1-1-2007

Sketchsurfaces : sketch-line initialized deformable surfaces for efficient and controllable interactive 3D medical image segmentation

Meisam Aliroteh
Ryerson University

SKETCHSURFACES: SKETCH-LINE INITIALIZED DEFORMABLE SURFACES FOR EFFICIENT AND CONTROLLABLE INTERACTIVE 3D MEDICAL IMAGE SEGMENTATION

by

Meisam Aliroteh
BASc, Electrical and Computer Engineering, University of Toronto, Toronto, 2005

A thesis
presented to Ryerson University
in partial fulfillment of the
requirement for the degree of
Master of Applied Science
in the Program of
Electrical and Computer Engineering.

Toronto, Ontario, Canada, 2007

Author's Declaration

Abstract

Meisam Aliroteh, *SKETCHSURFACES: SKETCH-LINE INITIALIZED DEFORMABLE SURFACES FOR EFFICIENT AND CONTROLLABLE INTERACTIVE 3D MEDICAL IMAGE SEGMENTATION*, MASc, Electrical and Computer Engineering, Ryerson University, Toronto, 2007

This thesis presents an intuitive, fast and accurate interactive segmentation method for visualizing and analyzing 3D medical images. This method combines a general deformable subdivision surface model with a novel sketch-line user initialization process. The model is simply and precisely initialized with a few quick sketch lines drawn across the width of the target object on several key slices of the volume image. The smooth surface constructed using these lines is extremely close to the shape of the object boundary, making the model's task of snapping to this boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. This subdivision-surface based deformable model provides a foundation for precise user steering/editing capabilities and all of the simple, intuitive user interactions are seamlessly integrated with advanced visualization capabilities. Furthermore, to demonstrate its efficiency and accuracy, this new model has been used to segment objects from several 3D data sets.

Keywords: sketch initialization, subdivision surface, medical image segmentation, interactive image analysis, human-computer interaction.

Acknowledgements

I would like to express my sincere gratitude to Dr. Tim McInerney for his invaluable guidance, constant encouragement, and infinite patience during the course of this work. The perfect balance that he maintains between encouraging independent student work and providing scientific advice when needed makes him a great mentor. I feel honoured to have worked under his supervision.

I also would like to thank the members of my thesis committee, Dr. Ling Guan, Dr. Kristiina M. Valter McConville, and Dr. Sridhar Krishnan for their time and effort in reviewing my thesis and their helpful feedback.

Lastly, I would like to express my gratitude to my parents for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image segmentation is commonly defined as the partitioning of an image or a volume image into homogeneous areas, separating regions of interest from the rest. A volume image can be visualized as a stack of 2D images that constitute a 3D volume when assembled. These images can be obtained through various acquisition methods such as Magnetic Resonance Imaging. Segmentation of volume images remains one of the major challenges in medical image visualization and analysis. In the medical imaging field, the extraction of anatomical structures from the background and from each other is a prerequisite to a host of medical image analysis (MIA) tasks, such as measurement, visualization, matching and labelling, reconstruction, and surgical planning. The ultimate goal of medical image segmentation is the development of fully automatic techniques that guarantee maximum repeatability, robustness, accuracy, and efficiency. The complexity and variability of anatomical shapes, the significant variation between images, anatomical structure abnormalities, non-uniform image acquisition, and the sheer size of the data sets have created imposing barriers to the development of robust, efficient, fully automatic medical image segmentation systems. Furthermore, the wide range and different properties of imaging modalities and the lack of image quality often result in fuzzy, indistinct, or disconnected object boundaries. The challenge is to efficiently extract the boundary elements belonging to the target anatomical structure and integrate these elements into a complete and consistent model of that structure. Researchers have struggled to effectively utilize prior knowledge of structure shape, position, orientation, symmetry, relationships to neighbouring structures, associated landmarks, and plausible image intensity characteristics in order to meet this challenge.

On the other hand, manual segmentation via boundary tracing is extremely labour-intensive, time-consuming, and error-prone. Traditional image processing techniques, such as thresholding and region growing, can be effective for a small set of specific segmentation problems. However, they only consider local intensity information and therefore often make incorrect assumptions during the boundary element integration process, resulting in infeasible object boundaries. As a result, these model-free techniques usually require considerable amounts of user intervention and editing. Consequently, a more immediate and significant impact on MIA may be realized by optimizing the capabilities of model-based semi-automatic segmentation techniques, to the point where only a small amount of user intervention is required to process complex data sets. To achieve this, the recognition capabilities of the human visual system must be fully exploited. Model-based semi-automatic techniques that assist the human operator in performing segmentations must be designed to not only be fast and intuitive, but also permit the interactive transfer of structure shape and appearance knowledge from the expert in order to ensure segmentation accuracy, robustness and reproducibility with minimal user editing.

Deformable models such as Snakes, Balloons, and their variants (as further discussed in chapter 3) have been devised in order to assist a human operator in performing segmentation of anatomical structures from 2D and 3D images, respectively. A Snake is an interactive, flexible contour model that is placed in the vicinity of the object of interest, and then iteratively adjusted to fit the boundary of the structure. In contrast, in a Balloon model an initial sphere is placed inside the target object (which can be far from object boundaries) and is then iteratively inflated and deformed in order to capture the shape of the object. Despite the large number of 3D semi-automatic segmentation methods developed over the past decade, no one technique has been widely adopted in clinical practice, and manual delineation and/or simple pixel-based tools are still heavily used. The lack of adoption of these techniques is in part related to the ineffective integration of simple, intuitive, and consistent interaction capabilities, with the necessary precision and power, into the segmentation work-flow. An interactive segmentation method can be simply broken down into two fundamental parts: an algorithmic/computational part, and an interactive part. As described in the scientific literature, many of the current segmentation methods have mostly emphasized the algorithmic and computational part, with little or no attention to the interactive part. These interactive segmentation methods use different strategies to combine the expertise of humans with the computational power of computers; hence, their outcome depends on the proposed interaction strategy as much as on algorithmic computation. As a result, a proper assessment of interactive segmentation methods requires the computational and interactive parts to be equally understood.

All 3D segmentation techniques can fail in noisy images and this failure can result in time-consuming and tedious user intervention. With many of the current techniques [1–7], once the algorithm is initiated, the ability to steer it is limited and/or the ability to edit it is restricted to a separate post processing phase, often with a separate tool-set and/or user actions. That is, while the mathematical formulations and numerical algorithms of these model-based methods enable simple initialization and complex shape extraction, the cost is incurred on the "back-end" of the segmentation process.

Other techniques [8–10] do provide interactive steering capabilities but editing facilities are not well integrated, the tracing actions are tedious and require considerable user concentration, and the 2D contour-based nature of the algorithm forces the user to pay careful attention to how it is applied and to mentally reconstruct the 3D shape of the object as the algorithm is applied.

1.1 Contributions of this Thesis

The deficiencies of user control functionality, simplicity of use, and the lack of interaction precision and intuitiveness of current methods have led to the exploration of an alternative research direction in 2D [11], creating a user-friendly interactive contour model suitable for segmenting medical images and image sequences which exhibit a significant amount of noise. This thesis extends the proposed approach of [11] to 3D segmentation. More specifically, the goals of this thesis are to provide a highly accurate initialization of a deformable surface model using fast, simple, user actions that require little concentration. After initialization, it should be visually apparent to the user what the resulting segmentation will look like. In addition, this thesis seeks a robust, noise-insensitive shape model that provides the foundation for precise, efficient, and intuitive steering and editing capabilities, and a method that requires little or no parameter tweaking and/or mode changes. Most importantly, all the interaction capabilities must be seamlessly "woven" together providing simple, consistent user interactions and complete user control throughout the entire segmentation work-flow.

The result of this thesis is the development of SketchSurfaces - a highly controllable subdivision surface based deformable model with a simple formulation that is coupled with a novel sketch-based 3D initialization method. SketchSurfaces offer "fluid", consistent transitions between initializing, fitting, steering, editing, visualizing, and zooming. The user is able to stop the segmentation at any time, examine the current initial surface model or partial segmentation result, zoom in and out, and make precise corrections before continuing, all with simple mouse actions.

Using SketchSurfaces, in order to segment an anatomical structure, the user starts by navigating through the volume image slices to one end of the target object. To perform this navigation, SketchSurfaces provide the user with both 2D and 3D image slice views (Fig. 1.1 a). As the user positions and orients a slice plane in the 3D view, the 2D view is updated in real-time, always showing a bird's-eye view of the 3D plane for enhanced visualization. Once the image slice is positioned at one end of the object, the user initializes a surface by sketching a few lines across the width of the target object, creating a contour that approximates its boundary (Fig. 1.1 b–e). This can be done on either the 2D or the 3D view. The user then proceeds to another slice plane by pushing the plane towards the other end of the object, and repeats this sketching process. After each sketch line process, the previous contours are automatically connected and converted into a subdivision surface (Fig. 1.1 f–j). This simple initialization process leads to the construction of an initial surface that is extremely close in shape to the target object. This makes the model's task of snapping to the object boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. Once the initial surface is constructed, the user will then proceed to segment the target object by fitting the initialized surface to the object boundaries (Fig. 1.1 k,l). After the fitting step, if the shape of the segmented object is not satisfactory, the user can perform more fitting steps or directly edit the shape of the segmented surface.

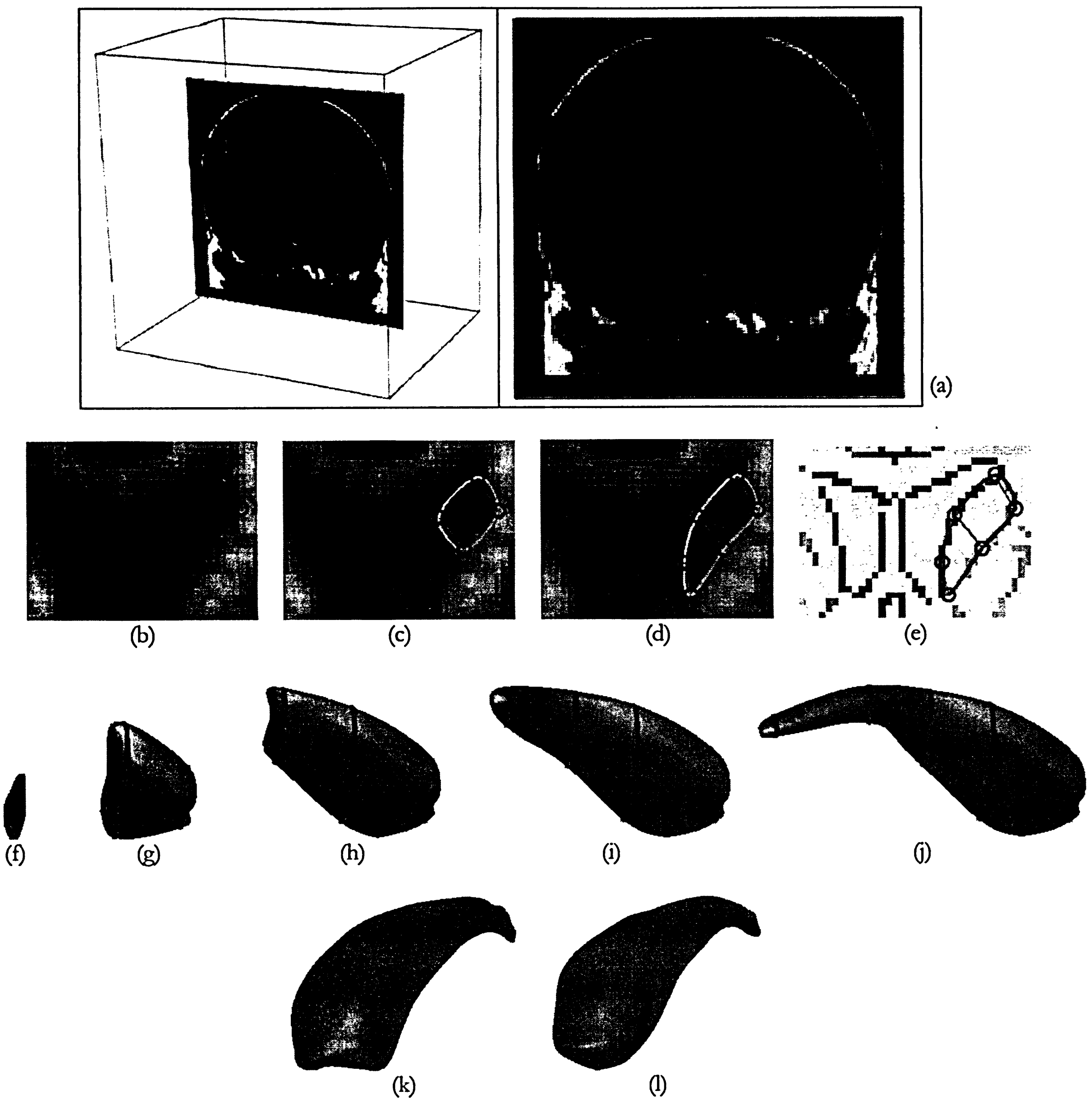

Figure 1.1: SketchSurfaces segmentation of the right caudate nucleus from an MR volume image. The user positions the image slice plane at one end of the caudate, either in a 3D view ((a) left) or a 2D view ((a) right). The user then sketches a few lines across the width of the caudate, creating a contour that approximates its boundary (b – d). The user may also view and operate on an edge detected image (e). This process is repeated for a few slice planes. After each sketch-line process, the previous contours are automatically connected and converted into a subdivision surface (f – j). The curves in f–j highlight the slice plane locations where the user has performed the sketching steps. In this example, the initial surface for the caudate was constructed using five slice planes. The initialized surface (k) is then fitted to the boundaries of the caudate in order to obtain a segmentation result (l).

1.2 Thesis Outline

Chapter 2 starts by providing a number of evaluation criteria that should be used as the requirements for the design of any interactive segmentation technique, providing the motivation for designing a new approach to interactive 3D segmentation that adheres to these guidelines. Next, in Chapter 3 a number of currently popular segmentation approaches are evaluated using these criteria, further demonstrating the need for a new approach with a better design. Chapter 4 presents a comprehensive review of semi-automatic deformable models, their use for medical image segmentation, and their mathematical foundations. Moving on to Chapter 5, the basics of subdivision surfaces are covered and the Modified Butterfly subdivision scheme is introduced. Chapter 6 introduces SketchSurfaces – a new approach to interactive 3D medical image segmentation – and some of the experimental results that were obtain using this approach are presented in Chapter 7. Finally, Chapter 8 presents some concluding remarks along with recommendations for future research.

Chapter 2

Evaluation Criteria and Motivation for a New Approach to Interactive 3D Segmentation

The objective evaluation of interactive segmentation methods is a rather difficult issue. The evaluation depends on the task, where the performance of a method may be considered reasonable for one application and not acceptable for another. In the literature, however, the main evaluation criteria that have been adopted consistently to measure the capabilities of interactive segmentation methods are accuracy, repeatability, and efficiency. In addition, since the ability to effectively steer and edit the segmentation affects all three criteria, these two capabilities warrant a separate treatment and are included as part of the evaluation criteria. Together these criteria offer a reasonable starting point to evaluate interactive segmentation methods.

2.1 Accuracy

The most common evaluation criterion is accuracy, indicating the degree to which the delineation of the object corresponds to the ground truth. To measure the accuracy of a method, the generated result of the segmentation tool is often compared against a ground truth obtained by allowing a human expert to perform the segmentation manually. A common problem of this approach is the bias factor introduced by the human experts. Different experts can segment the same object from the same data set differently, leading to a different set of ground truth results. To mitigate this issue, currently in the literature an averaging of the ground truth results is performed.

It should be noted that accuracy as a criterion is most applicable in the evaluation of segmentation results generated by fully automatic processing. In interactive methods, user participation is included in the process to improve accuracy to the point where the result obtained is "always" satisfactory. The only situation where this is not true occurs when user control is limited. Thus, an interactive method is potentially accurate when it provides full control to the user to generate any desired result. Nevertheless, achieving the desired accuracy through user interaction can affect the efficiency of the algorithm, especially when considerable user interaction is required.

2.2 Repeatability

Repeatability may be defined as the extent to which the same result would be produced over different segmentation sessions when the user has the same intention. In this case, the same image and object are segmented several times by one human operator and the results are compared. The same procedure is followed to assess the inter-operator repeatability. The differences indicate the intra-operator or inter-operator variability of results.

The variation of the results can be caused either by the difference in the operation of the segmentation tool (different initialization criterion for example) or by the difference in user judgment. A method potentially generates repeatable results when it takes precautions to minimize the effect of the first type of variation. Nothing can be done about the second type.

2.3 Efficiency

As suggested by [12], efficiency of a segmentation technique can be separated into two parts: efficiency of the computational part, and efficiency of the interactive part. Efficiency of the computational part is measured in terms of the time needed by the computer to generate the result. Computation should be fast enough to allow for interaction in real-time. With respect to the interactive part, efficiency is inversely proportional to the effort required from the user to accomplish the segmentation task. This effort is determined mostly by the amount and the nature of user interventions. The amount of interaction depends on the autonomy of the computational part, and it is often estimated in terms of the number of mouse clicks (or any other input device used). As for the nature of interaction, it is necessary to evaluate the complexity of the task performed by the user. Task complexity involves several issues, among them the demand posed on operation of the input device, the type of knowledge needed to input data during interaction, and the predictability of the method's behaviour in response to user input.

In conclusion, the evaluation of efficiency of interactive methods is mostly subjective and measuring total elapsed segmentation time should not be the definitive indicator. In general terms, it seems reasonable to say that an interactive method is potentially efficient when the computational part is fast, highly autonomous and predictable, and when user interventions are few, quick and simple. At any rate, the impact of complex user interventions is likely to be reduced or eliminated over time as the user learns to operate the segmentation tool.

2.4 Steering

Steering can be defined as the method by which the segmentation algorithm is driven towards the desired solution via the user interactions. As an example, the dragging of a Snake contour model towards desired image features or the dynamic creation of attracting spring forces pulling a Snake towards a fixed point can be considered as steering mechanisms. Steering mechanisms can have a great impact on the accuracy and efficiency of the segmentation tool. Since efficiency depends greatly on the amount and the nature of user interaction, a segmentation tool that provides an easy and intuitive method for guiding the algorithm would be highly desirable.

2.5 Editing

At first glance, editing and steering criterions may appear to be the same. Nevertheless, the steering mechanism defines the method by which the user interacts with and guides the algorithm towards a desirable solution, whereas the editing mechanism defines the method by which the user can fix any errors that the algorithm makes after reaching a temporary solution. In other words, steering is typically considered as a pre-segmentation operation and editing as a post-segmentation operation.

Similar to steering, the method by which the user can edit the segmentation result also impacts the efficiency and accuracy of the segmentation tool. Since all of the algorithms that have been developed to date produce some amount of error in their results, it is absolutely necessary for a segmentation tool to provide a simple and effective way to edit the obtained results. An algorithm can provide a very simple steering mechanism, such as a few mouse clicks to indicate where the object of interest is; however, if the segmented result is not accurate enough and if the editing mechanism is not simple and intuitive, correcting the erroneous result can be tedious and fatiguing, which negatively impacts the efficiency of the segmentation tool.

2.6 Motivation for a New Approach to Interactive 3D Segmentation

Upon extensive review of recent approaches to the 3D interactive segmentation of volume images, it can be noted that currently there does not exist a technique which effectively combines all of the above criteria to achieve a highly flexible, efficient, and user friendly tool. The five aforementioned evaluation criteria should be set as the requirements for the design of any interactive segmentation technique. More specifically, the following design principles should be followed to produce an efficient interactive segmentation method that generates accurate and repeatable results [12]:

1. Efforts should be put into seamlessly combining the computation and the user interaction processes.

2. Whenever input is required to the computational part, the input should be provided in pictorial form as much as possible.

3. The nature of user interactions should be such that the interactions can be carried out effortlessly and effectively.

4. Users should initialize the segmentation method with key information which will lead the method to an accurate result more quickly.

5. User control should be maintained throughout the entire process in order to obtain accurate results.

6. Proper visualization of the computational part is needed to enable an effective user response.

The next chapter introduces some of the currently popular segmentation techniques, which are then evaluated based on the above criteria to further emphasize the need for a better interactive 3D segmentation tool. The goal of this thesis is to develop a new tool that adheres to these principles. Throughout this thesis, the new approach – known as SketchSurfaces – is explained in detail. A few experimental segmentation results are then presented, further emphasizing the strengths of this new technique.

Chapter 3

Current Approaches to Model-Based Semi-Automatic 3D Medical Image Segmentation

This section presents and evaluates four main categories of approaches to 3D medical image segmentation. It should be noted that many researchers have implemented variations of the main algorithm for each of these approaches. Only a few of the more commonly used variations of the main algorithms have been picked from each approach and the following evaluations have been based upon these algorithms.

3.1 Slice-By-Slice Approach

Traditional boundary extraction methods, such as thresholding and region growing, can be effective for a small set of specific segmentation problems. However, they only consider local intensity information and therefore are sensitive to noise and sampling artefacts, making it difficult to generate closed, connected boundary surfaces. As a result, these model-free techniques usually require considerable amounts of user intervention and editing.

In contrast, deformable shape models, which include the popular Snakes [13] and deformable surfaces [14], have proved a powerful technique for the extraction of boundaries from medical images by combining the bottom-up approach of edge detection with the top-down approach of model-based geometric constraints. The model-based approach provides several desirable features such as inherent connectivity and smoothness that counteract noise and boundary irregularities, compact and analytic object representations, and the ability to incorporate prior knowledge of expected anatomic shape [3, 15].

There are two possible approaches for extracting the boundaries of anatomical structures from volumetric medical images using deformable models. One is a slice-by-slice approach using 2D deformable contours. Starting with an initial image slice, a Snake model is applied to extract the boundary contour of the structure. The resulting Snake is then propagated to neighbouring slices and used as an initial contour in these slices. This process is repeated until the entire 3D boundary surface is represented as a sequence of 2D contours generated from all slices which contain the object [16–18]. However, this approach often causes discontinuities or inconsistencies between neighbouring slices and has difficulties extracting the contours near the first and last image slices bounding the anatomic structure. A fair amount of user interaction is required on many slices as the Snake deforms in order to "pull" it out of an incorrect solution. Furthermore, as the resolution of the volume images increases with advances in imaging technology, slice by slice approaches become increasingly inefficient. The other approach is to extract the entire boundary surface of the structure all at once using a true 3D deformable surface model or "balloon" as discussed in the next section.

3.2 Deformable Balloon Models

As previously mentioned, one of the most well known 2D deformable models is the Snake model [13], which has been used as the basis for many of the 3D deformable models. Snakes are energy-minimizing contours controlled by internal and external energies. The internal energy imposes a smoothness constraint. The external energy terms, defined by scalar potential functions, couple the Snake to the image. These functions are defined such that the minimum of an external energy term represents salient image features, such as image edges, or user-defined constraint points. A common approach to minimizing the Snake energy is to transform the energy equation into a partial differential equation (PDE) representing force-based equations of motion and to use an iterative procedure to solve this PDE and bring the Snake into equilibrium (i.e. where the internal smoothness forces balance the external image forces and user-defined forces). If good initial conditions and well-defined potential functions are used, when this equilibrium is achieved the Snake will have converged to the boundary of the target object in the image.

Some of the popular 3D extensions of the Snake model are the various deformable elastic "*Balloon*" models [1–7], a subset of which also have the ability to dynamically change their topology [1], [3], [7]. These models provide a "one-click" initialization process in which the user performs the initialization step by clicking a point inside the object of interest. A small sphere (or balloon) is then initialized at that location and *inflates* (via expansion forces) to take on the shape of the target object (Fig. 3.1).

In the implementation provided by Miller [19], a polygonal approximation to a sphere (or "balloon") is constructed and geometrically deformed until the balloon surface conforms to the object surface in 3D CT data. The segmentation process is formulated as the minimization of a cost function where the desired behaviour of the balloon model is determined by a local cost function associated with each model vertex. The cost function is a weighted sum of three terms: a deformation potential that "expands" the model vertices towards the object boundary, an image term that identifies features such as edges and opposes the balloon expansion, and a term that maintains the topology and smoothness of the model by constraining each vertex to remain close to the centroid of its neighbours.

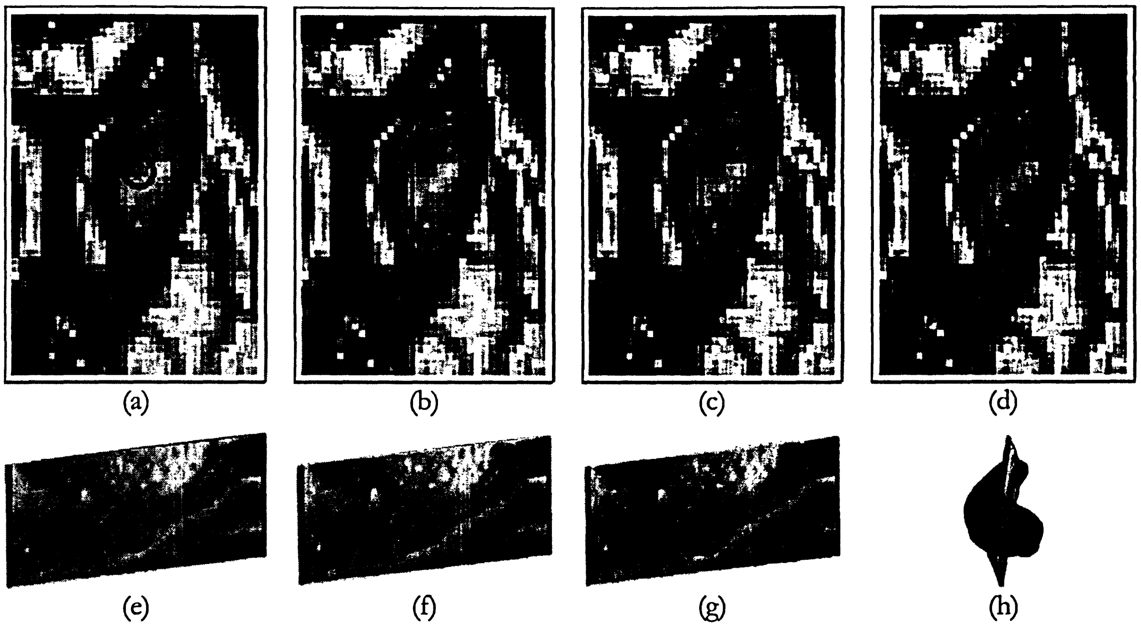
Figure 3.1: Segmentation of anatomical structures using the Balloon method. Examples of edge-based contour evolution (a – d). Segmentation of left hippocampus from MRI by initializing two balloons (e). The result of surface evolution after 6 iterations (f) and 18 iterations (g) and the final segmentation with a rotated view (h). (From [7]).

Cohen and Cohen [17, 20] and McInerney and Terzopoulos [21] use finite element and physics-based techniques to implement an elastically deformable cylinder and sphere, respectively. These models are used to segment the inner wall of the left ventricle of the heart from MR or CT image volumes. These deformable surfaces are based on a thin-plate under tension surface spline which controls and constrains the stretching and bending of the surface. The models are dynamically fitted to data using Lagrangian equations of motion in order to adjust the deformational degrees of freedom. Furthermore, the finite element method is used to represent the models as a continuous surface in the form of weighted sums of local polynomial basis functions. Unlike Miller's [22] polygonal model, the finite element method provides an analytic surface representation over the whole model surface and the use of high-order polynomials means that fewer elements are required to accurately represent an object.

As mentioned previously, researchers have created balloon models that automatically subdivide and change their topology. These models can flow into complex shapes, such as arterial trees, and/or disconnect and reconnect to take on object shapes that contain holes. To achieve this ability, many researchers have constructed implicit deformable models by adopting Osher and Sethian's [23] level-set evolution technique to the image segmentation problem (see [24] for a complete review). These models are formulated as evolving surfaces ("propagating fronts") which define the level set of some higher-dimensional function. The main feature of this approach is that topological changes are handled naturally, since the level set need not be simply connected; the higher-dimensional surface remains a simple function even as the level set changes topology. The inflation forces of the above techniques significantly increased their capture range. In this case, the model can be simply initialized using, for example, one mouse click to create a small sphere inside the target object. The inflation force expands the model and it automatically subdivides, allowing it to "flow" into complex shapes. As the model approaches the boundary of the target object, external image forces (based on edge strength, image region statistics, or area minimization) oppose the inflation, stopping the model on the boundary.

Implicitly-defined balloon methods have proved effective for some segmentation tasks, especially involving extremely complex-shaped objects, for example the cortex of the brain. These models work well in segmentation scenarios where the image feature map is relatively clean and homogeneous. However, clinical images are often noisy, contain many uninteresting edges and regions of low contrast, contain gaps in the object boundary, or exhibit a complex texture. Hence, these more automatic techniques may not generate the expected result - the added automation does not come without a cost. Some interactive control (steering capability) over the model is lost and the gaps in the object boundaries may allow the model to leak through, requiring user intervention in the form of barriers. Another common problem with Balloon models is they may not completely flow into all regions of the target object. In this case, the user is required to plant other seeds and re-run the algorithm, or provide additional constraint information to force the model into these regions. Editing in these techniques is typically performed in a post processing phase, often using a separate tool-set. In general, the user interaction model was not the focus of the design of these methods - the user does not have complete control during the entire segmentation process (especially during the inflation phase) and everywhere on the model and this lack of control can affect the user's experience with the tool, resulting in segmentation inefficiencies. In other words, most of the implementations of the Balloon models violate the design principles 1, 3, 4, and 5, as outlined in section 2.6.

3.3 Semi-automatic Boundary Tracing

Subsequent to the introduction of Snakes, a related technique, known as LiveWire or Intelligent Scissors [25–30] has emerged as an effective interactive boundary tracing tool which allows user interaction and control over the 2D segmentation process. Adobe Photoshop's image cut-out tool [31] Magnetic Lasso is an example of this type of algorithm. Similar to the Snakes, the idea behind the LiveWire technique is to perform the segmentation with minimal user interaction while at the same time allowing the user to guide or "steer" the segmentation process. In contrast to the Balloon models where the initial seed surface can be initialized far from the object boundary, in this technique the user utilizes the mouse to initially specify a seed point *on the object boundary* and then moves the mouse to advance the cursor to a point further along the object boundary. A globally optimum path (the trace) from the initial seed point to the current point is computed and displayed in real time. The optimal paths are determined by assigning a set of features and cost functions to boundary elements (such as edge strength), and then finding the minimum cost path. As the user moves the cursor slightly, different paths are computed and displayed in real-time, akin to an electrical arc - hence the name "LiveWire". If the cursor moves close to the boundary, the LiveWire snaps to the boundary (assuming the cost functions are set correctly). If the user is satisfied with the computed boundary segment, the user "deposits" the cursor point. This point becomes the new seed point and the recursive process continues.

A variety of 3D extensions of the LiveWire technique have been developed/applied by several research groups [8–10, 32]. In general the initialization of these 3D approaches is done by tracing the contour of the object of interest on a selected set of slices using the 2D LiveWire method. Various techniques are then applied to use these initialized contours and automatically generate the contours of the object on the unseen slices. For example, in the method presented by Schenk *et al.* [32], shape-based interpolation and adaptive propagation allow the automatic approximation of contours on slices between user-defined boundaries. Hamarneh *et al.* [10] also propose a 3D extension of the LiveWire approach in which the initial points are calculated from intersections of user-based LiveWire techniques with new slices. In this technique, the 3D live-wire extension requires that the user first traces out a few initial contours using 2D live-wire in slices of their choice. It is recommended that the initial contours be distributed in the volume such that they capture the topological features of the target structure. The specific points along the initial contours will then be used as seed and target points for automatically generating additional orthogonal live-wire contours. Each new slice is tested for intersection with the user defined contours and an automatic contour is generated (Fig. 3.2).

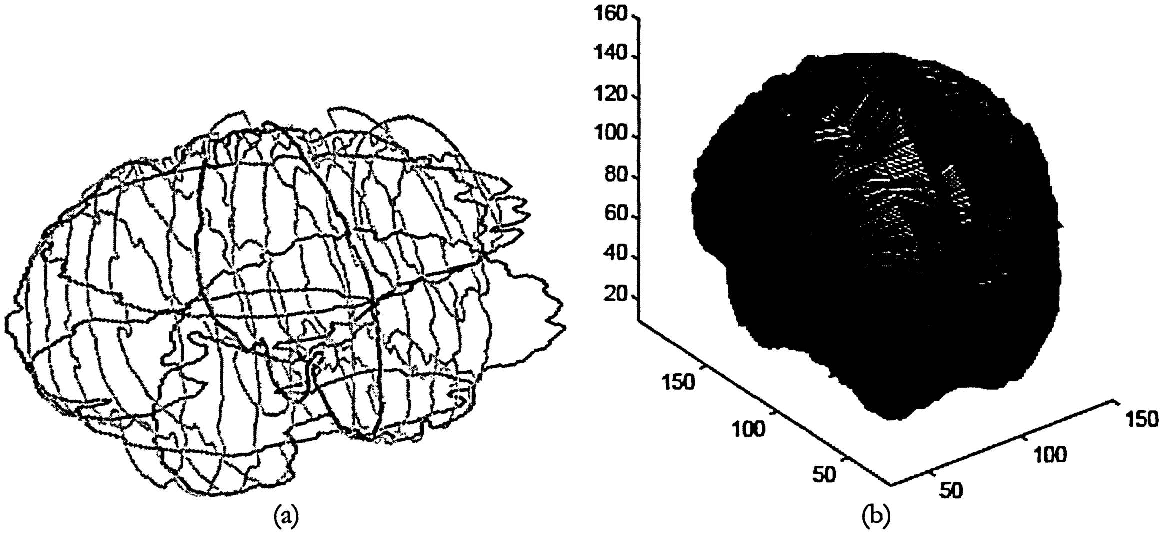

Figure 3.2: Example of 3D LiveWire segmentation of the cortical surface. The user starts by tracing the contour of the target object in a few slices (a), shown as dark curves. From these contours the algorithm will automatically calculate and detect contours on other slices, shown as light curves. The final extracted cortical surface is shown in (c). (From [10]).

Although LiveWire methods are much faster and more reproducible than manual tracing of object boundaries and other traditional pixel selection tools, and provide the user with good control during the segmentation process, these techniques can still demand a large amount of concentration from the user, especially in 3D. Once again, the 3D user interaction model was not the focus of the design of this tool, but rather it is fundamentally an extended 2D technique. A few problematic scenarios are:

1. Once the user deposits the curser point when tracing the boundary, the point is collected as a seed point and the trace is "frozen" and added to the extracted object boundary. The user has no further control over the trace other than returning to the previous point to remove it. This type of correction increases segmentation time and user interaction when dealing with a noisy and low contrast object or complex boundary. There is never a perfect match between the features used by the LiveWire algorithm and the desired object boundary. As a result, the user often must control the mouse carefully. If a mistake is made, the user must "backtrack" and try again.

2. LiveWire still requires tracing actions to position the cursor. Moving the cursor around an entire object under the control of a mouse (or other input devices) can be tedious and fatiguing, especially for a complex-shaped object.

3. When the desired object boundary has a relatively weak edge close to an insignificant but strong edge, the LiveWire snaps to the strong edge rather than the desired weak boundary. Falcão and Udupa [29] developed a technique called "LiveWire on-the-fly" in an attempt to minimize this problem but it assumes that edge characteristics are relatively consistent along the entire object boundary.

4. Objects which are complex in shape (contain highly curved regions or protrusions for example) often force the user to deposit many seed points.

5. When segmenting a 3D image, the 2D contour-based nature of the algorithm forces the user to pay careful attention to how it is applied and to mentally reconstruct the 3D shape of the object as the algorithm is applied.

6. In some LiveWire systems [31], any segmentation errors must be cleaned up with other traditional editing tools.

Although these approaches tend to produce accurate results, their accuracy depends greatly on the preliminary position of the initial contours. To ameliorate the sensitivity to the preliminary position of the initial contours, these methods allow for editing of the result by adding more contours to refine the segmentation. Nevertheless, adding contours using LiveWire implies additional tracing and user concentration, leading to fatigue. Similar to the Balloon models, the 3D LiveWire techniques also violate a few of the design principles, such as principles 1, 3, and 5.

3.4 Region Painting and Graph Cuts

More recently, researchers have developed semi-automatic volume/region painting methods for image segmentation. In this technique the user will first mark some regions of the volume, indicating the object of interest (foreground) and the background. There are various approaches to do this first step, as presented by [33–40]. The most common approach is to indicate these regions either by a few mouse clicks or by simple brush strokes in these areas.

An optimization algorithm then uses these inputs hints to extract the actual object boundary. To do this, a graph is formed by connecting all pairs of neighbouring image pixels (or volume voxels) using weighted edges, where the prior identified regions are used to provide necessary clues about the image content. The objective is then to find the least costly way to cut the edges in the graph (min-cut) so that the foreground regions are completely isolated from the background regions. If the edge cost is a decreasing function of the local intensity gradient then the minimum cost cut will produce object/background segmentation with compact boundaries along the high intensity gradient values in the image. An efficient, globally optimal solution is possible via standard min-cut algorithms for graphs with two terminals [38].

Similar to the LiveWire approach, when segmenting objects in 3D medical volume data using Graph Cuts, the user will first need to provide object/background regions on a selected number of 2D slices. The object and the background are then segmented in each of these slices, and these 2D segmentation results provide constrains for 3D segmentation in the volume dataset, allowing for the segmentation of the object from the volume. A different approach is that of [40], where the user directly operates on the rendered 3D images. In [40], the authors suggest that operating on single slices provides very limited information on the structure of a dataset, whereas the full rendering of the volume illustrates the overall 3D structure of the data. This provides an "overview" of the data, on top of which the user can select objects of interest or remove unwanted objects. However, this approach requires a pre-processing stage and the selection of a well suited transfer function in order to clearly display the internal structures simultaneously – a task which is currently not possible for most objects of interest in clinical (i.e. noisy) medical volume images.

The initialization and steering of these approaches are simple and user friendly. In contrast with the LiveWire approach, the tracing effect is eliminated by allowing the user to identify foreground and background regions using simple brush strokes or mouse clicks, which does not require too much user attention and concentration. However, these Graph Cuts techniques share a similar set of problems with the Balloon models. These techniques seem to work well on fairly clean images with bright homogeneous objects such as bone in a CT scan. For noisier objects these techniques can, like the balloon models, leak through or not flow completely into the object, requiring much user editing and intervention. To edit the segmentation results, the user should provide more information to the algorithm by identifying more foreground and background regions, or use a conventional cut-out tool by enclosing a region and cutting it away (Fig. 3.3). This approach is not user friendly since the user does not have "absolute" control over the segmentation result, and this drawback has a negative impact on the accuracy and efficiency of this approach as explained in section 2.1.

The evaluation of the efficiency of the Graph Cuts methods is a difficult and subjective task. On one hand the simple initialization and steering steps will help reduce the initialization time, particularly when compared to the LiveWire methods. On the other hand, due to the insufficient user control during the editing step, the editing time can significantly increase, especially in the case when the object of interest has weak boundaries or the image is noisy, causing erroneous results in the segmentation. The user will need to keep marking more and more regions in order to improve the result. [40] has integrated an additional editing tool in their algorithm, where the user can mark an object (by drawing an ellipse around it) and delete the object. This technique may be useful when dealing with noisy images. Noisy images can cause the Graph Cuts methods to include unwanted objects in their results, and using the above technique these unwanted objects can be removed. Nevertheless, this technique is only effective for removing segmented objects and not editing their boundaries. In conclusion, the Graph Cuts techniques also violate the design principles and guidelines of a semi-automatic segmentation method, particularly principles 4 and 5.

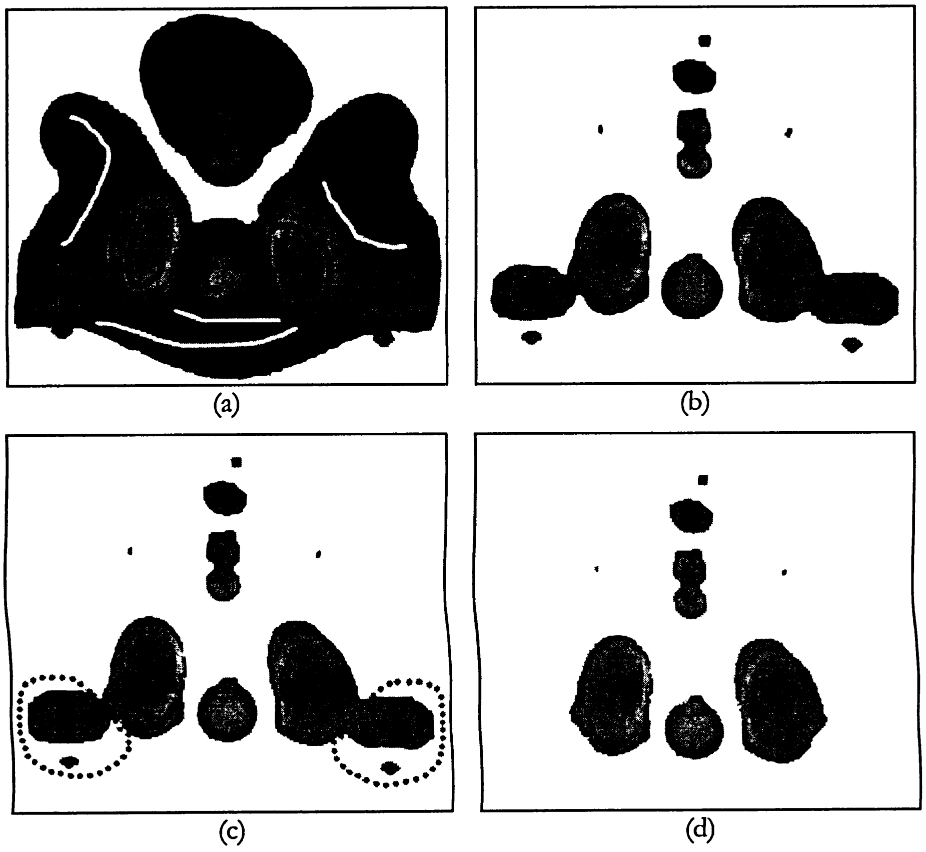

Figure 3.3: Graph Cuts segmentation of a synthetic data set (a Neghip). The user provides input to the algorithm by marking parts of the foreground (dark strokes) and parts of the background regions (light strokes) (a). The Graph Cuts algorithm is then used to segment the volume (b). To edit the result, the user can either provide more input by marking more regions, or select a region to be deleted (c) and cut out the erroneously segmented parts (d). (From [40]).

Chapter 4

Deformable Surface Models

Deformable surface models are the 3D extension of 2D active contour models, the most popular version of which is known as Snakes. Snakes were first introduced in 1988 by Kass, Witkin, and Terzopoulos [13] as a semi-automatic segmentation technique. Snakes have been successfully applied to many image analysis tasks, such as motion tracking and analysis, matching (labelling, registration), and shape recognition. Similar to a Snake, a deformable surface is essentially an elastic surface that is initialized by the user inside or close to the boundary of the target object. The surface will deform and converge towards the object boundary by minimizing an energy functional or alternatively by solving force-based equations of motion. In this thesis, the simple and flexible force-based formulation has been utilized. Using this approach, the equations of motion typically consist of internal forces controlling the smoothness of the surface and external forces which attract the surface toward image edges of the object boundary or user-defined points. The complete framework of a deformable surface model can be specified by a geometric model representation, internal, external and constraint forces, and the governing equations of motion. The remainder of this chapter presents a brief review of these components.

4.1 Geometric Representation

Deformable surface models are constructed using either continuous or discrete geometric representations. Each of these approaches has its advantages and disadvantages. With discrete representations, the geometry of the model is typically defined with a finite set of points and polygons. Deformable models using discrete representations are simple, efficient, and can be easily locally subdivided to add degrees of freedom in areas where the object boundary exhibits rapid variations or is highly curved. A disadvantage of the discrete scheme is the lack of compactness, the difficulty in controlling the model at multiple scales (and therefore the difficulty in utilizing higher-level mechanisms to intelligently control the model fitting), and the lack of an analytic representation over the whole model. On the other hand, continuous representations, in the context of deformable models, are defined as representations using higher-order local basis functions such as finite elements or B-splines [5] or global basis functions such as Fourier descriptors [41]. They provide a compact, local representation of a surface, converge faster to the solution than discrete deformable models, often are inherently smooth, provide an analytic representation over the whole surface (rather than only at discrete points), and require fewer degrees of freedom for the same level of accuracy as discrete models. These properties are advantageous for segmentation or tracking tasks involving noisy images where the target object boundaries may exhibit significant gaps in the edge image. However, a disadvantage is that the mathematical formulation and software implementation of these types of deformable surface models is often much more complex and less flexible than the discrete models. For example, parametric B-spline patch-based deformable models are unable to represent objects containing holes.

Despite having a discrete representation, SketchSurfaces employ the concept of subdivision surfaces which allows them to take the best of both discreet and continuous representation. As later discussed in chapter 5, subdivision surfaces, such as SketchSurfaces, can be constructed from a control mesh. Subdivision surfaces are inherently smooth and require fewer degrees of freedom to represent an accurate surface. This allows a SketchSurface to have a simple, efficient, and compact representation, and the model can be controlled at multiple scales (levels of subdivision). SketchSurfaces evolve by deforming their underlying control mesh which consists of a few degrees of freedom, allowing these surfaces to converge to the desired solution more quickly. Additionally, they can also be easily subdivided (globally or locally) to add more degrees of freedom (if needed) to better capture the shape of complex objects. Having a discrete representation also allows for a simple software implementation of SketchSurfaces.

4.2 Force-based Formulation

A discrete deformable surface, such as a SketchSurface, is a closed elastic triangular mesh consisting of a set of N nodes, indexed by $i = 0, \ldots , N\text{-}1$, and triangular elements. Each node i has an associated time varying position $X_i(t) = [\, x_i(t), y_i(t), z_i(t)\,]$, along with internal forces $f_i^{\text{int}}(t)$, external forces $f_i^{\text{ext}}(t)$, and constraint forces $f_i^{\text{cst}}(t)$. The behaviour of the mesh nodes is governed by simple first-order ordinary differential equations of motion:

$$\gamma_i \dot{X}_i - w_{\text{int}} f_i^{\text{int}} = w_{\text{ext}} f_i^{\text{ext}} + w_{\text{cst}} f_i^{\text{cst}} \quad \text{.........(4.1)}$$

where:

$\dot{X}_i$ - the velocity of node i

γ_i - a damping coefficient controlling the rate of dissipation of the kinetic energy

w_{int} , w_{ext} , w_{cst} - non-negative constant weights controlling strength of the forces

Equation 4.1 expresses the balance of internal, external and constraint forces when the model rests at equilibrium (i.e. $\dot{X}_i = 0$ for all nodes).

A variety of explicit and implicit numerical solvers can be used to integrate this equation forward through time. For example, a common and simple solver is the explicit first-order Euler method. This method approximates the temporal derivatives with forward finite differences and updates the positions of the model nodes from time t to time $t+\Delta t$ according to the formula:

$$X_i^{t+\Delta t} = X_i^t + \frac{\Delta t}{\gamma_i}\left(w_{\text{int}} f_i^{\text{int}} + w_{\text{ext}} f_i^{\text{ext}} + w_{\text{cst}} f_i^{\text{cst}}\right) \quad (4.2)$$

where:

$X_i^{t+\Delta t}$ - the new position of node i

X_i^t - the current position of node i

Δt - the time step

The explicit Euler method is simple, but it becomes unstable unless small time steps are used, thus impacting segmentation efficiency. SketchSurfaces, therefore, uses the Verlet integration scheme [42]:

$$X_i^{t+\Delta t} = 1.75 X_i^t - 0.75 X_i^{t-\Delta t} + \left(w_{\text{int}} f_i^{\text{int}} + w_{\text{ext}} f_i^{\text{ext}} + w_{\text{cst}} f_i^{\text{cst}}\right)\Delta t \quad (4.3)$$

where:

$X_i^{t+\Delta t}$ - the new position of node i

X_i^t - the current position of node i

$X_i^{t-\Delta t}$ - the previous position of node i

Δt - the time step

The Verlet scheme is much more stable than the Euler method and larger time steps can be used in order to reach convergence faster. Note that the coefficients in equation 4.3 were chosen to introduce considerable drag into the system and ensure the system would quickly settle [43].

4.2.1 Internal Forces

The internal forces perform regularization on the model to maintain some level of smoothness. In the original Snakes formulation, the internal force of each node *i* is composed of two types of forces – a tensile force $\alpha_i(t)$ and a flexural force $\beta_i(t)$:

$$f_i^{\text{int}}(t) = w_\alpha \alpha_i(t) + w_\beta \beta_i(t) \quad (4.4)$$

where w_α and w_β are non-negative constant weights controlling the amount of contributions from tensile and flexural forces respectively. The tensile forces allow the model to behave like a membrane. They control the elasticity or stretchiness of the model at different nodes and attempt to minimize the overall length of the Snake (or area of the surface for a Balloon model). The flexural forces allow the model to behave like a thin plate. They control the bending energy and attempt to minimize the overall contour/surface curvature. A general nonlinear strain energy for a parametric deformable surface is a function of the differential area and curvature at each point [44]. A more practical version of this deformation energy is the linear combination of the membrane and thin-plate functionals [45]. This linearized functional approximates the more general nonlinear strain energy functional for small deformations near the actual minimum (where higher order terms tend to zero), but is well behaved for large deformations and its quadratic form leads to computational benefits. The respective variational derivatives of the membrane and thin-plate functionals correspond to the Laplacian $L(X) = X_{uu} + X_{vv}$ and squared Laplacian $L^2(X) = LoL(X) = X_{uuuu} + 2X_{uuvv} + X_{vvvv}$ (where u,v represent the surface parameterization) and give rise to the internal tensile and flexural forces respectively.

In a discrete deformable surface model, the Laplacian at each node is approximated using the umbrella operator resulting in the internal tensile force:

$$\alpha_i(t)=\frac{1}{m}\sum_{j\in N(i)} X_j(t)-X_i(t) \quad \text{(4.5)}$$

where:
X_i - the x, y, z coordinate of node i
X_j - the x, y, z coordinate of the neighbours of node i
$N(i)$ - the set of neighbours of node i
m - the number of neighbours of node i

This force is typically normalized by dividing by the maximum distance among all neighbours of node *i*. To compute the internal flexural force at a model node, the squared Laplacian is approximated by convolving the umbrella operator over the node and its neighbours. Currently, SketchSurfaces does not make use of flexural forces (although they are included in the implementation). The strength of the internal forces is controlled by a non-negative weight w_{int}.

4.2.2 External Image Forces

The external forces attract the model towards image features such as image edges, causing the model to deform and take the shape of the target object boundary. In SketchSurfaces, the external image forces are spring forces that *pull* the model nodes towards object boundaries. This pulling force (versus the *pushing* of an inflation force) helps to ensure that the model does not leak in the presence of gaps in the object boundary. For each node, the external force f_i^{ext} is computed along the direction of the surface normal at that node. A user-specified range of voxels along this direction are analyzed and their intensities are compared to a user-specified threshold. When a match is found, the external force is computed as:

$$f_i^{ext} = X_{vox} - X_i \quad \text{.......} (4.6)$$

where:

X_i - the x, y, z coordinate of node i

X_{vox} - the x, y, z coordinate of the matching voxel

Similar to the internal forces, the strength of the external forces is controlled by a non-negative weight w_{ext}. Additionally, when searching for an edge voxel in the user-specified range, there may be several edge voxels having intensities that match the user-specific threshold. In SketchSurfaces, the user can configure the system to pick the voxel with the maximum intensity or to pick the first matching voxel.

4.2.3 Constraint Forces

Constraint forces are designed to allow user interaction with the model or to incorporate prior shape knowledge about the target object. The original approach by Kass [13] defined two constraints: a *spring* and a *volcano*. The spring constraint allowed the user to connect a spring to any point on the snake, usually by mouse click, in an interactive context. The other end of the spring can be a fixed position, a Snake point, or any point on the image. The opposite of the spring constraint, a volcano pushes the Snake out of one local minimum into another. In SketchSurfaces, constraint forces are employed to attract the model towards user-specified locations. The user can modify the surface by manually positioning a surface cross-sectional contour. Constraint forces are then applied to the affected model nodes, pulling them towards the user-specified locations. Further detail on the editing process and the construction of constraint forces are provided in section 6.3.

Chapter 5

Deformable Subdivision Surface Models

The success of interactive shape-model based segmentation techniques, such as Snakes and Balloons, is still heavily dependent upon the generality, controllability, and simplicity of the underlying shape representation scheme. This thesis proposes the use of subdivision curves and surfaces [46] as a very general shape representation basis for interactive deformable models which have a simple formulation. The following sections provide a brief overview of subdivision surfaces, and then proceed to describe the construction of a SketchSurface using these surfaces.

5.1 Subdivision Surfaces

The underlying idea behind subdivision methods [46] is very simple, using geometric algorithms to progressively subdivide a control mesh. Repeated subdivision leads to a hierarchy of increasingly refined models which approach the limit surface (Fig. 5.1). There are two main categories of subdivision surface algorithms: approximating, and interpolating (Fig. 5.2). In the approximating algorithms, the refined mesh is obtained by progressively subdividing the control mesh to get a smoother surface which *does not* pass through the vertices of the control mesh. Since the ability to exactly control the resulting surface is very important in many applications, modifications of approximating schemes have been proposed to force the limit surface to interpolate particular points. For example, one such algorithm in 3D is the interpolating

Modified Butterfly subdivision algorithm [46], which has been used in this thesis. Furthermore, the approximating and interpolating subdivision algorithms provide different levels of continuity. *Continuity* defines how well the pieces of a surface fit together. In general, a surface is C^n continuous if all of its derivatives up to n match across pieces. The Modified Butterfly subdivision scheme constructs a surface that is C^1 continuous.

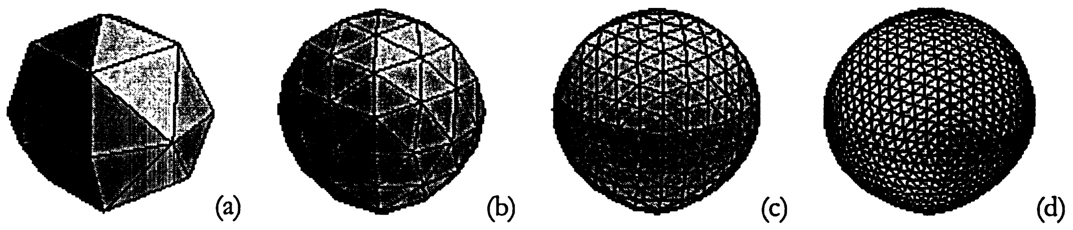

Figure 5.1: Subdivision surface algorithms start with a control mesh (a) and progressively refine the mesh to obtain a smoother mesh at level 1 (b), level 2 (c), level 3 (d), and so on.

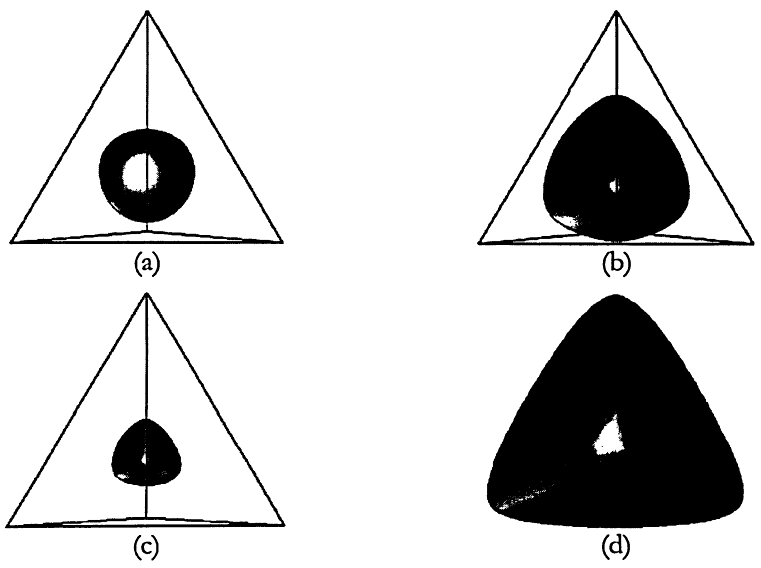

Figure 5.2: Examples of approximating and interpolating subdivision surface algorithms. A control mesh (displayed in wireframe) is subdivided by various algorithms: (a) approximating Catmull-Clark, (b) approximating Doo-Sabin, (c) approximating Loop, (d) interpolating Modified Butterfly.

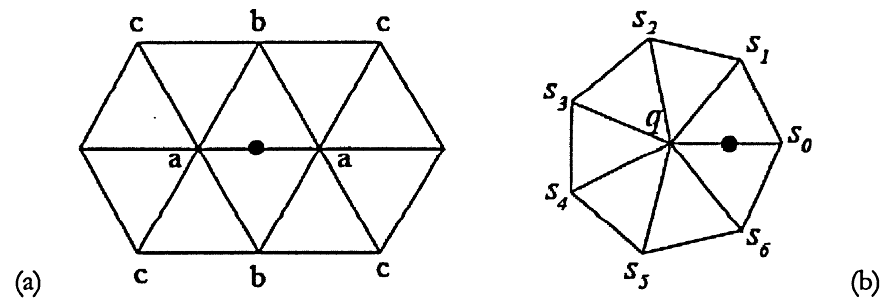

Figure 5.3: The subdivision masks of the Modified Butterfly algorithm. (a) Eight-point stencil. The dot indicates the midpoint of the edge for which a new value is computed. (b) Stencil for a vertex in the 1-neighborhood of an extraordinary vertex. (From [46]).

The Modified Butterfly subdivision scheme computes a new scalar value for each edge midpoint of the control mesh triangulation. The vertices of this triangulation can have different valences. The valence of a vertex is defined as the number of edges emanating from it. The difference in the valence of the end points of an edge results in four distinguishable scenarios when subdividing an edge:

1. The edge connects two vertices of valence 6, in which case the eight point stencil of Fig. 5.3a is used, and the weights are given by:

   $a = 1/2; \quad b = 1/8; \quad c = -1/16$ ....(5.1)

2. The edge connects a K-vertex (K $\neq$ 6) and a 6-vertex; the 1-neighbors of the K-vertex are used in the stencil as indicated in Fig. 5.3b.

   For K $\geq$ 5 the weights are given by:

   $$S_j = \frac{\frac{1}{4} + \cos(\frac{2\pi j}{K}) + \frac{1}{2}\cos(\frac{4\pi j}{K})}{K} \quad \text{with } j = 0,\ldots,K-1 \text{ ....(5.2)}$$

   For K = 3 the weights are:

   $S_0 = 5/12; \quad S_{1,2} = -1/12$ ....(5.3)

   For K = 4 the weights are:

   $S_0 = 3/8; \quad S_2 = -1/8; \quad S_{1,3} = 0$ ....(5.4)

3. The edge connects two extraordinary vertices ($K \neq 6$); in this case the average of the values computed using the appropriate scheme of the previous paragraph is used for each endpoint.
4. Although this thesis only considers closed surfaces, the Modified Butterfly subdivision algorithm can be applied to both open and closed surfaces. A closed surface is a surface that is topologically equivalent to a sphere. An open surface, on the other hand, is topologically equivalent to a disk, and contains boundary edges. A boundary edge is an edge that only belongs to one triangle (i.e. not shared by two triangles). Boundary edges are subdivided using the 1-dimensional 4 point scheme where the weights are:

$$S_{-1} = -1/16; \quad S_0 = 9/16; \quad S_1 = 9/16; \quad S_2 = -1/16 \quad \text{.......................(5.5)}$$

5.2 Global Subdivision vs. Local Subdivision

The Modified Butterfly subdivision method is considered a global subdivision scheme since it operates on the entire control mesh, globally refining it during each level of subdivision. The global subdivision scheme will rapidly increase the number of triangles in a model. In fact, each level of subdivision will quadruple the number of resulting triangles (Fig. 5.4).

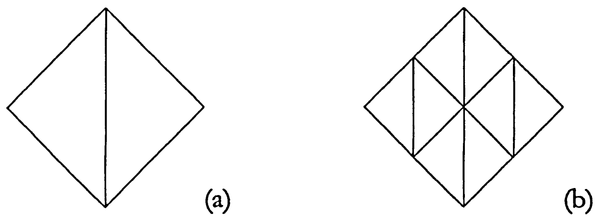

Figure 5.4: Modified Butterfly subdivision. Every refinement step on the coarser level (a) quadruples the number of triangles that will be generated in the finer level (b).

In many applications, after globally subdividing a mesh for a number of levels, the resulting subdivision surface might still have areas which require further refinements. Further global subdivision can refine these areas; however it drastically increases the number of triangles and vertices of the entire model, making it computationally expensive for an application to handle such large models. Local subdivision on the other hand, will not increase the number of triangles at such a high rate. Local subdivision operates on a subset of the triangles and can be used to only subdivide the triangles in the area which needs further refinement.

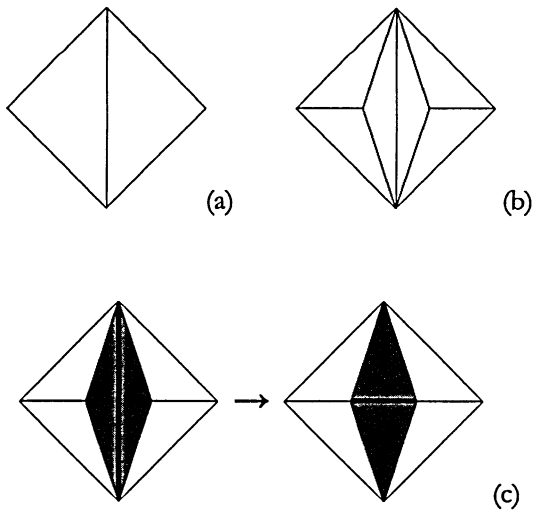

Figure 5.5: Example of local subdivision. The triangles to be subdivided (a) are first broken into three pieces each (b). During the edge-flipping phase (c), the shared edges of triangles in (a) are flipped in order to transform small and narrow triangles into larger triangles. In (c) the shared edge before and after flipping is marked with a thick line and the triangles that are transformed are shaded.

There are different local subdivision techniques, one of which is known as the $\sqrt{3}$ refinement technique [47]. In this method, every triangle to be subdivided is first broken into three new pieces by computing the center of the triangle and connecting it to the three triangle vertices (Fig. 5.5b). Since this phase can result in narrow and small triangles, it is followed by another phase referred to as the "edge-flipping" phase. During the edge-flipping phase, the edges that are shared by any two *original* triangles are flipped (Fig. 5.5c), transforming the small and narrow triangles into larger ones. SketchSurfaces provide local subdivision feature using the $\sqrt{3}$ refinement method in order to further refine a model while maintaining a small number of control points.

5.3 Constructing a Deformable Subdivision Surface Model

To construct a deformable model using a subdivision surface, the vertices of the coarsest level control mesh of the surface (the control points) are used as the degrees of freedom (d.o.f.), and the finest level vertices as the "sensors". The idea is to use a sufficient number of control points and a sufficient level of subdivision such that there is roughly one sensor point for each boundary voxel of the target object, making maximal use of all available image information. The number of required control points and level of subdivision can be determined with one or two trial segmentations. External image forces are computed at these sensor points and then distributed among the control points as illustrated in Fig. 5.6. The details of the process by which the external forces are computed are explained in the next chapter.

The distribution of the sensor forces is done by employing the rules of the Modified Butterfly subdivision such that a control point that is closer to a particular sensor point will receive proportionately more of the computed force than a control point farther away. As explained in the previous sections, when a model is being subdivided, new vertices are added to the model in order to obtain a more refined surface. The position of each newly added vertex is a weighted sum of its neighbours according to the subdivision masks that are illustrated in Fig. 5.3. When distributing the forces, the same mask is used for each vertex, but the rule is applied in the reverse order. In other words, each neighbour will receive a weighted portion of the force based on the weights of the subdivision mask. This distribution step is then followed by a normalization step, in which all of the distributed forces are normalized based on the amount of contributions they have received from the sensor points.

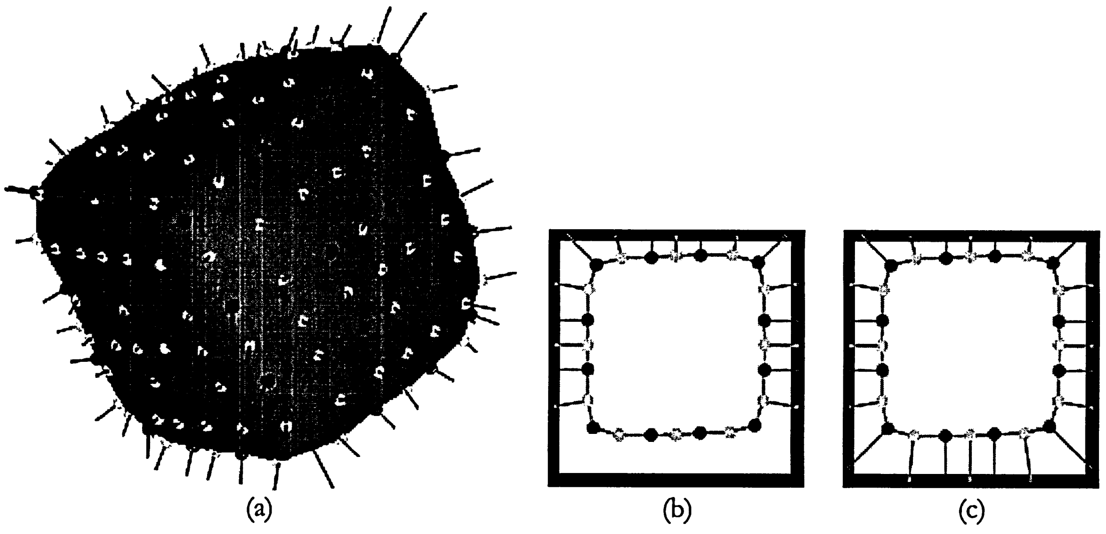

Figure 5.6: External forces are computed at the sensor points and distributed to the control points. The sensor points are presented as light circles and the sensor point forces are the lines emanating from them. The control points (which are also considered sensor points) are presented as dark circles and the distributed forces are the lines emanating from them. A cross-section of the surface is shown in (b). The external forces are computed at the sensor points along the subdivision surface pulling it to the edge of the object of interest (dark square). If a sensor point is within a user-defined range of an object edge point, a force proportional to the distance between the two points is computed. This force is then divided proportionally using the weights of the subdivision mask and added to its "parent" control points. The resulting normalized external force at the control points pulls the control points towards the object boundary. In (c) the user-defined range has been increased, resulting in more activated sensor points, and consequently more of the control points have received distributed forces.

Fig 5.7 shows an example of the force distribution process. In this example a control mesh is subdivided once to obtain a smoother level. After the external forces are computed for the entire model points (control points and sensor points), a control point X_i will receive a weighted portion of the external forces of some of the sensor points in its vicinity N_j (in this example $j \in [0 - 15]$). From the subdivision mask of Fig. 5.3a and equation 5.1, the amount of the distributed external forces to the control point X_i can be computed as:

$$f_i^{dist} = \frac{1}{2}\sum_{j=0}^{5} f_j^{ext} + \frac{1}{8}\sum_{j=6}^{11} f_j^{ext} + \left|-\frac{1}{16}\right|\sum_{j=12}^{15} f_j^{ext} \quad \text{(5.6)}$$

where:

f_i^{dist} - the distributed external force to the control point X_i

f_j^{ext} - the external force of a neighbouring node N_j

Furthermore, the amount of contributions made to a control point can be computed as the sum of the weights used to distribute the external forces of the sensor points:

$$\text{contribution}_i = \frac{6}{2} + \frac{6}{8} + \left|-\frac{4}{16}\right| = 4 \quad \text{(5.7)}$$

Finally the distributed and normalized external force of a control point is then given by:

$$f_i'^{\text{ext}} = \frac{f_i^{\text{ext}} + f_i^{\text{dist}}}{1 + \text{contribution}_i} \quad \text{(5.8)}$$

where $f_i'^{\text{ext}}$ is the new external force of the control point X_i.

After the forces of all sensor points are distributed to the control points, the deformable subdivision surface model is formulated numerically using the simple explicit scheme described in chapter 4 and the positions of the control points are updated. The subdivision surface is then rebuilt using these new positions and new external forces are calculated. This process is repeated until an accurate solution is reached or a user-specified number of iterations have been elapsed.

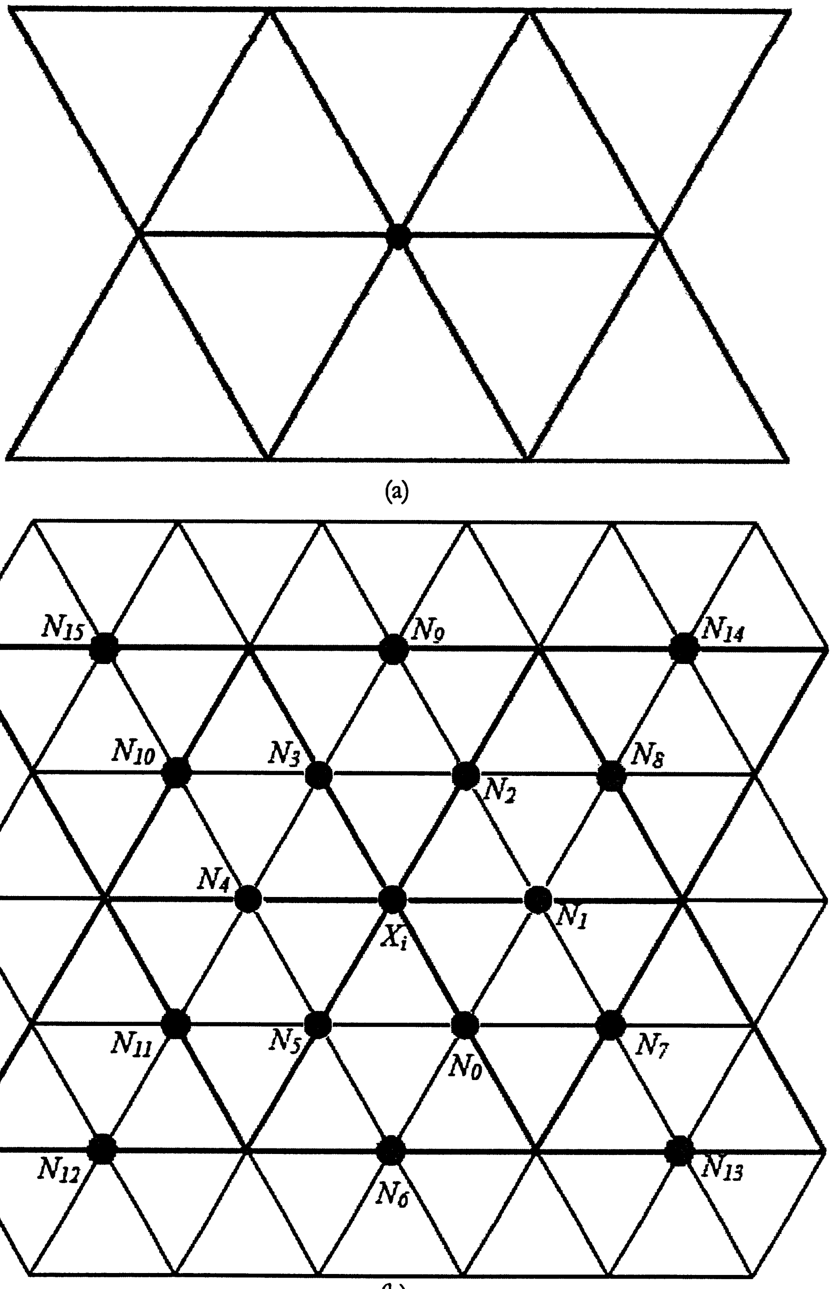

Figure 5.7: Example of force distribution process from sensor points (N_j) to a control point (X_i). (a) Portion of a control mesh and a control point (b) The control mesh and the subdivided surface are overlaid showing the sensor points whose external forces will be distributed to the control point.

Chapter 6

SketchSurfaces

SketchSurfaces are a discrete deformable surface model. However, they are constructed using the subdivision surface algorithm discussed in chapter 5 and the inherent properties of subdivision surfaces provide SketchSurfaces with the robustness against noise of a finite-element based model, the broadest possible shape coverage, a foundation for simple and precise editing, inherent smoothness, a natural hierarchical organization, and a simple representation that allows for the incorporation of high-level shape constraints.

The novel sketch-line initialization process of [11] has been carefully integrated into this model, which allows the subdivision surface to be quickly initialized with a few rough sketch lines drawn across the width of the target object in several key image slices. The result is an initial model that is extremely close in shape to the target object, making the deformable surface model's task of snapping to the object boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. It also minimizes the effect of the model's elasticity parameters so that user's need not search for appropriate values for these parameters – a well-known problem of Snakes and Balloons. Furthermore, it allows the model to primarily rely on image edge information and (optionally) expected edge transitions (e.g. bright to dark) so that it is generally applicable to many common imaging modalities without the need for modality-specific parameters. Finally, unlike 3D LiveWire, no tedious tracing actions are needed - only simple short sketch lines on a few slices are required. Steering and editing the model can take place at any time during the segmentation process and anywhere on the model. In other words, the model has been designed with the 3D user interaction model as the main focus. In the following sections, the construction of this subdivision surface model and the interaction methods are described.

6.1 Sketch Lines: A Simple, Fast, and Accurate Initialization Technique

As discussed in Section 3.2, the most common method of initializing a deformable surface model is to create a small spherical model from a user-defined seed point. Inflation forces are then used to drive the model towards the object boundary. The problem with this approach is the model needs to inflate a considerable amount to reach the object boundary so it is not immediately visually apparent whether the segmentation will succeed. This approach creates model steering issues as well as a separation between the inflation phase of the segmentation and the editing phase.

User fatigue is an important consideration in any interactive design and analysis task. For this reason, sketching actions are being actively researched for many of these tasks in an effort to reduce fatigue and user concentration and to "amplify" user input action and thereby maximize productivity. In the context of interactive segmentation, the idea behind sketching is to allow the user to provide an accurate initialization for the surface model - which minimizes subsequent steering and editing - with a low degree of concentration and simple mouse movements. The sketch-line initialization process proposed in [11] is a simple but effective technique that realizes this idea (Fig. 6.1). It does not require tedious tracing actions and can be performed with minimal concentration. Short sketch lines are quick, easy and comfortable to draw with a mouse or stylus.

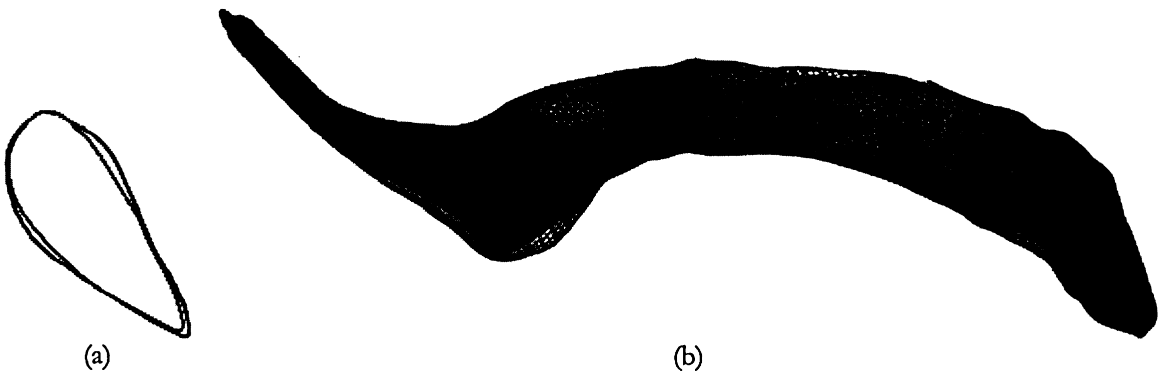

Figure 6.1: Result of the sketch line initialization process for the right brain ventricle. By quickly sketching a few lines in several slices of an MR volume image, a highly-accurate initial surface model is created. (a) cross-sectional contour of the initial model (light curve) shown with a cross section of the manually segmented ventricle (dark curve). (b) the initial surface model (solid semi-transparent) shown with the manually segmented ventricle surface (wireframe).

The user begins the initialization process by first positioning and orienting the image slice plane at the far end of the target object and then sketching a few short lines across the width of the object (Fig. 1.1b-e) on this plane (here on referred to as a *sketch plane*). Drawing these lines roughly perpendicular to the object boundary will result in an initial surface model that is approximately locally aligned with the boundary. If the object boundary exhibits strong edges near the sketch line, the user need not sketch the line carefully - the algorithm will automatically search for the strongest (or user-thresholded) edge close to and perpendicular to the line. However, through repeated experiments it has been observed that the cursor can be positioned precisely and quickly with minimal user effort.

The user then pulls the image slice plane towards himself/herself (re-orienting it as necessary) and again sketches a few lines across the object. A control polygon is created from these lines and is connected to the current control mesh using the Quick-Hull algorithm (http://www.qhull.org). The new control mesh is immediately subdivided to obtain a refined surface (Fig. 1.1f – j). A new cross-section of the refined surface is instantly displayed on the image slice. The entire process occurs in real-time and the user is given immediate visual feedback of the surface construction and can observe the initialization accuracy. The user repeats this sketching process several times until a sketch plane is close to the near-end of the object.

The user can view the 3D surface model as it is constructed and make corrections if desired by repositioning control points and/or by adding new sketch lines. Additional sketch planes can also be inserted. The number of sketch planes required in order to generate an accurate initial surface model is dependent upon the shape of the target object. For example, in our experiments with the caudate nucleus, typically five sketch planes are required, with one close to each end of the caudate nucleus and three evenly distributed in between. Furthermore, the number of sketch lines required on each sketch plane is dependent on the complexity of the shape of target object cross-section. For example, usually only two or three sketch lines are required for the caudate nucleus. It has been experimentally observed that there are obvious dents and/or bumps in the object cross-section or other landmark points that are good candidate positions for sketch line placement. In segmenting the caudate nucleus (Fig. 1.1) the lines have been placed in these types of critical locations and the initialized contour is very close to the caudate boundary.

The user typically performs a few trials, with a different number and position of sketch planes, until an accurate initial surface model is obtained. The experiments conducted in this thesis suggest that it is best to begin with two/three sketch planes (one close to the each end of the target object and optionally, one in the center), examine the initial surface, fit the model and examine the segmentation result, and then add intermediate sketch planes until the desired accuracy is obtained. This pattern of sketch planes and sketch lines can then be recorded and redisplayed when segmenting the target object in all subsequent volumes. Note that the pattern of sketch planes is only a rough guide – the algorithm is fairly robust to small differences in the number/positions of the sketch planes. For large differences accuracy is impacted, resulting in increased user editing.

Finally, the number of the control points of the constructed SketchSurface can be increased by automatically inserting *internal* control points on every edge of the model control mesh. This is done in order to increase the accuracy of the fitted SketchSurface while keeping user input minimal. The number of control points automatically added can be controlled by the user. The default number of internal control points is set to one and this value is typically adequate for many different objects. Typically the user experiments with this setting and increases the number of internal control points if the default setting proves inadequate. Internal control points are added to the control mesh in a manner such that the shape of the subdivision surface is not disturbed. This is to avoid any "surprises" to the user between the initialized surface and the surface used during a fitting step (Fig. 6.2).

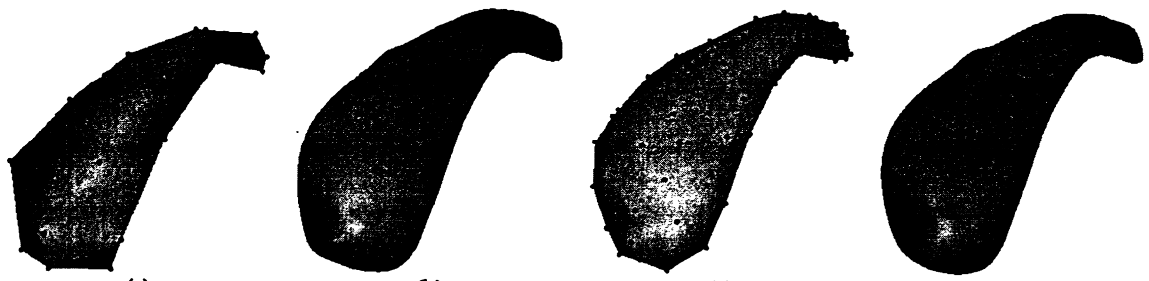

Figure 6.2: Internal control points can be added to a SketchSurface in order to increase model accuracy while keeping user input minimal. (a) Initial control mesh from sketch line initialization process and (b) initial subdivision surface. (c) Internal control points automatically added to the control mesh and (d) new subdivision surface obtained via the subdivision process. The control mesh of a SketchSurface is only used internally by the algorithm and is never presented to the user. The internal control points are added such that there are little or no changes to the shape of the new smooth surface (comparing b to d).

6.2 Fitting

After a SketchSurface has been initialized, the user may choose to fit the surface to the data set and segment out the object of interest. The fitting of the surface is done using the simple explicit scheme described in detail in chapter 4. More specifically, the control mesh of the smooth subdivision surface (which is invisible to the user) is used to deform the subdivision surface.

The sensor points (vertices of the subdivision surface) are used to compute the external forces. The external forces are computed by simply traveling along the normal of a sensor point within a user specified range. The user can specify or use the default threshold range for the acceptable edge voxel intensities. When traveling along a sensor point normal, the voxels that satisfy this range are considered as candidate object boundary points. The algorithm will then pick the voxel with the strongest intensity as the possible location for the object boundary. Subsequently, the external force acting on the sensor point would be a force proportional to the distance between the sensor point position and the edge voxel position. If no acceptable edge voxels have been observed, there will be no external force exerted to that sensor point. Once the external forces of all sensor points have been computed, they will be distributed to the control points as described in section 5.3.

When compared to the inflation forces of the Balloon models, this method of computing the external forces has the advantage of minimizing the leaking of the model to the surrounding regions when there are gaps in the object boundary. In Balloon models, a vertex is *pushed* outwards by an external force, which usually has a constant strength. Strong object boundaries will produce forces that oppose the inflation force, and the strength of these opposing forces is proportional to the strength of the object boundaries. The strength of the inflation force is typically chosen such that it is slightly smaller than the opposing boundary forces, and as a result, when the model reaches these boundaries, it will be stopped by the opposing forces. However, if there are regions in the object boundary where the boundary is weak, or if there are gaps in the boundary (for example due to image noise), then there will be a small or no opposing force to stop the inflation process, leading to the leakage of the model. On the other hand, using the method mentioned above, a vertex is *pulled* towards an object boundary. If no boundary is observed by a sensor point, then there will be no contributions of external forces from that point to its parent control points. In such a model, as the surface evolves towards the object boundaries, the strength of the external forces of sensor/control points diminishes, and in so doing, any chances of leaking is minimized (Fig. 6.3).

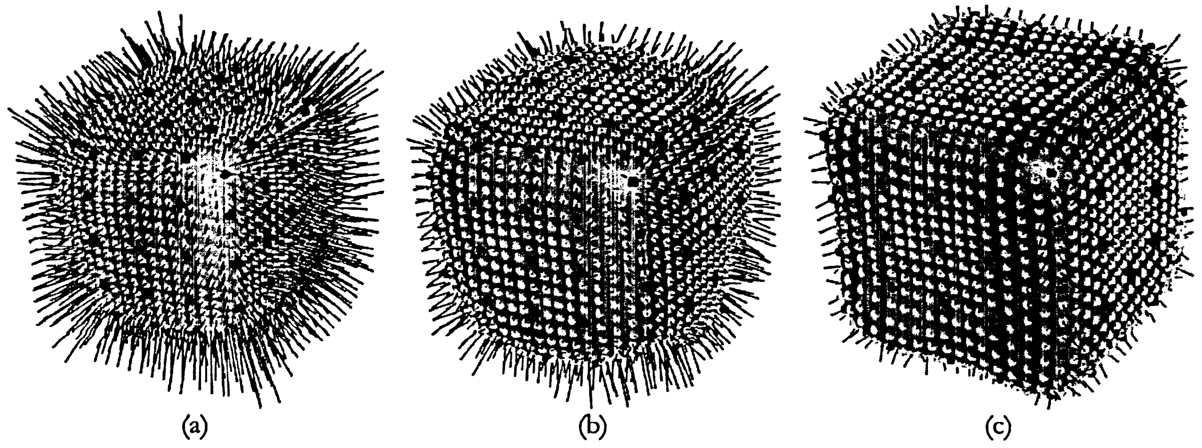

Figure 6.3: Fitting of a SketchSurface to a synthetic cube data set. As the surface evolves towards the edges of the cube, the strength of external forces at sensor points diminishes. Consequently, smaller external forces are distributed to the control points. In this figure, the control points are marked as dark circles and the sensor points are marked as light circles. The external forces are the lines that emanate from these circles.

Once the external forces are computed and distributed to the control points, the internal forces are computed for these control points. The internal forces ensure the smoothness of the control mesh. During the deformation process, it is desirable for a control point not to stray far away from its neighbours. It should be noted that since the surface will evolve by deforming its control mesh, the internal forces are only computed for the control points. However, the external forces are computed at the sensor points in order to make maximal usage of the available image information.

Once the internal forces are computed, the control point positions are updated using the Verlet step. After this deformation step, the subdivision surface is quickly rebuilt using the new position of the control points. The external and internal forces are then recomputed and this process continues until an accurate solution is reached or a user-specified number of iterations have been elapsed.

6.3 Editing and Steering

Editing a segmentation result typically implies making corrections to the segmentation after the algorithm has run to completion. Steering, on the other hand, implies guiding the segmentation process toward the correct result while the algorithm is running. The line between fitting, steering, and editing is blurred when considering the segmentation work-flow using SketchSurfaces. Once the model has been initialized, the user presses a key to begin the fitting step and the surface quickly snaps to the object boundary. The fitting step is terminated after a user-adjustable number of iterations have occurred. This snapping to the boundary is fairly quick for most objects - typically a few seconds. Other stopping criteria may be employed, such as when the average distance traveled by all control points from one iteration to the next falls below a threshold. However, because the initialization process creates a surface that is very close to its final position, only one fitting step is often sufficient to generate an accurate segmentation. Nevertheless, the user may repeat the fitting step by hitting the key. The fitting/snapping of the model to the object boundary is done in discrete steps rather than in a continuously deforming manner, allowing steering/editing to be performed at any time.

Before, between, or after these fitting steps, SketchSurfaces can be precisely and intuitively controlled using various geometric editing actions. The initialization, fitting, editing, steering, and zooming are all performed within a single seamless process, using only simple mouse actions. The user is not forced to constantly switch modes or select actions items from a menu or panel. Examples of the main SketchSurface editing actions are listed below:

1. *Before fitting*: The user can edit the initialized surface prior to any fitting step. Actions include adding new sketch planes, repositioning the existing control points and sketch lines, and adding new control points by drawing new sketch lines or by breaking an existing sketch line into two pieces. To break a sketch line the user can simply right-click anywhere on the line and a new control point will be inserted in that location. The new control point may be selected and dragged to a new location in order to fine-tune its position. The control mesh is automatically updated and the subdivision surface recomputed and displayed in real-time as the control point is dragged.

2. *After/between fitting*: Simple, precise control over the surface model position and shape is performed in 2D on an image slice plane. Constraining the user interactions to two dimensions significantly simplifies the interactions while improving accuracy and user efficiency and maintaining familiarity. Furthermore, no special input device is required. The surface model is cut by the image slice plane in any desired orientation to generate a 2D subdivision curve. The control points of the curve are automatically created from the cross-section of the surface model's control mesh and the resulting curve closely matches the shape of the subdivision surface cross-section. Because the initialization process and the fitting of the surface model provides an initially accurate segmentation, the cross section points of the control mesh are often in a "good" position and fine-tuning can be performed simply by nudging these points, causing a section of the smooth subdivision curve to be dragged into the desired position. The user may modify the curve either by clicking and dragging a point, or by dragging a curve arc. In the latter scenario, the two end points of the arc are selected and dragged such that the end point

that is closer to the mouse cursor is dragged more than the one farther from the cursor. New "soft" constraint forces are then computed from the nearby sensor points on the surface cross-section to the corresponding points on the newly deformed subdivision curve. Using a decay function [48] these forces are distributed to the neighbouring sensor points, followed by a quick refitting (snapping) of the surface model to these new locations. (Fig. 6.4) This process takes advantage of the powerful editing capabilities of subdivision curves and avoids the imprecision and tedium of editing using manual tracing. Curve control points can be fluidly dragged and precisely positioned, and new control points can be added by right-clicking anywhere on the curve.

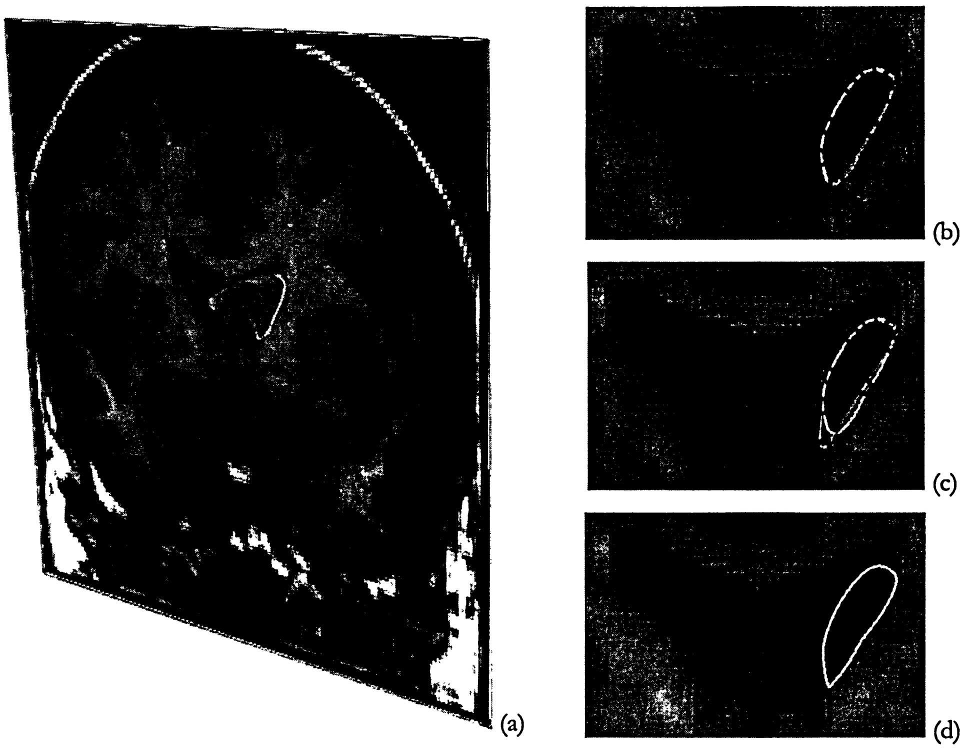

Figure 6.4: Editing in SketchSurfaces. The user manipulates the 3D image slice plane and generates a cross-section of the surface model (a). A subdivision curve is constructed from this cross-section (b). The user precisely reshapes the curve by dragging curve control points (c). Soft constraint forces are generated and the surface model is refitted such that the new cross-section matches the subdivision curve (d).

3. *Snipping*: Occasionally the initialization process creates a surface region that protrudes past the end of the target object. SketchSurfaces also provide the ability to snip away this protrusion by simply positioning the image slice plane such that the overhanging surface region is on one side of the plane. The user will then press a button to cut-out this excess portion (Fig. 6.5)

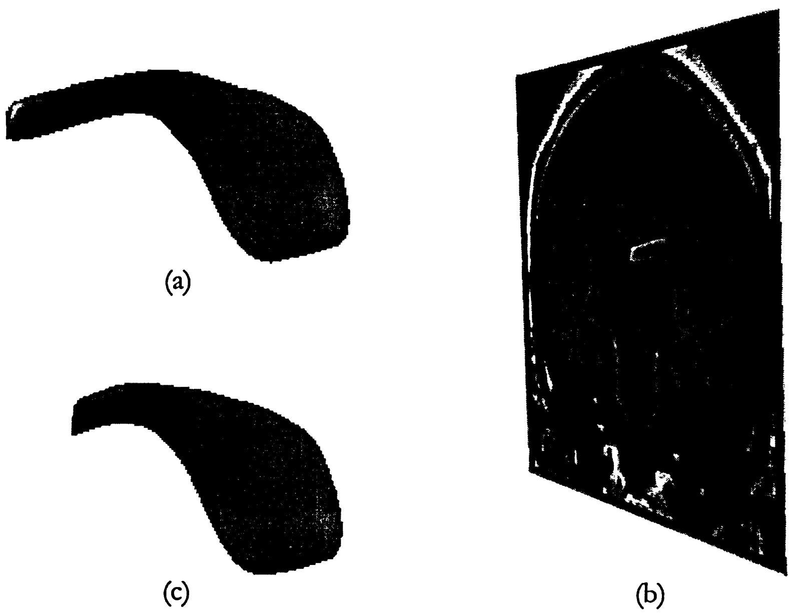

Figure 6.5: Snipping in SketchSurfaces. To remove any excess regions in a SketchSurface (a) the user orients and places the image slice plane where the overhang is located (b) and cuts it away (c).

4. *Pinning/Unpinning*: The user can pin/unpin any control points by simply double-clicking on a point or an edge. When a control point is pinned, regardless of any forces that are acting on it, it will remain stationary. This is a useful feature for segmenting objects with edge gaps or very weak edges. The user can manually place a control point in its "final" position and pin it so that it would never move towards spurious edges or any strong edges of the neighbouring objects (Fig 6.6). The user typically takes more care when

initially positioning these points, ensuring that the control points are placed where the user has interpreted the boundary location. Thus, optimal use of human recognition capabilities is exploited. The user is able to easily and dynamically transfer knowledge of the object boundary to the algorithm as the surface is constructed, maximizing the chance of segmentation success and thereby minimizing or eliminating post user editing/interacting phase. This philosophy is similar to LiveWire [28], where the user-controlled cursor speed is used to indicate weak and strong boundaries.

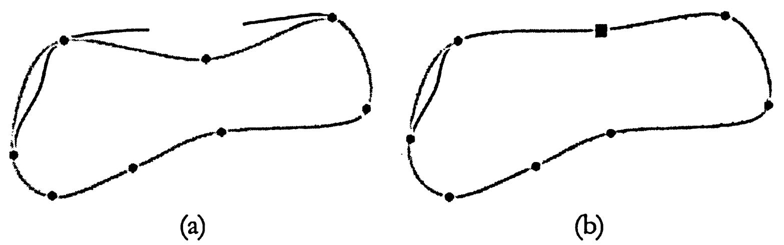

Figure 6.6: Pinning a control point on a cross-section of a SketchSurface. (a) For an object with weak edges or gaps in its boundaries (dark curve) the pinning action can be used to create stationary control points and increase segmentation accuracy. (b) The user drags a control point to its final location and pins it by double-clicking on it (indicated by a square) and the SketchSurface is then constrained to pass exactly through the pinned control point. Points can be pinned or unpinned as the user constructs the SketchSurface or after a fitting step.

5. *Local subdivision*: Often it may be necessary to further refine a subdivision surface in a local region. This is particularly the case when segmenting objects with narrow and long protrusions. As mentioned in section 5.2, further refinement of the surface using a global subdivision approach will drastically increase the number of surface points which can be problematic. As a result, local subdivision of a region is a necessity of any segmentation tool. In SketchSurfaces new control points and sensor points can be added in a *local*

region where editing is occurring. Currently this is explicitly initiated by the user, however as a future extension, this step can be fully automated. As discussed above, during editing the user operates on a cross-section of the subdivision surface and can break the cross-section curve to deform it. When a curve arc is broken, its corresponding control triangle is automatically detected and locally subdivided using the algorithm of section 5.2.

6. *Zooming*: Although a basic feature, the zooming functionality is a useful and necessary feature of any segmentation tool. The zooming should be done effortlessly since it is often used during all of the initialization, editing, and steering phases. In SketchSurfaces, the zooming is performed using the mouse wheel-button in either the 2D view or the 3D view. An upward wheel motion (scroll-up) will zoom into the scene while a downward wheel motion (scroll-down) will zoom out.

6.4 SketchSurfaces Parameters

Over the years, one of the main criticisms of deformable models has been the difficulty in finding suitable parameters for a specific target object and image modality. This criticism can also be levelled at 2D/3D LiveWire. For example, although default parameter settings for Adobe Photoshop's Magnetic Lasso (an implementation of LiveWire) are often adequate, for efficient segmentation parameter adjustments are needed to control the search width, frequency of automatic seed dropping, and edge contrast. Deformable models also have similar parameters plus additional bendiness and stretchiness controls and consequently some potential users have been reluctant to use these approaches. One of the goals of

SketchSurfaces is the creation of a segmentation tool that requires little or no parameter tweaking or mode changing. The initialization process and the subdivision surface model of SketchSurfaces go a long way towards achieving this goal. Due to the nature of SketchSurfaces initialization, the aforementioned additional parameters very rarely need adjusting. Nevertheless, the user is able to set several intuitive parameters:

1. *Edge intensity control*: The maximum and minimum edge intensity threshold can be set with two sliders. The model will search for edge intensities within this range. Most commonly, the strongest edges are searched for. Canny edges are used as they are visually simpler to "see".

2. *Bendiness/stretchiness*: The default setting for these parameters are typically adequate for almost all objects and image modalities. In fact, the same settings have been used for all the experiments in this thesis. This is primarily due to the proximity of the SketchSurface to the object boundary after initialization, making these parameters less important. Nevertheless, sliders are provided should the user need to change these settings.

3. *Image edge search range*: For each sensor point, a search along its normal is carried out for a small, user-specified distance (typically only four or five voxels). If a matching edge voxel is found, a spring force is applied to attract the model point to it. If no matching edge is found (in the case of a boundary gap or noisy edge voxels), there will be no contributions to the image forces from this sensor point. The search range typically starts at a few voxels in the

negative sensor point normal direction and continues up to a few voxels in the positive normal direction, and can be controlled by two sliders.

4. *Edge transition control*: This feature can be used to add more intelligence to the fitting algorithm in order to improve segmentation accuracy and repeatability. Using this functionality, the algorithm can be programmed to search for edges with either a bright-to-dark or a dark-to-bright transition in the intensity image. The user can also disable this feature, in which case the algorithm will search for edge voxels within the specified intensity range.

Although default settings are typically adequate, the SketchSurfaces control panel allows a user to "tune" the above parameters for a specific object and image modality and maximize segmentation performance. This tuning process is typically only performed once. The parameter settings can then be saved for future use.

Chapter 7

Experimental Results

SketchSurfaces have been successfully tested on a number of synthetic and real data sets. The synthetic data sets were created via voxelization of a few 3D meshes including a cube, an ellipsoid, a 3D model of human kidney, and a 3D model of human liver. The voxelization process was used to convert each of these 3D meshes into volumes of size 256 × 256 × 256. Five real brain image data sets and their corresponding hand-segmented results were obtained from the Internet Brain Segmentation Repository (http://www.cma.mgh.harvard.edu/ibsr/). The data sets were interpolated to obtain cubical voxels, with interpolated dimensions of 256 × 256 × 205 and 256 × 256 × 192. The right caudate nucleus, the right brain ventricle, and the right putamen were segmented from these data sets and their hand-segmented surfaces were used to calculate the accuracy of SketchSurface model. The accuracy was measured using 2-sided Hausdorff distancing [49], which is a more strict measure for accuracy than simpler measures such as volume overlap. All of the experiments were conducted on a PC with a CPU clock speed of 3.2 GHz and 1 GB of memory.

7.1 Segmenting Synthetic Data Sets

The first experiment was to use SketchSurfaces to segment a cube. The user initializes a roughly cubical shape using two sketch planes and by simply drawing two sketch lines per plane. The algorithm was set to insert three internal control points on each edge of the initialized control mesh in order to rapidly increase the degrees of freedom without requiring too much user interaction – the cube was initialized and internal control points were added in as little as 8 seconds. The fitting step was then initiated and the model deformed to a more cubical shape (Fig. 7.1.1) in under a second. By looking at the final segmentation result, it can be observed that the surface was not able to capture the cubical shape at its corners. This is simply the side-effect of using the Modified Butterfly subdivision algorithm. This subdivision process creates a smooth surface and removes sharp corners and that is why our subdivision-based surface does not entirely capture the corners. However, since SketchSurfaces are to be used for segmenting anatomical structures (which are inherently smooth), SketchSurfaces have employed this subdivision surface algorithm.

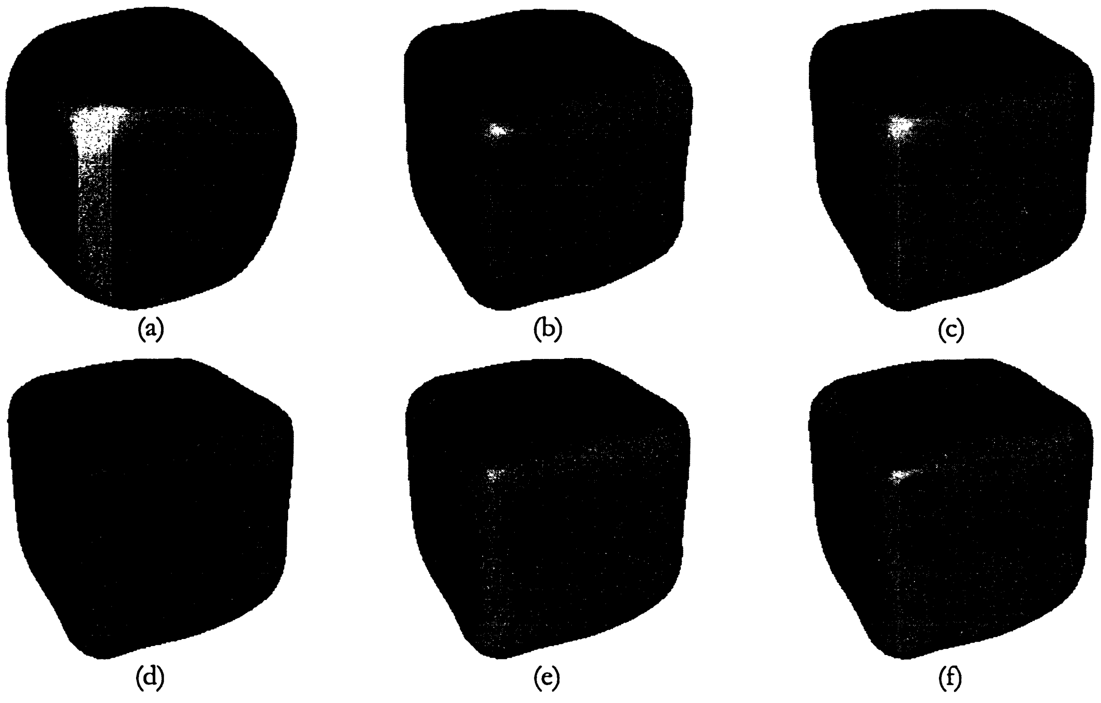

Figure 7.1.1: Example segmentation of a cube from a synthetic data set. (a) The user-initialized SketchSurface, (b–e) Surface deforming towards the object boundary, (f) Final segmentation result.

Next, SketchSurfaces were used to segment an ellipsoid. Similar to a cube, an ellipsoid has a topologically simple surface, but unlike a cube it does not have any sharp corners. A SketchSurface was initialized using five sketch planes and internal control points were added. This process only took 25 seconds. The surface was then fitted to the object boundary (in under a second) and the result is presented in Fig. 7.1.2.

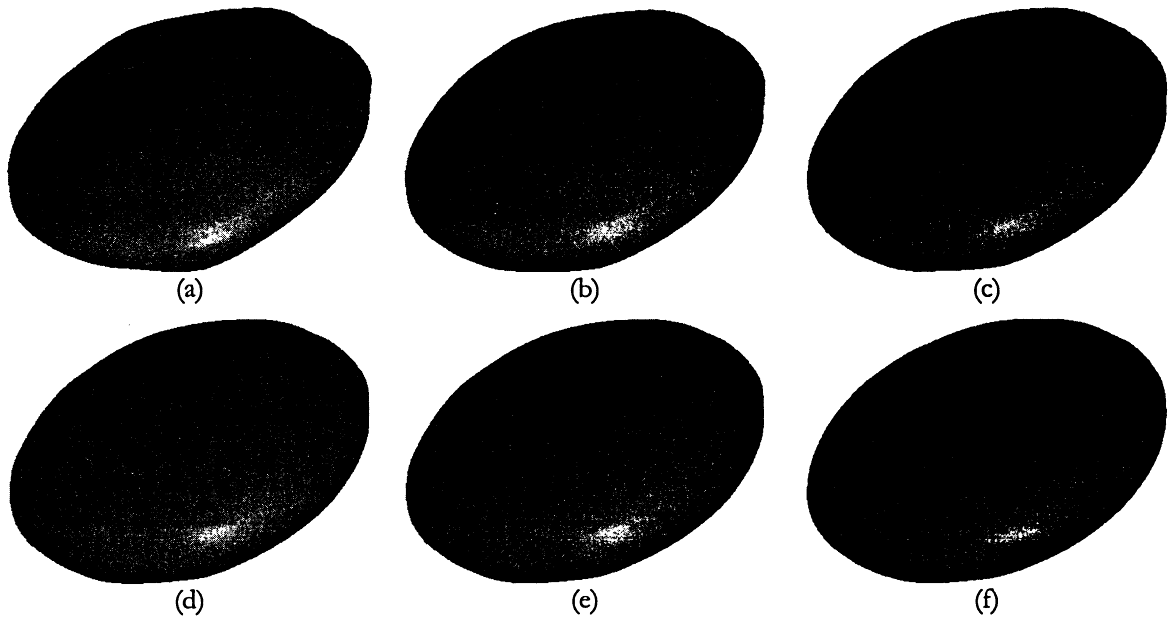

Figure 7.1.2: Example segmentation of an ellipsoid from a synthetic data set. (a) The user-initialized SketchSurface, (b–e) Surface deforming towards the object boundary, (f) Final segmentation result.

The next experiment was segmenting a human kidney from a synthetic data set. A low-polygon 3D model of a kidney was obtained from The 3D Archive (http://www.the3darchive.com). The model was processed by Autodesk 3DS Max (http://www.autodesk.com) to obtain a high-polygon mesh and then was voxelized to obtain a volume data set. A SketchSurface was then initialized using five sketch planes and by adding three internal control points per control edge, all in merely 32 seconds. The surface was then fitted to the object boundary in 9 seconds, for a total segmentation time of 41 seconds. Next, 2-sided Hausdorff distancing was used to measure the accuracy of this segmentation, which showed an average distance of 0.13 voxels and 99.7% sub-voxel accuracy. The result of this segmentation is presented in Fig. 7.1.3.

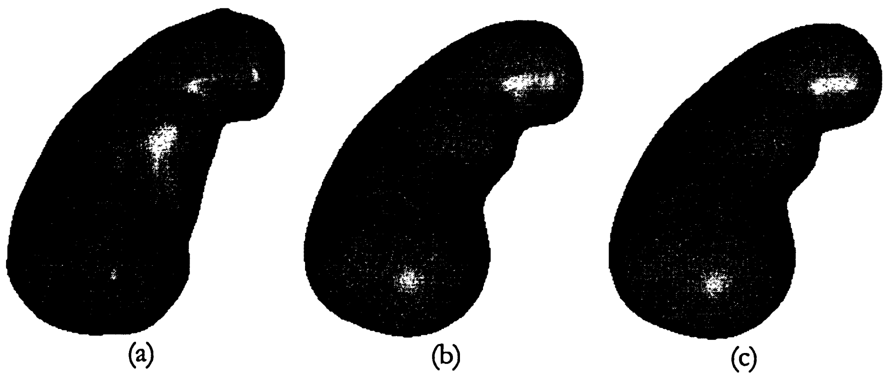

Figure 7.1.3: Example segmentation of human kidney from a synthetic data set. (a) The user-initialized SketchSurface. (b) Final segmentation result. (c) The hand-segmented surface.

Finally, SketchSurfaces were used to successfully segment a human liver from a synthetic data set, which is more geometrically complex than a kidney. Once again a low-polygon 3D model of a human liver was obtained from The 3D Archive and used the same pre-processing as above to generate the volume data set. A SketchSurface was then initialized using six sketch planes and by adding two internal control point per control edge, in approximately 37 seconds. The surface was then fitted to the object boundary in 9 seconds, for a total segmentation time of 46 seconds. Next, 2-sided Hausdorff distancing was used to measure the accuracy of this segmentation, which showed an average distance of 0.24 voxels and 98.1% sub-voxel accuracy. The result of this segmentation is presented in Fig. 7.1.4.

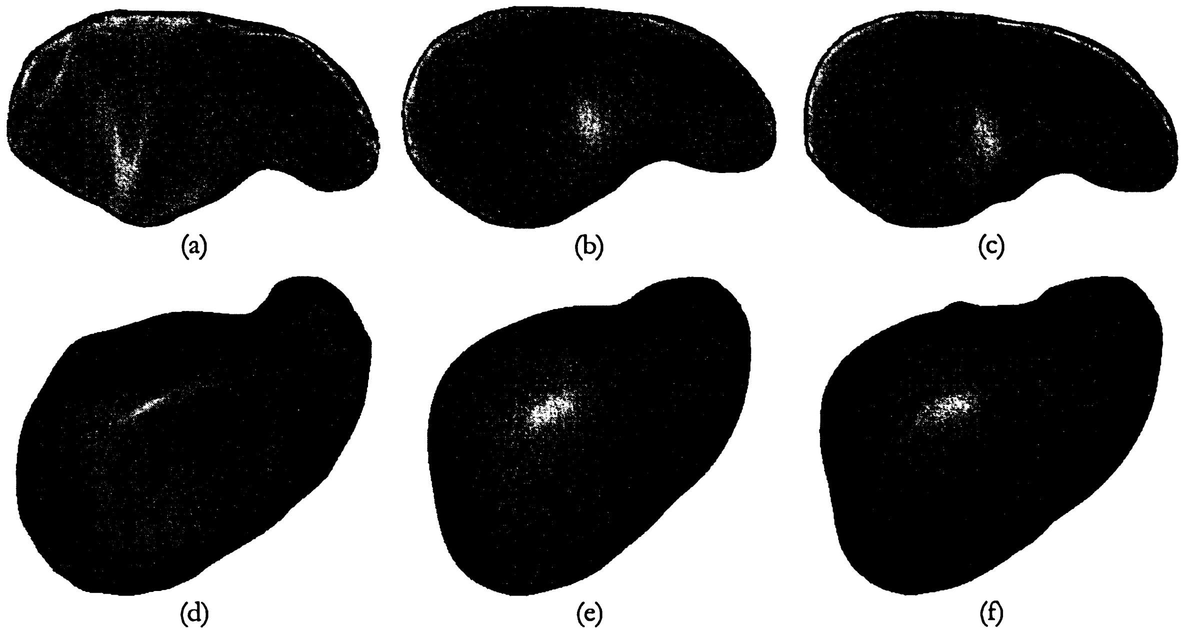

Figure 7.1.4: Example segmentation of a human liver from a synthetic data set. (a, d) Different views of the user-initialized SketchSurface. (b, e) Different views of the segmentation result (fitted surface). (c, f) Different views of the hand-segmented surface.

7.2 Segmenting the Right Caudate Nucleus

Figure 7.2 and Table 7.2 illustrate the results that were obtained from segmenting the right caudate nucleus from the five data sets. The same parameters have been used for all of the segmentations, supporting the repeatability of our method. Since the shapes of the caudate nucleus in the data sets are relatively similar, the same number of sketch planes was used in the construction of the initial surface. Table 7.2 demonstrates a significant increase in the speed of the segmentation process when compared to [7] and [9]. On average, the initialization and fitting were performed in about 36 seconds plus an additional 29 seconds to edit a few slices, resulting in a total segmentation time of about 65 seconds and sub-voxel accuracy. This can be roughly compared to 2.5 minutes in [7] (91.7% sub-voxel accuracy) and 3.5 minutes in [9] (where a caudate *mask* was segmented with 98.5% sub-voxel accuracy).

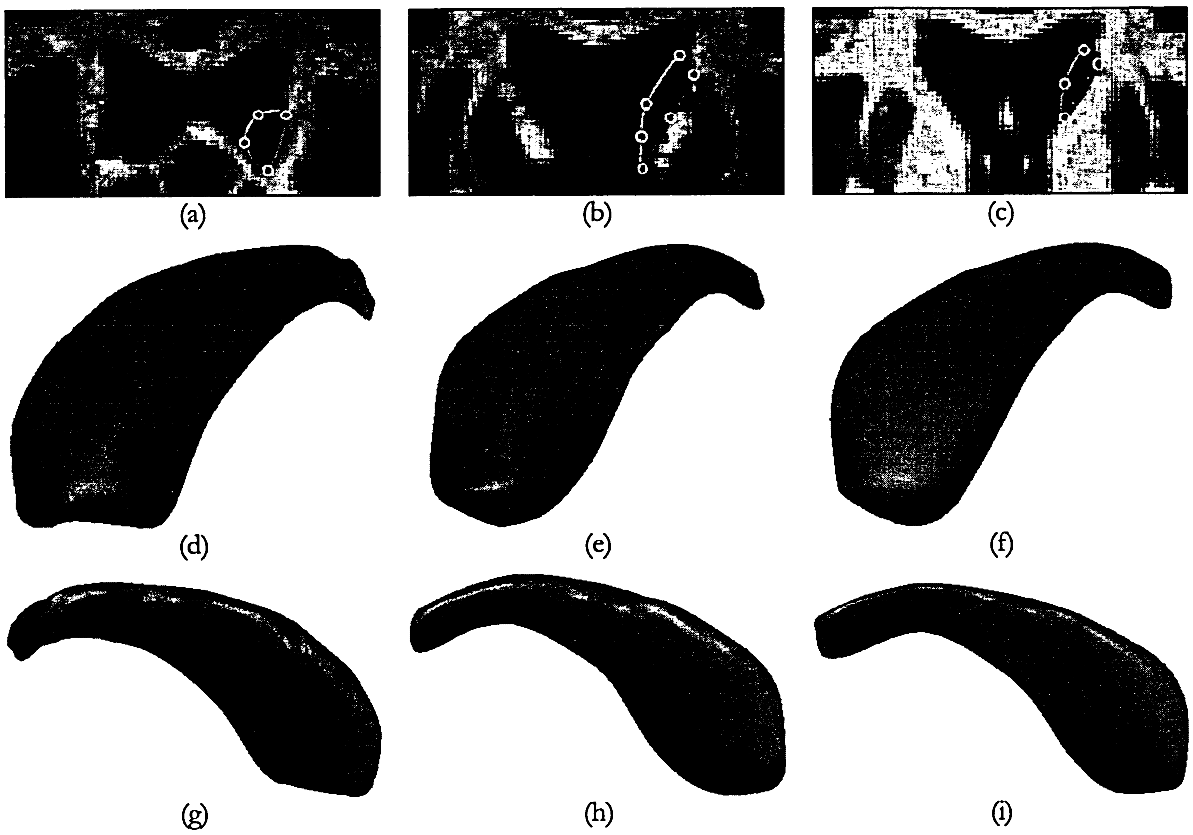

Figure 7.2: Example segmentation result for the right caudate nucleus. (a – c) A few user-initialized contours. (d, g) Different views of the user-initialized SketchSurface. (e, h) Different views of the segmentation result (fitted surface). (f, i) Different views of the hand-segmented surface.

Table 7.2: Results of segmenting the right caudate nucleus from five data sets.

					Hausdorff dist. stat. between the extracted and expert seg. surfaces		
Exp	# of Sketch Planes	# of Control / Sensor Points	Seg. Time (sec)	Editing Time (sec)	Avg. Dist.	Max Dist.	# of (% of) sensor points with sub-voxel accuracy
1	5	79 / 1219	33	20	0.308	1.81	1172 (96.1%)
2	5	87 / 1343	35	42	0.481	2.25	1224 (91.1%)
3	5	92 / 1412	37	24	0.444	2.11	1294 (91.6%)
4	5	87 / 1347	36	36	0.422	1.93	1257 (93.3%)
5	5	84 / 1284	37	24	0.373	1.36	1243 (96.8%)

7.3 Segmenting the Right Brain Ventricle

Figure 7.3 and Table 7.3 illustrate the results obtained from segmenting the right brain ventricle from the five data sets. Once again, the same parameters have been used for all of the segmentations. Moreover, with the exception of experiment 4, approximately the same numbers of sketch planes and control points have been used. Due to a significant difference in the shape of the Ventricle in experiment 4 (when compared to the other experiments) an extra sketch plane was inserted. On average, the initialization and fitting were performed in 41 seconds plus an additional 9 seconds to edit one slice, resulting in a total segmentation time of about 50 seconds with sub-voxel accuracy. This result can be roughly compared to approximately 7.5 minutes in [7] and 5.5 minutes in [9] (no accuracy values were reported by [7] or [9]).

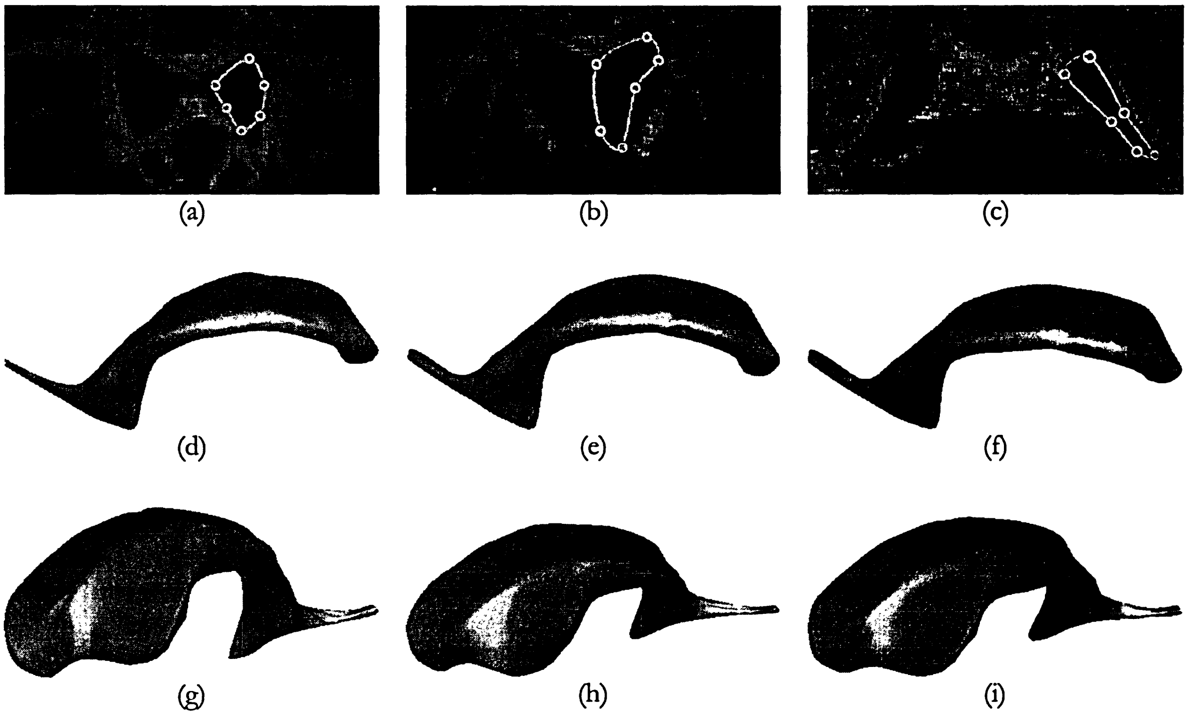

Figure 7.3: Example segmentation result for the right brain ventricle. (a – c) A few user-initialized contours. (d, g) Different views of the user-initialized SketchSurface. (e, h) Different views of the segmentation result (fitted surface). (f, i) Different views of the hand-segmented surface.

Table 7.3: Results of segmenting the right brain ventricle from five data sets.

					Hausdorff dist. stat. between the extracted and expert seg. surfaces		
Exp	# of Sketch Planes	# of Control / Sensor Points	Seg. Time (sec)	Editing Time (sec)	Avg. Dist.	Max Dist.	# of (% of) sensor points with sub-voxel accuracy
1	7	132 / 2052	40	14	0.460	2.07	1905 (92.8%)
2	8	153 / 2373	39	8	0.393	1.83	2290 (96.5%)
3	8	150 / 2310	41	7	0.412	2.43	2160 (93.5%)
4	9	169 / 2629	43	12	0.336	2.15	2500 (95.1%)
5	8	153 / 2373	42	5	0.357	1.62	2297 (96.8%)

7.4 Segmenting the Right Putamen

Finally, the right putamen was also segmented from one volume image. The shape of the putamen is more complex than that of the caudate, requiring 6 sketch planes instead of 5 to construct the initial surface model and one or two more sketch lines per sketch plane were required to create accurate contours. In addition, the putamen is a very noisy object and there are large regions where there are no edge voxels and where the user must interpret the location of the boundary. These factors demand precise segmentation control. With SketchSurfaces, the initialization and fitting were performed in 65 seconds plus an additional 8 seconds to edit one slice, resulting in a total segmentation time of about 73 seconds with sub-voxel accuracy (avg. distance 0.484 voxels, max. dist. 2.07 voxels, 90.3% of sensor points with sub-voxel accuracy).

Furthermore, from our experiments, it was observed that steering via control point pinning results in large efficiency gains. For example, on several sketch planes there are regions of the caudate nucleus or the putamen without any visible object boundaries in the edge-detected image. In these regions, the sketch lines were drawn more carefully and several control points were pinned. Due to the nature of our initialization technique and the flow of our segmentation method, this pinning action requires only a very slight increase in effort.

Chapter 8

Conclusions

8.1 Summary

Optimizing the performance of 3D semi-automatic medical image segmentation methods for noisy volume images requires a minimal number of fast, simple and fatigue-free interactions, and intuitive, flexible, precise user steering and editing capabilities. Furthermore, all of these capabilities must be seamlessly integrated into the segmentation work-flow so that they are available at any time and presented to the user using a consistent interface. In the technique presented by this thesis, the combination of sketching input lines across an object, a powerful subdivision-surface based deformable model, with subdivision curve editing flexibly derived from arbitrarily oriented cross-sections of this model, results in a tool that meets these requirements and is effective for many segmentation tasks that cannot be processed as efficiently with other techniques.

SketchSurfaces are implemented based on the design principles and guide lines of the interactive segmentation methods as described in section 2.6. The inputs to the algorithm are provided in pictorial form, in an effective and effortless manner, using an interface that seamlessly combines the initialization, fitting, steering, and editing phases. The user has full control in the entire segmentation process in order to obtain accurate results and only needs to provide minimal input to the algorithm. Moreover, proper and immediate visual feedback is maintained throughout the segmentation work-flow in order to better assist the user in obtaining accurate results.

8.2 Future Work

Our deformable subdivision surface model, along with its interaction facilities works best on moderately complex shaped objects, in clean or noisy images. For very complex-shaped objects, such as arterial trees or the cerebral cortex, too many sketch planes may be required and therefore a "flow" model may be better suited, at least on the "front-end" of the segmentation process. For this reason, a combination of Graph Cuts and SketchSurfaces is intriguing and is the subject of future research. The idea would be to use Graph Cuts to generate an initial segmentation of a complex-shaped object and to then automatically wrap a SketchSurface around the result, parameterizing the subdivision surface based on surface curvature. The fitting, steering and editing capabilities of SketchSurfaces could then be used to fine-tune the segmentation in difficult regions. This combination could potentially create a tool for efficiently segmenting any object from any image modality.

As discussed in the previous chapters, this thesis extends the sketch-line initialization technique of [11] to 3D. In [11] a pen device is used to perform all of the initialization, editing, and steering actions. Nevertheless, in order to achieve faster development, in completing this thesis it was decided to use a mouse as the input device. A future extension would be to replace mouse actions with pen stylus actions. A pen is a more efficient, natural and precise input device for drawing actions than a mouse, which will give the user more flexibility and manoeuvrability.

Other extensions to our method currently being implemented are allowing the subdivision surface model to be initialized for more topologically complex objects, and allowing multiple objects to be segmented. Additionally, the use of a 3DOF input device is being explored in order to determine if moving through the volume image along slices orthogonal to the medial surface of a highly curved object can be simplified.

From the experiments that were conducted, it was observed that the sketch planes are always placed in specific locations for segmenting objects with various shapes and sizes. There is always a sketch plane near each end of the object. Other planes are located in between and in places where there is significant change in the shape of the cross-section of the object in a slice. Consequently, the automation of sketch plane placements is being considered as a future enhancement. The user would mark the near and far ends of the object and the algorithm automatically determines where the interior sketch planes are positioned. The user then just clicks a button and the image slice plane is automatically advanced to the next sketch plane position.

The ability to apply SketchSurfaces to an object with a significant protrusion is also a future enhancement. When the user clicks the mouse pointer on or near a control mesh edge, a region of the surface model around this edge becomes the active region and is highlighted. Consequently, when new lines are sketched by the user on a new sketch plane, the resulting contour is connected to the surface model using a contour delineating the active region, forming a surface protrusion. This extrusion action would allow a user to quickly construct complex initial surface shapes that may have significant protrusions or while keeping the initialization process flexible and intuitive.

Finally, the automatic local subdivision of a SketchSurface should be investigated. Currently, when a local refinement of an object is needed, local subdivision is used by SketchSurfaces to subdivide that particular region. However this step is explicitly initiated by the user. A future extension would be to automatically detect and subdivide such regions without requiring user intervention.

Bibliography

1. J.P. Pons, J.D. Boissonnat, "Delaunay deformable models: Topology-adaptive meshes based on the restricted Delaunay triangulation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007.

2. J. Montagnat, H. Delingette, "Spatial and temporal shape constrained deformable surfaces for 3D and 4D medical image segmentation," Technical report RR-4078, INRIA, 2000.

3. T. McInerney, D. Terzopoulos, "Topology adaptive deformable surfaces for medical image volume segmentation," in *IEEE Transactions on Medical Imaging*, vol. 18, No. 10, pp. 840–850, 1999.

4. J.Y. Park, T. McInerney, D. Terzopoulos, M.H. Kim, "A Non-Self-Intersecting Deformable Surface for Complex Boundary Extraction from Volumetric Images," in *Computers and Graphics*, Vol. 25, No. 3, pp. 421–440, 2001.

5. T. McInerney, D. Terzopoulos, "A finite element model for 3D shape reconstruction and nonrigid motion tracking," in *Proceedings of the Fourth International Conference on Computer Vision* (ICCV'93), Berlin, Germany, pp. 518–523, May, 1993.

6. J. Bredno, T.M. Lehmann, K. Spitzer, "A General Discrete Contour Model in Two, Three, and Four Dimensions for Topology-Adaptive Multichannel Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, No. 5, pp. 550–563, 2003.

7. P. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S. Ho, J. Gee, G. Gerig, "User-guided 3D active contour segmentation of anatomical structures: significantly improved efficiency and reliability," in *NeuroImage*, Vol. 31, No. 3, pp. 1116–1128, 2006.

8. A.X. Falcão, J.K. Udupa, "A 3D Generalization of User-Steered Live Wire Segmentation," in *Medical Image Analysis*, Vol. 4, No. 4, pp. 389–402, 2000.

9. K. Poon, G. Hamarneh, R. Abugharbieh, "Segmentation of Complex Objects with Non-Spherical Topologies from Volumetric Medical Images using 3D Livewire," in *Proceedings of SPIE Medical Imaging: Image Processing*, Vol. 6512, No. 31, pp 1–10, 2007.

10. G. Hamarneh, J. Yang, C. McIntosh, M. Langille, "3D live-wire-based semi-automatic segmentation of medical images," in *Proceedings of SPIE Medical Imaging: Image Processing*, Vol. 5747, pp. 1597–1603, 2005.

11. T. McInerney, M.R. Akhavan Sharif, "Sketch initialized snakes for rapid, accurate, and repeatable interactive medical image segmentation," in *IEEE International Symposium on Biomedical Imaging* (ISBI'06), Arlington, Virginia, pp. 398–401, April 2006.

12. S.D. Olabarriaga, A.W.M. Smeulders. "Interaction in the segmentation of medical images: A survey," in *Medical Image Analysis*, Vol. 5, pp. 127–142, 2001.

13. M. Kass, A. Witkin, D. Terzopoulos, "Snakes: Active contour models," in *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321–331, 1988.

14. D. Terzopoulos, A. Witkin, M. Kass, "Constraints on deformable models: recovering 3D shape and nonrigid motion", in *Artificial Intelligence*, Vol. 36, No. 1, pp. 91–123, 1988.

15. T. McInerney, D. Terzopoulos D, "Deformable models in medical image analysis: a survey," in *Medical Image Analysis*, Vol. 1, No. 2, pp. 91–108, 1996.

16. I. Carlbom, D. Terzopoulos, K. Harris, "Computer-assisted registration, segmentation, and3D reconstruction from images of neuronal tissue sections," in *IEEE Transactions on Medical Imaging*, Vol. 13, No. 2, pp. 351–362, 1994.

17. L.D. Cohen, I. Cohen "Finite element methods for active contour models and balloons for 2D and3D images," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 11, pp. 1131–1147, 1993.

18. R. Durikovic, K. Kaneda, H. Yamashita, "Dynamic contour: a texture approach and contour operations," in *The Visual Computer*, Vol. 11, pp. 277–289, 1995.

19. J. Miller, D. Breen, W. Lorensen, R. O'Bara, M. Wozny, "Geometrically deformed models: A method for extracting closed geometric models from volume data," in Computer Graphics (Proc. SIGGRAPH'91 Conf.), Las Vegas, NV, Vol. 25. No. 4, pp. 217–226, 1991.

20. I. Cohen, L. Cohen, N. Ayache, "Using deformable surfaces to segment 3D images and infer differential structures," in *Computer Vision, Graphics, and Image Processing*, Vol. 56, No. 2, pp. 242–263, 1992.

21. T. McInerney, D. Terzopoulos, "A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis," in *Computerized Medical Imaging and Graphics*, Vol. 19, No. 1, pp. 69–83, 1995.

22. D. Metaxas, D. Terzopoulos, "Shape and nonrigid motion estimation through physics-based synthesis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 6, pp. 580–591, 1993.

23. S.J. Osher, J.A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," in *Journal of Computational Physics*, Vol. 79, pp. 12–49, 1988.

24. J. Suri, S. Singh, S. Laxminarayan, X. Zeng, K. Liu, and L. Reden, "Shape recovery algorithms using level sets in 2-D/3-D medical imagery: A state-of-the-art review," in *IEEE Transactions on Information Technology in Biomedicine*, Vol. 6, pp. 8–28, 2002.

25. E.N. Mortensen, W.A. Barrett, "Intelligent scissors for image composition," in *Proceedings of Computer Graphics* (SIGGRAPH'95), Los Angeles, CA, pp. 191–198, August 1995.

26. W.A. Barrett, E.N. Mortensen, "Interactive live-wire boundary extraction," in *Medical Image Analysis*, Vol. 1, No. 4, pp. 331–341, 1997.

27. E.N. Mortensen, W.A. Barrett, "Interactive segmentation with intelligent scissors," in *Graphical Models and Image Processing*, Vol. 60, No.5, pp. 349–384, 1998.

28. A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, "User-steered image segmentation paradigms: Live wire and live lane," in *Graphical Models and Image Processing*, Vol. 60, No. 4, pp. 233–260. 1998.

29. A.X. Falcão, J.K. Udupa, F.K. Miyazawa, "An ultra-fast user-steered segmentation paradigm: Live-wire-on-the-fly," in *IEEE Transactions on Medical Imaging*, Vol. 19, No. 1, pp. 55–62, 2000.

30. A.X. Falcão, J.K. Udupa, S. Samarasekera, B.F. Hirsch, "User-steered image boundary segmentation," in *Proceedings of SPIE on Medical Imaging*, Vol. 2710, Newport Beach, CA, pp. 278–288, 1996.

31. Adobe Systems Inc., *Adobe Photoshop CS3 User Guide.* California: Adobe, pp. 245–247, 2007.

32. A. Schenk, G. Prause, H.O. Peitgen. "Efficient Semiautomatic Segmentation of 3D Objects in Medical Images," in *Medical Image Computing and Computer-Assisted Intervention*, pp. 186–195, Springer, 2000.

33. Y.Y. Boykov, G. Funka-Lea, "Graph Cuts and Efficient N-D Image Segmentation," in *International Journal of Computer Vision*, Vol. 70, No. 2, pp. 109–131, 2006.

34. A.X. Falcão, F.P.G. Bergo, "Interactive Volume Segmentation with Differential Image Foresting Transforms," in *IEEE Transactions on Medical Imaging*, Vol. 23, No. 9, pp. 1100–1108, 2004.

35. S. Owada, F. Nielsen, T. Igarashi, "Volume Catcher," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pp. 111–116, 2005.

36. Y. Li, J. Sun, C.K. Tang, H.Y. Shum. "Lazy Snapping," in *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 303–308, August 2004.

37. C. Rother, V. Kolmogorov, A. Blake, "GrabCut – Interactive Foreground Extraction using Iterated Graph Cuts," in *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 309–314, August 2004.

38. Y. Boykov, O. Veksler, R. Zabih. "Fast Approximate Energy Minimization via Graph Cuts," in *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 11, pp. 1222–1239, 2001.

39. Y. Boykov, M.P. Jolly. "Interactive Organ Segmentation using Graph Cuts," in Medical Image Computing and Computer-Assisted Intervention, pp. 276–286, 2000.

40. X. Yuan, N. Zhang, M.X. Nguyen, B. Chen. "Volume cutout," *in IEEE Trans. Visual. Comput. Graph.*, Vol. 21, pp. 745–754, 2005.

41. L.H. Staib, J. S. Duncan, "Deformable Fourier models for surface finding in 3-D images," in *Proceeding of SPIE: Visualization in Biomedical Computing*, Vol. 1808, pp. 90–104, 1992.

42. T. Jakobsen, T. "Advanced Character Physics," IO Interactive, Farvergade 2, DK-1463 Copenhagen K, Denmark, 2001

43. C. Murray, D. Merrick, M. Takatsuka, "Graph Interaction through Force-Based Skeletal Animation", in *Proc. Australasian Symp. on Information Visualisation* (InVis.au 2004), CRPIT 35, pp. 81–90, 2004.

44. D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, "Elastically deformable models," in *Proceedings of Computer Graphics* (SIGGRAPH'87), Anaheim, CA, Vol. 21, pp. 205–214, July 1987.

45. D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, pp. 413–424, 1986.

46. D. Zorin, P. Schröder, , W. Sweldens, "Interpolating Subdivision for Meshes with Arbitrary Topology," in *Proceedings of Computer Graphics* (SIGGRAPH 96), pp. 189–192, 1996.

47. U. Labsik, G. Greiner., "Interpolatory $\sqrt{3}$-Subdivision," in *Proceedings of Eurographics, Computer Graphics Forum*, Vol. 19, No. 3, pp. 131–138, September, 2000.

48. K. Singh, E. Fiume, "Wires: A Geometric Deformation Technique," in *Proceedings of Computer Graphics* (SIGGRAPH'98), pp. 405–414, 1998.

49. M. Beauchemin, K.P.B. Thomson, G. Edwards, "On the Hausdorff distance used for the evaluation of segmentation results," in *Canadian Journal of Remote Sensing*, Vol. 24, No. 1, pp. 3–8, 1998.

[illegible] BL - 15 - 45