Winter 12-15-2016

Guided Medical Data Segmentation Using Structure-Aligned Planar Contours

Michelle Vaughan Holloway
Washington University in St. Louis

WASHINGTION UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:
Tao Ju, Chair
Yasutaka Furukawa
Cindy Grimm
Caitlin Kelleher
Robert Pless
Ruth West

Guided Medical Data Segmentation Using Structure-Aligned Planar Contours

by

Michelle Holloway

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

December 2016
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

First, I want to express my deepest gratitude to my advisor Tao Ju for all his amazing support and instruction he provided me during my time at Washington University in St. Louis. I am very appreciative of always being able to discuss my problems and ideas with him, and I never would have reached the end of this milestone in my life without him. I have learned so much under his guidance and my time as his student is invaluable to me.

I would also like to thank my collaborators Cindy Grimm and Ruth West. Cindy was my original advisor who set me on the path that my research has taken to this day. She has provided an inmeasurable amount of aid for my research, and had always been willing help and provide suggestions. Ruth is an amazingly diverse individual, and her knowledge and kind words have always helped me immensely. She could always provide new insights and different viewpoints that changed the way I thought about my research.

I would like to extend thanks to my other committee members Robert Pless, Caitlin Kelleher, and Yasutaka Furukawa for their time and support for this dissertation. I would also like to thank my lab members, Ming Zou, Zhiyang Huang, Hang Dou, Yaji Yan, and Yixin Zhuang for collaborating on projects, discussing ideas, and helping preparing talks.

I would like to thank Dr. Daniel Low, formerly of the Washington University School of Medicine in St. Louis for the liver data sets, Dr. Sandra Rugonyi of Oregon Health &

Science University for the aorta data sets, and Dr. Philip Bayly of Washington University in St. Louis for the ferret brain data sets.

This work was supported by the National Science Foundation under grants DEB-1053554, IIS-1302142, IIS-1302200, and IIS-0846072.

Michelle Holloway

Washington University in Saint Louis
December 2016

Dedicated to my parents and my loving husband who have always supported me in everything that I do.

ABSTRACT OF THE DISSERTATION

Guided Medical Data Segmentation Using Structure-Aligned Planar Contours

by

Michelle Holloway

Doctor of Philosophy in Computer Science

Washington University in St. Louis, December 2016

Research Advisor: Professor Tao Ju

Segmentation of 3D/4D biological images is a critical step for a wide range of applications such as treatment planning, quantitative analysis, virtual simulations, and rendering visualizations. Automatic segmentation methods are becoming more reliable, but many experts still rely on manual intervention which makes segmentation a time and resource intensive bottleneck. Marking boundary contours in 3D images can be difficult when images are often noisy or the delineation of biological tissue is unclear. Non-parallel contours can be more accurate and reduce the amount of marking necessary, but require extra effort to ensure boundary consistency and maintain spatial orientation. This dissertation focuses three problems that pertain to drawing non-parallel contour networks and generating a segmentation surface from those networks.

First a guided structure-aligned segmentation system is detailed that utilizes prior structure knowledge from past segmentations of similar data. It employs a contouring protocol to aid in navigating the volume data and support using arbitrarily-oriented contouring planes placed to capture or follow the global structure shape. A user study is provided to test how

well novices perform segmentation using this system. The following two problems then aim to improve different aspects of this system. A new deformation approach to reconstruction is discussed which deforms previous segmentation meshes to fit protocol drawn contours from new data instances in order to obtain accurate segmentations that have the correct topology and general shape and preserves fine details. The focus is on the problem of finding a correspondence between a mesh and a set of contours describing a similar shape. And finally, a new robust algorithm that resolves inconsistencies in contour networks is detailed. Inconsistent contours are faster and less demanding to draw, and they allow the segmenter to focus on drawing boundaries and not maintaining consistency. However, inconsistency is detrimental to most reconstruction algorithms, so the network must be fixed as a post process after drawing.

Chapter 1

Introduction

Advances in 3D/4D imaging techniques in the past few decades have fundamentally impacted the process of biomedical discovery and clinical practice. Volumetric imaging produces spatial data which consists of nothing more than grayscale voxels. To reveal useful knowledge about the physiology, morphology, and mechanics of the imaged subject, this information needs to be processed. The data-to-knowledge pipeline usually starts with the delineation of the anatomical structure(s) of interest (e.g. an organ, a tissue, a cell, or a collection of these), a process commonly known as segmentation, in order to produce a 3D model of said structure(s). Studying the structure as a model, as opposed to the raw spatial data, can greatly increase the quality and pace of research [1]. Segmentations are increasingly critical in a wide range of applications such as treatment planning, quantitative analysis, virtual simulations, and rendering visualizations. Current segmentation methods range from automatic to manual which require varying amounts of human input and labor.

1.1 Automatic Segmentation

Automatic segmentations methods have an algorithm delineate the image while a human provides little input. The most naive automatic method is to choose an intensity value and threshold the image to find the boundaries (i.e. anything above the threshold value is inside the boundary). A variety of research in automatic segmentation is ongoing [53]. A bulk of these methods, such as Graph-cuts [9] and Random walks [19], automatically determines the entire boundary using the gradients of the image intensity values. While these methods are becoming more reliable, they are still susceptible to noise, poor image quality, and ambiguous tissue boundaries and require some form of human intervention. Figure 1.1 shows an example where these automatic methods will fail and only an expert familiar with this type of data will know where to place the boundary. Thus, many segmentations are still created manually to ensure the necessary quality and accuracy.

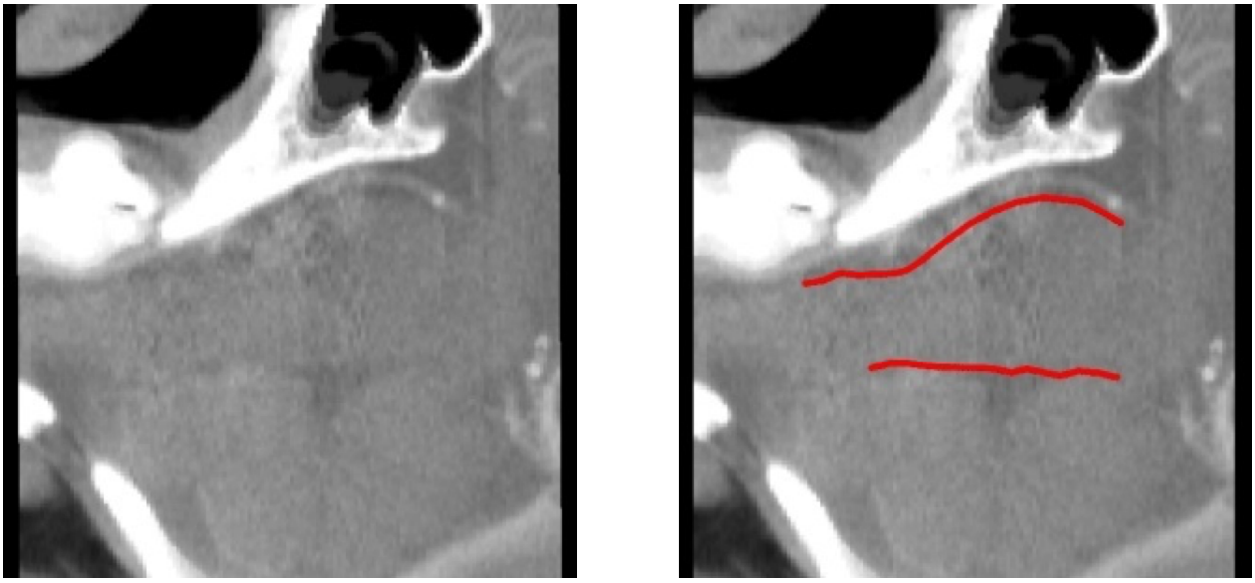

Figure 1.1: Example image data from a brainstem data set. The left shows the data, while the right shows a partially marked boundary and the rest is almost indiscernible.

1.2 Manual Segmentation

Manual segmentation is a timely process that can take hours, days, or weeks, to complete depending on the structure. It entails marking the boundary of the structure throughout the volume commonly done by drawing contours on multiple 2D slices, as seen in Figure 1.2. In standard practice, the slicing planes are either parallel along, or orthogonal to, the scanning direction of the image data, but can also be placed in any arbitrary orientation. Delineating enough contours to create an accurate segmentation can be both difficult and time intensive, depending on the shape of the structure and the size and quality of the image data.

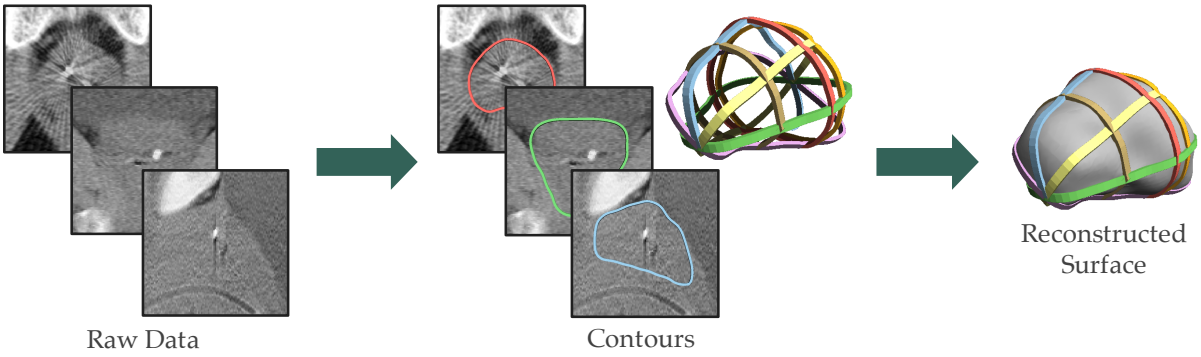

Figure 1.2: Example manual segmentation process in medical data. The right shows several 2D slices of the raw 3D image data, the middle shows the planar contours drawn on those slices and the complete 3D contour network comprising these contours, and the left shows the interpolated surface from those contours.

1.3 Overview

In this dissertation I focus on three problems that improve the two parts of manual segmentation: drawing contours, and reconstructing a surface from those contours. Often in clinical settings the segmenter repeatedly segments the same type of anatomical structure

such as multiple patients' aorta or a ferret brain over time, as seen in Figure 1.3. They are an expert in how to segment their specific structure and their past segmentations provide much knowledge of the structure's general shape and topology that largely go unused when segmenting a new data set. Therefore I also focus on using this prior structure knowledge to help guide these problems. The first problem introduces a new segmentation system, and the other two problems focus on improving different aspects of this system.

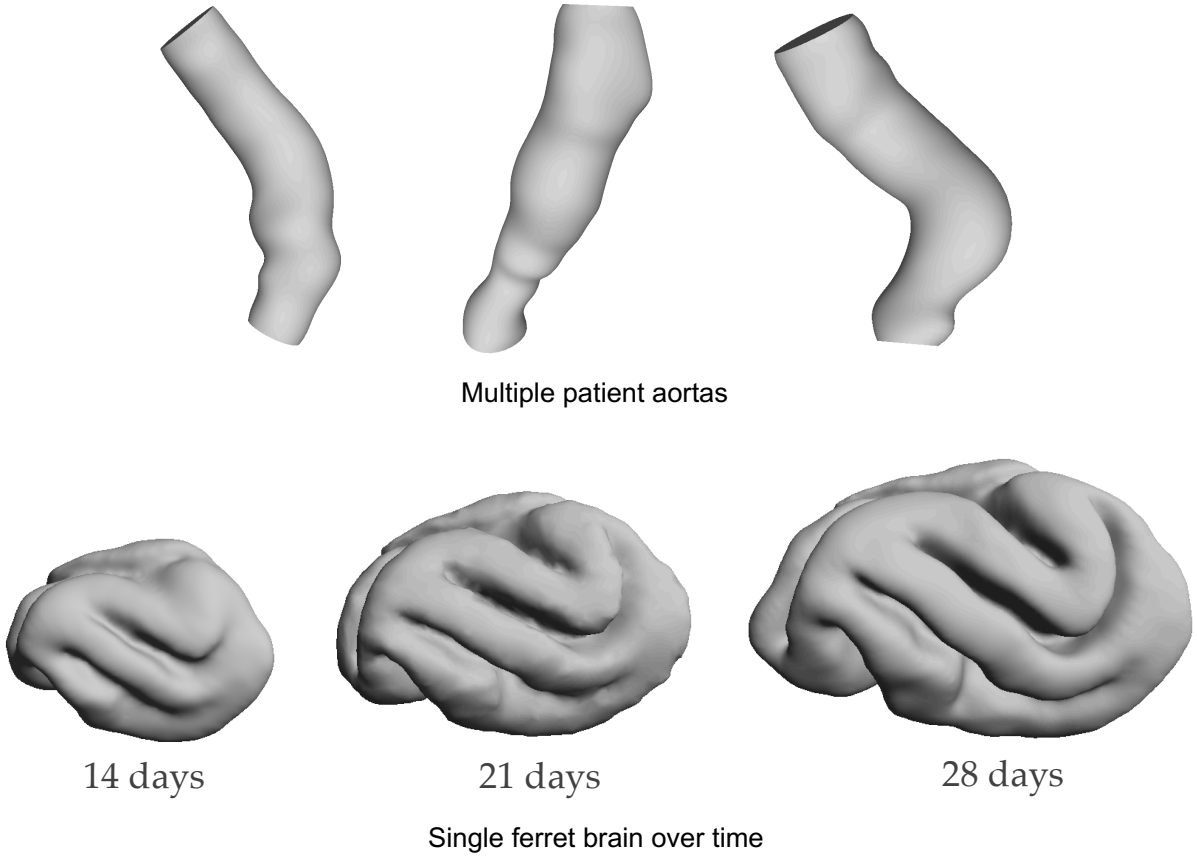

Figure 1.3: Top: Aorta segmentations from three different patients. Bottom: Ferret brain segmentations at different stages of development.

In Chapter 2 we will look at a guided approach to segmentation of volumetric medical data. In manual segmentation the key difficulties in contouring 3D volume data include navigating the data, determining where to place marks, and maintaining consistency between marks and

segmentations across similar data sets. Therefore I will leverage the similarity of a repeated segmentation task to address these difficulties and reduce the cognitive load for segmenting on non-traditional planes. I introduce the concept of a guided contouring protocol that provides step-by-step guidance in the form of an automatic navigation path to preselected arbitrarily-oriented contouring planes placed to capture the global structure shape, example contours from similar data sets, and text instructions. I present a user study that shows the usability of this approach with non-expert users in terms of segmentation accuracy, consistency, and efficiency. This work is published in [24].

In Chapter 3 I will explore a new approach to the problem of surface reconstruction from curves. Existing reconstruction methods focus on producing smooth interpolations and may require a large number of cross-sections to recreate feature-rich objects. The observation that anatomical structures usually share a common overall shape across subjects motivates taking a template-based approach that can better capture geometric features, instead of solving for the surface from the contour curves alone. I use prior segmentations whose shape represents the structure of interest as templates and deform these templates to fit a set of target contours. This approach can preserve topology and fine details even with a sparse set of contours. I describe my algorithm with the main focus on finding a correspondence between a mesh and a contour network. I demonstrate results with both synthetic and real medical data sets and compare to current reconstruction methods. This work is published in [23].

In Chapter 4 I address some issues with drawing non-parallel contour networks. A key difficulty in drawing intersecting contours is maintaining inside/outside consistency across all cross-section planes. With each new contour care must be taken to include existing intersection points with prior contours, and to not create new intersections with prior planes. It

is much easier to draw inconsistent contours which allows the focus to be on capturing the boundary and not on satisfying consistency. However, inconsistencies are detrimental to reconstruction algorithms, many of which can't even run with such input. Therefore, I develop a new robust algorithm for resolving inconsistencies in a set of non-parallel contours. I make use of implicit functions to represent the curve network and easily guarantee consistency across all planes. I test my algorithm on both synthetic and medical data to show how the results lead to better surface reconstructions.

Chapter 2

Guided Structure-Aligned Segmentation

2.1 Motivation

Revealing useful knowledge from 3D/4D biological imaging usually starts with a segmentation of the structure(s) of interest. Effective segmentations need to be accurate, consistent, and created efficiently. When segmentation is the bottleneck in the data-to-knowledge pipeline, it hinders research and could even result in health risks in clinical practice [16].

In many settings the same type of structure is repeatedly segmented (e.g. patient livers, mitochondria in cells, etc.). Depending on the use case, anywhere from a few to thousands of instances of the same structures must be segmented from an equal number of volumes, each of which is comprised of hundreds to thousands of scan slices. Aside from the sheer quantity of data, there are several key challenges in processing this data: the time required to produce a segmentation, how to navigate within the 3D image data, determining where to place marks, and maintaining consistency between marks and segmentations.

In this work I leverage the fact that the same general shape is segmented in each instance in order to define a *protocol* to guide segmentation of a specific class of structures. This protocol can help to reduce navigation time as well as encourage consistency both in how and where they place marks from one instance to another. This protocol is useful in the traditional parallel or orthogonal contouring approach, but more importantly it supports using arbitrarily-oriented contouring planes that can be placed to follow the global shape of the structure — for instance, tracking a tubular structure as it curves in space. Arbitrarily placed planes can offer more useful views of the data and capture known features more efficiently [52]. The primary finding of this work is that these structure-specific contouring planes (which reduces both time and error in the segmentation process) can be supported without unduly increasing the cognitive load over more traditional contouring methods.

2.2 Contributions

I propose the first stage of a new conceptual approach for working with volume data — guided 3D segmentation through structure aligned contouring protocols — and a novice friendly interface for evaluating this approach. The main contributions include:

1. A guided, structure-aligned approach to segmentation that reduces the cognitive burden of using non-traditional contouring planes.

2. A user study to show that this guided approach allows novices to produce segmentations of comparable quality to experts.

3. Identification of factors that influence segmentation time and quality, enabling comparison of different contouring protocols.

2.3 Related Work

2.3.1 Manual 3D Image Segmentation

Manually creating accurate segmentations in an efficient manner is a non-trivial task. Biological tissues are often not clearly separated, in addition to noise and lack of sufficient resolution and contrast in the image. Within volumetric imaging this is further compounded by the difficulties of maintaining orientation and structural awareness when navigating the 3D data. As a result, segmentation of biomedical data in practice is usually done only by domain experts who have undergone years, if not decades, of training in inspecting their specific imaging modality (e.g. MRI or CT) for their specific anatomical structures.

Traditionally, experts mark boundaries on parallel slices throughout the 3D volume. Many systems also allow marking contours on orthogonal intersecting slices, as generally less marking is needed and they can better capture surface curvature extrema. Orthogonal contouring has spawned various interactive systems that take user contour input and reconstruct a surface using splines or implicit functions [13, 22]. The work in [21] makes use of the live-wire technique for drawing contours on orthogonal planes and for interpolating between the contours. These methods, however, are all dependent on the user's ability to choose planes to contour on and do not provide guidance in this regard. Additionally methods that rely on the image data to detect boundaries may lead to undesirable results or require more user input when the boundaries are unclear.

2.3.2 Segmentation Guidance

The main difficulty in any interactive segmentation system is navigating and choosing where to mark boundaries. Thus new research has explored using guidance in segmentation systems. The work in [38] uses the random walks algorithm in a seed-based semi-automatic segmentation system where users place marks inside the structure instead of drawing boundary contours. The random walk probability is used to direct users to areas where more seeds could be placed.

Contour-based guidance approaches aim to automatically choose planes to mark contours on. In [48] the authors extend the work in [21] to arbitrary planes, and use the live-wire cost of the segmentation result to suggest planes that need marking. Similarly in [49] the authors have users provide drawn contours to seed the random walks algorithm, and in turn actively suggest new arbitrary planes with high uncertainty in the segmentation. Both of these methods require some initial contour(s) to find an initial segmentation to use as training data. There is little contouring consistency between similar data sets as the algorithms' choices depend on the training input and uncertainty probabilities.

These automatic plane selection methods are quite useful and my guided structure-aligned protocols fits nicely in tandem with them. My system doesn't require any initial user input for training since I instead define structure specific protocols that encode a preset path to automatically navigate to contouring planes. This structure specific path encourages consistency between similar data sets. My protocols could be used to provide the training data to these other methods that could fine tune segmentation accuracy. Experts could also use these plane selection methods to create the structure protocols used in this work.

2.4 Methods

2.4.1 Challenges

The two major challenges posed by manual volume segmentation are the cognitive strain of free-form data navigation and knowing where to draw contours. It is for these reasons that many segmentation systems limit the selection of planes to only parallel or orthogonal placement. However, arbitrarily placed planes can offer more useful views of the data that are aligned to the structure's shape and capture known features more efficiently [52]. My approach leverages prior knowledge of the structure's general shape in order to use arbitrary planes without increasing the segmenter's cognitive load to complete the task.

To address 3D free-form navigation I propose to remove the need for large-scale manual navigation. Navigation in 3D data is time consuming and inherently difficult to understand and perform, leading to loss of awareness of orientation and location in the volume. This is true for domain experts as well as novices who lack knowledge of the biological structures. My approach provides a navigation path to a relative set of structure-aligned planes to draw contours on for a given structure. Here "relative" means the planes shift according to the initial position of the first plane placed in the volume. Plane locations incorporate domain knowledge of structure shape, and they describe a set path that increases predictability from data set to data set. This work side steps the larger problem of choosing the best set of planes for a given structure and has an expert manually choose a reasonable structure-aligned sequence of planes.

In [42] it was shown that references images can substantially improve contour drawing consistency for novice users for a single, arbitrary plane. Thus, to address difficulties knowing

where and how to draw on a given plane, I provide visual cues in the form of images of example expert-drawn contours from similar data sets along with text instructions. This can help users fine-tune plane placement and decide where a boundary should be demarcated if more than one possible boundary is present, or if the boundary is noisy. Using reference visuals transforms a complex perceptual and cognitive task requiring domain expertise to, essentially, a pattern matching task that can be performed by novices. This allows us to use knowledge of the structure while still taking advantage of people's abilities to adapt to the data.

2.4.2 Contouring Protocol Definition

The *guided contouring protocol* incorporates a preset navigation path of arbitrary planes within the volume, images of example contours from similar data sets for each plane in the navigation path, and text instructions. It is essentially step-by-step instructions for drawing a set of contours that result in a segmentation surface for a specific type of structure.

The navigation path is defined as a set of successive 3D transformations between the chosen planes in a temporal order. For each step, the system automatically navigates to a plane by means of a smooth animated transition. Text instructions and a small set of expert-drawn example contours from a similar data set provide guidance for drawing the new contour. Additionally, users can make local plane navigation adjustments to better match the example images before drawing. A segmentation surface is constructed and progressively refined after the addition of each contour to provide immediate visual feedback. An example sequence of surfaces for each step of a protocol is shown in Figure 2.1. Surface reconstruction is done with [37] for its speed and reliability in producing a surface with partial or distorted input.

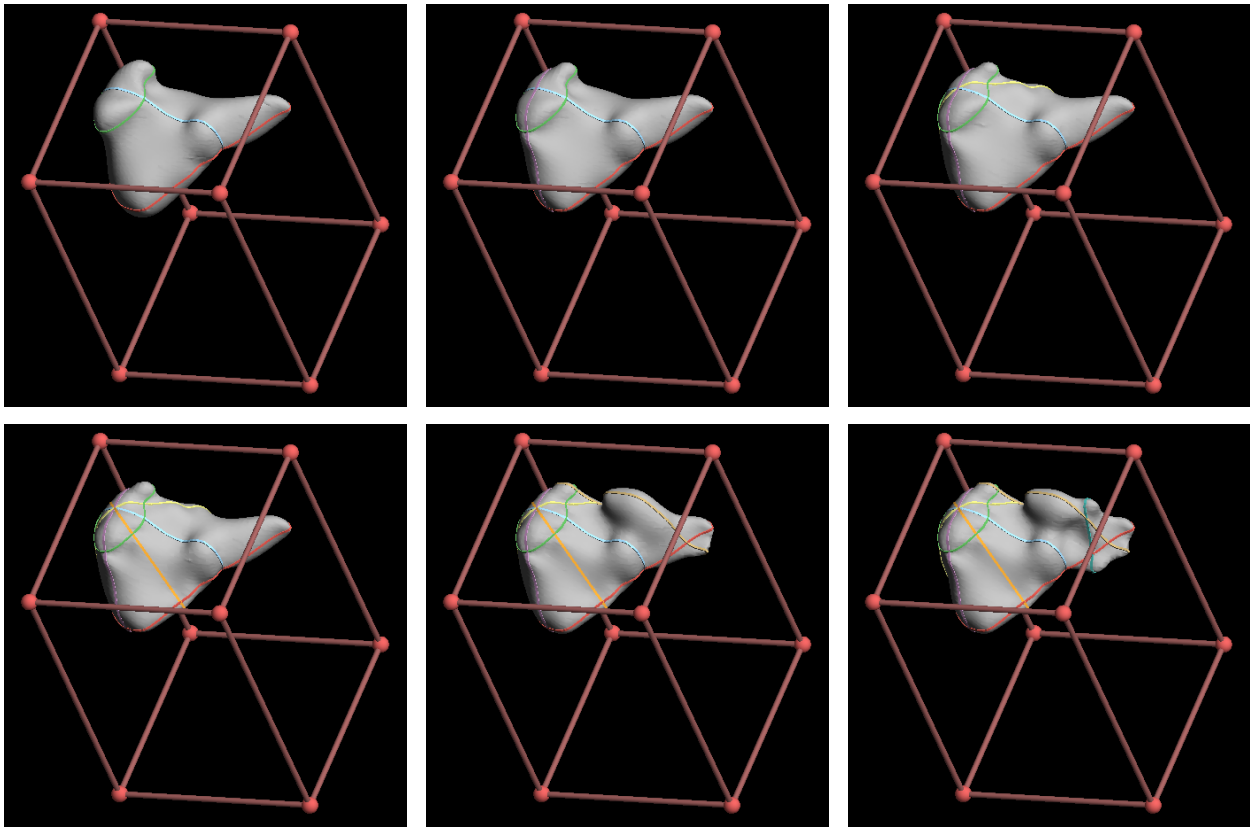

Figure 2.1: Example sequence showing the incremental updates of the segmentation as contours are added.

2.4.3 Interface

Contouring protocols could easily be implemented into any standard segmentation interface. For novice testing I built an interface, inspired by [42]'s design, that focused on supporting clear orientation and enhancing structural awareness, and exposing all functionality through in-screen elements to minimize required clicks. Fig. 2.2 depicts the four areas of the interface.

1) Main Window: The primary focus for interaction that shows the current slice, drawn contours, and if desired, the current segmentation surface. Includes standard navigation

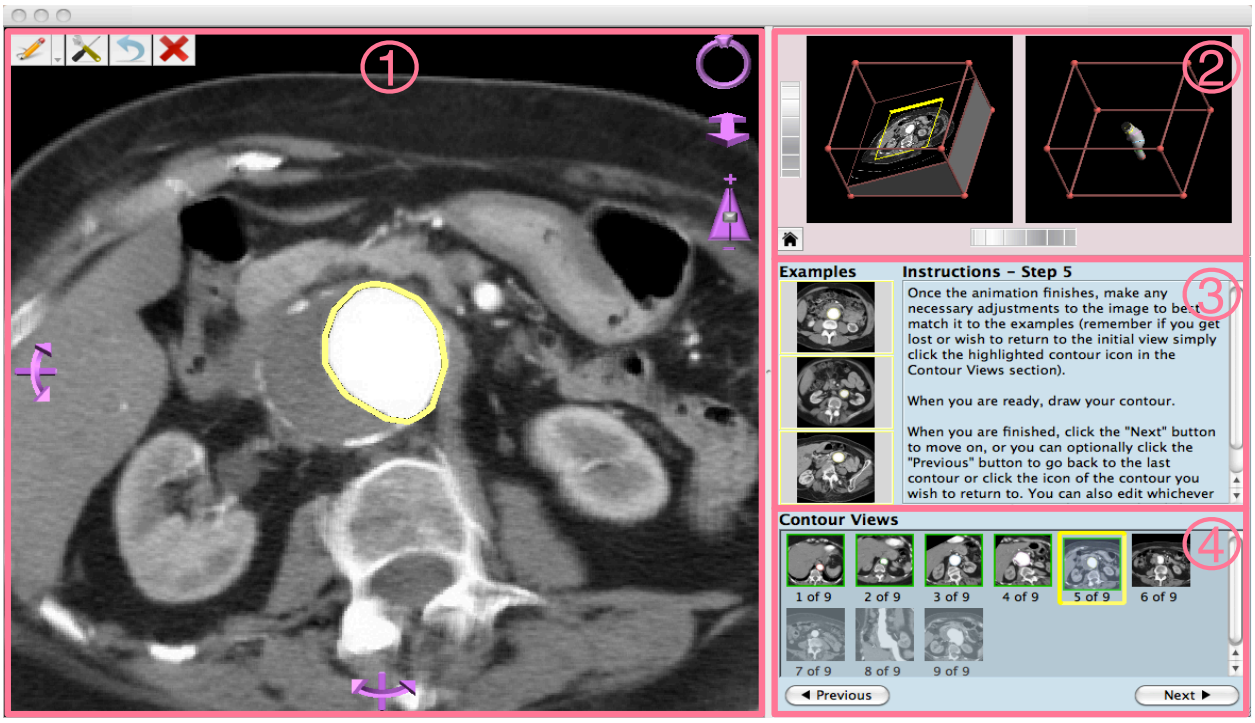

Figure 2.2: Interface layout. Area 1 is the main window, 2 is the 3D volume localization, 3 is the protocol instructions, and 4 is the navigation path visualization.

and free hand drawing tools. During drawing intersection points with other contours are displayed and the user must draw through them.

2) Localization: The pair of linked views visualizes the current slice's position in the volume (left) and the location of all drawn contours and the resulting segmentation surface (right). They allow the user to visualize where the current view in the main window is within the volume and in relation to the drawn contours and segmentation surface. These views are presented as YAH (You-Are-Here) cubes, inspired by [30, 32, 44, 20]. Turntable controls rotate the cubes providing users the freedom to find a useful view of the volume without losing global orientation.

3) Instructions: Provides both the image-based and text-based instructional information. Each of the example image icons can be clicked to show a large view of it in the main window. This gives the user the ability to toggle overlaying the full example image on top of the data they are trying to analyze.

4) Navigation Path: The navigation path is visualized as a sequence of thumbnails representing each plane/step in the contouring protocol. This was inspired by how key frame sequences are visualized in commercial tools such as YouTube™ and iMovie®. The user navigates through the path using either the next and previous buttons or by clicking on a thumbnail in the enforced order. The thumbnail for the current view/step is highlighted in yellow. The thumbnail for a completed step shows the slice view plus their drawn contour and is outlined in green; thumbnails for uncompleted steps are grayed out and filled with an example image. The navigation path helps users maintain a sense of orientation within the volume, and also provides a simple method for "scanning" through the volume. It also serves to show the user their progression through the protocol in terms of what steps are done, which step they are on, and how many steps are left.

Typical User Interaction

The guided contouring protocol begins with the user selecting the first plane for step 1. For ease of use the starting point is standardized as a plane in the center of the volume from which the user can freely navigate. The instructions and example images for step 1 provide sufficient detail for users to navigate to and select the first plane upon which to draw a contour. This only requires simple panning through the slices in the starting view direction (no rotations).

When they are ready to draw the contour, they click the pencil button to initiate drawing within the selected slice. While drawing, all global intra-plane navigation is disabled. After the user draws a closed contour, it becomes visible in the YAH cube, and global navigation is re-activated. If there are at least two contours then the working segmentation surface is constructed and shown in the YAH cube. They can now proceed to the next view in the navigation path where they, again, initiate the drawing process to create another closed contour. When drawing contours on intersecting planes, the system displays intersection points at the locations where prior contours pass through the current view, and the new contour must pass through these points to ensure consistency.

Upon completion of the last contour, the complete 3D segmentation surface is constructed and visible in the YAH cube. The "next" button changes to "finish" for the user to click to end the session.

2.5 User Study

To demonstrate that novices are able to complete a valid segmentation using the guided contouring protocol, I conducted a study with novice users and compared the results to expert segmentations. The main hypotheses tested are:

H1 Given sufficient guidance in the form of example images and predetermined contouring planes, novices can reliably produce valid segmentations for relatively complex structures.

H2 Using a set of structure-aligned planes increases segmentation accuracy and consistency compared to a set of parallel or orthogonal planes with the same amount of work.

H3 Total contouring time is dependent on contour length and curvature and the number of contours.

For H1, I compare novices both pairwise and to ground truth expert segmentations. For H2 I compare the accuracy, consistency, and completion time for three types of protocols used on a single dataset where each protocol requires approximately the same amount of drawing. For H3 I look at novice completion time and contour drawing times in contrast to their length and curvature. For mesh comparisons I use Dice's coefficient (DC) [14] and the mean percent distance (MPD), which is the mean distance [12] normalized by the size of the structure.

2.5.1 Study Design

Participants: Twenty volunteers were recruited from the Washington University in St. Louis community to segment three different data sets using structure-aligned protocols. Only 20% were moderately familiar with medical images and only 10% claimed to be moderately or highly experienced with image segmentation. Additionally eight volunteers were recruited from the Oregon State University community to segment one of the data sets using traditional parallel and orthogonal protocols. Only 1 of these users claimed to be experienced with image segmentation.

Image Data: The image data used were CT scans of a liver, an aorta, and a ferret brain. The selected data sets enabled sufficient evaluation of the approach to demonstrate validity. They range from simple (aorta) to complex (ferret) with differing image/contrast quality representative of data encountered in real segmentation tasks. Fig. 2.3 shows example images from each data set and an expert segmentation. The contours show the planes used

in the structure-aligned protocols. Note that only the right hemisphere of the ferret brain is segmented, and since the aorta is a tube users segmented slightly more than the expert and then the results were clipped.

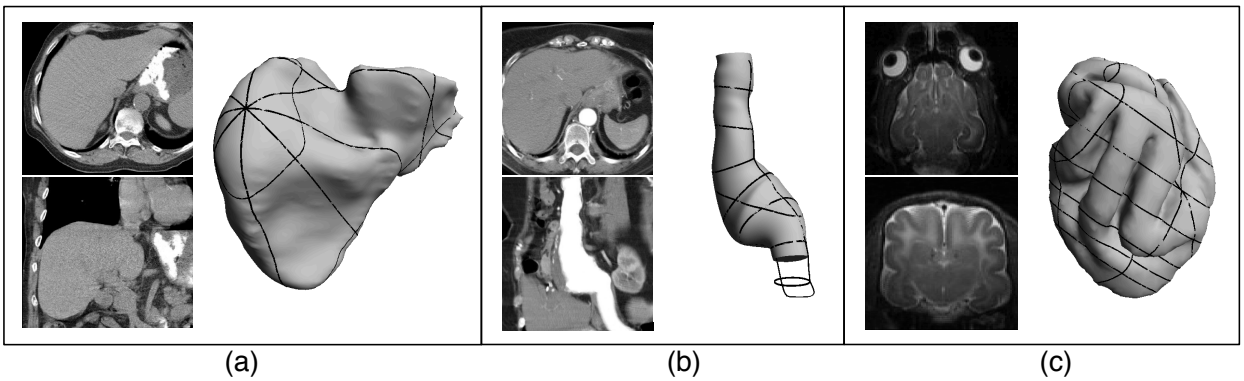

Figure 2.3: Example images, structure aligned contours, and segmentations from the three data sets: (a) liver (b) aorta (c) ferret brain.

The liver data is 512x512x512 voxels, the aorta data is 512x512x256 voxels, and the ferret brain data is 128x128x128 voxels. For the example images in the protocols, three example images were provided for each step of the liver and aorta protocols, and two example images were provided for each step of the ferret brain protocol.

Set Up: All participants used identical workstations and contouring tools. The entire interface was maximized on a single 15" MacBook Pro® screen, and they were provided with an Apple mouse.

Training: Training was designed to help the user understand 3D volumes, the guided contouring process, and to familiarize themselves with the interface features. Each user received a brief verbal introduction to the study including the task at hand and simplified descriptions of 3D volumes and segmentation. Next they explored the interface, using a data set not included in the main tasks, to let them practice using all of the features and to ask questions. During this session all interface features were tracked as to make sure they explored

each feature and understood it. Users were prompted with scripted questions for elements they did not directly explore to ensure understanding. Training lasted, on average, 15 to 20 minutes and all participants completed training in under 30 minutes. A limit was not set in order to ensure all participants obtained exposure to all interface features.

Procedure: The first twenty users used structure-aligned protocols to segment the three data sets in a random order. Random order was determined pseudo-randomly such that the six possible orderings to see the data in were evenly represented across the users. The number of protocol planes were eight for the liver, nine for the aorta, and ten for the ferret brain. The eight additional users segmented only the liver using both a parallel and orthogonal protocol in a random order consisting of seven and eight planes respectively. The planes for the parallel and orthogonal protocols were chosen such that they required about the same total amount of drawing as the structure-aligned protocol.

Users were allowed to take short breaks between each task if needed. No user failed to complete each of the segmentation tasks. A questionnaire was administered at the completion of the tasks to gauge user background and opinions on the task.

2.5.2 Results

The following results will look at the accuracy, consistency, and efficiency of the novices that used the system for the three different data sets. Expert consistency and efficiency data obtained in [43] for the liver data set is provided for reference. Seven experts contoured the eight protocol planes using the interface in [42].

Accuracy:

The accuracy of novice segmentations is evaluated by comparing to a ground truth expert segmentation for each structure (Fig. 2.4 Top). The liver ground truth is an average of the expert segmentations, and the aorta and ferret brain ground truths were provided to us with the data.

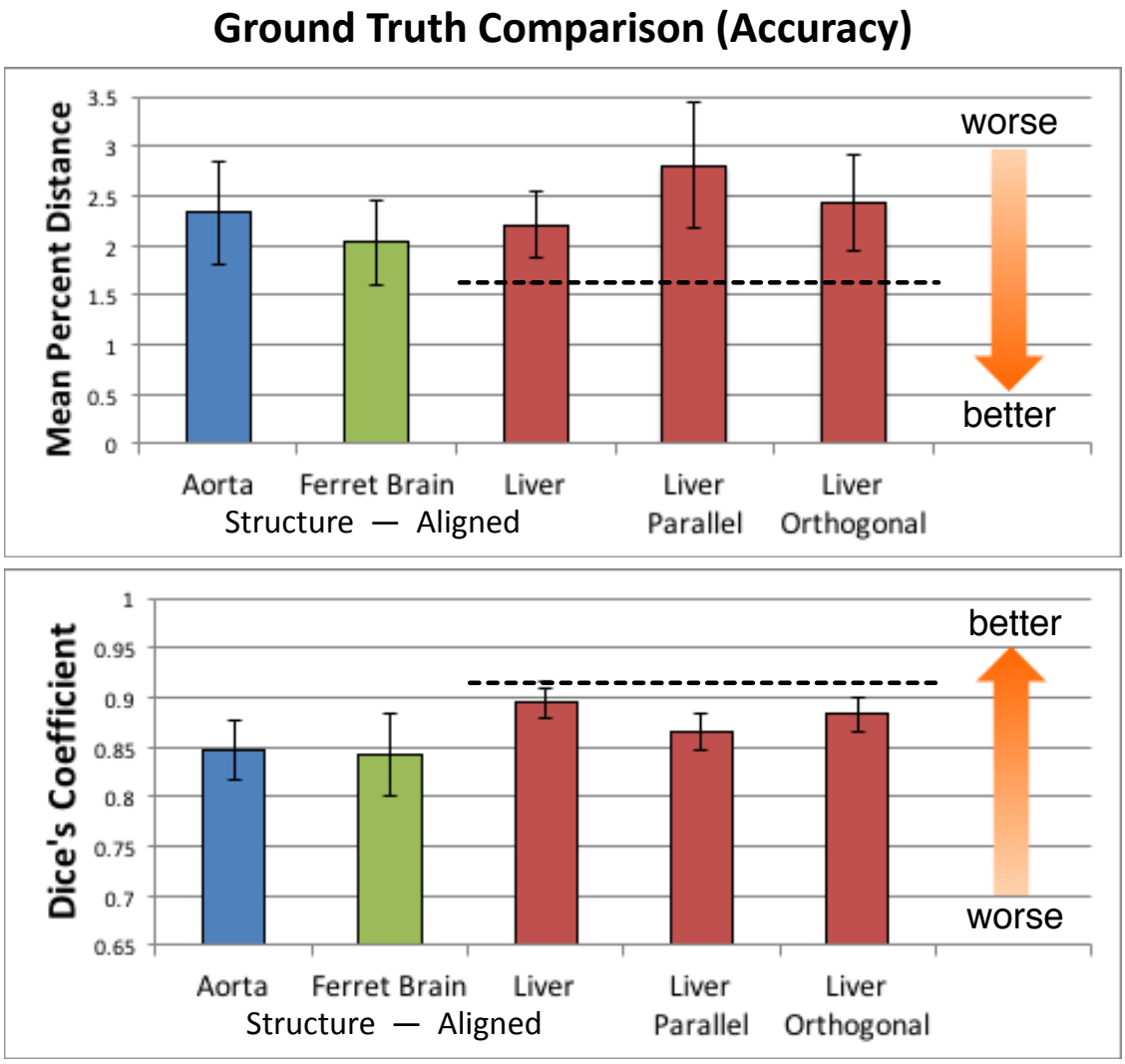

Figure 2.4: The average MPD and DC between novice segmentations and the ground truth. The dashed line indicates the average expert consistency (i.e. the variance among experts).

Users were able to construct reasonably accurate segmentations for the three structure-aligned data sets. The ferret brain was least accurate due to its complexity. Much of the expert segmentation boundaries don't fall on strong gradients as shown with the solid line in Fig. 2.5. Despite the example images, many users tended to follow stronger boundaries, shown with the dotted lines. The variance in the aorta was because some contouring planes had a second visible boundary that some chose to follow, as shown in Fig. 2.5. For the liver we can see that the novice accuracy is close to the variance seen among expert liver segmentations. These results indicate that using a guided contouring protocol, novice users are capable of producing segmentations of comparable quality to expert segmentations, confirming H1.

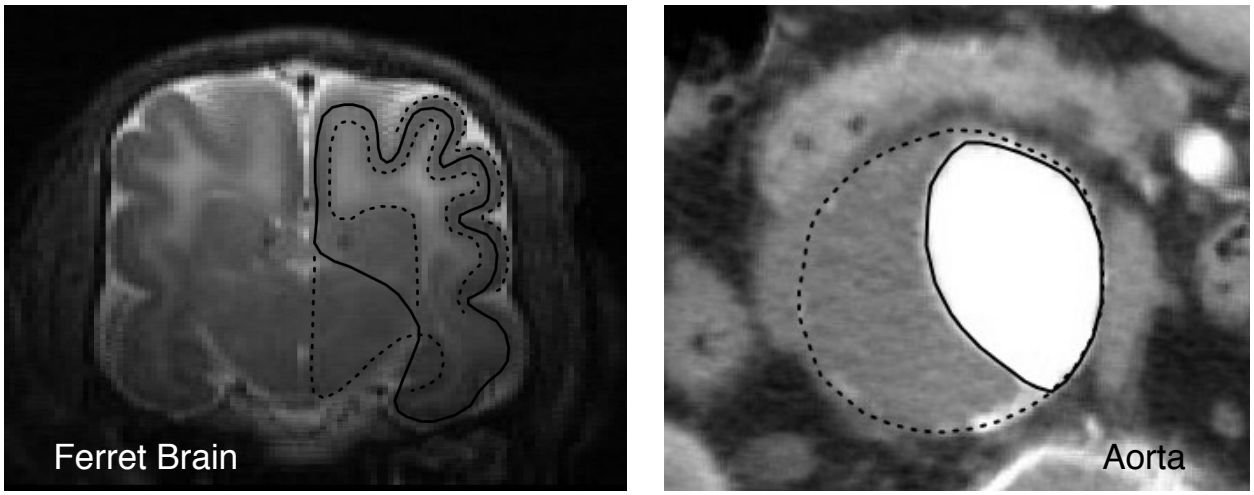

Figure 2.5: Examples of ambiguous boundaries. The solid outline is the expert segmentation, and the dotted lines show boundaries that some novices followed instead.

For the liver cases, the structure-aligned protocol produced a significantly (T-Tests MPD: $p=0.00007 \leq 0.05$, DC: $p=0.00016 \leq 0.05$) more accurate segmentation compared to the parallel protocol confirming H2. And while the structure-aligned results were slightly more accurate than the orthogonal, they were not significantly so (T-Tests MPD: $p=0.1015 \geq 0.05$,DC: $p=0.0836 \geq 0.05$).

Consistency:

The consistency of the novice segmentations is evaluated by performing pairwise comparisons among all users for each protocol (Fig. 2.6 Bottom).

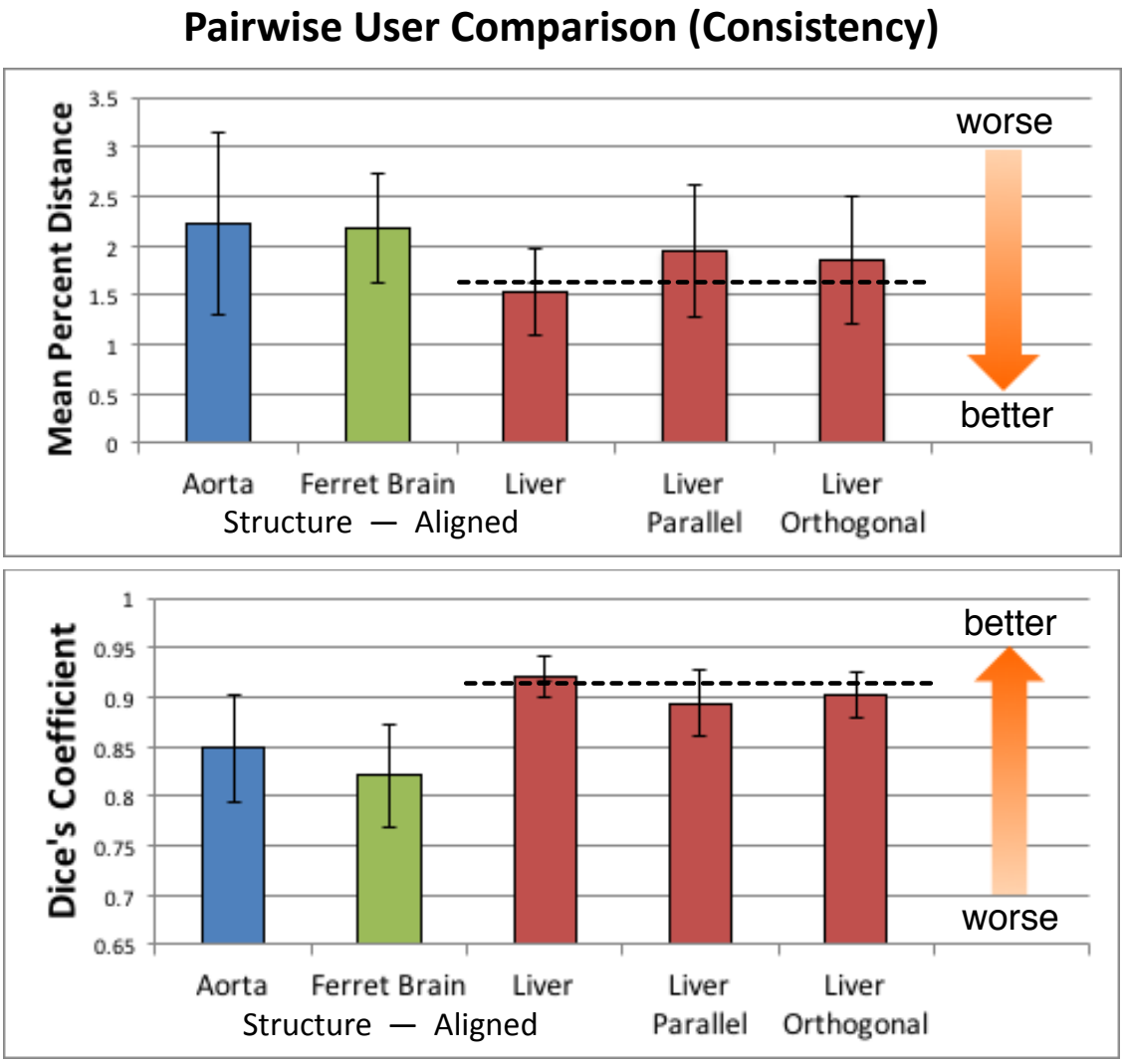

Figure 2.6: The average MPD and DC between all pairwise novice segmentations. The dashed line indicates the average expert consistency (i.e. the variance among experts).

Users were reasonably consistent for the three structure-aligned cases. The ferret brain and aorta had more variation for the same reasons discussed in the accuracy section. Looking

at the dashed line, the novice liver segmentations were as consistent as the experts, which shows that novice users can reliably produce similar segmentations using these protocols, confirming H1. The protocol approach encourages consistency because it enforces a defined contouring scheme so that segmentations are performed in a relatively uniform manner.

For the liver cases, the structure-aligned protocol produced significantly more consistent segmentations than both the parallel (T-Tests MPD: p=0.00001 $\leq$ 0.05, DC: p=1.19144$E-8 \leq$ 0.05) and orthogonal (T-Tests MPD: p=0.00024 $\leq$ 0.05 ,DC: p=0.00005 $\leq$ 0.05) protocols, confirming H2.

Efficiency:

The efficiency of guided contouring protocols is evaluated by computing the average completion time for each protocol. I also examine contouring time in terms of contour length and curvature.

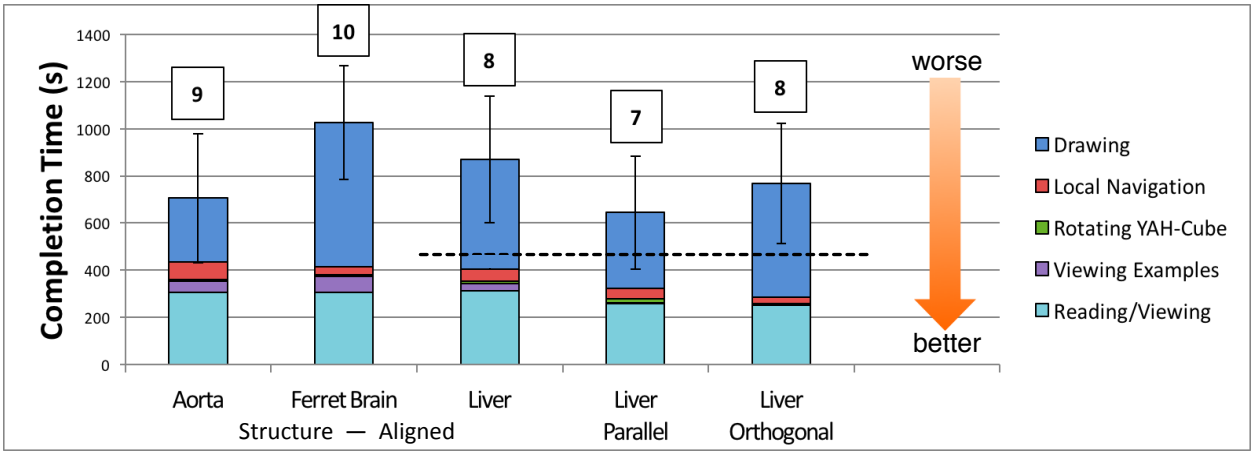

Figure 2.7: Average completion time broken down by user actions. The number of protocol planes is shown above each bar. The dashed line indicates average expert contouring time.

The average completion time for each protocol was between 10-15 minutes (Fig. 2.7) with the exception of the ferret brain due to its complexity. For the liver, novice users on average took less than double the expert time. I feel that these results are compelling considering the users had little to no prior experience. Using the protocols the segmentation time is firstly dominated by time spent drawing contours and secondly by time spent idling (i.e. reading/observing), confirming H3; little time is used for manual navigation.

For the liver cases, the completion time for the three protocols were very similar since they were designed to have approximately the same amount of drawing. The parallel case was slightly significantly faster than the structure-aligned (T-Test: p=0.04914 $\leq$ 0.05) but it was also less accurate, meaning that more parallel contours are needed which would increase the time.

Because segmentation time is now mostly dependent on the time taken to draw contours, I consider contour length and curvature, and their influence on drawing time. In Fig. 2.8 the general trend shows that as the length or curvature increases so does the drawing time, confirming H3. The aorta contours all have similar length and curvature, so we see them clustered together in both plots. The liver contours have similar curvature but vary in length. The complex ferret brain contours varied fairly equally on both factors.

From these results, the approximate time, T, needed to draw n contours can be represented as a function of the contours' length and curvature, plus an uncertainty factor, as shown in Equation 2.1 below.

$$T = \sum_{i=1}^{n} \alpha L_i + \beta C_i + U_i \tag{2.1}$$

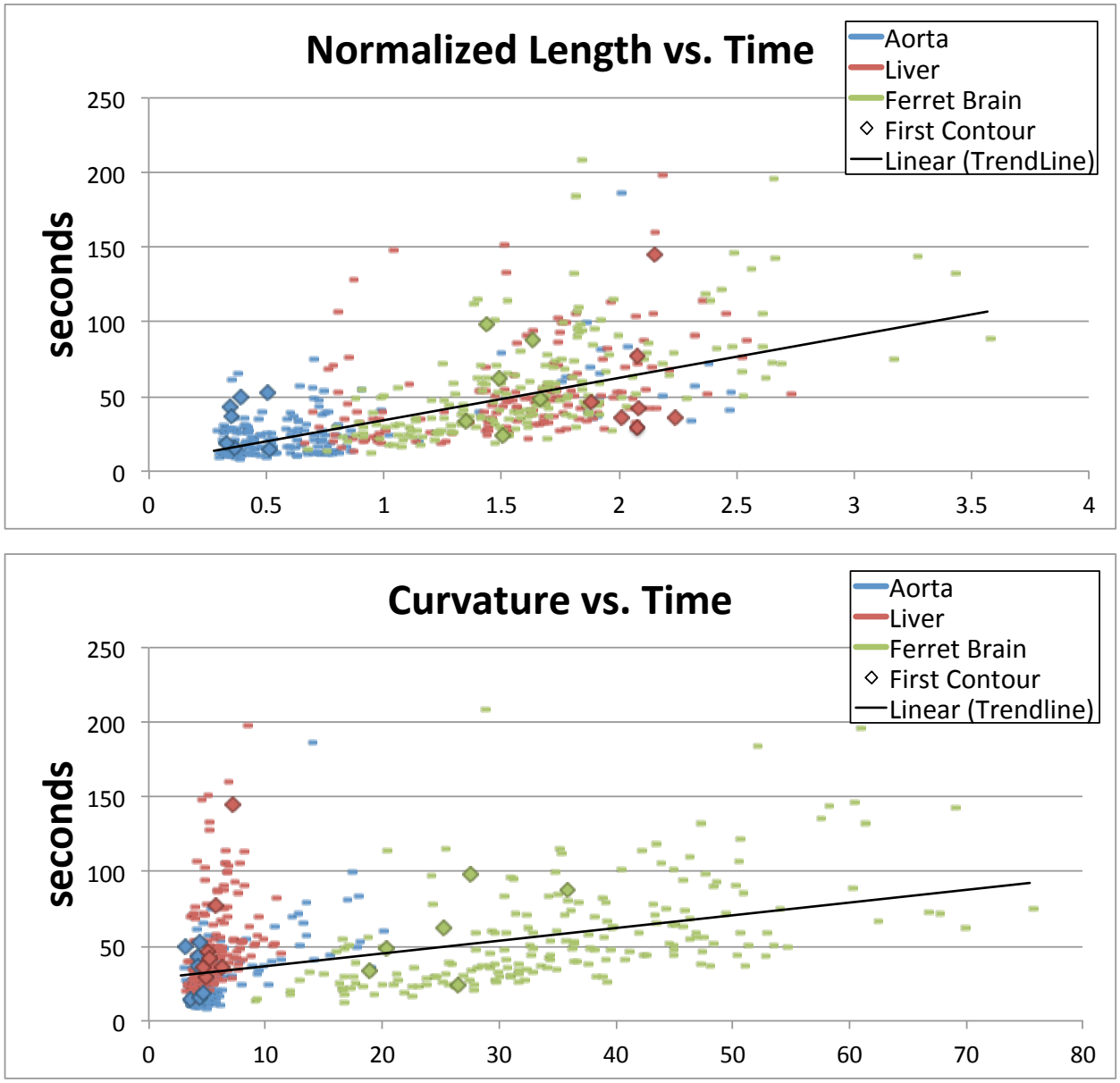

Figure 2.8: Scatter plots of contour length and curvature versus drawing time. The contour length was normalized to a unit square drawing window. The curvature for each contour was measured as the sum of the absolute curvature across all contour points.

For each contour, i, L_i is its length, C_i is its curvature, and U_i is an estimate of the time needed for its uncertainty. The curve length is normalized by mapping the visible window to a unit square, and the curvature is the sum of the absolute curvature across the contour. I fit a linear polynomial surface to the novice 3D data points of contour length vs. curvature vs. time (with outliers removed) to find α as 23.85 and β as 0.3135 for the data.

User Feedback

The majority of users reported the tasks as easy to complete, with only 25% reporting it as moderately difficult. The majority of users also felt they were fairly successful at performing the segmentation tasks. This shows that not only was the guided approach easy enough for novices to attempt, it also made them feel confident in what they produced. Most users even thought the task was fun and engaging while only 15% felt otherwise.

The most notable reported difficulty was not knowing how to start, especially if they ignored the text instructions. They were unsure of exactly what to do at first, and a few had difficulties with the first step since it required them to pick the first plane. Other difficulties included not knowing what to draw if the examples didn't exactly correspond to what they saw or the boundaries were noisy, and also not knowing how to draw the contour with the existing intersection points from other contours.

The majority of users thought that the example images were the most helpful feature in completing the contour drawing task. Approximately 50% of users reported that the YAH cubes were helpful for the contour drawing task, noting their utility for visual context, but not necessarily for drawing contours. Some users also noted a desire for seeing larger versions of the YAH cubes.

2.6 Discussion

I have described a new method for performing segmentations for structures of known shape. I have shown that the guided contouring protocol is usable with any level of proficiency and enables efficient production of segmentations that are accurate and consistent.

Segmentation quality is defined by what the domain expert deems as useful. This notion drives the usefulness and flexibility of expert designed protocols that is more powerful than an arbitrary "gold standard". Future work can develop tools for protocol creation with automatic plane selection to aid in choosing the best structure-aligned planes. Since segmentation accuracy is dependent on the planes chosen to draw contours, experts need to be able to easily choose planes for a protocol that capture important structure features.

In the next two chapters I will address two problems that can improve this system. First, in addition to plane placement, the number of planes contoured and the reconstruction method used also affect segmentation accuracy. Ideally we want to use as few contours as possible, but that also means less details when using an interpolation based reconstruction. While the novices in this study drew good contours their accuracy was limited to the capabilities of the reconstruction method. Therefore I will examine a new reconstruction approach that utilizes prior structure knowledge to guarantee the correct general shape and topology and preserve fine details while using a sparse set of contours.

Secondly, many users reported it difficult the draw contours when they are forced to incorporate intersection points with previous contours. This was especially true for the ferret brain when there were often several on one plane and the shape was complex. They couldn't figure out which order to connect them in and ended up focusing on trying to connect all the dots instead of drawing the boundary. Removing the requirement to use intersection

points will result in inconsistent contour networks which are problematic to reconstruction methods. Therefore I will detail a new robust algorithm for resolving these inconsistencies to make drawing less burdensome.

Chapter 3

Template-based Surface Reconstruction from Cross-sections

3.1 Motivation

In the last chapter I detailed a guided segmentation system for drawing a set of planar contours around the boundary of an object in a 3D image as seen in Figure 3.1(a). The segmentation can then be found by reconstructing a surface from the contours using methods such as [37, 55]. However, the accuracy of the segmentation is dependent on the number of contours drawn and the reconstruction method used as seen in Figure 3.1(b) and (c).

Reconstructing a surface from planar cross-sections is a well studied problem in geometric modeling, and a variety of methods have been developed. Most of them treat reconstruction as an interpolation problem, and the shape of the surface is governed by a certain smoothness energy. While these methods can reasonably recreate the shape of smooth objects from a small number of contours, they have inherent difficulty in recreating objects that are rich in geometric features if such features are not completely captured by the contours. This is

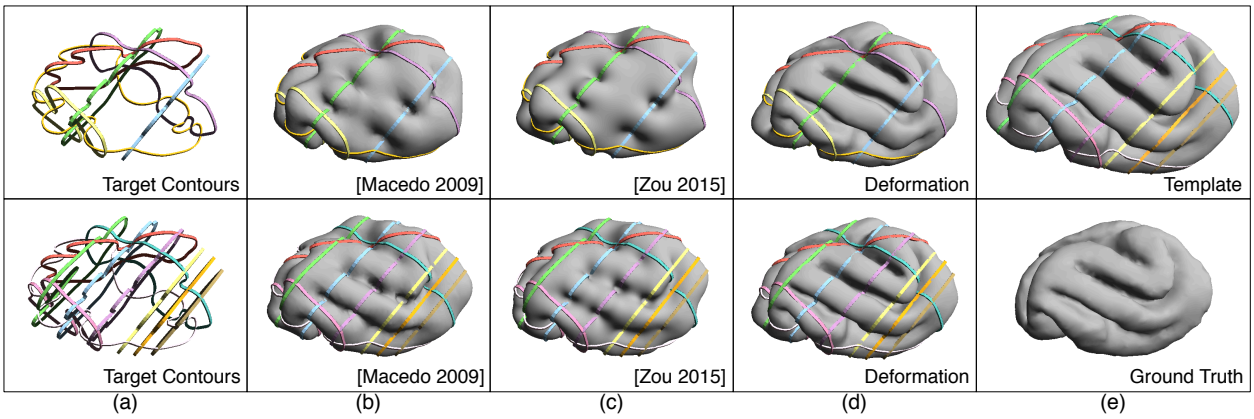

Figure 3.1: Reconstructing a ferret brain from two sets of cross-section contours (a), containing six (top) and ten (bottom) contours respectively: when compared with existing interpolation-based methods such as [37] (b) and [55] (utilizing the underlying image volume) (c), my template-based method (d) does a much better job at recovering prominent geometric features of the brain (e.g., the grooves) even with a sparse set of contours. My method utilizes a template equipped with a set of source planes (see (e) top). The template for ten source planes case is shown and a subset of these planes is used for the six planes case. The ground truth shape is shown in (e) bottom.

highlighted in the ferret brain example in Figure 3.1: given just a few contours (a), existing methods create interpolating surfaces (b,c) that, although being smooth, fail to capture the "grooves" of the brain surface (e). Adding additional contours only yield small improvement in the reconstructed features, implying that a significant number of contours would be needed to achieve a satisfactory reconstruction.

In this work I develop a reconstruction approach that can faithfully reproduce geometric features without requiring an excessive amount of contours. I focus on the application setting of interactive medical image segmentation, and make two observations in this context. First, due to human bias and inherent ambiguities in images, the contours only approximate the actual object boundary, and hence exact interpolation is not necessary. Second, and more importantly, despite the variation among subjects, anatomical structures usually share a common overall shape between individuals in a group (e.g., those at the same developmental

stage). These two observations motivate a new approach to reconstruction: instead of building surfaces directly from the contours, *deform* an existing prior segmentation surface (called the *template*), whose shape is representative of the anatomical structures in the group, to fit the contours (called the *target contours* henceforth).

The key challenge in this template-based approach is establishing the mapping between the template and the target contours. This is a highly ill-posed problem, since the 1-dimensional contour curves carry very little geometric information to constrain its mapping to the surface. To make the problem more tractable, I will make use the contouring protocols from Chapter 2 which use pre-defined planes that are aligned with the shape and features of the structure. The template surface is then assumed to be equipped with the cross-section planes (called *source planes*) from the protocol. The target contours are then drawn on planes (called *target planes*) in a new data set using the same contouring protocol. This way the source and target planes are in one-to-one correspondence, where the position and orientation of each source plane will be similar to the position on orientation of the corresponding target plane (only similar because the segmenter can make adjustments to the placement of the planes). These source planes therefore serve as a constraint on how the target contours are mapped onto the template surface. An example template with source planes is shown in Figure 3.1 (e) top.

The geometric problem I will address here can be stated as follows: given a template, a set of possibly intersecting source planes, and a corresponding set of target planes containing target contours, deform the template so that the neighborhood of the source planes on the template fit the target contours. Note that the cross-sections of the template at the source planes cannot fit the target contours, since the network topology of the target contours can be different from the cross-sections of the template at the source planes.

I solve the problem using the classical alternative optimization approach that alternates between computing the deformation and finding the mapping. The key step in this method is mapping a possibly complicated network of target contours onto the template surface, so that the mapped network preserves the shape of the target network while staying close to the source planes. This is achieved by a novel algorithm that combines dynamic programming with a greedy combinatorial optimization.

I test my method on the real-world data sets used in the last chapter (namely the ferret brain and the liver). The results demonstrated significantly improved geometric features over traditional, interpolation-based approaches for surface reconstruction from curves, even with only a few target contours (e.g., Figure 3.1 (d)).

3.2 Contributions

To the best of my knowledge, I propose the first template-based reconstruction method from spatial curves, and make the following technical contributions:

1. Formulate the problem of template-based reconstruction from cross-section curves in the context of biomedical applications (Section 3.4), and make the problem tractable by leveraging the availability of source planes and their correspondences with the supporting planes of the cross-section curves.

2. Propose an alternating optimization solution for the above-mentioned problem (Section 3.5), and in particular, introduce a novel algorithm for mapping a network of spatial curves onto a surface.

3.3 Related Work

3.3.1 Surface Reconstruction from Curves

Building surfaces from cross-section contours has been extensively studied in the graphics community since the 70s [29, 17]. I briefly summarize the works in this area, and refer readers to more detailed reviews in recent papers such as [5, 55].

Most methods for cross-section reconstruction fall into three general classes. The first class is based on the Delaunay triangulation of cross-section curves and/or planes [6, 10, 7]. The second class obtains the surface by projecting the curves onto a set of auxiliary sheets (e.g., medial axis or straight skeleton of a cell in the plane arrangement) [2, 3, 4, 28, 36]. The last class defines and extracts the iso-surface of an implicit function, such as radial basis functions [50, 37], mean-value interpolants [5], and harmonic functions [55]. While earlier methods focus on the simple input setting of a stack of parallel cross-sections containing disjoint contour loops, recent methods can handle non-parallel cross-sections and contour networks (for multiply-labelled domains). More recently, the topology of the output surface can also be controlled [55].

Although capable of generating closed and interpolating surfaces, existing methods all rely on some smoothness models to constrain the *shape* of the surface that connects the input curves. Reconstruction methods based on implicit functions rely on that function's underlying smoothness model. For example, radial basis functions [50, 37] minimize the thin-spline energy. Other reconstruction methods typically require a post-process fairing on the initial mesh surface, and the shape of the final surface depend on the fairing technique (e.g., [36, 55]

uses surface-diffusion flow [54]). Note that these smoothness models inherently prevent existing methods from creating interesting geometric features that are not well characterized by the curves themselves (e.g., the grooves in the Ferret brain in Figure 3.1).

3.3.2 Template-based Reconstruction

Templates are often used for surface reconstruction from incomplete data, in particular point clouds (see the survey article [45]). Template deformation is often formulated as a minimization problem where the objective function consists of a fitting term (i.e., how well the template fits the data) and a shape term (i.e., how much distortion is introduced to the template shape). For non-rigid deformations, a variety of ways exist to formulate the shape term, such as using differential coordinates [40, 41], local affine transformations [34], thin-plate spline [11], and statistical shape models [25].

Formulating the fitting terms requires establishing the mapping between the template and the data. Correspondences between two (complete or partial) surfaces have been well-studied (see the survey article [51]). The simplest approach is based on proximity in the Euclidean space, as used in the classical ICP (Iterative Closest Point), but the application is limited to templates in close range of the input data. To deal with large deformations, additional constraints have been adopted to regularize the mapping. Common constraints are based on either surface-based geometric features, such as spin images [27], integral descriptors [18], and SHOT features [47], or the deformation space of the surface, such as isometry [26, 46] and conformality [35, 31]. However, these constraints are designed for input data with a 2-dimensional structure (e.g., point cloud covering a surface area). To the best of my

knowledge, no correspondence methods have been developed that map between a template surface and a target spatial curve network.

3.4 Problem Formulation

The input is a template mesh M with vertices $\{v_1, \ldots, v_n\}$ and target contours C representing the cross-sections of an object O on target planes $B = \{b_1, \ldots, b_l\}$. The target planes can be non-parallel with each other, and hence C in general is a curve network composed of planar *segments* meeting at *junctions*. The vertices of C are denoted as $\{p_1, \ldots, p_m\}$. Each vertex is associated with one or multiple planes of B. To further constrain the problem, assume the knowledge of source planes $A = \{a_1, \ldots, a_l\}$, such that the position and orientation of each a_i $(i = 1, \ldots, l)$ with respect to M is similar to those of the corresponding target plane b_i with respect to O (Figure 3.2 far-left).

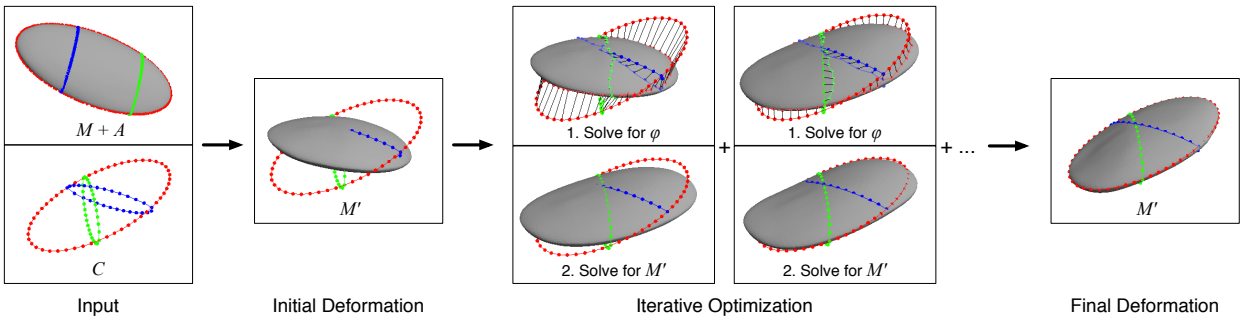

Figure 3.2: Work flow of my algorithm. The input includes a template mesh M, equipped with source planes A, and target contours C lying on target planes B. Note that C can have a different network topology than the intersection of M with A. After computing an initial deformation M', repeatedly alternate between solving for the mapping ϕ while fixing M' and updating the deformation M' given ϕ for a fixed number of iterations.

The goal is to compute a deformed template M' with new vertex locations $\{v'_1, \ldots, v'_n\}$, so that a neighborhood of the source planes on the un-deformed template fits the target contours

after deformation. To formulate the problem, a mapping $\phi : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ that maps each target contour vertex p_i to some template vertex $v_{\phi(i)}$ must also be found. Intuitively, the deformed vertices $v'_{\phi(i)}$ should be as close to p_i as possible.

It is important to not limit the image of ϕ to be only vertices of the template that are *on* the source planes. Since the network topology of C can be different from the topology of the curve network obtained by intersecting M with source planes A, a mapping from C to the latter network may not exist. In the example of Figure 3.2 far-left, the intersection between M and A is a network with 4 junctions and 8 segments, while the target contours C contains 6 junctions and 12 segments. The other choice is computing a separate mapping for each target plane b_i that maps the target contours on b_i to the points on the corresponding source plane a_i. However, when the topology of C differs from that of the intersection of M and A, these per-plane mappings may disagree at a global level and create highly distorted deformations (see Results section).

3.4.1 Energy

Formally, I seek the pair $\{M', \phi\}$ that minimizes the following 3-part energy:

$$E(M', \phi) = w_{fit}E_{fit}(M', \phi) + w_{dis}E_{dis}(M') + w_{map}E_{map}(\phi) \tag{3.1}$$

The three energy terms are explained as follows:

- The fitting term, E_{fit}, measures the deviation of the target contour vertices from their mapped locations on the deformed template:

$$E_{fit}(M', \phi) = \sum_{i=1}^{m} \|p_i - v'_{\phi(i)}\|^2 \tag{3.2}$$

- The distortion term, E_{dis}, measures the distortion of the template shape due to deformation. The literature has suggested many ways to quantify such distortion. For computational efficiency, I adopt the linear formulation in [41] that approximately measures non-rigid motions,

$$E_{dis}(M') = \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} \|(v'_i - v'_j) - R_i(v_i - v_j)\|^2 \tag{3.3}$$

  where $N(i)$ denotes the indices of the 1-ring neighbors of vertex v_i, R_i is an estimated local rotation matrix that best aligns v_i and its 1-ring neighborhood with v'_i and its 1-ring neighborhood, and w_{ij} is the cotangent weight. Note that replacing this term with alternative choices of distortion measures (e.g., discrete Laplacian [40] or local affine transformations [34]) will not affect the rest of the algorithm.

- The mapping term, E_{map}, measures the irregularity of the mapping ϕ. Intuitively, it favors mappings whose image on the template is a curve network that lies in the neighborhood of the source planes and bears similar shape as the target network C. To measure dissimilarity in curve shapes, I adopt the first-order differences that are commonly used for matching planar curves [15]. In this case, the differences are computed after transforming the target planes to align with the source planes. My definition of E_{map} captures these two goals, deviation from source planes and shape dissimilarity,

as follows:

$$E_{map}(\phi) = w_{src} \sum_{i=1}^{m} \sum_{j \in H(i)} d(a_j, v_{\phi(i)}) + \sum_{\{i,j\} \in E(C)} \|T_{H(i,j)}(p_i - p_j) - (v_{\phi(i)} - v_{\phi(j)})\|^2 \quad (3.4)$$

Here, w_{src} is a balancing weight, $d(a, v)$ measures the shortest geodesic distance from point v to any point on the cross-section of the template at plane a, $H(i)$ gives the list of indices of the target planes that contain the contour vertex p_i, $H(i, j)$ is the unique index of the target plane that contains two edge-adjacent contour vertices p_i, p_j (assume the generic situation that no edge of the target contours lies on the intersection of two target planes), $E(C)$ is the list of polygon edges on C, and T_i is the transformation that aligns target plane b_i with the source plane a_i. T_i is obtained by first translating the centroid of those contour vertices on b_i to the centroid of the intersection of M and a_i and then rotating b_i around the cross-product of the normal vectors of a_i and b_i.

The scalars $w_{fit}, w_{dis}, w_{map}$ are balancing weights of the three energy terms. I use the setting $w_{dis} = 10, w_{map} = 1, w_{src} = 0.25$ for all the experiments and justify the choice of these values with examples in Section 3.6.

3.5 Optimization

3.5.1 Overview

Minimizing the energy $E(M', \phi)$ (Equation 3.1) is a mixed continuous-discrete optimization problem. I perform alternative optimization that breaks the problem down into a continuous

optimization and a discrete optimization (see Figure 3.2). Note that this is a common strategy for constrained geometry optimization [8].

After an initial deformation, I alternate between finding the mapping ϕ while keeping M' fixed (i.e., the discrete problem) and computing the deformation M' while keeping ϕ fixed (i.e., the continuous problem). The continuous problem can be readily solved using existing mesh deformation techniques. Since the distortion term $E_{dis}(M')$ is borrowed from [41], I use their As-Rigid-As-Possible technique to solve the continuous problem. The key contribution is solving the discrete mapping problem, which will be discussed in the next section.

To initialize the iterative process, an initial, gross mapping ϕ, is computed which then gives rise to the initial deformed mesh M' by solving the continuous problem given ϕ. To get this mapping, I utilize the transformation T_i that aligns a target plane b_i with its corresponding source plane a_i, and take the vertex on the template mesh that is closest to the transformed location of a target contour vertex. If the contour vertex lies on multiple target planes (i.e., it is a junction of the contour network C), then the average location after applying the transformation on each of those planes is used. Specifically, for each target contour vertex p_i, I compute

$$\overline{p_i} = \frac{\sum_{j \in H(i)} T_j p_i}{|H(i)|},$$

where $H(i)$, as explained earlier, is the list of target planes that contain p_i, and transformations T_j are computed as discussed in Section 3. Then set $\phi(i)$ as the index of the vertex on M closest to $\overline{p_i}$.

The two steps, solving the continuous and discrete problems, are iterated until either a fixed number of iterations is reached or the change in the template shape is smaller than a

threshold. I use the former strategy with typically 6 iterations. Since the quality of mapping improves over time, w_{fit} is initially set to 1.25 and incremented by 2 in each iteration.

3.5.2 Curve Network Mapping

By fixing the deformation M', the non-constant portion of the energy of Equation 3.1 be can rewritten as follows:

$$E(\phi) = \sum_{i=1}^{m} \alpha(i, \phi(i)) + \sum_{\{i,j\} \in E_C} \beta(i, j, \phi(i), \phi(j)) \tag{3.5}$$

where $\alpha(i, i^*)$ is the cost of mapping a single contour vertex p_i to a template vertex v_{i^*},

$$\alpha(i, i^*) = w_{fit} \| p_i - v'_{i^*} \|^2 + w_{map} w_{src} \sum_{j \in H(i)} d(a_j, v_{i^*})$$

and $\beta(i, j, i^*, j^*)$ is the cost of mapping two edge-adjacent contour vertices $\{p_i, p_j\}$ to a pair of template vertices $\{v_{i^*}, v_{j^*}\}$,

$$\beta(i, j, i^*, j^*) = w_{map} \| T_{H(i,j)}(p_i - p_j) - (v_{i^*} - v_{j^*}) \|^2$$

Minimizing $E(\phi)$ becomes a *quadratic assignment* problem, which is known to be NP-hard. While approximate solutions can be computed efficiently, for example using spectral techniques [33], the results are often far from being optimal.

The key observation is that, since the domain of the mapping, C, has a 1-dimensional structure, the problem is not as difficult as the general assignment problem. In fact, the optimal mapping can be found in polynomial time for connected components of C that are

free of junctions (i.e., disjoint loops). Building on top of this optimal algorithm, I devise a greedy combinatorial search algorithm to find an approximate solution for the rest of C that contains junctions (i.e., networks).

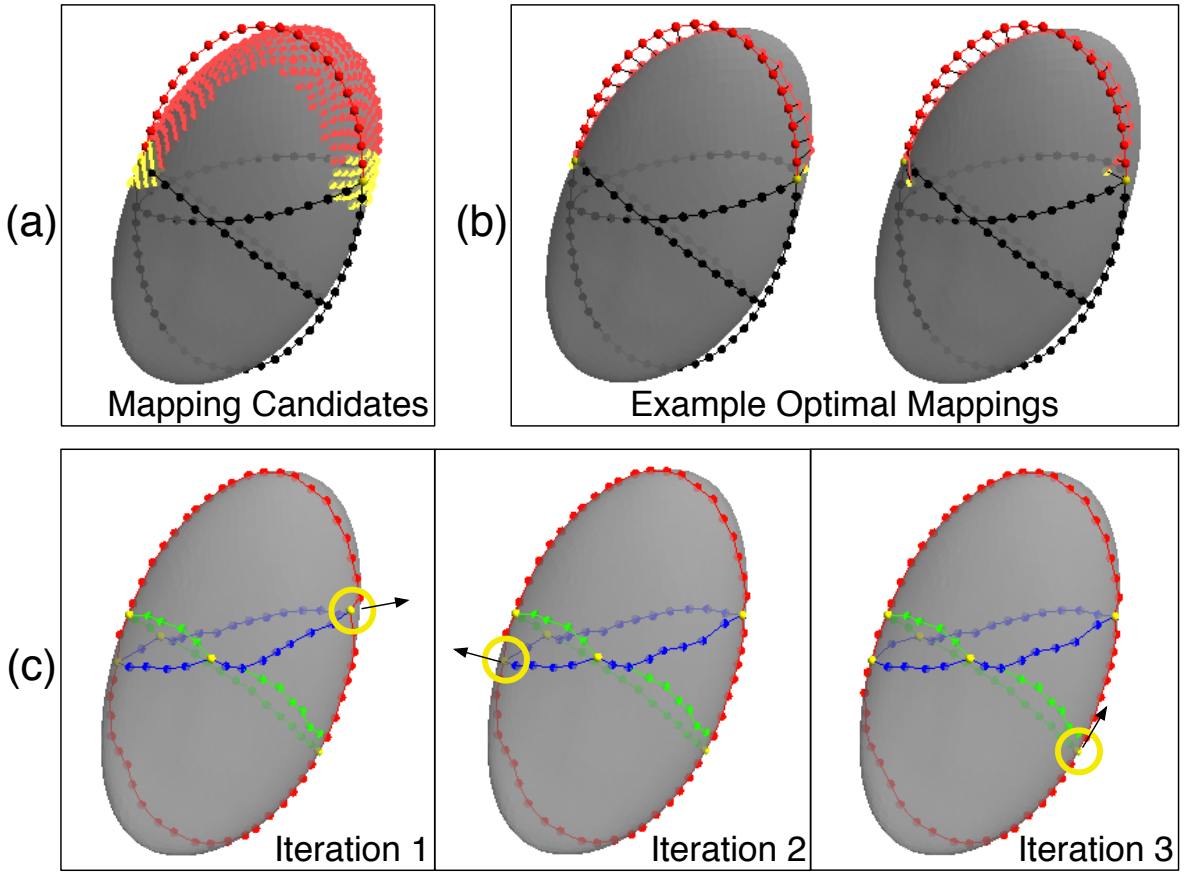

Figure 3.3: Illustration of curve network mapping. (a): mapping candidates Φ (as dots on the surface) for two junctions on the target contours (green dots) and interior vertices on a contour curve segment (red dots). (b): Optimal mapping ϕ (as connected dots on the surface) of one contour curve segment (red) under two different choices of mappings of the junctions. (c): Three iterations in the greedy algorithm, showing the mapped curve network in each iteration (as connected dots on the surface) and highlighting (in yellow circle) the mapped junction point that is adjusted in the iteration.

To reduce the computational cost, I limit the possible choices of $\phi(i)$ to a list of candidate template vertices $\Phi_i \subset \{1, \ldots, n\}$ for each contour vertex i. That is, *I want to find the*

mapping ϕ that minimizes $E(\phi)$ under the restriction that $\phi(i) \in \Phi_i$ for all $i = 1, \ldots, m$. In my implementation, I create the list Φ_i by computing the cost $\alpha(i, j)$ for each vertex v_j of M and taking the k vertices with the least cost. In all of the experiments, choosing $k = 50$ gives satisfactory results. Figure 3.3 (a) illustrates the candidate list for two junction vertices (green) and vertices on a contour curve segment (red).

Mapping Loops

To motivate the algorithm for mapping closed loops, let's first consider the (rather impractical) case where C is a single, open curve segment connecting consecutive vertices in the list $\{p_1, \ldots, p_m\}$. The energy-minimizing mapping ϕ in this case can be formulated as the shortest path in a weighted graph constructed as follows. Nodes $n_{i,s}$ are created for all $i = 1, \ldots, m$ and $s = 1, \ldots, k$, each representing a possible mapping $\phi(i) = \Phi_i(s)$, the latter being the s-th candidate in the list Φ_i. Each node is given a weight $\alpha(i, \Phi_i(s))$. It is helpful to imagine the nodes as points on a grid, so that $n_{i,s}$ lies on column i and row s. Every pair of nodes on two consecutive columns is then connected, or $n_{i,s}, n_{i+1,t}$ for any s, t and $i = 1, \ldots, m-1$. The weight of this arc is $\beta(i, i+1, \Phi_i(s), \Phi_{i+1}(t))$. Define the length of a path in this graph as the sum of node weights and arc weights along the path. The optimal mapping ϕ is thus encoded by the nodes along the shortest path from any node in the first column, $n_{1,s}$, to any node in the last column, $n_{m,t}$, for any s, t. Using dynamic programming, this path can be found in $O(k^2 m)$ time.

The algorithm can be easily extended, with the same time complexity, to the case when C is a single closed loop. Assume that the order of vertices on the loop is still $\{p_1, \ldots, p_{m+1}\}$ such that $p_{m+1} = p_1$. The graph is constructed as above. Then to ensure a unique mapping

for p_1, find the shortest path between any two nodes in the first and last column that lie on the *same row*, that is, between $n_{1,s}$ and $n_{m+1,s}$ for any $s = 1, \ldots, k$.

In general, C can be made up of several connected components, and each component is either a single loop or a network. The algorithm above can be applied to finding the optimal ϕ over any component of C that is a closed loop.

Mapping Networks

To find the mapping for a network C (or a network component of C), it is not enough to find a mapping for each individual curve segment (e.g., using the algorithm described above), since the mapping on different segments meeting at a junction may disagree for that junction point. On the other hand, if the choice of mapping at each junction point of C is fixed, the optimal mapping for the rest of C can be found by applying the algorithm above to each curve segment. The only adaptation needed is that the shortest path should be found between two fixed nodes in the first and last columns of the graph that correspond to the fixed mapping of the two junctions points.

This observation motivates the following greedy strategy. First make a best guess at the mapping $\phi(i)$ for each junction p_i and compute the optimal mapping for the remaining vertices of C. Then iteratively alternate between adjusting the mapping of one junction and updating the optimal mapping for curve segments incident to the junction. To choose that junction, pick the one whose adjusted mapping would result in the greatest drop in total energy $E(\phi)$ after updating the mapping of its incident segments. In essence, I take a steepest-descent route in the space of mappings of the junctions.

To implement the strategy efficiently, quick evaluation of the energy of the optimal mapping of a curve segment, bounded by junctions p_i, p_j, under different choices of $\phi(i)$ and $\phi(j)$ is necessary. I do so by pre-computing, for *all* pairs of junction mappings $\{\phi(i) = i^*, \phi(j) = j^*\}$ where $i^* \in \Phi_i$ and $j^* \in \Phi_j$, the energy of optimally mapping the rest of the vertices on the curve segment to the template. This can be done in $O(k^3 m)$ time using a similar dynamic-programming pass as described above. The resulting pair-wise energy values are stored in a matrix $D_{i,j}$, such that $D_{i,j}(i^*, j^*)$ gives the minimal mapping energy of the curve segment when $\{\phi(i) = i^*, \phi(j) = j^*\}$. As an example, Figure 3.3 (b) shows two optimal mapping of the same curve segment (red) for two pairs of junction mappings.

The complete algorithm proceeds as follows:

1. Initialize: Set $\phi(i) = \Phi_i(1)$ for each junction p_i. For each junction p_i and every candidate mapping $i^* \in \Phi_i$, let $\gamma_i(i^*) = \sum_{j \in N(i)} D_{i,j}(i^*, \phi(j))$ be the total mapping energy of all incident curve segments to p_i when $\phi(i) = i^*$ (here $N(i)$ is the list of indices of junctions that share a common curve segment with p_i). Let $\psi(i) = \arg\min_{i^* \in \Phi_i} \gamma_i(i^*)$ and $\delta(i) = \gamma_i(\phi(i)) - \gamma_i(i^*)$. Intuitively, setting $\phi(i) = \psi(i)$ would result in the lowest total mapping energy for all incident curve segments (while fixing the mapping at all other junctions), and $\delta(i)$ is the decrease in energy resulted from changing the current choice of mapping $\phi(i)$ to $\psi(i)$.

2. Repeat: Pick the junction p_i with the greatest $\delta(i)$ among all junction vertices. If $\delta(i) < 0$, the algorithm terminates, since no more adjustment can be made to junction mappings that would lower the energy (i.e., a local minimum is reached). Otherwise, set $\phi(i) = \psi(i)$, $\delta(i) = 0$, and update $\psi(j), \delta(j)$ for all neighboring junctions $j \in N(i)$.

Note that the algorithm always terminates, because the energy $E(\phi)$ decreases monotonically in each iteration and the space of all mappings is finite. The number of iterations is proportional to the number of junctions and also depends on how far M' is from C. In my experiments, I found that the algorithm typically iterates for about half the number of junctions or less. A few iterations of the algorithm are illustrated using the ellipsoid example in Figure 3.3 (c).

3.6 Results

3.6.1 Parameters

My method utilizes several parameters, some offering balance among various terms in the definition of energy E in Equation 3.1 (i.e., $w_{fit}, w_{dis}, w_{map}, w_{src}$) and others providing trade-off between speed and optimality during curve network mapping (i.e., the number k of nearest neighbors). I use the following set of parameter values for all real-world examples: $w_{dis} = 10$, $w_{map} = 1$, $w_{src} = 0.25$, $k = 50$, and $w_{fit} = 1.25$ initially and incremented by 2 in each iteration.

I demonstrate the results under different values of several key parameters on the synthetic ellipsoid example in Figure 3.4. The observations for each parameter are summarized as follows:

w_{dis} (second row): This parameter balances the two goals of fitting and shape preservation in deformation. If the parameter value is too small, large distortion of the shape would occur (as seen in the insert for $w_{dis} = 1$). On the other hand, setting the parameter

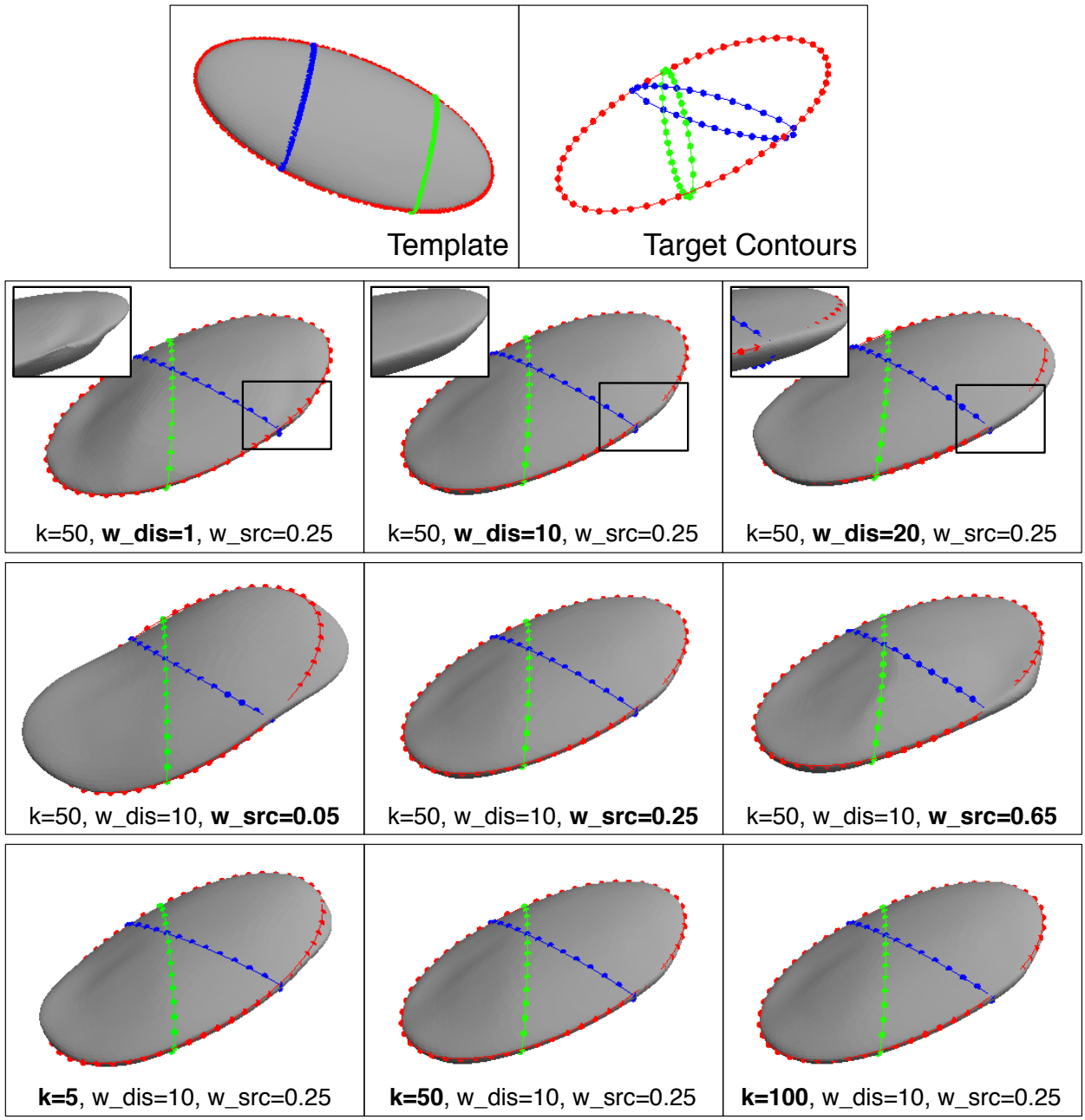

Figure 3.4: Deformation results for the ellipsoid example (top, same as in Figure 3.2) under different parameter settings. Each row varies a single parameter (bold text), and the remaining parameters assume the default values ($k = 50, w_{dis} = 10, w_{src} = 0.25$).

too high (e.g., $w_{dis} = 20$) would prevent fitting to the target contours (see the insert). I found that $w_{dis} = 10$ offers a good trade off between the two goals.

w_{src} (third row): This parameter controls the strength of the source plane constraint in the mapping term of the energy (Equation 3.4). The parameter is particularly important when the source planes and target planes are aligned respectively with geometric features of the template and the unknown object. For example, one of the source-target plane pairs in the ellipsoid example are aligned with the outer ridge of the ellipsoid (red curves in Figure 3.4 (top)). Insufficient constraint (e.g., $w_{src} = 0.05$) would fail to deform the ridge of the ellipsoid to the corresponding target contour. On the other hand, too much constraint (e.g., $w_{src} = 0.65$) may create shape distortions (see the lower-right part) when the configuration of the target planes differ significantly with that of the source planes.

k (last row): A small k would allow faster computation of mapping, but it offers only limited number of candidates that could be insufficient for mapping (e.g., see result of $k = 5$). I found that the choice of $k = 50$ achieves similar results as larger values (e.g., $k = 100$).

3.6.2 Real-world Examples

I tested my method on the ferret brain and liver data sets used in Chapter 2. I took a set of planes that a novice contoured on from the user study in Chapter 2, and set the target contours to be the cross-sections of the ground truth mesh with those planes. This way my method is isolated from potential error to due novice drawn contours. The templates

used are ground truth meshes from different ferret brain and liver volumes than the ones contoured.

For each data set, I compared with existing methods and alternative algorithmic choices. The findings are summarized below.

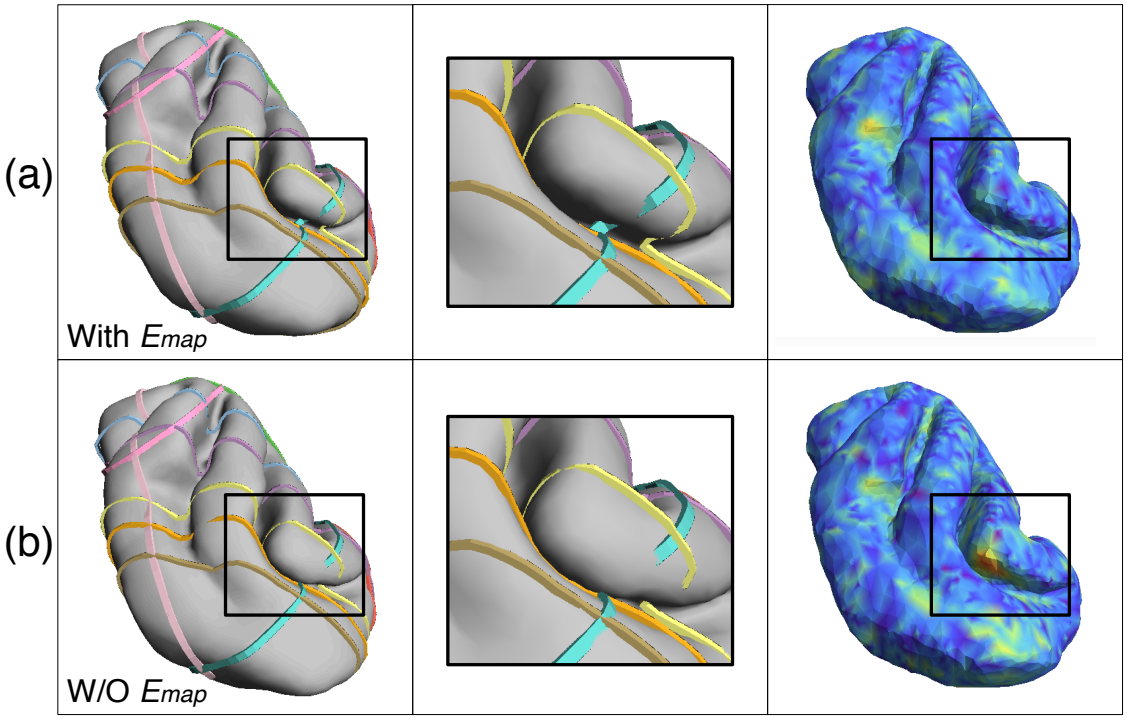

Figure 3.5: The deformation results using the 10 contour ferret brain input as in Figure 3.1 using my algorithm (a) and using my algorithm without E_{map} (b). The deformations are shown on the left and draw attention to the boxed area which is shown zoomed in the middle. Without the E_{map} term the groove connects to the orange contour instead of preserving the gap. The Hausdorff distance from the ground truth is mapped on the right to illustrate this error.

Ferret brain: The main results are shown in Figure 3.1. We can see that my approach, compared to two other representative interpolation-based reconstruction methods [37, 55], achieves results much closer to the ground truth even when using a sparser set of contours.

In Figure 3.5, I further demonstrate the importance of the mapping term (E_{map}) in the energy formulation: not only does it constrain the network mapping to be close to the source planes, it also considers similarity of curve shapes. Enforcing similarity avoids matching from one contiguous target contour to disjoint parts of the template surface. In the result produced without the mapping term (by setting $w_{map} = 0$ during iterative optimization), shown in Fig 3.5 (b), the gap space between the two nearby grooves highlighted in the box is significantly reduced, due to the incorrect correspondence between the groove and the orange contour (which should be under the groove). In contrast, the gap space is preserved with the mapping term.

Liver: The main results are shown in Figure 3.6 from two viewpoints. Even though the liver is not as rich in features as the ferret brain, the result in Figure 3.6 (e) is still closer to the ground truth than the interpolation-based approach in Figure 3.6 (c).

As mentioned in Section 3, an alternative to finding a mapping for the entire target contour network is to find separate mappings, one for the set of contours vertices on each target plane. This alternative can be implemented in my deformation framework by minimizing the energy terms that involve the mapping, E_{fit} and E_{map}, separately for each per-plane mapping. However, since each per-plane mapping is computed without knowledge of others, they may disagree where the target contours intersect, causing distortions in the deformation. Such distortions are demonstrated on the liver example in Figure 3.6 (d).

Finally, I compare the results of using two different templates mapping to the same target contours in Figure 3.7. Although the choice in template can make a difference, as seen in the varying spots of error, both results are still very similar and closer to the ground truth than the interpolation-based method (see the insert).

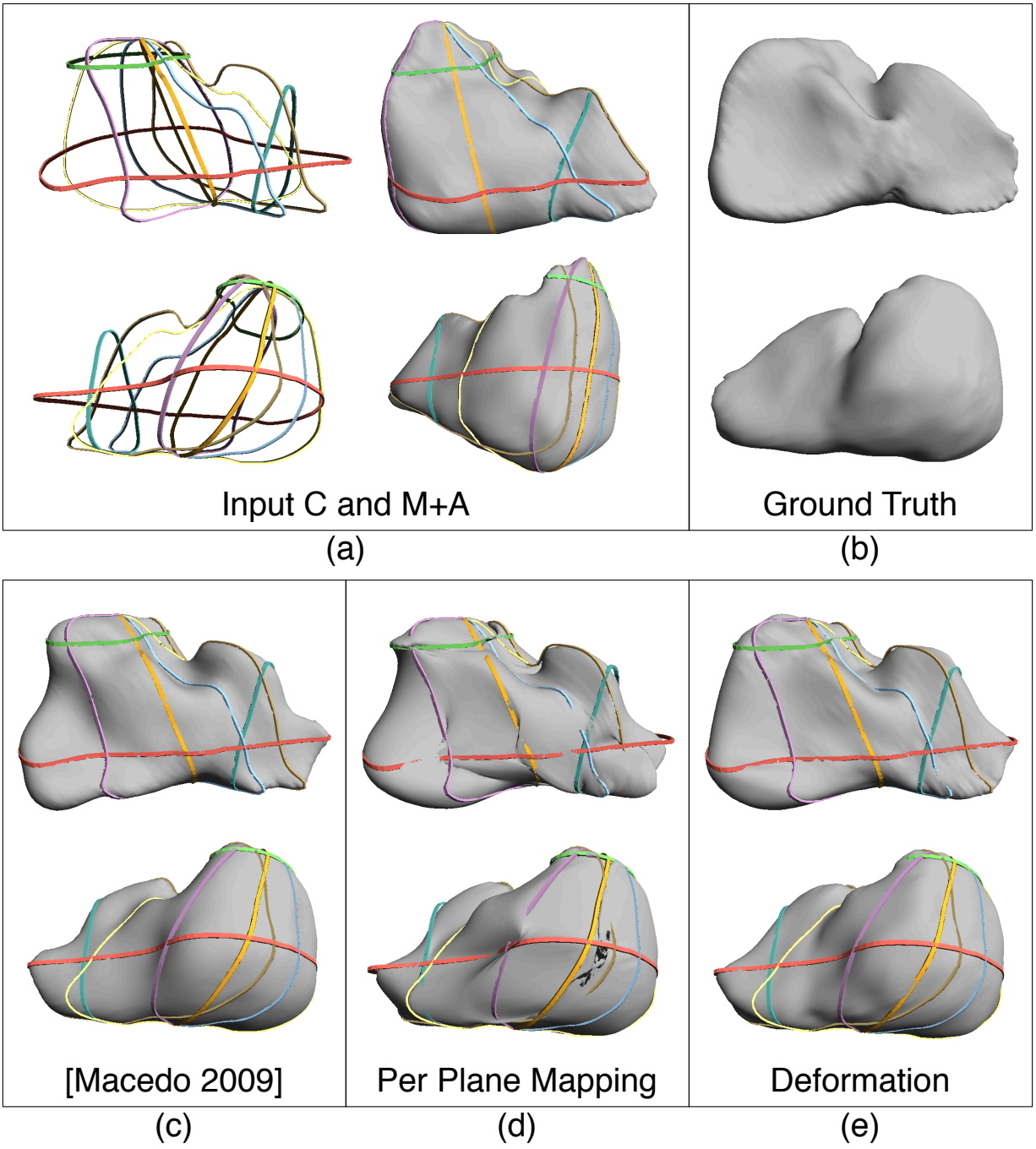

Figure 3.6: Two views (top and bottom) of the results using different reconstruction approaches for a liver. The input target contours and template (a), the ground truth (b), the results using [37] (c), the results of mapping each plane independently (d), my deformation result (e). Make note of the different layout of the source planes and target planes in the bottom view and how this affects the result in (d).

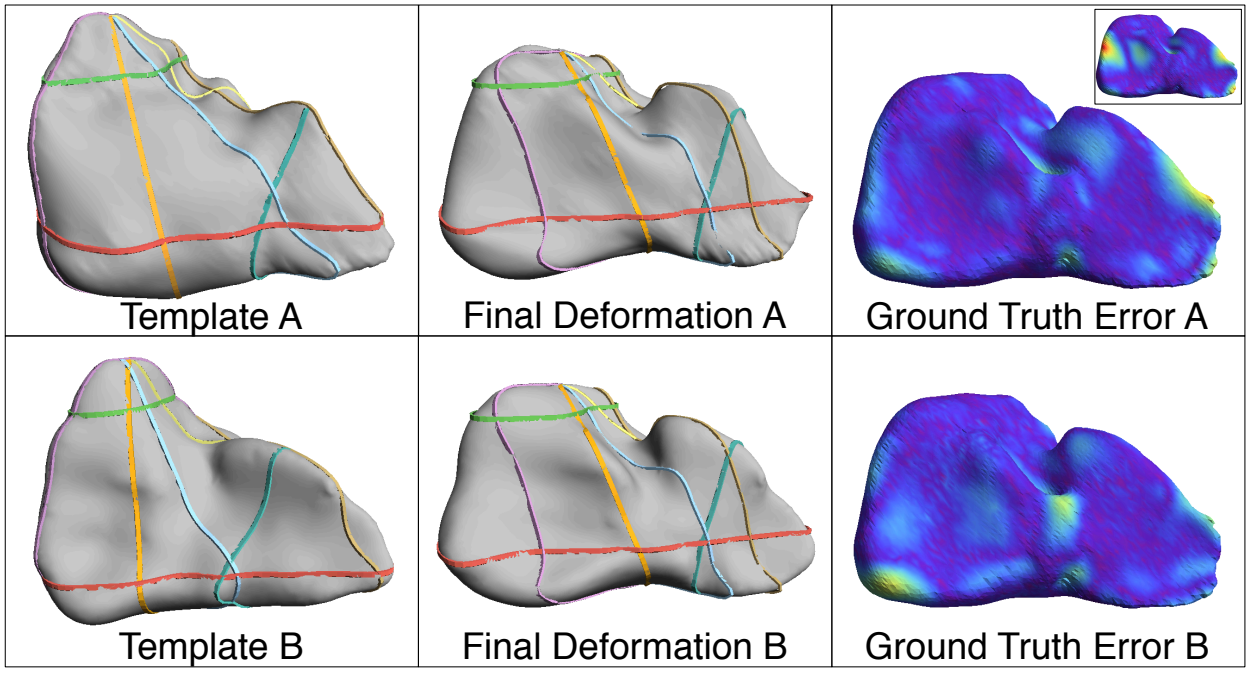

Figure 3.7: The results of the algorithm using two different templates and one set of target contours. The source is on the left, my result in the middle, and the Hausdorff distance from the ground truth on the right. The small insert image shows the Hausdorff distance of the result using [37] from the ground truth. Similar results are achieved using different templates with a few spots of variation and have much less error than the reconstruction method.

3.6.3 Performance

I implemented my algorithm in C++ on a Mac Pro (Late 2013), with a 3.7Ghz Quad-Core Intel Xeon E5 processor, and 16GB DDR3 RAM. I utilized the ANN library for k-nearest neighbors calculations. The template mesh and target contour information as well as the timing results (broken down into mapping and deformation) are shown in Table 3.1.

Model	M Vert #	C #	C Vert #	Time per Iter
Ellipsoid	2162	3	109	0.197+0.091s
Torus	4416	3	330	0.604+0.228s
Ferret	5002	4	522	0.926+0.238s
Ferret	5002	6	749	1.306+0.238s
Ferret	5002	8	945	1.622+0.237s
Ferret	5002	10	1157	1.974+0.236s
LiverA	10036	8	761	1.366+0.565s
LiverB	7502	8	761	1.337+0.399s

Table 3.1: Specifications for different timing trials. The ferret tests used one template and subsets of the ten target contours. The liver tests used two different templates and the same target contours. The first column shows the template name, followed by the number of vertices in the template mesh. Then the number of target contour planes followed by the number of target contour vertices. Lastly, the run time is the mapping time plus the deformation time for each iteration (each test was run with six iterations).

3.7 Discussion

I have detailed a new template-based surface reconstruction method from cross-section curves. I formulated the problem with the help of additional protocol planes on the template surface that are in correspondence with the target contour planes. I then used alternative optimization that includes a novel algorithm for mapping a network of spatial curves onto a surface. I tested my algorithm on both synthetic and real world data to illustrate the algorithmic choices and the algorithm's effectiveness.

Note that the algorithm is not limited to genus-zero shapes, as shown in Figure 3.8 (a). Observe that a reasonable deformation is produced even when the number of contour curves on the source and target planes differs. However, the method may not produce a satisfactory deformation when the source and target contours differ significantly and an insufficient number of planes are provided, as shown in Figure 3.8 (b).

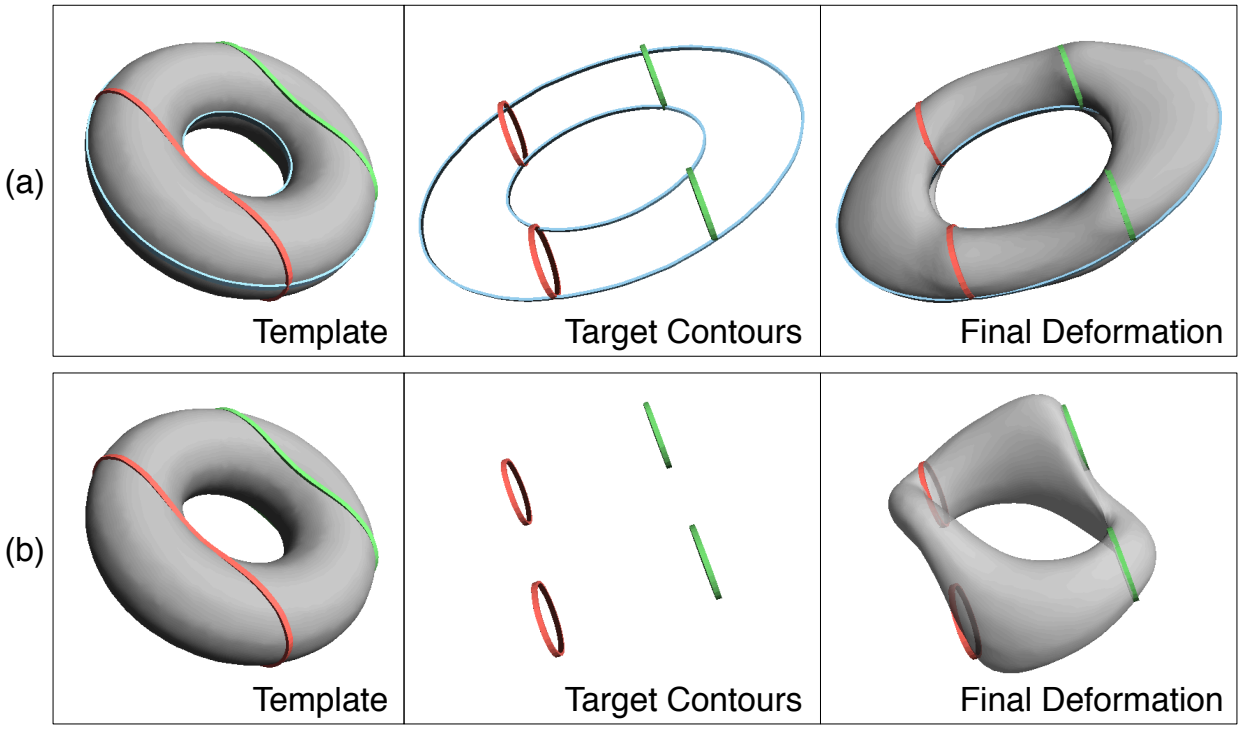

Figure 3.8: The results of the algorithm for a genus-1 synthetic shape. With sufficient planes a genus-1 shape can be deformed even when the number of source and target curves differ, as shown in (a). However with insufficient information, it can fail to find a satisfactory deformation, as shown in (b).

My method requires a set of source planes in addition to the template. While I use the contouring protocols to do this, different protocol configurations could lead to different reconstruction results given a specific target. To remove this bias, future work could explore automated methods for creating these planes on the template by analyzing the geometric features of one surface as well as the variation among a group of surfaces from past segmentations of similar data sets.

The curve network mapping algorithm currently does not enforce the topology of the mapped curve network on the template. A single target curve segment may be mapped to a self-intersecting curve on the template, and disjoint target segments maybe mapped to intersecting curves on the template. I have observed that such intersections often appear during the first few iterations of optimization, but they tend to disappear as optimization progresses and the template surface gets closer to the target contours. It would be interesting to explore topology equivalence as an additional constraint in mapping, although it would make the optimization task more challenging.

Chapter 4

Resolving Inconsistencies in Non-parallel Contour Networks

4.1 Motivation

As we've seen in the last two chapters, non-parallel contours in volumetric data are very useful in creating accurate segmentations. However, intersecting planar boundaries have some inherent difficulties. When two contouring planes intersect, then the two contours must be consistent with each other. This means that if the contours cross the line created by the intersection of their planes, then they must share common intersection points along that line and therefore have the same "inside" and "outside" regions along that line. In Fig 4.2(a) on the left we see two inconsistent contours with their plane intersection line, and the inside regions marked on the line for each contour. As you can see in the boxed portions, the line is marked as both inside and outside which causes conflicts during surface reconstruction. In Fig 4.2(a) on the right we see the consistent version of the contours where they meet at common intersection points along the line and their inside/outside regions match.

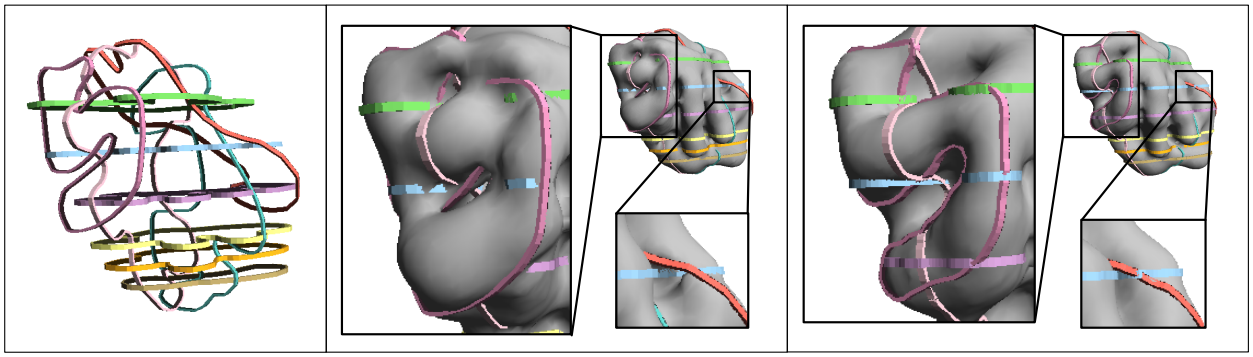

Figure 4.1: The surface results for an inconsistent and a consistent contour network. Part (a) shows the original inconsistent contour network. Part (b) highlights the surface issues that arise when trying to reconstruct a surface from the inconsistent network. Part (c) highlights how those issues are resolved by modifying the network to be consistent.

Inconsistent contours are detrimental to existing surface reconstruction algorithms, including my deformation approach from Chapter 3. Most of these algorithms simply fail to run given such inputs since they require consistency and exact intersection points. A few interpolation methods, such as [37], can tolerate inconsistency, but the results are far less satisfying than using a consistent input. Fig 4.1(a) shows an example of an inconsistent contour network, and Fig 4.1(b) shows the reconstruction result. You can see many reconstruction errors, such as topological holes and not exactly interpolating the contours, since the network contains conflicting inside/outside regions at the plane intersections. Fig 4.1(c) on the other hand, shows the reconstruction result using a consistent contour network.

One way to ensure contour consistency is to enforce the use of common intersection points while drawing each contour, which is what my system in Chapter 2 did. When the segmenter draws a contour on a plane, if any previous contours intersect that plane, then the intersection points are displayed and the segmenter is forced to use them when drawing the new contour. Though this can help ensure consistency, there are still issues with this solution. For complex shapes there can be many intersection points to pass through. Fig 4.2(b) shows the last plane

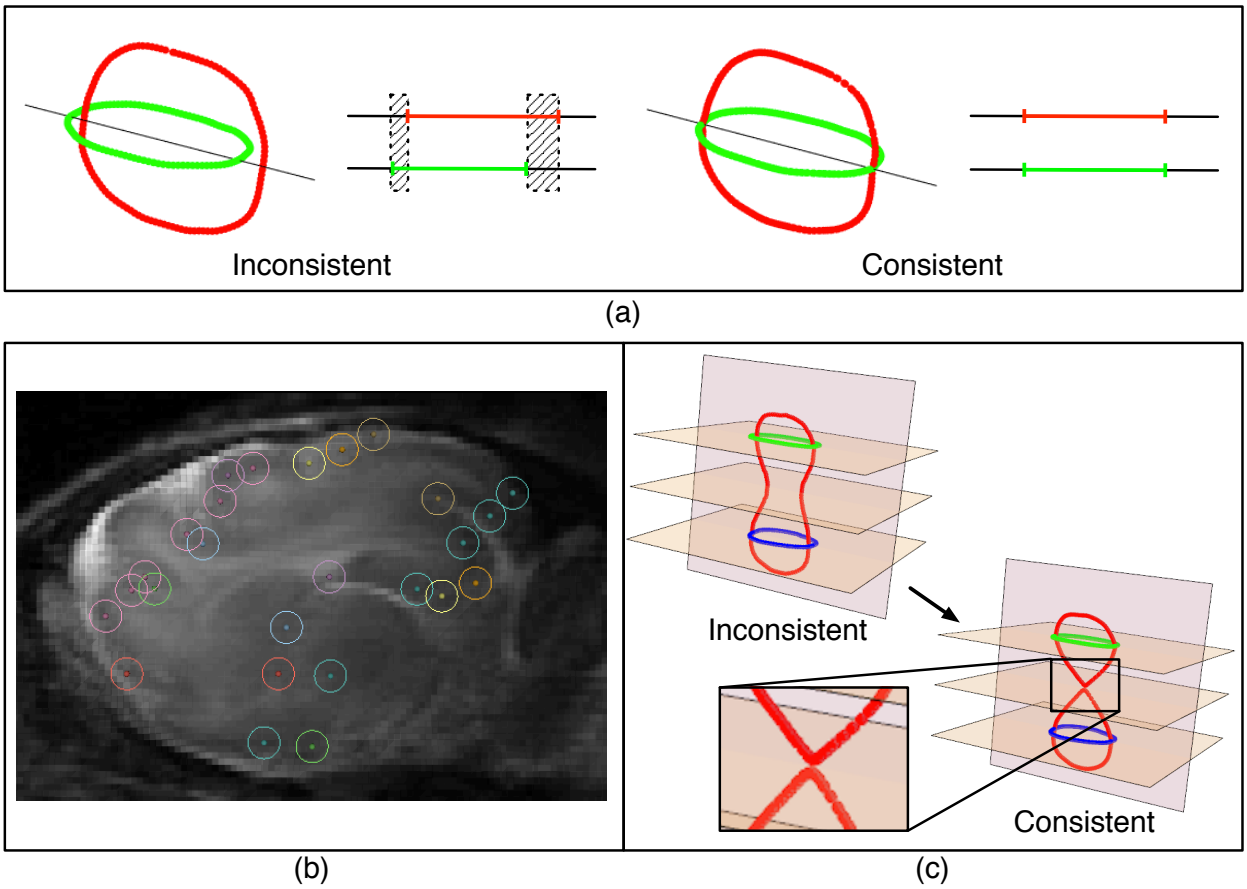

Figure 4.2: Contour network consistency. Part (a) shows the difference between two inconsistent contours and two consistent contours. Part (b) shows a difficult contour drawing example that requires intersections with many other contours. Part (c) shows an example change in topology that must occur to make the contour network consistent.

from the ferret brain contouring protocol used in the user study in Section 2.5. There are so many intersection points to use that it is difficult to even understand how to connect them across the boundary, and connecting them in the wrong order can still lead to inconsistencies. Also, the user has to additionally take care to not create new intersection points with existing planes. These requirements add a significant burden on the user. Furthermore, placing these intersection points on the screen can be distracting and could potentially lead them to draw

incorrectly. Instead of drawing what they think is the boundary, they end up trying to just connect the dots instead.

I would like to automate the work of maintaining contour network consistency as a post process after drawing all individual contours. This way the user can focus on the more important task of drawing the boundary they see in the image and not have to worry about consistency during segmentation. I want to take an inconsistent contour network as input and modify it to resolve any inconsistencies given two objectives: (1) guarantee consistency among all planes, and (2) maintain the shape of the contours as much as possible.

One possible approach to this problem is to deform each curve to resolve the inconsistencies. I could first compute the location of common intersection points (e.g. by averaging the inconsistent points along the lines of intersection between each plane) and then deform each curve to pass through those points. However, there are several issues with this method. It is difficult to make sure new inconsistencies are not introduced after deforming each curve. More importantly, deformation cannot handle making topological changes to the curve, which could be necessary as shown in the example in Fig 4.2(c).

I propose to use implicit functions in order to resolve all contour network inconsistencies. In this strategy, the contour curves are represented as the level sets of scalar functions, and the curves are modified by changing the functions. Function values that are negative mean inside the boundary, positive mean outside the boundary, and zero gives the boundary contour curve. With this approach consistency can be easily guarantee since we only need to make sure that the functions on intersecting planes have matching signs along the intersection line. Additionally, topological modifications to the contours comes for free since iso-contours

are derived from the implicit function values. I formulate the problem as a quadratic programming problem with the goal of modifying the functions to have matching signs while preserving the shape of the level set.

In the following I present, to the best of my knowledge, the first automatic and robust solution for resolving inconsistencies in non-parallel contour networks.

4.2 Algorithm

The overall workflow of my algorithm is shown in Fig 4.3(a). The input is an inconsistent contour network shown on the left. We can see the major inconsistencies between the red contour and the other three. The output will be a consistent contour network shown on the right.

4.2.1 Step I - Implicit Function

The first step of the algorithm is to define the implicit function on each contour plane. For each plane, construct a 2D graph that includes the intersection lines with that plane, and the contour points. I pre-sample vertices on the lines so that these vertices can be shared on the graphs for each plane. Each 2D graph is then triangulated using Triangle [39]. Voronoi points of the contour are inserted into the triangulation to avoid boundary issues in narrow regions where one vertex can have multiple edges that meet the contour. The resulting triangulated graph for the red contour plane is shown in Step I of Fig 4.3(a).

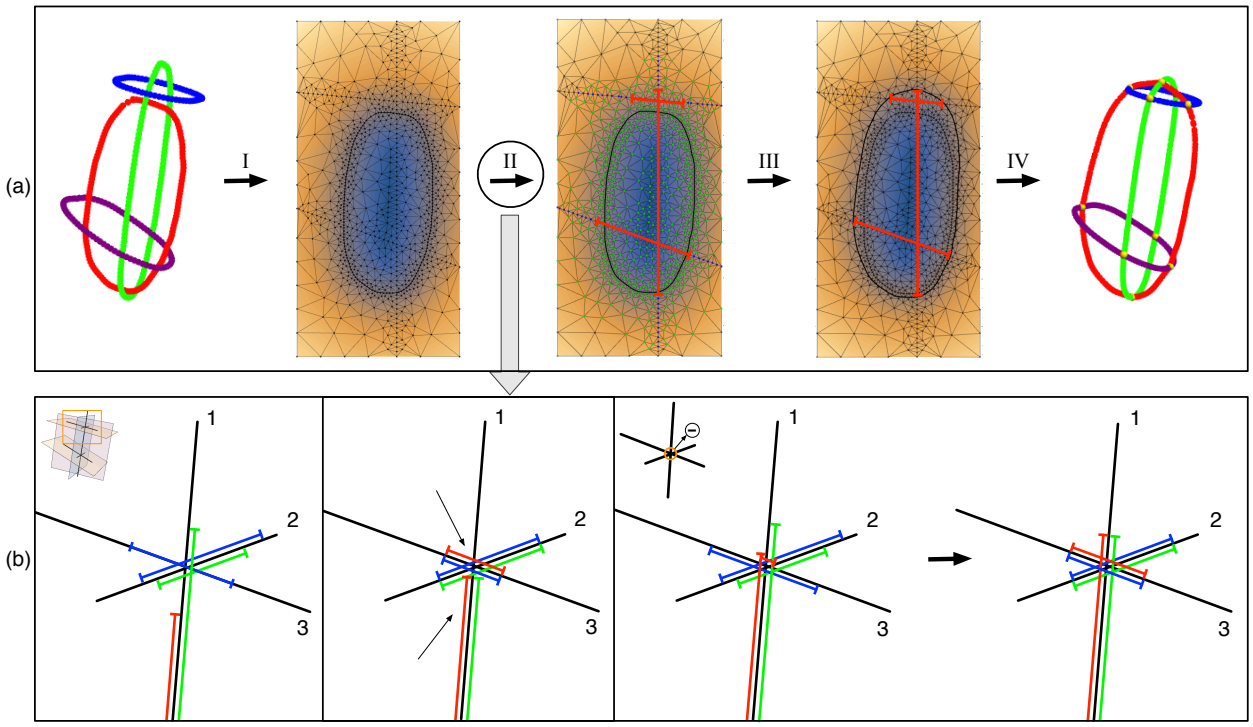

Figure 4.3: Work flow of the algorithm. Part (a) shows the main work flow and part (b) shows details for step II. Given an inconsistent contour network, step I constructs a triangulated graph of each plane and defines the implicit function across it. Step II finds the label constraints for the plane intersection lines by first constraining the line intersection points and modifying the implicit functions on those lines. Step III then modifies the implicit function on each plane according to the line constraints. Lastly, in step IV the zero level set iso-contours are extracted.

The implicit function can then be defined such that the value at each vertex is the signed distance from the contour curve on that plane (positive is outside, negative is inside). Vertices that belong to more than one plane (i.e. all the vertices on the plane intersection lines) will hold more than one value. The values at these vertices can have different signs, which are the inconsistencies that need to be fixed. The implicit function for the red contour plane is plotted as shown in Step I of Fig 4.3(a), where yellow is positive and blue is negative.

4.2.2 Step II - Constraints

The end goal of the algorithm is to modify the implicit functions on each plane so that the contours are consistent. In order to do this constraints must be placed on some of the graph vertices. Since the issues with consistency reside on the plane intersection lines, the function values on these vertices will be constrained. There are two types of constraints: positive constraints (i.e. "Outside") and negative constraints (i.e "Inside"). These constraints specify the sign that the implicit function values should be. In other words, for this step I want to set which regions of the lines are "Inside" and "Outside".

To start, a label is applied to each vertex as either "Inside", "Outside", or "Fuzzy". I define a constant, ϵ, to represent a fuzzy region around the original contour (i.e. a maximum distance from the contour curve). This allows control for how far the contours can be modified from the originals. Each vertex is then labeled as follows: "Inside" if any of its values is less than $-\epsilon$, "Outside" if any of its values is greater than ϵ, "Fuzzy" if all values are within $-\epsilon \rightarrow \epsilon$, and "Error" if a vertex has two values that qualify as both "Inside" and "Outside". If a vertex is labeled as "Error" then the contours that use this vertex are flagged as containing a large drawing error and require editing. Therefore the larger ϵ is, the larger inconsistencies are tolerated.

If there are no errors, then go to all the line vertices that are labeled as "Fuzzy". I want to change their labels to be either "Inside" or "Outside" according to the sign of the average of the values at each vertex, so that each line has a consistent labeling between its planes. However consistency across all planes cannot be guaranteed by just averaging the initial implicit function values.

For an example we can look at just the intersection lines formed by the red, green, and blue contour planes in Fig 4.3. On the left in Fig 4.3(b) we see these three lines with the inside regions on each line colored by the contour. Line 2 is the intersection of the green and blue contours and its regions are already fairly close. However lines 1 and 3 which are the intersections of the red contour with the green and blue contours respectively, greatly differ. The red region on line 1 is ends much lower than the green region, while line 3 doesn't even contain a red region. If the sign of the average of the values on each line is used to obtain matching regions, we get the result in the middle of Fig 4.3(b). While each individual line's regions are now consistent, they are not consistent with each other at the intersection point. If you look at the red regions one crosses the intersection points while the other one doesn't (same for the green regions).

Therefore before averaging the values on the lines, I will run an initial modification step for each implicit function on each line using the line intersection points as constraints. Each intersection point is set to be a positive or negative constraint according to the sign of the average of its values. In this example the intersection point is set to be negative (i.e. "Inside"). Each implicit function on each line is then modified using a quadratic programming method that is detailed in Section 4.3. In this example the functions are modified on each line so that the intersection point has a negative value, as shown in the third picture of Fig 4.3(b). Now the values on the lines can be averaged to get the fully consistent labeling shown on the right in Fig 4.3(b).

After modifying the functions on all lines according to the intersection point constraints, the labels for all line vertices are set as either "Inside" or "Outside" depending on the sign of the average value at each vertex. The results of this labeling on the red contour plane is

shown in Step II of Fig 4.3(a). The green points are “Fuzzy”, the blue points are “Outside” and the red line regions are “Inside”.

4.2.3 Step III - Function Modification

The next step is to modify the implicit function on each plane, and using the line vertices (and vertices outside the fuzzy region) on each plane as constraints. All vertices labeled as “Inside” will be negative constraints, and all vertices labeled as “Outside” will be positive constraints. The implicit functions on each plane are then modified so that the signs of the function values match the constraints on that plane and maintains the shape of the function as much as possible. This is done using quadratic programming which is detailed in Section 4.3. The updated implicit function and new iso-contour are shown in Step III of Fig 4.3(a).

4.2.4 Step IV - Contour Extraction

The final step is to extract the zero level set iso-contour on each plane. Marching triangles is used to generate the contour points on all sign changing edges. If the two vertices of a sign changing edge have more than one value, then use the average of the values. This way the same point will be generated on the planes that share that edge which guarantees the common intersection points between the contours. The final consistent contour network is shown in Fig 4.3(a) on the right, and the yellow spheres highlight the intersection points of the contours.

4.3 Quadratic Programming

In two parts of my algorithm I must modify a function given some sign constraints (i.e. positive/outside and negative/inside). In this section I will explain how I use quadratic programming to achieve this.

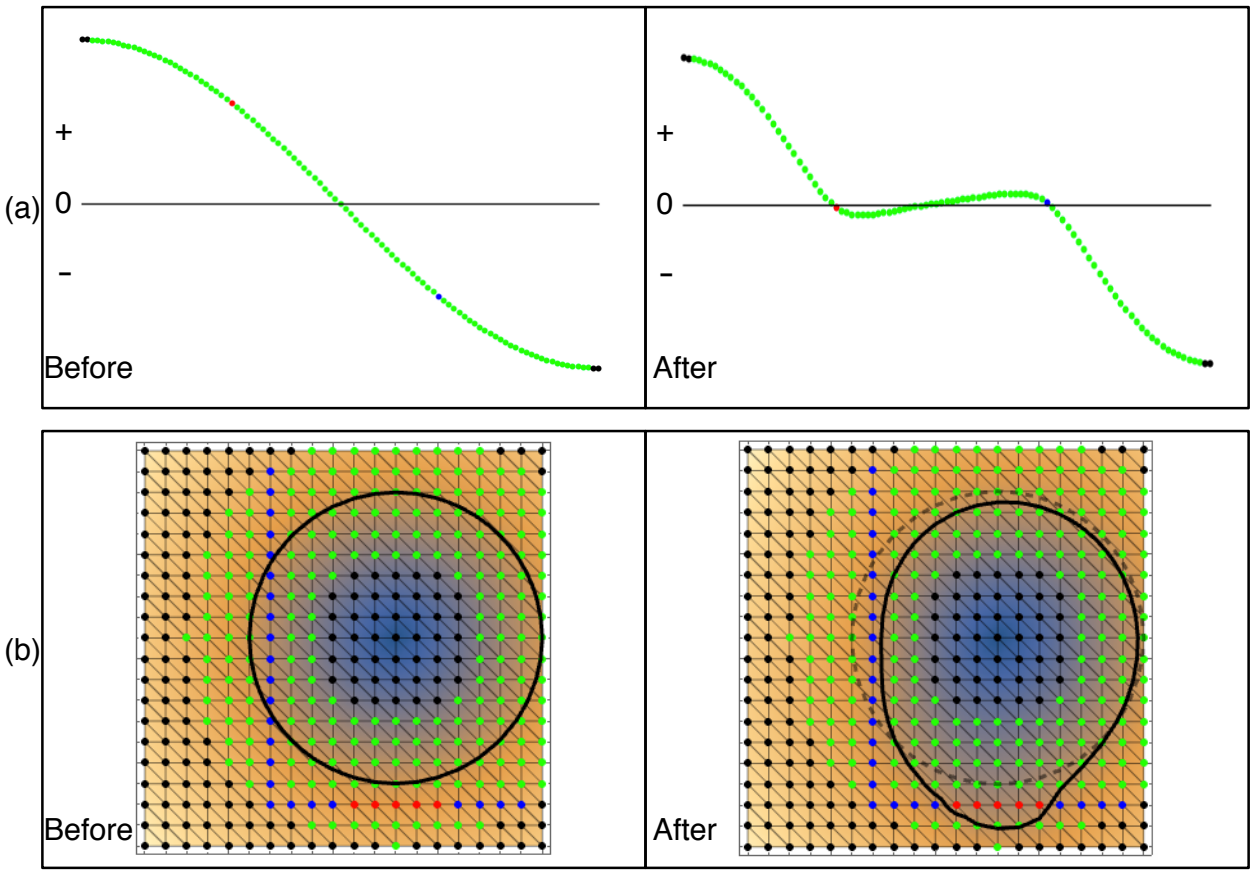

Figure 4.4: Example implicit function modification for 1D and 2D input. Part (a) shows 1D input and the constrained nodes (red is negative, blue is positive) plus the "Fuzzy" nodes (green) on the left, and the result of the modification on the right. Part (b) shows 2D input and the constrained nodes on the left, and the result of the modification on the right.

The input is a graph with function values at each node, and for a subset of these nodes the sign of the function value is constrained to be either positive or negative. See the "Before" parts of Fig 4.4(a) and (b) for example 1D and 2D input. For the 1D case the points' height

represents the function value. The constrained nodes are shown as red for negative and blue for positive, and the "Fuzzy" nodes are green. The output is the modified function values at each node such that the signs of the constrained nodes are satisfied. See the "After" parts of Fig 4.4(a) and (b) for example 1D and 2D output. Notice how the red constrained nodes are now negative, and the blue nodes are now positive.

I formulate this problem as a least-square problem with inequality constraints using the objective function shown in Eq 4.1. This functions aims to accomplish two things: minimize the distortion of the shape of the implicit function, f, and minimize change in the function values. The first part utilizes the standard weighted sum of squared Laplacians using cotan weights (w_{ij}) to minimize the change in Laplacian, $\nabla^2 f - \nabla^2 f'$. This encourages using different level sets of the implicit function since that won't change the Laplacian. The second part minimizes the difference or change in function values with a weight, α, to control its influence. Since I only care about the *change* in function values and not the values themselves, the variables I will solve for are the *change in values*, $x_n = v_i - v_i'$, for each node in the graph. This works since change in Laplacian ($\nabla^2 f - \nabla^2 f'$) is equivalent to the Laplacian of the change ($\nabla^2(f - f')$).

$$argmin \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} \|\mathcal{L}(v_i - v_i')\|^2 + \alpha \|v_i - v_i'\|^2 \tag{4.1}$$

Now an inequality is set up for each constrained node:

Positive Constraints For the positive constrained nodes the inequality is $x_n > -f(n)+d$. The change in value, x, for node n must be greater than minus the current value, $f(n)$, plus a positive constant d. Since this node must be positive, the change in value needs to make

the new value greater than zero. The constant d is added to keep the constrained nodes from being exactly zero and falling on the contour curve. It is a fraction of the maximal difference in values between any edge in the graph. The greater d is the further from zero the new value should be.

Negative Constraints For the negative constrained nodes the inequality is $x_n < -f(n) - d$. The change in value x for node n must be less than minus the current value $f(n)$ plus a positive constant d. Since this node must be negative, the change in value needs to make the new value less than zero. The constant d is the same as before.

Eq 4.1 is then minimized according to these constraints to find the change in functions values for all nodes on each plane.

4.4 Results

This section will show the results of my algorithm on some varying contour networks. In all examples I use [37] for surface reconstruction unless otherwise noted. I use this interpolation method because it can handle inconsistencies and find a surface even if it can't exactly interpolate the input.

4.4.1 Synthetic Examples

The results for the synthetic example used in Fig 4.3 are shown in Fig 4.5(a) and (b). Fig 4.5(a) shows the inconsistent contours and the resulting surface. You can clearly see the surface issues that occur, such as the red contour passing through the surface, because of

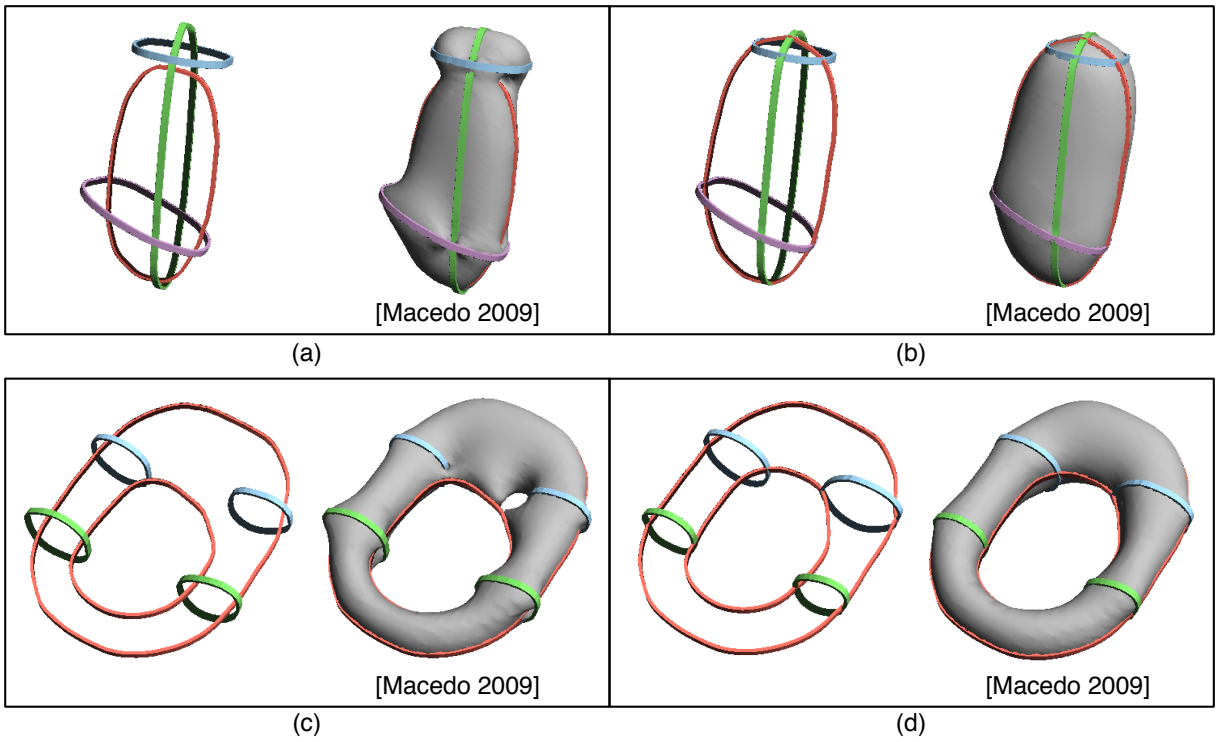

Figure 4.5: The results of the algorithm on two synthetic shapes. Part (a) shows the inconsistent contours and resulting surface for the workflow example, and part (b) shows the resulting consistent contours and surface. Part (c) shows the inconsistent contours and surface for a torus shape, and part (d) shows the resulting consistent contours and surface.

the inconsistencies of the red contour with the others. The result of my algorithm to obtain a consistent network is shown in Fig 4.5(b). The contours exactly intersect each other and the surface is much cleaner and free of issues.

The results for a synthetic torus shape are shown in Fig 4.5(c) and (d). Fig 4.5(c) shows the inconsistent contours and the resulting surface. Again, you can clearly see how even small contour inconsistencies affect the final surface. The result of my algorithm is shown in Fig 4.5(d), and provides a much more desired final surface.

4.4.2 Real-world Examples

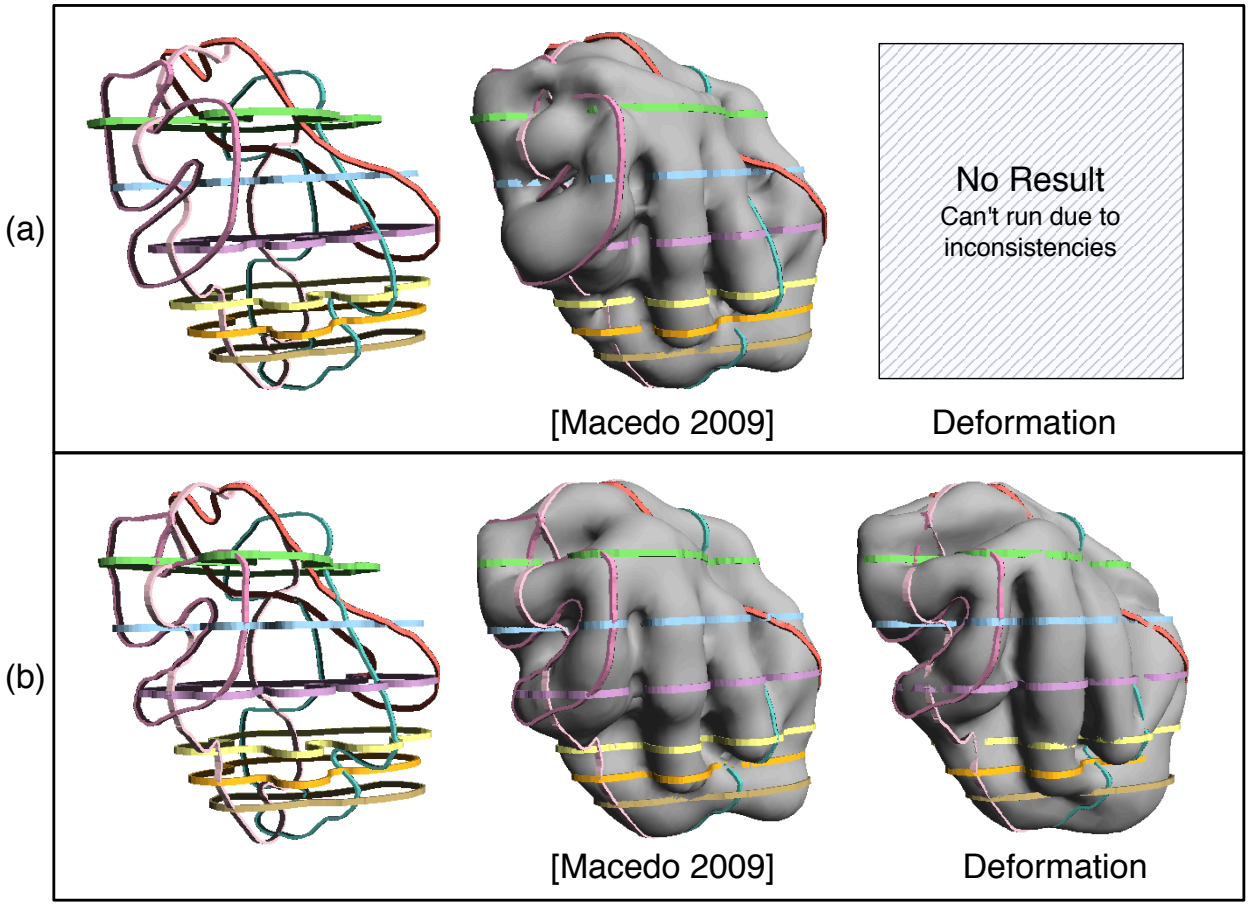

Figure 4.6: The results of the algorithm on a real world ferret brain data set. Part (a) shows the inconsistent contours and resulting surface. Part (b) shows the consistent contours plus the surface results from two different methods: [37] and my deformation technique from Chapter 3. Note that we can't use the deformation technique on the inconsistent contour as it requires exact intersection points.

Ferret Brain The results for a ferret brain data set using ten contours are shown in Fig 4.6. The inconsistent contours and the resulting surface are shown in Fig 4.6(a), and the new consistent contours using my algorithm and the resulting surface are shown in Fig 4.6(b). The consistent surface is much more pleasing and can exactly interpolate all of the contours.

Additionally with a consistent contour network my deformation technique from Chapter 3 can be used to obtain an even more accurate surface reconstruction.

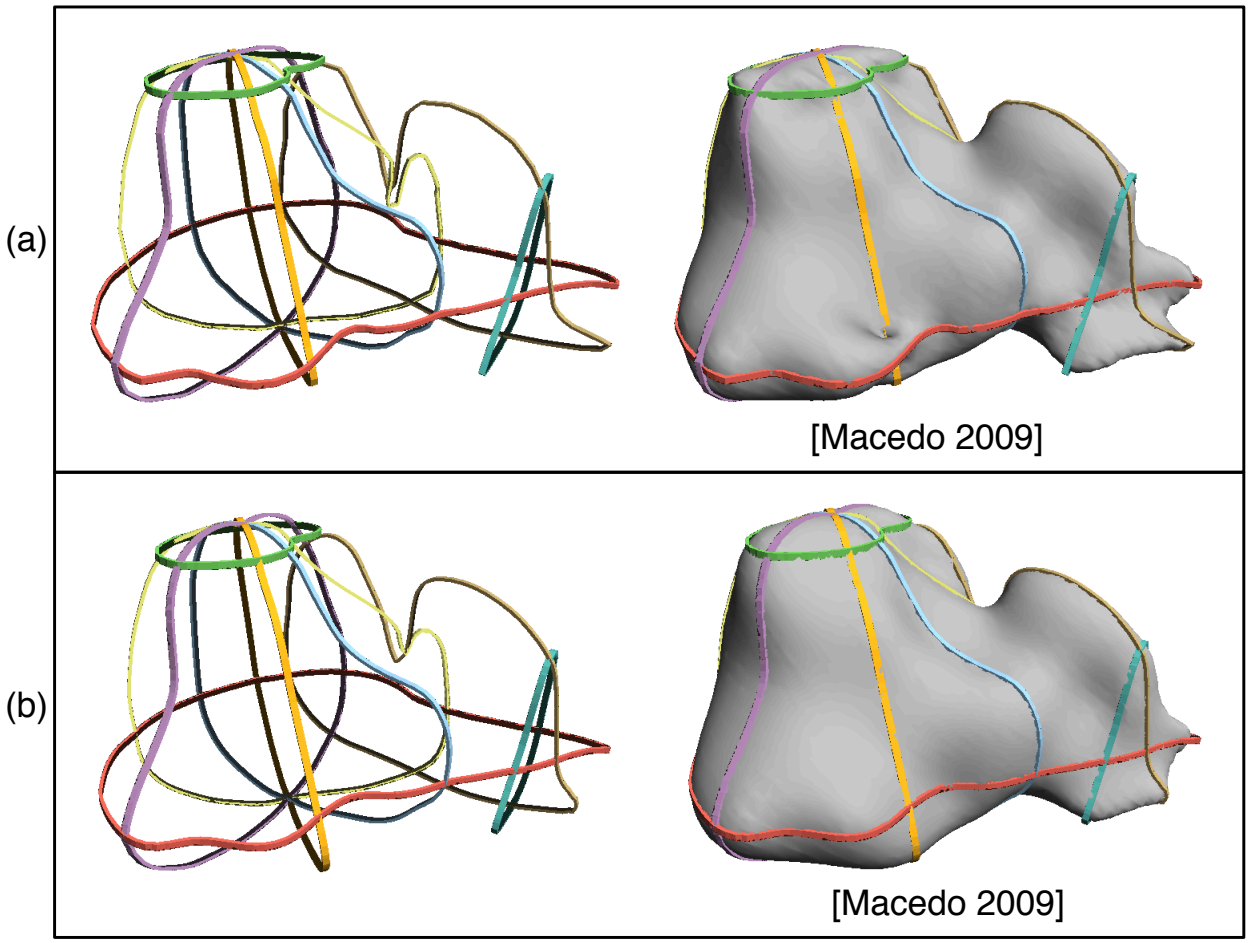

Figure 4.7: The results of the algorithm on a real world liver data set. Part (a) shows the inconsistent contours and resulting surface. Part (b) shows the consistent contours and resulting surface.

Liver The results for a liver data set using eight contours are shown in Fig 4.7. The inconsistent contours and the resulting surface are shown in Fig 4.7(a), and the new consistent contours and the resulting surface are shown in Fig 4.7(b). The consistency issues with this data set are fairly small, but you can see the differences in smoothness of the surface around the intersection points.

4.4.3 Limitations

There are some cases where although the final resulting contour network will be consistent, it is not necessarily what a human would expect or want to happen.

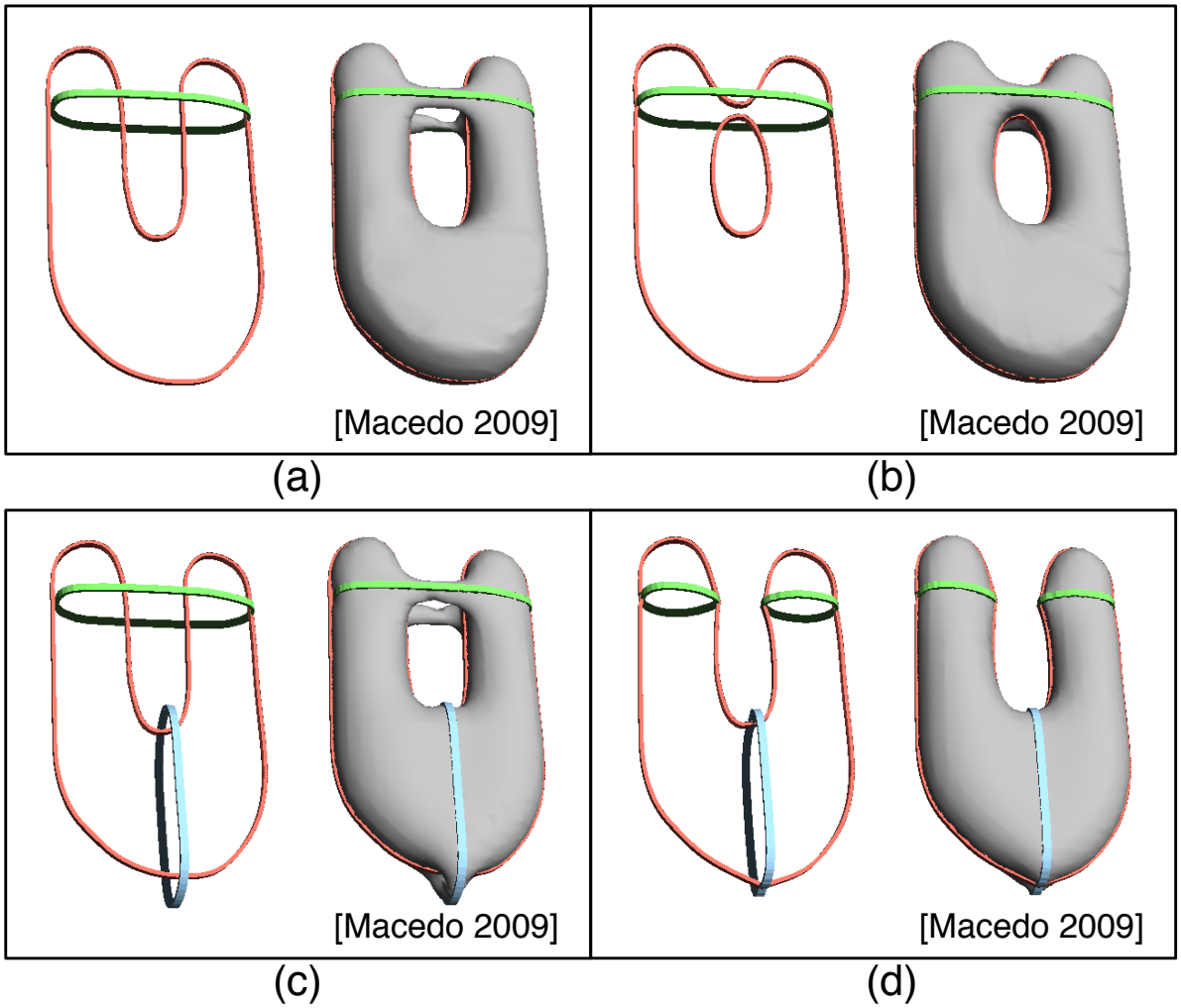

Figure 4.8: An example limitation for the algorithm. Part (a) shows the inconsistent contours and part (b) shows the consistent contours that the algorithm would produce. Part (c) shows the same inconsistent contours with one additional contour and part (d) shows how this addition changes the result of the algorithm.

If there's a narrow U shape with an oval around the top as shown in Fig 4.8(a), we would mostly likely expect the oval shape to split into two contours around each end. However since

my algorithm is based on signed distances to the contour, when you average the values along the plane intersection line, the inside values from the oval will be greater than the outside values from the U. The result therefore merges the tops of the U to make the contours consistent as shown in Fig 4.8(b). This problem can be remedied to obtain the more desired result by adding another contour plane down the center of the U as shown in Fig 4.8(c) and (d).

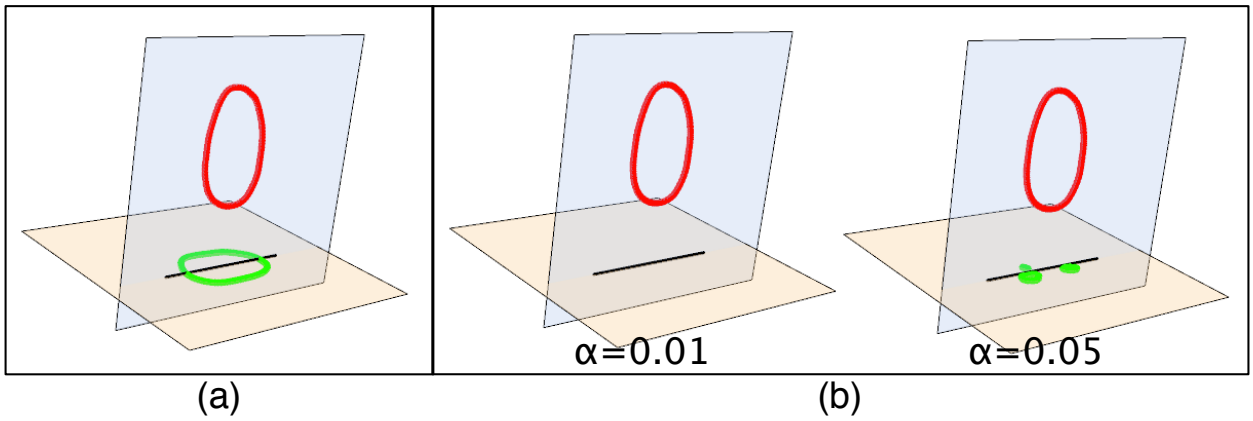

Figure 4.9: Another example limitation for the algorithm. Part (a) shows the inconsistent contours, the red oval on top and the green oval further below. Part (b) shows two different results of the algorithm by varying the weight α.

Another situation that could cause issues is shown in Fig 4.9(a) with the red oval above and the green oval far below. In this case the entire intersection line would be constrained to be "Outside". Depending on how we set the α weight the result could either completely erase the green contour or split it into small components on either side of the line as shown in Fig 4.9(b). This happens because the change in Laplacians is minimized and the different level sets of an oval shape don't naturally split into two curves. This is a case we would most likely want to flag as an error since there should probably be a second contour on the red plane. The value for ϵ would have to be set low enough so that at least one vertex inside the green contour is outside the "fuzzy" region.

Again, note that none of these results fail to produce a consistent result, but these cases fail to meet a human's expectations due to a lack of sufficient input.

4.5 Discussion

In this work I have detailed the first automatic and robust algorithm for resolving inconsistencies in contour networks. I make use of implicit functions to define the contours and use quadratic programming to modify these functions so that the resulting iso-contours are consistent among all planes. I demonstrate my algorithm on both synthetic and real world datasets.

In the future the algorithm could be augmented to use image gradient data when available. It could be used during Step II of the algorithm when finding the constraints for the intersection lines. It could also help in Step III when modifying the implicit function on each plane so that the curves also consider strong image gradients when fitting the given constraints. Additionally, instead of using a constant ϵ, it could vary across the curve. Contour drawing with different brush sizes would determine how large or small inconsistencies would be tolerated and fixed (i.e. a confidence measure in the drawn boundary).

Chapter 5

Conclusions

In this disseration we have looked at three problems to improve the process of manual segmentation with a focus on biomedical applications using guidance and alignment with structure features. I focused on the use case were similar classes of data sets are repeatedly segmented, building up knowledge on the common shape, topology, and fine details of the structure(s) of interest. I utilized this prior structure knowledge in order to facilitate a more approachable contour drawing experience and reconstructing a more accurate surface from a set of drawn contours.

First I detailed a guided segmentation system that addresses key difficulties in working with 3D volumetric data. I utilized prior structure knowledge to define a contouring protocol that provides step-by-step guidance in the form of automatic navigation through a path of preselected planes that are placed to capture the global shape of the structure, plus examples of past segmentations. This supported using arbitrarily-oriented planes in the data without unduly increasing the cognitive load to do so. I then looked at two problems that improved two aspects of this system. First I examined a new deformation approach to reconstructing a surface from curves that utilizes past segmentations and contouring protocols. This allows the preservation of general shape, topology, and fine details that can't be captured through

smoothly interpolating a sparse set of curves. And lastly, I detail a robust algorithm for resolving inconsistencies in non-parallel contour networks. Consistency is required in most reconstruction algorithms (including my deformation approach), but it is much easier on the user to initially draw inconsistent contours.

These problems open up some interesting directions for future work. Mostly notably in these problems we assume that sets of planes for a given structure are already preselected. Devising an algorithm to automatically select a certain number of best contouring planes using the volumetric data or structure surfaces could increase the usefulness of these guided approaches to segmentation. As discussed earlier, manual segmentation is used because the data is often not amenable to automatic methods. However, parts of this work could be augmented to consider image data in cases were it is strong enough to provide guidance towards achieving a more accurate segmentation.

References

[1] Christopher Abraham, Daniel Low, Ross Sowell, Garima Gokhroo, Cindy Grimm, and Tao Ju. VolumeViewer: a tool for examining the use of non-axial image planes in treatment planning. In *Proceedings of the 16th International Conference on the Use of Computers in Radiation Therapy*, 2010.

[2] Chandrajit L. Bajaj, Edward J. Coyle, and Kwun-Nan Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graph. Models Image Process.*, 58(6):524–543, 1996.

[3] Gill Barequet and Micha Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63:251–272, 1996.

[4] Gill Barequet and Amir Vaxman. Nonlinear interpolation between slices. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 97–107, 2007.

[5] Amit Bermano, Amir Vaxman, and Craig Gotsman. Online reconstruction of 3d objects from arbitrary cross-sections. *ACM Trans. Graph.*, 30(5):113:1–113:11, October 2011.

[6] Jean-Daniel Boissonnat. Shape reconstruction from planar cross sections. *Comput. Vision Graph. Image Process.*, 44(1):1–29, 1988.

[7] Jean-Daniel Boissonnat and Pooran Memari. Shape reconstruction from unorganized cross-sections. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 89–98, 2007.

[8] Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum*, 31(5):1657–1667, August 2012.

[9] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[10] Siu-Wing Cheng and Tamal K. Dey. Improved constructions of delaunay based contour surfaces. In *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 322–323. ACM Press, 1999.

[11] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, 89(2-3):114–141, February 2003.

[12] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17:167–174, 1998.

[13] P.W. de Bruin, V.J. Dercksen, F.H. Post, A.M. Vossepoel, and G.J. Streekstra. Interactive 3D segmentation using connected orthogonal contours. *Computers in Biology and Medicine*, 35:329–346, 2005.

[14] Lee Raymond Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[15] Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27(3):203–216, 2007.

[16] Franca Foppiano, Claudio Fiorino, Giovanni Frezza, Carlo Greco, and Riccardo Valdagni. The impact of contouring uncertainty on rectal 3D dose-volume data: results of a dummy run in a multicenter trial. *Int J Radiat Oncol Biol Phys*, 57(2):573–9, 2003.

[17] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20(10):693–702, 1977.

[18] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, 2005.

[19] Leo Grady. Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1768–1783, 2006.

[20] Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. Interaction techniques for 3d modeling on large displays. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 17–23. ACM, 2001.

[21] Ghassan Hamarneh, Johnson Yang, Chris McIntosh, and Morgan Langille. 3d live-wire-based semi-automatic segmentation of medical images. In *SPIE Medical Imaging*, volume 5747, pages 1597–1603, 2005.

[22] Frank Heckel, Olaf Konrad, Horst Karl Hahn, and Heinz-Otto Peitgen. Interactive 3D medical image segmentation with energy-minimizing implicit functions. *Computers & Graphics: Special issue on Visual Computing for Biology and Medicine*, 35(2):275–287, 2011.

[23] Michelle Holloway, Cindy Grimm, and Tao Ju. Template-based surface reconstruction from cross-sections. *Computers & Graphics*, 58:84 – 91, 2016. Shape Modeling International 2016.

[24] Michelle Holloway, Anahita Sanandaji, Deniece Yates, Amali Krigger, Ross T. Sowell, Ruth West, and Cindy Grimm. Guided structure-aligned segmentation of volumetric data. In *Advances in Visual Computing - 11th International Symposium, ISVC 2015, Proceedings, Part I*, pages 307–317, 2015.

[25] Hidekata Hontani, Takamiti Matsuno, and Yoshihide Sawada. Robust nonrigid icp using outlier-sparsity regularization. In *CVPR*, pages 174–181, 2012.

[26] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1449–1457, 2008.

[27] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, May 1999.

[28] Tao Ju, Joe D. Warren, James Carson, Gregor Eichele, Christina Thaller, Wah Chiu, Musodiq Bello, and Ioannis A. Kakadiaris. Building 3d surface networks from 2d curve networks with application to anatomical modeling. *The Visual Computer*, 21(8-10):764–773, 2005.

[29] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1):2–11, 1975.

[30] Azam Khan, Igor Mordatch, George Fitzmaurice, Justin Matejka, and Gordon Kurtenbach. Viewcube: a 3d orientation indicator and controller. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D '08, pages 17–25. ACM, 2008.

[31] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM Trans. Graph.*, 30(4):79:1–79:12, July 2011.

[32] Regis Kopper, Tao Ni, Doug A. Bowman, and Marcio Pinho. Design and evaluation of navigation techniques for multiscale virtual environments. In *Proceedings of the IEEE conference on Virtual Reality*, VR '06, pages 175–182. IEEE Computer Society, 2006.

[33] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1482–1489, 2005.

[34] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1421–1430, 2008.

[35] Yaron Lipman and Thomas Funkhouser. MÖbius voting for surface correspondence. *ACM Trans. Graph.*, 28(3):72:1–72:12, July 2009.

[36] Lu Liu, C. Bajaj, Joseph Deasy, Daniel A. Low, and Tao Ju. Surface reconstruction from non-parallel curve networks. *Comput. Graph. Forum*, 27(2):155–163, 2008.

[37] I. Macedo, J.P. Gois, and L. Velho. Hermite interpolation of implicit surfaces with radial basis functions. In *Computer Graphics and Image Processing (SIBGRAPI), 2009 XXII Brazilian Symposium on*, pages 1–8, Oct 2009.

[38] J.-S. Prassni, T. Ropinski, and K. Hinrichs. Uncertainty-aware guided volume segmentation. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1358–1365, Nov 2010.

[39] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[40] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, 2004.

[41] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 109–116, 2007.

[42] Ross Sowell, Lu Liu, Tao Ju, Cindy Grimm, Christopher Abraham, Garima Gokhroo, and Daniel Low. VolumeViewer: an interactive tool for fitting surfaces to volume data. In *SBIM '09: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 141–148. ACM, 2009.

[43] Ross Taylor Sowell. *Modeling Surfaces from Volume Data Using Nonparallel Contours*. PhD thesis, Washington Univ. in St. Louis, 2012.

[44] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 265–272. ACM, 1995.

[45] Gary K. L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C. Langbein, Yonghuai Liu, A. David Marshall, Ralph R. Martin, Xianfang Sun, and Paul L. Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Trans. Vis. Comput. Graph.*, 19(7):1199–1217, 2013.

[46] Gary K. L. Tam, Ralph R. Martin, Paul L. Rosin, and Yu-Kun Lai. Diffusion pruning for rapidly and robustly selecting global correspondences using local isometry. *ACM Trans. Graph.*, 33(1):4:1–4:17, 2014.

[47] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, pages 356–369, 2010.

[48] Andrew Top, Ghassan Hamarneh, and Rafeef Abugharbieh. Active learning for interactive 3d image segmentation. In *Proceedings of the 14th International Conference on Medical Image Computing and Computer-assisted Intervention - Part III*, pages 603–610, 2011.

[49] Andrew Top, Ghassan Hamarneh, and Rafeef Abugharbieh. Spotlight: Automated confidence-based user guidance for increasing efficiency in interactive 3d image segmentation. In *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging*, volume 6533 of *Lecture Notes in Computer Science*, pages 204–213. 2011.

[50] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 1999.

[51] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Danial Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[52] Elisabeth Weiss, Susanne Richter, Thomas Krauss, Silke I Metzelthin, Andrea Hille, Olivier Pradier, Birgit Siekmeyer, Hilke Vorwerk, and Clemens F Hess. Conformal radiotherapy planning of cervix carcinoma: differences in the delineation of the clinical target volume. A comparison between gynaecologic and radiation oncologists. *Radiother Oncol*, 67(1):87–95, 2003.

[53] Oliver Wirjadi. Survey of 3D image segmentation methods. Technical Report 123, Fraunhofer ITWM, 2007.

[54] Guoliang Xu, Qing Pan, and Chandrajit Bajaj. Discrete surface modelling using partial differential equations. *Computer Aided Geometric Design*, 23(2):125–145, 2006.

[55] Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. Topology-constrained surface reconstruction from cross-sections. *ACM Trans. Graph.*, 34(4):128:1–128:10, 2015.

Vita

Michelle Holloway

Degrees

B.S. Summa Cum Laude, Computer Engineering, May 2010
B.S. Summa Cum Laude, Computer Science, May 2010
Ph.D. Computer Science, November 2016

Publications

In Progress: Michelle Holloway and Tao Ju. **Resolving inconsistencies in non-parallel contour networks**.

Michelle Holloway, Cindy Grimm, and Tao Ju. **Template-based surface reconstruction from cross-sections**. In: *Computers & Graphics*, 58:84-91, 2016. Shape Modeling International 2016.

Michelle Holloway, Anahita Sanandaji, Deniece Yates, Amali Krigger, Ross Sowell, Ruth West, and Cindy Grimm. **Guided structure-aligned segmentation of volumetric data**. In *Advances in Visual Computing - 11th International Symposium, ISVC 2015, Proceedings, Part 1*, pages 307-317, 2015.

R. West, M. Kajihara, M. Parola, K. Hays, L. Hillard, A. Carlew, J. Deutsch, B. Lane, M. Holloway, B. John, A. Sanandaji, and C. Grimm. **Eliciting tacit expertise in 3d volume segmentation**. In *Proceedings of the 9th International Symposium on Visual Information Communication and Interaction*, VINCI '16, pages 59-66. ACM, 2016.

Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. **Topology-constrained surface reconstruction from cross-sections**. *ACM Trans. Graph.*, 34(4):128:1-128:10, 2015.

Sebastian Kurtek, Jingyong Su, Cindy Grimm, Michelle Vaughan, Ross Sowell, and Anuj Srivastava. **Statistical analysis of manual segmentations of structures in medical images**. *Computer Vision and Image Understanding*, 117(9):1036-1050, 2013.

December 2016