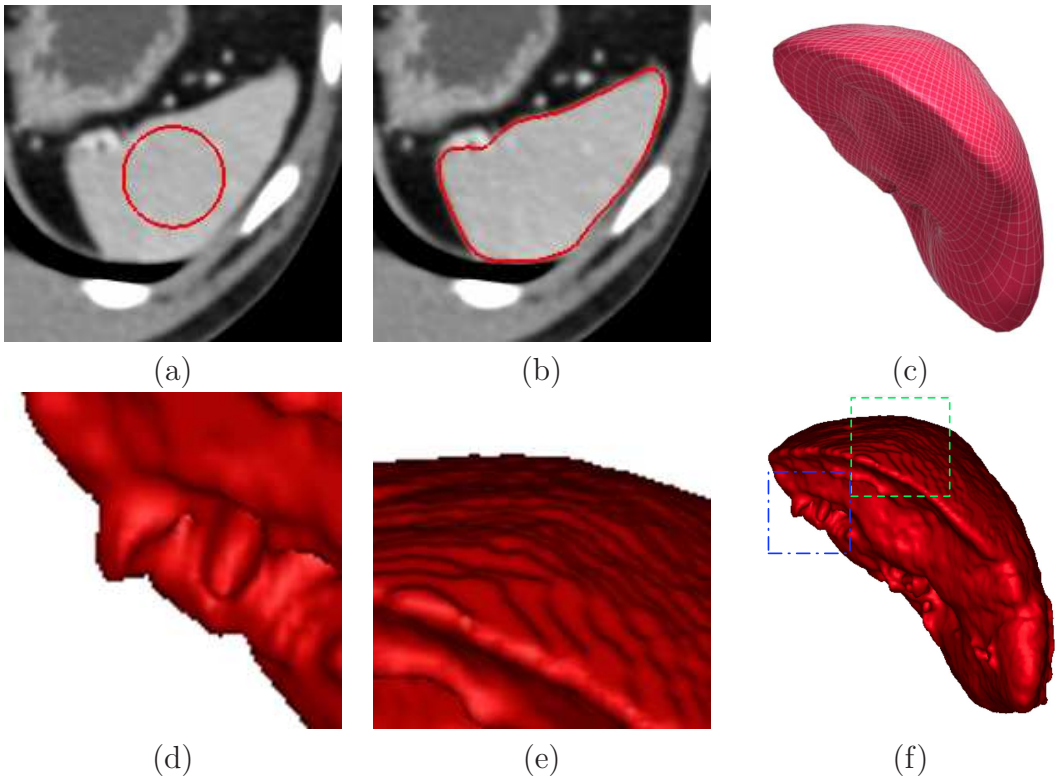


Figure 5.6: Segmentation of spleen. (a) Initialization. (b, c) Segmentation results of the single-object segmentation algorithm in 2D slice view and 3D view. (d, e) Zoom-in views of artifacts in segmentation results of level set (f).

thread only.

Segmentation was also performed on another abdominal organ, i.e., spleen. As shown in Table 5.1, the segmented spleens of the level set algorithm have an average 76.10±15.06 volume overlap with and 4.43±3.59mm average surface distance to their respective ground truth. In contrast, those of ours are $88.87 \pm 2.98\%$ and 1.00 ± 0.27mm on average. Graph cut has 96.00% of area overlap which is due to the clear boundary of the spleen on the input slice. Results produced by level set (Fig. 5.6(f)) is less smooth than those produced by our algorithm (Fig. 5.6(c) top) due to voxelization, and they have leakage artifacts. Our algorithm performed more accurately than level set in segmenting the spleen. Per slice execution time was also computed across different algorithms. The results shown in Table 5.1 once again suggest that the single-object segmentation algorithm is much faster than the level set algorithm and graph cut.

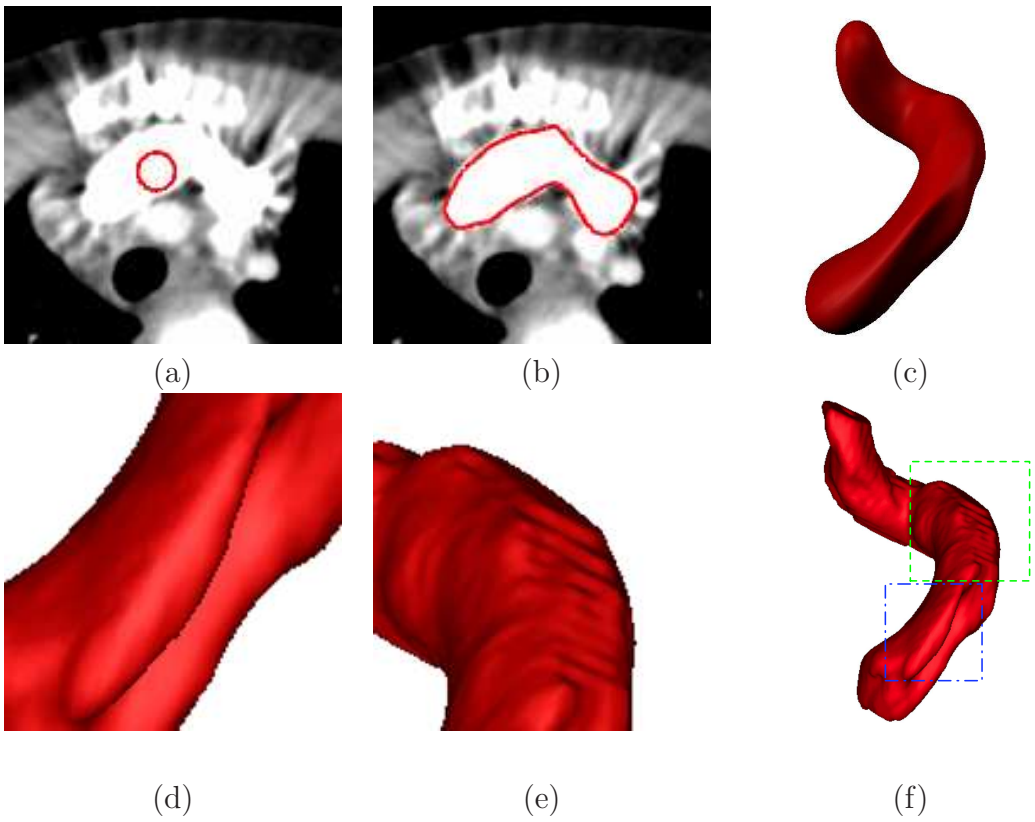
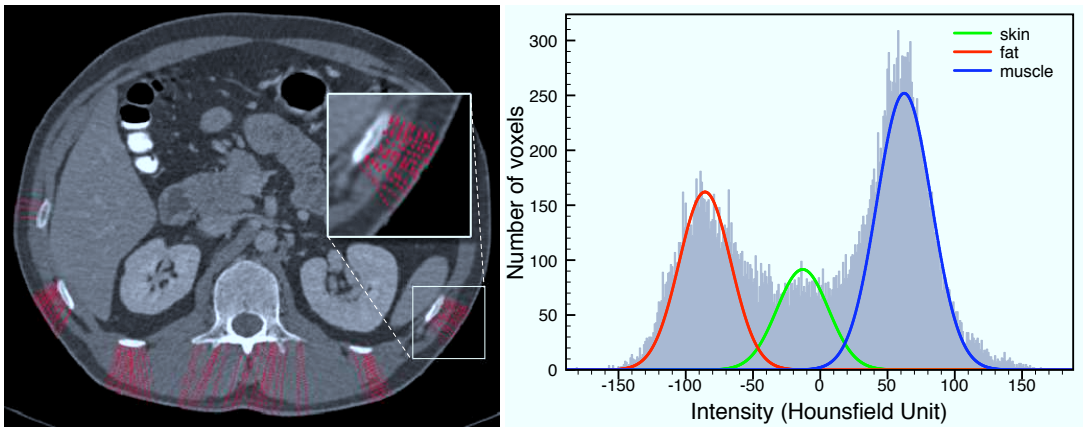


Figure 5.7: Segmentation of left brachiocephalic vein. (a) Initialization. (b, c) Segmentation results of the single-object segmentation algorithm in 2D slice view and 3D view. (d, e) Zoom-in views of artifacts in segmentation results of level set (f).



(a) (b)

Figure 5.8: Feature extraction of the abdominal wall. Feature extraction of the abdominal wall. (a) Extract feature points (red) in the abdominal wall for (b) building intensity GMM.

5.4.3 Segmentation of Tubular Organ

Segmentation of left brachiocephalic vein which has a tubular shape from real medical images was performed. The initialization was inside the blood vessel (Fig. 5.7(a)). The left brachiocephalic vein was successfully segmented (Fig. 5.6(b, c)) thanks to the uniform vertex distribution constraint, which facilitated the large shape change. Results were compared with those obtained by the level set methods and graph cut (Table 5.1). Segmentation accuracy using our algorithm is only slightly better than that of the level set algorithm and graph cut because this left brachiocephalic vein sample has a very small volume (6 cm^3) and a segmentation error of a single voxel will result in large error in volume overlap. Again, our algorithm is more efficient.

5.4.4 Removal of Abdominal Wall

To further test the capability of the single-object segmentation algorithm to handle different shapes, the single-object segmentation algorithm was also applied to segment abdominal walls in abdominal CT images [DLV09]. By removing the wall in the 3D volume, the organs in the abdomen can be exposed and visualized directly. It also makes segmentation more accurate and efficient by reducing the search space of any segmentation

algorithm.

The test data include abdominal CT volume images of 1mm to 3mm thickness. The initial spherical models were manually placed inside the abdominal walls. The feature extraction stage estimates the intensity distribution of abdominal wall voxels. Since voxels between the skin surface and the bone structure clearly belong to the abdominal wall, they were used to build an intensity distribution to approximate that of the abdominal wall. Therefore, the feature extraction stage contains two steps: (1) identify the skin surface and the bones , and (2) build the intensity distributions of the body wall voxels.

Identification of the skin surface is straight-forward by using a contour tracing algorithm [SA85]. Since accurate segmentation of the bone structure is not necessary, identification of bone voxels can simply be performed by applying thresholding with a high threshold and then by extracting the largest connected component.

To extract the voxels in between, a ray is projected along the inward surface normal direction until it meets a bone voxel. The voxels along the ray definitely belong to the body wall and are extracted (Fig. 5.8(a)). If a ray cannot find any bone voxel within certain distance, all the voxels along the ray are discarded. The extracted voxels are used to build the GMM model.

Sample results of the extracted inner boundaries of the abdominal walls for two input CT volumes are shown in Fig. 5.9(b,e) and Fig 5.9(c,f) respectively, in both 3D view and 2D axial view.

The voxels belong to the abdominal wall can be easily removed after the surface of the abdominal wall is extracted. The removal of the abdominal wall can be applied to visualization of the organs.

Two examples for the volume rendering of CT images are shown in Fig. 5.10. In Fig. 5.10(a) and (c), the opacity transfer functions were adjusted so that the abdominal walls appear transparent and some of the organs can be observed. Since the opacity transfer functions were applied globally across the whole input volume, other inner organs which have similar voxel intensities such as colons and blood vessels also became transparent and cannot be observed. In contrast, after removing the abdominal walls by the single-object segmentation algorithm, they can be clearly visualized as shown in Fig. 5.10(b) and (d).

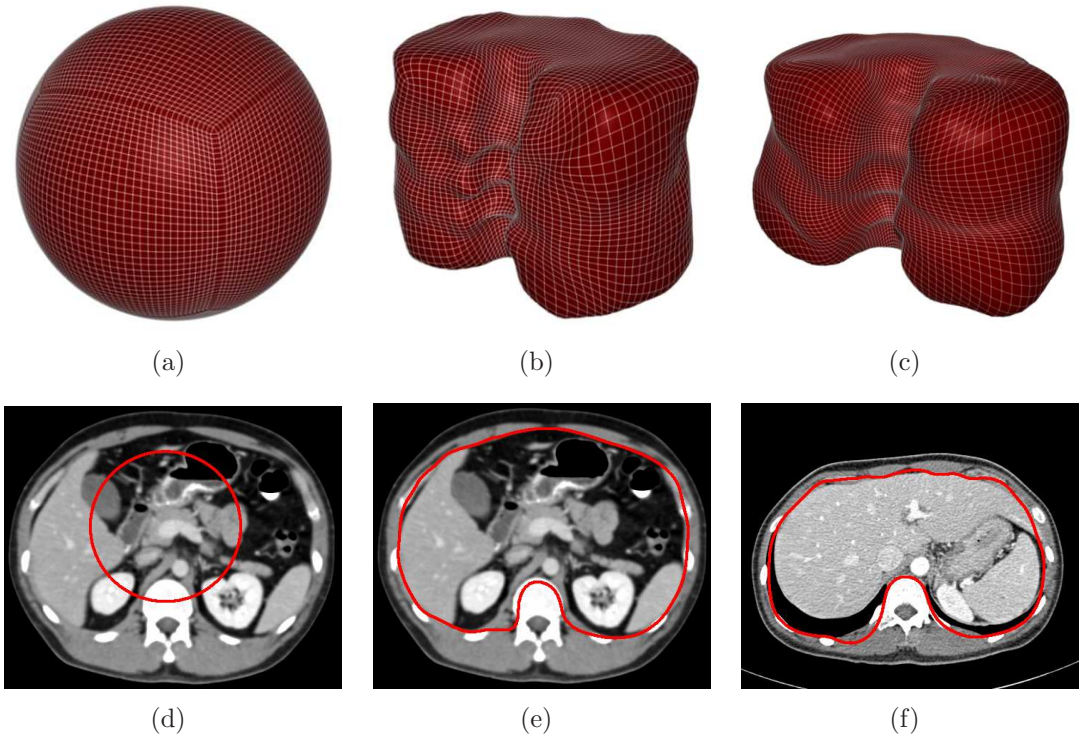
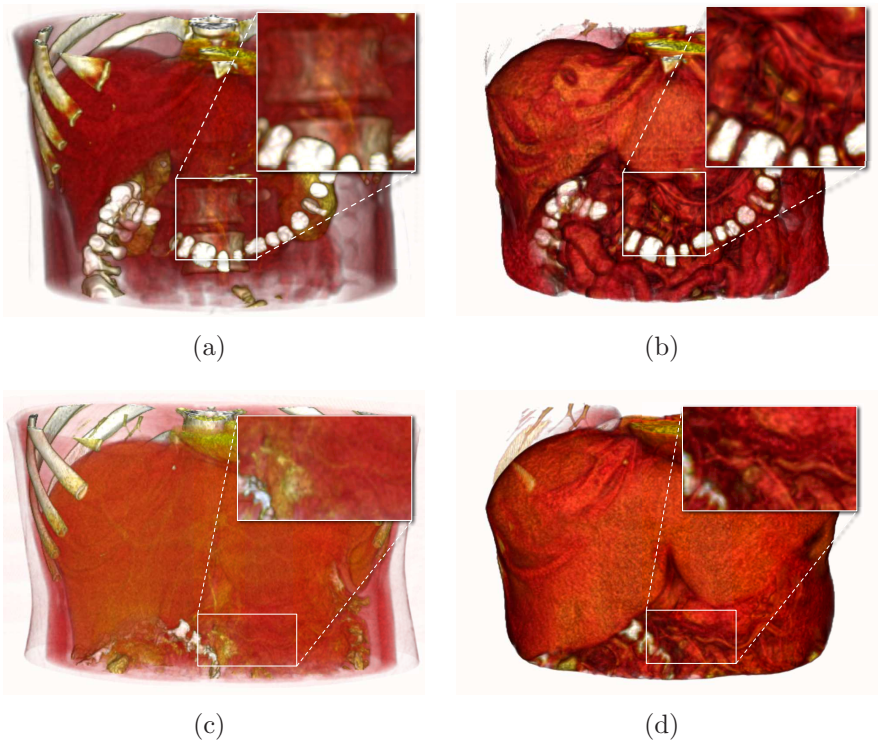


Figure 5.9: Extraction of abdominal wall. (a) Initial 3D quadrilateral mesh and (d) its 2D view in one axial slice. (b,c) Extracted 3D surfaces and (e,f) their respective 2D axial views.



(a) (b)

(c) (d)

Figure 5.10: Volume rendering. (a,c) Volume rendering of two CT volumes by setting the abdominal wall to transparent. (b,d) Volume rendering of organs after removing the abdominal wall. Some of the organs such as colons and blood vessels that cannot be visualized in the former can be clearly visualized in the latter. Best viewed in color.

5.5 Summary

This chapter presented a single-object segmentation algorithm to solve the 3D soft organ segmentation in medical volume images. The proposed segmentation algorithm is based on the 3D flipping-free mesh deformation algorithm discussed in the previous chapter. The user is required to initialize the segmentation algorithm by placing a quadrilateral spherical mesh inside the target organ. The intensities of voxels inside the initial sphere are used to model the intensity distribution of the target organ. The distribution model is used for the segmentation algorithm to determine whether a voxel belongs to the foreground or the background. The segmentation algorithm searches for a corresponding point for each vertex by examining voxel intensities along the surface normal direction. In some low contrast regions, the algorithm may not be able to find correspondence for some vertices. After all the vertices have been examined, the flipping-free mesh deformation algorithm can be applied.

The single-object segmentation algorithm was tested on segmenting various soft organs inside the abdomen. Test on segmenting livers show that the proposed algorithm is very efficient and noise resilient. Compared to the 2D snake segmentation algorithm, the single-object segmentation algorithm is affected less by the image noise because it looks for correspondence over a long range. Compared to the level set algorithm, it is shows less leakage problem thanks to its local geometric constraints. It is faster because it takes a explicit mesh representation. Compared to the graph cut methods, it is much faster and uses less memory thanks to the explicit representation as well. The algorithm was used to segment tubular objects such as blood vessels, suggesting its capability to handle organs with very different shapes. This property was further verified as the algorithm was also used to extract and remove the abdominal wall, which has another different shape. The removal of the abdominal wall helps to visualize the internal soft organs more clearly.

Although the single-object segmentation algorithm demonstrated less leakage problem than the level set method due to its local shape constraint, leakage may still happen at a low contrast region. As a result, if the deformation is not stopped at a proper time, some part of the mesh surface may bleed into the voxels that belong to another organ. For example, this may happen during liver segmentation between the boundaries of liver and kidney, liver and heart, liver and gall bladder, liver and abdominal wall, etc. This

problem can be illustrated by Fig. 6.5 in the next chapter.

Chapter 6

Segmentation of Multiple Objects

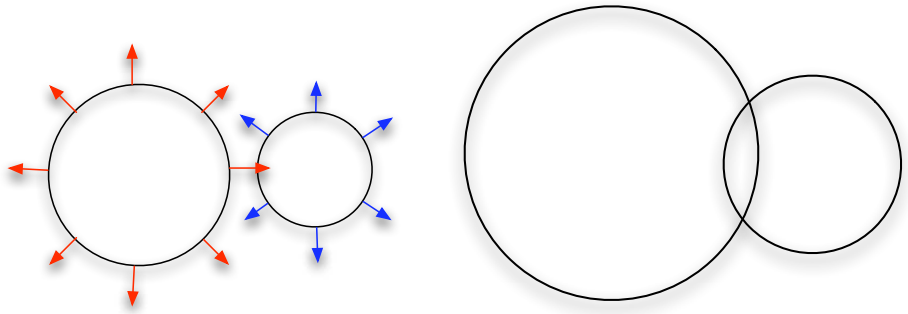
This chapter extends the segmentation algorithm presented in Chapter 5 to simultaneous segmentation of multiple soft organs. The main idea is that the segmentation of each organ is performed within its own bounding region. These organs' bounding regions have no intersection with each other, and are updated during the segmentation process. Each bounding region serves as a constraint for segmenting neighboring organs. In this way, organs are segmented simultaneously and they have no intersections with each other, which is a highly desired property.

The multiple organ segmentation begins with a manual initialization process (Section 6.1). For each target organ, feature extraction is performed as in Chapter 5. This is followed by an iterative process that (1) computes the bounding region of each organ based on the organ-specific statistical model and the current position of the respective mesh surfaces (Section 6.2), and (2) performs single-object segmentation within each bounding region (Section 6.3) until convergence. The segmentation algorithm was evaluated both qualitatively and quantitatively, which shows that it can improve the segmentation accuracy of an individual organ, and prevent neighboring organs from intersecting each other (Section 6.4).

The overview of the extended multiple-object segmentation algorithm is as below.

Algorithm 3 Multiple-object segmentation.

Manual input: put a spherical quadrilateral mesh inside each target object.
Feature extraction: compute image statistics within the initialized sphere.
Repeat until convergence:
  Compute the bounding region for each deformable model.
  Perform single-object segmentation within the bounding region.



(a) (b)

Figure 6.1: Inter-object collision. Collision happens (b) when two neighboring meshes deform according to their vertex displacement vectors (red and blue arrows respectively) (a).

6.1 Initialization of Mesh Models

Just like initialization for single-object segmentation (Section 5.1), initialization of the multiple-object segmentation algorithm involves manually placing a spherical mesh inside each target organ. The voxels inside each initial sphere are used to estimate voxel intensity statistics and construct bounding region (Section 6.2) for each target organ.

6.2 Bounding Region Computation

The single-object segmentation algorithm, if applied directly to segment multiple target organs simultaneously, will produce inter-object intersections (or collisions) as shown in Fig. 6.1. The two meshes have a collision(Fig. 6.1(b)) if they are deformed according to the displacement vectors (red and blue arrows in Fig. 6.1(a)).

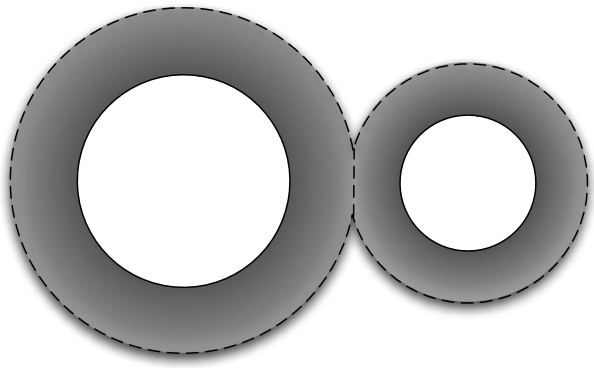


Figure 6.2: Computation of deformation bounding regions. The bounding regions are computed using distance transform. The darker the pixel, the further the point to the mesh.

To prevent this, each mesh model is restricted to deform within its own bounding region, and each bounding region has no intersection with each other.

The Bounding region B_i of M_i can be computed in several ways. B_i can be computed using distance transform as shown in Fig. 6.2, where the intensities of pixels inside the region correspond to the distances to the meshes.

To compute B_i mesh M_i, an binary volume which corresponds to the inside part of the mesh is constructed. From the binary volume, a Euclidean distance transform can be computed so that each voxel inside the volume image has a scalar value representing its approximate Euclidean distance to the original mesh surface. A pre-defined threshold is be set as the width of the bounding region so that the mesh vertices are only allowed to be displaced inside the bounding region.

However, the location of the computed bounding region relies on the Euclidean position of the mesh surface, i.e., it is sensitive to the initialization of the mesh. Therefore, the applicability of the bounding regions computed using distance transform is limited. It is inherently unsuitable for segmentation of body soft organs that have complex shapes.

Instead, we proposed to compute the bounding region based on geodesic distance. The

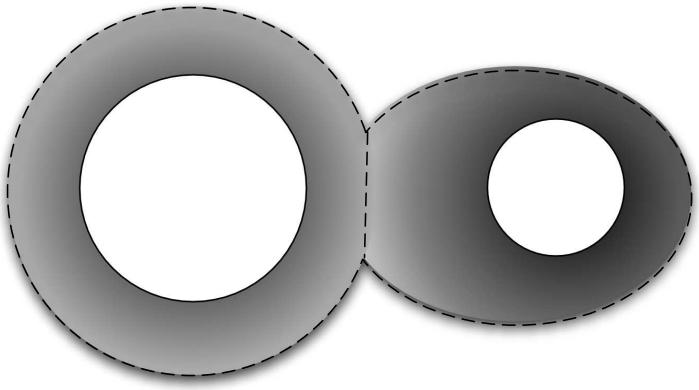


Figure 6.3: Computation of deformation bounding regions. The bounding regions are computed using the fast marching method.

distance metric takes into account the intensities of image voxels. To efficiently compute the bounding region, the fast marching method [Set99b] is used. It incorporates image characteristics to compute a soft segmentation based on front propagation starting from the initial mesh. The propagation of the front is described by

$$|\nabla T_i| S_i = 1 \tag{6.2.1}$$

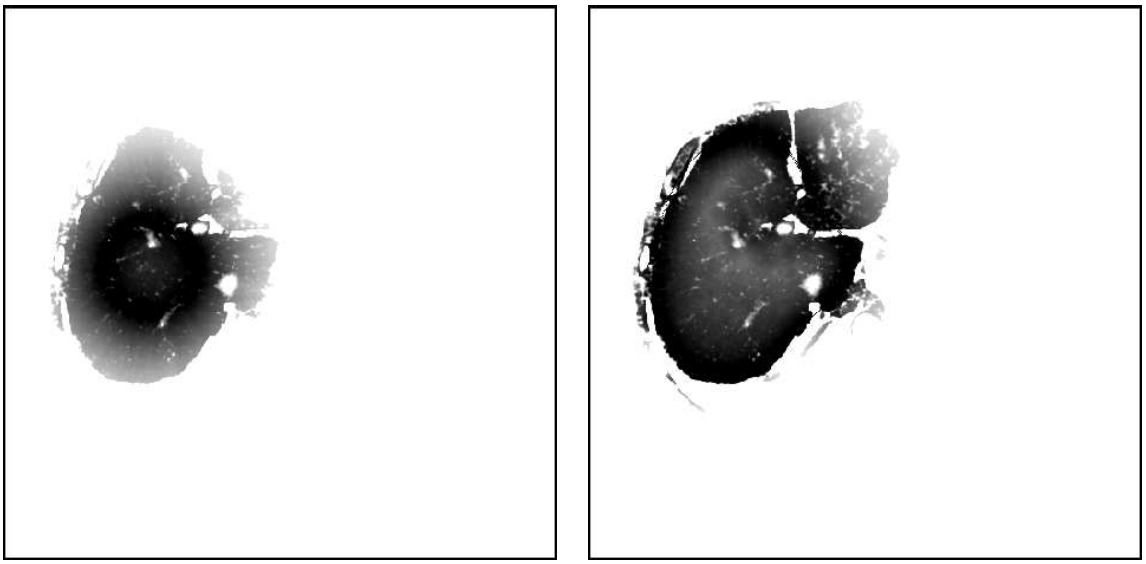
where T_i is the arrival times of the front of M_i, and S_i is the corresponding speed functions. If S_i is constant (e.g., $S_i = 1$), i.e., the front propagates with constant speed across the whole image, T_i corresponds to the Euclidean distance transform of the original binary volume. In the case of soft segmentation, S_i should be large inside the region of the target organ, and small or zero outside. Therefore, the speed functions are proportional to the posterior probabilities,

$$S_i(\mathbf{x}) \propto P(F_i|I(\mathbf{x})), \tag{6.2.2}$$

where

$$P(F_i|I(\mathbf{x})) = \frac{P(I(\mathbf{x})|F_i)P(F_i)}{P(I(\mathbf{x}))}, \tag{6.2.3}$$

$P(I(\mathbf{x}))$ is the normalization constant. $P(F_i)$ can be determined *a priori* according to the objects of interest in the volume image and the type of the image. Given S_i, solving T_i produces a volume image in which each voxel records the arrival time of the corresponding



(a) (b)

Figure 6.4: Bounding Regions (dark pixels) generated by fast marching during liver segmentation. Bounding Regions (a) and (b) are generated at different iterations. The darker the intensity, the closer the pixel to the current mesh surface.

front. The initial front is generated using intersection between the initial spherical mesh and the input volume image.

Each bounding region of the individual object is updated during segmentation process. It has the shape of a sphere at the beginning, and gradually develops into the shape of the target object. Examples of computed bounding regions are shown in Fig. 6.4.

Since there are n target organs, there will be n propagating fronts, and n corresponding bounding regions. Each bounding region has an ID $i(i = 0, \ldots, n)$. $i = 0$ means background. Each voxel $\mathbf{x}$ will be assigned with a bounding region ID $\mathcal{L}$. If a voxel $\mathbf{x}$ belongs to bounding region i,

$$\mathcal{L}(\mathbf{x}) = i. \tag{6.2.4}$$

The bounding region ID of each voxel $\mathbf{x}$ can be determined by the arrival time at $\mathbf{x}$ $(T_i(\mathbf{x}))$ of each front. If $T_i(\mathbf{x}) < T_j(\mathbf{x}(j = 0, \ldots, n|i))$, $\mathbf{x}$ is more likely a voxel belonging to organ i instead of the remaining organs. If front i arrives the earliest, $\mathcal{L}(\mathbf{x}) = i$.

$$i = \arg\min T_i(\mathbf{x}). \tag{6.2.5}$$

Since a typical CT or MR volume image takes up hundreds of Megabytes of memory, computing the arrival time of n fronts will result in n volume. These intermediate resulting volume will take huge amount of memory. To reduce the memory footprint, we only compute arrival time of the voxels within a bounding box for each front. Let $BB_i = \{x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}\}$ denote such a bounding box corresponding to the target organ i, and $BB'_i = \{x'_{min}, x'_{max}, y'_{min}, y'_{max}, z'_{min}, z'_{max}\}$ denote the bounding box of the mesh M_i, and w_i denote the pre-defined width of the bounding region i. BB_i can be computed by offsetting BB'_i by width w_i.

$$x_{min} = x'_{min} - w_i, \tag{6.2.6}$$

$$x_{max} = x'_{max} + w_i, \tag{6.2.7}$$

$$y_{min} = y'_{min} - w_i, \tag{6.2.8}$$

$$y_{max} = y'_{max} + w_i, \tag{6.2.9}$$

$$z_{min} = z'_{min} - w_i, \tag{6.2.10}$$

$$z_{max} = z'_{max} + w_i. \tag{6.2.11}$$

Front propagation within BB_i will greatly reduces the memory footprint and computation time, especially when the size of the target organ is relatively small compared to the entire CT/MR volume.

6.3 Segmentation within the Bounding Region

Segmentation of each object in multiple-object segmentation algorithm is similar as that in the single-object segmentation method. The difference is that the deformed mesh remains within a bounding region. In order to do this, (1) the algorithm searches correspondence for each mesh vertex within the respective bounding region, (2) mesh vertices close to the border of neighboring bounding regions will remain in their own bounding region during deformation.

To satisfy (1), correspondence search for each vertex along the surface normal direction needs to check whether current voxel belongs to the neighboring bounding regions. If that is the case, the very last voxel along the direction within its own bounding region will be its corresponding point, and this vertex is defined as *border* vertex. To satisfy (2), the

positional weights for border vertices are increased to a large value so that the positions of these vertices will not be affected by other constraints such as Laplacian preservation.

6.4 Experiments and Discussions

Experiments have been carried out to verify the effectiveness of the proposed multipleobjects segmentation algorithm regarding to the leakage problem (Section. 6.4.1). Convergence of the algorithm is also investigated (Section 6.4.2). Results of simultaneous segmentation of multiple important soft organs inside abdomen are also given (Section. 6.4.3).

6.4.1 Alleviation of the Leakage Problem

The objective of this test is to demonstrate that the leakage problem can be alleviated using the proposed algorithm. The data set was again obtained from the same MICCAI data as described in the previous chapter.

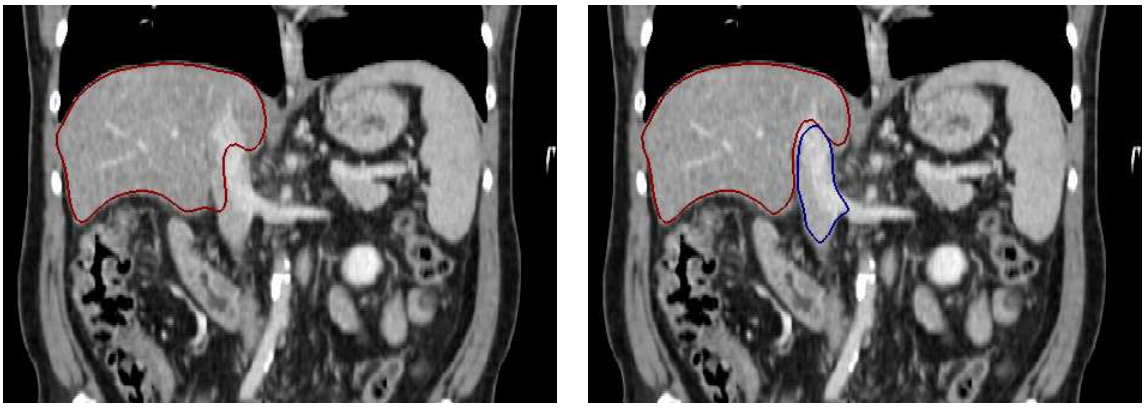
The test data include all the CT volumes tested for the single-object segmentation algorithm (Chapter 5). Results produced by the multiple-object segmentation algorithm are compared with those produced by single-object segmentation algorithm both qualitatively and quantitatively. For quantitative evaluation, the volume overlap error (Chapter 5) and the average surface distance are computed between the segmentation results and the ground-truth data.

The average distance between the surface (A) of the segmented volume and the surface (B) of the ground truth volume is computed as follows:

$$d = \frac{\sum_{a \in A}[\min_{b \in B} \mathcal{D}(a,b)] + \sum_{b \in B}[\min_{a \in A} \mathcal{D}(a,b)]}{N_A + N_B}, \tag{6.4.1}$$

where a and b denote points on surface A and B respectively. N_A and N_B denote the number of points on A and B. $\mathcal{D}(a,b)$ denotes the distance between point a and b [DD08].

Most of the leakage problems in the liver segmentation experiments happen at the boundary between the liver and the inferior vena cava (IVC). This is because the intensity values of IVC is very close to those of livers. Therefore, apart from the initial spherical



(a) (b)

Figure 6.5: Leakage problem. Liver segmentation result (dark red) produced by the single-object segmentation algorithm (a) leaks into IVC. The result produced by the multiple-object segmentation algorithm (b) has no leakage problem thanks to the constraint provided by IVC (blue).

model for liver, an extra spherical mesh is placed inside IVC. The multiple-object segmentation algorithm is executed for fixed number of iterations, with exactly the same parameters across different test volume images.

To show that the leakage problem in the single-object segmentation can be alleviated by using the multiple-object segmentation algorithm, liver segmentation results using both of the algorithms from the same input volume image are compared. As shown in one coronal view (Fig. 6.5(a)), the segmented liver (red) region leaks into IVC. In contrast, as shown in Fig. 6.5(b), with the constraint provided by IVC (blue), the segmented liver has no leakage problem, and segmented liver and IVC have no overlap with each other.

To further verify the effectiveness of the multiple-object segmentation algorithm, a quantitative comparison was also performed. The liver segmentation results produced by the multiple-object segmentation algorithm were evaluated according to the ground truth data in terms of average surface distance and volume overlap error. Fig. 6.6 shows that the multiple-object segmentation algorithm (red) achieved both lower volume overlap error (a) and smaller average surface distance (b) compared to single-object segmentation algorithm (green), for all test cases.

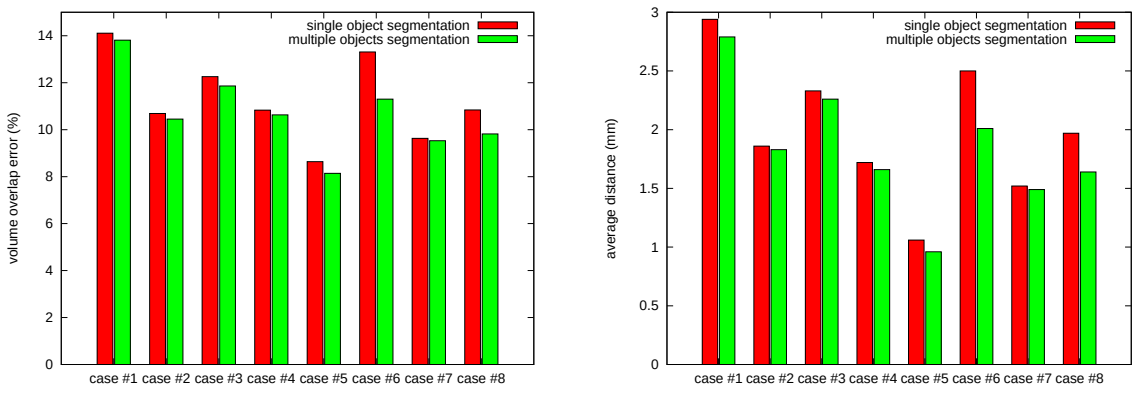


Figure 6.6: Single-object segmentation vs multiple-object segmentation. For all 8 test cases, multiple-object segmentation algorithm (red) achieved both lower volume overlap error (a) and smaller average surface distance (b) compared to single-object segmentation algorithm (green), thanks to alleviated leakage problem.

6.4.2 Convergence

The convergence of the algorithm can be demonstrated by volume overlapping error curves when the algorithm iterates. The volume overlapping error curves of a segmented liver using both the single-object segmentation algorithm (green)and the multiple-object segmentation algorithm (red) were plotted in Fig. 6.7. As can be seen in the figure, the multiple-object segmentation algorithm can converge as it iterates. It achieved lower error thanks to simultaneous segmentation of the neighboring organ (IVC), which alleviate the leakage problem.

6.4.3 Qualitative Segmentation Results

To demonstrate that the proposed multiple-object segmentation algorithm can segment many objects simultaneously, an experiment on segmenting multiple important abdominal soft organs including liver, part of the heart chamber, gallbladder, left and right kidneys, IVC and spleen was carried out using the same testing images.

The algorithm was initialized by placing an initial spherical mesh inside each of the target object. The segmentation results were examined qualitatively.

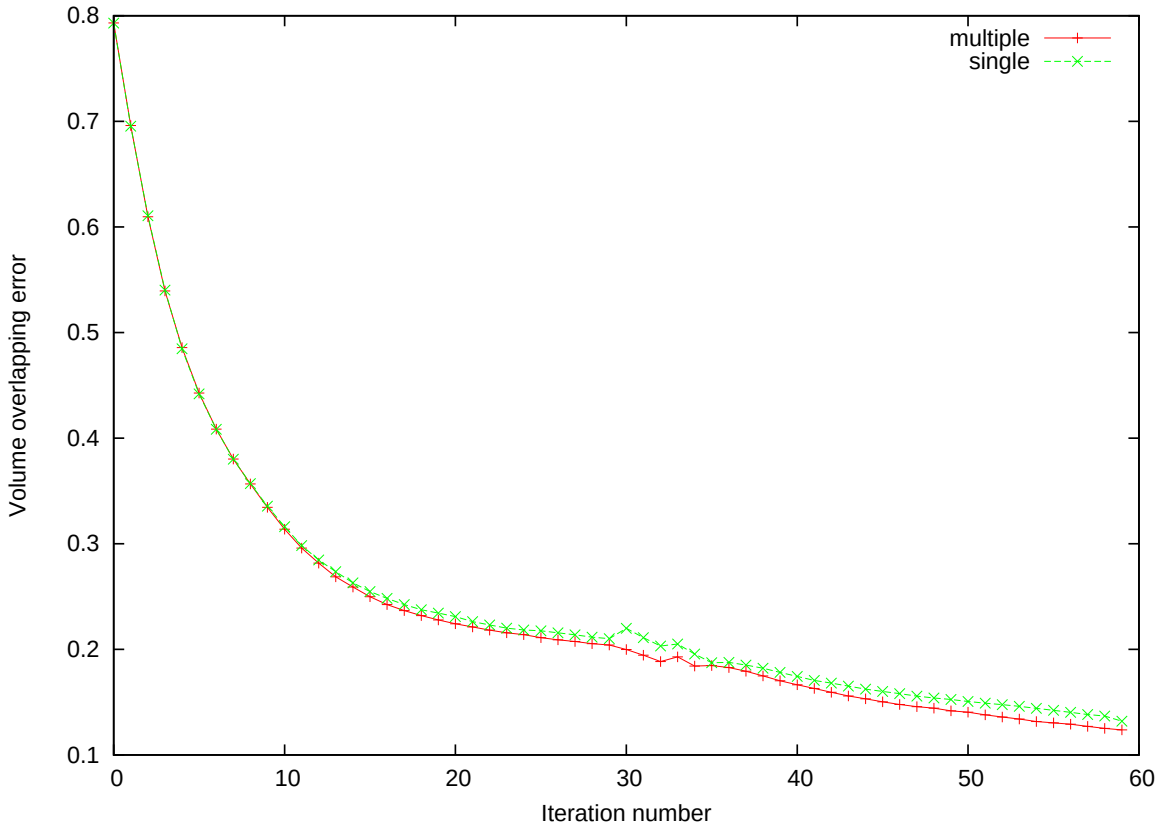


Figure 6.7: Convergence curve. The multiple-object segmentation algorithm (red) can converge, and it achieves lower segmentation error than the single-object segmentation algorithm (green) does.

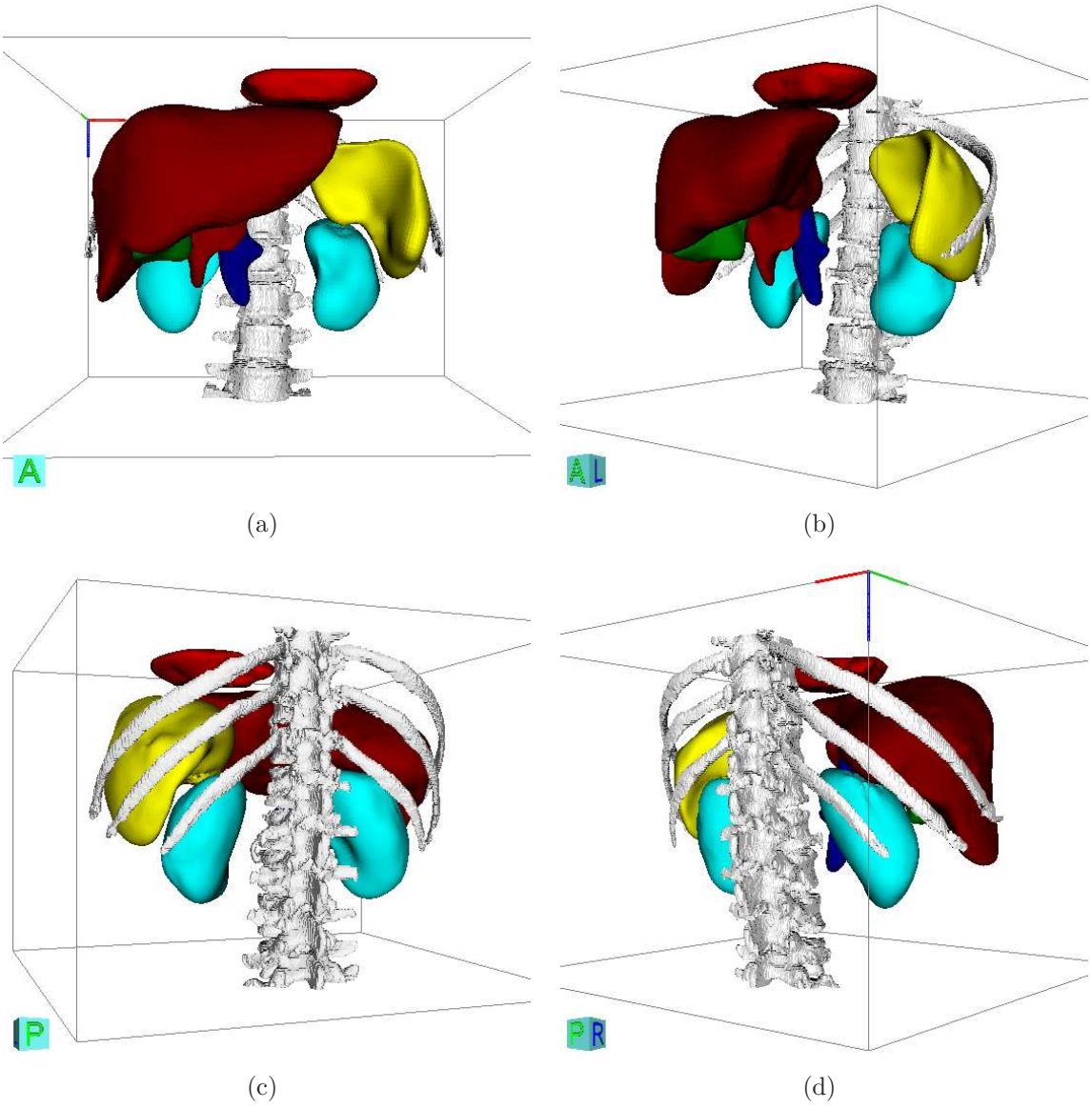
As shown in Fig. 6.8, multiple abdominal organs, i.e., liver (dark red), part of the heart chamber (red), gallbladder (green), left and right kidneys (blue), IVC (dark blue) and spleen (yellow), are successfully segmented. The spine and ribs (white) are extracted using simple thresholding algorithm and used to indicate the pose of the patient. As can be seen in the figure, the segmented organs have no intersections with each other. More multiple-object segmentation examples are given in Fig. 6.9, Fig. 6.10 and Fig. 6.11 where the sizes, shapes, orientations and positions of the segmented organs from various patients are significantly different. This proves the capabilities of the multiple-object segmentation algorithm to handle images from different patients.

6.4.4 Execution Time

This test compares the execution time spent on computing the deformation bounding region and that spent on computing the deformation itself with the current implementation of the algorithm, which has not been optimized. To measure the time, single-object segmentation in one test image was performed for just 1 iteration, after bounding region computation. The required time to compute the bounding region takes about 2.65 times of that required to compute the deformation itself does. Computation of the bounding regions using the fast marching algorithm is computationally expensive, but is highly parallel-able. Accelerating bounding region computation will certainly result in a much faster multiple-object segmentation algorithm. Possible means for acceleration are discussed in the future work section (Section 7.2).

6.5 Summary

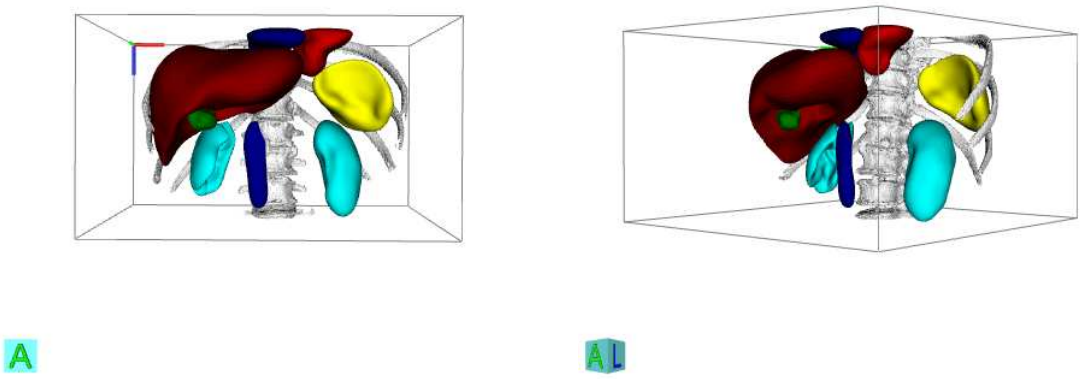
This chapter presented a multiple-object segmentation algorithm that can segment multiple objects int he input volume image simultaneously. During the segmentation process, the proposed algorithm computes a bounding region for each target object using the fast marching algorithm. These bounding regions are updated during iterations to gradually resemble the shape of the target object. The border of the neighboring bounding regions is determined by the neighboring fronts that arrive at the same time. The computed bounding regions are used to constrain the segmentation of each individual object such that



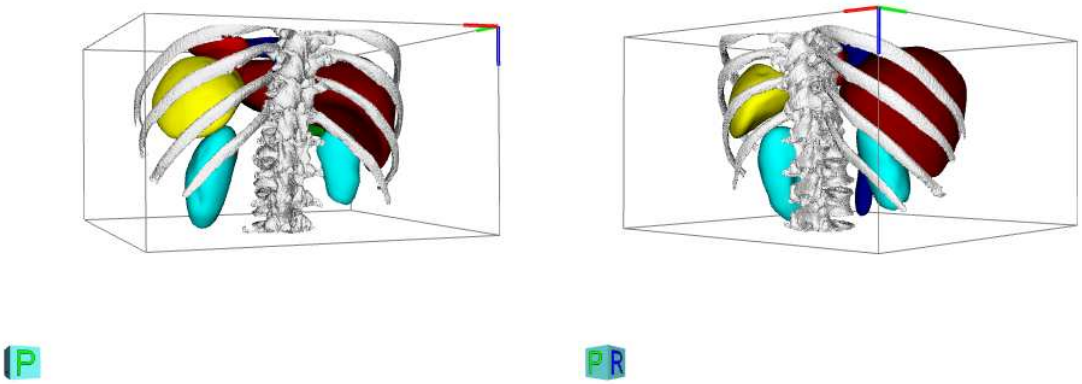
(a) (b)

(c) (d)

Figure 6.8: Multiple-object segmentation of abdominal organs, first example. (a)–(d) Different views of the segmented multiple abdominal organs, i.e., liver (dark red), part of the heart chamber (red), gallbladder (green), left and right kidneys (blue), IVC (dark blue) and spleen (yellow), using the multiple-object segmentation algorithm. The spine and ribs (white) are extracted using simple thresholding algorithm and used to indicate the pose of the patient.

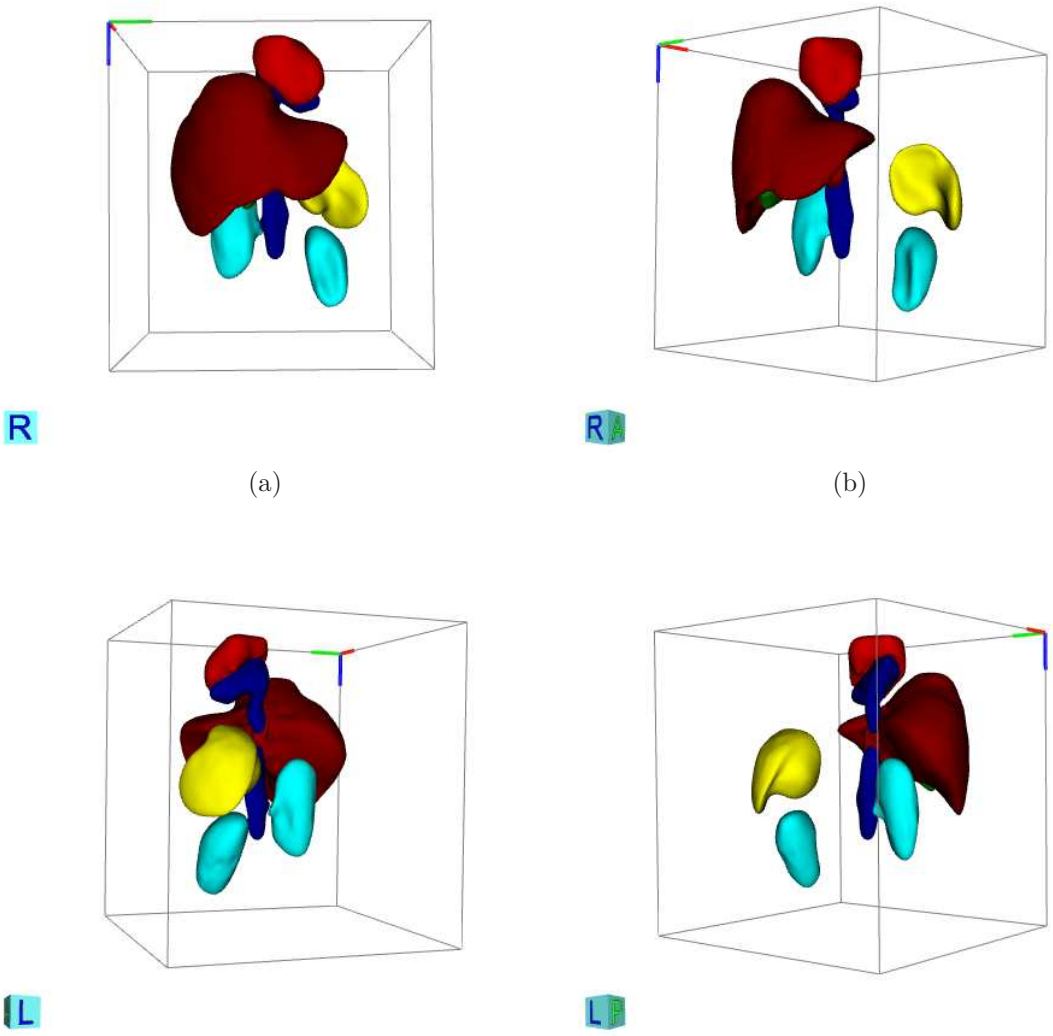


(a) (b)



(c) (d)

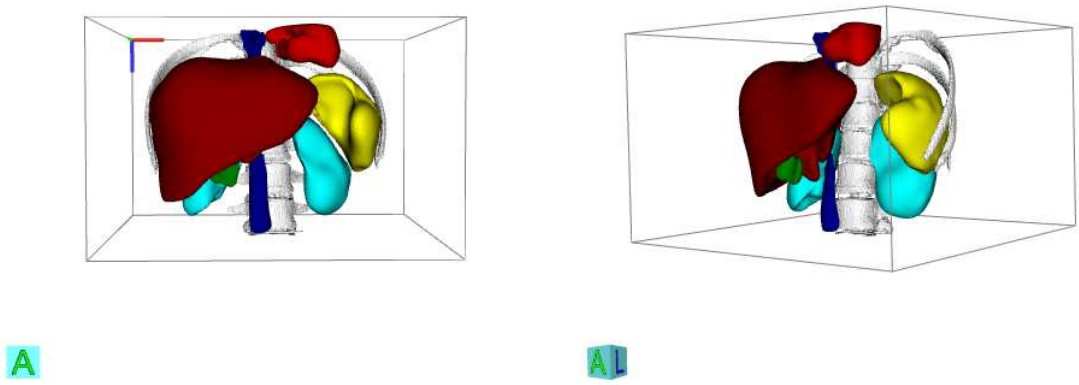
Figure 6.9: Multiple-object segmentation of abdominal organs, second example.



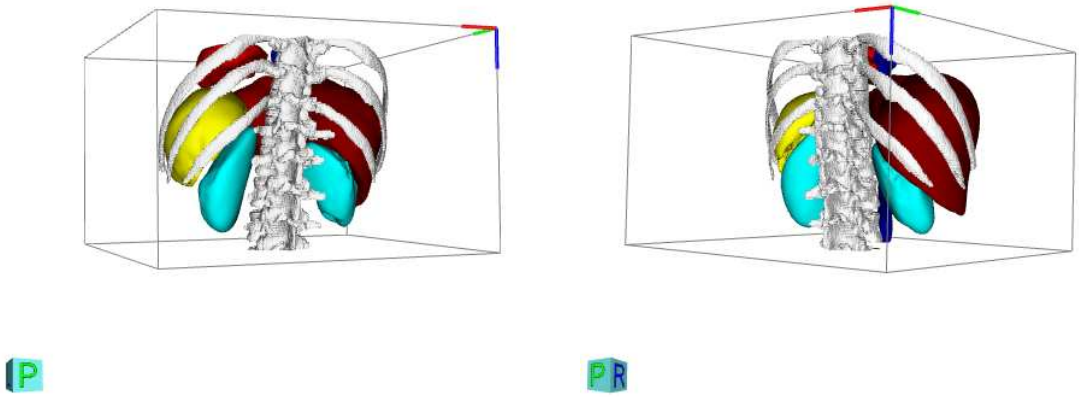
(a) (b)

(c) (d)

Figure 6.10: Multiple-object segmentation of abdominal organs, third example.



(a) (b)



(c) (d)

Figure 6.11: Multiple-object segmentation of abdominal organs, fourth example.

no overlap will happen between neighboring segmented objects. Within each bounding region, the single-object segmentation algorithm (Chapter 5) is performed with modifications to satisfy (1) correspondence search is within the bounding region and (2) border vertices have very large positional weights during segmentation. There conditions ensure that single-object segmentation is carried out within the respective bounding region.

Experiments were carried out to evaluate the performance of the multiple-object segmentation algorithm in terms of accuracy. Test results show that the algorithm achieved better liver segmentation results that the single-object segmentation algorithm did. The results produced by the multiple-object segmentation algorithm have lower volume overlap error and smaller average surface distance, thanks to the alleviated leakage problem. Experiment on segmenting multiple major abdominal soft organs verified its capabilities in segmenting multiple objects simultaneously.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Medical image segmentation has been a very hot research topic over the years. Many existing segmentation algorithms make use of statistical shape constraints of target objects. With appropriate training, these algorithms can perform well on certain objects with normal shapes. However, shapes of organs can vary significantly among patients. They can also be greatly deformed due to diseases. Parts of the organs may be removed by surgical operations. The shapes of these organs are unlikely to be predictable. It is almost impossible for an algorithm to handle these deformed objects with existing normally-shaped objects as training samples. This thesis presents a novel 3D segmentation algorithm based on flipping-free mesh deformation to segment organs of various shapes from different patients. The proposed segmentation algorithm does not rely on any shape priors, and therefore is able to segment various complex shapes. It is extended to segment multiple objects simultaneously in medical volume images.

To solve the segmentation problem, a deformable model-based segmentation algorithm is proposed. The model is explicitly represented as a mesh surface. In contrast to implicit representation such as the level set, the mesh representation is much more efficient. However, improperly deforming a mesh may produce mesh flipping that will severely affect the correctness and the usefulness of the segmentation results. To prevent flipping, a possible flipping detection and avoidance mechanism is introduced before each deformation iteration. Experiments of the proposed flipping-free mesh deformation algo-

rithm show that the algorithm is flipping-free, efficient, able to handle concave objects, and insensitive to parameter setting.

The proposed mesh deformation algorithm is then applied to 3D object segmentation. The algorithm searches for each vertex its possible correspondence in the image. The search is based on estimated intensity distribution of the target object constructed from the initial spherical model. The estimated correspondences function as positional constraints to displaced the vertices. Vertices with no correspondence can be displaced according to local geometric properties. Experiments of the single-object segmentation algorithm demonstrate that it is much faster than other segmentation algorithms such as snake, level set, and graph-cut. It also produced more accurate segmentation results compared to these algorithms. Experiments also show that the proposed segmentation algorithm can segment objects with very different shapes.

To extend the segmentation algorithm to segmenting multiple objects simultaneously, the proposed algorithm computes a bounding region for each target object using the fast marching algorithm. These bounding regions are updated during iterations to gradually resemble the shape of the target object. The borders of the neighboring bounding regions are determined by the neighboring fronts that arrive at the same time. The computed bounding regions are used to constrain the segmentation of each individual object such that no overlap will happen between neighboring segmented objects. Within each bounding region, the single-object segmentation algorithm (Chapter 5) is performed subject to (a) correspondence search is within the bounding region and (b) border vertices have very large positional weights during segmentation. There conditions ensure that single-object segmentation is carried out within the respective bounding region. Test results show that the multiple-object segmentation algorithm achieved better liver segmentation results than the single-object segmentation algorithm did. It produced results with lower volume overlap error and smaller average surface distance. Experiment on segmenting multiple major abdominal soft organs verified its capabilities in segmenting multiple objects simultaneously.

7.2 Future Work

The proposed 3D segmentation algorithm can be enhanced and extended to reduce its execution time and further increase its robustness.

During mesh deformation, solving the linear equations in each iteration takes up most of the time. In current implementation, the linear system is solved by CPU using the Taucs library. Solving the linear equation using GPU implementation [BFGS03, JW03] will reduce the time by a large fraction.

During the segmentation process, computation of bounding regions using the fast marching method takes a lot of time. The computational complexity of the fast marching method is $O(n \log n)$, where n is the number of voxels in the image. This process can be speeded up by down sampling the volume since the bounding region needs not to be of single voxel accuracy. Down sampling may speed up the fast marching algorithm a lot. Further acceleration is possible if the fast marching algorithm is implemented using GPU. A GPU-based implementation working on 2D images is currently available [JW07].

To further increase the robustness of the algorithm, different features may be incorporated into the algorithm. Current implementation use estimated intensity distribution to guide correspondence search. Other features such as edges and textures can be easily incorporated into the current framework. These features may help the algorithm to differentiate between foreground and background when voxel intensities of neighboring objects are similar.

Publication List

[1] Feng Ding, Wee Kheng Leow and Shih-Chang Wang "Segmentation of 3D CT Volume Images Using a Single 2D Atlas", *Proc. of Computer Vision for Biomedical Image Applications*, pages 459–468, 2005.

[2] Feng Ding, Wee Kheng Leow and Tet Sen Howe "Automatic Segmentation of Femur Bones in Anterior-Posterior Pelvis X-Ray Images", *Proc. of Computer Analysis of Images and Patterns*, pages 205–212, 2007.

[3] Feng Ding, Wee Kheng Leow and Sudhakar Venkatesh "Removal of Abdominal Wall for 3D Visualization and Segmentation of Organs in CT Volume", *Proc. of IEEE International Conference on Image Processing*, 2009.

[4] Feng Ding, Jiayin Zhou, Wee Kheng Leow and Sudhakar Venkatesh "3D Volume Image Segmentation by Flipping-Free Mesh Deformation with Application to Liver Segmentation", *Radiological Society of North America, abstract*, 2009.

[5] Feng Ding, Wenxian Yang, Wee Kheng Leow and Sudhakar Venkatesh "3D Segmentation of Soft Organs by Flipping-Free Mesh Deformation", *Proc. of IEEE Workshop on Applications of Computer Vision*, 2009.

[6] Feng Ding, Hao Li, Yuan Cheng and Wee Kheng Leow "Medical Volume Image Summarization", *Proc. of IEEE Workshop on Applications of Computer Vision*, 2009.

[7] Jia-Yin Zhou, Damon W.K. Wong, Feng Ding, Sudhakar K. Venkatesh, Qi Tian, Ying-Yi Qi, Wei Xiong, Jimmy J. Liu and Wee-Kheng Leow "Liver Tumor Segmentation Using Contrast-enhanced Multi-detector CT data: Performance Benchmarking of Three Semi-automated Methods", *European Radiology*, 2009.

[8] Feng Ding, Wee Kheng Leow and Sudhakar Venkatesh "3D segmentation of soft organs by flipping free mesh deformation", *European Congress of Radiology, e-poster*, 2010.

Bibliography

[AB94] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(6):641–647, June 1994.

[AM00] M. Stella Atkins and Blair T. Mackiewich. Fully automated hybird segmentation of the brain. In Issac N. Bankman, editor, *Handbook of Medical Imaging, Processing and Analysis*, chapter 11, pages 171–183. Academic Press, 2000.

[ANWD99] Georges B. Aboutanos, Jyrki Nikanne, Nancy Watkins, and Benoit M. Dawant. Model creation and deformation for the automatic segmentation of the brain in MR images. *IEEE Transaction on Biomedical Engineering*, 46(11):1346–1356, 1999.

[BAA09] Timothy A. Burkhart, Katherine L. Arthurs, and David M. Andrews. Manual segmentation of dxa scan images results in reliable upper and lower extremity soft and rigid tissue mass estimates. *Journal of Biomechanics*, 42(8):1138–1142, 2009.

[BCT+08] Kolawole Babalola, Tim Cootes, Carole Twining, Vlad Petrovic, and Chris Taylor. 3d brain segmentation using active appearance models and local regressors. In *Medical Image Computing and Computer-Assisted Intervention*, pages 401–408, 2008.

[BFGS03] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *ACM Transactions on Graphics*, 22(3):917–924, 2003.

[BHTR90] Michael Bomans, Karl-Heihz Höhne, Ulf Tiede, and Martin Riemer. 3-D segmentation of MR images of the head for 3-D display. *IEEE Transactions on Medical Imaging*, 9(2):177–183, June 1990.

[BJ01] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of ojbects in N-D images. In *Proceedings of International Conference on Computer Vision*, pages 105–112, 2001.

[BL79] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Proc. International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, September 1979.

[BMC+05] Pierre-Yves Bondiau, Grégoire Malandain, Stéphane Chanalet, Pierre-Yves Marcy, Jean-Louis Habrand, Francois Fauchon, Philippe Paquis, Adel Courdi, Olivier Commowick, Isabelle Rutten, and Nicholas Ayache. Atlas-based automatic segmentation of MR images: Validation study on the brainstem in radiotherapy context. *International Journal of Radiation Oncology Biology Physics*, 61(1):289–298, January 2005.

[BMGNJM+97] Matthew S. Brown, Michael F. Mcnitt-Gray, Jonathan G. Goldin Nicholas J. Mankovich, John Hiller, Laurence S. Wilson, and Denise R. Aberle. Method for segmenting chest CT image data using an anatomical model: Preliminary results. *IEEE Transactions on Medical Imaging*, 16(6):828–839, 1997.

[BS09] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, 82(2):113–132, 2009.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[CBD+03] P. Cachier, E. Bardinet, D. Dormont, X. Pennec, and N. Ayache. Iconic feature based nonrigid registration: The PASHA algorithm. *CVIU*

— *Special Issue on Nonrigid Registration*, 89(2-3):272–298, Feb.-march 2003.

[CCMT01] M. B. Cuadra, O. Cuisenaire, R. Meuli, and J. P. Thiran. Automatic segmentation of internal structures of the brain in MR images using a tandem of affine and non-rigid registration of an anatomical brain atlas. In *Image Processing*, pages 1083–1086, Oct. 2001.

[CET98] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. European Conference on Computer Vision*, volume 2, pages 484–498, 1998.

[CHT+03] Yunmei Chen, Feng Huang, Hemant D. Tagare, Murali Rao, David Wilson, and Edward A. Geiser. Using prior shape and intensity profile in medical image segmentation. In *Proceedings of Ninth IEEE International Conference on Computer Vision*, 2003.

[CHTJ94] T. F. Cootes, A. Hill, C. J. Taylor, and J.Haslam. The use of active shape models for locating structures in medical images. *Image and Vision Computing*, 12(6):355–366, July 1994.

[CKRP98] Mei Chen, T. Kanade, H. A. Rowley, and D. Pomerleau. Quantitative study of brain anatomy. In *Mathematical Methods in Biomedical Image Analysis*, pages 84–92, June 1998.

[CL00] Yongchoel Choi and Seungyong Lee. Injectivity conditions of 2D and 3D uniform cubic b-spline functions. *Graphical Models*, 62(6):411–427, 2000.

[CMA+98] Fiorino C, Reni M, Bolognesi A, Cattaneo GM, and Calandrino R. Intra- and inter-observer variability in contouring prostate and seminal vesicles: implications for conformal treatment planning. *Radiotherapy and Oncology*, 47(3):285–292, 1998.

[CMB+98] Luigi Franco Cazzaniga, Maria Antonella Marinoni, Alberto Bossi, Ernestina Bianchi, Emanuela Cagna, Dorian Cosentino, Luciano Scandolaro, Marica Valli, and Milena Frigerio. Interphysician variability in

defining the planning target volume in the irradiation of prostate and seminal vesicles. *Radiotherapy and Oncology*, 47(3):293–296, 1998.

[CP00] Pascal Cachier and Xavier Pennec. 3D non-rigid registration by gradient descent on a gaussian-windowed similarity measure using convolutions. In *Mathematical Methods in Biomedical Image Analysis*, pages 182–189, June 2000.

[CP04] Fernand S. Cohen and Chuchart Pintavirooj. Invariant surface alignment in the presence of affine and some nonlinear transformations. *Medical Image Analysis*, 8(2):151–164, June 2004.

[CPB+04] Meritxell Bach Cuadra, Claudio Pollo, Anton Bardera, Olivier Cuisenaire, Jean-Guy Villemure, and Jean-Philippe Thiran. Atlas-based segmentation of pathological MR brain images using a model of lesion growth. *IEEE Transaction on Medical Imaging*, 23(10):1301–1314, 2004.

[CT04] T. F. Cootes and C. J. Taylor. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering, Manchester M13 9PT, U.K., March 2004.

[CV01] T. F. Chang and L. A. Vese. Active contours without edges. *Image Processing, IEEE Transcations on*, 10(2):266–277, 2001.

[DD08] Xiang Deng and Guangwei Du. Editorial: 3D segmentation in the clinic: A grand chanllenge ii - liver tumor segmentation. 2008.

[DHT+99] B. M. Dawant, S. L. Hartmann, J. P. Thirion, F. Maes, D. Vandermeulen, and P. Demaerel. Automatic 3-D segmentation of internal structures of head in MR images using a combination of similarity and free-form transformations: Part I, methodology and validation on normal subjects. *IEEE Trans. on Medical Imaging*, 18(10):909–916, 1999.

[DHT05] Valérie Duay, Nawal Houhou, and Jean-Philippe Thiran. Atlas-based segmentation of medical images locally constrained by level sets. Technical report, Signal Processing Institute (ITS), Ecole Polytechnique Fédérale de Lausanne (EPFL), 2005.

[DLCL09] Feng Ding, Hao Li, Yuan Cheng, and Wee Kheng Leow. Medical volume image summarization. In *Proc. of IEEE Workshop on Applications of Computer Vision*, 2009.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[DLV09] Feng Ding, Wee Kheng Leow, and Sudhakar Venkatesh. Removal of abdominal wall for 3D visualization and segmentation of organs in CT volume. In *Proc. of IEEE International Conference on Image Processing*, 2009.

[DLW05] Feng Ding, Wee Kheng Leow, and Shih-Chang Wang. Segmentation of 3D CT volume images using a single 2D atlas. In *Proc. First International Workshop on Computer Vision for Biomedical Image Applications (CVBIA 2005)*, pages 459–468, 2005.

[DM01] H. Delingette and J. Montagnat. Shape and topology constraints on parametric active contours. *Computer Vision and Image Understanding*, 83(2):140–171, 2001.

[DMRS01] M. Droske, B. Meyer, M. Rumpf, and K. Schaller. An adaptive level set method for medical image segmentation. In *Proc. of the Annual Symposium on Information Processing in Medical Imaging*, 2001.

[FCWO07] Jurgen Fripp, Stuart Crozier, Simon Warfield, and Sébastien Ourselin. Automatic segmentation of articular cartilage in magnetic resonance images of the knee. In *MICCAI*, pages 186–194, 2007.

[FLC91] Jie Feng, Wei-Chung Lin, and Chin-Tu Chen. Epicardial boundary detection using fuzzy reasoning. *Medical Imaging, IEEE Transactions on*, 10(2):187–199, June 1991.

[GDMW04] Klaus A. Ganser, Hartmut Dickhaus, Roland Metzner, and Christian R. Wirtz. A deformable digital brain atlas system according to talairach and tournoux. *Medical Image Analysis*, 8(1):3–22, March 2004.

[GMA+04] V. Grau, A.U.J. Mewes, M. Alcañiz, R. Kikinis, and S.K. Warfield. Improved watershed transform for medical image segmentation using prior information. *IEEE Transaction on Medical Imaging*, 23(4):447–458, 2004.

[GSSA+02] S. S. Gleason, H. Sari-Sarraf, M. A. Abidi, O. Karakashian, and F. Morandi. A new deformable model for analysis of X-Ray CT images in preclinical studies of mice for polycystic kidney disease. *IEEE Transactions on Medical Imaging*, 21(10):1302–1309, October 2002.

[GT95] Ardeshir Goshtasby and David A. Turner. Segmentation of cardiac cine MR images for extraction of right and left ventricular chambers. *IEEE Transactions on Medical Imaging*, 14(1):56–64, March 1995.

[GW01] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2001.

[HAHR08] Ilker Hacihaliloglu, Rafeef Abugharbieh, Antony Hodgson, and Robert Rohling. Bone segmentation and fracture detection in ultrasound using 3d local phase features. In *Medical Image Computing and Computer-Assisted Intervention*, pages 287–295, 2008.

[HF98] Michael Hagenlocker and Kikuo Fujimura. CFFD: a tool for designing flexible shapes. *The Visual Computer*, 14:271–287, 1998.

[HHK92] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *ACM SIGGRAPH*, pages 177–184, 1992.

[HKR+08] Piotr Habas, Kio Kim, Francois Rousseau, Orit Glenn, A. Barkovich, and Colin Studholme. Atlas-based segmentation of the germinal matrix from in utero clinical MRI of the fetal brain. In *Medical Image Computing and Computer-Assisted Intervention*, pages 351–358, 2008.

[HPMD99] Steven L. Hartmann, Mitchell H. Parks, Peter R. Martin, and Benoit M. Dawant. Automatic 3-D segmentation of internal structures of head in

MR images using a combination of similarity and free-form transformations: Part II, validation on severely atrophied brains. *IEEE Trans. on Medical Imaging*, 18(10):917–926, 1999.

[HPS+99] S. L. Hartmann, M. H. Parks, H. Schlack, W. Riddle, R. R. Price, P. R. Martin, and B. M. Dawant. Automatic computation of brain and cerebellum volumes in normal subjects and chronic alcoholics. In Attila Kuba, Martin Sámal, and Andrew Todd-Pokropek, editors, *IPMI*, volume 1613, pages 430–435. Springer, 1999.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the 4th ALVEY Vision Conference*, pages 147–151, 1988.

[HTC92] A. Hill, C. J. Taylor, and T. Cootes. Object recognition by flexible template matching using genetic algorithms. In *European Conference on Computer Vision*, pages 852–856, May 1992.

[HTVF92] William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Nemerical Recipes in C: The Art of Scientific Computing (second edition)*. Cambridge University Press, 1992.

[JAA+95] Lebesque JV, Bruce AM, Kroes AP, Touw A, Shouman RT, and van Herk M. Variation in volumes, dose-volume histograms, and estimated normal tissue complication probabilities of rectum and bladder during conformal radiotherapy of T3 prostate cancer. *International journal of radiation oncology, biology, physics*, 33(5):1109–1119, 1995.

[JSC04] W. Jung, H. Shin, and B. K. Choi. Self-intersection removal in triangular mesh offsetting. In *Computer-Aided Design and Applications*, pages 477–484, 2004.

[JW03] Krüger Jens and Rüdiger Westermann. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Transactions on Graphics*, 22(3):908–916, 2003.

[JW07] W. K. Jeong and R. Whitaker. A fast eikonal equation solver for parallel systems. In *SIAM: Conference on Computational Science and Engineering*, pages 1–6, 2007.

[KAB+05] Ali Khan, Elizabeth Aylward, Patrick Barta, Michael Miller, and M. Faisal Beg. Semi-automated basal ganglia segmentation using large deformation diffeomorphic metric mapping. In *MICCAI*, pages 238–245, 2005.

[KWJK98] M. Kaus, S. K. Warfield, F. A. Jolesz, and R. Kikinis. Adaptive template moderated brain tumor segmentation in MRI. In *Bildverarbeitung fur die Medizin*, pages 102–106. Springer Verlag, 1998.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[LAGBA05] Marius George Linguraru, Miguel Ángel González Ballester, and Nicholas Ayache. Deformable atlases for the segmentation of internal brain nuclei in magnetic resonance imaging. *Acta Universitatis Cibiniensis, Technical Series*, 2005. to appear.

[LHD+08] Chunming Li, Rui Huang, Zhaohua Ding, Chris Gatenby, Dimitris Metaxas, and John Gore. A variational level set approach to segmentation and bias correction of images with intensity inhomogeneity. In *Medical Image Computing and Computer-Assisted Intervention*, pages 1083–1091, 2008.

[LL03] W. K. Leow and R. Li. The analysis and applications of adaptive-binning color histograms. *Computer Vision and Image Understanding*, 94(1–3):67–91, 2003.

[LM98] Jacques-Olivier Lachaud and Annick Montanvert. Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction. *Medical Image Analysis*, 3(2):187–207, 1998.

[LMT99] Jianming Liang, Tim McInerney, and Demetri Terzopoulos. Interactive medical image segmentation with united snakes. In *Proc. of Medical Image Computing and Computer-Assisted Intervention*, pages 116–127, 1999.

[LVRMSO03] M. Lorenzo-Valdés, D. Rueckert, R. Mohiaddin, and G. I. Sanchez-Ortiz. Segmentation of cardiac MR images using the em algorithm with 4D probabilistic atlas and a global connectivity filter. In *International Conference of the IEEE Engineering in Medicine and Biology Society*, 2003.

[LVSOE+04] M. Lorenzo-Valdés, G. I. Sanchez-Ortiz, A. G. Elkington, R. H. Mohiaddin, and D. Rueckert. Segmentation of 4D cardiac MR images using a probabilistic atlas and the EM algorithm. *Medical Image Analysis*, 8:255–265, 2004.

[LVSOMR02] M. Lorenzo-Valdés, G. I. Sanchez-Ortiz, R. Mohiaddin, and D. Rueckert. Atlas-based segmentation and tracking of 3D cardiac MR images using non-rigid registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 642–650, 2002.

[MS89] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Communications in Pure and Applied Mathematics*, 42:577–685, 1989.

[MSV95] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, February 1995.

[MT99] Tim McInerney and Demetri Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *TMI*, 18(10):840–850, 1999.

[MTA+08] Jonathan Morra, Zhuowen Tu, Liana Apostolova, Amity Green, Arthur Toga, and Paul Thompson. Automatic subcortical segmentation using a

contextual model. In *Medical Image Computing and Computer-Assisted Intervention*, pages 194–201, 2008.

[NS96] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18, Dec 1996.

[OS83] L. O'Gorman and A. C. Sanderson. The converging squares algorithm: An efficient multidimensional peak picking method. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 112–115, April 1983.

[OSS+07] Toshiyuki Okada, Ryuji Shimada, Yoshinobu Sato, Masatoshi Hori, Keita Yokota, Masahiko Nakamoto, Yen-Wei Chen, Hironobu Nakamura, and Shinichi Tamura. Automated segmentation of the liver from 3d ct images using probabilistic atlas and multi-level statistical shape model. In *Medical Image Computing and Computer-Assisted Intervention, MICCAI 2007*, pages 86–93, 2007.

[OYH+08] Toshiyuki Okada, Keita Yokota, Masatoshi Hori, Masahiko Nakamoto, Hironobu Nakamura, and Yoshinobu Sato. Construction of hierarchical multi-organ statistical atlases and their application to multi-organ segmentation from ct images. In *Medical Image Computing and Computer-Assisted Intervention*, pages 502–509, 2008.

[PBM03] H. Park, P. H. Bland, and C. R. Meyer. Construction of an abdominal probabilistic atlas and its application in segmentation. *IEEE Trans. on Medical Imaging*, 22(4):483–492, 2003.

[PGLG05] Marcel Prastawa, John H. Gilmore, Weili Lin, and Guido Gerig. Automatic segmentation of MR images of the developing newborn brain. *Medical Image Analysis*, 2005.

[PM90] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[PMWH05] Charnchai Pluempitiwiriyawej, José M. F. Moura, Yi-Jen Lin Wu, and Chien Ho. STACS: New active contour scheme for cardiac MR image segmentation. *IEEE Transactions on Medical Imaging*, 24(5):593–603, 2005.

[PT01] R. Pohle and K. Toennies. Segmentation of medical images using adaptive region growing. In *Proc. SPIE (Medical Imaging 2001)*, 2001.

[PXP98] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince. A survey of current methods in medical image segmentation. Technical report, Department of Electrical and Computer Engineering, The Johns Hopkins University, 1998.

[RD99] Peter J. Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, August 1999.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH*, pages 309–314, 2004.

[RM01] Jos B.T.M. Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41, 2001.

[RM03] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *International Conference on Computer Visioin*, Nice, France, October 2003.

[Rog00] Jadwiga Rogowska. Overview and fundamentals of medical image segmentation. In Isaac N. Bankman, editor, *Handbook of Medical Imaging, Processing and Analysis*, chapter 5, pages 69–85. Academic Press, 2000.

[RSOLV+02] D. Rueckert, G. I. Sanchez-Ortiz, M. Lorenzo-Valdés, R. Chandrashekara, and R. Mohiaddin. Non-rigid registration of cardiac MR: Application to motion modelling and atlas-based segmentation. In *IEEE International Symposium on Biomedical Imaging*, 2002.

[SA85] S. Suzuki and K. Abe. Topological structural analysis of digital binary images by border following. *CVGIP*, 30:32–46, 1985.

[SCK+03] Matús Straka, Alexandra La Cruz, Arnold Köchl, Milos Srámek, Meister Eduard Gröller, and Dominik Fleischmann. 3d watershed transform combined with a probabilistic atlas for medical image segmentation. In *MIT 2003*, pages

[SD00] Dinggang Shen and Christos Davatzikos. An adaptive-focus deformable model using statistical and geometric information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):906–913, August 2000.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[Set97] J. A. Sethian. Tracking interfaces with level sets. *American Scientist*, May–June 1997.

[Set99a] J. A. Sethian. *Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[Set99b] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[SHD01] Dinggang Shen, Edward H. Herskovits, and Christos Davatzikos. An adaptive-focus statistical shape model for segmentation and shape modeling of 3-D brain structures. *IEEE Transactions on Medical Imaging*, 20(4):257–270, April 2001.

[SL97] Stephanie Sandor and Richard Leahy. Surface-based labeling of cortical anatomy using a deformable atlas. *IEEE Transactions on Medical Imaging*, 16(1):41–54, February 1997.

[SLA08] Hong Shen, Andrew Litvin, and Christopher Alvino. Localized priors for the precise segmentation of individual vertebras from ct volume data. In *Medical Image Computing and Computer-Assisted Intervention*, pages 367–375, 2008.

[SLCO+04] Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Eurographics*, pages 179–188. ACM Press, 2004.

[SLZ+04] Dinggang Shen, Zhiqiang Lao, Jianchao Zeng, Wei Zhang, Isabel A. Sesterhenn, Leon Sun, Judd W. Moul, Edward H. Herskovits, Gabor Fichtinger, and Christos Davatzikos. Optimized prostate biopsy via a statistical atlas of cancer spatial distribution. *Medical Image Analysis*, 8(2):139–150, June 2004.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 151–160, 1986.

[SPvG05] Ingrid Sluimer, Mathias Prokop, and Bram van Ginneken. toward automated segmentation of the pathological lung in CT. *IEEE Transactions on Medical Imaging*, 24(8):1025–1038, August 2005.

[SS04] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, January 2004.

[SS08] Yonghong Shi and Dinggang Shen. Hierarchical shape statistical model for segmentation of lung fields in chest radiographs. In *MICCAI*, pages 417–424, 2008.

[TB92] D. L. Toulson and J. F. Boyce. Segmentation of MR image using neural nets. In *Image and Vision Computing*, volume 10, pages 324–328, 1992.

[Thi98] J. P. Thirion. Image matching as a diffusion process: An analogy with maxwell's demons. *Medical Image Analysis*, 2(3):243–260, 1998.

[TT07] Zhuowen Tu and Arthur Toga. Towards whole brain segmentation by a hybrid model. In *Medical Image Computing and Computer-Assisted Intervention*, pages 169–177, 2007.

[Tu05] Zhouwen Tu. Probabilistic boosting tree: Learning discriminative models for classification, recognition, and clustering. In *Proc. of ICCV*, pages 1589–1596, 2005.

[vGBvR08] Bram van Ginneken, Wouter Baggerman, and Eva van Rikxoort. Robust segmentation and anatomical labeling of the airway tree from thoracic CT scans. In *Medical Image Computing and Computer-Assisted Intervention*, pages 219–226, 2008.

[VJYL03] B. C. Vemuri, Y. Chen J. Ye, and C. M. Leonard. Image registration via level-set motion: Applications to atlas-based segmentation. *Medical Image Analysis*, 7(1):1–20, March 2003.

[WCA+08] Michael Wels, Gustavo Carneiro, Alexander Aplas, Martin Huber, Joachim Hornegger, and Dorin Comaniciu. A discriminative model-constrained graph cuts approach to fully automated pediatric brain tumor segmentation in 3-D MRI. In *Medical Image Computing and Computer-Assisted Intervention*, pages 67–75, 2008.

[WHOF96] S. Wegner, T. Harms, H. Oswald, and E. Fleck. The watershed transformation on graphs for the segmentation of CT images. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 498–502, August 1996.

[WL93] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Thoery and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, November 1993.

[WS98] Yongmei Wang and Lawrence H. Staib. Boundary finding with correspondence using statistical shape models. In *Computer Vision and Pattern Recognition*, pages 338–345, June 1998.

[WS03] Song Wang and Jeffrey Mark Siskind. Image segmentation with ratio cut. *IEEE Transcations on Pattern Analysis and Machine Intelligence*, 25(6):675–690, June 2003.

[XP98] Chenyang Xu and Jerry L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transaction on Image Processing*, 7(3), March 1998.

[XXE+08] Ye Xing, Zhong Xue, Sarah Englander, Mitchell Schnall, and Dinggang Shen. Improving parenchyma segmentation by simultaneous estimation of tissue property 1 map and group-wise registration of inversion recovery MR breast images. In *Medical Image Computing and Computer-Assisted Intervention*, pages 342–350, 2008.

[YPCH+06] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006.

[YSD04] Jing Yang, Lawrence H. Staib, and James S. Duncan. Neighbor-constrained segmentation with level set based 3-d deformable models. *IEEE Transactions on Medical Imaging*, 2004.

[YYJH+92] Xiaohan Yu, Juha Ylä-Jääski, Olli Huttunen, Tuomo Vehkomäki, Outi Sipilä, and Toivo Katila. Image segmentation combining region growing and edge detection. In *Pattern Recognition, International Conference on*, volume III, pages 481–484, April 1992.

[YZX+04] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, pages 644–651, 2004.

[ZBE+07] Yuanjie Zheng, Sajjad Baloch, Sarah Englander, Mitchell Schnall, and Dinggang Shen. Segmentation and classification of breast tumor using

dynamic contrast-enhanced MR images. In *Medical Image Computing and Computer-Assisted Intervention*, pages 393–401, 2007.

[ZBH07] Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Transformesh : A topology-adaptive mesh-based approach to surface evolution. In *ACCV*, pages 166–175, 2007.

[ZRA+08] Xiahai Zhuang, Kawal Rhode, Simon Arridge, Reza Razavi, Derek Hill, David Hawkes, and Sebastien Ourselin. An atlas-based segmentation propagation framework using locally affine registration - application to automatic whole heart segmentation. In *MICCAI*, pages 425–433, 2008.