1-1-2003

User-defined B-Spline template snake

Hoda Dehmeshki
Ryerson University

NOTE TO USERS

This reproduction is the best copy available.

User-Defined B-Spline Template Snake

by

Hoda Dehmeshki
B.Sc., Shahid Beheshti University, Iran, 1998

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, 2003

Instructions on Borrowers

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis has created a new Snake model that overcomes many of the limitations of the traditional finite difference snake. This new deformable model combines a novel user initialization process with a finite element B-spline snake to create a powerful semi-automatic segmentation method. Using the simple but powerful initialization process, the user recognizes critical points and regions in a specified order, and transfers this knowledge to the model. By drawing lines *across* the object of interest, important information pertaining to the global shape of the object, such as width and symmetry, is imparted to the model.

The snake is parameterized using minimum number of model degrees of freedom necessary and these degrees of freedom are placed in optimal positions around the object, based on the critical points and features recognized by the user via the input lines. Thus, the model is more like a deformable template than a local snake model- it is less sensitive to noise and more amenable to propagation to subsequent image slices in a volume image or time series. Unlike a traditional deformable template model however, it is constructed and positioned by the user rather than preconstructed and automatically initialized by the segmentation system. The template snake is initialized very close to the object boundary and is very similar in shape. Furthermore, it is "aware" of its position with respect to the object.

This thesis also describes the computation of the external image forces and how the known initial position and shape of the snake can be used to design object-specific image forces.

Keywords: snakes, B-spline curves, medical image segmentation, interactive image analysis.

Acknowledgments

I would like to thank my supervisor, Dr. Tim McInerney, for his guidance, advice, and support. His extraordinary patience and positive outlook always inspired me and gave me confidence to continue.

I would also like to thank the members of my thesis committee, Dr. Sridhar Krishnan, Vadim Geurkov, Rafeef Abu-Gharbieh, and Rachel Jiang for their time and effort in reviewing my thesis and their helpful feedback.

I wish to express my deepest gratitude to my parents for their unconditional love and constant encouragement.

I also thank the Ryerson University for financial assistance.

Last, but certainly not least, my utmost thanks goes to my husband, Hojjat Ghaderi, for his love, support, and understanding. I cannot imagine completing this thesis without his encouragement and support. I dedicate this thesis to him.

Contents

1 Introduction **1**
1.1 Thesis Outline 4

2 Snake Fundamentals **5**
2.1 Introduction 5
2.2 Snake Representation 5
2.3 Snake Energy Functional 6
2.3.1 Internal Deformation Energy 7
2.3.2 External Image Energy 7
2.3.3 Constraint Energy 8
2.3.4 Euler Equation for Snakes 9
2.3.5 Inflation Force 9
2.4 Dynamic Deformable Models 10
2.5 Snake Discretization 10
2.6 Finite Element Snakes 11

3 Motivation and Background **13**
3.1 User Interaction in Semi-automatic Segmentation 13
3.2 Active Contour Models (Snakes) 17
3.2.1 GVF(Gradient Vector Flow) Snakes 20
3.3 Live Wire 22
3.4 Deformable Templates 25

4 Finite Element Snakes Formulation **28**
4.1 Finite Element Method(FEM) 28
4.2 Finite Element Snake 30
4.3 Galerkin Finite Element Formulation Of Snakes 31
4.3.1 Formulation Along the X And Y Directions 34
4.3.2 Global Assembly 35
4.4 Snakes Deformation 35
4.4.1 Hermite Snake 37
4.5 B-Spline Shape Functions 38

5 User-Defined B-Spline Template Snakes **40**
5.1 Initialization Process 40
5.1.1 Editing the control polygon 42
5.1.2 On-the-fly Segmentation 42
5.2 Object-Specific External Image Forces 43
5.2.1 Snake-Point Control 43
5.2.2 Knowledge-based Image Feature Search 44
5.2.3 User-input Derived Forces 45
5.2.4 Local Intensity Statistics 46
5.3 Object-Specific Global Shape Constraints 46
5.4 Intelligent Fitting Algorithms 49

6 Results **51**
6.1 Corpus Callosum(CC) Segmentation 51
6.2 Bone Segmentation 56
6.3 Bladder Segmentation 58
6.4 Putamen Segmentation 58

7 Conclusion **60**

A Hermite and B-Spline Curves: An Overview **62**
A.1 B-Spline Approximation 62
A.2 Uniform Closed Periodic B-Splines 63
A.3 Hermite Interpolation 64

B Finite Difference Snakes **66**

C Euler Equations for Snakes **68**

List of Figures

1.1 Example of initialization process. In (a) the user enterslines, starting from the left side of the corpus callosum andproceeding to the right. In (b) the B-spline control polygon isshown, (c) initial B-spline snake, (d) segmentation result. 3

5.1 Example of initialization process. In (a) the user enters lines, starting from the left side of the corpus callosum and proceeding to the right. In (b) the B-spline control polygon is shown, (c) initial B-spline snake, (d) segmentation result. 41

5.2 Example of adding a new point. In (a) the user enters lines. In (b) the control polygon and P0 (the user added point) are shown. In (c),the corresponding line is found and divided into two, where the point is an end point for the two lines 43

5.3 Example of on-the-fly segmentation. Partially segmented object after drawing(a) three lines (b) four lines (c) five lines (c) six lines. (d) Final segmentation result. 44

5.4 Example of searching along the normal direction for 3 points on an snake element. 44

5.5 Example of pin spring forces.(a) shows the initial curve and control points (used as pin points) at the beginning of phase one. (b) shows the curve at the end of phase one. 46

5.6 Example of segmentation using local intensity statistics around user specified lines. (a)User-input lines (b)Areas where intensity statistics are collected. After this, for each snake point the intensity statistics from its closest area will be used. (c)Final Result. 47

5.7 (a) A snake with global constraint. (b) effect of a user-click on the snake: because of the global constraint the whole snake is pulled toward the user clicked point. 47

5.8 Example of global shape constraint. The three control points within the triangle are connected together with springs, forcing this region of the snake to remain triangular in shape. The snake is initialized incorrectly in (a) but manages to move towards the correct position in (b) so that the rostrum tip of the snake ends up at the rostrum tip of the data. 48

5.9 Example of intelligent fitting. (a) user-input lines, (b) initial B-spline snake, (c) snake after phase I, (d) final result without using ziplock in phase II, (e-i) using ziplock in phase II -circles show image forces-: (e) after 100 iterations, (f) 200 iterations, (g) 300 iterations, (h) 400 iterations, (i) final result. 50

6.1 Samples of segmenting CCs images. The first column is user init lines, the second column is initial snake, and the last column is final result. 53
6.2 Samples of segmenting CCs images (continued from figure 6.1). 54
6.3 Samples of segmenting CCs images (continued from figure 6.2). 55
6.4 Complex CC images segmented with 6 user init lines. 56
6.5 Segmenting two arm bones in an X-ray image. From left to right: user init line, initial snake, and the final result. 57
6.6 Segmenting the balder using 5 input lines. 58
6.7 The putamen (left) and its potential (right). Notice that even with lots of enhancements, the edges are still quite noisy. 59
6.8 Segmenting a single frame potamen, using 5 input lines. 59
6.9 Segmenting a sequence of putamen frames (from left to right). There is no need for user initialization, instead the result of each frame is used as the initial snake for the next frame. 59

List of Tables

6.1 Mean, min, max and standard deviation of the shortest distances between automatically extracted and expert segmented CC boundaries.The last column is pixel distance between the automatically and manually labeled rostrum tip. 52

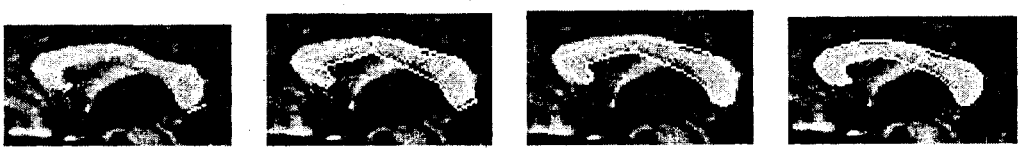
Chapter 1

Introduction

Segmentation, partitioning an image into homogeneous areas, is adifficult intermediate step in the analysis of digital images. Inthe medical imaging field, the extraction of anatomical structuresfrom the background and from each other is crucial to a host ofmedical image analysis (MIA) tasks, such as measurement,visualization, matching and labeling, reconstruction, surgicalplanning, and compression. The ultimate goal of medical imagesegmentation is the development of fully automatic techniques thatguarantee maximum repeatability, robustness, accuracy, andefficiency.Fully automatic segmentation systems have proved extremelyelusive. The complexity and variability of anatomic shapes ofinterest, the significant variation between images, anatomicstructure abnormalities, nonuniform image acquisition, and thesheer size of the data sets have created imposing barriers to thedevelopment of robust systems. Furthermore, the problems ofsampled data - noise, low contrast, partial volume pixels,sampling artifacts etc. - often result in indistinct, fuzzy and/ordisconnected object boundaries. The challenge is to extract theboundary elements belonging to the target anatomical structure,and integrate these elements into a complete and consistent modelof that structure. Researchers have struggled to effectivelyutilize prior knowledge of structure shape, position, orientation,symmetry, relationships to neighboring structures, associatedlandmarks, and plausible image intensity characteristics in orderto meet this challenge.At the other end of the spectrum, manual segmentation via boundarytracing is extremely labor-intensive, time-consuming, anderror-prone. Traditional low-level image processing techniques,such as

thresholding and region growing, can be effective for asmall set of specific segmentation problems. However, theyconsider local intensity information only and therefore often makeincorrect assumptions during the boundary element integrationprocess, resulting in infeasible object boundaries. As a result,these model-free techniques usually require considerable amountsof expert intervention. Consequently, a more immediate andsignificant impact on MIA may be realized by optimizing thecapabilities of *model-based* semi-automatic segmentationtechniques, to the point where only a small amount of time andlabor is required to process complex data sets. To achieve thisgoal, the recognition capabilities of the human expert must befully exploited. Model-based semi-automatic techniques that assistthe human expert in the extraction of the structures must bedesigned to not only be fast and intuitive, but also permit theinteractive transfer of structure shape and appearance knowledgefrom the expert in order to ensure segmentation accuracy,robustness and reproducibility with a minimal user editing phase.This is especially important when processing a large number ofimage slices from a volume image or a time series, or whenprocessing very noisy images.Deformable curve (and surface) models, such as Snakes [17]and their variants have become widely popular in medical imagesegmentation and are still intensively applied and researched[23]. A snake is an interactiveflexible contour model that is placed in the vicinity of thestructure to be segmented and then iteratively adjusted to fit theboundary of the structure. The shape of the snake in an image istypically dictated by an energy functional and the final shape ofthe contour corresponds to the minimum of this energy. Deformablemodels are particular well-suited for segmentation of images thathave artifacts, noise, and weak boundaries between structures.Although deformable models are powerful segmentation techniques,they are still prone to latching onto spurious or neighborstructure boundaries and they remain sensitive to theirinitialization. Also, the traditional Snake initialization processof roughly tracing around the object boundary to create theinitial contour, continues to be a tedious and time-consumingtask. The difficult challenge in improving the Snakes technique isto develop more effective user initialization mechanismsintegrated with control mechanisms that can guide the energyminimization process at an appropriately high level of abstraction[22].This thesis

describes a simple but powerful custom Snakeinitialization process to meet this challenge. The initializationprocess uses the human expert to recognize landmarks and othercritical shape features and transfer this information in such away that the snake is explicitly 'aware' of where it is in theimage, how its 'parts' are arranged, and what structure it issegmenting. The process is intuitive, efficient and general andmakes use of simple line primitives that are quickly drawn acrossthe target structure at critical points in a pre-specified order.These line primitives are then used to construct a control polygonof a finite element B-spline snake[19]. By taking advantage of theproperties of B-splines, a model is created that is more like atemplate - a snake constrained by its control polygon that isinitially extremely close to and similar in shape to the targetstructure. The initialization process acts as almost apre-segmentation and labeling step, making the model's job muchsimpler and hence more likely to succeed without user editing. Byimposing an order on the initialization process and by drawinglines across the target structure, the human expert is able totransfer knowledge of global shape, symmetry, landmark position,image appearance etc. to the model. This information can then beutilized by high-level snake fitting algorithms. Finally, therecognition and identification of critical shape features by theexpert also provides key information to subsequent shape analysisand comparison. In order to compute a minimum energy solution



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

Figure 1.1: Example of initialization process. In (a) the user enterslines, starting from the left side of the corpus callosum andproceeding to the right. In (b) the B-spline control polygon isshown, (c) initial B-spline snake, (d) segmentation result.

of a Snakenumerically, the energy must be discretized. The Finite Element(FE) method and the Finite Difference (FD) method are the two mostcommon approaches for representing a contour in terms of linearcombinations of local-support basis functions. Using FD methods,deformable models are represented as a finite ordered set ofvertices. The derivatives at

each point are then replaced by afinite difference approximation. In the FE method, the domain isdivided into sub-domains commonly referred to as *finiteelements*. The contour is represented within each element by aninterpolating polynomial which is continuous along with itsderivatives to a specified order within the element. In thisthesis the FE method is used for the following reasons:

- The FE approach is more compact - it requires fewerdiscretization points and consequently produces a smaller linearsystem to solve, thus reducing significantly the algorithmiccomplexity.
- The FE approach produces more accurate result.
- The FE method provides an analytical representation of theentire contour rather than only at discrete points.

1.1 Thesis Outline

Chapter 2 and 3 presents a comprehensive review of semi-automaticdeformable models, their use for medical image segmentation, andtheir mathematical foundations.Chapter 4 presents a development of the finite element formulationof deformable models including an FE B-Snake where the basisfunctions are cubic B-splines.Chapter 5 describes the new snake initialization process in detailand how the known initial position and shape of the snake controlthe optimization-driven segmentation process.In Chapter 6, the B-Spline template snake is applied to several2D images in order to demonstrate its potential. Some of theimages are very noisy with many large gaps in the boundaries.

Chapter 2

Snake Fundamentals

2.1 Introduction

Active contour models, or Snakes, were first introduced in 1988 by Kass, Witkin, and Terzopoulos as a semi-automatic segmentation technique. Snakes have been successfully applied to many image analysis tasks, such as motion tracking and analysis, matching (labeling, registration), and shape recognition.A Snake is essentially an elastic contour that is initialized by the user close to the boundary of the target object. The Snake converges towards the object boundary by minimizing an energy functional which controls the shape of the snake. The energy functional generally consists of two terms - one controlling the smoothness of the snake and the other coupling the snake to the image. The complete snake framework is specified by: a model representation, an energy functional, and an optimization method. The remainder of this chapter presents a complete review of the Snakes framework.

2.2 Snake Representation

Deformable models such as Snakes use either continuous or discrete geometric representations. With discrete representations, the geometry of the model is only known at a finite set of points. Discrete representations are simple, efficient and can be easily locally subdivided to add degrees of freedom in areas where the object boundary exhibits rapid variation 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 a representation between model points. Continuous representations must be discretized for computational needs but they are generally more compact and hence provide the same level of accuracy with fewer degrees of freedom.

Furthermore, they offer the ability to compute differential quantities such as normals or curvature almost everywhere on the curve, and are more amenable to guidance by a higher-level control mechanism. In continuous representations, Snakes may be defined through implicit or a parametric(explicit) equations. An implicit curve is generally defined as the zero level set of functions f in $\Re$ as follows:

$$s(\mathbf{p}) = \{\mathbf{p} \in \Re^2 | f(\mathbf{p}) = 0|\}.$$

When using parametric equations, snakes are represented as $\mathbf{v}(s) = (x(s), y(s))^\top$, where x and y are coordinate functions and $s \in [0, 1]$ is the parametric domain. The comparison of the efficiency and implementation of these two frameworks is difficult because of the large variety of existing algorithms. However, in general, parametric representations are regarded as being more efficient, compact and easier to implement than the implicit form. In addition, they can describe open snakes. In the thesis, we focus on parametric form.

2.3 Snake Energy Functional

The energy functional controls the shape of the snake subject to an image $I(x, y)$. The final position of the contour corresponds to a local minimum of E_{snake}. It consists of internal, external and constraint energies as follows:

$$E^*_{snake}(\mathbf{v}(s)) = \int_0^1 (E_{internal}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{constraint}(\mathbf{v}(s)))\, ds. \qquad (2.1)$$

$E_{internal}$ represents the internal deformation energy of the contour due to stretching and bending and serves as a smoothness constraint. E_{image} gives rise to the image forces that push the snake towards salient image features like edges. $E_{constraint}$ gives rise to the external

constraint forces such as user-initiated mouse forces which push or pull points on the snake. In the following sections we describe each of these terms in more detail.

2.3.1 Internal Deformation Energy

The internal deformation energy acts as a regularizer and imposes a smoothness constraint (elasticity and rigidity) on a snake. The most commonly used internal energy consists of two terms:

$$E_{internal} = E_{tension}(\mathbf{v}(s)) + E_{rigidity}(\mathbf{v}(s)). \tag{2.2}$$

The first term, $E_{tension}$, makes the snake act like a membrane. It controls the elasticity or stretchiness of the snake and attempts to minimize the overall length of the shape. In the absence of an external energy, this term will cause the snake to shrink to a point. It is defined as follows:

$$E_{tension}(\mathbf{v}(s)) = \frac{\alpha(s)}{2}\left|\frac{\partial \mathbf{v}(s)}{\partial s}\right|^2 = \frac{\alpha(s)}{2}\left((\frac{\partial x(s)}{\partial s})^2 + (\frac{\partial y(s)}{\partial s})^2\right). \tag{2.3}$$

The second term, $E_{rigidity}$, makes the snake act like a wire(i.e. controls the bending energy) and attempts to minimize the overall curvature. In the absence of an external energy, this term will cause an open snake to become a straight line and a closed snake to become circular. It is defined as follows:

$$E_{rigidity}(\mathbf{v}(s)) = \frac{\beta(s)}{2}\left|\frac{\partial^2 \mathbf{v}(s)}{\partial s^2}\right|^2 = \frac{\beta(s)}{2}\left((\frac{\partial^2 x(s)}{\partial s^2})^2 + (\frac{\partial^2 y(s)}{\partial s^2})^2\right). \tag{2.4}$$

$\alpha(s)$ and $\beta(s)$, can be used to control the tension and rigidity along different parts of the contour. For example we can adjust weights to make the contour more elastic than rigid. For simplicity, in most applications they are set to constant values. The values of these constants are typically based on empirical results.

2.3.2 External Image Energy

To couple the snake to images, external potential energies are designed such that their local minima coincide with intensity extrema, edges, and other image features of interest. For

example,the snake will be attracted to intensity extrema in $I(x,y)$ by setting the potential to:

$$E_{image} = w_{extrema} G_\sigma(x,y) * I(x(s), y(s)), \tag{2.5}$$

where the sign of the weight $w_{extrema}$ controls whether the snake will be attracted to dark or bright regions. Note that G_σ denotes a Gaussian smoothing filter of standard deviation σ. The snake will be attracted to intensity edges by choosing a potential

$$E_{image} = -w_{edge} \|\nabla[G_\sigma(x,y) * I(x(s), y(s))]\|^2, \tag{2.6}$$

where w_{edge} controls the magnitude of the potential and ∇ is the gradient operator. The gradient of E_{image} gives rise to external force vectors which are normal to the image edges. Therefore, deformable models initialized close to an edge will coverage to a stable configuration near it. Other edge-detectors, such as the Sobel or Canny edge detectors, are often used rather than the gradient magnitude. Note that a larger σ of the Gaussian smoothing filter will blur the edges. Such large σ are often necessary to increase the capture range of the deformable contour.

2.3.3 Constraint Energy

Constraint energy terms are designed to allow user interaction with the snake or to incorporate prior shape knowledge about the target object. The original paper [17] 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 spring can be a fixed position, a snake point, or any point on the image. Creating a spring between point $\mathbf{v}_1$ and $\mathbf{v}_2$ adds $-k(\mathbf{v}_1 - \mathbf{v}_2)$ to the external constraint energy, $E_{constraint}$. The opposite of the spring constraint, a volcano pushes the snake out of one local minimum and into another by adding energy function $\frac{1}{r}$ to the constraint energy. This energy gives rise to a repulsion force $\frac{1}{r^2}$ controllable by the mouse.

2.3.4 Euler Equation for Snakes

In accordance with the calculus of variations, the contour $\mathbf{v}(s)$ which minimize the snake energy must satisfy the Euler equation (see appendix C for details):

$$-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}} = 0. \tag{2.7}$$

Equation 2.7 expresses the balance of internal and external forces when the contour rests at equilibrium. The first two terms represent the internal stretching and bending forces respectively, while the third term represents the external forces that couple the snake to the image data. The usual approach to solving 2.7 is through the application of numerical algorithms (section 2.5).

2.3.5 Inflation Force

One of the shortcomings of the original snake is that it must be initialized close to the boundaries of the target object. The external image edge energy (Eq.2.6) gives rise to image forces in the Euler equations. These are short-range forces with their range depending on the width of the Gaussian smoothing filter. Cohen [4] proposed an inflationary force which pushes the contour outward.The contour must be initialized inside the target object but it no longer needs to be close to the boundaries. The contour passes over weak edges and is stopped only if the edge strength is strong. The inflation force is defined as follows:

$$F_{balloon} = k_1\mathbf{n}(s),$$

where $\mathbf{n}(s)$ is the unit normal vector at point $\mathbf{v}(s)$,and k_1 is the amplitude of the inflation. The external image forces now become:

$$F = k_1 n(s) - k\frac{\nabla E_{image}}{\|\nabla E_{image}\|}(\mathbf{v}(s)). \tag{2.8}$$

Coefficients k_1 and k are chosen such that they are of the same order, which is smaller than a pixel size (unit length). $k$is set slightly larger than k_1 so an edge point can stop the inflation force of the corresponding snake point. Changing the sign of k_1 will have an effect of deflation instead of inflation. The main problem with inflation forces is that the Snake can 'leak' through gaps in the boundary edges of a noisy image.

2.4 Dynamic Deformable Models

While it is natural to view energy minimization as a static problem, a another approach is to construct a dynamical system where the contour shape is a function of t(time) and s and apply the principles of Lagrangian mechanics. This formulation leads to dynamic deformable models, making it possible to quantify not just the static shape of the model but also its shape evolution through time (i.e. its motion). Dynamic models exhibit intuitively meaningful physical behaviors, making their evolution amenable to interactive guidance from a user. A simple example is a dynamic snake which can be represented by introducing a time-varying contour $\mathbf{v}(s,t) = (x(s,t), y(s,t))^\top$ with a mass density $\mu(s)$ and a damping density $\gamma(s)$. The Lagrange equations of motion of a dynamic snake are

$$\mu\frac{\partial^2\mathbf{v}}{\partial t^2} + \gamma\frac{\partial\mathbf{v}}{\partial t} - \frac{\partial}{\partial s}\left(\alpha\frac{\partial\mathbf{v}}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2\mathbf{v}}{\partial s^2}\right) = -\nabla E_{image}(\mathbf{v}) + f. \tag{2.9}$$

The first two terms in the above equation represent inertial forces due to the mass density (μ) and damping forces due to the dissipation density(γ). The next two terms represent the internal stretching and bending deformation forces. On the right hand side are the external forces, where $\nabla E_{image}(\mathbf{v})$ is the negative gradient of the image potential energy function and $f(s,t)$ are time-varying interaction forces applied by user (usually through a mouse). Other forces, such as an inflation force, may also appear on the right hand side. Note that the inertial forces are typically only used in object tracking applications. Consequently μ is most often set to 0and the system retains only first order dynamics. Equilibrium is achieved when the internal and external forces balance and the contour comes to rest (i.e. $\frac{\partial\mathbf{v}}{\partial t} = \frac{\partial^2\mathbf{v}}{\partial t^2} = 0$ which is equal to Euler equation).

2.5 Snake Discretization

In order to numerically compute a minimal energy solution numerically, the snake energy function (C.2) must be discretized. The typical approach is to represent the continuous geometric model $\mathbf{v}$ in terms of linear combinations of local support basis functions. Finite Elements(FE)and Finite Differences(FD) are the two most common local representation

methods, transforming $\mathbf{v}(s)$ into a discrete vector $\mathbf{u}$ of shape parameters. The discrete form of snake energy may be written as follows:

$$E(\mathbf{u}) = \frac{1}{2}\mathbf{u}^\top \mathbf{K}\mathbf{u} + E(\mathbf{u}),$$

where $\mathbf{K}$ is called the *stiffness* matrix and $E(\mathbf{u})$ is the discrete version of the external potential. The discrete form of the equations of motion (2.9)can be written a set of second order ordinary differential equations for $\mathbf{u}(t)$ as follows:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}, \tag{2.10}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is a damping matrix and $\mathbf{f}$ is the external force vector. The time derivatives are approximated by finite differences and explicit or implicit numerical time-integration methods are applied to solve the system of equations.

Finite difference method(FDM) is the traditional method for local representation of snakes. A $2D$ FD snake contour is represented as an ordered set of discrete points $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \cdot, \mathbf{v}_n)$ where $\mathbf{v}_i(s) = (x_i(s), y_i(s))^\top$, with $\mathbf{v}_1 = \mathbf{v}_n$ for closed snakes. The points (vertices) are connected by straight lines, resulting in a piece-wise linear curve.(see appendix B) As an alternative to FDM, one may use Finite Element method(FEM).

2.6 Finite Element Snakes

The Finite Element Method (FEM) is a computer-aided mathematical technique for obtaining approximate numerical solutions to equations where exact solutions are difficult or even impossible to find. Using the FEM, the domain Ω is subdivided into finite number of simpler sub-domains, called *finite elements.* Each element(segment) is represented geometrically with specific shape functions and its corresponding nodes. FE snakes are represented as follows:

$$x(s,t) = \sum_{i=1}^{N} x_i(t) B_i(s), \tag{2.11}$$

$$y(s,t) = \sum_{i=1}^{N} y_i(t) B_i(s), \tag{2.12}$$

where $B_i(s)$ are basis functions or shape functions and(x_i, y_i) are unknown variables(usually called degrees of freedom) whose values must be computed. Different basis functions generate different snakes, each having specific advantages and disadvantages. In chapter4 FEsnakes with B-spline and Hermite Basis functions will be discussed in detail.

Chapter 3

Motivation and Background

3.1 User Interaction in Semi-automatic Segmentation

The most important criteria to evaluate semi-automatic segmentation techniques are accuracy, efficiency, and repeatability. Accuracy is defined as the degree to which the delineation of the anatomical structure agrees with the truth. Accuracy is typically measured subjectively by a human expert, or objectively by comparing the segmentation result against the ground truth using various metrics. Examples of these metrics are the difference in the object area or the average distance between points on the segmentation result and closest points on the ground truth boundary. Note that accuracy as a criterion makes more sense in the evaluation of fully automatic segmentation algorithms. In interactive methods, the user may improve the accuracy to the desired level unless the method does not provide sufficient user control. Therefore, a semi-automatic method can be considered accurate when it provides sufficient control to the user.

Repeatability is measured as the difference between the segmentation result over different segmentation sessions with the same target object. The smaller the difference, the higher is the degree of repeatability. Furthermore, the difference must be minimal not only when one user performs the segmentation in different sessions (known as intra-operator bias), but also when multiple users perform the segmentation of the same object in a single or several segmentation sessions (inter-operator bias). The difference in the segmentation results can be due to the difference in the operation of the segmentation tool (for example, the user

clicks the mouse at different positions in the image in different sessions) or a difference in user judgment of the correct object delineation. Nothing can be done about the latter but the method must minimize the effects of the former. This can generally be achieved when the segmentation result is generated mostly by the computational part of the method.

Efficiency can be defined in several ways. The total elapsed time to perform the segmentation is one way to measure efficiency. However, this definition may be too task dependent and it also may hide the contributions of the computational and interactive parts of the method. An alternative definition measures the amount and nature of user interaction (for example, the degree of interaction might be the number of mouse clicks required), with the goal of evaluating the overall effort required by the user to perform the segmentation. Efficiency of the computational part is secondary as long as the method is fast enough to allow for interaction in real-time.

Evaluating user effort (and hence efficiency) in terms of the nature of the interaction requires the measuring of task complexity. Task complexity involves several issues. One issue is the type of mouse operations required by the segmentation method (the use of alternative input devices, such as a pen and tablet, and their effect on the interaction efficiency is beyond the scope of this thesis). For example, the effort to control the mouse increases with the requirement to trace the object boundary more carefully, resulting in reduced efficiency and user fatigue. Another issue is the type of knowledge required by the user to input data during interaction. For example, if the user is required to enter a parameter value for the method, some knowledge about inner workings of the segmentation algorithm are needed. This may lead to reduced efficiency as it separates the effect of the user input from the effect on the segmentation tool. For efficient interaction, the user should be able to predict the impact of the interactions on the segmentation result and visual feedback of the effect of the interaction should be provided in real-time.

In semi-automatic techniques, the goal is to combine the user and a computer to obtain an accurate delineation of the target object as efficiently as possible. The assumption is the user knows what the correct delineation and his/her role is to guide the computational part

of the method to find it efficiently. Specifically, the user's role typically consists of some or all of the following actions: judge the correctness of the computational part of the method and edit the results, set parameters for the computational part, select among different results which are generated automatically.

Although editing of the segmentation result enhances its accuracy, it can reduce efficiency and hence increase segmentation time, and produces non-repeatable interaction, especially if the editing is performed at the pixel level. Using curve manipulation tools of some boundary-based methods is more effective.

Setting parameter values is performed at initialization or while the algorithm is running (known as steering). At initialization, the user sets parameters and the computational part is executed and the result is displayed on the screen for user evaluation. If the result is not satisfactory, the user may initialize again and the computation part is repeated. The parameters often affect the computation globally, an undesirable characteristic. Another example of setting parameters values is where the user draws an initial contour of a deformable model, and configures the model parameters such as the weights controlling the balance between internal and external model constraints. Steering involves dynamically providing local information indicating the desired segmentation outcome, guiding the computational part in a process of progressive refinement. The information input by the user is used to locally adjust the parameter values so that regions where the delineation is correct remain unchanged. When properly designed, interactive steering has the following advantages: manual editing at the end is not necessary, integration of computation and interaction into one process produces a result with uniform properties, and finally, knowledge about the segmentation problem can be updated based on user modification. Live wire is an example for this approach. In live wire, wherever the user moves the mouse, the shortest path between initial point and the current mouse position is determined and the resulting line is shown to the user in real-time. If the result is not satisfactory, user again moves the mouse to get a better result.

When the user's role is to select among different results, the idea is to let the computer find a number of plausible delineations and then the user decides on the most reasonable

one. Since the final result is mostly generated by the computational part, interaction can be very efficient. Also, the result is repeatable and has uniform properties since it is generated by one process. It should be noted that accuracy of the final result depends on the success of the segmentation method.

In summary, the following design principles should be followed to produce an efficient interactive segmentation method that generates accurate and repeatable results [30]:

1. Computation and user interaction should be integrated into one process.
2. Use pictorial input to the computational part
3. User interaction should be minimized by including options to select from.
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 whole process to generate accurate result.
6. Proper visualization of the computational part is needed to enable an effective user response.
7. Emphasize computation after each user interaction for optimal repeatability.
8. Add intelligent behavior to elevate the abstraction level of the interaction.
9. Add the capability to learn from the user interaction and consequently eliminate the need for further interaction.

In the following sections, the most currently popular model-based segmentation techniques, specifically deformable models and livewire, are reviewed and discussed with respect to these strategies.

3.2 Active Contour Models (Snakes)

As described in Chapter 2, one of the most well known deformable models are Active Contour Models, commonly known as Snakes [17]. Snakes are energy-minimizing contours controlled by internal and external energies. The internal energy enforces smoothness of the contour and the external energy attracts the contour towards salient features in the image. An iterative procedure is used to minimize the energy causing the snake to converge (i.e. lock on) to the boundary of the target object in the image. Unlike more traditional techniques such as edge linking, the model-based snake proposed by Kass, Witkin, and Terzopoulos [17], considered the target object boundary as single connected structure and designed to be interactive via constraint energy terms. Users are able to intuitively pull and push on the snake, pulling it off of spurious or incorrect image edges onto the correct edges (and hence moving it out of one local minima into another). Although Snakes introduced a new and powerful approach to image segmentation, the original formulation in [17] had several limitations that mitigated their utility across the full range of MIA problems and limited their potential for automation (and hence affected their degree of repeatability and efficiency for some tasks):

- Sensitivity to parameter settings. As described in the previous chapter, Snakes performance is strongly influenced by parameters α (internal elastic force parameter), β(internal bending force parameter), the strength of the external image force term, and the relative weighting of these terms. Unfortunately, these parameters depend heavily on the underlying image and object of interest and optimal settings are difficult to derive and are often based on empirical evidence rather than mathematical theory or a higher-level learning algorithm. Proper assignment of these parameters usually is a time-consuming task.

- Small capture range. There is no external force on snake points in regions of homogeneous image intensity as the gradient of the image in these areas is almost zero. This can be ameliorated somewhat by gradually adjusting the degree of Gaussian blurring

using the σ parameter but then another parameter is introduced into the mix. In addition, many optimization techniques like the variational calculus used to minimize the energy functional are local. Therefore snakes have a very small capture range and must be initialized close enough to the object boundary to ensure convergence. Otherwise the user must spend time pulling the snake onto the boundary.

- Failure to detect boundary concavities. A traditional snake may fail to converge to concave boundary regions or cavities. The Snake is roughly initialized around the target object. The external force acting on a snake point in the concave regions pulls it in a sideways direction towards the closest edge and there is no force acting inwards pulling the point further into the concave region towards the inner edge.

- Geometric and topological inflexibility. The original Snakes model had a fixed geometric parameterization and this fact in conjunction with the internal deformation energies made it incapable of changing its topology (connectivity) or conforming to long tubular shapes or shapes with significant branching or protrusions without significant and tedious user initialization.

To overcome these limitations, researchers have developed various constraint energies and forces to the parametric Snakes model formulation. Balloon (or inflation) forces were introduced [5]. An inflation force significantly increased the capture range of a traditional snake but it must be set to push outwards or push inwards, a condition that mandates careful initialization. If a snake is initially drawn so that part of it is outside the target object and part is inside, then an incorrect segmentation will result. To overcome this problem, several researchers have created automatic snake element subdivision mechanisms [24]. In this case, the model can be simply initialized using one mouse click as a small circle inside (or outside) the target object. The inflation force expands the model and the model automatically subdivides, allowing it to 'grow' into complex shapes. As the model approaches the boundary of the target object, external image forces oppose the inflation, stopping the model on the boundary. However, this added feature does not come without

a cost. Some interactive control (steering capability) over the model is lost and there are typically always gaps in the object boundaries where the model will leak through. Another interesting solution to the boundary concavity and small capture range problems is the use of a gradient vector flow field (GVF) [36] as an external driving force. This force is detailed in the next section.

To introduce both geometric flexibility (like the automatic subdivision models) and topological flexibility, several researchers have developed implicit snakes to the image segmentation problem [2]. These models are formulated as evolving contours or 'fronts' which define the level set of some higher-dimensional surface over the image domain. The main feature of this approach is that the topology changes of the contour are handled naturally, since the level set of the higher-dimensional surface need not be simply connected. While this is an attractive mathematical technique, implicit model formulations are not nearly as convenient as explicit parametric formulations when it comes to incorporating additional constraints and user interactive guidance. Note that McInerney and Terzopoulos developed a topology adaptive Snake that retains the explicit parametric form [24].

All of these Snakes extensions and re-formulations can and do work well when the image feature map is relatively clean. However, most clinical images are noisy, contain many uninteresting edges, contain regions of low contrast or have gaps in the object boundary, or texture is present. Hence, these more automatic techniques may not work as expected and are sensitive to parameter settings. Furthermore, the loss of interactive model control may not be offset by the gain in automation. Other researchers [14, 3, 12, 21] have integrated object region information in addition to the traditional object edge information into deformable contour models in an effort to decrease sensitivity to noise, insignificant edges and texture and initial model placement. One problem with these techniques is that regional image intensity information is not uniform over the entire target object.

The use of global energy minimization approaches, such as dynamic programming and simulated annealing, have been proposed [1, 27, 13] as an alternative to the more common local optimization schemes. The speed of convergence of these techniques is an issue as is

the ability to describe the correct segmentation result using energy functionals. That is, global optimization schemes will find a global minimum, but this may not correspond to the desired result.

B-spline Snakes have been proposed by several researchers. These models are built using parametric B-spline curves and may offer certain advantages for many segmentation problems over the traditional finite difference Snakes. They provide a compact, local representation of a curve, in terms of its control points. B-Snakes are expressed as a linear combination of the set of B-spline basis functions and the shape of the object is defined by the coefficients of this linear combination. B-snakes significantly converge faster than original snakes, the smoothness is implicitly built into the model, they provide an analytic representation over the whole snake, and they require fewer degrees of freedom for the same level of accuracy as finite difference Snakes. They also provide powerful contour editing semantics that have been well developed for CAD/CAM applications and can be exploited for segmentation problems. Finally, the control polygon can be utilized as an effective constraint framework - one of the main topics of this thesis.

3.2.1 GVF(Gradient Vector Flow) Snakes

Gradient Vector Flow(GVF) Snakes were introduced by [36] to overcome some of the limitations mentioned previously. The vector force field is used as long range forces by the Snake points, pushing each point towards the closest edge. The specific advantages of the GVF snake over traditional model is its insensitivity to its initialization, and its ability to move into boundary concavities. The initial snake can be inside, outside, across and far away from the object's boundary. Unlike deformable models that use inflation focus, a GVF snake doesn't need prior knowledge about whether to shrink or expand toward the boundary. The GVF snake also has a large capture range. This is achieved through a generalized diffusion process that does not blur the edges themselves.

The GVF snake is defined as a contour $\mathbf{x}(s,t) = (x(s,t), y(s,t))$ which satisfies the

following Euler equation

$$-(\alpha \mathbf{x}_s)_s + (\beta \mathbf{v}_{ss})_{ss} + \mathbf{V}(\mathbf{x}) = 0, \tag{3.1}$$

where $\mathbf{V}(\mathbf{x})$ is gradient vector flow and a substitution for the potential force ∇E_{ext} in (B.10) and (B.11). It is defined as the equilibrium to the following vector diffusion equation:

$$\mathbf{u}_t = g(|\nabla f(\mathbf{x})|)\nabla^2 \mathbf{u} - h(|\nabla f(\mathbf{x})|)(\mathbf{u} - \nabla f(\mathbf{x})), \tag{3.2}$$

with the following boundary condition:

$$u(\mathbf{x}, 0) = \nabla f(\mathbf{x}). \tag{3.3}$$

In 3.2, the first term on the right is referred to as the smoothing term since this term alone will produce a smoothly varying vector field. The second term is referred as data term since it encourages the vector field $\mathbf{u}$ to be close to ∇f computed from the data. $f(\mathbf{x})$, called edge map, is derived from the image $I(\mathbf{x})$ and has the property that it is larger near the image edges. It can be defined as $f(\mathbf{x}) = E_{ext}(\mathbf{x})$. The weighting functions$g(.)$ and $h(.)$ apply to the smoothing and data terms, respectively. They can be chosen as follows:

$$g(|\nabla f|) = \mu, \tag{3.4}$$

$$h(|\nabla f|) = |\nabla f|^2. \tag{3.5}$$

Since $g(.)$ is constant, smoothing occurs everywhere. Weighting function $h(.)$ grows larger near strong edges and get it's maximum values at the boundaries. Using these weighting functions provides good edge localization. One problem is when there are two edges in close proximity, such as when there is a long, thin indentation along the boundary. In this case, GVF tends to smooth between opposite edges, losing the forces necessary to drive a deformable contour into this region. To address this problem, they must be defined so that $g(.)$ gets smaller as $h(.)$ becomes larger. Then, in the proximity of large gradients, there will be very little smoothing and the effective vector field will be nearly equal to the gradient of the edge map. One example of such pairs of weighting functions is as follows:

$$g(|\nabla f|) = e^{-(\frac{|\nabla f|}{K})^2}, \tag{3.6}$$

$$h(|\nabla f|) = 1 - g(|\nabla f|). \tag{3.7}$$

The GVF field computed using such weighing functions will conform to the edge map gradient at strong edges, but will vary smoothly away from the boundaries. The specification of K determines to some extent the degree of trade-off between field smoothness and gradient conformity.

A main criticism of the GVF technique is that it does not work as well in noisy images. The resultant vector field will drive the snake towards spurious edges. As a result, the same Snakes initialization problem remains, although the GVF field does improve the performance of the snake for objects with concavities and prevents the model from leaking through gaps in the boundary edges. Multi-scale versions of GVF are beginning to appear in the literature to ameliorate the noise sensitivity.

3.3 Live Wire

Live wire (also known as Intelligent Scissors) [19] is a semi-automatic segmentation technique which allows user interaction and control over the segmentation process. In this technique, the user initially specifies a seed point on the object boundary. For any subsequent position of the cursor, a globally optimum path 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, 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 'live wire'. If the cursor moves close to the boundary, the live wire snaps to the boundary (assuming the cost functions are set correctly). If the user is satisfied with the boundary segment computed, the user 'deposits' the cursor, the point becomes the new seed point and the recursive process continues.

The mathematical formulation of live wire is as follows [8]. A 2D-scene (i.e. image) C is defined as a pair (C, g) where C is a finite matrix of pixels and g is a function, defined as $g(p) : C \rightarrow [L, H]$, that assigns an intensity value lying in an interval $[L, H]$ to each pixel p in C. Also, a directed graph is associated with C in which the pixels represent the nodes of

the graph and their order represents the directions of the graph arcs.

The boundary of interest in C is a closed, oriented, and connected contour made up of oriented pixel edges. The 'closeness' here means that the contour partitions the space into two disjoint components. 'Orientedness' means that one of the components can be identified as the 'interior' of the boundary and the other as 'exterior'. 'Connectedness' guarantees that the boundary is a single connected curve.

Since there are numerous possible closed, oriented and connected contours, every pixel edge in C is a potential boundary element. A boundary element(*bel* for short) is defined as an ordered pair of four adjacent pixels. The location of $b = bel(p, q)$ is that of the unique edge shared by pixels p and q. To each bel $b(p, q)$, a set of features is assigned which expresses the likelihood of the bel belonging to the boundary of interest in C. The features describe certain properties of the object (the 'interior' of the boundary), of the background (the 'exterior' of the boundary) and of the boundary itself. Then, the feature values are converted to a single cost value which describes the cost of having b as part of the boundary we are seeking. To every closed, oriented, connected contour that can be defined in C, we assign a cost which is simply the sum of the cost of all bels comprising the contour. The aim is to find a closed, oriented, connected contour with minimum total cost starting from the first user inputed bel $b0$. Thus, the problem of finding the best boundary segment (live wire segment) between any two edge points is translated to finding the minimum-cost path between the corresponding two vertices of the graph which can be solve by dynamic programming.

The main issues in live wire are feature selection, converting feature values into cost values, and finding the optimum global path between any two vertices specified in C. There are many ways to select features and define transformation function to assign cost values to them. Typically, features are considered as functions that assign to every bel $b = bel(p, q)$ of a given scene $C = (c, g)$, an integer representing a property value. Most feature selection methods are based on the intensity or gradient of the neighbor pixels.

Feature transform functions(cost functions) can be any functions $(c_j(f_i))$ converting integer feature values (f_i) into the range of $[0, 1]$, which assigns sufficiently low cost values to

bels on the desired object boundaries. Two example cost functions are:

- linear (c_1): c_1 is a linear mapping within an interval $[l_1, h_1]$ of feature values. Values outside this interval are mapped to
- Gaussian c_3: c_3 is a Gaussian function with mean l_3 and standard deviation h_3
- Modified hyperbolic (c_2): only a part of a hyperbolic function with non-negative values is considered

$$c_2(x) = \begin{cases} 1 & \text{for} x \leq l_5 + \frac{a^2}{2} \\ \frac{a^2}{2(x-l_5)} & \text{for} l_5 + \frac{a^2}{2} \leq x \leq h_5 \\ 0 & \text{for} x \leq h_5 \end{cases} . \tag{3.8}$$

here a, l_5, and h_5 are free parameters. Parameter a represent the distance of the focus of the hyperbola from its two asymptotes, l_5 represents the distance of the vertical asymptotes $x = l_5$ from the original, and h_5 is the upper cut-off values for x. The process of feature selection, transforms and their parameters can be facilitated by training, obviating the need for the the users to become knowledgeable about the segmentation methodology. Although, live wire methods are much faster and more reproducible than manual tracing of object boundaries, and provide the user with good control during the segmentation process, the following disadvantages must be noted:

- When the desired object boundary has a relatively weak edge close to an insignificant but strong edge, the live wire snaps to the strong edge rather than the desired weak boundary. To eliminate this problem, Falcão and Udupa [8] developed a technique called 'live wire on-the-fly'. Basically this technique dynamically updates the cost map to filter out the image features which do not have similar edge characteristics to the sample boundary specified by the user. That is, the edge property is assumed to be relatively consistent along the object boundary. Therefore, it doesn't work well for objects with sudden and/or dramatic changes in their boundary properties.
- Another problem of live wire is possibility of jumping live wire from boundary gaps and passing through noisy areas. Even if there are no image features at all between

two points, live wire can still provide a minimal path-a straight line between the two points [20] (although a straight line paths may not be desired).

- Once the user places the cursor when tracing the boundary, the point is collected as a seed point and the trace adds to the extracted object boundary. At this point the user has no further control over the trace other than return to the previous point and remove it. This correction increases segmentation time and user interaction when dealing with a noisy and low contrast object or complex boundary.

- Like the snakes model, live wire is only as good as the accuracy of its cost functions to describe the target boundary. Many images are noisy and the image features vary across the target object. In addition, objects which are complex in shape (contain highly curved regions or protrusions for example) often force the user to deposit many seed points.

- Although far superior to manual tracing, Live wire still requires tracing-like actions to position the cursor. Moving the cursor around an object under the control of a mouse (or other input device) is tedious and therefore fatiguing.

- Live wire is not as amenable to segmenting multiple image slices in a time series (object tracking) or volume image as are Snakes. This is due to the fact that it is fundamentally tracing-based. Snakes, on the other hand, can be modified anywhere at any time by the user.

- Unlike Snakes, live wire is limited in its ability to be controlled by 'intelligent' high-level control algorithms.

3.4 Deformable Templates

Interactive techniques such as Snakes and live wire are based on only small scale, local boundary information and are unable to take advantage of higher-level object shape information that manifests itself at multiple scales and locations with respect to the object. For

example, they cannot make use of an object's width or symmetry. Deformable template models on the other hand, are generally able to take advantage of this prior object information. These models usually use *hand-crafted* global shape parameters to embody prior knowledge of expected shape and/or shape variation of the structures. The use of prior knowledge makes them much less sensitive to noise and boundary gaps than snakes. On the other hand, they are obviously not as general a technique and are typically designed to be completely automatic.

The idea of deformable templates can be traced back to the early work on spring-loaded templates by Fischler and Elshlager [9]. Yuille et al. constructed a deformable face template, consisting of ellipses and curves connected by springs, to detect and describe features of faces, such as the eyes and the mouth. Several deformable templates based on superquadrics [25, 34] have been developed. Superquadrics are defined by a small number of intuitive global shape parameters that can be set such that the superquadric takes on the average shape of a target anatomical structure. The global shape parameters can be coupled to local shape representations, such as spline curves, to allow for a greater range of shape coverage. The global shape parameters efficiently capture the gross shape features of the object, while the local shape parameters reconstruct the fine details.

Several researchers have cast the deformable model-based approach in a Bayesian framework, where the model prescribes a prior distribution on allowable shape variations and where the likelihood function describes how well measurements derived from the image are in accordance with a given geometric state of the model [31, 35, 33, 10]. Although they are can represent large shape variability, the task of choosing an appropriate probabilistic deformation model is very complex and not necessarily intuitive. In active shape models(ASM), introduced by Cootes [6, 15], a statistics-based technique for building deformable shape templates is used. The statistical parameterization provides global shape (and appearance) constraints and allows the model to deform only in ways found by the training set. The shape models represent objects by sets of landmark points which are placed by an expert in the same way on an object boundary in each image of a training set of images. Once the

shapes are aligned and properly annotated, principal component analysis(PCA) is used to generate an average shape(prototype) along with a series of modes of variation. Dutahas proposed a method to automatically align and annotate the training examples. New shapes are generated using the mean shape and a weighted sum of the major modes of variation. Object boundaries are segmented using this point distribution model by examining a region around each model point to calculate the displacement required to move it towards the boundary. These displacements are then used to update the shape parameter weights.

Deformable templates are constructed for a particular object (or small set of related objects) and are specifically designed to be robust, automatic, and exhibit more intelligent segmentation behavior than their semi-automatic counterparts. However, it has been challenging for researchers to build completely reliable systems . As a result, it may be desirable to imbue the semi-automatic techniques with some of the properties of a deformable template but in such a way that the user can easily construct and/or train the model rather than the programmer. Preliminary work on this idea is explored in chapter 5 of this thesis.

Chapter 4

Finite Element Snakes Formulation

4.1 Finite Element Method(FEM)

The Finite Element Method (FEM) is a computer-aided mathematical technique to find approximate numerical solutions of equations where exact solutions are difficult or even impossible to be found. Usually the equations predict the response of the system subjected to external influences.

In the FEM, the domain of the problem is divided (partitioned) into smaller regions (sub-domains) called *elements*. Adjacent elements touch without overlapping and there are no gaps between elements.

The approximate solution of a 1D dynamic equation is a function of $s, x_1(t), x_2(t) \cdots, x_N(t)$ as follows:

$$x(s,t;a) = \sum_{j=1}^{N} B_j(s) x_j(t), \tag{4.1}$$

where $x_1(t), x_2(t), \cdots, x_N(t)$ are unknown parameters, frequently called degrees of freedom (DOF) and must be found. Subsequently, a in x(s,t;a) represents all the parameters $x_1, x_2, \cdots, x_N$. x is a function of parameter s, t (time),as well as $x_1(t), x_2(t), \cdots, x_N(t)$. The functions $B_1, B_2, \cdots, B_N$ are known functions, called basis functions. They are also called trial or coordinate functions.

The FEM is mainly categorized into two approaches[29]:

- the Ritz Variational method(RVM) which is applicable when the equation is a varia-

tional (integral) equation.

- the Method of Weighted Residuals (MWR), which is applicable when the equation is a differential equation. In this thesis we use the MWR.

Given equation $L(x(s;a(t))) = F(s)$, where L is a differential operator and $R(s,t;a)$ is an expression of error, the MWR finds numerical values for $x_1, x_2, \cdots, x_N$ which make $R(s,t;a)$ as close as possible to zero for all values of s in the domain. $R(s,t;a)$ in 1D is defined as follows:

$$R(s;a) = L(x(s;a(t))) - F(s).$$

Applications of MWR produce a set of algebraic equations, the solutions of which are the *best* set of numerical values of x_j.

The Galerkin method is one of the most commonly used methods in the MWR category, which is applicable to a wide range of applications. Applying this method to any boundary-value problem produces N residual algebraic equations for the elements as follows:

$$\int_{a_1}^{a_N} R(s;a(t))B_1(s)dx = 0,$$

$$\int_{a_1}^{a_N} R(s;a(t))B_2(s)dx = 0,$$

$$\vdots$$

$$\int_{a_1}^{a_N} R(s;a(t))B_N(s)dx = 0.$$

where N is the number of degrees of freedom(NDOF). The solution of above equations is the best set of numerical values for a. Depending on the trial functions chosen, we may have different solutions that are very close to each other and to the exact solution (i.e. an acceptable solution for the user). The main steps on solving an equation using Galerkin's method are as follows:

- Choose the number of elements, basis functions and mesh (if the application is 2D or 3D).

- Write Galerkin element residual equations.
- Integrate by parts.
- Substitute the general form of the trial functions (Eq. 4.1) into the system of equations and transform the integrals into an appropriate form for numerical evaluation.
- Apply boundary conditions to the system of equations.
- Solve the system of equations.
- Display the solution and estimate its accuracy.

4.2 Finite Element Snake

In a finite element snake, the parametric domain $0 \leq s \leq L$ is partitioned into finite subdomains$[s_1, s_2], [s_2, s_3], \cdots [s_{N-1}, s_N]$. Consequently, the snake is divided into $N-1$ *elements* if it is open, and N elements if it is closed. Each element e is represented geometrically with shape functions that involve its corresponding shape parameter.

Snakes in a FE formulation are approximated by:

$$\mathbf{v}(s,t) \cong \mathbf{B}(s)\mathbf{u}(t),$$

or in Cartesian coordinates:

$$u_j(t) = u_{x_j}(t)\vec{i} + u_{y_j}(t)\vec{j},$$

$$\mathbf{v}(s,t) = \mathbf{X}(s,t)\vec{i} + \mathbf{Y}(s,t)\vec{j},$$

$$\mathbf{X}(s,t) = \mathbf{B}(s)\mathbf{u}_x(t),$$

$$\mathbf{Y}(s,t) = \mathbf{B}(s)\mathbf{u}_y(t),$$

where $\mathbf{B}(s) = [B_1(s), B_2(s), B_3(s) \ldots B_n(s)]$ are basis functions, and $\mathbf{u} = [u_1(t), u_2(t), u_3(t), \ldots u_n($ ^{T} are unknown variables(degrees of freedom) of the snake contour model. For closed snakes, we have $v(0,t) = v(L,t)$ as a boundary condition.

4.3 Galerkin Finite Element Formulation Of Snakes

Recalling Eq.2.7, the optimum solution of E_{snake} using Euler equations must satisfy the following condition:

$$-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}} = 0. \tag{4.2}$$

Using Galerkin's method to solve the above equation, the residuals for the equation are:

$$R(s,u) = \int_0^L \left[-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}})\right] ds. \tag{4.3}$$

The N Galerkin residual equations are:

$$\int_0^L R(s,\mathbf{u})B_i(s)ds = 0 \quad i = 1 \cdots N. \tag{4.4}$$

Substituting 4.3 into 4.4 we have:

$$\int_0^L \left[-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}})\right] B_i(s)ds = 0, \tag{4.5}$$

or

$$\int_0^L \left[-(\alpha(s)\mathbf{v}_s)_s B_i(s) + (\beta(s)\mathbf{v}_{ss})_{ss} B_i(s) + \frac{\partial E_{ext}}{\partial \mathbf{v}} B_i(s)\right] ds = 0. \tag{4.6}$$

Integration by parts for the first term results in:

$$\int_0^L (\alpha(s)\mathbf{v}_s)_s B_i(s)ds = [-\alpha(s)\mathbf{v}_s B_i(s)]_0^L + \int_0^L \alpha(s)\mathbf{v}_s \frac{\partial B_i}{\partial s}(s)ds, \tag{4.7}$$

and for the second term results in:

$$\begin{aligned} \int_0^L (\beta(s)\mathbf{v}_{ss})_{ss} B_i(s)ds &= [(\beta(s)\mathbf{v}_{ss})_s B_i(s)]_0^L - \int_0^L \beta(s)\mathbf{v}_{ss})_s \tfrac{\partial B_i}{\partial s}(s)ds \\ &= [(\beta(s)\mathbf{v}_{ss})_s B_i(s)]_0^L - [\beta(s)\mathbf{v}_{ss}\tfrac{\partial B_i}{\partial s}(s)]_0^L + \int_0^L \beta(s)\mathbf{v}_{ss}\tfrac{\partial^2 B_i}{\partial s^2}(s)ds. \end{aligned} \tag{4.8}$$

Substituting 4.7 and 4.8 in 4.6 gives

$$\begin{array}{ccccc} \int_0^L \alpha(s)\mathbf{v}_s \frac{\partial B_i}{\partial s}(s)ds & + & \int_0^L \beta(s)\mathbf{v}_{ss}\frac{\partial^2 B_i}{\partial s^2}(s)ds & + & \int_0^L \frac{\partial E_{ext}}{\partial \mathbf{v}} B_i(s)ds \\ [-\alpha(s)\mathbf{v}_s B_i(s) & + & (\beta(s)\mathbf{v}_{ss})_s B_i(s) & - & \beta(s)\mathbf{v}_{ss}\frac{\partial B_i}{\partial s}(s)]_0^L = 0. \end{array} \tag{4.9}$$

Using the Finite Element Method, snakes are approximated as

$$\mathbf{v}(s,t) = \mathbf{B}(s)\mathbf{u}(t), \tag{4.10}$$

or $\mathbf{v} = \sum_{j=1}^{N} B_j u_j(t)$ and therefore,

$$\begin{aligned} \mathbf{v}_s &= \frac{\partial \mathbf{B}(s)}{\partial s}\mathbf{u}(t) \\ &= \left[\frac{\partial B_1(s)}{\partial s}, \frac{\partial B_2}{\partial s}, \cdots, \frac{\partial B_n(s)}{\partial s}\right][u_1(t), u_2(t), u_3(t), \ldots u_n(t)]^T \qquad (4.11) \\ &= \sum_{j=1}^{N} \frac{\partial B_j(s)}{\partial s} u_j(t), \end{aligned}$$

and

$$\begin{aligned} \mathbf{v}_{ss} &= \frac{\partial^2 \mathbf{B}(s)}{\partial s^2}\mathbf{u}(t) \\ &= \left[\frac{\partial^2 B_1(s)}{\partial s^2}, \frac{\partial^2 B_2(s)}{\partial s^2}, \cdots, \frac{\partial^2 B_n(s)}{\partial s^2}\right][u_1(t), u_2(t), u_3(t), \ldots u_n(t)]^T \qquad (4.12) \\ &= \sum_{j=1}^{N} \frac{\partial^2 B_j(s)}{\partial s^2} u_j(t). \end{aligned}$$

Substituting Eqs.4.10, 4.11 and 4.12 into 4.9 gives:

$$\begin{aligned} &\int_0^L \left(\sum_{j=1}^{N} \frac{\partial B_j(s)}{\partial s} u_j(t)\right) \alpha(s) \frac{\partial B_i(s)}{\partial s} ds \; + \; \int_0^L \left(\sum_{j=1}^{N} \frac{\partial^2 B_j(s)}{\partial s^2} u_j(t)\right) \beta(s) \frac{\partial^2 B_i(s)}{\partial s^2} ds \; + \\ &\int_0^L B_i(s) \frac{\partial E_{ext}}{\partial \mathbf{v}} ds \; + \; \left[-B_i(s)\alpha(s)\left(\sum_{j=1}^{N} \frac{\partial B_j(s)}{\partial s} u_j\right) \; + \qquad (4.13) \\ &B_i(s)\beta(s)\left(\sum_{j=1}^{N} \frac{\partial^2 B_j(s)}{\partial s^2} u_j\right)_s \; - \; \frac{\partial B_i(s)}{\partial s}\beta(s)\left(\sum_{j=1}^{N} \frac{\partial^2 B_j(s)}{\partial s^2} u_j\right)\right]_0^L \; = 0. \end{aligned}$$

Eq. 4.13 is the finite element formulation of snakes.

Let

$$\begin{aligned} \mathbf{K}_{i,\alpha} &= \int_0^L \frac{\partial B_j(s)}{\partial s} \alpha(s) \frac{\partial B_i(s)}{\partial s} ds, \\ \mathbf{K}_{i,\beta} &= \int_0^L \frac{\partial^2 B_j(s)}{\partial s^2} \beta(s) \frac{\partial^2 B_i(s)}{\partial s^2} ds, \\ \mathbf{K}_i &= \mathbf{K}_{i,\alpha} + \mathbf{K}_{i,\beta}, \\ \mathbf{F}_i &= \int_0^L B_i(s) \frac{\partial E_{ext}}{\partial \mathbf{v}} ds, \qquad (4.14) \end{aligned}$$

$$\begin{aligned} \mathbf{P}_i = &\left[-B_i(s)\alpha(s)\left(\sum_{j=1}^{N} \frac{\partial B_j(s)}{\partial s} u_j(t)\right) + \right. \\ &\left. B_i(s)\left(\beta(s) \sum_{j=1}^{N} \frac{\partial^2 B_j(s)}{\partial s^2} u_j(t)\right)_s - \frac{\partial B_i(s)}{\partial s}\beta(s)\left(\sum_{j=1}^{N} \frac{\partial^2 B_j(s)}{\partial s^2} u_j(t)\right)\right]_0^L, \end{aligned}$$

where $i = 1 \cdots N$ and $j = 1 \cdots N$. Then Eq. 4.13 becomes:

$$\mathbf{K}_i \mathbf{u} + \mathbf{P}_i + \mathbf{F}_i = 0.$$

where $\mathbf{K}_i$ is called the element stiffness matrix, $\mathbf{P}_i$ is called the boundary force, and $\mathbf{F}_i$ is called the nodal force.

To calculate an *element force*($\mathbf{F}_i$), Gaussian-quadrature is used to approximate the value of the integral as follows:

$$\mathbf{F}_i = \ell \sum_j \rho_j B_h(\xi_j)^T(-\nabla P_I^x(v(\xi_j)), \tag{4.15}$$

where ξ_j and ρ_j are the jth Gaussian integration point and its corresponding weighting coefficient, respectively.

To calculate the *element boundary force*($\mathbf{P}_i$), assuming $\alpha(s)$ and $\beta(s)$ are constant for all values of s, we have:

$$\begin{aligned}\mathbf{P}_i &= -B_i(L)\alpha(\textstyle\sum_{j=1}^N \frac{\partial B_j(L)}{\partial s}u_j(t)) + B_i(L)(\beta\sum_{j=1}^N \frac{\partial^2 B_j(L)}{\partial s^2}u_j(t))_s - \frac{\partial B_i(L)}{\partial s}\beta(\sum_{j=1}^N \frac{\partial^2 B_j(L)}{\partial s^2}u_j(t)) \\ &- \left(-B_i(0)\alpha(\textstyle\sum_{j=1}^N \frac{\partial B_j(0)}{\partial s}u_j(t)) + B_i(0)(\beta\sum_{j=1}^N \frac{\partial^2 B_j(0)}{\partial s^2}u_j(t))_s - \frac{\partial B_i(0)}{\partial s}\beta(\sum_{j=1}^N \frac{\partial^2 B_j(0)}{\partial s^2}u_j(t))\right).\end{aligned}$$

If $v(s;u(t))$ interpolates nodes $u_j(x_j, y_j)$ i.e $v(s_j, t) = u_j(t)$ the following condition is true for all basis functions.

$$B_i(s = s_j) = \begin{cases} 1 & if\ i = j \\ 0 & otherwise \end{cases}.$$

So

$$B_i(s = L = s_N) = \begin{cases} 1 & if\ i = N \\ 0 & otherwise \end{cases}, \tag{4.16}$$

and

$$B_i(s = 0 = s_1) = \begin{cases} 1 & if\ i = 1 \\ 0 & otherwise \end{cases}. \tag{4.17}$$

Substituting Eq.4.16 and Eq.4.17 into the boundary force equation ($\mathbf{P}_i$), we have

$$\begin{aligned}\mathbf{P}_N &= \alpha\tfrac{\partial B_N(L)}{\partial s}u_N(t) + B_i(L)\beta(\tfrac{\partial^2 B_N(L)}{\partial s^2}u_N(t))_s - \tfrac{\partial B_N(L)}{\partial s}\beta(\tfrac{\partial^2 B_N(L)}{\partial s^2}u_N(t)), \\ \mathbf{P}_1 &= -\left(-B_1(0)\alpha(\tfrac{\partial B_1(0)}{\partial s}u_1(t)) + B_1(0)(\beta\tfrac{\partial^2 B_1(0)}{\partial s^2}u_1(t))_s - \tfrac{\partial B_1(0)}{\partial s}\beta(\tfrac{\partial^2 B_1(0)}{\partial s^2}u_1(t))\right).\end{aligned}$$

Applying the condition $v(0,t) = v(L,t)$ for closed interpolating snakes results in $\mathbf{P} = 0$, but for open snakes $\mathbf{P}_1$ and $\mathbf{P}_N$ must be calculated.

4.3.1 Formulation Along the X And Y Directions

For convenience of manipulation, the formulations can also be set up based on deflections along X and Y directions of Cartesian coordinates. Considering that $\mathbf{v}$ can be expressed in Cartesian coordinates as $\mathbf{v} = \mathbf{X}\vec{i} + \mathbf{Y}\vec{j}$, we can substitute this Cartesian equation into the internal energy equation. So, we have

$$E_{internal} = \left(\alpha(s)\mathbf{X}_s^2 + \beta(s)\mathbf{X}_{ss}^2 + \alpha(s)\mathbf{Y}_s^2 + \beta(s)\mathbf{Y}_{ss}^2\right)/2, \tag{4.18}$$

and the energy of the snake becomes

$$\begin{aligned} E_{snake}^* &= \int_0^L \left[\alpha(s)\mathbf{X}_s^2 + \beta(s)\mathbf{X}_{ss}^2 + \alpha(s)\mathbf{Y}_s^2 + \beta(s)\mathbf{Y}_{ss}^2 + E_{ext}\right] ds \\ &= \int_0^L F(s, v, v_s, v_{ss}). \end{aligned} \tag{4.19}$$

Using the Calculus of Variations, minimizing the above integral corresponds to solving the following Euler equations:

$$F'_X - (F'_{X_s})_s + (F'_{X_{ss}})_{ss} = 0, \tag{4.20}$$

$$F'_Y - (F'_{Y_s})_s + (F'_{Y_{ss}})_{ss} = 0, \tag{4.21}$$

i.e.

$$-(\alpha(s)X_s)_s + (\beta(s)X_{ss})_{ss} + (E'_{ext})_X = 0, \tag{4.22}$$

and

$$-(\alpha(s)Y_s)_s + (\beta(s)Y_{ss})_{ss} + (E'_{ext})_Y = 0. \tag{4.23}$$

Eq.4.13 can be obtained separately for X and Y by following exactly the same steps that have been applied to Eq4.2.

Additionally, X and Y can share the same shape functions. The finite element formulation along X and Y directions can be given by the following two equations, respectively:

$$\mathbf{KX} + \mathbf{F}_X = 0, \tag{4.24}$$

and

$$\mathbf{KY} + \mathbf{F}_Y = 0. \tag{4.25}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the X and Y components of $\mathbf{u}$, and $\mathbf{F}_X$ and $\mathbf{F}_Y$ are the forces due to the external energy. The solutions to 4.24 and 4.25 will update X and Y coordinates of nodal points on the finite element mesh.

Similarly, $\mathbf{F}_X$ and $\mathbf{F}_Y$ can be calculated from the following:

$$\mathbf{F}_X^{e_i} = \int_0^l B_i(s)(E_{ext})'_x ds, \tag{4.26}$$

$$\mathbf{F}_Y^{e_i} = \int_0^l B_i(s)(E_{ext})'_y ds. \tag{4.27}$$

Furthermore, end point forces $\mathbf{P}_0$ and $\mathbf{P}_1$ along X and Y direction must be calculated for open snakes.

4.3.2 Global Assembly

The global $\mathbf{K}$,$\mathbf{P}$ and $\mathbf{F}$ matrices are obtained by assembling $\mathbf{K}_i^e$,$\mathbf{P}_i^e$,$\mathbf{F}_i^e$ over all the elements. Assembling here means summing up the items corresponding to shared nodes(degrees of freedom) by adjacent elements. That is $\mathbf{K} = \sum \mathbf{K}_i^e$, $\mathbf{P} = \sum \mathbf{P}_i^e$ and $\mathbf{F} = \sum \mathbf{F}_i^e$ The global FE formulation can be written as follows:

$$\mathbf{Ku} + \mathbf{P} + \mathbf{F} = 0. \tag{4.28}$$

As described in the previous section, $\mathbf{P}_0$ and $\mathbf{P}_1$ are non-zero for open snakes. To simplify equation4.28, we assume the non-zero items in vector $\mathbf{P} = [P_0, 0, 0, \cdots, P_n]$ have been added to the external force vector $\mathbf{F}$.

4.4 Snakes Deformation

To solve the discrete form of the Lagrange Eq. 2.10, we calculate time derivatives of u by backward finite differences as follows:

$$\ddot{u} = \frac{u^{t+\triangle t} - 2u^t + u^{t-\triangle t}}{\triangle t^2}, \tag{4.29}$$

$$\dot{u} = \frac{u^{t+\triangle t} - u^t}{\triangle t}. \tag{4.30}$$

Where the superscripts in Eq.4.29 and Eq.4.30 show the time quantity evaluated and $\triangle t$ is the time step. Substituting them in the Eq. 2.10, yields the update formula

$$\mathbf{a}u^{t+\triangle t} = \mathbf{b}u^t + \mathbf{c}u^{t-\triangle t} + \mathbf{g}, \tag{4.31}$$

where

$$\mathbf{a} = \frac{\mathbf{M}}{\triangle t^2} + \frac{\mathbf{C}}{\triangle t} + \mathbf{K},$$

$$\mathbf{b} = \frac{2\mathbf{M}}{\triangle t^2} + \frac{\mathbf{C}}{\triangle t},$$

and

$$\mathbf{c} = -\frac{\mathbf{M}}{\triangle t^2}, \ \mathbf{g} = \mathbf{F}.$$

Since $\mathbf{a}$ has coefficients of $u^{t-\triangle t}$ and $u^{t+\triangle t}$ and u^t, it is a pentadiagonal banded matrix and can be saved in skyline storage to save memory and the computational cost of solving Eq4.31. With negligible mass, we have the following form:

$$\mathbf{K}\mathbf{X}_t + \mathbf{F}_{Xt-1} = -\gamma(\mathbf{X}_t - \mathbf{X}_{t-1}), \tag{4.32}$$

$$\mathbf{K}\mathbf{Y}_t + \mathbf{F}_{\mathbf{Y}t-1} = -\gamma(\mathbf{Y}_t - \mathbf{Y}_{t-1}). \tag{4.33}$$

The matrix $\mathbf{K}$ can be factorized uniquely into the form $\mathbf{K} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{D}$ is a diagonal matrix. The solution $u^{t+\triangle t}$ to Eq. 4.31 is obtained by first solving $\mathbf{L}s = \mathbf{b}u^t + \mathbf{c}u^{t-\triangle t} + \mathbf{g}$ with forward substitution, then $\mathbf{L}^T u = \mathbf{D}^{-1}s$ with backward substitution. Since $\mathbf{K}$ is constant, only a single factorization is necessary . Therefore, at each time step, only the forward/backward substitutions are performed to integrate the snake forward through time. The following sections describe B-snakes, and Hermite snakes which are variations of the FE Snake deformation described above.

4.4.1 Hermite Snake

In the case of Hermitian snakes, $x(s)$ $(0 \leq s \leq l$, where l is the element parametric length) is approximated with cubic polynomial functions, parameterized by position x and slope θ at the end points $s = 0$ and $s = l$ of an element. We can show that $x^e(s) = Bu^{e_i}$, where $u^{e_i} = [x_i, \Theta_i, x_{i+1}, \Theta_{i+1}]$ are the shape parameters of element e_i and $B = S \times H$ are the Hermitian shape functions, with $S = [1, s, s^2, s^3]$ and the following*Hermitian shape matrix* (see appendix A)

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/l^2 & -2/l & 3/l^2 & -1/l \\ 2/l^3 & 1/l^2 & -2/l^3 & 1/l^2 \end{bmatrix}. \tag{4.34}$$

Considering the tension function $\alpha(s)$ and rigidity function $\beta(s)$ as constant values, the stiffness matrices associated with the tension and rigidity components for element e_i are, respectively

$$\mathbf{K}_{\alpha}^{e_i} = \frac{\alpha_i}{30l} \begin{bmatrix} 36 & 3l & -36 & 3l \\ 3l & 4l^2 & -3l & -l^2 \\ -36 & -3l & 36 & -3l \\ 3l & -l^2 & -3l & 4l^2 \end{bmatrix}, \tag{4.35}$$

$$\mathbf{K}_{\beta}^{e_i} = \frac{\beta_i}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix}. \tag{4.36}$$

An analytic form of the external forces generally is not available. Therefore, Gauss-Legendre quadrature may be employed to approximate the value of the integral for the element external force vector $\mathbf{F}^e$. For element e_i we have

$$\begin{aligned} \mathbf{F}_x^{e_i} &= \int_0^l N_h q_x(v(s))\, ds \\ &= l \sum_j \rho_j N_h(\xi_j) q_x(v(\xi_j)), \end{aligned} \tag{4.37}$$

where the subscript x indicates the association with coordinate function $x(s)$, and ξ_j and ρ_j are the jth Gaussian integration point and its corresponding weighting coefficient, respectively. The force vector $\mathbf{F}_y^{e_i}$ is derived in a similar way.

To make the global matrix assembly process identical for all shape functions, Liang([18]) has introduced *a*ssembling matrices. Suppose that there is a snake with n elements and N nodes ($N = n$ for closed snake and $N = n + 1$ for open snakes). For the ith element e_i of the snake ($0 \leq i \leq n-1$), the assembling matrices are $\mathbf{G}_{\alpha}^{e_i} = \mathbf{G}_{\beta}^{e_i} = \mathbf{G}_{\mathbf{F}}^{e_i} = \mathbf{G}^{e_i}$, where

$$(\mathbf{G}^{e_i})_{jk} = \begin{cases} 1 & \text{if } (j + di) \bmod (dN) = k \\ 0 & \text{otherwise} \end{cases} \tag{4.38}$$

are $(2 \cdot d) \times (d \cdot N)$ matrices, with d the number of degrees of freedom of each node in an element (here $d = 2$). Hence, K_{α}, K_{β} and F may be assembled as follows:

$$\begin{aligned} \mathbf{K}_{\alpha} &= \textstyle\sum_{i=0}^{n-1} (\mathbf{G}_{\alpha}^{e_i}) \mathbf{K}_{\alpha}^{e_i} (\mathbf{G}_{\alpha}^{e_i}), \\ \mathbf{K}_{\beta} &= \textstyle\sum_{i=0}^{n-1} (\mathbf{G}_{\beta}^{e_i}) \mathbf{K}_{\beta}^{e_i} (\mathbf{G}_{\beta}^{e_i}), \\ \mathbf{F} &= \textstyle\sum_{i=0}^{n-1} (\mathbf{G}_{\mathbf{F}}^{e_i}) \mathbf{F}^{e_i}. \end{aligned}$$

4.5 B-Spline Shape Functions

For B-spline shape functions, $x(s)$ coordinate function of $\mathbf{v}(s)$ is constructed as a weighted sum of N_B basis functions $B_n(s)$, $n = 0, ..., N_B - 1$ as follows: $x(s) = \mathbf{B}(s)\mathbf{Q}^x$, where $\mathbf{B}(s) = [B_0(s), ..., B_{N_B-1}(s)]$, $\mathbf{Q}^x = \left[P_0^x, ..., P_{N_B-1}^x\right]$ and P_i^x are weights (control points) that are applied to the respective basis functions $B_n(s)$. The number of control points determines the number of degree of freedom available in the fitting process and an optimal choice for this is needed to maintain both smoothness and closeness to the target object.

Similar to Hermite Snakes, the nodal variables (snake shape parameters), the shape matrix, and the assembling matrix associated with an element must be determined.

When all B-Spline elements have equal length, the knot multiplicities at the breakpoints are $m_0, ..., m_L$ (L is the number of elements and the total number of knots $N_B = \sum_{i=0}^{L} m_i$), the knot values k_i are determined by $k_i = l$, such that $0 \leq (i - \sum_{j=0}^{l} m_j) < m_{l+1}$. Furthermore, the nth polynomial $B_{n,d}^{\sigma}$ in span σ can be computed as follows:

$$B_{n,1}^{\sigma}(s) = \begin{cases} 1 & \text{if } k_n \leq \sigma < k_{n+1} \\ 0 & \text{otherwise} \end{cases}, \tag{4.39}$$

$$B^{\sigma}_{n,d}(s) = \frac{(s+\sigma-k_n)B^{\sigma}_{n,d-1}(s)}{k_{n+d-1}-k_n} + \frac{(k_{n+d}-s-\sigma)B^{\sigma}_{n+1,d-1}(s)}{k_{n+d}-k_{n+1}}. \tag{4.40}$$

The indices of basis functions for span σ can be calculated as follows:

$$I = [b_\sigma, (b_\sigma+1) \bmod N_B, ..., (b_\sigma+d-1) \bmod N_B].$$

where $b_\sigma = [(\sum_{i=0}^{\sigma} m_i) - d] \bmod N_B$. Now, the shape matrix for span σ can be constructed by collecting the coefficients of each of the d basis functions $B^{\sigma}_{n,d}$ as its columns. In this thesis, our implementation is based on cubic uniform B-Splines. For a *uniform* cubic B-spline (where spacing between knot values is constant) the shape matrix for all spans is the same as follows: (see appendix A))

$$\mathbf{H} = \begin{bmatrix} 1/6 & 2/3 & 1/6 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1/2 & -1 & 1/2 & 0 \\ -1/6 & 1/2 & -1/2 & 1/6 \end{bmatrix}, \tag{4.41}$$

and the element stiffness matrices for element e_i are

$$\mathbf{K}^{e_i}_{\alpha} = \alpha_i \begin{bmatrix} 0.0500 & 0.0583 & -0.1000 & -0.0083 \\ 0.0583 & 0.2833 & -0.2417 & -0.1000 \\ -0.1000 & -0.2417 & 0.2833 & 0.0583 \\ -0.0083 & -0.1000 & 0.0583 & 0.0500 \end{bmatrix}, \tag{4.42}$$

$$\mathbf{K}^{e_i}_{\beta} = \beta_i \begin{bmatrix} 0.3333 & -0.5000 & 0 & 0.1667 \\ -0.5000 & 1.0000 & -0.5000 & 0 \\ 0 & -0.5000 & 1.0000 & -0.5000 \\ 0.1667 & 0 & -0.5000 & 0.3333 \end{bmatrix}. \tag{4.43}$$

The assembling matrix $\mathbf{G}^{e_i}$ can be defined as

$$(\mathbf{G}^{e_i})_{jk} = \begin{cases} 1 & \text{if } (j+b_\sigma) \bmod N_B = k \\ 0 & \text{otherwise} \end{cases}. \tag{4.44}$$

In a similar fashion as above, other kinds of snakes with different shape functions can be constructed; for instance, NURBS shape functions[32], Catmull-Rom shape functions, Bézier shape functions and Fourier shape functions.

Chapter 5

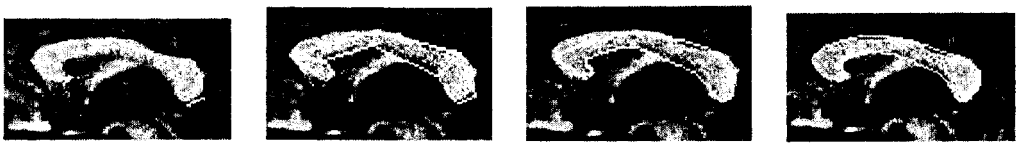
User-Defined B-Spline Template Snakes

In this chapter a new Snakes model is described that overcomes many of the limitations of the traditional finite difference snake. This new deformable model combines a novel user initialization process with a finite element B-spline snake to create a powerful semi-automatic segmentation method. The FE B-spline snake is initialized such that it is very similar in shape and very close to the boundary of the target object. The model has a structure imposed on it - it 'knows' what parts it has and where these parts are located with respect to the object. This 'template' snake is built on top of the general FE Snakes model, hence there is no need to use predefined, restrictive shape representations like superquadrics. Furthermore, B-spline snakes have many desirable properties. They are a compact, explicitly parameterized model with a control polygon which can be used for global deformation control and customized deformation handles. This chapter also describes the computation of the external image forces and how the known initial position and shape of the snake can be used to design object-specific image forces.

5.1 Initialization Process

The user draws, with a mouse or pen input device, cross-sectional lines (or individual points) on the target object. A point is clicked on one side of the object boundary and a line is stretched and rotated interactively to a point on the opposite boundary. For end cap regions

of objects, the user draws lines approximately tangent to the region.



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

Figure 5.1: Example of initialization process. In (a) the user enters lines, starting from the left side of the corpus callosum and proceeding to the right. In (b) the B-spline control polygon is shown, (c) initial B-spline snake, (d) segmentation result.

Lines are drawn in prescribed critical locations, such as landmark points, and in a prescribed order for a particular object. For example, to segment the corpus callosum (CC) from a mid-sagittal MR brain image, the user starts at the end cap region near the rostrum (extreme left). A small line, tangent to the end cap region, is drawn (Figure 5.1(a)). The user then identifies the genu (highly curved region on the left) and draws a line across the CC in this region. The user then draws two lines that cross the CC, where the first line roughly divides the CC into half and the next line demarcates the splenium region (the circular-shaped right end cap region). The exact positions of these lines are not important. The splenium is then identified and a line is drawn tangent to the splenium end cap region.

This process, once learned, is fast and intuitive. Drawing cross-sectional lines is simple and less tedious than tracing or moving a cursor around an object. Figure 5.1(b) shows the resulting control polygon, Figure 5.1(c) shows the initial B-spline curve, while Figure 5.1(d) shows the resulting segmentation. Once the prescribed lines are drawn, the algorithm uses them to automatically construct a control polygon, and displays the resulting B-Spline curve.

Using this simple but effective process, the user recognizes critical points and regions in a specified order, and transfers this knowledge to model. By drawing lines *across* the object, important information pertaining to the global shape of the object, such as width and symmetry, is imparted to the model. The template snake is initialized very close to the object boundary and is very similar is shape. Furthermore, it is 'aware' of it's position

with respect to the object. The snake is parameterized using minimum number of model degrees of freedom necessary and these degrees of freedom are placed in optimal positions around the object, based on the critical points and features recognized by the user via the input lines. Thus, the model is more like a deformable template than a local snake model - it is less sensitive to noise and more amenable to propagation to subsequent image slices in a volume image or time series. Unlike a traditional deformable template model however, it is constructed and positioned by the user rather than preconstructed and automatically initialized by the segmentation system.

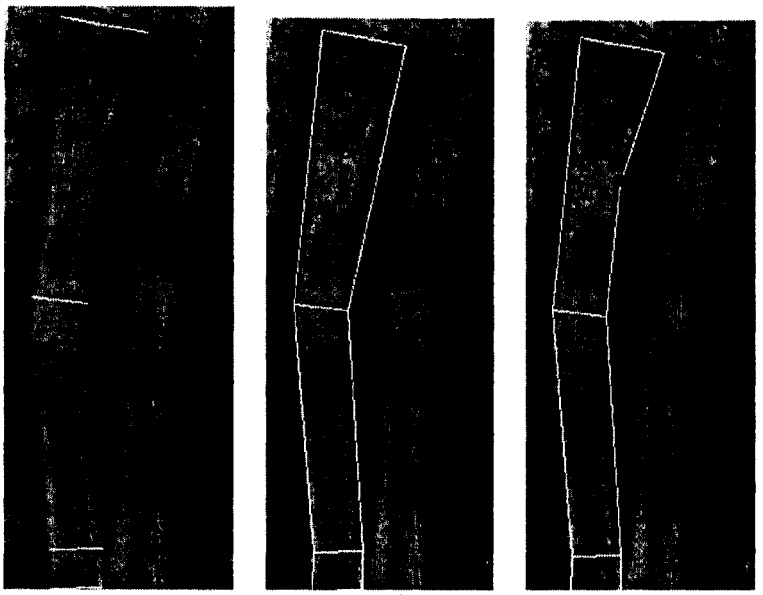
5.1.1 Editing the control polygon

The control polygon can be shown simultaneously as the user is drawing the cross sectional lines. The user may 'click' on control polygon edges or control points, and add new lines or control points if desired. The B-spline control polygon is then updated, incorporating this new information and the new B-spline curve displayed. These new lines and points may be added during initialization or while the snake is running.

When adding a new control point, the control polygon edge corresponding to the control point is removed and two new edges are created by connecting each of the original end-points to the new control point as shown in Figure 5.2. To simplify the interaction and minimize the number of mouse clicks, the user need not explicitly select the edges needed to be divided. Instead, the user may press and hold a designated key on the keyboard(the control key for example) while clicking on the desired point. The program automatically finds the closest edge and subdivides it.

5.1.2 On-the-fly Segmentation

It may be advantageous in many segmentation scenarios to run the snake algorithm as the user enters each line. This 'on-the-fly' or 'as-you-go' segmentation mode (5.3) allows the user to see the result of his/her actions in real-time (principle number 6 and 7 from Section) and hence allow immediate modification/correction (by adding or deleting lines or control



(a) (b) (c)

Figure 5.2: Example of adding a new point. In (a) the user enters lines. In (b) the control polygon and P0 (the user added point) are shown. In (c),the corresponding line is found and divided into two, where the point is an end point for the two lines

points, moving existing control points, or applying traditional spring forces to points on the snake). This segmentation mode provides the user with more complete control throughout the entire segmentation process (principle number 5), thereby ensuring an accurate result.

5.2 Object-Specific External Image Forces

Since the initialized template snake is very similar in shape to and very close to the boundary of the target object, each snake element roughly corresponds to a specific object boundary segment. This knowledge can be used to construct object-specific external image forces. The following sections describe several of these forces.

5.2.1 Snake-Point Control

For each snake element, the number of points in the element, which are termed *snake points*, at which to compute image forces can be specified. For example, the number of snake points

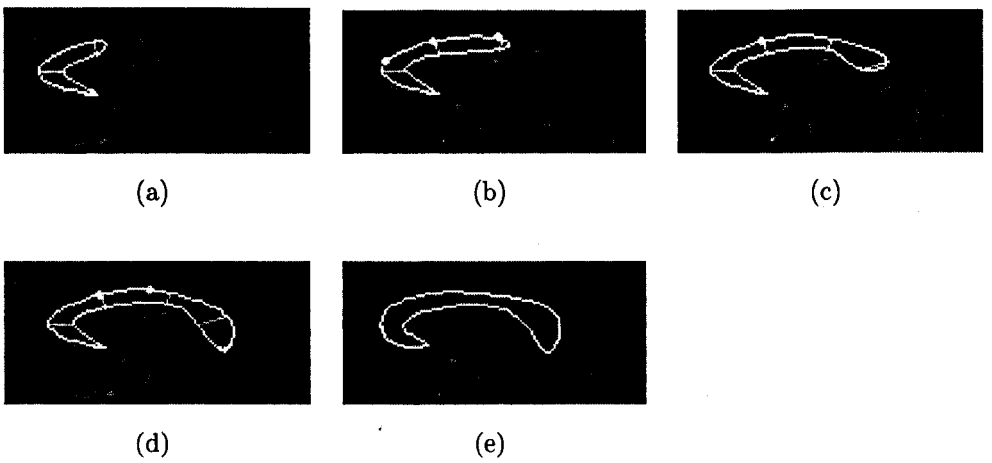


Figure 5.3: Example of on-the-fly segmentation. Partially segmented object after drawing(a) three lines (b) four lines (c) five lines (c) six lines. (d) Final segmentation result.

can be matched to the image resolution so that there is roughly one snake point for each pixel along the element. The forces computed at a point on the B-spline snake are then distributed to the corresponding control points according to equation 4.15. This feature makes the snake less sensitive to noise or spurious image edges. Furthermore, if a snake element is known to correspond to a boundary region with strong image edges, the number of snake points can be reduced.

5.2.2 Knowledge-based Image Feature Search

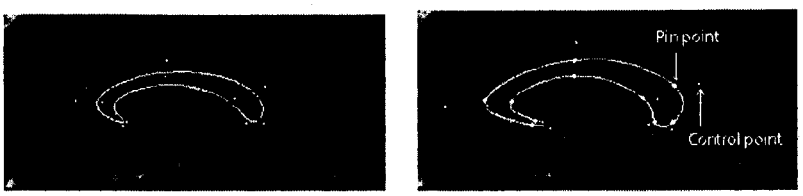
For each snake element, the initialization process provides knowledge of approximately how far away the object boundary is, that it is roughly locally parallel to the element, and what are the neighboring objects. Therefore, at each snake point, a search along a direction normal to the snake point can be performed for a pre-specified distance.

Figure 5.4: Example of searching along the normal direction for 3 points on an snake element.

Search criteria can be either the strongest edges with the correct direction (orthogonal to the snake point normal) or having edge strength in a specific range. If a match is found, a spring force is applied to attract the snake point to the matched edge point. If no matching edge is found at a snake point, this point does not contribute to the image forces. Furthermore, if the image feature information is known to be weak or missing along a particular snake element or snake region, the entire element can be deactivated.

5.2.3 User-input Derived Forces

The user specified input lines provide control points that can be an effective means to bootstrap the snake segmentation process. These points are known to be on or at least close to the object boundary. This knowledge is utilized by constructing a scheduled or phased snake fitting algorithm. In the first phase, for each boundary point derived from the input control points (termed 'pin' points), we find the closet snake point ($s = 0$ in each segment is a good approximation, where s is the parametric sub-domain of the B-spline element) and attach them with a zero rest length spring. After a few iterations, the spring forces bring the entire snake closer to the object boundaries. The result is that the snake even more closely matches the shape of the target object (see Fig 5.5). In the second and subsequent phases, all remaining snake points are activated and forces are computed as described in the previous section. Some or all of the 'pin' points can be deactivated after phase one. This helps reduce the sensitivity to the positioning of the user specified input lines. If these lines are off by a couple of pixels (i.e. the end points are not exactly on the object boundary), the snake is still close enough to the boundary to successfully segment it. In future work, the use of a small zoom window or nonlinear magnification (known as a 'fisheye' zoom) around a small region centered at the cursor will be explored to determine if these techniques reduce user fatigue and hence increase efficiency.



(a) (b)

Figure 5.5: Example of pin spring forces.(a) shows the initial curve and control points (used as pin points) at the beginning of phase one. (b) shows the curve at the end of phase one.

5.2.4 Local Intensity Statistics

As an alternative to the phase II forces described in the previous section, an inflation force based on local intensity statistics can be applied to each snake point. Local intensity statistics can be computed along rectangular regions centered around (and aligned with) the user specified lines. Since these lines are used to construct B-spline control points, each control point data structure stores the local intensity statistics from its associated user line. Each snake point can then use the statistics from its nearest control point to compute a local object intensity model. Custom image-based inflation forces can then be designed for each snake point, rather than using global intensity statistics for all snake points. The inflation force is defined as $\mathbf{f}_{infl}(s) = wF(I(\mathbf{v}(s))\mathbf{n}(s)$ where $\mathbf{n}(s)$ is the unit normal vector at a snake point, and w is the amplitude of this force. The local image intensity statistics are incorporated into the inflation force as follows [16]:

$$F(I(x,y)) = \left\{ \begin{array}{ll} +1 & \text{if } |I(x,y) - \mu| \leq k\sigma \\ -1 & \text{otherwise,} \end{array} \right\}. \tag{5.1}$$

where μ is the mean local image intensity, σ the local standard deviation and k is a user defined constant. Fig.5.6 is an example of segmentation using local statistic intensity.

5.3 Object-Specific Global Shape Constraints

The B-spline control polygon is a coarse approximation of the curve and hence a coarse approximation of the target object boundary (Fig. 5.1b). It is therefore a convenient frame

(a) (b) (c)

Figure 5.6: Example of segmentation using local intensity statistics around user specified lines. (a)User-input lines (b)Areas where intensity statistics are collected. After this, for each snake point the intensity statistics from its closest area will be used. (c)Final Result.

upon which to build global model deformation control. With the push of a button, the control points of the control polygon can be connected with springs (including control points on opposite sides of the polygon).

The spring force between two terminal nodes with positions x_i and x_j is defined as follows:[11]:

$$f_{ij}^{hook} = -k_s \left(\|x_i - x_j\| - r_{ij} \right) \frac{x_i - x_j}{\|x_i - x_j\|},$$

where k_s is Hook's spring constrant.

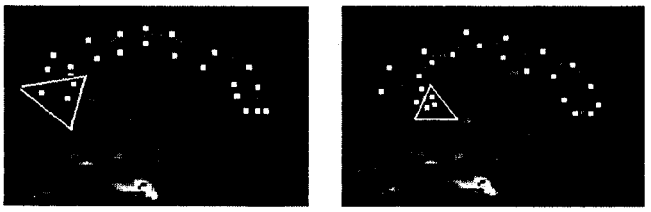
These spring constraint forces can be included on the right hand side of equation (2.10). The control polygon acts as a spring-mass lattice, constraining the global shape or symmetry of the snake (Fig. 5.7).

(a) (b)

Figure 5.7: (a) A snake with global constraint. (b) effect of a user-click on the snake: because of the global constraint the whole snake is pulled toward the user clicked point.

As an example, since we know that the rostrum region of the corpus callosum is consistently triangular in shape, we can connect the control points in this region with springs

fig.(5.8). The forces exerted by the springs will constrain the snake to remain triangular in this region and thus help ensure the tip of the B-spline template snake is attracted to the rostrum tip of the corpus callosum boundary, when the snake is pulled by image forces or by user input forces.



(a) (b)

Figure 5.8: Example of global shape constraint. The three control points within the triangle are connected together with springs, forcing this region of the snake to remain triangular in shape. The snake is initialized incorrectly in (a) but manages to move towards the correct position in (b) so that the rostrum tip of the snake ends up at the rostrum tip of the data.

These shape constraints are invaluable for processing a number of image slices in a time series (i.e. tracking the motion of an object) or in an image volume. A common method of segmenting multiple image slices is to propagate the fitted snake of image n to image $n+1$ (or $n-1$). The fitted snake of the previous slice acts as the initial snake of the next or neighboring image. The idea is for the user to enter cross sectional lines in the first image and segment the target object with the template snake in a normal fashion. Some user editing may be required if the image is very noisy or there are neighboring structures with similar image features. Once the user is satisfied, the global shape constraint springs are activated, and their rest lengths computed. The stiffness values of these springs are currently empirically determined. The constrained snake is then used as the initial snake in the next image. The shape constraints prevent the propagated snake from latching on to neighboring edges, a common problem with a propagated snake that is not constrained, resulting in tedious user editing.

Currently, all control polygon edges and control point pairs on opposite sides of the snake are connected with springs. However, many useful user-definable spring constraint

ırrangements are possible for different objects.

5.4 Intelligent Fitting Algorithms

The initialization process transfers knowledge of object shape and landmark positions to the emplate snake. This information, combined with the properties of a B-spline curve, provides he means to design more intelligent fitting algorithms. Intelligent fitting algorithms can ake advantage of object shape and appearance knowledge, resulting in more robust (i.e. nsensitivity to noise, edge gaps etc.) segmentations [22]. These algorithms also increase the legree of automation and, consequently, reduce the burden on the user. For example, the ıumber of input lines can be reduced for some objects. The tradeoff is algorithm complexity.

An example of an intelligent fitting algorithm that works very well with the B-spline emplate snake is a 'ziplock' algorithm [26] Because of the properties of a B-spline curve, he initial template snake is very close to the target object boundary near the end points ›f the user input lines (i.e. near the 'pin' points). The initial snake is also nearly parallel o the boundary at these locations. At other parts of the initial snake, the snake points nay be quite a distance from the object boundary, especially if the object is highly curved ınd/or a reduced number of input lines is used. Therefore, the external image forces can ›e activated for snake points in a phased manner. Initially, the external image forces are turned on' only for the snake points near the pin points. The internal forces are activated or the entire snake so that as the snake points near the pin points are pulled towards the ›oundary, the neighboring snake points are pulled along by the internal forces. In the next ›hase, external image forces are activated for these neighboring snake points, which are now lose to the target boundary. These snake points pull their neighbors towards the boundary 'ia the internal forces and then in the next phase, more snake point external image forces ıre activated, and so on. The effect is analogous to doing up a zipper on a coat. Fig. 5.9 hows the effect of applying ziplock on a CC, initialized with only four lines. As Fig. 5.9(d) ınd Fig. 5.9(i) show, while the non-intelligent algorithm fails, the ziplock version segments he image properly.

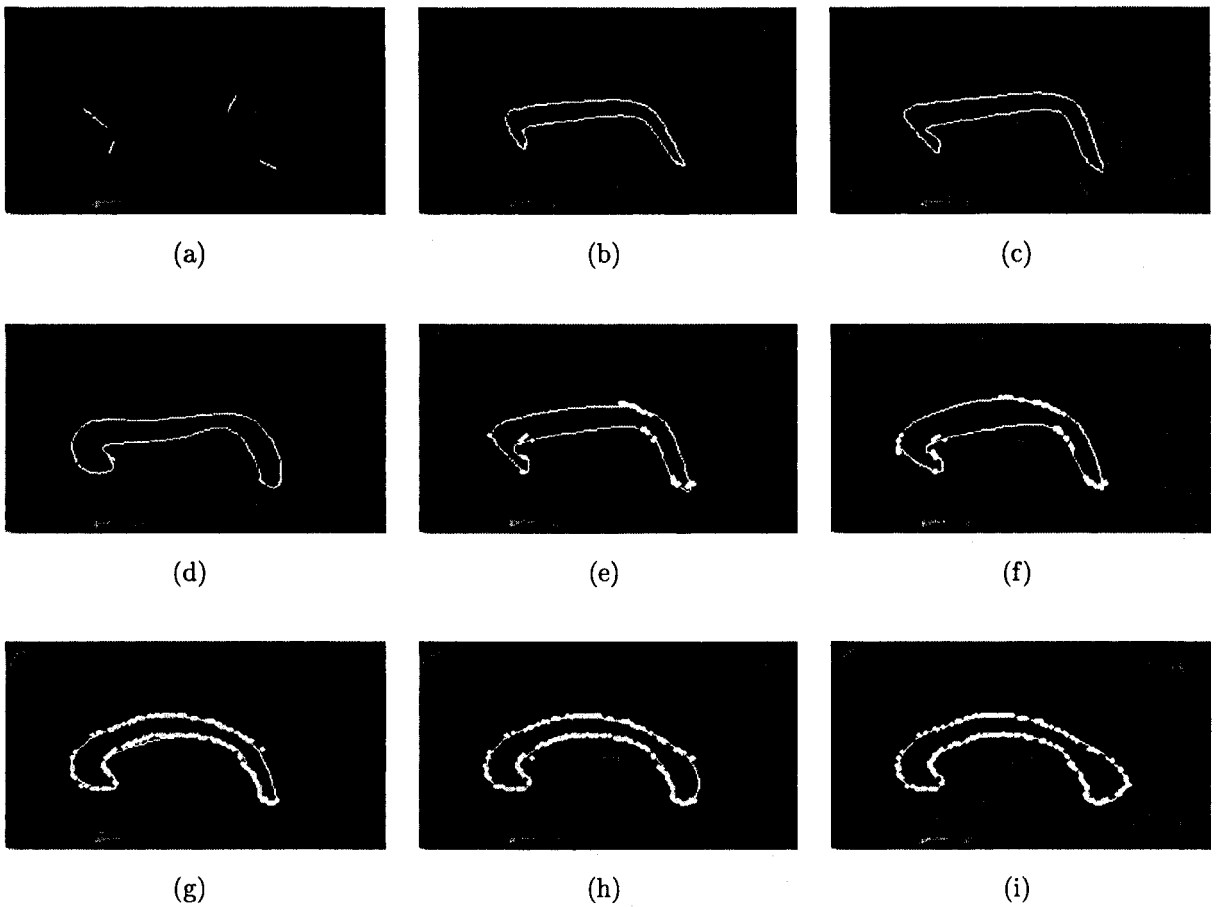


Figure 5.9: Example of intelligent fitting. (a) user-input lines, (b) initial B-spline snake, (c) snake after phase I, (d) final result without using ziplock in phase II, (e-i) using ziplock in phase II -circles show image forces-: (e) after 100 iterations, (f) 200 iterations, (g) 300 iterations, (h) 400 iterations, (i) final result.

Other algorithms may activate external image forces for snake points in a more elaborate manner, proceeding from object boundary regions that exhibit strong edges to regions with less strong edges etc. [22]. In regions where there is a high probability of little if any image edge evidence, noise, or neighbor object interference, the internal forces can be increased and the external image forces deactivated.

Chapter 6

Results

We have applied our B-Spline template snake to several 2D images in order to demonstrate its potential. This chapter represents samples of segmentation results.

6.1 Corpus Callosum(CC) Segmentation

We have tested our B-Spline template snake on 25 CC images using only five user input lines for each. The average error when compared against expert manual segmentation is 0.66 pixel, where the error is defined as the shortest distance between the snake points and the expert hand-segmented boundaries. This error can be reduced by using additional program-added degrees of freedom or by additional user input lines in the initialization step. Note that once the input lines are entered, no further user editing of the snake is needed. Quantitive error measurements for 25 CCs relative to manual segmentation is summarized in table 6.1. Important details to consider regarding table 6.1 are as follows:

- All the images are segmented with only 5 initial lines.
- A *soft* control point has been automatically added in the middle of every polygon edge.
- 7 points per element have been saved for comparison with hand segmented images. As a result, we have $5 \times 2 \times 2 \times 7 = 140$ saved points per snake (assuming there are 5 user lines and auto middle point for every edge).
- Image forces are based on local statistical intensities described in section 5.2.4

Table 6.1: Mean, min, max and standard deviation of the shortest distances between automatically extracted and expert segmented CC boundaries.The last column is pixel distance between the automatically and manually labeled rostrum tip.

Case no.	Mean (pixel)	Min (pixel)	Max (pixel)	S.D. (pixel)	Rostrum tip (pixel)
1	0.7664	0.0081	1.8056	0.4106	1.8468
2	0.5874	0.0043	1.6456	0.3781	0.8621
3	0.6447	0.0001	1.8093	0.3955	1.0095
4	0.5530	0.0033	1.8015	0.3804	1.5201
5	0.5573	0.0124	2.0024	0.3136	0.9568
6	0.6976	0.0047	2.1281	0.4918	1.1551
7	0.7340	0.0011	2.3320	0.4771	1.7614
8	0.5562	0.0003	2.0921	0.4105	1.8577
9	0.6665	0.0005	2.6744	0.4822	1.4507
10	0.6478	0.0100	1.8693	0.3315	1.9780
11	0.7516	0.0109	3.3611	0.5449	2.0826
12	0.6643	0.0091	1.8679	0.4174	2.8225
13	0.7033	0.0067	2.2122	0.4814	1.5439
14	0.6183	0.0009	1.5600	0.3698	0.5282
15	0.6836	0.0050	1.6469	0.3946	1.8151
16	0.6444	0.0042	2.1135	0.4502	2.6555
17	0.6542	0.0005	2.8491	0.4361	0.8521
18	0.6508	0.0027	1.9678	0.4038	2.1007
19	0.7355	0.0168	2.2821	0.4410	1.7261
20	0.6784	0.0137	2.1251	0.3979	3.4104
21	0.6334	0.0029	1.8341	0.4103	0.3101
22	0.6980	0.0027	3.0262	0.5451	1.1841
23	0.7494	0.0055	4.5166	0.6004	2.0205
24	0.5748	0.0116	1.2332	0.2672	2.7903
25	0.7556	0.0027	2.0192	0.5202	1.8805
Average	**0.6643**	**0.0056**	**2.1910**	**0.4301**	**1.6848**

Figures 6.1, 6.2, and 6.3 represent some of the segmentation results. The first column represents the user input lines, the second column represents the corresponding initial snake, and the last column represents the final results.

As we mentioned before by adding even 1 more user line we get more accurate results. Figure 6.4 shows two complex CC's that have been segmented using 6 initial lines.

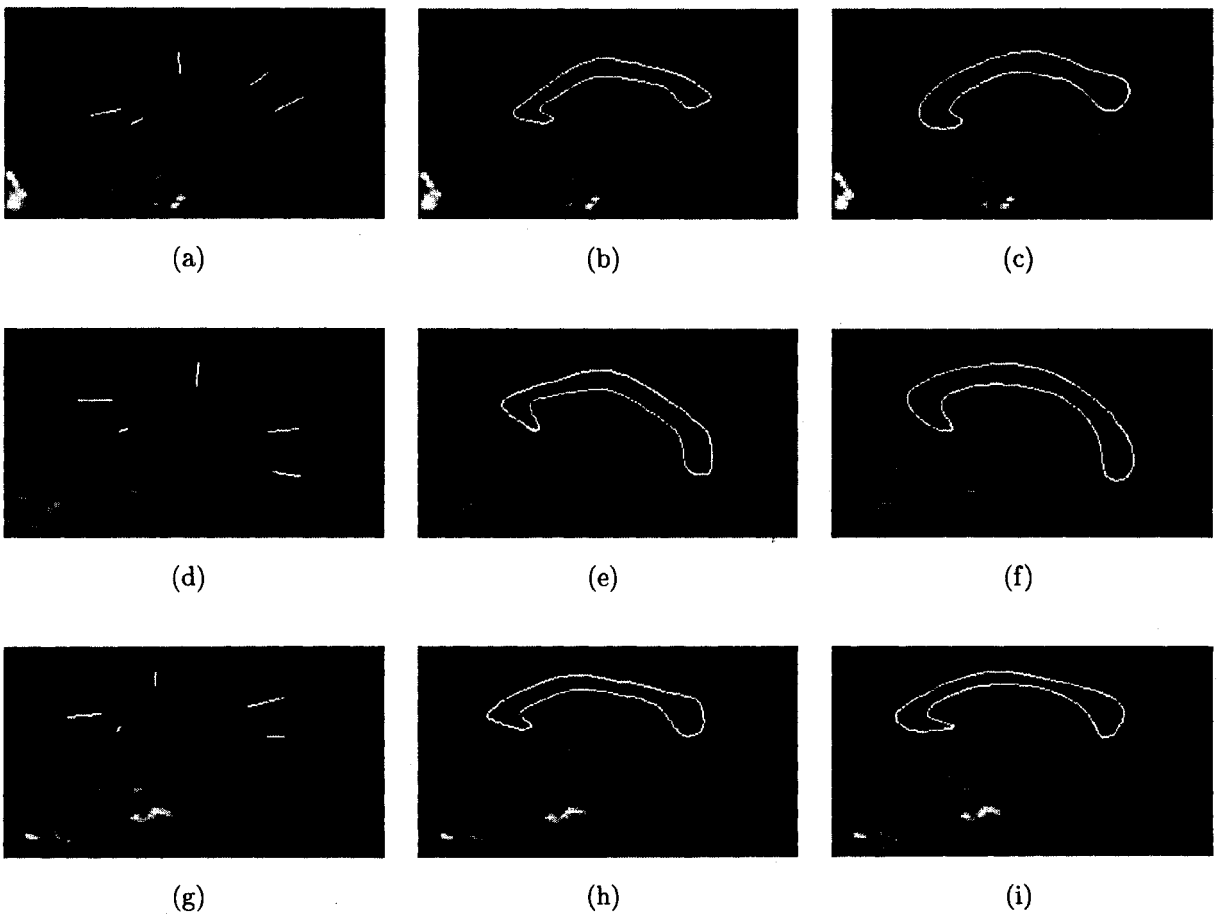


Figure 6.1: Samples of segmenting CCs images. The first column is user init lines, the second column is initial snake, and the last column is final result.

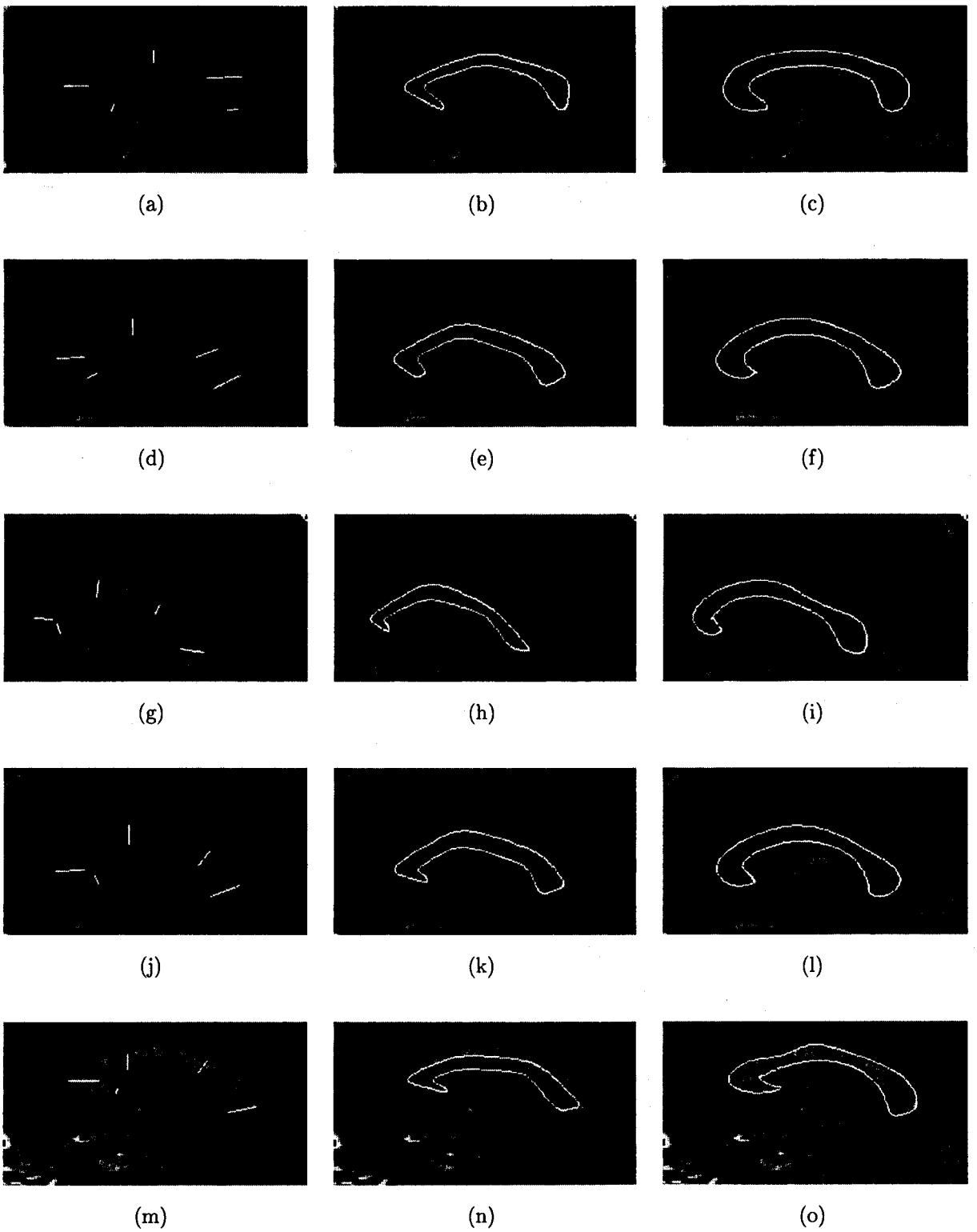


Figure 6.2: Samples of segmenting CCs images (continued from figure 6.1).

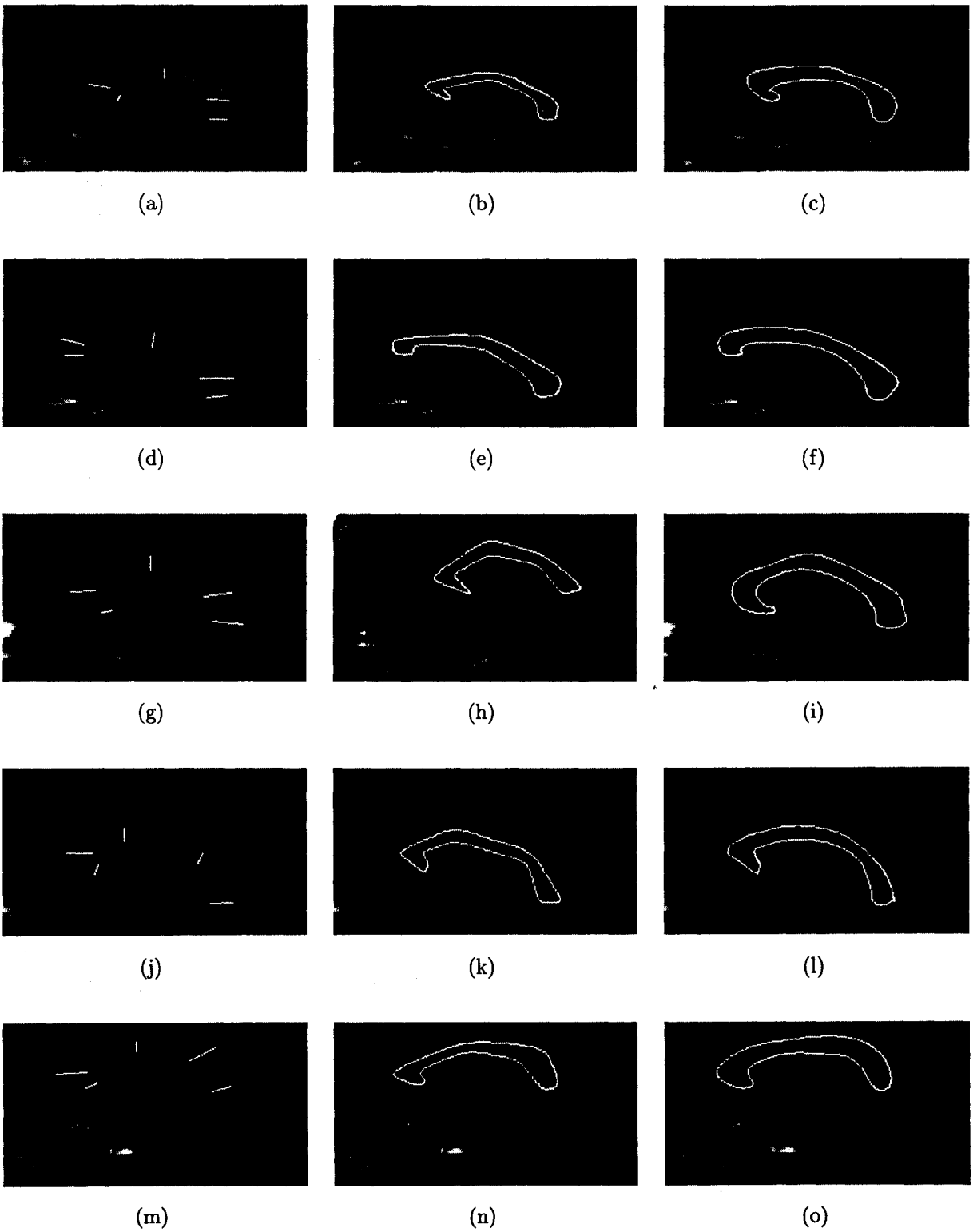


Figure 6.3: Samples of segmenting CCs images (continued from figure 6.2).

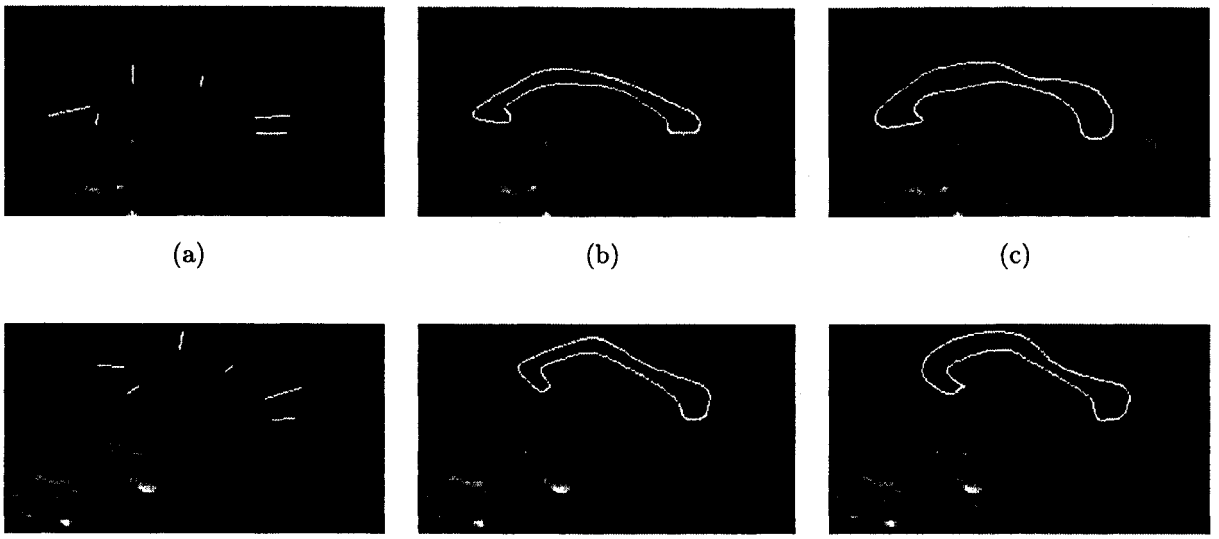


Figure 6.4: Complex CC images segmented with 6 user init lines.

6.2 Bone Segmentation

Figure 6.5 shows the initial user lines, the resulting initial snake and the final segmentation of two arm bones in an X-ray image. The image is very noisy, especially where the two bones overlap. There are many large gaps in the edges of the bone boundary and many spurious edges inside the bone. Notice how with only a few input lines the initial snake is almost the same shape as the bone. The model ignores edges that are not of a specific magnitude and direction.

To show power of our model, we applied an inflating snake on the bone image, with different parameters, but it leaked into the other bone. Usually, setting the parameters of an inflating snake is difficult for an object with a lot of texture. If we set the internal too small, or if we set the inflation force too strong, the snake may leak. If we set the internal forces too high or the inflation snake too weak, the snake gets caught on spurious/noisy edges or cannot flow into protrusions in the object.

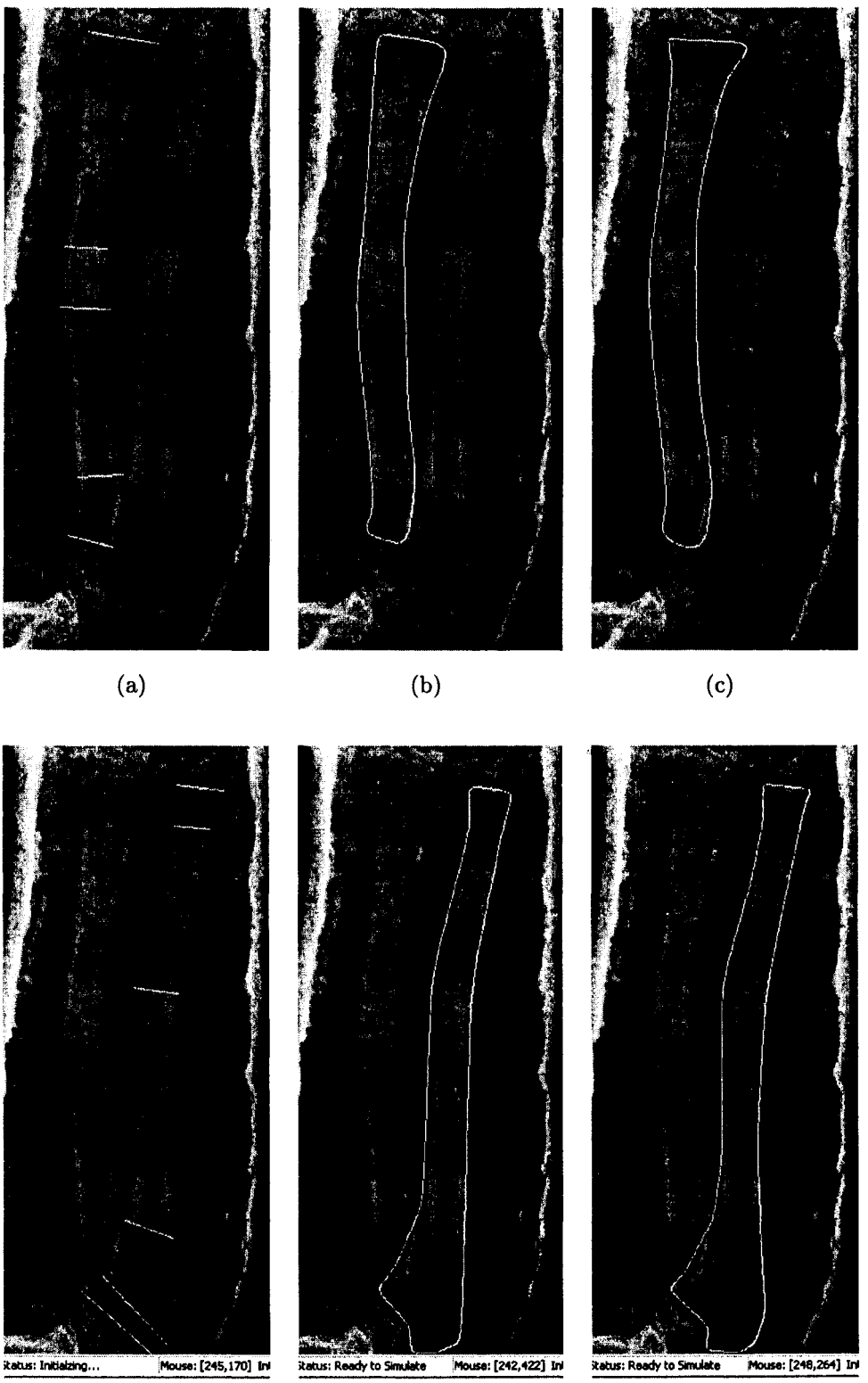


Figure 6.5: Segmenting two arm bones in an X-ray image. From left to right: user init line, initial snake, and the final result.

6.3 Bladder Segmentation

Figure 6.6 shows the bladder segmentation using 5 input lines. Notice that the model successfully segments the complex lower part of the bladder.

(a) (b) (c)

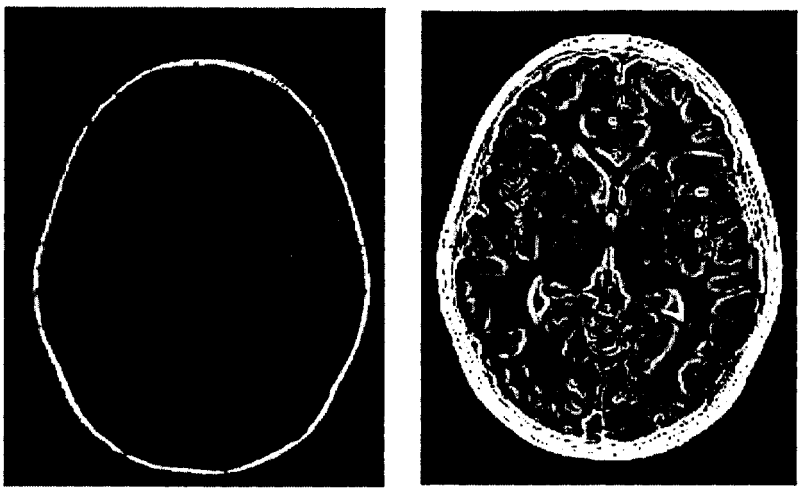
Figure 6.6: Segmenting the balder using 5 input lines.

6.4 Putamen Segmentation

Segmenting the putamen is a difficult task. The putamen is adjacent to the grey matter and to globus pallidus (both with highly similar intensity to the putamen). Consequently, there are many large gaps , noise, and texture in the putamen edges (see figure 6.7).

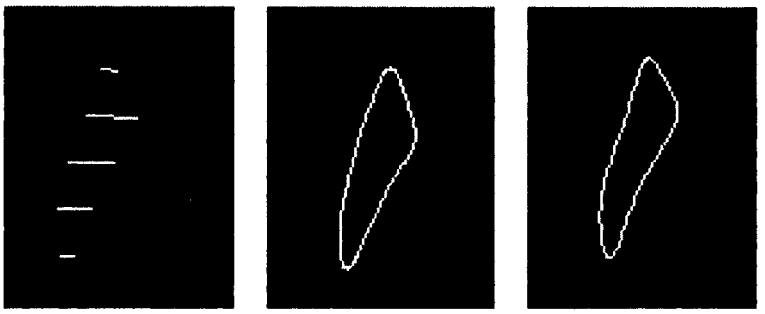
We have applied our model to segment the putamen from several slices of an MR image volume. The user enters lines in the first image and turns on the control polygon spring constraints, so all control polygon edges and control point pairs on opposite sides of the snake are connected with spring (Fig. 6.8).

Once the initial model has been fitted to the first slice (Fig. 6.8(c)), the snake is able to successfully "track" the putamen in neighboring slices with no need to user interaction (Fig. 6.9). This is the case especially if images are generated at high frequencies. When two consequent slices are considerably different, user may initialize a new snake on the new slice to get more accurate result and less editing.



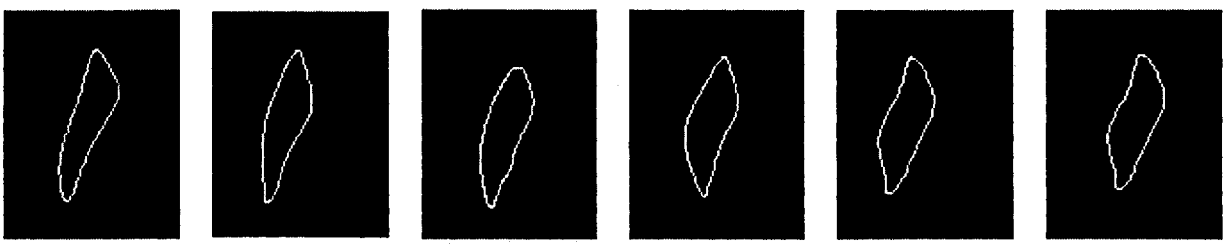
(a) (b)

Figure 6.7: The putamen (left) and its potential (right). Notice that even with lots of enhancements, the edges are still quite noisy.



(a) (b) (c)

Figure 6.8: Segmenting a single frame potamen, using 5 input lines.



(a) (b) (c) (d) (e) (f)

Figure 6.9: Segmenting a sequence of putamen frames (from left to right). There is no need for user initialization, instead the result of each frame is used as the initial snake for the next frame.

Chapter 7

Conclusion

The thesis has developed a user-definable deformable template model using a B-Spline snake.

A simple but effective and efficient initialization process, coupled with the properties of a B-spline, enables the construction of a snake that is extremely close to and similar in shape to the target anatomical structure. Since each snake element roughly corresponds to a specific object boundary segment, the user is able to create customized external forces and utilize custom fitting algorithms, ensuring a more robust and automatic segmentation result.

Moreover, since the B-spline control polygon is a coarse approximation of the curve and hence a coarse approximation of the target object boundary, it is a convenient frame upon which to build global model deformation control.

In chapter 2 several design principles for optimal interactive segmentation techniques were listed. This chapter will briefly discuss some of these principles with respect to the B-spline template snake.

Principle 1 states that computation and user interaction should be integrated into one process. The physics-based framework of snakes is inherently integrative. The computational component solves force balance equations and includes user derived forces. Pushing and pulling on the snake via mouse spring forces is very intuitive - a key feature of semi-automatic deformable models. Principle 3 states that users should initialize the segmentation method with key information which will lead the method to an accurate result more quickly. This is the key feature of B-spline template snakes. They were designed to ex-

ɔloit user recognition abilities. The user input lines are often the only interaction required subsequent manual editing of the snake is seldom needed. Principle 4 states: user control ɨhould be maintained throughout the whole process to generate accurate results. Again, 3-spline template snakes maintain the powerful force-based editing semantics of traditional ɨnakes. At any time during the segmentation, the user is able to correct the result, either ɔy pulling on the snake, pulling on the control polygon, or adding more user input lines or ɔoints. Furthermore, in on-the-fly segmentation, the user has even tighter control over the ɨntire segmentation process. Principle 5 is: proper visualization of the computational part s needed to enable an effective user response. This property is, again, inherent to all snakes. Principle 6 states: emphasize computation after each user interaction for optimal repeatability. The use of the user input lines, entered at landmark locations and in a specific order ɨnsures repeatability. As mentioned, there is little need for subsequent manual editing - the ɔomputational component of template snakes has enough user information in the beginning ɔ perform an accurate segmentation. Principle 7 states: add intelligent behavior to elevate ɨhe abstraction level of the interaction. This was the guiding principle behind the design of 3-spline template snakes and is the thrust of this thesis.

Finally, principle 8 - add the capability to learn from the user interaction and consequently ɨliminate the need for further interaction - is a topic for future research.

Also, as future works, the following features can be considered:

- The use of nonlinear magnification in a small region around the cursor to ease user burden of interactive line placement. Also, the use of a small secondary zoom window.
- The use of a pressure sensitive pen input device and a tablet rather than a mouse for more comfortable and natural user input.

Appendix A

Hermite and B-Spline Curves: An Overview

The following sections discuss Hermite and B-Spline parametric splines. Generally, they are composed of individual parametric-curve segments, joined to form a single curve. Furthermore, their continuity is controlled at segment joints. The spline degree refers to the degree of its piecewise polynomials.

A.1 B-Spline Approximation

B-Spline approximation is a piecewise polynomial curve defined by a set of control points that ordinarily are not interpolated. The degree of the polynomial functions representing the curve (basis functions) is independent of the number of control points. The control points influence only a few of the nearby curve segments. As a result, changes in the position of a control point do not propagate shape changes globally.

Although, B-splines are smooth curves or surfaces with high level of geometric continuity, they can have sharp corners (continuity C^0) by duplicating some control points.

B-spline curves are defined by:

$$P(u) = \sum_{0}^{n} p_k B_{k,d}(u) \quad u \in [0,1] \quad d \in [2, n+1], \tag{A.1}$$

where p_k are the input set of $n+1$ control points. Blending functions for B-Spline curves

are defined by the following recursion formula:

$$B_{k,1} = \begin{cases} 1 & u \in [u_k, u_{k+1}] \\ 0 & \text{otherwise} \end{cases}, \tag{A.2}$$

$$B_{k,d}(u) = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u), \tag{A.3}$$

where $B_{k,d} = 0$ for $B_{k,d} = \frac{0}{0}$, and n is the number of control points. The degree of the polynomial curve is $d - 1$ and its continuity over the range of u is C^{d-2}. Each blending function is defined over d subinterval of the total range of u. The selected set of subinterval endpoints u_j is referred to as a knot vector which are $u_j \leq u_{j+1}$. Each control point controls d curve segments.

A valuable property of B-Splines is their *invariance under an affine transformation (translation, rotation, scaling)* [7]. In other words, the following two methods produce the same result:

1. Computing the point $p(u_i) = \sum_0^n p_i B_{i,d}$ and then applying an affine transformation A such that $p'(u_i) = Ap(u_i)$.

2. Applying an affine transformation A to control points such that $p'_i = Ap_i$ and then calculate the curve represented by the transformed control points at u_i, where $p'(u_i) = \sum_0^n p'_i B_{i,d}$.

This property allows us to translate, rotate, and scale a B-spline curve while preserving the relationship between evaluated points and the corresponding control points. Also, we can change the number of control points without changing the degree of the polynomial.

A.2 Uniform Closed Periodic B-Splines

In a uniform B-Spline, where spacing between knot values is constant, blending functions are periodic, i.e. for given values of n and d all blending functions have the same shape.

$$B_{k,d}(u) = B_{k+1,d}(u + \triangle u) = B_{k+2,d}(u + 2\triangle u)$$

, where Δu is the interval between adjacent knot values.

Uniform B-spline curves are particularly useful for generating closed curves. Blending functions of a uniform cubic (d=4) B-spline using Eq. A.3 are:

$$\begin{cases} B_{1,4} &= \frac{1}{6}(-u^3+3u^2-3u+1) \\ B_{2,4} &= \frac{1}{6}(3u^3-6u^2+4) \\ B_{3,4} &= \frac{1}{6}(-3u^3+3u^2+3u+1) \\ B_{4,4} &= \frac{1}{6}u^3 \end{cases}$$

We have the following equation for open and closed curves, respectively:

$$p_i(u) = B_{1,4}(u)p_{i-1} + B_{2,4}(u)p_i + B_{3,4}(u)p_{i+1} + B_{4,4}p_{i+2} \quad i \in [1 \ldots n-2],$$

$$p_i(u) = B_{1,4}(u)p_{i-1} + B_{2,4}(u)p_{(i\oplus(n+1)} + B_{3,4}(u)p_{i+1\oplus(n+1)} + B_{4,4}p_{i+2\oplus(n+1)} \quad i \in [1 \ldots n+1],$$

where, $\oplus$ stands for modulo.

A.3 Hermite Interpolation

A Hermite spline is an interpolating piecewise cubic polynomial defined by its two end points and the tangent vectors at those points. Hermite splines can be adjusted locally because each curve element is only dependent on its end point constraints. If $P(u)$ represents a parametric cubic point function for the curve element section between control points the boundary conditions that define this Hermite curve section are:

$$\begin{array}{llll} P(0) = p_k & & P(1) = p_{k+1} \\ p'(0) = Dp_k & & P'(1) = Dp_{k+1}, \end{array}$$

where Dp_k and Dp_{k+1} specify the values of the parametric derivatives (slope of the curve) at control points p_k and p_{k+1}, respectively.

$$p(u) = au^3 + bu^2 + cu + d, \tag{A.4}$$

where the x component of P is $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad u \in [01]$.

Applying the boundary conditions to the Eq. A.4 to find vector coefficients a,b,c, and d, we obtain the polynomial form:

$$\begin{aligned} p(u) &= p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + \\ &\quad Dp_k(u^3 - 2u^2 + u) + Dp_{k+1}(u^3 - u^2) \\ &= p_k H_0(u) + p_{k+1} H_1(u) + Dp_k H_2(u) + Dp_{k+1} H_3(u). \end{aligned} \tag{A.5}$$

The polynomials H_k for $k = 0, 1, 2, 3$ are referred to as blending functions Since they blend the boundary constraint values (end point coordinates and slopes) to obtain each coordinate position along the curve. Hermite curves are fairly easy to subdivide. They have C^1 continuity. i.e. the first parametric derivatives (tangent lines) of the coordinate functions for two successive curve segment are equal at their joining point. Its lack of invariance under affine transformations can be troublesome if not accounted for.

Appendix B

Finite Difference Snakes

The use of Finite differences (FD) is the traditional method for local representation of snakes. A $2D$ FD snake contour is represented as an ordered set of discrete points $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \cdot, \mathbf{v}_n)$ where $\mathbf{v}_i(s) = (x_i(s), y_i(s))^\top$, with $\mathbf{v}_1 = \mathbf{v}_n$ for closed snakes. The points (vertices) are connected by straight lines,resulting in a piece-wise linear curve. Letting $E_{ext} = E_{image} + E_{con}$ and setting $\alpha(s) = \alpha$ and$\beta(s) = \beta$ to constants, minimizing the energy functional(C.2) gives rise to the following two independent Euler equations:

$$-\alpha x_{ss} + \beta x_{ssss} + \frac{\partial E_{ext}}{\partial x} = 0. \tag{B.1}$$

$$-\alpha y_{ss} + \beta y_{ssss} + \frac{\partial E_{ext}}{\partial y} = 0. \tag{B.2}$$

The discrete form of the snake energy functional can be written :

$$E_{snake} = \sum_{i=1}^{n} E_{int}(i) + E_{ext}(i). \tag{B.3}$$

Let

$$\frac{\partial E_{ext}}{\partial x} = F_x(i) = \frac{[E_{ext}(x(i)+h, y(i)) - E_{ext}(x(i)-h, y(i))]}{2h}, \tag{B.4}$$

$$\frac{\partial E_{ext}}{\partial y} = F_y(i) = \frac{[E_{ext}(x(i), y(i)+h) - E_{ext}(x(i), y(i)-h)]}{2h}, \tag{B.5}$$

and approximating the derivatives with finite differences:

$$x_{ss}(i) = x(i-1) - 2x(i) + x(i+1), \tag{B.6}$$

$$y_{ss}(i) = y(i-1) - 2y(i) + y(i+1), \tag{B.7}$$

$$x_{ssss}(i) = x_{ss}(i-1) - 2x_{ss}(i) + x_{ss}(i+1), \tag{B.8}$$

$$y_{ssss}(i) = y_{ss}(i-1) - 2y_{ss}(i) + y_{ss}(i+1). \tag{B.9}$$

Substituting (B.4), (B.5), (B.6),(B.7), (B.8) and (B.9), into (B.1) we have

$$\mathbf{XA} + \mathbf{F}_x(\mathbf{X}, \mathbf{Y}) = \mathbf{0}, \tag{B.10}$$

$$\mathbf{YA} + \mathbf{F}_y(\mathbf{X}, \mathbf{Y}) = \mathbf{0}, \tag{B.11}$$

where the stiffness matrix A is pentadiagonal and banded. A dynamic snake is constructed (first order dynamic equations of motion only) with a damping density γ:

$$\mathbf{X}_t\mathbf{A} + \mathbf{F}_x(\mathbf{X}_{t-1}, \mathbf{Y}_{t-1}) = -\gamma(\mathbf{X}_t - \mathbf{X}t - 1), \tag{B.12}$$

$$\mathbf{Y}_t\mathbf{A} + \mathbf{F}_y(\mathbf{X}_{t-1}, \mathbf{Y}_{t-1}) = -\gamma(\mathbf{Y}_t - \mathbf{Y}t - 1). \tag{B.13}$$

or

$$\mathbf{X}_t = (\mathbf{X}_{t-1} - \mathbf{F}_x(x_{t-1}, yt - 1))\,(\mathbf{A} + \gamma\mathbf{I})^{-1}, \tag{B.14}$$

$$\mathbf{Y}_t = (\mathbf{Y}_{t-1} - \mathbf{F}_y(x_{t-1}, yt - 1))\,(\mathbf{A} + \gamma\mathbf{I})^{-1}. \tag{B.15}$$

In equations B.12, the internal forces (i.e. $\mathbf{XA}$ and $\mathbf{YA}$) are evaluated at time t which yields an implicit Euler step. Taking into account derivatives of the external forces can break the banded structure of $\mathbf{A}$.Therefore, it is assumed that $\mathbf{F}_x$ and $\mathbf{F}_y$ are constant during a time step and they are evaluated at time $t-1$,yielding and explicit Euler step with respect to these forces. At equilibrium, the time derivatives vanish and equations (B.12) and (B.13) reduces to equations (B.10) and (B.11).

Appendix C

Euler Equations for Snakes

The task of fitting a snake contour to the desired object boundary is a classic optimization or minimization problem. There are several methods to perform this task, one oft hem is the variational calculus. In variational calculus, function $\mathbf{v}(s)$ minimizes the functional

$$E(\mathbf{v}) = \int_a^b F(s, \mathbf{v}, \mathbf{v}', \mathbf{v}'')ds,$$

if the following Euler-Lagrange equation is satisfied:

$$\frac{\partial F}{\partial \mathbf{v}} - \frac{d}{ds}\frac{\partial F}{\partial \mathbf{v}'} - \frac{d}{ds^2}\frac{\partial^2 F}{\partial \mathbf{v}''} = 0. \tag{C.1}$$

Assuming $E_{ext} = E_{image}(\mathbf{v}(s)) + E_{constraint}$, we have

$$E^*_{snake} = \int_0^1 [\left(\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2\right)/2 + E_{ext}(s, \mathbf{v}, \mathbf{v}_s, \mathbf{v}_{ss})]ds. \tag{C.2}$$

Letting

$$F(s, \mathbf{v}, \mathbf{v}_s, \mathbf{v}_{ss}) = \left(\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2\right)/2 + E_{ext}(s, \mathbf{v}, \mathbf{v}_s, \mathbf{v}_{ss})],$$

where $v_s = \frac{\partial \mathbf{v}}{\partial s}$ and $\mathbf{v}_{ss} = \frac{\partial^2 \mathbf{v}}{\partial s^2}$, we have

$$\frac{\partial F}{d\mathbf{v}} = \frac{\partial E_{ext}}{\partial \mathbf{v}}, \tag{C.3}$$

$$\frac{\partial F}{\partial \mathbf{v}_s} = \alpha(s)\mathbf{v}_s + \frac{\partial E_{ext}}{\partial \mathbf{v}_s}, \tag{C.4}$$

$$\frac{\partial F}{\partial \mathbf{v}_{ss}} = \beta(s)\mathbf{v}_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}_{ss}}. \tag{C.5}$$

ıbstituting Eq.C.3 and Eq.C.4 and Eq.C.5 into Eq.C.1 gives:

$$\frac{\partial E_{ext}}{\partial \mathbf{v}} - \left(\alpha(s)\mathbf{v}_s + \frac{\partial E_{ext}}{\partial \mathbf{v}_s}\right)_s + \left(\beta(s)\mathbf{v}_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}_{ss}}\right)_{ss} = 0, \tag{C.6}$$

ɔ.

$$-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + \frac{\partial E_{ext}}{\partial \mathbf{v}} - (\frac{\partial E_{ext}}{\partial \mathbf{v}_s})_s + (\frac{\partial E_{ext}}{\partial \mathbf{v}_{ss}})_{ss} = 0. \tag{C.7}$$

ɛtting

$$Q(s, \mathbf{v}, \mathbf{v}_s, \mathbf{v}_s s) = \frac{\partial E_{ext}}{\partial \mathbf{v}} - (\frac{\partial E_{ext}}{\partial \mathbf{v}_s})_s + (\frac{\partial E_{ext}}{\partial \mathbf{v}_{ss}})_{ss}, \tag{C.8}$$

ıen the Euler equation in C.6 becomes as follows:

$$-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + Q(s, \mathbf{v}, \mathbf{v}_s, \mathbf{v}_{ss}) = 0, \tag{C.9}$$

ɛtting $(\frac{\partial E_{ext}}{\partial \mathbf{v}_s})_s = 0$ and $(\frac{\partial E_{ext}}{\partial \mathbf{v}_{ss}})_{ss} = 0$ in Eq.C.8 gives :

$$-(\alpha(s)\mathbf{v}_s)_s + (\beta(s)\mathbf{v}_{ss})_{ss} + (\frac{\partial E_{ext}}{\partial \mathbf{v}}\mathbf{v}_{ss}) = 0, \tag{C.10}$$

hich is the equation of equilibrium for the snake model. This vector-valued partial differ-
ıtial equation expresses the balance of internal and external forces when the Snake rests
equilibrium. The first two terms represent the internal stretching and bending forces
spectively, while the third term represents the external forces.

Bibliography

[1] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Trans. on PAMI*, 12(9):855–867, 1990.

[2] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *Proc. Fifth International Conf. on Computer Vision (ICCV'95), Cambridge, MA, June, 1995*, pages 694–699, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[3] A. Chakraborty, L.H. Staib, and J.S. Duncan. Deformable boundary finding influenced by region homogeneity. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR'94), Seattle, WA, June, 1994*, pages 624–627, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[4] L.D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, March 1991.

[5] L.D. Cohen and I. Cohen. Finite element methods for active contour models and balloons for 2D and 3D images. *IEEE Trans. on PAMI*, 15(11):1131–1147, November 1993.

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

[7] Michael E.Mortenson. *Geometric Modeling.* Wiley Computer Publishing, 1997.

[8] A.X. Falão, J.K. Udupa, S. Samarasekera, and S. Sharma. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing*, 60, 1998.

[9] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. on Computers*, 22(1):67–92, 1973.

[10] D. Fritsch, S. Pizer, L. Yu, V. Johnson, and E. Chaney. Segmentation of medical image objects using deformable shape loci. In *Information Processing in Medical Imaging: Proc. 15th Int. Conf. (IPMI'97), Poultney, Vermont, June, 1997*, pages 127–140, 1997.

[11] T. McInerney G. Hamarneh. Physics-based shape deformations for medical image analysis. *Technical report CSRG-436, Department of Computer Science, University of Toronto*, 2001.

[12] J.M. Gauch, H.H. Pien, and J. Shah. Hybrid boundary-based and region-based deformable models for biomedical image segmentation. In *Mathematical Methods in Medical Imaging III*, volume 2299 of *SPIE Proc.*, pages 72–83, San Diego, CA, 1994. SPIE.

[13] R.P. Grzeszczuk and D.N. Levin. Brownian strings: Segmenting images with stochastically deformable contours. In Robb [28], pages 72–89.

[14] I.L. Herlin, C. Nguyen, and C. Graffigne. A deformable region model using stochastic processes applied to echocardiographic images. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR'92), Urbana, IL, June, 1992*, pages 534–539, Los Alamitos, CA, 1992. IEEE Computer Society Press.

[15] A. Hill, A. Thornham, and C.J. Taylor. Model-based interpretation of 3D medical images. In *Proc. 4th British Machine Vision Conf. (BMVC'93), Surrey, UK, September, 1993*, pages 339–348. BMVA Press, 1993.

[16] J. Ivins and J. Porrill. Statistical snakes: Active region models. In *Proc. 5th British Machine Conf. (BMVC'94)*, pages 377–386. BMVA Press, 1994.

[17] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[18] Jianming Liang, Tim McInerney, and Demetri Terzopoulos. Interactive medical image segmentation with united snakes. In *Proc. Second International Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI 99)*, Cambridge, England, September 1999. Springer.

[19] Jianming Liang, Tim McInerney, and Demetri Terzopoulos. United snakes. In *Proc. Seventh International Conf. on Computer Vision (ICCV'99)*, Kerkyra (Corfu), Greece, September 1999.

[20] Jianming Liang, Tim McInerney, and Demetri Terzopoulos. United snakes. Technical report, Department of Computer Science, University of Toronto, Canada, 1999. (submitted to ICCV).

[21] J.F. Mangin, F. Tupin, V. Frouin, I. Bloch, R. Rougetet, J. Regis, and J. Lopez-Krahe. Deformable topological models for segmentation of 3D medical images. In Y. Bizais, C. Barillot, and R. Di Paola, editors, *Information Processing in Medical Imaging: Proc. 14th Int. Conf. (IPMI'95), Ile de Berder, France, June, 1995*, volume 3 of *Computational Imaging and Vision*, pages 153–164, Dordrecht, The Netherlands, 1995. Kluwer Academic.

[22] Tim McInerney, Ghassan Hamarneh, and Demetri Terzopoulos. Deformable organisms for automatic medical image analysis. *Medical Image Analysis*, 6:251–266, 2002.

[23] Tim McInerney and Demetri Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[24] Tim McInerney and Demetri Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4:73–91, 2000.

[25] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Trans. on PAMI*, 15(6):580–591, 1993.

[26] W. Neuenschwander, P. Fua, L. Iverson, G. Székely, and O. Kubler. Ziplock snakes. 25(3):191–201, 1997.

[27] C. S. Poon, M. Braun, R. Fahrig, A. Ginige, and A. Dorrell. Segmentation of medical images using an active contour model incorporating region-based images features. In Robb [28], pages 90–97.

[28] R.A. Robb, editor. *Proc. Third Conf. on Visualization in Biomedical Computing (VBC'94), Rochester, MN, October, 1994*, volume 2359 of *SPIE Proc.*, Bellingham, WA, 1994. SPIE.

[29] Davis S.Burnett. *Finite Element Analysis From Concept To Applications.* Addison-Wesley, 1987.

[30] A.W.M Smeulders S.D Olabarriaga. Interaction in the segmentation of medical images:a survey. 2000.

[31] L.H. Staib and J.S. Duncan. Boundary finding with parametrically deformable models. *IEEE Trans. on PAMI*, 14(11):1061–1075, November 1992.

[32] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[33] B.C. Vemuri and A. Radisavljevic. Multiresolution stochastic hybrid shape models with fractal priors. *ACM Trans. on Graphics*, 13(2):177–207, April 1994.

[34] B.C. Vemuri, A. Radisavljevic, and C. Leonard. Multiresolution 3D stochastic shape models for image segmentation. In A.C.F. Colchester and D.J. Hawkes, editors, *Information Processing in Medical Imaging: Proc. 13th Int. Conf. (IPMI'93), Flagstaff, AZ, June, 1993*, Lectures Notes in Computer Science, pages 62–76. Springer–Verlag, 1993.

[35] M. Worring, A.W.M. Smeulders, L.H. Staib, and J.S. Duncan. Parameterized feasible boundaries in gradient vector fields. In A.C.F. Colchester and D.J. Hawkes, editors,

Information Processing in Medical Imaging: Proc. 13th Int. Conf. (IPMI'93), Flagstaff, AZ, June, 1993, Lectures Notes in Computer Science, pages 48–61. Springer–Verlag, 1993.

[36] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.