

**FACULTY FOR MATHEMATICAL AND NATURAL  
SCIENCES  
UNIVERSITY OF TROMSØ**

**DIPLOMA PROJECT**

**A Comparative Study of Active Contour  
Models for Boundary Detection in Brain  
Images**

**Vedad Hadziavdic**

# Contents

<b>1</b>	<b>Foreword</b>	<b>5</b>
<b>2</b>	<b>Introduction</b>	<b>6</b>
2.1	What is a Snake? . . . . .	7
2.2	What is a Deformable Template? . . . . .	9
2.3	What is a Dynamic Contour? . . . . .	10
2.4	The Objective of the Work . . . . .	11
<b>3</b>	<b>Classification</b>	<b>13</b>
<b>4</b>	<b>Representation of Parametric Deformable Contours</b>	<b>16</b>
4.1	The Choice of Curve Representation . . . . .	16
4.2	Definition of the Parametric Curve Representing the Snake . . . . .	17
4.3	Arc Length as the Curve Parameter . . . . .	17
4.4	Spline Basis . . . . .	18
4.4.1	B-Splines . . . . .	18
4.4.2	Splines . . . . .	21
4.4.3	Cubic B-Splines and Polygon Approximation . . . . .	21
4.4.4	Discussion on B-Splines as Basis for Snakes . . . . .	26
4.5	Fourier Basis . . . . .	27
4.5.1	Fourier Series Representation of a Periodic Functions . . . . .	27
4.5.2	Alternative Fourier Series Representation of a Real Periodic Function . . . . .	28
4.5.3	The Speed of Convergence of the Fourier Series - Trunca- tion of the Series . . . . .	30
4.5.4	Fourier Series Computational Algorithms . . . . .	31
4.5.5	Fourier Basis Properties and Snakes . . . . .	34
4.5.6	Comparison and Conclusion . . . . .	35
<b>5</b>	<b>The Image Energy</b>	<b>36</b>
5.1	Some Relevant Definitions from Vector Calculus . . . . .	37
5.2	Image Energy Models . . . . .	38
5.2.1	The Original Force Field . . . . .	38
5.2.2	The “Balloon” Force Field . . . . .	45

5.2.3	The Gradient Vector Flow Force Field . . . . .	46
5.3	Experimental Results . . . . .	49
5.4	Discussion . . . . .	113
<b>6</b>	<b>Minimizing the Energy Functional</b>	<b>114</b>
6.1	Calculus of variations . . . . .	114
6.2	Computation of the Snake Evolution Equation . . . . .	116
<b>7</b>	<b>Numerical Solutions to the Snake Evolution Equation</b>	<b>118</b>
7.1	Finite Difference Method Applied to the Snake Evolution Equation .	118
7.1.1	Finite Difference Methods . . . . .	119
7.2	Fourier Spectral Method Applied to the Snake Evolution Equation .	122
7.2.1	Spectral Methods . . . . .	122
7.2.2	Fourier Spectral Method . . . . .	122
7.3	A Fourier Spectral Algorithm for the Snake Evolution Equation . .	124
7.4	Dynamic Programming Solution to the Snake Evolution Equation .	124
7.4.1	Dynamic Programming: Continuous Case . . . . .	125
7.4.2	Dynamic Programming Algorithm for Snake Evolution E- quation . . . . .	126
7.4.3	Experimental Results . . . . .	127
7.4.4	Comparison and Conclusion . . . . .	133
<b>8</b>	<b>A Statistical Approach to Snakes</b>	<b>135</b>
8.1	Maximum Likelihood Estimators . . . . .	136
8.2	Snake Within the Bayesian Framework . . . . .	137
<b>9</b>	<b>Deformable Templates</b>	<b>138</b>
9.1	Principles Behind Deformable Templates . . . . .	139
9.1.1	Representation of Deformable Templates . . . . .	139
9.1.2	Model Definition Framework . . . . .	139
9.1.3	Deformation Methods . . . . .	142
9.1.4	Principal Component Analysis . . . . .	145
9.2	An Overview of the Deformable Templates . . . . .	147
9.2.1	The Hough Transform . . . . .	147
9.2.2	Parametrically Deformable Models . . . . .	151
9.2.3	Active Shape Point Distribution Model . . . . .	153

9.2.4	Deformable Template Approach to Some Specific Applications . . . . .	157
9.3	Deformable Template Model Proposed by Jain et.al . . . . .	158
9.3.1	A Probabilistic Model of Deformation . . . . .	159
9.3.2	A Priori Probability Function . . . . .	160
9.3.3	Likelihood Function . . . . .	161
9.3.4	A Posteriori Probability Function . . . . .	162
<b>10</b>	<b>Discussion</b>	<b>162</b>
<b>11</b>	<b>Conclusion</b>	<b>165</b>
<b>12</b>	<b>Acknowledgments</b>	<b>165</b>

# 1 Foreword

Active contour models are an attractive approach to boundary detection in medical images. Over the last ten years, numerous methods have been developed within different theoretical frameworks.

There are few articles [39],[40] and books [6] that attempt to give an overview of these methods within a structured framework. These works are, however, surveys and not comparative studies.

The objective of this work is to

- study the theoretical frameworks of different active contour models
- classify the methods according to several properties
- compare their properties
- implement some of the methods and test them on simulated and real images
- determine which methods satisfy the requirements posed by the nature of the medical images

Furthermore, the partial differential equation governing the so called snake, which is an active contour model, has been implemented using implicit finite difference method [22], dynamic programming [16] and finite element method [13], whereby the two first methods have been most frequently used in the literature.

In this work, a new algorithm based on Fourier spectral method will be developed. It will also be shown that it has better performance than the methods used in literature.

This work was done over the period of four month at the University of Bergen and University of Tromsø. The medical images are the courtesy of the Haukeland University Hospital.

## 2 Introduction

For a scientist, an image is an information carrier, but the information of interest may not always be perceivable with the human eye only. It may be corrupted by noise or simply be tied up with details and information of no interest so that it is not useful for purposes. In such cases we may have to develop techniques to extract that information from the image. This area of image processing is called image analysis.

The first and the most important step in image analysis is to segment the image (often referred to as boundary estimation, boundary detection or object localization). Segmentation subdivides an image into its constituent parts. Segmentation algorithms for monochrome images generally are based on one of two basic properties of grey-level values: discontinuity and similarity. The first method partitions an image based on abrupt changes in grey-level (isolated points, lines and edges) while the second method is based on thresholding, region growing and region splitting and merging [31].

In both of these methods, one employs only information present in image. The problem with these methods is that the edges found do not necessarily correspond to boundaries of objects. With the exception of high-quality images from controlled environments, these methods produce spurious edges and gaps. They are of limited use in general and of no use in poor-quality images. The limitation of these methods is due to their complete reliance on information contained in the local neighborhood of pixels in the image. They ignore both model-based information and higher order organization of the image. Another problem associated with these methods is edge grouping. After extracting edges from the image, they have to be grouped in order to determine boundaries. This is usually done by first associating edge elements into edge segments and then by associating segments into boundaries. These methods often resort to arbitrary interpolation in order to complete boundary gaps. Further, it is often difficult to identify and classify spurious edges. We might consider these methods as low-level methods.

Alternatively, one may use high-level segmentation methods, high-level meaning that a priori knowledge about shape, texture, color, or position of the object in question is somehow included in the search procedure. In this work, a group of high-level methods, the so called active shape models, is investigated. Active shape models encompass several approaches to object segmentation: snakes, deformable templates and dynamic contours. Before we start a more formal mathematical anal-

ysis of those methods, lets to try get an intuitive understanding of the idea behind them.

## 2.1 What is a Snake?

Snake is a parametric curve defined within domain of an image. All snake properties and its behavior is specified through a function called energy functional by analogy with physical systems. A partial differential equation controlling the snake causes it to evolve so as to reduce its energy. The physical analogy can be extended, and the motion of the snake can be viewed as being due to simulated forces acting upon it. In order to gain an intuitive understanding of the snake model, it is therefore suitable to compare it with a real physical model.

Let us view the edge map of the image as a landscape on which the parametric curve can slither. A force is acting upon the curve and it is moving across the landscape trying to reach the energy equilibrium. The model driving it across the landscape has two components: one telling the snake how to behave (to preserve the original shape, to develop a corner etc.), another one instructing it where to go. We want the curve to cling to the boundary of a specific object in the scene. The boundary can be recognized as low values of the negative edge map, so the equilibrium equation should be set up in such a way that the curve tends to minimize the term involving the negative edge map, or as we will denote it - the potential energy of the dynamic system. Since we know the general shape of the object in question, we may design the evolution equation in such a way that the snake easily can embrace the object; it may have to be elastic, stiff, be able to develop a corner or similar.

Now, as we understand the idea, we have to design the system mathematically in order to be able to implement it. The first snake model was proposed by Kass [22] in 1987. The energy functional which the snake was to minimize in order to achieve equilibrium was defined as following

$$E_{snake} = \int_0^1 \{E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s))\} ds \quad (2.1)$$

where the position of the snake on the image is represented parametrically by a planar curve  $\mathbf{v}(s) = (x(s), y(s))$ ,  $E_{int}$  represents the internal energy of the curve due to the bending and the  $E_{image}$  represents the image forces pushing the snake

toward the desired object. The proposed internal energy model was defined as

$$E_{int} = (\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2)/2, \quad s \in [0, 1] \quad (2.2)$$

where  $\mathbf{v}_s(s)$  is the first derivative and  $\mathbf{v}_{ss}(s)$  the second derivative of  $\mathbf{v}(s)$  with respect to  $s$ . Note that we assume continuous image and curve coordinates. In applications, we work with digital images and a discretization must be formulated.

Since the object in interest should be recognized by the snake as a set of low values on the negative edge map, i.e. spatial gradient magnitude, the model for the image energy was defined as

$$E_{ext} = -|\nabla I(x, y)|^2 \quad (2.3)$$

if the object was homogeneous inside (both the boundary and the area inside the boundary have approximately the same grey level) or

$$E_{ext} = -I(x, y) \quad (2.4)$$

if the image is a line drawing (black on white). The term  $I(x, y)$  represents the grey level values of the image.

Lets now try to analyze it by viewing it in terms of our intuitive landscape model. The first derivative of  $\mathbf{v}(s)$  with respect to  $s$  gives us the rate of change of length of the curve, which means the longitudinal contraction of the curve. The coefficient  $\alpha(s)$  allows the curve to have smaller or larger degree of contraction and therefore makes the snake act like an elastic string. Large values of  $\alpha(s)$  mean large contraction of the snake in the direction of the force. Therefore is  $\alpha(s)$  denoted as the elasticity coefficient.

The second derivative gives us the rate of convexity or the curvature. The coefficient  $\beta(s)$  regulates than the rate of the change of the curve in the direction normal to its boundary. This term makes the snake act like a rigid string. That means that the curve preserves the smoothness, the straight -line shape but does not contract. If the value of  $\beta(s)$  is high the curve is hard and resists bending, while small values of  $\beta(s)$  is small allow the curve to develop a corner. By adjusting these two coefficients, the curve gets an appropriate elasticity and is able to embrace the object of interest.

The second energy term is easy to interpret. We can regard the image intensity function as a landscape and the snake is rolling down to a valley driven by a gravity

alike force . If we think of the low values of the negative edge map as the valley, calculate the edge map over the image and make the snake go in the direction of the minima on the edge map, it will roll to the valley and stay there.

Having defined the various energy terms that derive the snake, the initial position must be interactively specified by the user.

Those are the principles behind the snake. It will be shown later that this model may be improved in many ways to be able to estimate a variety of complex object.

## **2.2 What is a Deformable Template?**

Deformable templates constitute another important approach to object estimation. The mathematical background of the deformable templates can be traced back to the shape class description based on pattern theory [32].

This approach employs prior knowledge about the shape of the object in a direct manner. This prior shape information is specified as a sketch, binary template or a parametric prototype. The a priori information is then encoded either in the form of edge information from a binary template or the parameter vector. That information does not need to be exact in the sense that it matches the boundaries of the image exactly.

We may say that the difference between snakes and deformable templates is that snakes are form-free energy minimizing functions. In snakes model, there is no global structure of the curve except for some general regularization constraints such as continuity and smoothness of the boundary. On the other hand, parametric deformable templates control deformations using a set of parameters which are capable of encoding a specific shape. This type of model is used when more specific shape information is available, which can be described either by a binary template or a set of parameters.

The prototype template describes one and the most likely instance of the object shape. We apply a parametric transformation to the prototype and deform its boundaries varying the deformation parameters in order to capture a large variety of possible instances. If the object in interest is of biological nature, the form will resemble, but still vary from individual to individual. Those small variations may be captured by the random deformations of the prototype, so that a deformed template may match the object in interest better than the original template. Objects may also be noise corrupted or degraded in some way so that the original shape is lost. In that

case too a deformed template may match the object better than the original one.

Using an appropriate edge detector [48], we extract boundaries of the objects in the image. Then, we match all found objects with our template base and check for similar objects by aligning the templates in the data base with the image in question using some potential energy function.

We can simplify and improve detection further by imposing a probability distribution on the images in the data base. Since not all transformations result in templates that visually resemble the prototype template, and all the prior shape information is represented in the prototype template, it is natural to assume that the prototype template is the most likely a priori shape of the object. Further, small deformations that leave the template similar to its original shape are more likely than larger displacements. Therefore, we impose a probability distribution on the images in data base to bias the possible deformed templates. Varying parameters of the distribution, we adjust the confidence about our prototype template.

Having understood the idea behind deformable templates, the distinction between low- and high-level methods becomes clear and we see how did they get their respective names. In order to project a shape found in an image onto the shape base, we have to extract that shape from the image. For this purpose are low-level methods suitable. We might say that they operate on a lower level doing the rough part of the work - extracting objects - while the high-level methods are analyzing the results and making a decision.

It will be shown that there are many ways to generate a prototype template, deform it and make a decision as to which template resembles the object most.

### **2.3 What is a Dynamic Contour?**

Dynamic contours will not be discussed in this work but a short description will be given [6], [41]. It is also important to mention that both snakes and deformable templates may be labeled as dynamic contours because they show dynamic behavior. In this work we will, however, reserve this label for a specific group of active contours.

Active contours can be applied either statically, to single images, or dynamically to image sequences. In dynamical applications, some additional moments may be incorporated in the model to convey any prior knowledge about object motions and deformation. As opposed to snake where only the the active contour is varying,

in dynamic contours the edge map is varying too and the snake is applied on a sequence of images. The equation of motion for such a system extends from the snake model to a new model with additional terms governing inertia and viscosity.

This group of active contours find their applications in motion tracking, traffic monitoring, visual speech recognition and so on.

## **2.4 The Objective of the Work**

The underlying application domain for this work is medical imaging. Medical imaging allows scientists to observe anatomic structures inside the human body in a non-invasive way. The technological advances within medical imaging and medical image analysis have had a great impact on the medicine, both within clinical and research applications. It has expanded beyond simple visualization of anatomic structures and become a tool in surgical planning and simulation, intra-operative navigation, diagnosis, tracking the progress of a disease etc. As a tool in the medical research, medical imaging has for example become an irreplaceable aid in brain research applications such as cognitive psychology and morphological analysis of brain structures.

As it was mentioned in the introductory part of this chapter, the problem of extracting information from images is the most important and challenging problem in the image analysis. Modern medical imaging devices such as PET, CT and especially MR, provide visually satisfactory images of the internal human organs. Problems arise when we want to utilize computers in order to automate the extraction of information from an image set. In order to support a wide range of biomedical investigations, that frequently require large sets of image data in order to reach a reliable conclusion, there is a need for a fully automated computer aided methods for information extracting. The computer programs that implement these methods should be able to operate on different types of images with different objects of interest. In addition, it may be required that the program is operative by personnel with no profound technical or medical training. This means that the interaction between the user and the computer interface should be minimal.

As it is shown throughout this work, many active contour models give satisfactory results in some specific situations but fail to meet the requirements mentioned above. The contours may be too rigid and therefore unable to capture complexity of anatomic structures or they may be designed for a specific problem and can not

be used on a variety of shapes. Another problem which arises in some models is a time consuming preprocessing of the images required to create a model capable of detecting a specific structure. It is often required to study a so called training set which is a set of images containing the object which is to be segmented. The training set is used to collect statistical information about the object. This information is then integrated in the search model. In some models, collecting the information requires correctly placing control points on the shapes in the training set. This is a task for someone familiar with the application to choose the most appropriate set of points and to be able to reproducibly place them on different examples. In addition to being time consuming, this process can lead to a model which is incapable of capturing the variability of the shape if the control points are not placed correctly. Even though, these models give satisfactory results in some applications, they are not well suited for large scale automated processing of images in clinical applications.

In this work we will take a look at the representatives of different classes of active contour models, study them theoretically, then choose those which are may be applicable in the applications mentioned above and study them closer. What we will try to determine is how “good” different active contour models are for medical image segmentation. We will also classify these methods according to several different criteria.

Another original contribution of this work is the Fourier spectral algorithm for iterating the evolution equation, which has not been used in the literature.

### 3 Classification

Active contour models can be classified according to several different criteria.

One of the classifications is based on the “flexibility” of the active contour in was proposed in a slightly modified version by Jain [17]. The active contour models proposed in the literature can be accordingly partitioned in two classes

- Free form active contour models
- Limited form active contour models

In the free form active contour models, there is no global structure of the contour; it is constrained only by local continuity and smoothness constraints. The snakes mentioned in the previous chapter are free form active contours [30], [6], [12], [13], [49], [16], [20], [22], [28], [33] [42], [45]. These models do not use a priori information about the shape in a direct manner. This information is on the other hand used in adjusting the model parameters so that the contour possesses properties (elasticity, rigidity, etc.) that enable it to embrace the object of interest.

The limited form active contour models use a priori information about the geometrical shape directly. This information is available in the form of a sketch or a parameter vector that encodes the shape of interest. The geometric shape of the contour is adjusted by varying the parameters. Deformable templates from the introduction are limited form active contour models [4], [6], [10], [15] [17], [18], [23], [24], [25], [26], [27], [37], [47]. The reason that we call these models limited is the fact that they can not take any arbitrary shape. The variability of the shape is limited by the prototype template. parameterization and the way we generate deformed templates. We will take a closer look at both free form and limited form active contours.

Another criteria for classification can be the way the image information is utilized to align the active contour with the object of interest. The models can be accordingly classified into

- Region-based models
- Boundary-based models

Region based models derive a contour representation from the segmentation of the image into well defined regions. The image is examined point-wise in order to decide if a pixel belongs inside an object, outside all objects or at the object boundary.

A pixel belongs to the boundary if it is in the object region and has neighbors in the background. This segmentation is then used to produce an image force field which aligns the active contour with object of interest.

Edge-based methods use a continuous approximation of the original image intensity function so that the boundary can be characterized by a differential property. By point-wise examination of the image, a pixel is said to belong to the boundary if it is a local maximum of the image gradient. In this boundary detection process, the fact that these boundary points constitute a closed geometric contour is not taken into account. In Figures these two criteria are illustrated, where arrows represent the force field which attracts the contour to the object boundary.

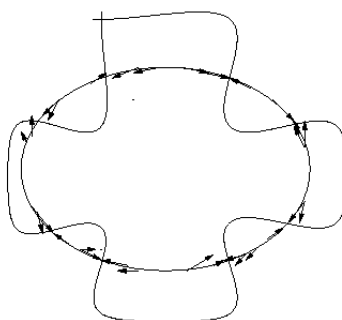


Figure 1: Edge-based external forces acting on the contour



Figure 2: Region-based external forces acting on the contour

The methods in [15], [23] and [42] are region-based methods. The rest of the methods in the literature are edge-based.

Image force field is easily computed for the edge based methods from a potential energy function. In region-based models, it is more complicated. One can not use

the potential function since it only defines boundaries and is therefore not a function in the image plane which is required for the image segmentation.

Because of the limited scope of this work, we will only study edge-based models.

Yet another classification partitions the active contours into

- Parametric active contours
- Non-parametric active contours

This classification can be ambiguous in some cases. A parametric active contour is a contour that is represented by a small number of parameters that capture the shape of the object. If the parameterization is achieved by expressing the curve in terms of a basis, where the discrete function representing the curve is expressed as a weighted sum of a set of known functions, distinction between parametric and non-parametric contours is clear. First, there is a parameter space different from the physical space where the curve is initially defined. Second, different bases give parameter spaces with different properties and third, some of the operations that are performed on the contour can be defined in the parameter space. Examples of such parameterizations are Fourier, B-spline and Wavelet representation.

If on the other hand, the parameters are some characteristic points on the contour in the physical space, as it is the case in the Point Distribution Model explained later, there is no distinct parameter space. In addition, the parameter vector is just a set of points of the contour which are sufficient to describe the shape of the curve. This set can be extended to contain all contour points available if the shape is very complex. In this case, there is no clear distinction between the parametric and non-parametric representation.

In this work, both of these classes are investigated.

Another classification can be made based on the model definition framework

- Physical active contours
- Statistical active contours

These two classes are actually equivalent and they differ in interpretation of the terms in the model. There are, however, some differences and they will be made clear later in the work.

In the next chapter, we will take a look at the parametric representation of active contours.

## 4 Representation of Parametric Deformable Contours

A parametric deformable contour is constructed as a weighted sum of some bases function and the expansion coefficients constitute the parameter vector. From many basis to choose, Fourier and B-Spline basis have been most frequently used in computer vision. Each parameterization has particular properties that make it suitable for different purposes. Some restrictions that may be comprised within some basis representations (such as smoothness or continuity) can be convenient because the necessary constraints are build directly into the representation and they do not have to be regularized subsequently. It is also desirable for the parameterization to be concise that determines the complexity of the optimization process.

Before considering any of these bases, lets define mathematically some of the terms we have used so far and introduce an appropriate notation.

### 4.1 The Choice of Curve Representation

There are three ways of expressing a curve which are normally used. These are the following:

1. explicit :  $y = f(x)$
2. implicit :  $f(x, y) = 0$
3. parametric :  $x = x(t), y = y(t)$

The difficulties associated with (i) are that it is impossible to get multiple values of  $y$  for a single  $x$ , so that closed curves such as circles and ellipses, must be represented by multiple curve segments. Further, it is difficult to represent a curve with the vertical tangent, since a slope of infinity is difficult to represent. Since the curve representing the snake often has both of those properties, the explicit representation is not well suited for it.

The problems with the implicit representation are the following: the equation may have more than one solution. For example, the implicit equation of a circle is  $x^2 + y^2 = 1$ . Should we want to model a half circle, we would have to add constraints such as  $x \geq 0$  which can not be contained within the explicit equation. Furthermore, if two implicitly defined segments are joined together, it may be difficult to determine if their tangent directions agree at the point.

The parametric representation for curves  $x = x(t), y = y(t)$  overcomes the problems caused by implicit or explicit forms. We can get multiple values of  $y$  for a single  $x$  on the plot since we do not plot  $y$  against  $x$  but we plot both  $x$  and  $y$  against a third variable  $t$ , which is not shown on the plot. The parametric curve form solves the slope of infinity problem too, since it replaces the use of geometric slopes with parametric tangent vector which is never infinite.

## 4.2 Definition of the Parametric Curve Representing the Snake

Let  $\theta \in \Theta$  denote the parameter vector of a closed contour where  $\Theta$  is the space of all configurations and deformations of the closed curves. The number of elements in the parameterization vector is the order of parameterization. In Fourier representation it is the number of Fourier descriptors (coefficients), in Spline representation it is the number of control points. The curve represented by such a vector is a continuous, periodic, vector function  $\mathbf{v}(s) = [x(s), y(s)]$ , of period  $L$  (in Fourier representation  $L = 2\pi$ , in B-Spline representation  $L$  is an arbitrary real number); its  $N$ -point discrete version is a  $(N \times 2)$  vector

$$\mathbf{v} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{N-1} \end{pmatrix} = \begin{pmatrix} x_0 & y_0 \\ x_1 & y_1 \\ \vdots & \vdots \\ x_{N-1} & y_{N-1} \end{pmatrix} \quad (4.1)$$

where  $\mathbf{v}_i = \mathbf{v}[iL/(N - 1)]$  for  $i = 0, 1, \dots, N - 1$

Now, given a parameter vector, we construct a deterministic operator  $\mathcal{V}$ , defined on  $\Theta$ , that maps  $\theta$  into  $\mathbf{v}$ , i.e. we write  $\mathbf{v} = \mathcal{V}\theta$ .

## 4.3 Arc Length as the Curve Parameter

A single parametric curve can be represented by more than one vector function depending on the choice of the parameter.

Suppose that  $C$  is a piecewise-smooth curve given by a vector function

$$\mathbf{v}(t) = [x(t), y(t)] = x(t)\mathbf{i} + y(t)\mathbf{j} \quad (4.2)$$

where  $\mathbf{i}$  and  $\mathbf{j}$  are unit vectors along the  $x$  and  $y$  axis, respectively. We define the arc

length function of  $C$  by

$$\begin{aligned} l(s) &= \int_a^s |\mathbf{v}'(u)| du \\ &= \int_a^s \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2} du \end{aligned} \tag{4.3}$$

where  $u$  is a dummy integration variable.

Thus,  $l(s)$  is the length of the part of  $C$  between  $\mathbf{v}(a)$  and  $\mathbf{v}(s)$ . It is often useful to parameterize a curve with respect to arc length, because arc length length arises naturally from the shape of the curve and does not depend on particular coordinate system.

## 4.4 Spline Basis

One of the most frequently used curve representations in the computer vision is the B-spline representation. It is important to distinguish between splines and B-splines. B-splines are polynomial functions with minimal support. Splines are linear combinations of B-splines. In the literature, splines are usually defined as a divided difference of a truncated power function. For computational purposes, it is more convenient to define splines via the recurrence relation [14], [21],[11].

### 4.4.1 B-Splines

The starting point for all splines is a nondecreasing knot sequence  $\mathbf{s} = (s_i)$ . B-splines of order 0 are the characteristic functions of this partition

$$B_{i,0}(s) := \begin{cases} 1, & \text{if } s_i \leq s < s_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

where  $:=$  denotes “equality by definition”. The only constraint is that these B-splines should form a partition of unity, i.e.,

$$\sum_i B_{i,0}(s) = 1, \text{ for all } s \tag{4.5}$$

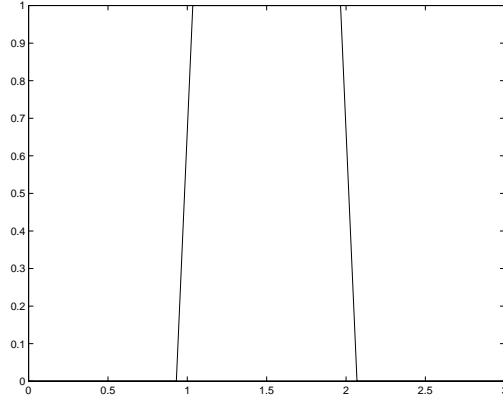


Figure 3: First order B-spline

In particular

$$s_i = s_{i+1} \text{ implies } B_{i,0} = 0 \quad (4.6)$$

From these first-order B-splines we obtain higher-order B-splines by recurrence:

$$B_{i,k} := \omega_{i,k} B_{i,k-1} + (1 - \omega_{i+1,k}) B_{i+1,k-1} \quad (4.7)$$

with

$$\omega_{i,k} := \begin{cases} \frac{s - s_i}{s_{i+k-1} - s_i}, & \text{if } s_i \neq s_{i+k-1} \\ 0, & \text{otherwise} \end{cases} \quad (4.8)$$

Thus, the second-order B-spline is given by

$$B_{i,1} = \omega_{i,2} B_{i,0} + (1 - \omega_{i+1,2}) B_{i+1,0} \quad (4.9)$$

and so consists in general of two nontrivial linear pieces which join continuously to form a piecewise linear function which vanishes outside the interval  $[s_i, s_{i+2}]$ .

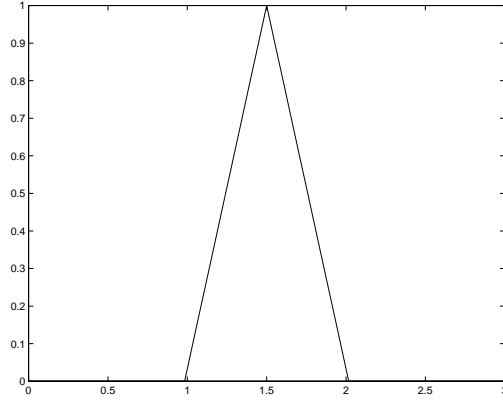


Figure 4: Second order B-spline

The third order B-spline is given by

$$\begin{aligned}
 B_{i,3} &= \omega_{i,3}B_{i,2} + (1 - \omega_{i+1,3})B_{i+1,2} \\
 &= \omega_{i,3}\omega_{i,2} + (\omega_{i,3}(1 - \omega_{i+1,2})(1 - \omega_{i+1,3})\omega_{i+1,2})B_{i+1,0} + \\
 &\quad + (1 - \omega_{i+1,3})(1 - \omega_{i+2,2})B_{i+2,0}
 \end{aligned} \tag{4.10}$$

So,  $B_{i,3}$  consists of 3 (nontrivial) quadratic pieces.

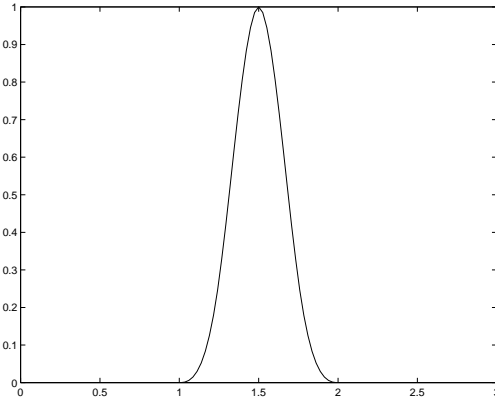


Figure 5: Third order B-spline

After  $k - 1$  steps of the recurrence, we obtain  $B_{ik}$  in the form

$$B_{i,k} = \sum_{j=1}^{i+k-1} b_{j,k} B_{j,0} \tag{4.11}$$

with each  $b_{jk}$  polynomial of degree  $< k$  since it is the sum of products of  $k - 1$  polynomials.

Many properties of B-splines are derived most easily by considering not just one B-spline but the linear span of all B-splines of given order  $k$  for a given knot sequence  $\mathbf{s}$  and this brings us to splines.

#### 4.4.2 Splines

A spline of order  $k$  with knot sequence  $\mathbf{s}$  is, by definition, a linear combination of the B-splines  $B_{i,k}$  associated with that knot sequence. We denote by

$$S_{k,\mathbf{s}} := \sum_i B_{i,k} a_i : a_i \in \mathbf{R} \quad (4.12)$$

the collection of all such splines. We will pay special attention to the following knot sequence

$$\mathbf{Z} := (\dots, -2, -1, 0, 1, 2, \dots) \quad (4.13)$$

which is the set of all integers.

The spline associated with this knot sequence is called a cardinal spline. Because of the uniformity of the knot sequence  $\mathbf{Z}$ , formulae involving cardinal B-splines are often much simpler than corresponding formulae for general B-splines. To begin with, all cardinal B-splines (of a given order) are translates of one another. With the natural indexing  $s_i := i$ , for all  $i$ , we have

$$N_k := N_k(s) := B_{0,k} \quad (4.14)$$

and

$$B_{i,k} = N_k(s - i) \quad (4.15)$$

The recurrence relation (9) simplifies as follows

$$(k - 1)N_k(s) = sN_{k-1}(s) + (k - s)N_{k-1}(s - 1) \quad (4.16)$$

#### 4.4.3 Cubic B-Splines and Polygon Approximation

Now, we will take a closer look at the B-spline theory that is relevant for our snake applications.

The reason for choosing B-splines and not so called natural splines which we are familiar with from the classical interpolation theory, is the so called local control

property of the B-splines. The polynomial coefficients of the natural splines are dependent on all  $N$  control points. Their calculation involves inverting a  $(N + 1) \times (N + 1)$  matrix. This has two disadvantages: moving one control point affects the entire curve and the computation time needed to invert the matrix can interfere with rapid interactive reshaping of a curve. B-splines, on the other hand, consist of curve segments depending on just a few control points. This is called the local control. Thus, moving a control point affects only a small part of the curve. B-splines have the same continuity as the natural splines but do not interpolate their control points. Therefore, we speak of polygon approximation and not of interpolation of control points.

The first step would be to choose the order of the basis splines in order to achieve the desired smoothness and still maintain reasonable computational efficiency. We choose cubic splines, which means the splines of order 3:  $B_{i,3}(t)$ . The reasons are following:

1. The lower-degree polynomials give too little flexibility in controlling the shape of the curve. The 1. order splines (straight lines) do not give satisfactory smoothness of the approximating curve. The 2. order splines (quadratic curves) give a smooth curve but a problem arises at the points where different curve segments join. In order to understand this problem we introduce a new criterion:

**Definition 1** *Let  $Q_i(s)$  denote a curve segment. If the direction and the magnitude of  $\frac{d^n}{dt^n}Q_i(s)$  and  $\frac{d^n}{dt^n}Q_{i+1}(s)$  are equal at the join point, the curve consisting of these two segments is called  $C^n$  continuous.*

The 2. order splines are  $C^0$  and  $C^1$  continuous, which does not insure a satisfactory continuity at the joint points. The problem is solved by using cubic splines which are  $C^0, C^1$  and  $C^2$  continuous.

2. The higher-degree polynomials are more time-consuming in the computational process and can introduce unwanted wiggles. The curve might “wobble” back and forth in ways that are difficult to control.
3. We say that cubic splines give “satisfactory” continuity because it is found that the eye can not detect a geometric discontinuity of degree higher than two and it is sufficient in practice to consider splines of degree three.

In the rest of the chapter we will introduce the mathematical formalism for the snake curve as a weighted sum of B-splines.

Parametric spline curves

$$\mathbf{v}[s] = (x[s], y[s]) \quad (4.17)$$

have coordinates  $x[s]$  and  $y[s]$ , each of which is a spline function of the curve parameter  $s$ ,  $0 \leq s \leq 1$ . Since  $B_{i,4}$ , the cardinal cubic B-spline, has been used in the snake model throughout the literature, we will from now on denote it as  $B_i$  and use it in further work. We will now define a spline curve in the plane representing the snake.

For each basis function  $B_i$  a control point  $\mathbf{q}_i = [q_i^x, q_i^y]$  is defined and the snake curve is a weighted sum of control points

$$\mathbf{v}[s] = \sum_{i=0}^N B_i[s] \mathbf{q}_i \quad (4.18)$$

which becomes a smooth curve that follows approximately “the control polygon” defined by linking control points by lines.

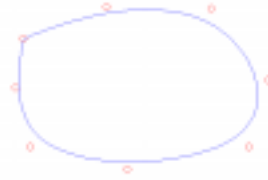


Figure 6: Potential field of the image in Fig

Now we introduce a more compact notation for the the control points by defining a space of control vectors  $K_Q$  consisting of

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}^x \\ \mathbf{Q}^y \end{pmatrix} \quad \text{where} \quad \mathbf{Q}^x = \begin{pmatrix} q_0^x \\ \vdots \\ q_{N-1}^x \end{pmatrix}^T \quad (4.19)$$

Then the coordinate functions can be written as

$$x[s] = \mathbf{B}[s]^T \mathbf{Q}^x \quad (4.20)$$

and

$$y[s] = \mathbf{B}[s]^T \mathbf{Q}^y \quad (4.21)$$

where  $\mathbf{B}[s] = (B_0[s], B_1[s], \dots, B_{N-1}[s])^T$  is a vector of B-spline functions. The parametric curve  $\mathbf{v}[s]$  becomes then

$$\mathbf{v}[s] = U[s] \mathbf{Q} \text{ for } 0 \leq s \leq 1 \quad (4.22)$$

where

$$U[s] = I_2 \otimes \mathbf{B}[s]^T = \begin{pmatrix} \mathbf{B}[s]^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}[s]^T \end{pmatrix} \quad (4.23)$$

where  $\otimes$  denotes the Kronecker product.

Comparing this result with our definition of the snake curve in the section 4.2, we see that we can interpret the set of control points defining the spline as the parameter vector

$$\theta = [\mathbf{q}_0^T \mathbf{q}_1^T \cdots \mathbf{q}_2^T] \quad (4.24)$$

describing the shape of the curve. The matrix operator  $\mathcal{V}$  is in this case

$$\mathbf{v} = \mathcal{V}\theta = \mathbf{B}\theta \quad (4.25)$$

where the elements of the matrix  $\mathbf{B}$  are given by  $[B]_{ij} = B_i(2\pi j/(N-1))$ .

We can rewrite the expression (4.18) further. Restricting  $B_i[s]$  to the interval  $[s_i, s_{i+1}]$ , we can identify it with a polynomial in  $s$  of degree  $\leq k$ . We will always assume in this section that the interval  $[s_i, s_{i+1}]$  is in fact  $[0, 1]$ ; this can be obtained with an affine change of variable. Each  $B_i[s]$  is a polynomial in  $s$  which may be written as its Taylor series at 0. We may therefore write

$$(B_{i-k}[s], \dots, B_i[s]) = (1, s, \dots, s^k) \mathbf{M} \quad (4.26)$$

where  $\mathbf{M}$  is a  $(k+1) \times (k+1)$  matrix. The curve  $\mathbf{v}$  is thus represented in the

following matrix form

$$\begin{aligned} x[s] &= (1, s, \dots, s^k) \mathbf{M} \begin{pmatrix} q_0^x \\ \vdots \\ q_{N-1}^x \end{pmatrix} \quad \text{and} \\ y[s] &= (1, s, \dots, s^k) \mathbf{M} \begin{pmatrix} q_0^y \\ \vdots \\ q_{N-1}^y \end{pmatrix} \end{aligned} \quad (4.27)$$

Now let us assume that we have a set of control points  $(x_i, y_i)$ . First we parameterize the set:

$$x[s_i] = x_i \quad y[s_i] = t_i; \quad (4.28)$$

where  $s_i, 0 \leq i \leq N - 1$  are evenly spaced points - integers - on the real axis. We parameterize the curve for the reasons explained in the beginning of the chapter, and knots are chosen to be evenly spaced since it leads to cardinal splines which are easier and less computational costly to generate than the non-uniform splines.

Next, we generate the splines using the recurrence formulae (4.7). We notice that each spline of order 1 is locally controlled by three knots, splines of order 2 are locally controlled by four knots and splines of order 3 are controlled by five knots. We begin by computing the expression of  $B_3[s]$  on each interval  $[s_i, s_{i+1}]$ ,  $0 \leq i \leq 3$  and the knots are 0, 1, 2, 3, 4.

$$\begin{aligned} B_3[s] &= B_{0,3}[s] = \frac{1}{3}(sB_{0,2}[s] + (4-s)B_{1,2}[s]) \\ B_{0,2}[s] &= \frac{1}{2}(sB_{0,1}[s] + (3-s)B_{1,1}[s]) \\ B_{1,2}[s] &= \frac{1}{2}((s-1)B_{1,1}[s] + (4-s)B_{2,1}[s]) \\ B_{0,1}[s] &= sB_{0,0}[s] + (2-s)B_{1,0}[s] \\ B_{1,1}[s] &= (s-1)B_{1,0}[s] + (3-s)B_{2,0}[s] \\ B_{2,1}[s] &= (s-2)B_{2,0}[s] + (4-s)B_{3,0}[s] \end{aligned} \quad (4.29)$$

which gives

$$B[s] = \begin{cases} \frac{1}{6}s^3, & \text{for } 0 \leq s < 1 \\ \frac{1}{6}(-3s^3 + 12s^2 - 12s + 4), & \text{for } 1 \leq s < 2 \\ \frac{1}{6}(3s^3 - 24s^2 + 60s - 44), & \text{for } 2 \leq s < 3 \\ \frac{1}{6}(4-s)^3 = \frac{1}{6}(-s^3 + 12s^2 - 48s + 64), & \text{for } 3 \leq s < 4 \end{cases} \quad (4.30)$$

On the interval  $[0, 1]$ , the function  $x[s]$  reduces than to (say for  $x[s]$  and correspondingly for  $y[s]$ )

$$\begin{aligned}
& B[s+3]q_{-3}^x + B[s+2]q_{-2}^x + B[s+1]q_{-1}^x + B[s]q_0^x \\
&= \frac{1}{6}[(1-s)^3 q_{-3}^x + (3(s+2)^3 - 24(s+2)^2 + 60(s+2) - 44)q_{-2}^x \\
&+ (-3(s+1)^3 + 12(s+1)^2 - 12(s+1) + 4)q_{-1}^x + s^3 q_0^x] \quad (4.31) \\
&= \frac{1}{6}[(-s^3 + 3s^2 - 3s + 1)q_{-3}^x + (3s^3 - 6s^2 + 4)q_{-2}^x \\
&+ (-5s^3 + 3s^2 + 3s + 1)q_{-1}^x + s^3 q_0^x]
\end{aligned}$$

and correspondingly for  $y[s]$ . It gives the following matrix representation

$$x[s] = \frac{1}{6}(1, s, s^2, s^3) \begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} q_{-3}^x \\ q_{-2}^x \\ q_{-1}^x \\ q_0^x \end{pmatrix} \quad (4.32)$$

The portion of the curve  $x[s]$  (and correspondingly  $y[s]$ ) corresponding to  $i \leq s + 1 \leq i + 1$  is written with the same matrix as

$$x[s] = \frac{1}{6}(1, s, s^2, s^3) \begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} q_{i-3}^x \\ q_{i-2}^x \\ q_{i-1}^x \\ q_i^x \end{pmatrix} \quad (4.33)$$

#### 4.4.4 Discussion on B-Splines as Basis for Snakes

When constructing a snake, it is convenient to chose a set of control points on the image and than generate the snake as an approximation to the polygon which arises by connecting the point with straight lines. The question is why to choose the spline representation?

- Splines are polynomial, and being polynomial they can be evaluated efficiently on computer.
- They are piecewise polynomial which makes them very flexible.
- They can be represented as a linear combination of B-splines and this representation provides geometric information and insight.

- The snake curve is closed which implies periodicity. This is easily achieved by treating the parameter  $t$  as periodic.
- In the snake model it is important that the curve is smooth. We impose constraints on the curve in the energy model in order to give it flexibility and smoothness. But the splines minimize the deformation-like energy so the search for the solution-curve may be confined to a set of such functions under the action of the image potential. The choice of the cubic-spline as basis-spline is a pragmatic approach. The linear spline does not give a smooth function since the first derivative of the function is not continuous. The cubic spline gives satisfactory smoothness while it still is computationally relatively efficient.
- Splines preserve the shape which means that a spline has the same shape as its control polygon, or more precisely: A spline crosses any straight line no more often than does its control polygon. In particular, if the control polygon is monotone (convex), then so is the spline.

All these properties make B-splines a very popular choice for curve representation. We will compare it with another parameterization which has been used to represent the active contour models, Fourier series representation.

## 4.5 Fourier Basis

In this chapter the Fourier series description of discrete-time periodic functions will be discussed. The Fourier series representation of a discrete time periodic function is a finite series. As a consequence, there are no mathematical issues of existence of convergence. What will be of interest to us is the speed of convergence since we will truncate the Fourier series to make computations more efficient [19],[31],[36],[38].

### 4.5.1 Fourier Series Representation of a Periodic Functions

A discrete time function  $x[n]$  is periodic with period  $N$  if

$$x[n] = x[n + N] \quad (4.34)$$

The fundamental period is the smallest positive integer  $N$  for which (4.40) holds, and  $\omega_0 = 2\pi/N$  is the fundamental frequency.

We want to expand the function in terms of a finite sequence of orthogonal functions

$$\phi_k[n] = e^{jk\omega_0 n}, \quad k = 0, \pm 1, \pm 2, \dots \quad (4.35)$$

It can be shown that such an expansion exists for all periodic functions  $x[n]$  and the conversion formulas are

$$x[n] = \sum_{k=0}^{N-1} a_k e^{jk\omega_0 n} \quad (\text{synthesis}) \quad (4.36)$$

$$a_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk\omega_0 n} \quad (\text{analysis}) \quad (4.37)$$

Discrete time complex exponentials  $e^{jk\omega_0 n}$  which differ in frequency by a multiple of  $2\pi$  are identical, i.e.

$$e^{jk\omega_0 n} = e^{jk(\omega_0 + m2\pi)n} = e^{jk\omega_0 n} \quad (4.38)$$

and so are the expansion coefficients in period which is a multiple of  $N$

$$a_k = a_{k+mN} \quad (4.39)$$

In this Fourier series representation, one set of  $N$  coefficients  $\{a_k\}$  is generated. There is, however, another representation which can be used for real periodic functions. It is actually a reformulation of the approach just described, where the fact that the function is real is used. Staib [47] used this alternative approach in his deformable template model.

#### 4.5.2 Alternative Fourier Series Representation of a Real Periodic Function

Suppose that  $x[n]$  is real and that it can be represented in the form

$$x[n] = \sum_{k=0}^{N-1} a_k e^{jk\omega_0 n} \quad (4.40)$$

The coefficient formula may be rewritten using Euler equation for the trigonometric form of the complex exponential in the following way

$$\begin{aligned}
a_k &= \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk\omega_0 n} \\
&= \frac{1}{N} \sum_{n=0}^{N-1} x[n] (\cos(k\omega_0 n) - j \sin(k\omega_0 n)) \\
&= \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cos(k\omega_0 n) - j \frac{1}{N} \sum_{n=0}^{N-1} x[n] \sin(k\omega_0 n) \\
&= A_k - jB_k
\end{aligned} \tag{4.41}$$

where

$$A_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cos(k\omega_0 n) \tag{4.42}$$

$$B_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \sin(k\omega_0 n) \tag{4.43}$$

The syntheses formula becomes

$$\begin{aligned}
x[n] &= \sum_{k=0}^{N-1} (A_k - jB_k) e^{jk\omega_0 n} \\
&= \sum_{k=0}^{N-1} (A_k - jB_k) (\cos(k\omega_0 n) + j \sin(k\omega_0 n)) \\
&= \sum_{k=0}^{N-1} (A_k \cos(k\omega_0 n) + B_k \sin(k\omega_0 n)) + j(A_k \sin(k\omega_0 n) - B_k \cos(k\omega_0 n))
\end{aligned} \tag{4.44}$$

Since the function  $x[n]$  is real, the imaginary part of the syntheses formula has to be equal 0. The formula (4.50) becomes then

$$x[n] = \sum_{k=0}^{N-1} (A_k \cos(k\omega_0 n) + B_k \sin(k\omega_0 n)) \tag{4.45}$$

In this representation, the parameter vector has  $2N$  elements while the formula (4.42) produces  $N$  parameters.

Through the literature both the original and the alternative Fourier series representation of a real periodic signal have been used to represent the snake curve.

### 4.5.3 The Speed of Convergence of the Fourier Series - Truncation of the Series

When dealing with a periodic function, we know that all frequency information needed to achieve the perfect reconstruction of the function is contained within the range of the fundamental frequency. But what if we want to reduce that information for computational purpose by truncating the series? How will that change the function.

We define the norm of a discrete periodic function  $x[n]$  with period  $N$  as

$$\|x\| = \left( \sum_{n=0}^{N-1} |x[n]|^2 \right)^{1/2} \quad (4.46)$$

and Parseval's identity

$$\|x\|^2 = N \sum_{k=0}^{N-1} |a_k|^2 \quad (4.47)$$

Denote the truncated Fourier series by

$$\hat{x}[n] = \sum_{k=0}^M a_k e^{jk\omega_0 n} \quad M < N - 1 \quad (4.48)$$

By Parseval's identity, we have

$$\|x - \hat{x}\| = \left( N \sum_{k=M+1}^N |a_k|^2 \right) \quad (4.49)$$

and if  $x$  is sufficiently smooth,

$$\max_{0 \leq n \leq N} |x[n] - \hat{x}[n]| \leq \sum_{k=M+1}^N |a_k|^2 \quad (4.50)$$

This shows that the size of the error created by replacing  $x$  with its  $M$ -th order truncated Fourier series depends upon how fast the Fourier coefficients of  $x$  decay to zero. This in turn depends on the regularity of  $x$  in the domain of  $[0, N - 1]$ . We say that a function is regular at a point in the domain if it is defined and differentiable at that point. It can be shown that if the function  $x[n]$  is periodic and  $m$ -times differentiable on the domain then the Fourier series coefficients decay as

$$a_k = O(k^{-m}) \quad (4.51)$$

From this we see that if function does not have sharp peaks or other abrupt changes which would lead to discontinuities of higher derivatives, we can use very few coefficients to describe the function without any visible information loss, because only few first coefficients would significantly differ from zero.

In the next section we will show how we can use the Fourier series representation to describe the snake curve and we will show on examples some things we stated here.

#### 4.5.4 Fourier Series Computational Algorithms

We will present two ways of calculating the Fourier series expansion for a snake curve. The first one reduces a two-dimensional problem (a plane curve) to a one dimensional problem, thereby reducing the number of calculations [31]. The second one uses the alternative Fourier series representation of a real periodic signal and exploits the connection between the expansion coefficients and the FFT algorithm [47].

Lets say we have a N-point digital boundary i  $xy$  plane as shown in Figure



Figure 7: A digital boundary

Starting at an arbitrary point  $(x_0, y_0)$ , coordinate pairs  $(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_{N-1}, y_{N-1})$  are encountered in traversing the boundary, say, clockwise. These coordinates can be expressed in the form  $x(k) = x_k$  and  $y(k) = y_k$  and the boundary itself  $s(k) = [x(k), y(k)]$ , for  $k = 0, 1, 2, \dots, N - 1$ . Moreover, each coordinate pair can treated as a complex number so that

$$s(k) = x(k) + iy(k) \tag{4.52}$$

This representation reduces a 2-D to a 1-D problem. The discrete Fourier transform of  $s(k)$  is

$$S(\omega) = \frac{1}{N} \sum_{k=0}^{N-1} s(k)e^{-i2\pi\omega k/N} \quad (4.53)$$

for  $\omega = 0, 1, 2, \dots, N - 1$ . The complex coefficients  $S(\omega)$  are often called the Fourier descriptors of the boundary. The inverse Fourier transform of the  $S(\omega)$  is

$$s(k) = \sum_{\omega=0}^{N-1} S(\omega)e^{i2\pi\omega k/N} \quad (4.54)$$

for  $k = 0, 1, 2, \dots, N - 1$ . Suppose, however that only the first  $M$  coefficients are used in the reconstruction, which is equivalent to setting  $S(\omega) = 0$  for  $\omega > M - 1$ . The results is the following approximation to  $s(k)$

$$\hat{s}(k) = \sum_{\omega=0}^{M-1} s(\omega)e^{i2\pi\omega k/N} \quad (4.55)$$

for  $k = 0, 1, 2, \dots, N - 1$ .

Although only  $M$  terms are used to obtain each component of  $\hat{s}(k)$ ,  $k$  still ranges from 0 to  $N - 1$ . The results of the truncation are shown in Figure 7.

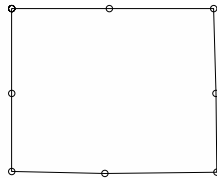


Figure 8: The curve before truncation

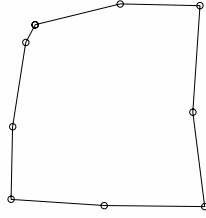


Figure 9: The curve after truncating two frequency coefficients

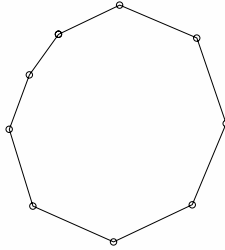


Figure 10: The curve after truncating half of the frequencies

Another approach to computing the Fourier series expansion coefficients is based on the alternative Fourier series representation discussed in section 4.5.2, where .

$$\begin{aligned}
 x[n] &= A_0^x + \sum_{k=1}^{N-1} (A_k^x \cos(k\omega_0 n) + B_k^x \sin(k\omega_0 n)) \\
 y[n] &= A_0^y + \sum_{k=1}^{N-1} (A_k^y \cos(k\omega_0 n) + B_k^y \sin(k\omega_0 n))
 \end{aligned}
 \tag{4.56}$$

In this case we do not convert the real two-dimensional problem in to a complex one-dimensional problem. We treat each of the coordinates separately and exploit the connection between the alternative Fourier series representation coefficients and the FFT (Fast Fourier Transform) coefficients. We start by computing the FFT coefficients for  $x[n]$  and  $y[n]$  which we denote by  $X[k]$  and  $Y[k]$ . It can be shown

that we can calculate the coefficients  $A_k^x, B_k^x, A_k^y, B_k^y$  by

$$\begin{aligned}
A_0^x &= 2X[1]/N \\
A_0^y &= 2Y[1]/N \\
A_k^x &= 2\mathcal{R}(X[k+1])/N \\
B_k^x &= 2\mathcal{I}(X[k+1])/N \\
A_k^y &= 2\mathcal{R}(Y[k+1])/N \\
B_k^y &= 2\mathcal{I}(Y[k+1])/N
\end{aligned} \tag{4.57}$$

where  $\mathcal{R}$  and  $\mathcal{I}$  represent the real and the imaginary part of the numbers.

The first algorithm reduces the number of computations but does not use FFT which speeds up computations. If the number of points is a power of two, the FFT algorithm can be used in the second approach and both methods turn out to use same computational time. If the number of points is not a power of two, the first algorithm uses less time for computation.

#### 4.5.5 Fourier Basis Properties and Snakes

The Fourier basis is well suited for representation of curves which do not have high gradients. The reason is that we may keep only few low-order coefficients in the truncation process and still capture gross shape. This makes the Fourier basis representation of the snake computationally efficient. Another desired property of the snake parameters is the insensitivity to translation rotation and scale changes. The Fourier coefficients are not insensitive to these geometrical changes of the curve but they can be related in a simple manner to the corresponding transformations on the coefficients.

<i>Transformation</i>	<i>Boundary</i>	<i>FourierCoeff.</i>
<i>Identity</i>	$s(k)$	$a(u)$
<i>Rotation</i>	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
<i>Translation</i>	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}$
<i>Scaling</i>	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$

Table 1:

#### 4.5.6 Comparison and Conclusion

Whether or not to use Fourier basis to describe the snake curve is directly related to the way we generate the curve in the first place.

One way of initializing the curve is to use a discrete version of some appropriate analytical function like a circle or an ellipse. We can then generate a curve with so many points as needed in order to get a smooth curve.

It is important to notice that in Fourier series representation of a curve, the new curve obtained by truncating the Fourier series of the original curve has the same number of points as the original curve. It contains fewer coefficients in the frequency space but the same number of points in the physical space. So if the polygon obtained by connecting the points of the original curve is approximately smooth, the new polygon will have the same degree of smoothness. On the other hand, if the curve is given by a small set of nodes which give an approximate shape of the curve (which is often the case), we have to interpolate the polygon in order to obtain a visually continuous curve. In the Fourier case, this has to be done additionally, while splines approximate the polygon as we showed in chapter.

It is also important to notice that B-splines can be used in two ways. One way is to generate a set of nodes, let the model on these nodes (parameters) and after each iteration approximate the polygon with a spline curve. This would be the parametric representation approach which has been used through the literature. It will be shown later that this parametric approach has some drawbacks. The shape of the object has to be well-defined and not too complex if it is to be described by few parameters. IN medical imaging, however, we often encounter shapes with great complexity. In such cases, it is necessary to use curves with high resolution to describe these shapes. It will also not suffice to let the model act on few characteristic points and approximate the rest. We have to iterate a large number of points in order to capture the objects complexity. In this case the B-splines are used as an interpolation method.



Figure 11: An active contour specified by a set of control points

When used as a parametric representation, the B-spline approach has an important advantage compared with Fourier representation. As we pointed out in section 4.4.4, the splines minimize deformation-like energy so that the term  $v_s(s)$  may be left out in the energy functional (2.1). If the shape of the object is regular and without sharp corners, we may represent it with very few coefficients in Fourier space (2-3).

Experiments using algorithms developed in this work have shown that parametric representation using both B-splines and Fourier descriptors give satisfactory results only in the case of simulated, regular objects with no noise. When tested on the midsagittal images of the human brain trying to segment corpus callosum, large number of parameters were required in order to get good results. The precision increased as the resolution of the active contour increased and it was exceptionally visible when images were degraded with Gaussian noise. The computational time increased too. We chose, however, the non parametric representation but used B-splines to approximate the node polygon with a continuous curve.

## 5 The Image Energy

In the partial differential equation which models the snake movement, the gradient of the image energy is the driving force and we may say that the segmentation result depends most on how good the image energy is defined.

Before we start discussion on different image energy models lets introduce some of the terms we will be using throughout the chapter.

## 5.1 Some Relevant Definitions from Vector Calculus

here follow some definitions form the vector calculus [35].

**Definition 2** *If  $f$  is a scalar function of two variables, its gradient is defined by*

$$\nabla f(x, y) = f_x(x, y)\mathbf{i} + f_y(x, y)\mathbf{j} \quad (5.1)$$

where  $\mathbf{i}$  and  $\mathbf{j}$  are unit vectors along  $x$ - and  $y$  axis, while  $f_x(x, y)$  and  $f_y(x, y)$  are the partial derivatives of  $f$  with respect to  $x$  and  $y$ .

Therefore,  $\nabla f(x, y)$  is a vector field and is called a gradient vector field.

**Definition 3** *A vector field  $\mathbf{F}$  is a conservative vector field if it is the gradient of some scalar function, that is if there exists a function  $f$  such that  $\mathbf{F} = \nabla f$ . Then,  $f$  is called the potential function for  $\mathbf{F}$ .*

**Definition 4** *In physics, the potential energy of an object in the point  $(x, y)$  of a conservative force field  $\mathbf{F}$  is defined as*

$$P(x, y) = -f(x, y, z) \quad (5.2)$$

where  $\mathbf{F} = \nabla f$ .

**Definition 5** *If  $\mathbf{F} = P\mathbf{i} + Q\mathbf{j}$  is a vector field and the partial derivatives of  $P$  and  $Q$  exist, then the curl of  $\mathbf{F}$  is the vector field defined by*

$$\text{curl}\{\mathbf{F}\} = \nabla \times \mathbf{F} \quad (5.3)$$

where  $\nabla$  is the vector differential operator defined as

$$\nabla = \mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} \quad (5.4)$$

and  $\times$  is the cross product.

**Definition 6** *If  $\mathbf{F} = P\mathbf{i} + Q\mathbf{j}$  is a vector field and the partial derivatives of  $P$  and  $Q$  exist, then the divergence of  $\mathbf{F}$  is the vector field defined by*

$$\text{div}\{\mathbf{F}\} = \nabla \cdot \mathbf{F} \quad (5.5)$$

Geometrically is curl a measure of how much the vector field  $\mathbf{F}$  curls around the point in question, and divergence is a measure of how much the vector field  $\mathbf{F}$  spreads out from the point in question.

If  $\text{curl}\{\mathbf{F}\} = 0$  at point  $P$ , then  $\mathbf{F}$  is called curl-less or irrotational. If  $\text{div}\{\mathbf{F}\} = 0$  at point  $P$ , then  $\mathbf{F}$  is called divergence-less or solenoidal.

## 5.2 Image Energy Models

We will now see how the different energy models work on an example, but before we start lets introduce a alternative representation of the Euler evolution equation. We remember that the snake that minimizes the energy must satisfy the equation

$$\alpha \mathbf{v}_{ss}(s) - \beta \mathbf{v}_{ssss}(s) - \nabla E_{image} = 0 \quad (5.6)$$

This can be viewed as a force balance equation

$$\mathbf{F}_{int} + \mathbf{F}_{image} = 0 \quad (5.7)$$

where  $\mathbf{F}_{int} = \alpha \mathbf{v}_{ss}(s) - \beta \mathbf{v}_{ssss}(s)$  and  $\mathbf{F}_{image} = -\nabla E_{image}$ . The internal force  $\mathbf{F}_{int}$  prevents stretching and bending while the external force  $\mathbf{F}_{image}$  pulls the snake toward the desired image edges. This model of snake equation is well-suited for theoretical studies.

### 5.2.1 The Original Force Field

In the original work by Kass [22], the image energy was defined as

$$E_{image} = -I(x, y) \quad (5.8)$$

for a image containing objects that are line drawings such as the object in Figure



Figure 12: An ellipse

The potential energy according to (5.8) is

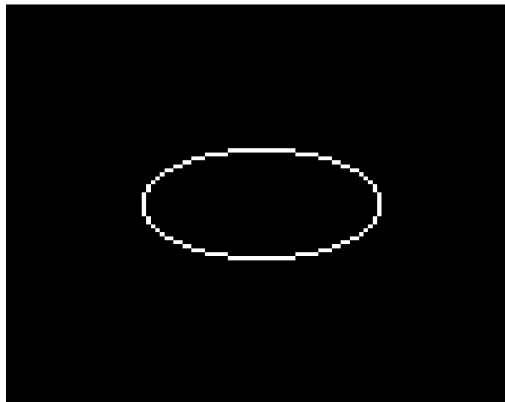


Figure 13: Potential field of the image in Figure 11

and the gradient vector field of such an object is shown in Figure 13.



Figure 14: Force field of the image in Figure 11

A zoom on the boundary in Figure shows how the force field acts on the active contour

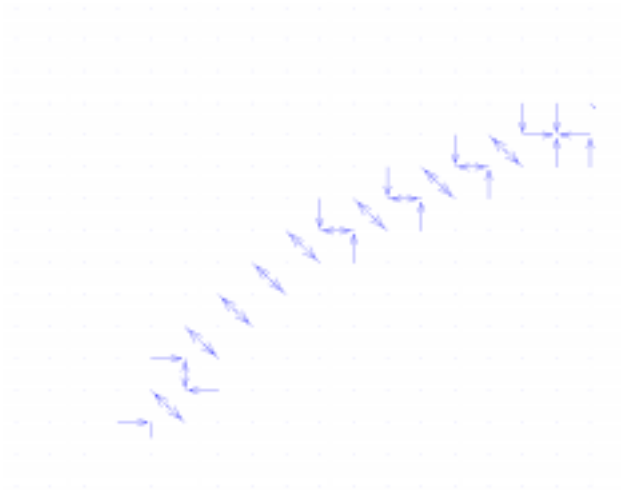


Figure 15: Zoom on the object boundary in the force field

If the object of interest is, however, homogeneous, such as the one in Figure 15



Figure 16: Zoom on the object boundary in the force field

this energy model does not give a satisfactory force field. The potential energy of this image calculated according to (5.8) is shown in Figure 17

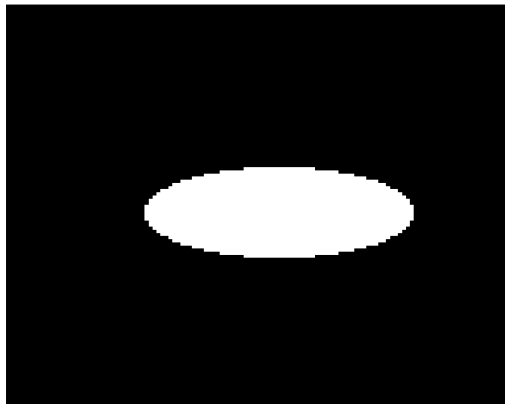


Figure 17: Zoom on the object boundary in the force field

A zoom on the boundary in the force field is in Figure

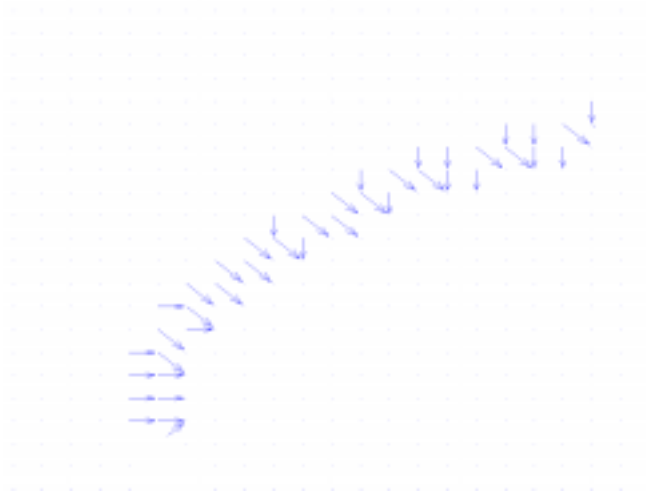


Figure 18: Zoom on the object boundary in the force field

We see that the force field points toward the object boundaries on the outside of the object and away from the boundary on the inside of the object. A contour pulled by such a force field would not stop on the boundary, but continued to the center of the object. We calculate, therefore, an edge map of such an image and used that as image energy

$$E_{image}(x, y) = -|\nabla I(x, y)|^2 \quad (5.9)$$

The result is shown in Figure 18



Figure 19: Image Potential

The gradient vector field of this image potential is shown in Figure 19

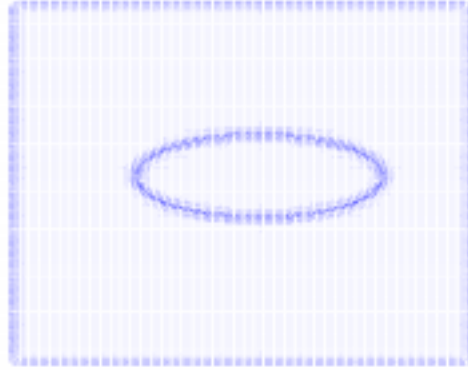


Figure 20: Force field

and a zoom

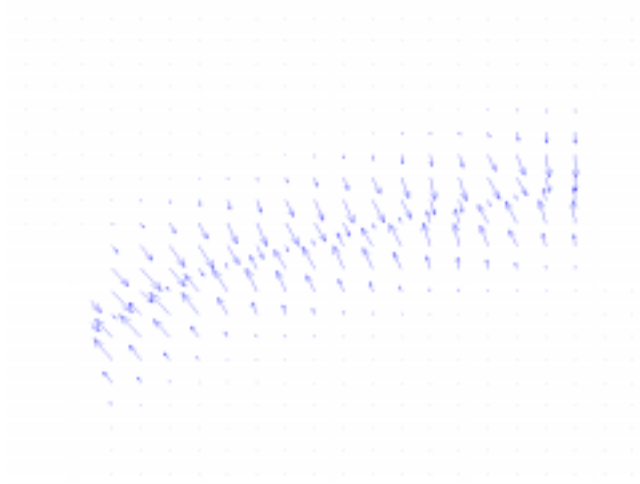


Figure 21: Zoom on the object boundary in the force field

From the range of the of the gradient vector field we see that the curve, if not placed in the vicinity of the object, may not be attracted to it. One way to increase the capture range of the field is to smooth out the boundaries. An improved energy model would be than

$$E_{image}(x, y) = -|\nabla(G_{\sigma}(x, y) * I(x, y))|^2 \quad (5.10)$$

where  $G_\sigma(x, y)$  is a two-dimensional Gaussian function with standard deviation  $\sigma$ . It is easy to see that larger  $\sigma$  will cause the boundaries to become blurry but this has both positive and negative effects: the range will increase but the boundary localization will become less accurate and distinct. For the image in Figure 11, potential of the blurred image is shown in Figure 21



Figure 22: Image potential

and a zoom on the boundary in the force field

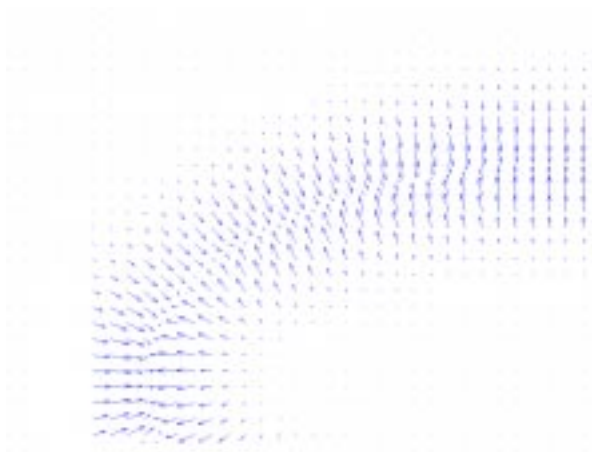


Figure 23: Zoom on the object boundary in the force field

We see that the potential function becomes less sharp and the boundary is thus more difficult to localize. On the other hand, the capture range is increased.

Time and space discretization may cause some additional problems to this image energy model. Even though the initial guess can be close to an edge, instabilities can occur due to the discretization of the evolution problem. If the time discretization step is too large, the curve may move too far across the desired boundary and never come back. This problem can be avoided by manually tuning the time step but is not very practical in large scale applications and it does not solve all the problems. If we choose the time step to be small enough, for example never larger than a pixel size, then the problem is avoided, but only very few high gradient points will attract the curve and the small values of the image force field will not affect the curve much.

### 5.2.2 The “Balloon” Force Field

One way to improve the model was proposed by Cohen [12]. Instead of acting on the time step, he modified the force field  $\mathbf{F}_{image}$  by normalizing it, taking  $\mathbf{F}_{image} = -k\nabla E_{image}/\|\nabla E_{image}\|$ , where the product of the time step and  $k$  is on the order of the pixel size. Since the magnitude of  $\mathbf{F}_{image}$  is about one pixel, when a point of the curve is close to an edge point, it is attracted to the edge and stabilizes there. Thus smaller and larger gradients have the same influence on the curve.

It happens often that, due to noise, some isolated points are gradient maxima and can stop the curve when it passes by as shown in Figure 23

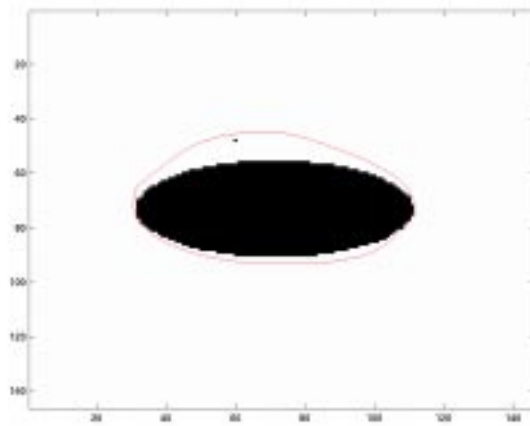


Figure 24: An object with a local maxima in the vicinity of the boundary

To solve the problems mentioned in the previous section, Cohen added another

force which makes the contour have a more dynamic behavior. The curve is considered as a balloon (in 2D) that we inflate. We add to the previous forces a pressure force pushing outside. The force now becomes

$$\mathbf{F} = k_1 \mathbf{n}(s) - k_2 \frac{\nabla E_{image}}{\|\nabla E_{image}\|} \quad (5.11)$$

where  $\mathbf{n}(s)$  is the normal unit vector to the curve at point  $\mathbf{v}(s)$  and  $k_1$  is the amplitude of the force. Now, the curve expands and it is attracted and stopped by edges as before, but since there is a pressure force, if the edge is too weak the curve can pass through this edge. If the curve runs into an isolated point, it tends to create a tangent discontinuity at this point. The smoothing effect with the help of the inflation force removes the discontinuity and the curve passes through the edge as shown in Figure 24

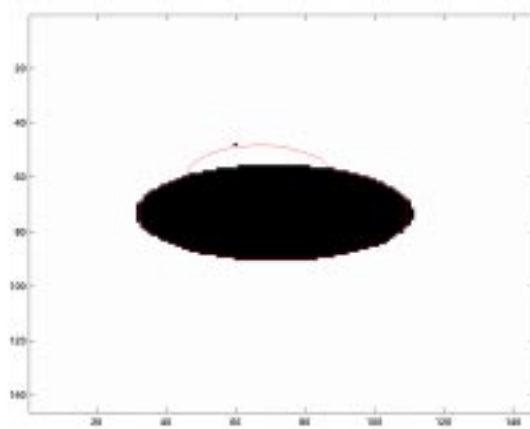


Figure 25: The balloon snake after passing the local maxima

### 5.2.3 The Gradient Vector Flow Force Field

The energy model proposed by Cohen solved some of the problems of the original model but not all. If the object to segmented has a concavity, none of the previously proposed models manages to force the snake into the concavity. The reason may be seen from the Figure 25.

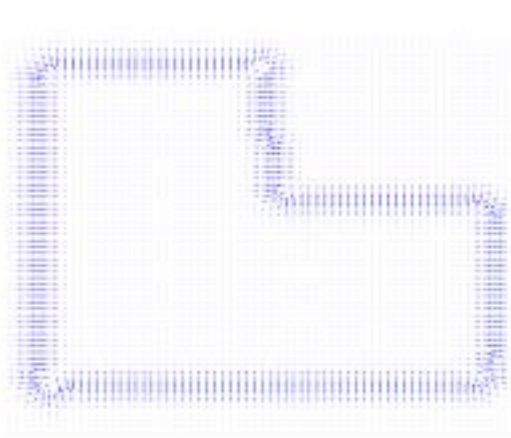


Figure 26: Force field of a polygon with concavity

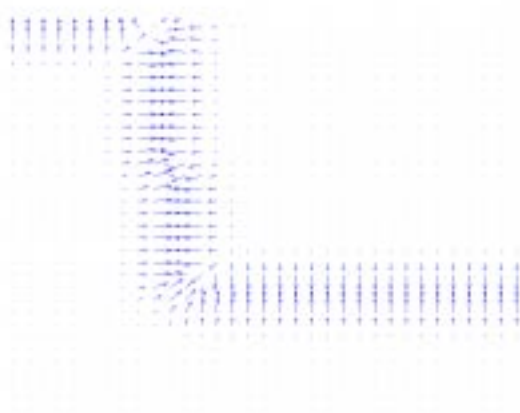


Figure 27: Zoom no the boundary

Although the image forces point correctly toward the object boundary, the range of the force field is small. Therefore, the curve is “pulled apart” but not made to progress into the concavity. The problem is not solved by the Cohen’s model since it only changes the magnitude and not the range of the force.

Xu suggested a new force to solve the problem. The underlying mathematical premise for this new force comes from the Helmholtz theorem, which states that the most general static vector field can be decomposed into two components: an irrotational and a solenoidal component. The original potential force becomes an

irrotational field since it is the gradient of a scalar potential function.

Xu proposed to generate a more general field by allowing the possibility that it comprises both an irrotational component and a solenoidal component. They designed a new force field that has both the desired properties of both a large capture range and the presence of forces that point into boundary concavities. In the original evolution equation,

$$\mathbf{v}_t(s, t) = \alpha \mathbf{v}_{ss}(s, t) - \beta \mathbf{v}_{ssss}(s, t) - \nabla E_{image} \quad (5.12)$$

the image force field  $\nabla E_{image}$  is replaced by a new field, the so called gradient vector flow field  $\mathbf{G}(x, y)$ , yielding

$$\mathbf{v}_t(s, t) = \alpha \mathbf{v}_{ss}(s, t) - \beta \mathbf{v}_{ssss}(s, t) - \mathbf{G} \quad (5.13)$$

where  $\mathbf{G}(x, y) = (c(x, y), d(x, y))$  is a vector field that minimizes the energy functional

$$\mathcal{E} = \int \int \mu \{ (c_x^2 + c_y^2 + d_x^2 + d_y^2) + |\nabla f|^2 |\mathbf{G} - \nabla f|^2 \} dx dy \quad (5.14)$$

where  $f(x, y)$  is the edge map defined as  $f(x, y) = |\nabla I(x, y)|$  or  $f(x, y) = |\nabla(G_\sigma(x, y) * I(x, y))|$  for gray level images.

We see that when  $|\nabla f|$  is small, the energy is dominated by sum of the squares of the partial derivatives of the vector field, yielding a slowly-varying field. On the other hand, when  $|\nabla f|$  is large, the second term dominates the integrand and minimized by setting  $\mathbf{G} = |\nabla f|$ . This produces the desired effect of keeping  $\mathbf{G}$  nearly equal to the gradient of the edge map when it is large, but forcing the field to be slow-varying when in homogeneous regions. The parameter  $\mu$  is a regularization parameter governing the tradeoff between the first term and the second term in the integrand.

The desired vector field is calculated from the energy functional using the calculus of variation which we will say more about in the next section. The new force field is shown in Figure 27

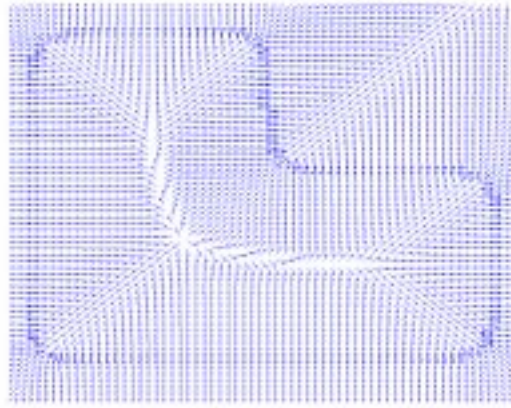


Figure 28: Force field

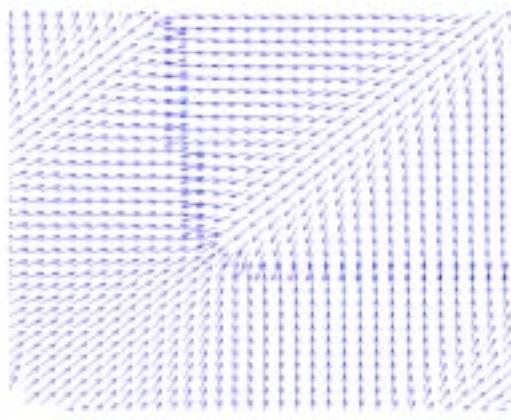


Figure 29: Zoom on the boundary of the force field

In the section 6, we will test these three models experimentally on simulated and real MR brain images.

### 5.3 Experimental Results

In all the following simulations, one and the same set of control points was used with different snake models. The parameters were set to  $\alpha = 0.2$  and  $\beta = 0.2$ .

First, the methods were tested on a simple image with control points placed in

the vicinity of the object. The goal was to observe the convergence rate and the accuracy of the models. We start with the original snake model

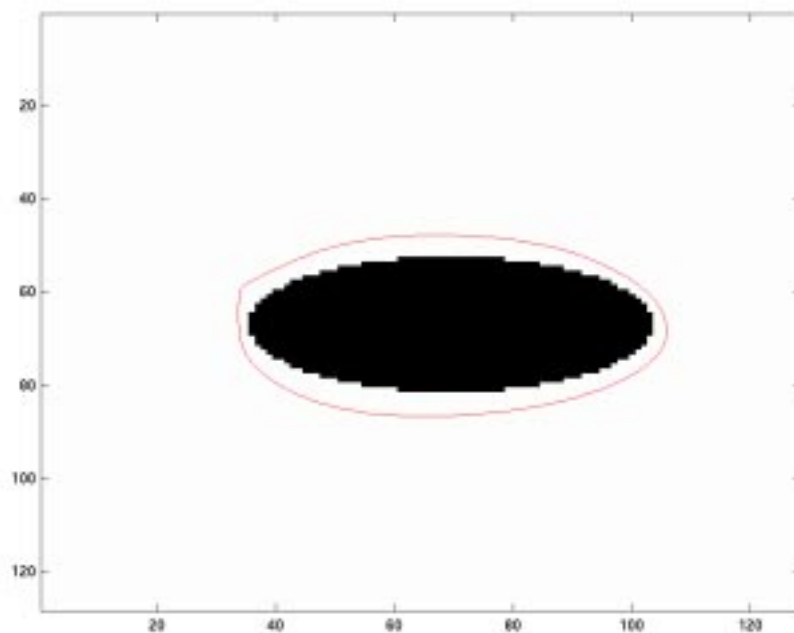


Figure 30: Original snake: Iteration 1

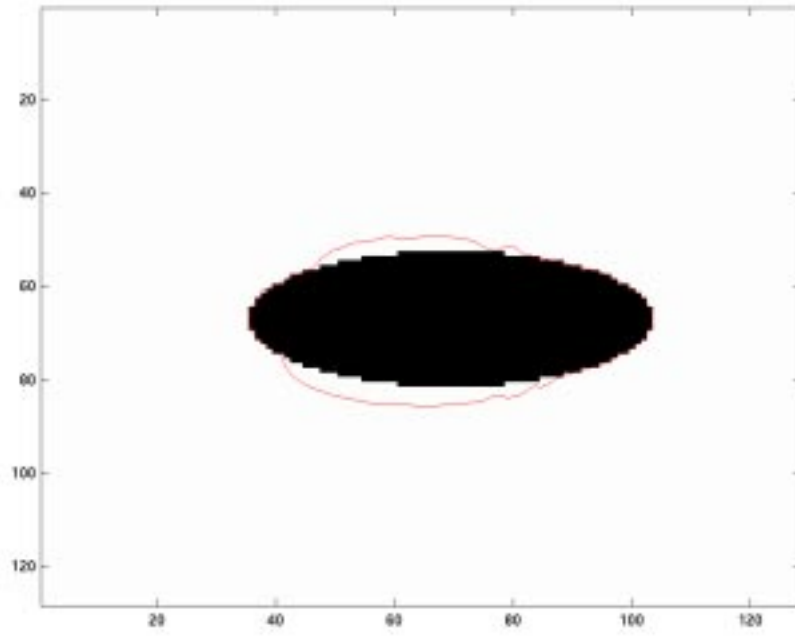


Figure 31: Original snake: Iteration 15

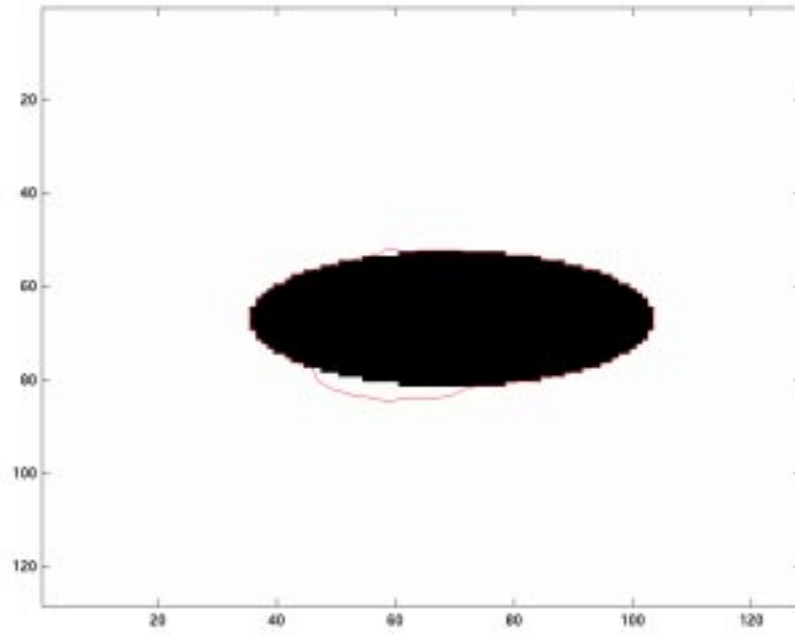


Figure 32: Original snake: Iteration 25

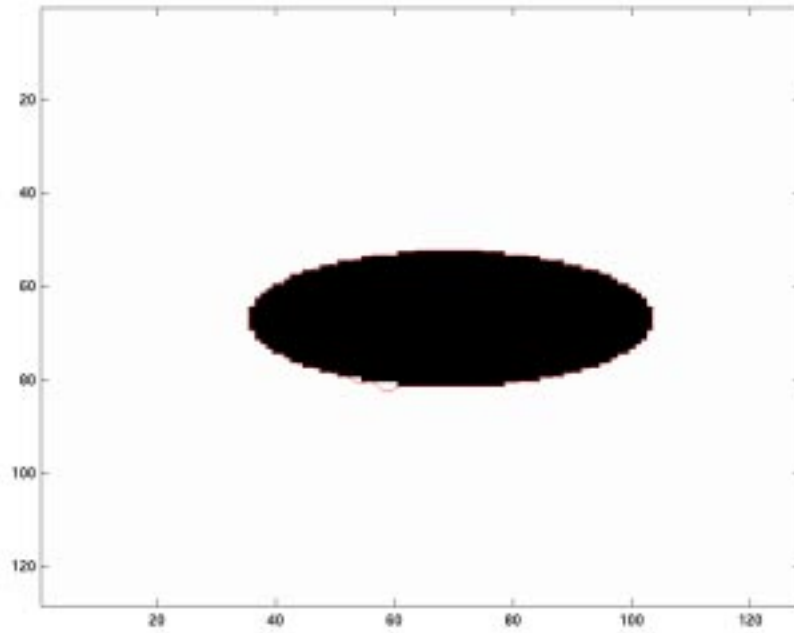


Figure 33: Original snake: Iteration 35

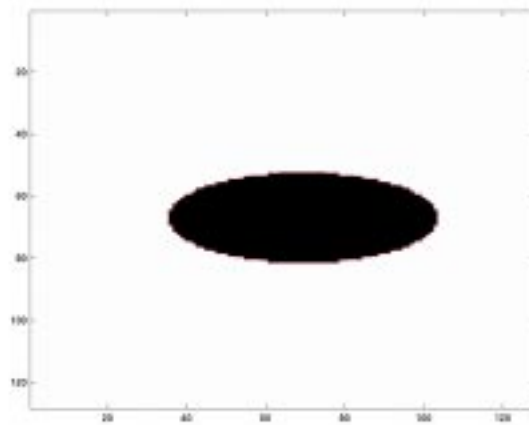


Figure 34: Original snake: Iteration 45

Then, using the same set of control points and the images, the balloon snake was tested.

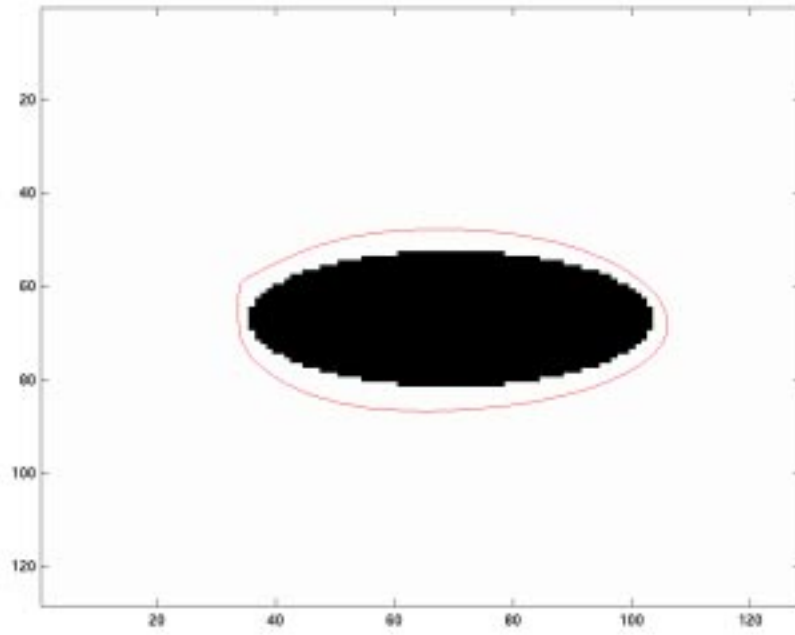


Figure 35: Balloon snake: Iteration 1

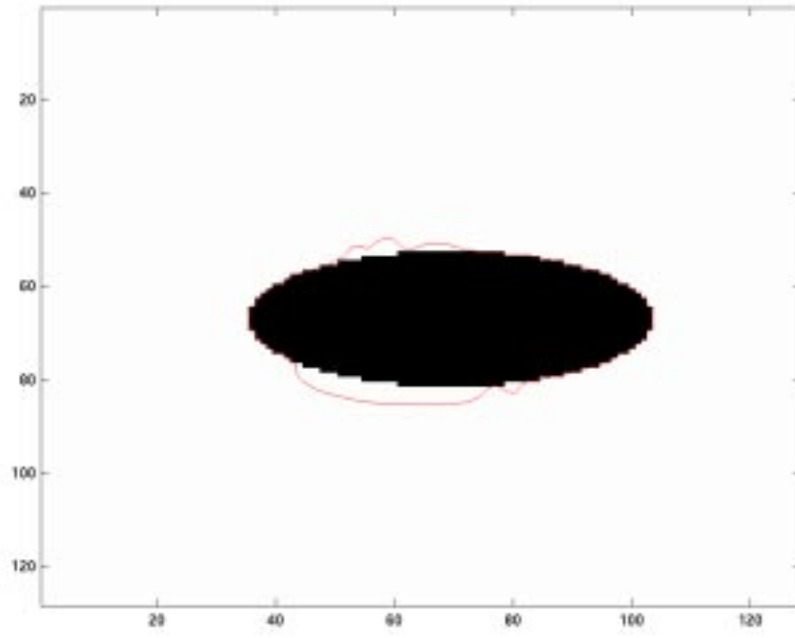


Figure 36: Balloon snake: Iteration 15

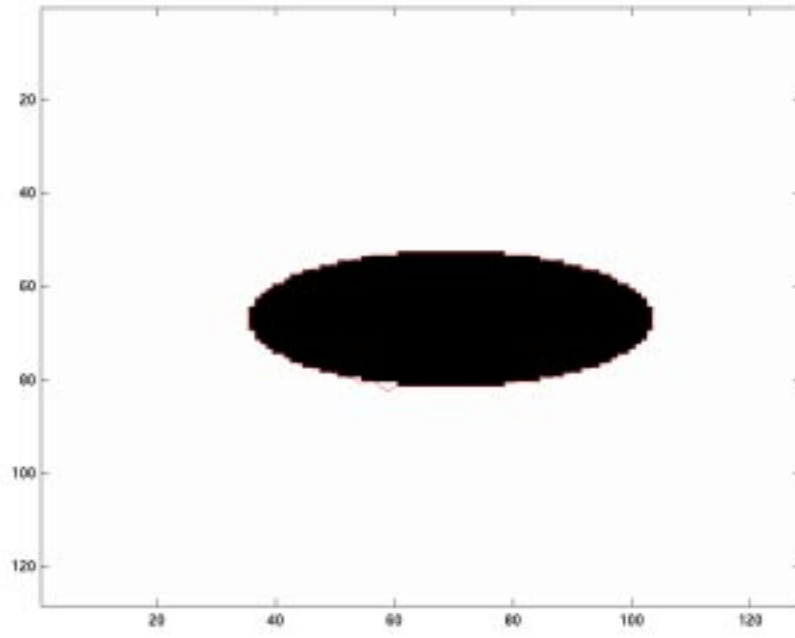


Figure 37: Ballon snake: Iteration 25

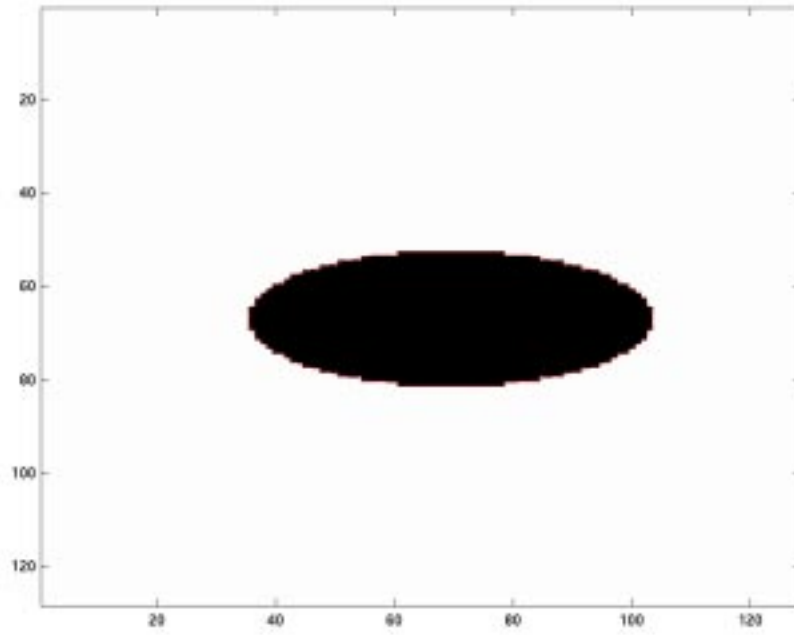


Figure 38: Balloon snake: Iteration 35

In the next images, the GVF snake is shown acting on the same image with the same initial set of control points.

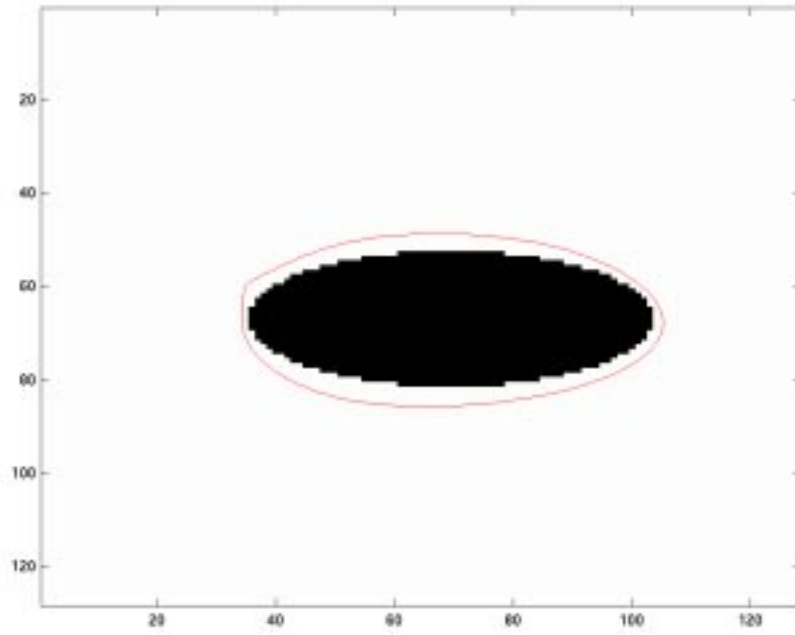


Figure 39: GVF snake: Iteration 1

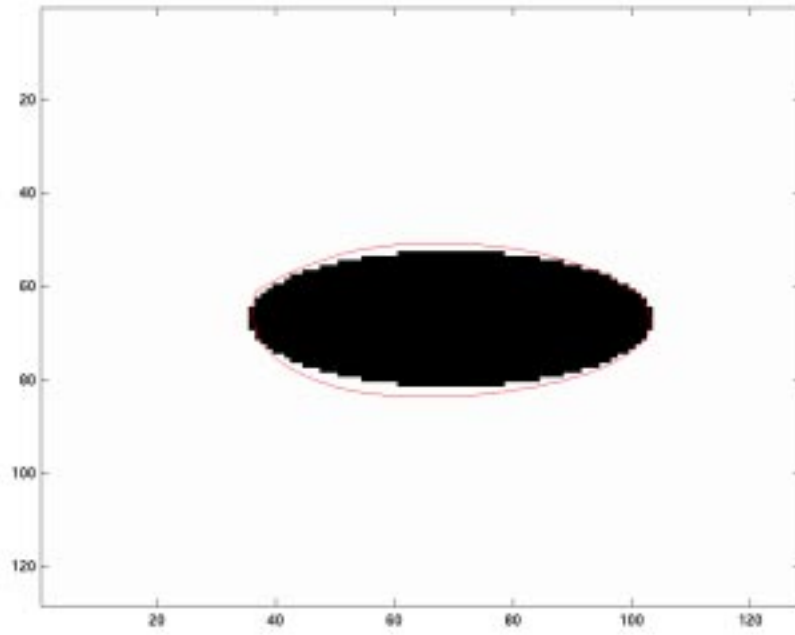


Figure 40: GVF snake: Iteration 3

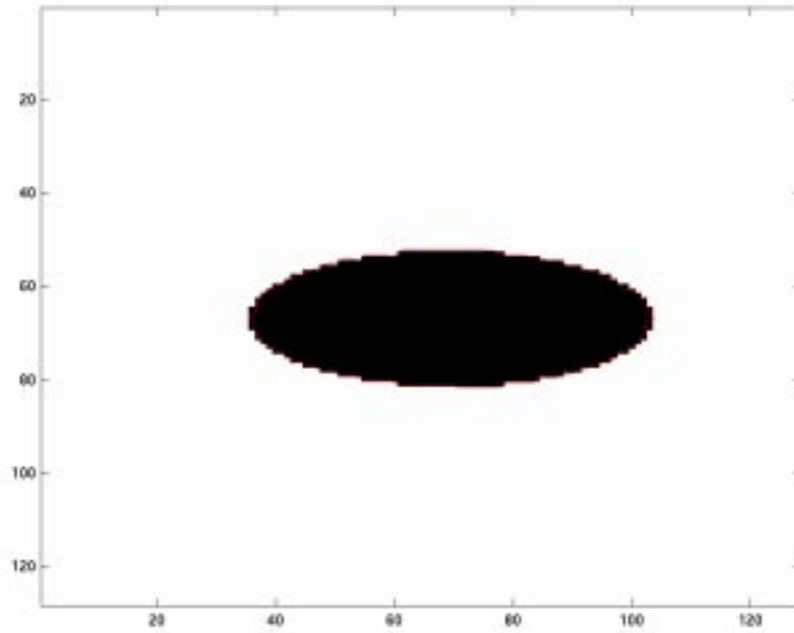


Figure 41: GVF snake: Iteration 6

In the next set of images, “concavity” problem was treated. The snake parameters were  $\alpha = 0.2$ ,  $\beta = 0.2$ .

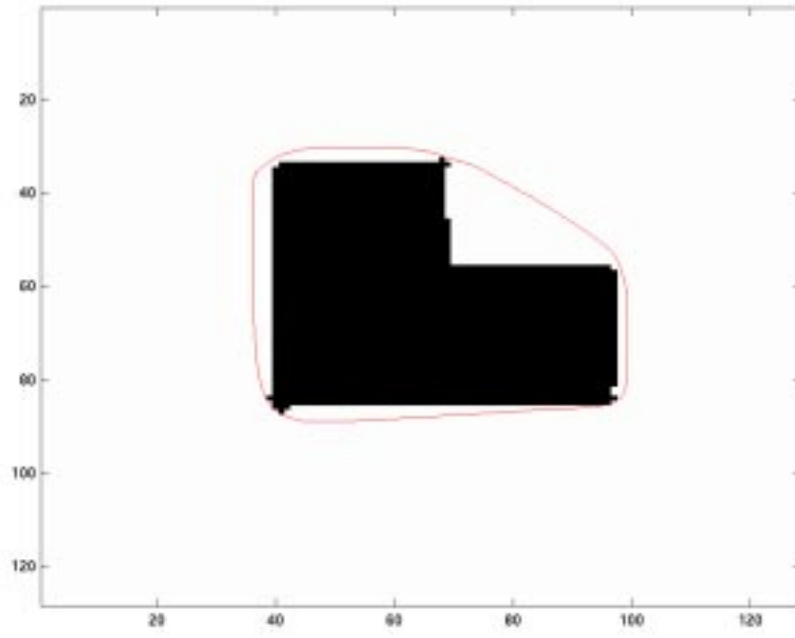


Figure 42: Traditional snake: Iteration 1

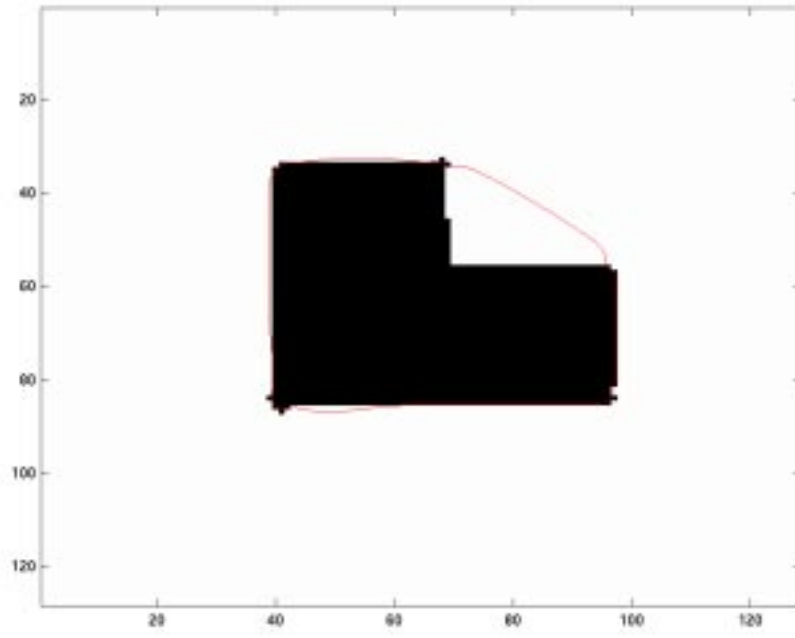


Figure 43: Traditional snake: Iteration 5

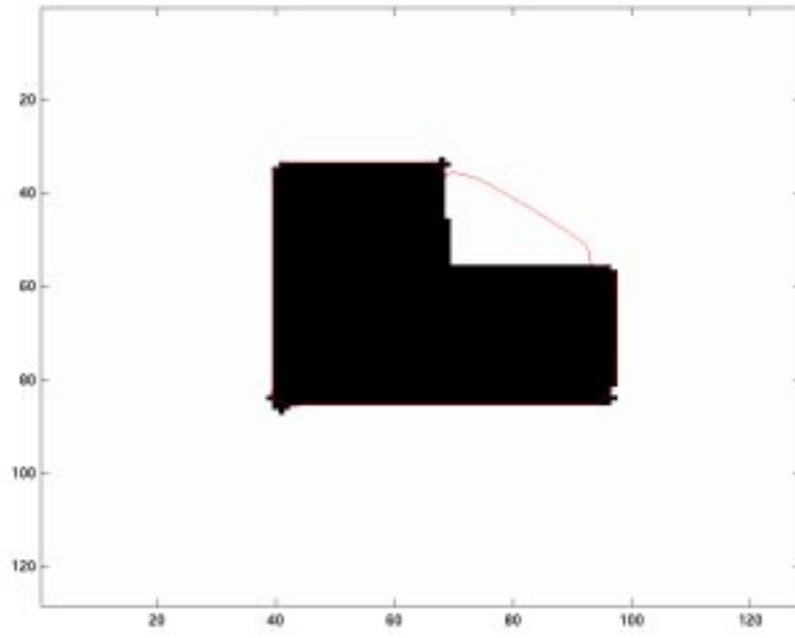


Figure 44: Traditional snake: Iteration 15

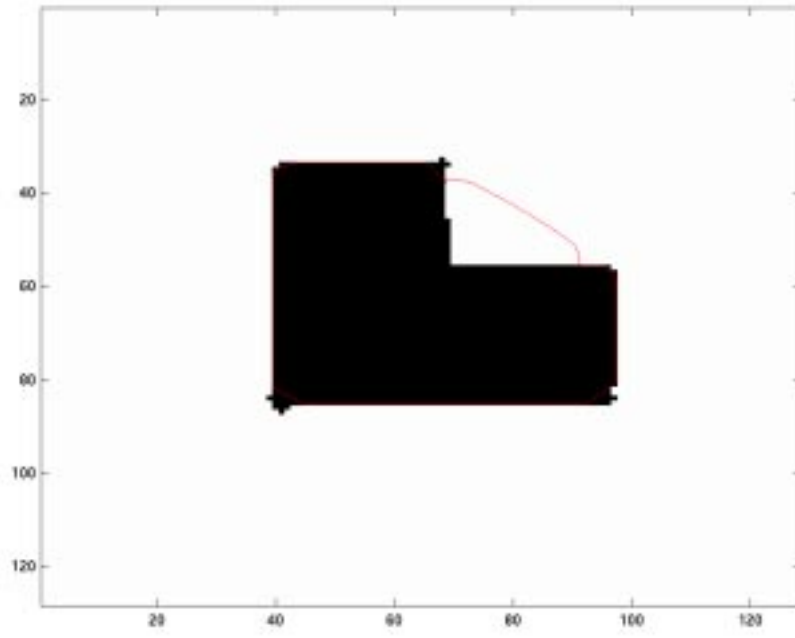


Figure 45: Traditional snake: Iteration 25

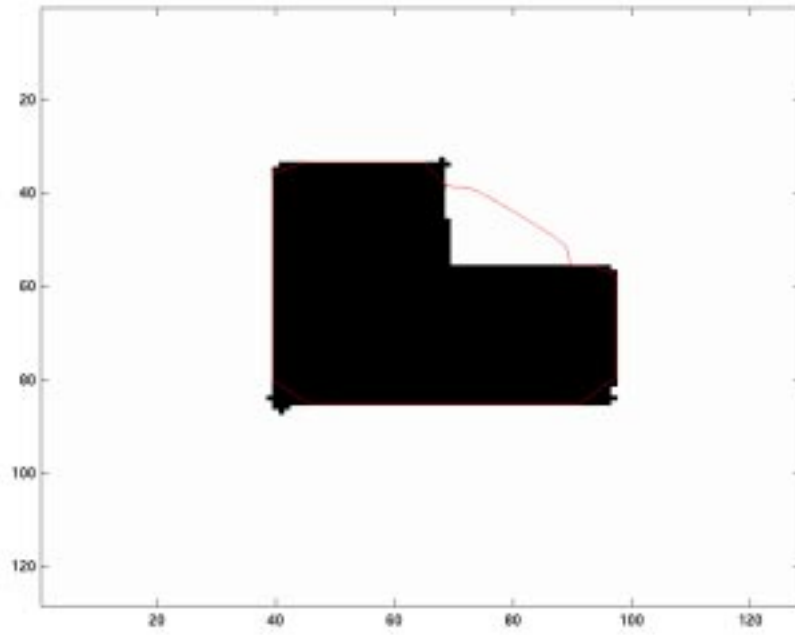


Figure 46: Traditional snake: Iteration 35

Further iterations did not produce significant changes. Now, the balloon model is applied to the same image.

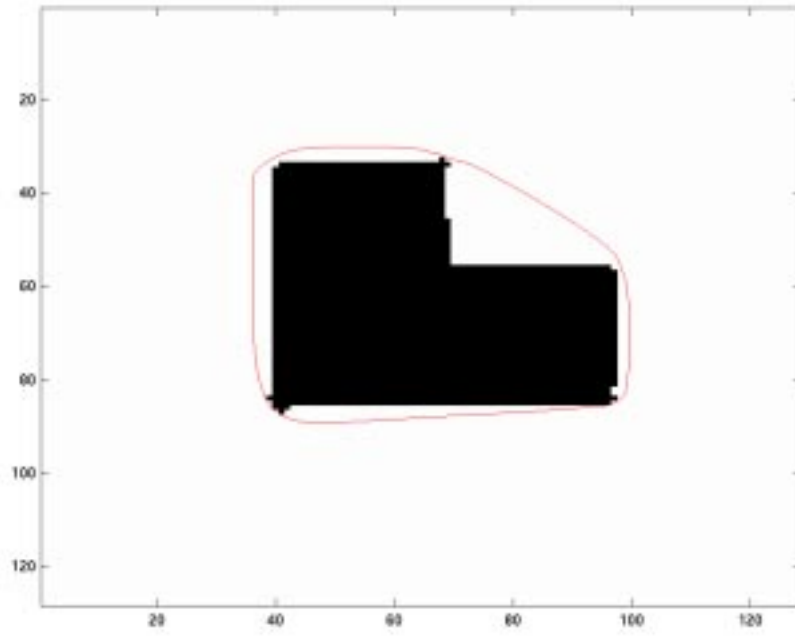


Figure 47: Balloon snake:Iteration 1

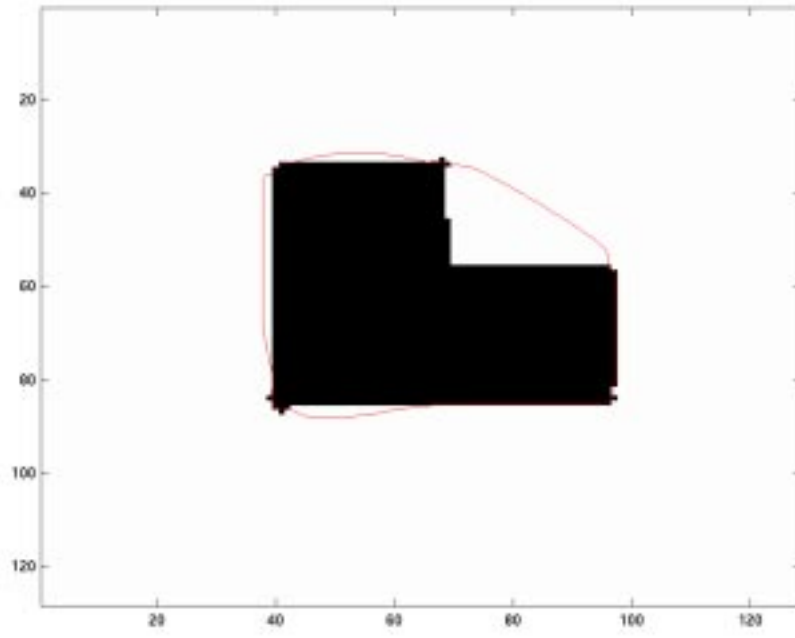


Figure 48: Balloon snake: Iteration 5

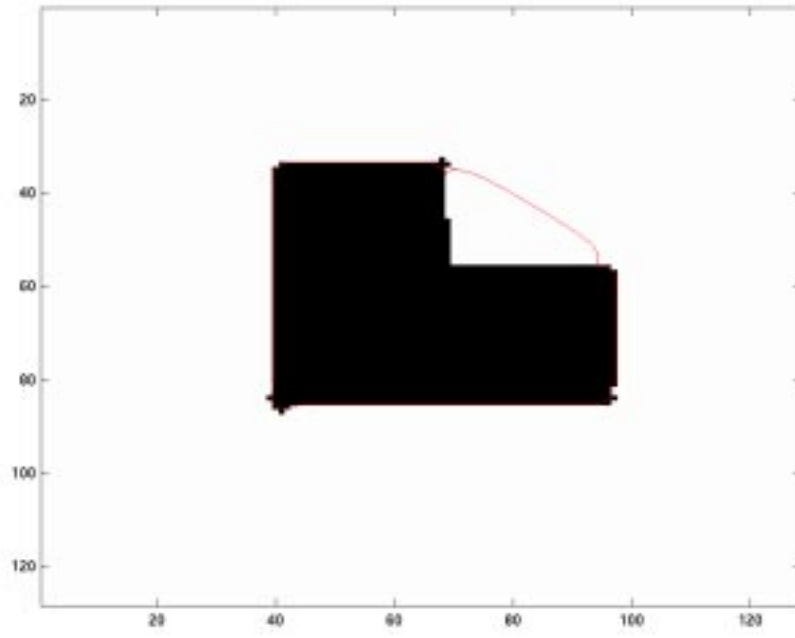


Figure 49: Balloon snake: Iteration 15

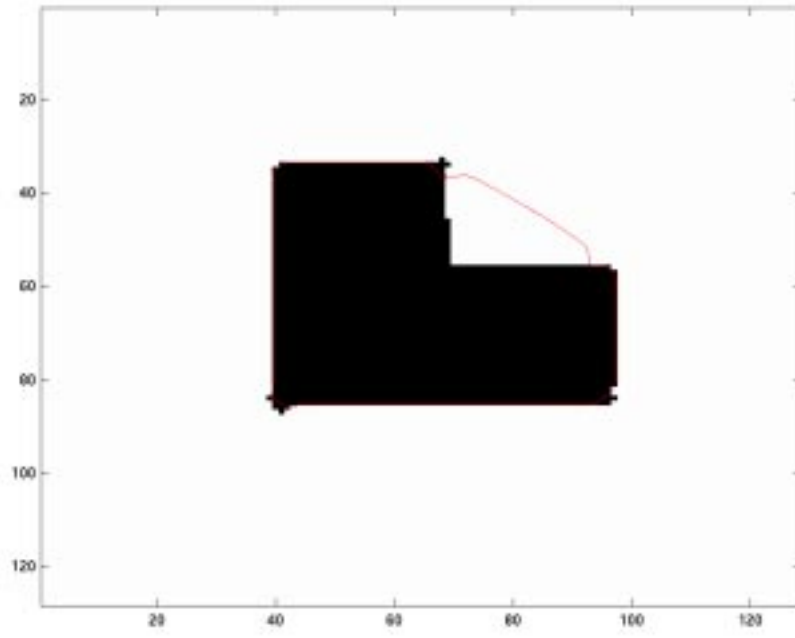


Figure 50: Balloon snake: Iteration 25

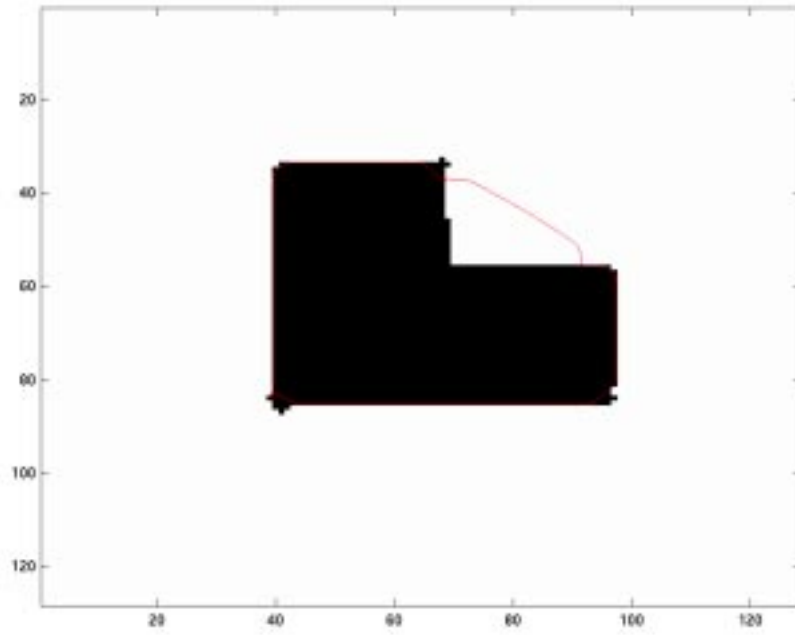


Figure 51: Balloon snake: Iteration 35

Then, the gvf snake was tested on the image.

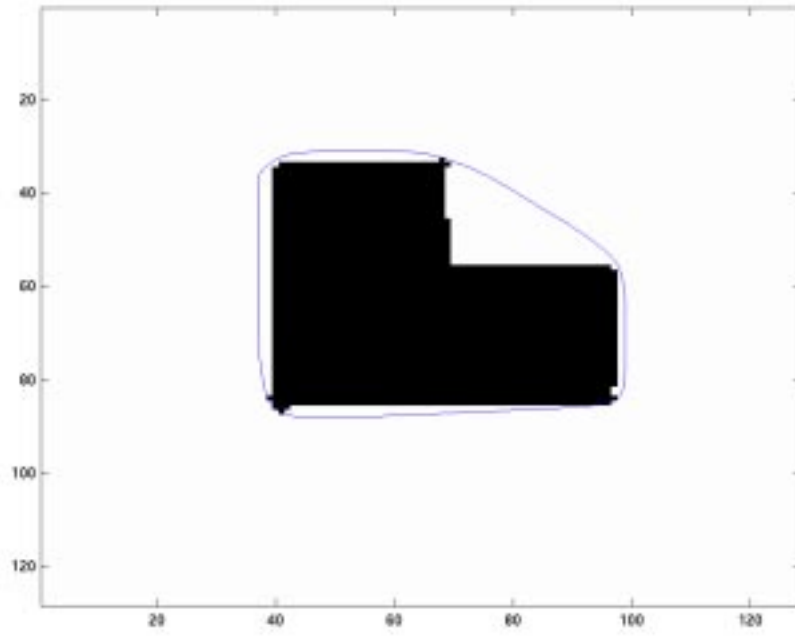


Figure 52: GVF snake: Iteration 1

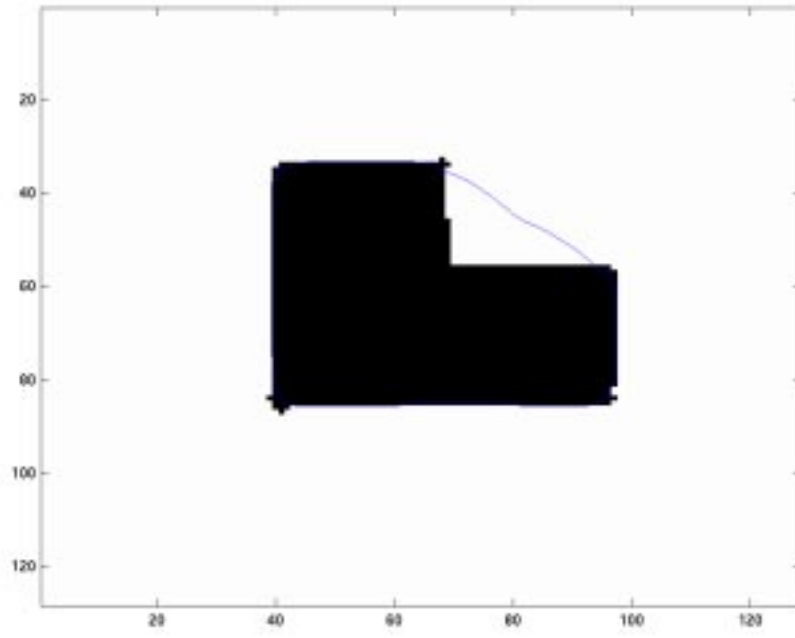


Figure 53: GVF snake: Iteration 5

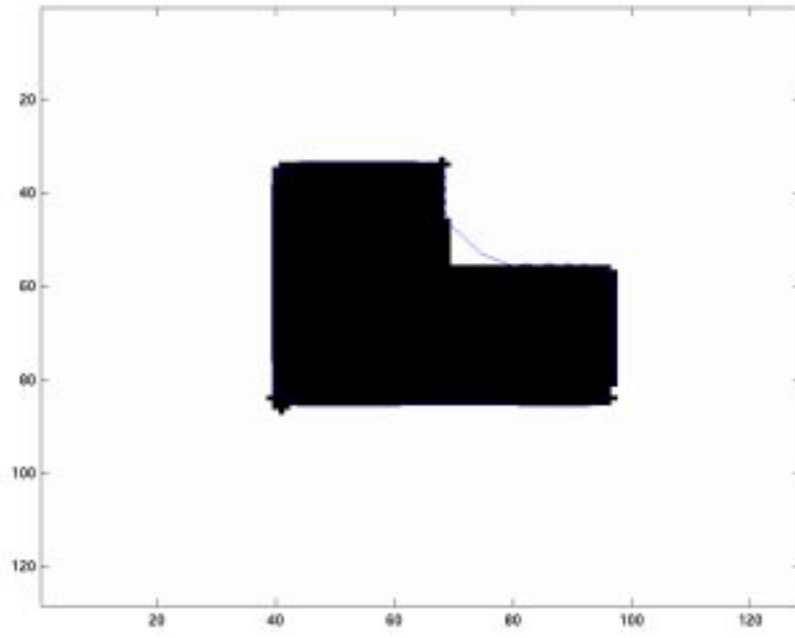


Figure 54: GVF snake: Iteration 15

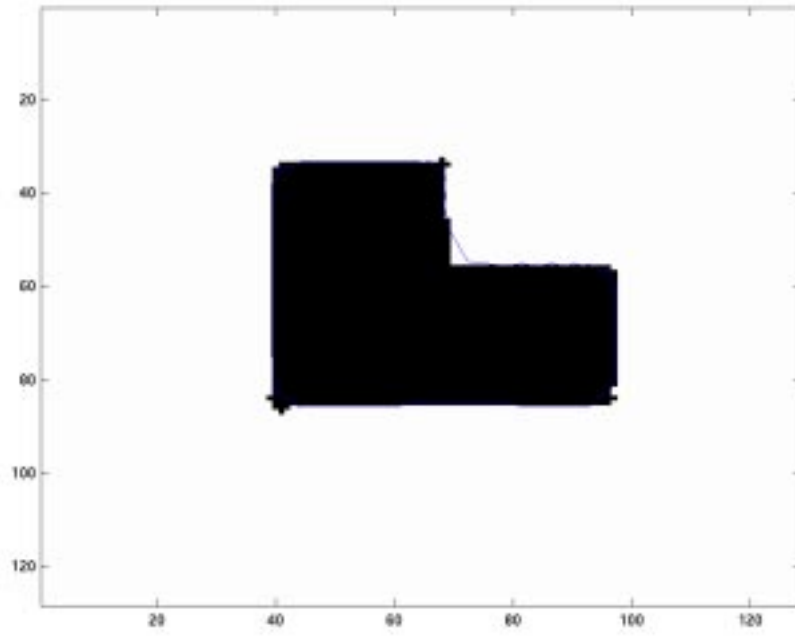


Figure 55: GVF snake: Iteration 25

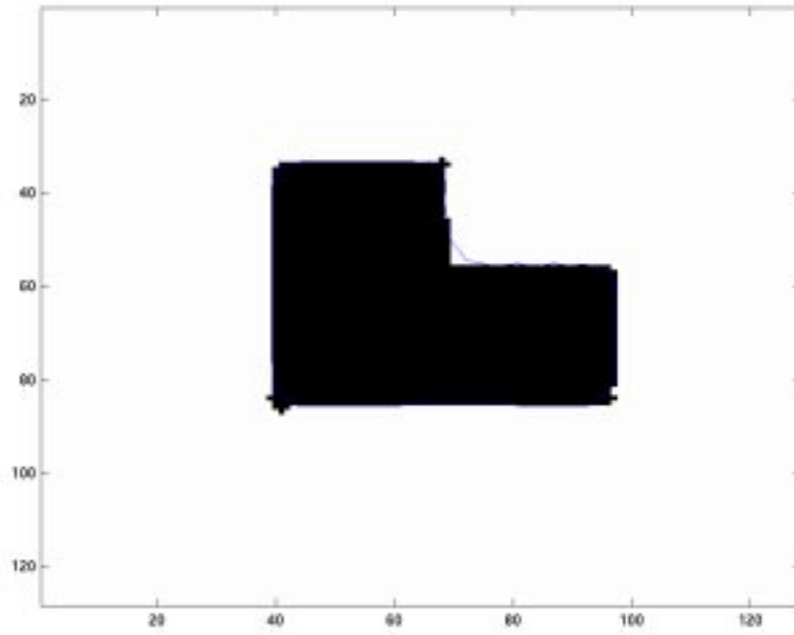


Figure 56: GVF snake: Iteration 35

We further tested the noise tolerance of these methods. The noise is additive Gaussian noise with zero-mean and variance  $\sigma^2 = 1$

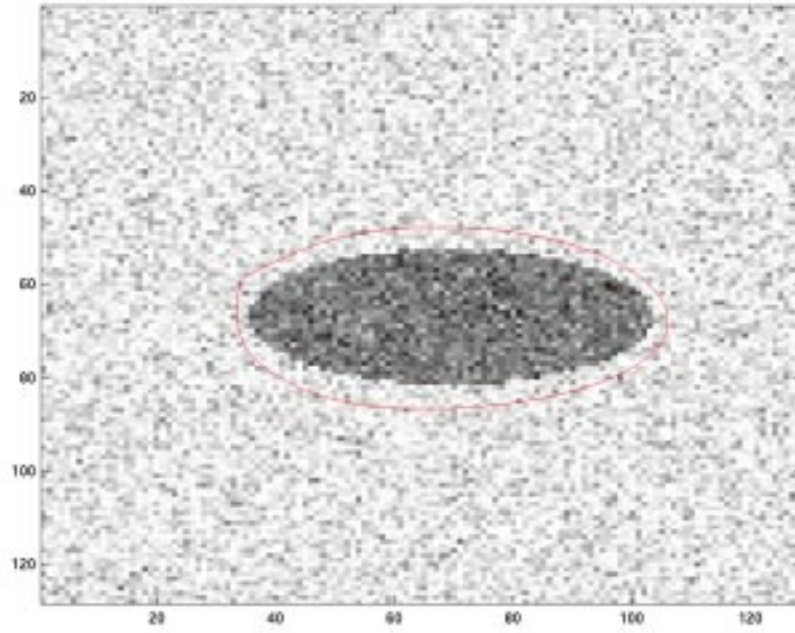


Figure 57: Original snake: Iteration 1

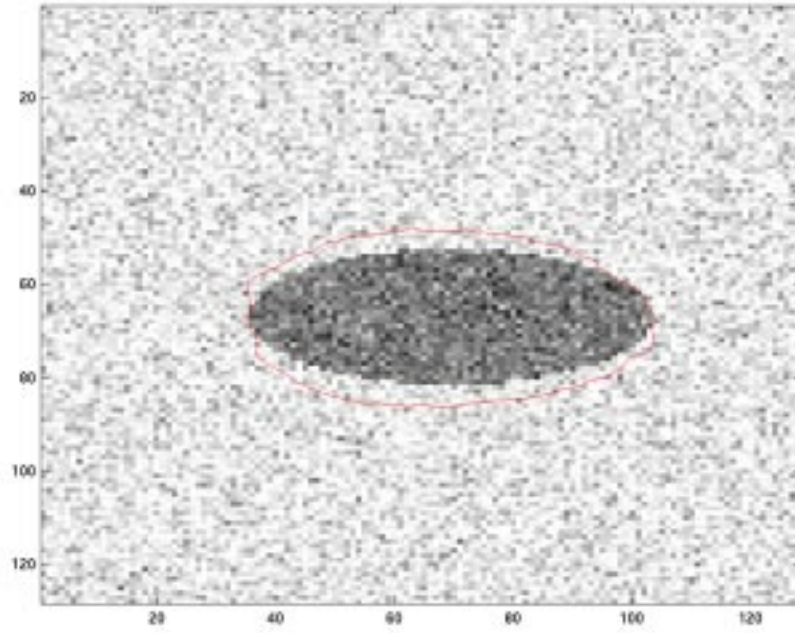


Figure 58: Original snake: Iteration 10

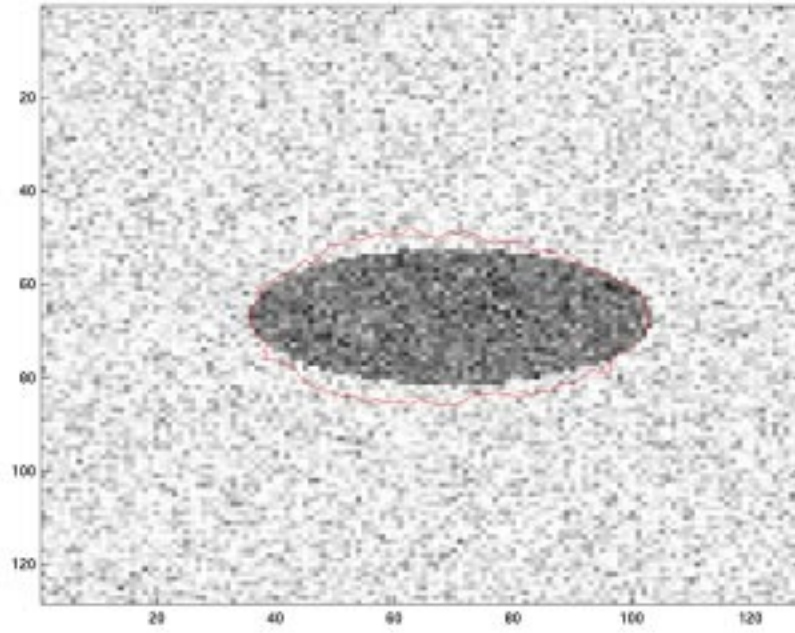


Figure 59: Original snake: Iteration 35

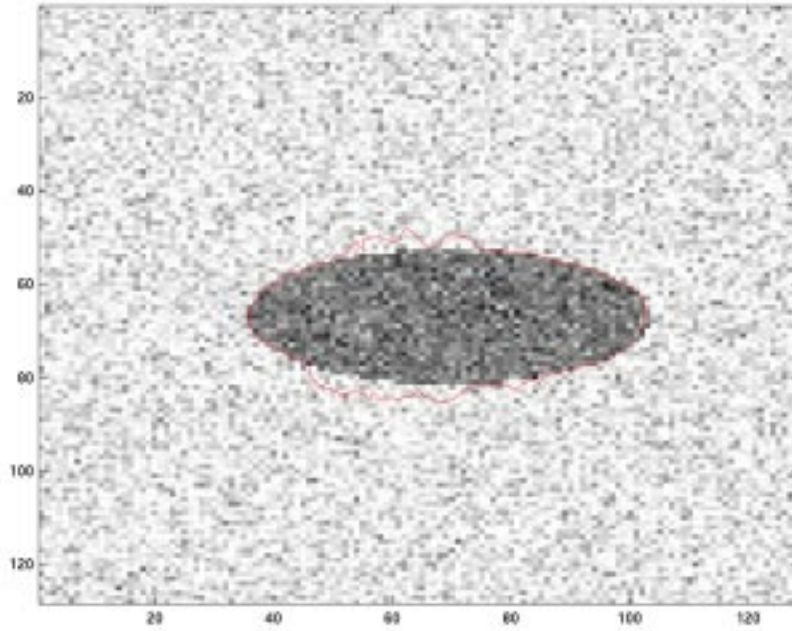


Figure 60: Original snake: Iteration 55

In the next image set, the performance of the Balloon snake is shown.

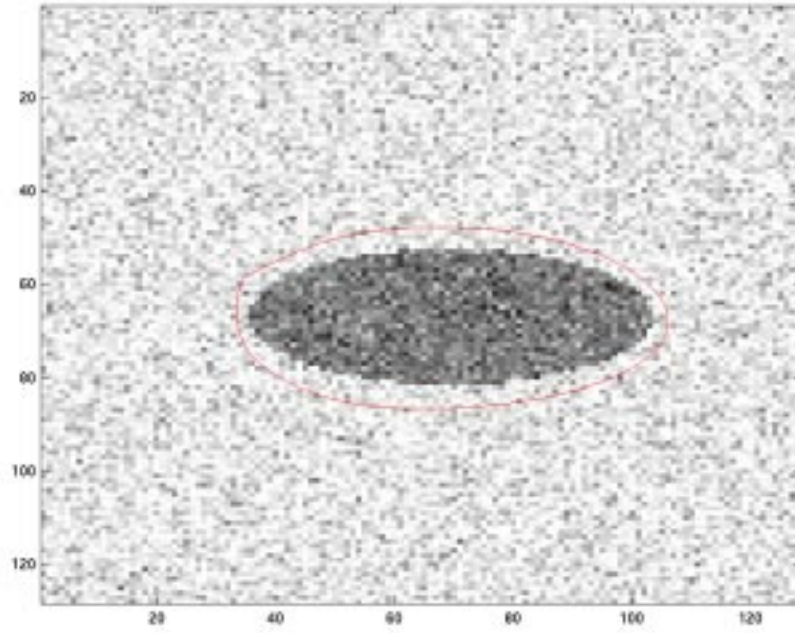


Figure 61: Balloon snake: Iteration 1

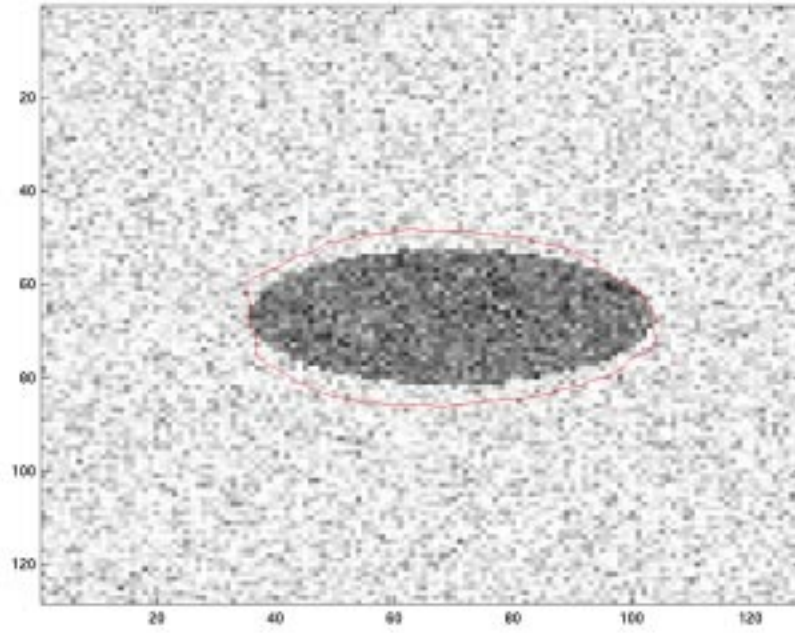


Figure 62: Balloon snake: Iteration 10

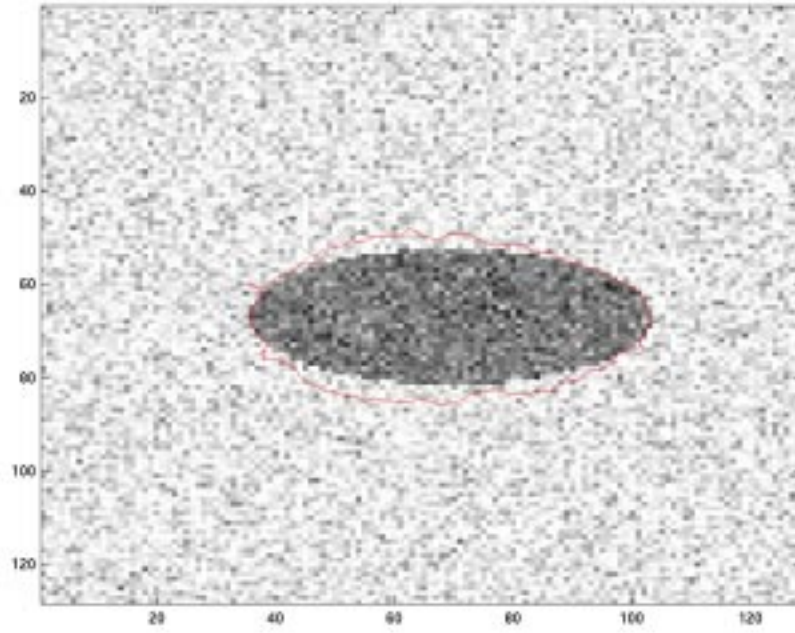


Figure 63: Balloon snake: Iteration 20

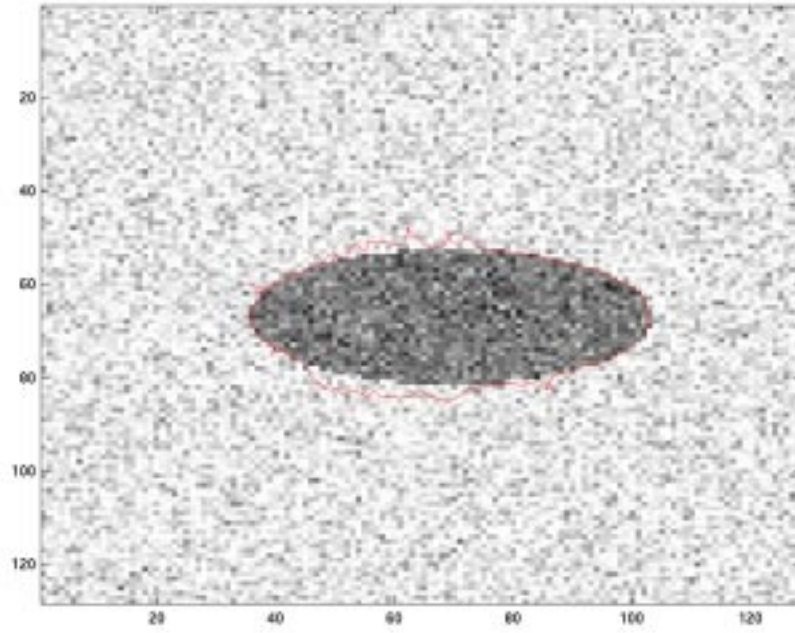


Figure 64: Balloon snake: Iteration 30

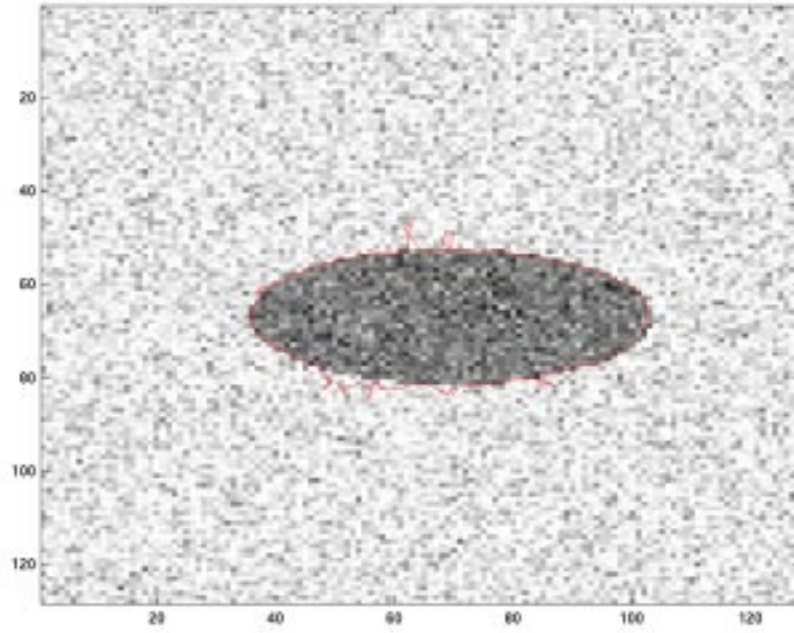


Figure 65: Balloon snake: Iteration 40

The next image set shows the performance of the GVF snake on the image.

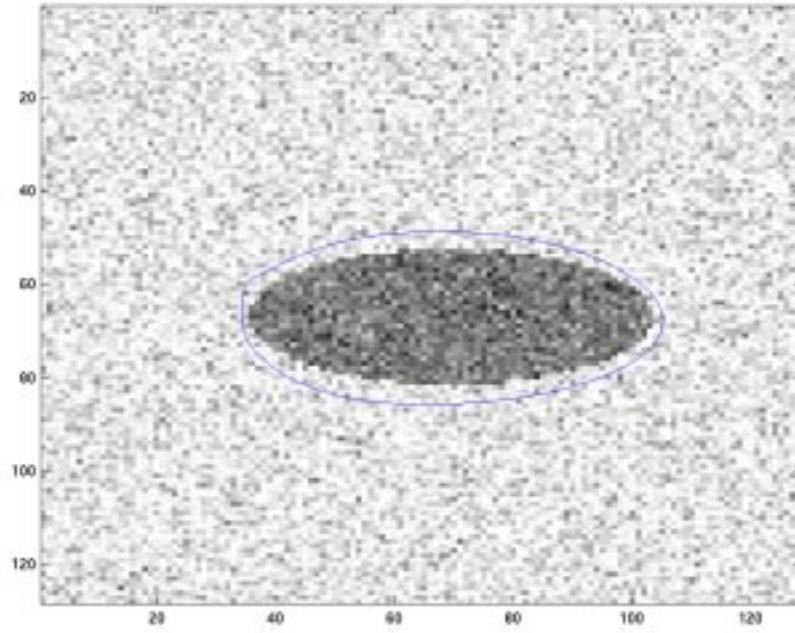


Figure 66: GVF snake: Iteration 1

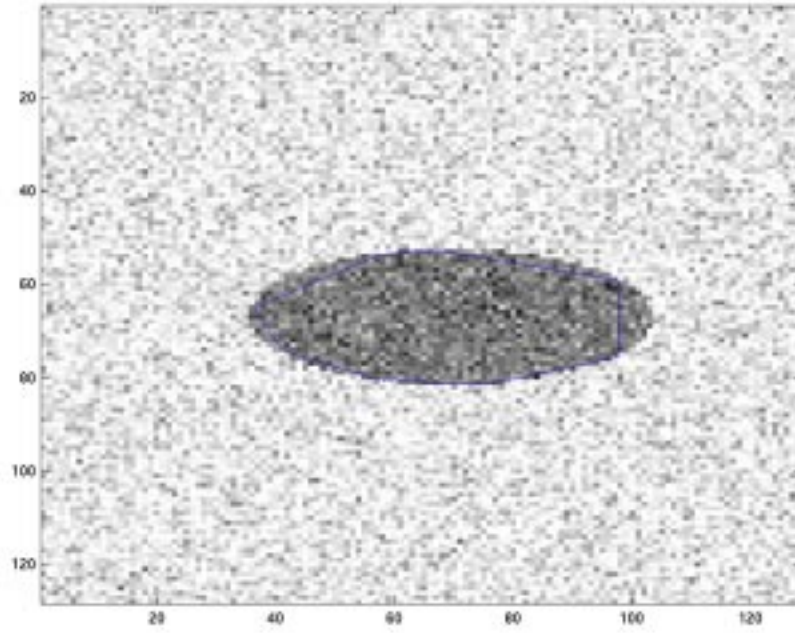


Figure 67: GVF snake: Iteration 25

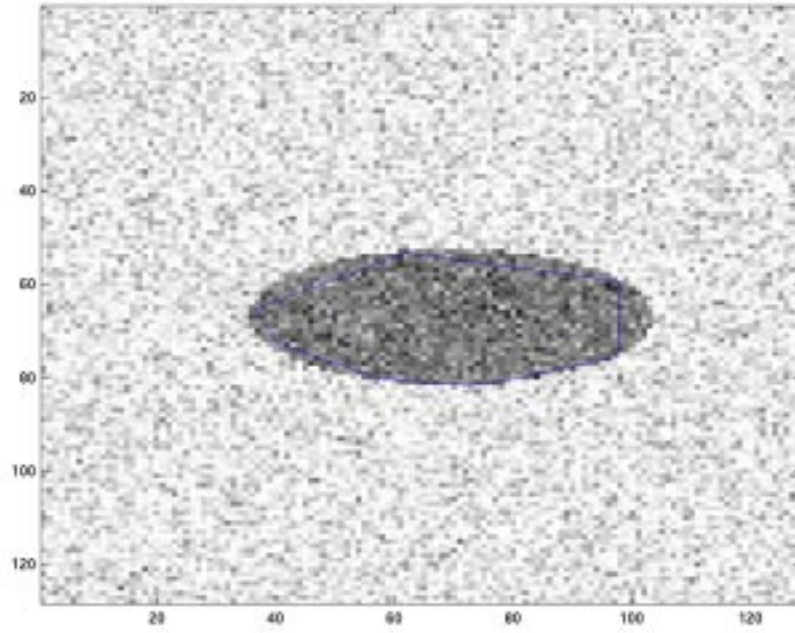


Figure 68: GVF snake: Iteration 50

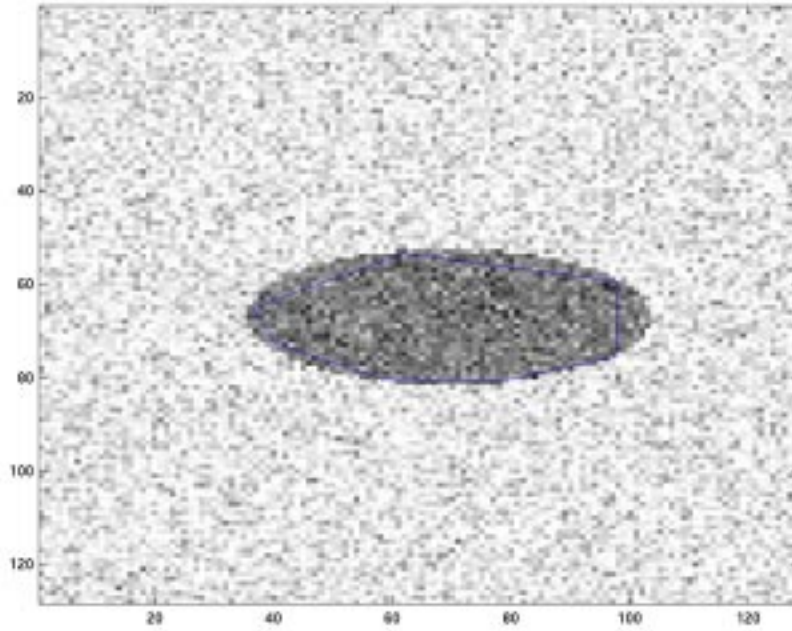


Figure 69: GVF snake: Iteration 75

The methods were then tested on a MR midsagittal brain image. The curve was initialized as close as possible to the boundary of the corpus callosum in order to ensure the convergence.

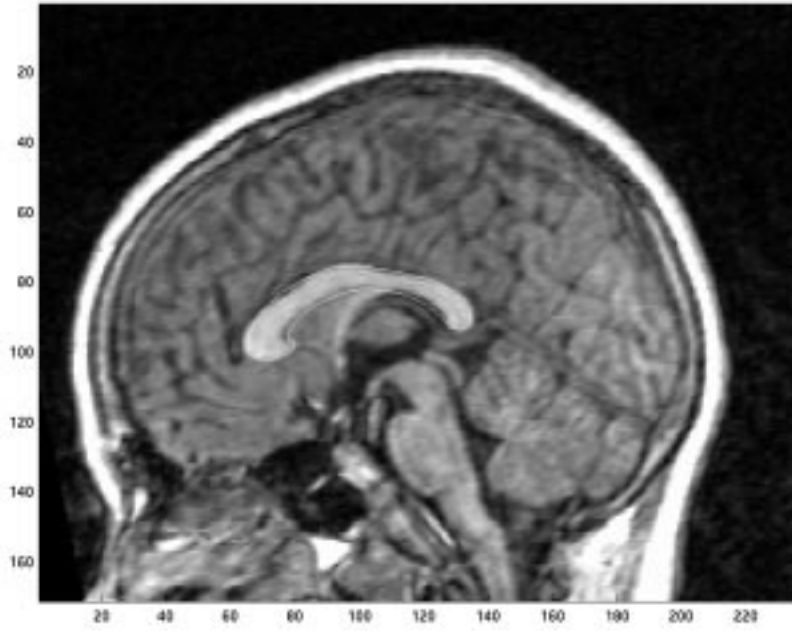


Figure 70: Original snake: Iteration 1

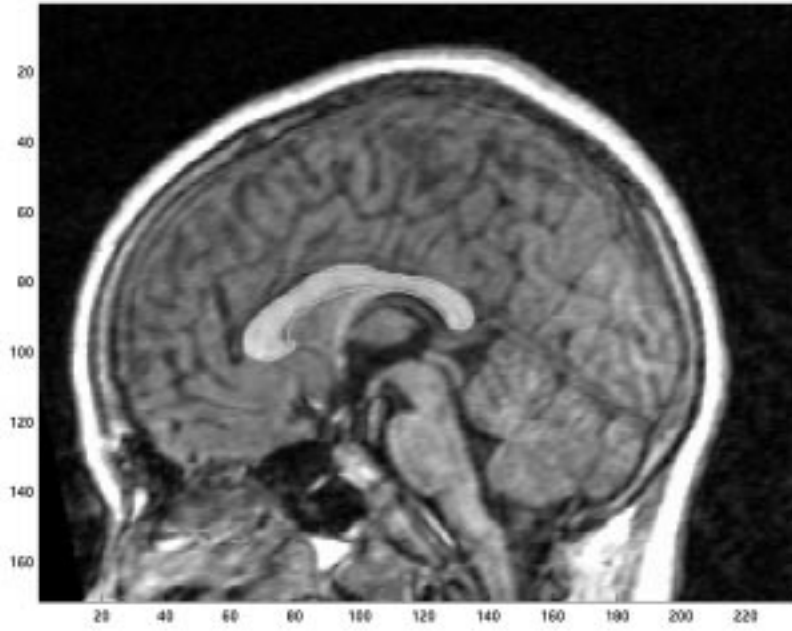


Figure 71: Original snake: Iteration 5

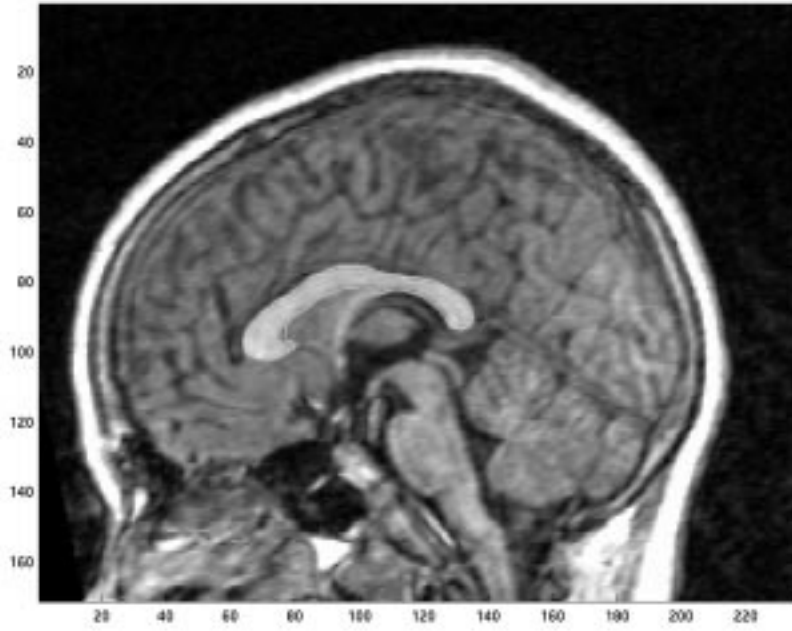


Figure 72: Original snake: Iteration 15

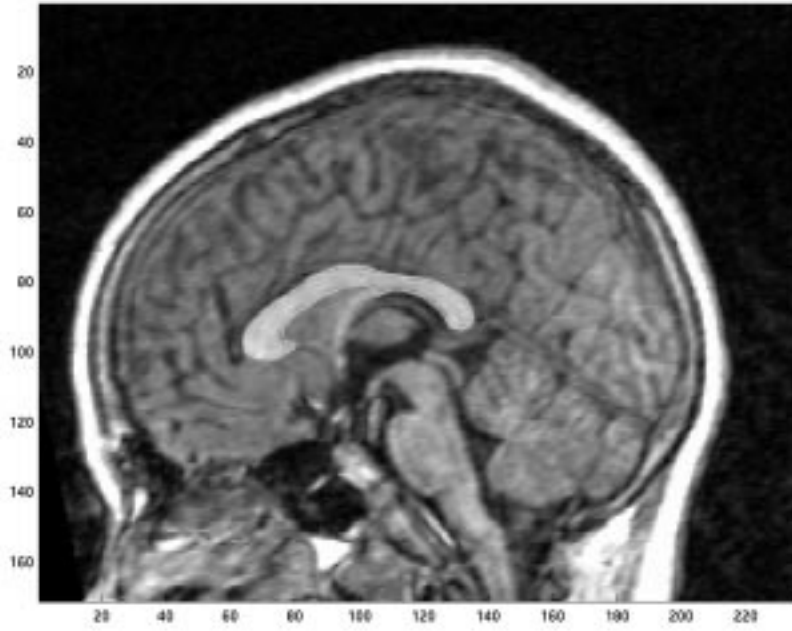


Figure 73: Original snake: Iteration 25

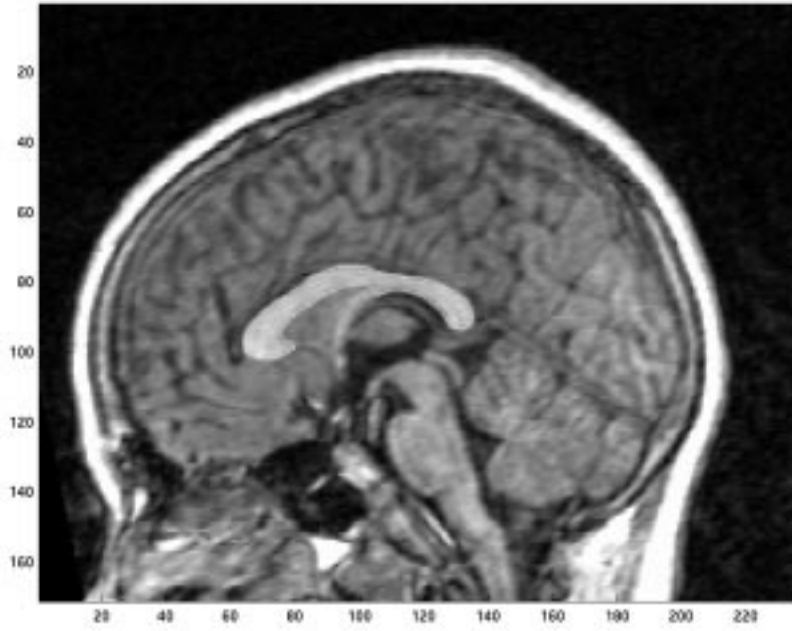


Figure 74: Original snake: Iteration 35

The balloon snake was tested next using the same model parameters.

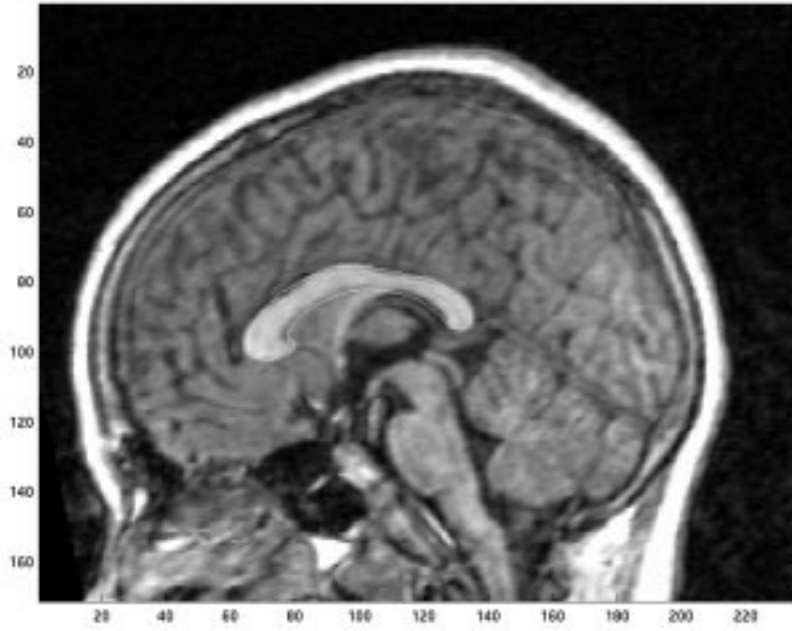


Figure 75: Balloon snake: Iteration 1

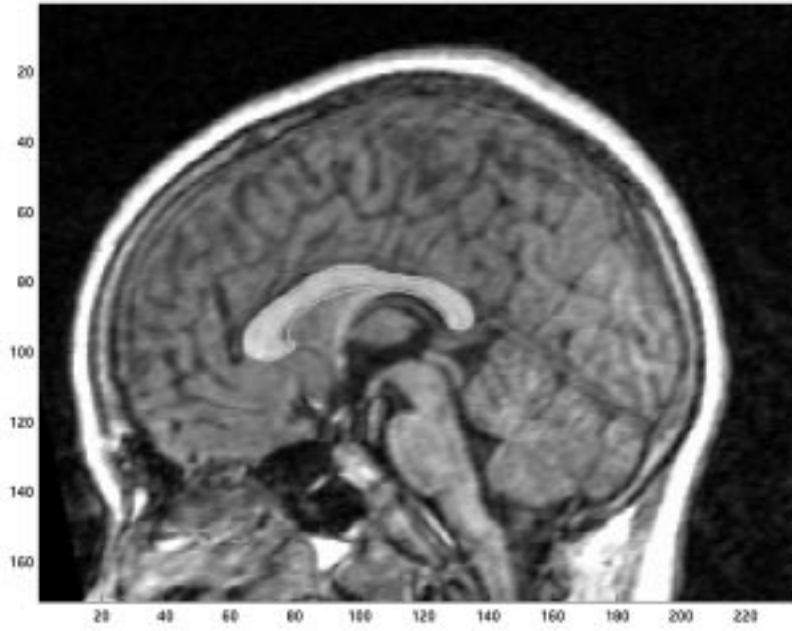


Figure 76: Balloon snake: Iteration 5

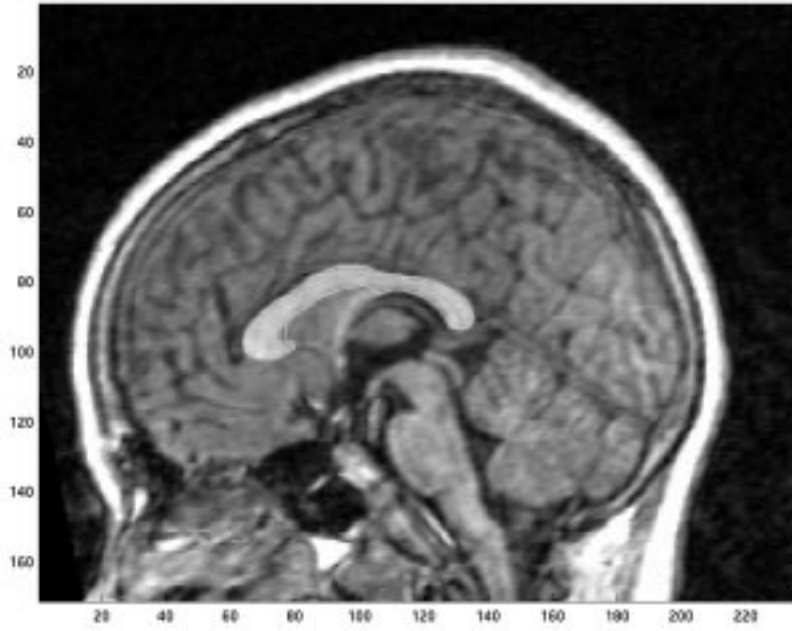


Figure 77: Balloon snake: Iteration 15

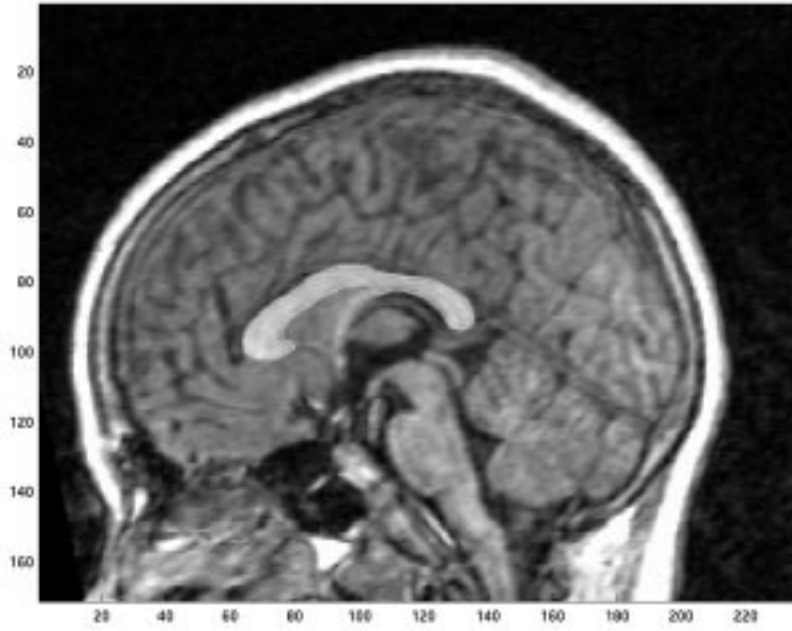


Figure 78: Balloon snake: Iteration 25

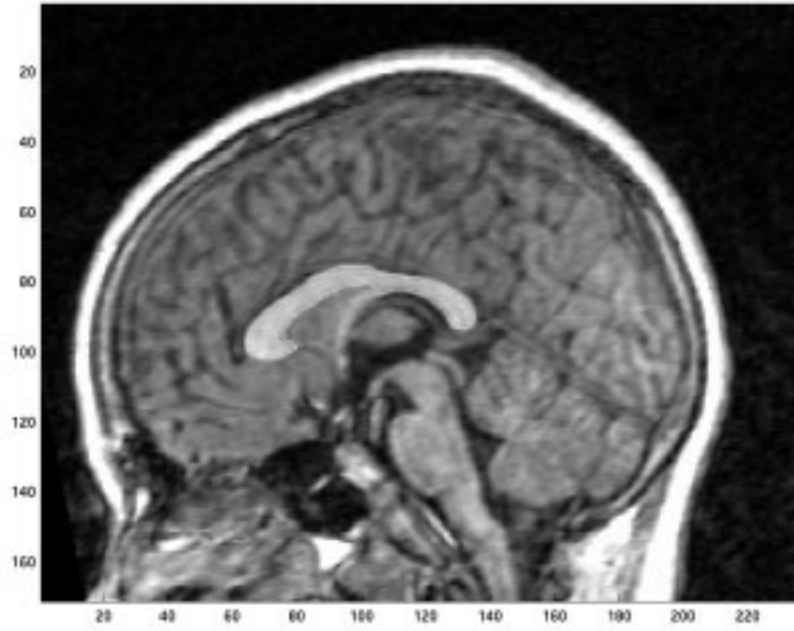


Figure 79: Balloon snake: Iteration 35

Using the same model parameters again, the GVF snake was tested.

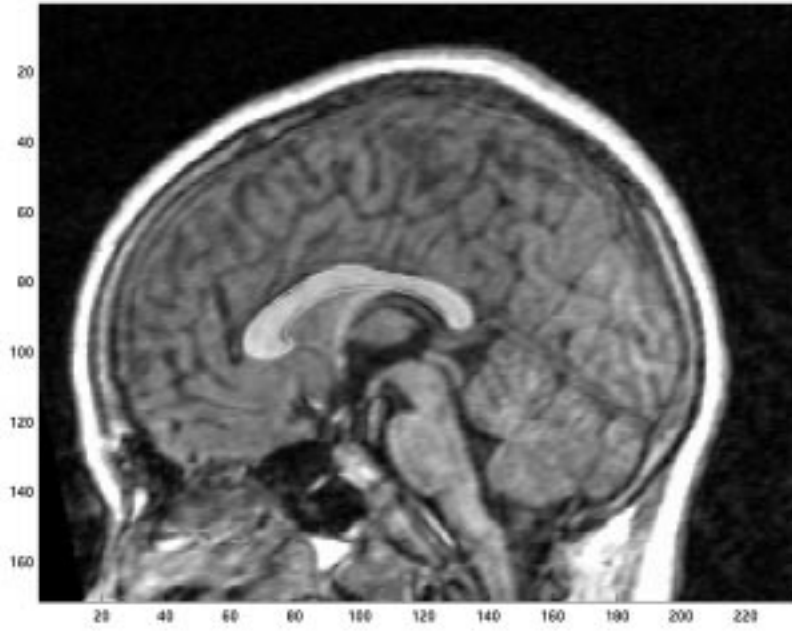


Figure 80: GVF snake: Iteration 1

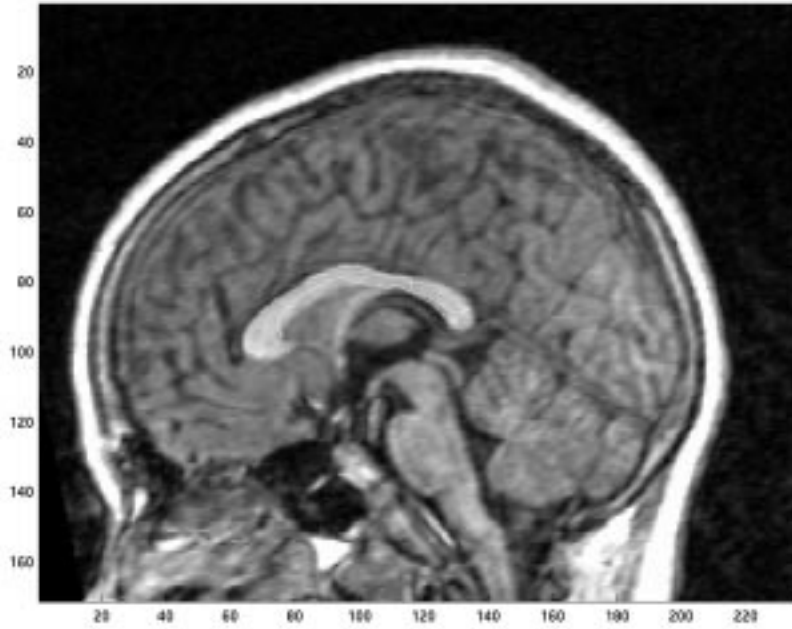


Figure 81: GVF snake: Iteration 5

We see that the snake “breaks” the boundary after 5 iterations. Several different values of  $\mu$  (see section 5.2.3) were tried, but no significant improvement was achieved. In order to tone down the effect of the gradient vector field and let the snake evolve more slowly, the image force term was scaled down with factor 0.8 and the simulation repeated.

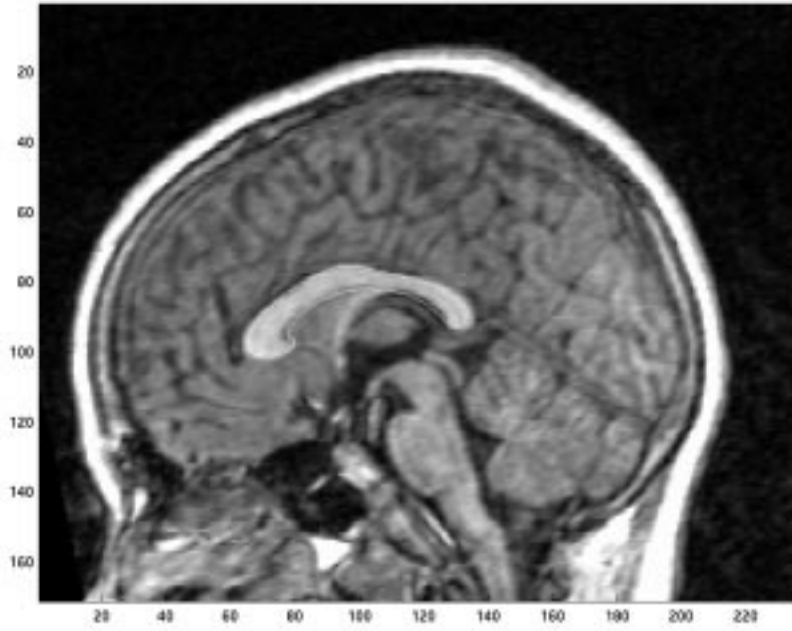


Figure 82: GVF snake: Iteration 1

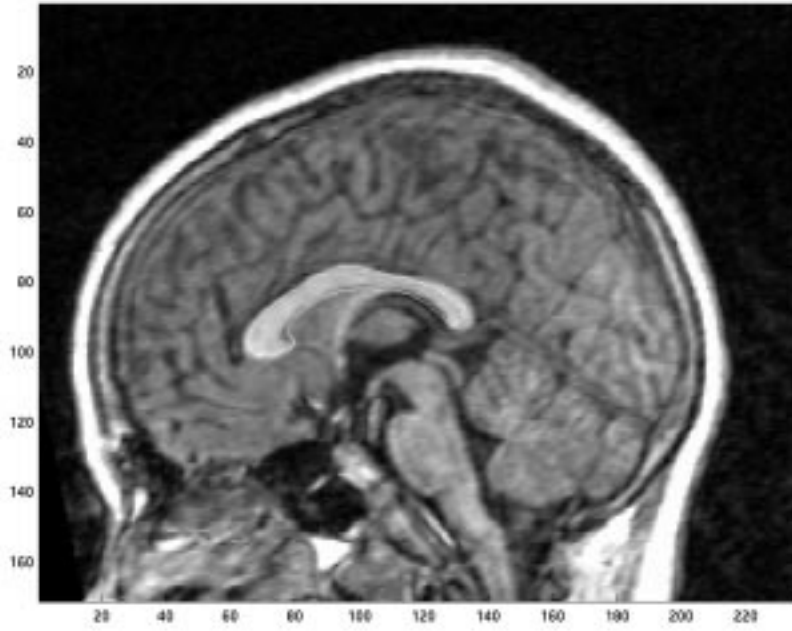


Figure 83: GVF snake: Iteration 3

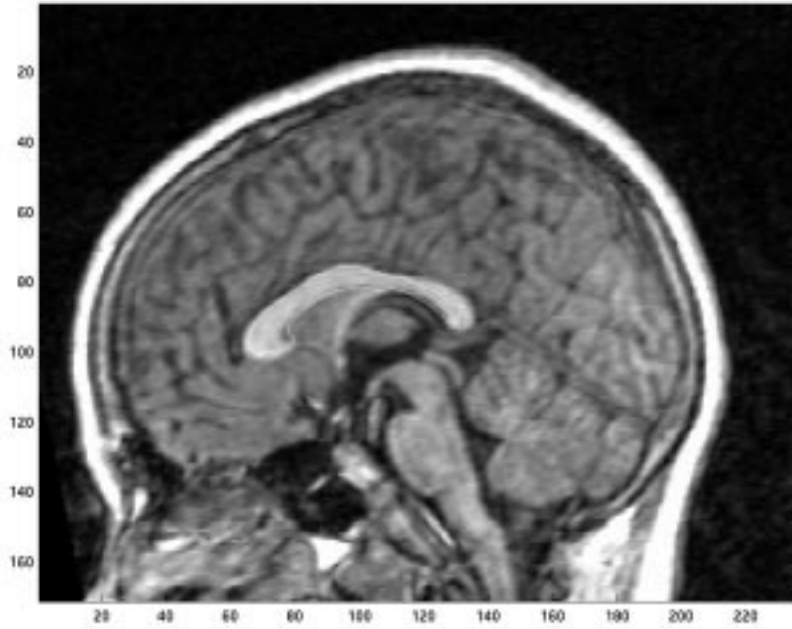


Figure 84: GVF snake: Iteration 5

Further downscaling and parameter adjusting did not produce any significant changes.

Next, the image was corrupted by additive Gaussian noise with zero mean and variance  $\sigma^2 = 1$ . Then, the original and balloon snakes were tested. The GVF snake was not tested because of the poor results obtained earlier.

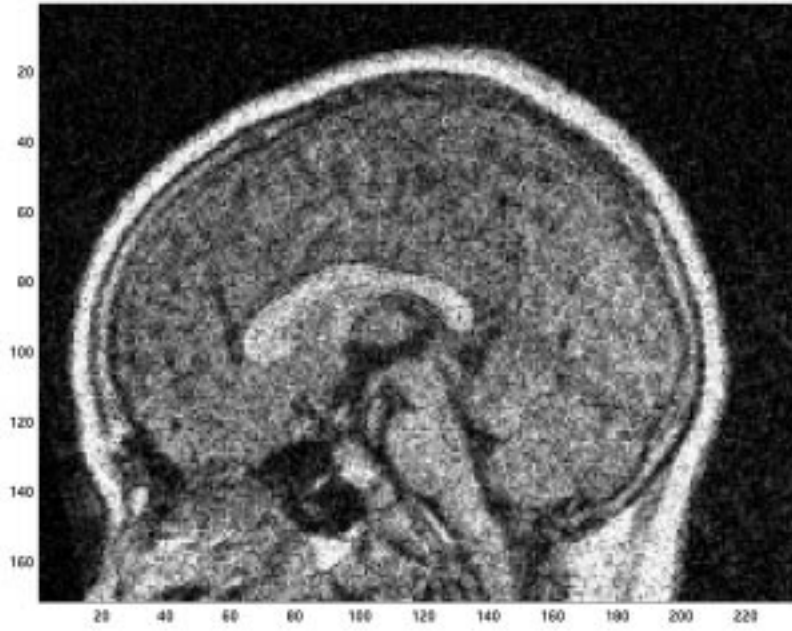


Figure 85: Original snake: Iteration 1

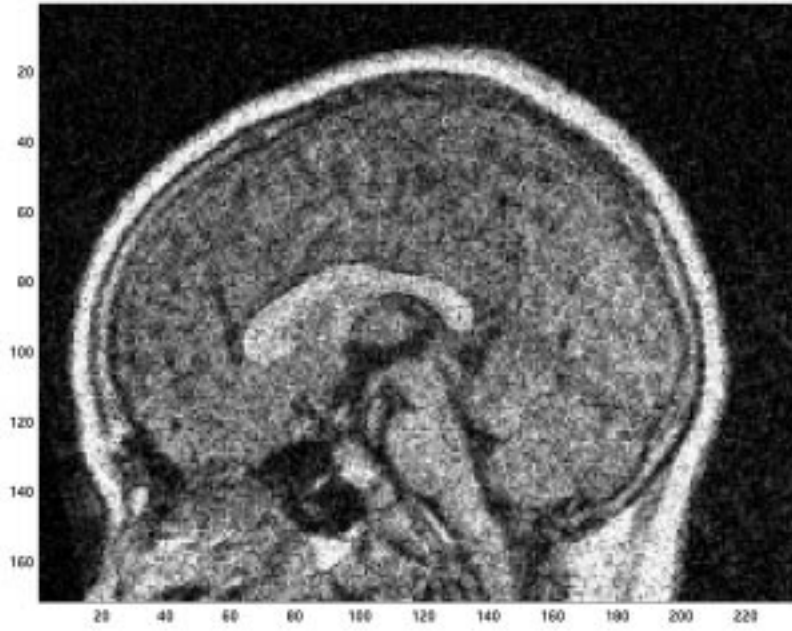


Figure 86: Original snake: Iteration 5

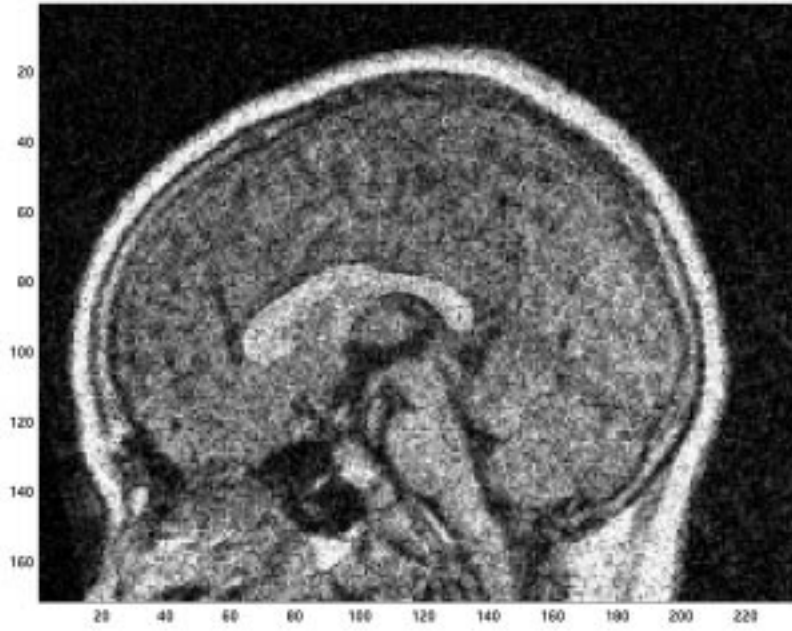


Figure 87: Original snake: Iteration 15

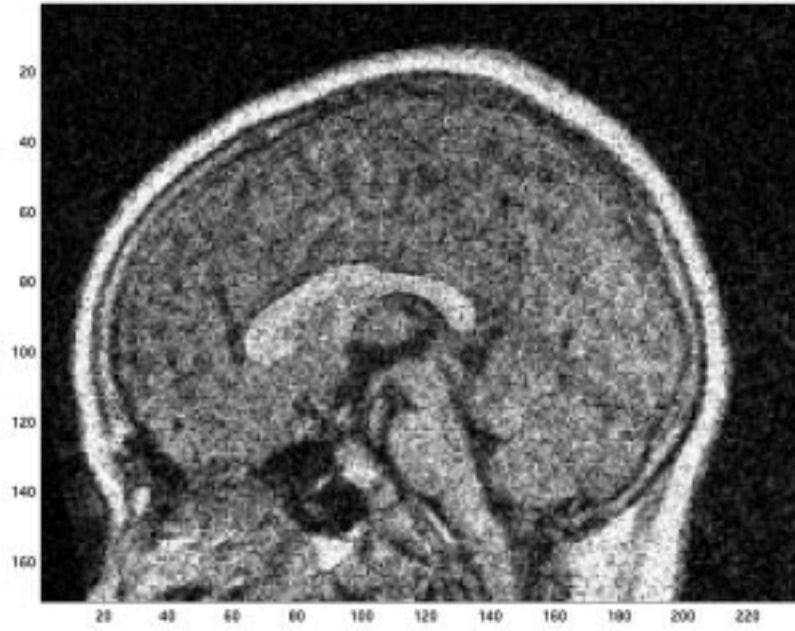


Figure 88: Original snake: Iteration 25

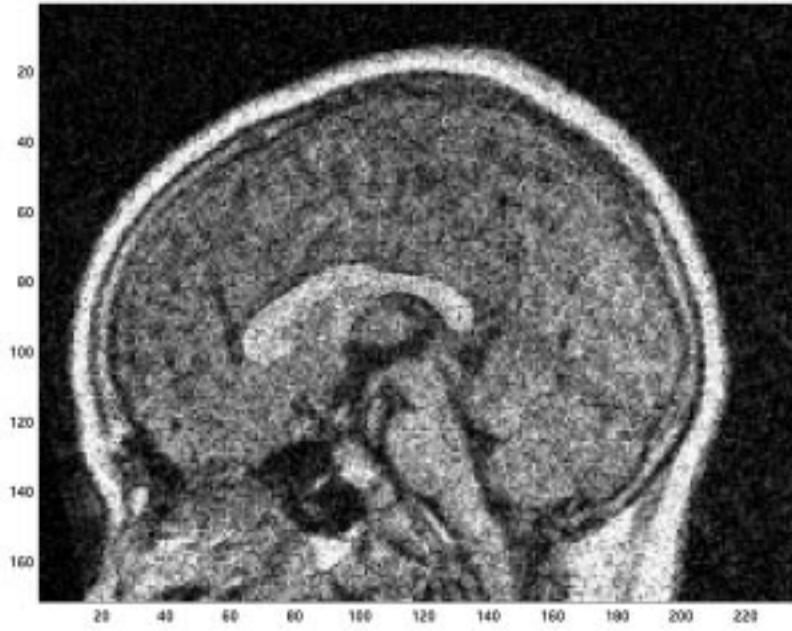


Figure 89: Original snake: Iteration 35

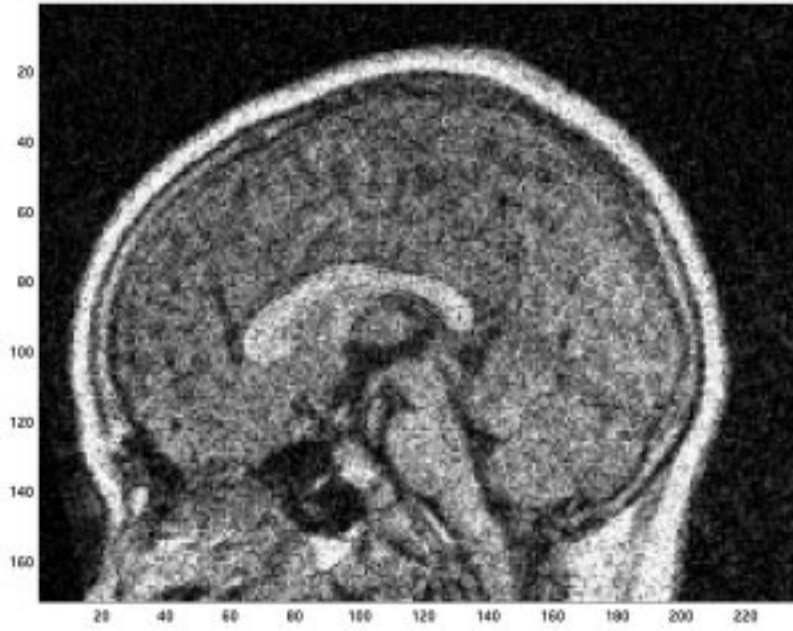


Figure 90: Balloon snake: Iteration 1

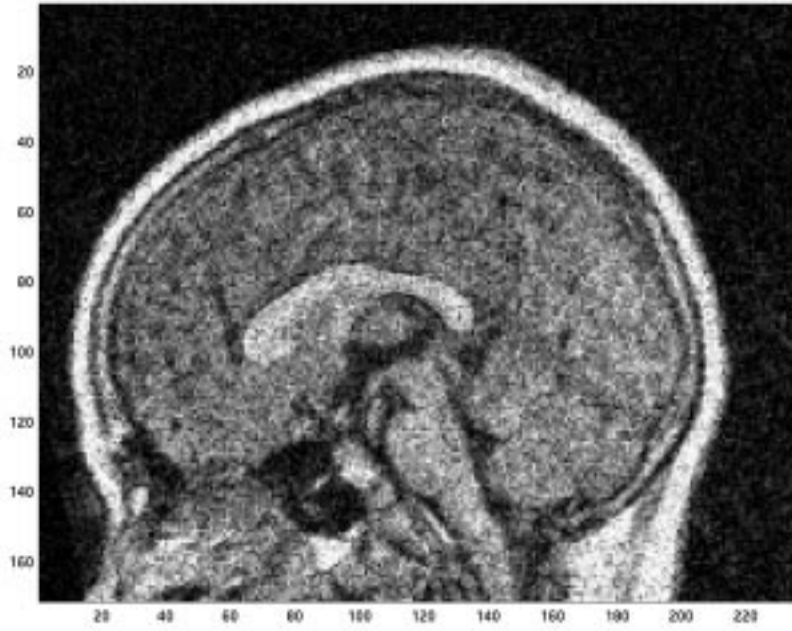


Figure 91: Balloon snake: Iteration 5

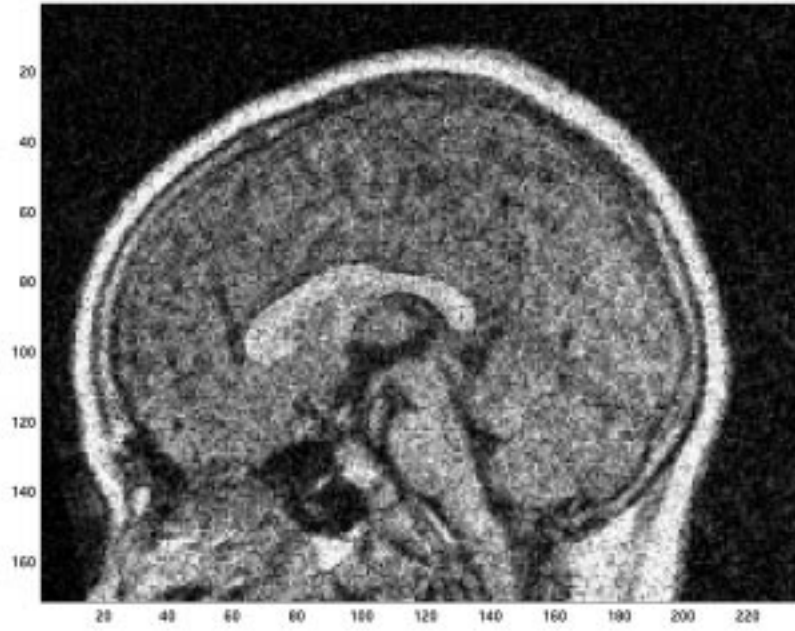


Figure 92: Balloon snake: Iteration 15

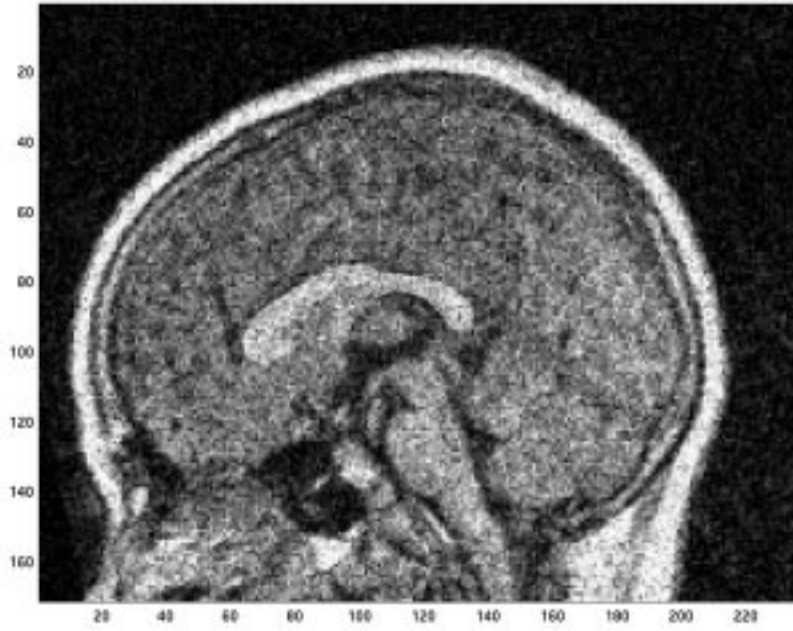


Figure 93: Balloon snake: Iteration 25

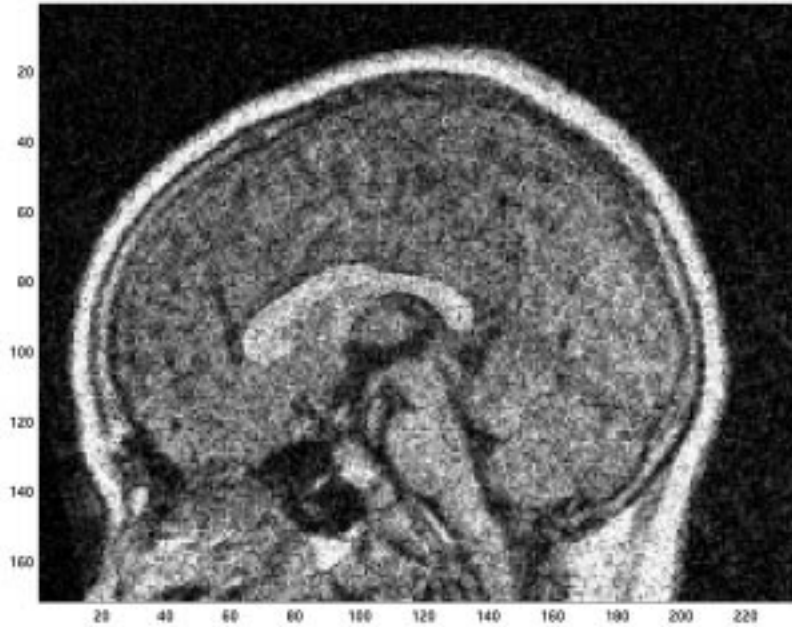


Figure 94: Balloon snake: Iteration 35

## 5.4 Discussion

When tested on an image with regular shape (ellipse), with no concavities and with no noise present in the image, the GVF snake showed the highest rate of convergence. The accuracy seems to be equal in all methods.

Tests on an image (polygon) with a concavity, showed what we had expected from the theoretical studies. Neither original or balloon snake managed to enter the concavity. The GVF snake did so in few iteration.

When tested on an image which was corrupted by noise, original and especially balloon snake were distracted but still able to extract the boundary appropriately. The GVF snake, on the other hand, became unstable and “broke” the boundary. The same did the balloon snake when the pressure force was increased or the image was blurred in order to increase the capture range of the object boundary when the model parameters were adjusted properly.

Similar results were obtained when the models were tested on real MR images. Judging the performance visually, the balloon snake achieved best accuracy. All models were numerically implemented using Fourier spectral algorithm, explained

in the next Chapter.

In the next section we will study the transition from the static minimization-of-energy-functional problem to the dynamic partial differential equation governing evolution of the snake.

## 6 Minimizing the Energy Functional

With energy functional defined as

$$\begin{aligned} E_{snake} &= \int_0^1 \{E_{int}(\mathbf{v}(s)) + E_{image}\} ds \\ &= \int_0^1 \left\{ \frac{1}{2}(\alpha |\mathbf{v}_s(s)|^2 + \beta |\mathbf{v}_{ss}(s)|^2) + E_{image} \right\} ds \end{aligned} \tag{6.1}$$

and an appropriate choice of image energy  $E_{image}$ , it remains to find the function  $\mathbf{v}(s)$  that minimizes the functional. There have been basically three approaches, through the literature, in solving this problem. One is to solve the integral with an appropriate numerical method and we will denote this approach as static. The other two dynamic approaches involve using the variational calculus to derive a dynamic system (a partial differential equation) and allow it to arrive at a minimal energy state as it achieves equilibrium and using the dynamic programming to find the position of the snake that minimizes the discretized energy. These approaches offer a variety of interesting possible behaviors of the curve that are not evident from the static energy minimization point of view. For example the curve may be guided as it moves across the image, either by adjusting the system parameters or interactively, and by studying its behavior as it moves, may reveal some possible improvements of the model. The static method can be implemented by using any of the standard numerical integration methods. We will take a closer look at the dynamical solutions.

In the next section we will shortly explain the principles of the variational calculus and derive the model equation.

### 6.1 Calculus of variations

Part of mathematics dealing with finding the maxima or the minima of values of mathematical entities called functionals is the calculus of variations [29]. Lets de-

find a couple of terms we will be using in the calculations:

(1) The variable  $z$  is called a functional depending on a function  $y(x)$ , in writing,

$$z = z(y(x)) \quad (6.2)$$

if to each function  $y(x)$  from a certain class of functions corresponds a certain value of  $v$ .

(2) The increment of the variation  $\delta y$  of the argument  $y(x)$  of a functional  $z(y(x))$  is the difference of two functions

$$\delta y = y(x) - y_1(x) \quad (6.3)$$

It is assumed that the argument  $y(x)$  runs through a certain class of functions.

(3) Two curves  $y = y(x)$  and  $y = y_1(x)$  are close or neighboring in the sense of closeness of order  $k$ , if the absolute values of the differences

$$y(x) - y_1(x), y'(x) - y_1'(x), \dots, y^{(k)}(x) - y_1^{(k)}(x) \quad (6.4)$$

are small.

(4) A functional  $v(y(x))$  is a continuous along  $y = y_0(x)$  in the sense of closeness of order  $k$ , if for arbitrary positive number  $\varepsilon$  there exists a  $\delta > 0$  such that

$$|y(x) - y_0(x)| < \delta \quad (6.5)$$

whenever

$$\begin{aligned} |y(x) - y_0(x)| &< \delta \\ |y'(x) - y_0'(x)| &< \delta \\ &\vdots \\ |y^{(k)}(x) - y_0^{(k)}(x)| &< \delta \end{aligned} \quad (6.6)$$

(5) If the increment

$$\Delta z = z(y(x) + \delta y) - z(y(x)) \quad (6.7)$$

of a functional is of the form

$$t\Delta z = L(y(x), \delta y) + \gamma(y(x), \delta y) \max|\delta y| \quad (6.8)$$

where  $L(y(x), \delta y)$  is a linear functional in  $\delta y$ ,  $\max|\delta y|$  is the maximal value of  $|\delta y|$  and  $\gamma(y(x), \delta y) \rightarrow 0$  whenever  $\max|\delta y| \rightarrow 0$ , then the linear part of this increment that is linear in  $\delta y$  is called the variation of the functional, and is denoted by  $\delta z$ . Now we state a theorem and a lemma without proofs.

**Theorem 1** *If the variation of a functional  $z(y(x))$  exists, and if  $z$  takes on a maximum or a minimum along  $y = y_0(x)$ , then*

$$\delta z = 0 \quad (6.9)$$

along  $y = y_0(x)$ .

**Lemma 1** *If a function  $\Phi(x)$  is continuous on an interval  $(x_0, x_1)$  and if*

$$\int_{x_0}^{x_1} \Phi(x)\eta(x)dx = 0 \quad (6.10)$$

for an arbitrary function  $\eta(x)$  subject to some conditions of general character only, then  $\Phi(x) \equiv 0$  throughout the interval  $x_0 < x < x_1$ .

## 6.2 Computation of the Snake Evolution Equation

We now examine the extrema of the functional

$$E_{snake}(\mathbf{v}(s)) = \int_0^1 \left\{ \frac{1}{2}(\alpha|\mathbf{v}_s(s)|^2 + \beta|\mathbf{v}_{ss}(s)|^2) + E_{image}(\mathbf{v}(s)) \right\} ds \quad (6.11)$$

For simplicity reasons, let us denote  $\frac{1}{2}(\alpha|\mathbf{v}_s(s)|^2 + \beta|\mathbf{v}_{ss}(s)|^2) + E_{image}(\mathbf{v}(s))$  by  $F(\mathbf{v}(s), \mathbf{v}_s(s), \mathbf{v}_{ss}(s))$ , so that the functional takes the form

$$E(\mathbf{v}(s)) = \int_0^1 F(\mathbf{v}(s), \mathbf{v}_s(s), \mathbf{v}_{ss}(s)) ds \quad (6.12)$$

A necessary condition for an extremum of a functional is that its variation vanishes. We take any admissible curve  $\mathbf{v} = \mathbf{v}^*(s)$  neighboring to  $\mathbf{v} = \mathbf{v}(s)$  and we set up a one-parameter family of curves

$$\mathbf{v}(s, a) = \mathbf{v}(s) + a(\mathbf{v}^*(s) - \mathbf{v}(s)). \quad (6.13)$$

As mentioned earlier, the difference  $\mathbf{v}^*(s) - \mathbf{v}(s)$  is called a variation of the function  $\mathbf{v}(s)$  and is denoted by  $\delta\mathbf{v}$ . If we consider the values of the functional (8.12) taken along the curves of the family  $\mathbf{v} = \mathbf{v}(s, a)$  then we have a function of the variable  $a$ :

$$E(\mathbf{v}(s, a)) = \phi(a) \quad (6.14)$$

It follows that for  $a = 0$  this function takes an extremum with respect to any neighboring admissible curve. As is well known, the necessary condition that the function  $\phi(a)$  has an extremum for  $a = 0$  is that its derivative for  $a = 0$  should vanish

$$\phi'(0) = 0 \quad (6.15)$$

Since

$$\phi(a) = \int_0^1 F(\mathbf{v}(s, a), \mathbf{v}_s(s, a), \mathbf{v}_{ss}(s, a)) ds \quad (6.16)$$

we have

$$\phi'(a) = \int_0^1 \left[ F_{\mathbf{v}} \frac{\partial}{\partial a} \mathbf{v}(s, a) + F_{\mathbf{v}_s} \frac{\partial}{\partial a} \mathbf{v}_s(s, a) + F_{\mathbf{v}_{ss}} \frac{\partial}{\partial a} \mathbf{v}_{ss}(s, a) \right] ds \quad (6.17)$$

Because of the relations

$$\begin{aligned} \frac{\partial}{\partial a} \mathbf{v}(s, a) &= \frac{\partial}{\partial a} (\mathbf{v}(s) + a\delta\mathbf{v}) = \delta\mathbf{v} \\ \frac{\partial}{\partial a} \mathbf{v}_s(s, a) &= \frac{\partial}{\partial a} (\mathbf{v}_s(s) + a\delta\mathbf{v}_s) = \delta\mathbf{v}_s \\ \frac{\partial}{\partial a} \mathbf{v}_{ss}(s, a) &= \frac{\partial}{\partial a} (\mathbf{v}_{ss}(s) + a\delta\mathbf{v}_{ss}) = \delta\mathbf{v}_{ss} \end{aligned} \quad (6.18)$$

we have

$$\phi'(0) = \int_0^1 [F_{\mathbf{v}}\delta\mathbf{v} + F_{\mathbf{v}_s}\delta\mathbf{v}_s + F_{\mathbf{v}_{ss}}\delta\mathbf{v}_{ss}] ds \quad (6.19)$$

As we have remarked,  $\phi'(0)$  is called a variation of the functional and is denoted by  $\delta E$ . The necessary condition for a functional to have an extremum is that its variation should vanish  $\delta E = 0$ , i.e.

$$\phi'(0) = \int_0^1 [F_{\mathbf{v}}\delta\mathbf{v} + F_{\mathbf{v}_s}\delta\mathbf{v}_s + F_{\mathbf{v}_{ss}}\delta\mathbf{v}_{ss}] ds = 0 \quad (6.20)$$

Keeping the first term, integrating the second term by parts once and integrating the third term by parts twice, we have

$$\begin{aligned} \delta E &= [F_{\mathbf{v}}\delta\mathbf{v}]_0^1 + [F_{\mathbf{v}_s}\delta\mathbf{v}_s]_0^1 - \left[ \frac{d}{ds} F_{\mathbf{v}_{ss}} \delta\mathbf{v}_{ss} \right]_0^1 + \\ &\int_0^1 \left[ F_{\mathbf{v}} - \frac{d}{ds} F_{\mathbf{v}_s} + \frac{d^2}{ds^2} F_{\mathbf{v}_{ss}} \right] \delta\mathbf{v} ds \end{aligned} \quad (6.21)$$

Since all the admissible curves pass through the same end points, it follows that

$$\begin{aligned} \delta\mathbf{v}|_{s=0} &= \delta\mathbf{v}|_{s=1} = 0 \\ \delta\mathbf{v}_s|_{s=0} &= \delta\mathbf{v}_s|_{s=1} = 0 \\ \delta\mathbf{v}_{ss}|_{s=0} &= \delta\mathbf{v}_{ss}|_{s=1} = 0 \end{aligned} \quad (6.22)$$

and consequently

$$\delta E = \int_0^1 \left[ F_{\mathbf{v}} - \frac{d}{ds} F_{\mathbf{v}_s} + \frac{d^2}{ds^2} F_{\mathbf{v}_{ss}} \right] \delta\mathbf{v} ds = 0 \quad (6.23)$$

Applying Lemma 1 we conclude that the expression on the left side will be equal 0 when

$$F_{\mathbf{v}} - \frac{d}{ds}F_{\mathbf{v}_s} + \frac{d^2}{ds^2}F_{\mathbf{v}_{ss}} = 0 \quad (6.24)$$

This is the Euler equation for our optimization problem. Evaluating the it we get

$$\nabla E_{image}(\mathbf{v}(s)) - \frac{d}{ds}\alpha\mathbf{v}_s(s) + \frac{d^2}{ds^2}\beta\mathbf{v}_{ss}(s) = 0 \quad (6.25)$$

which equals

$$\alpha\mathbf{v}_{ss}(s) - \beta\mathbf{v}_{ssss}(s) - \nabla E_{image}(\mathbf{v}(s)) = 0 \quad (6.26)$$

This equation must be discretized and solved numerically. We will devote next section to different numerical approaches used in the literature.

## 7 Numerical Solutions to the Snake Evolution Equation

The curve that minimizes the given energy functional, must satisfy the Euler equation (8.26). In order to make solution to that equation dynamical, we treat the function  $\mathbf{v}$  as a function of time  $t$  as well as  $s$ . Then, the partial derivative of  $\mathbf{v}$  with respect to  $t$  is then set equal to the left hand side of (8.26)

$$\mathbf{v}_t(s, t) = \alpha\mathbf{v}_{ss}(s, t) - \beta\mathbf{v}_{ssss}(s, t) - \nabla E_{image}(\mathbf{v}(s, t)) \quad (7.1)$$

When the solution  $\mathbf{v}(s, t)$  stabilizes, the term  $\mathbf{v}_t(s, t)$  vanishes and we achieve a solution of (8.26).

In order to solve the equation (9.2) numerically, we discretize the equation and solve the discrete system iteratively.

### 7.1 Finite Difference Method Applied to the Snake Evolution Equation

In this chapter, the principles behind the finite difference method will be discussed and an algorithm for implementation of the snake evolution equation using implicit finite difference method will be developed [9],[1],[46],[7].

### 7.1.1 Finite Difference Methods

The main idea behind the finite difference methods is to approximate derivatives of the function using one of the finite difference operators. If we specify each of the independent variables of a two dimensional function  $\mathbf{v}(s, t)$  on a grid as

$$s_i = s_0 + i\Delta s, \quad i = 0, \dots, N_s \quad (7.2)$$

$$t_j = t_0 + j\Delta t, \quad j = 0, \dots, N_t \quad (7.3)$$

$$\mathbf{v}_{ij} = \mathbf{v}(s_i, t_j) \quad (7.4)$$

the partial derivatives of  $\mathbf{v}$  can be approximated by a finite difference expressions in the  $s$  and  $t$  directions. One can choose from forward difference operator, backward difference operator, central difference operator, averaging difference operator, etc. The ones we will use in our calculations are the backward difference approximation

$$\frac{\partial^2 \mathbf{v}}{\partial s^2} \approx \frac{(\mathbf{v}_{i+1,j} - 2\mathbf{v}_{i,j} + \mathbf{v}_{i-1,j})}{2(\Delta s)^2} \quad (7.5)$$

$$\frac{\partial^4 \mathbf{v}}{\partial s^4} \approx \frac{(\mathbf{v}_{i-2,j} - 4\mathbf{v}_{i-1,j} + 6\mathbf{v}_{i,j} - 4\mathbf{v}_{i+1,j} + \mathbf{v}_{i+2,j})}{2(\Delta s)^4} \quad (7.6)$$

and the forward difference approximation

$$\frac{\partial \mathbf{v}}{\partial t} \approx \frac{(\mathbf{v}_{i,j+1} - \mathbf{v}_{i,j})}{\Delta t} \quad (7.7)$$

Explicit finite difference method is probably simplest finite difference method but it is not well suited for the snake equation. The reason is that the method gives a useful approximation if the stability limit  $\Delta t/(\Delta s)^2 < \frac{1}{2}$ , where  $\Delta t$  is the time iteration step and  $\Delta s$  is the space iteration step, is fulfilled. However, that requirement places a severe restriction on the time interval size which results in significant increase in computation.

Implicit finite difference method avoids the problem stated above and has been the method, most frequently used through the literature.

An implicit formula is one in which one or more unknown values in the next iteration step are specified in terms of known values of in the current step. We will now develop an implicit finite difference algorithm for the snake evolution equation.

Using finite difference operators defined in equations (9.5),(9.6),(9.7) we get the following finite difference approximation of the equation (9.1)

$$\begin{aligned} \frac{\mathbf{v}_{i,j+1} - \mathbf{v}_{i,j}}{\Delta t} &= \alpha \frac{(\mathbf{v}_{i+1,j+1} - 2\mathbf{v}_{i,j+1} + \mathbf{v}_{i-1,j+1})}{2(\Delta s)^2} \\ &- \beta \frac{(\mathbf{v}_{i-2,j+1} - 4\mathbf{v}_{i-1,j+1} + 6\mathbf{v}_{i,j+1} - 4\mathbf{v}_{i+1,j+1} + \mathbf{v}_{i+2,j+1})}{2(\Delta s)^4} \\ &- (E_x^{im}(j), E_y^{im}(j)) \end{aligned} \quad (7.8)$$

where

$$\begin{aligned} E_x^{im} &= \frac{\partial E_{image}}{\partial x} \\ E_y^{im} &= \frac{\partial E_{image}}{\partial y} \end{aligned} \quad (7.9)$$

This method is called the Euler backward method. After some rearranging we get

$$\begin{aligned} \mathbf{v}_{i,j} &= \mathbf{v}_{i,j+1} - K_1(\mathbf{v}_{i+1,j+1} - 2\mathbf{v}_{i,j+1} + \mathbf{v}_{i-1,j+1}) \\ &+ K_2(\mathbf{v}_{i+2,j+1} - 4\mathbf{v}_{i+1,j+1} + 6\mathbf{v}_{i,j+1} - 4\mathbf{v}_{i-1,j+1} + \mathbf{v}_{i-2,j+1}) \\ &+ (E_x^{im}(j), E_y^{im}(j)) \end{aligned} \quad (7.10)$$

where

$$\begin{aligned} K_1 &= \frac{\alpha \Delta t}{2(\Delta s)^2} \\ K_2 &= \frac{\beta \Delta t}{2(\Delta s)^4} \end{aligned} \quad (7.11)$$

and further

$$\begin{aligned} \mathbf{v}_{i,j} &= K_2 \mathbf{v}_{i+2,j+1} + (-K_1 - 4K_2) \mathbf{v}_{i+1,j+1} \\ &+ (1 + 2K_1 + 6K_2) \mathbf{v}_{i,j+1} + (-K_1 - 4K_2) \mathbf{v}_{i-1,j+1} \\ &+ K_2 \mathbf{v}_{i-2,j+1} + (E_x^{im}(i), E_y^{im}(i)) \end{aligned} \quad (7.12)$$

This equation can be written in matrix form as

$$\begin{pmatrix} A & B & C & 0 & 0 & 0 & \cdots & 0 \\ B & A & B & C & 0 & 0 & \cdots & 0 \\ C & B & A & B & C & 0 & \cdots & 0 \\ 0 & C & B & A & B & C & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & A & B & C \\ 0 & 0 & 0 & 0 & \cdots & B & A & B \\ 0 & 0 & 0 & 0 & \cdots & C & B & A \end{pmatrix} \begin{pmatrix} \mathbf{v}_{1,j+1} \\ \mathbf{v}_{2,j+1} \\ \mathbf{v}_{3,j+1} \\ \mathbf{v}_{4,j+1} \\ \vdots \\ \mathbf{v}_{N_s-2,j+1} \\ \mathbf{v}_{N_s-1,j+1} \\ \mathbf{v}_{N_s,j+1} \end{pmatrix} + \begin{pmatrix} \nabla \mathbf{E}_{image}(1) \\ \nabla \mathbf{E}_{image}(2) \\ \nabla \mathbf{E}_{image}(3) \\ \nabla \mathbf{E}_{image}(4) \\ \vdots \\ \nabla \mathbf{E}_{image}(N_s - 2) \\ \nabla \mathbf{E}_{image}(N_s - 1) \\ \nabla \mathbf{E}_{image}(N_s) \end{pmatrix} \\
= \begin{pmatrix} \mathbf{v}_{1,j} \\ \mathbf{v}_{2,j} \\ \mathbf{v}_{3,j} \\ \mathbf{v}_{4,j} \\ \vdots \\ \mathbf{v}_{N_s-2} \\ \mathbf{v}_{N_s-1} \\ \mathbf{v}_{N_s} \end{pmatrix} \tag{7.13}$$

or as a pentagonal matrix system

$$\mathbf{M}\mathbf{V}_{j+1} + \mathbf{E}_{image} = \mathbf{V}_j \tag{7.14}$$

where

$$\begin{aligned} A &= 1 + 2K_1 + 6K_2 \\ B &= -K_1 - 4K_2 \\ C &= K_2 \end{aligned} \tag{7.15}$$

The coefficient matrix  $\mathbf{M}$  is diagonally dominant, banded and sparse. So, this equation is well suited for either iterative techniques or direct sparse matrix solutions. But more importantly, this equation can be shown to be stable for all values of  $\Delta x$  and  $\Delta t$ . So, the only restraints on the choice of step size are those dictated by accuracy and computation time consideration.

The iteration equation is then

$$\mathbf{V}_{j+1} = \mathbf{M}^{-1}(\mathbf{V}_j - \mathbf{E}_{image}) \tag{7.16}$$

## 7.2 Fourier Spectral Method Applied to the Snake Evolution Equation

In this chapter, the principles behind the Fourier spectral method will be discussed [19],[8] and an algorithm for implementation of the snake evolution equation using Fourier spectral method will be developed.

### 7.2.1 Spectral Methods

The basic idea of spectral methods for solving partial differential equations is to assume that  $u(x)$ , the solution to the equation, can be approximated by a sum of  $N + 1$  “basis functions”  $\phi_n(x)$ :

$$u(x) \approx u_N(x) = \sum_{n=0}^N a_n \phi_n(x) \quad (7.17)$$

When this series is substituted into the differential equation

$$Lu = f(x) \quad (7.18)$$

where  $L$  is the operator of the differential, the result is the so-called residual function defined by

$$R(x; a_0, a_1, \dots, a_N) = Lu_N - f \quad (7.19)$$

If the basis function individually satisfy the homogeneous boundary conditions on  $u(x)$ , then their sum will too. Therefore, the only error in  $u_N(x)$  is that it does not exactly satisfy the differential equation. Since the residual function is identically 0 for the exact solution, the challenge is to invent systematic methods for choosing the series coefficients  $a_n$  in such way that the residual function is made as small as possible. The different spectral methods differ in their method of minimizing the residual.

### 7.2.2 Fourier Spectral Method

When the boundary conditions require the solution to be spatially periodic, the solution generation procedure is normally much simpler than for non-periodic boundaries. In the periodic case, the sines and cosines of a Fourier series automatically

and individually satisfy the boundary conditions. Consequently, our only remaining task is to choose the coefficients of the Fourier series to minimize the residual function.

Consider a set of points

$$x_j = \frac{2\pi j}{N} \quad j = 0, 1, \dots, N - 1 \quad (7.20)$$

which are called collocation points. The discrete Fourier coefficients are

$$\tilde{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ikx} \quad -N/2 \leq k \leq N/2 - 1 \quad (7.21)$$

Due to the orthogonality relation

$$\frac{1}{N} \sum_{j=0}^{N-1} e^{ipx_j} = \begin{cases} 1 & \text{if } p = Nm, m = 0, \pm 1, \pm 2, \dots \\ 0 & \text{otherwise} \end{cases} \quad (7.22)$$

we have the inversion formula

$$u(x_j) = \sum_{k=0}^{N-1} \tilde{u}_k e^{ikx_j} \quad j = 0, 1, \dots, N - 1 \quad (7.23)$$

So, the polynomial

$$I_N u(x) = \sum_{k=0}^{N-1} \tilde{u}_k e^{ikx_j} \quad (7.24)$$

is the  $N$  - degree trigonometric interpolant of  $u$  at the collocation points.

Differentiation in transform space is performed by multiplying each Fourier coefficient by the imaginary unit times the corresponding wave number.

If

$$Su = \sum_{k=-\infty}^{\infty} \hat{u}_k \phi_k \quad (7.25)$$

is the Fourier series of a function  $u$ , then

$$Su' = \sum_{k=-\infty}^{\infty} ik \hat{u}_k \phi_k \quad (7.26)$$

is the Fourier series of the derivative of  $u$ .

### 7.3 A Fourier Spectral Algorithm for the Snake Evolution Equation

As it was shown in the previous section, derivative of a function in the physical space corresponds to multiplication by the imaginary unit and the wave number in the Fourier space. Transforming the internal energy terms should, therefore, not be a problem. The image energy, on the other hand, is a non-linear function. Transforming such a function and integrating it into the resulting differential equation in the Fourier space may cause some computational problems. An alternative algorithm was therefore developed. Each iteration of the snake evolution equation was split into two steps. The internal energy term was iterated in the Fourier space, transformed back to the physical space where the image energy term was added.

We start by transforming the equation

$$\mathbf{v}_t(s, t) = \alpha \mathbf{v}_{ss}(s, t) - \beta \mathbf{v}_{ssss}(s, t) \quad (7.27)$$

into Fourier domain

$$\mathcal{F} \left[ \frac{\partial \mathbf{v}}{\partial t} \right] = \alpha \mathcal{F} \left[ \frac{\partial^2 \mathbf{v}}{\partial x^2} \right] - \beta \mathcal{F} \left[ \frac{\partial^4 \mathbf{v}}{\partial x^4} \right] \quad (7.28)$$

which gives

$$\begin{aligned} \frac{\partial \tilde{\mathbf{v}}(\omega, t)}{\partial t} &= \alpha (-i\omega)^2 \tilde{\mathbf{v}}(\omega, t) - \beta (-i\omega)^4 \tilde{\mathbf{v}}(\omega, t) \\ &= -\alpha \omega^2 \tilde{\mathbf{v}}(\omega, t) - \beta \omega^4 \tilde{\mathbf{v}}(\omega, t) \\ &= -(\alpha \omega^2 + \beta \omega^4) \tilde{\mathbf{v}}(\omega, t) \end{aligned} \quad (7.29)$$

This is a first-order differential equation with respect to  $t$  and its general solution is

$$\tilde{\mathbf{v}}(\omega, t) = \mathcal{F} [\mathbf{v}(s, 0)] e^{-(\alpha \omega^2 + \beta \omega^4)t} \quad (7.30)$$

where  $\mathcal{F} [\mathbf{v}(x, 0)]$  is evaluated using FFT.

We iterate this explicit solution with a time step  $dt$  and after each step transform it back to the physical space where the image energy term is added. In the  $k$ th step the iteration equation is

$$\mathbf{v}(s, t_k) = \mathcal{F}^{-1}(\tilde{\mathbf{v}}(\omega, t_k)) - \nabla E_{image}(\mathbf{v}(s, t_k)) \quad (7.31)$$

### 7.4 Dynamic Programming Solution to the Snake Evolution Equation

We will first take a short look at the continuous dynamic programming theory [5]

### 7.4.1 Dynamic Programming: Continuous Case

Consider the functional

$$J(y) = \int_{x_0}^{x_1} F(x, y, y') dx \quad (7.32)$$

where  $y = y(x)$ . Assuming that the solution curve is unique, what we seek is the curve that yields a number  $J$ , that is smaller than the number yielded by any other admissible curve. The value of  $J$  corresponding to such a curve is called the absolute (global) minimum of the functional  $J(y)$  and the associated curve is the absolute (global) minimizer.

We will now derive the fundamental equation for dynamic programming for the problem stated above. We rewrite the equation ( ) to

$$J(y) = \int_{\xi_0}^{\xi_1} F(\xi, y, y') d\xi \quad (7.33)$$

where  $y = y(\xi)$ . Further, we introduce the optimal value function  $S(x, y)$  by

$$S(x, y) = \min_P \left[ \int_x^{\xi_1} F(\xi, y, y') d\xi \right] \quad (7.34)$$

where  $P$  are all admissible curves that connect the point  $(x, y(x))$  with the end point  $(\xi, y(\xi))$ .

We pick an  $x < \xi_1$  and then choose a small increment  $\Delta\xi$  such that  $x + \Delta\xi \leq \xi_1$ . The goal is to find the optimal infinitesimal curve over  $[x, x + \Delta\xi]$ . The principle of optimality [x] assures that such a curve will in fact be the absolutely minimizing curve for our problem.

We assume that  $y'$  and  $F$  are continuous on the interval  $[x, x + \Delta\xi]$ . If we make a first order discrete approximation to the interval in (9.34) and use the Taylor expansion of  $y(x, x + \Delta\xi)$  in the neighborhood of  $x$ , we get

$$S(x, y) \leq F(x, y, y')\Delta\xi + \mathcal{O}(\Delta\xi) + S(x + \Delta\xi, y + y'\Delta\xi + \mathcal{O}(\Delta\xi)) \quad (7.35)$$

The equality is achieved if the optimum  $y'$  is chosen on the interval  $[x + \Delta\xi, \xi_1]$ , i.e

$$S(x, y) = \min_{y'} \{ F(x, y, y')\Delta\xi + \mathcal{O}(\Delta\xi) + S(x + \Delta\xi, y + y'\Delta\xi + \mathcal{O}(\Delta\xi)) \} \quad (7.36)$$

Assuming that  $S$  is differentiable with respect to its both arguments, we can further expand the third argument on the right hand side of (9.36) as

$$S(x + \delta\xi, y + y'\delta\xi + \mathcal{O}(\delta\xi)) = S(x, y) + S_x\delta\xi + S_y y'\delta\xi + \mathcal{O}(\delta\xi) \quad (7.37)$$

Substituting (9.37) into (9.34), subtracting  $S(x, y)$  from both sides, dividing by  $\Delta\xi$ , and letting  $\Delta\xi$  approach zero yields

$$0 = \min_{y'} [F(x, y, y') + S_x + S_y y'] \quad (7.38)$$

Equation (9.38) is a partial differential equation in  $S$  and is known as the fundamental equation for dynamic programming. This equation must be satisfied if the optimal value function is to attain the absolutely minimal value of the functional.

## 7.4.2 Dynamic Programming Algorithm for Snake Evolution Equation

Discrete numerical treatment of the fundamental equation is one method for finding solutions of the optimization problem,[2] We discretize the integral in (8.11)

$$E_{snake} = \sum_{i=0}^{N-1} E_{int}(\mathbf{v}_i) + E_{image}(\mathbf{v}_i) \quad (7.39)$$

where

$$E_{int}(\mathbf{v}_i) = (\alpha|\mathbf{v}_i - \mathbf{v}_{i-1}|^2 + \beta|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}|^2)/2 \quad (7.40)$$

and  $E_{image}(\mathbf{v}_i)$  is defined by one of the methods discussed in section 9.1.1.

Minimizing the (9.39) using dynamic programming can be viewed as a discrete multistage decision process. Starting from the initial point on the contour, we choose a point from a finite set of admissible points at each of the finite set of stages  $i_0, i_1, \dots, i_{N-1}$ . However, with his approach, the discrete process introduces for numerical purposes has nothing to do with the original discrete grid upon which the contour points are initially placed. Therefore, we will use another algorithm that takes advantage of the inherent discrete nature of the problem [16], [33].

A correspondence can be made between minimization of the energy in (9.39) and the problem of minimizing a function of the form

$$E(\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{N-1}) = E_0(\mathbf{v}_0, \mathbf{v}_1) + E_1(\mathbf{v}_1, \mathbf{v}_2) + \dots + E_{N-2}(\mathbf{v}_{N-2}, \mathbf{v}_{N-1}) \quad (7.41)$$

We use now the discrete dynamic programming, with  $\mathbf{v}_i$  corresponding to the state variable in the  $i$ th decision state. We generate a sequence of functions of one variable  $S_i, i = 0, 1, \dots, N - 1$  (the optimal value function), and a minimization is

performed over a single dimension to obtain each  $S_i$ . For example, for  $n = 5$  we have

$$\begin{aligned}
S_1(\mathbf{v}_2) &= \min_{\mathbf{v}_1} \{E(\mathbf{v}_1, \mathbf{v}_2)\} \\
S_2(\mathbf{v}_3) &= \min_{\mathbf{v}_2} \{S_1(\mathbf{v}_2) + E(\mathbf{v}_2, \mathbf{v}_3)\} \\
S_3(\mathbf{v}_4) &= \min_{\mathbf{v}_3} \{S_2(\mathbf{v}_3) + E(\mathbf{v}_3, \mathbf{v}_4)\}
\end{aligned} \tag{7.42}$$

and

$$\min_{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_5} \{E(\mathbf{v}_1, \dots, \mathbf{v}_5)\} = \min_{\mathbf{v}_4} \{S_3(\mathbf{v}_4) + E(\mathbf{v}_4, \mathbf{v}_5)\} \tag{7.43}$$

In the general case the equation becomes

$$\begin{aligned}
S_k(\mathbf{v}_{k+1}) &= \min_{\mathbf{v}_k} \{S_{k-1}(\mathbf{v}_k) + E(\mathbf{v}_k, \mathbf{v}_{k+1})\} \\
&= \min_{\mathbf{v}_k} \{S_{k-1}(\mathbf{v}_k) + E_{int}(\mathbf{v}_{k-1}, \mathbf{v}_k, \mathbf{v}_{k+1}) + E_{image}(\mathbf{v}_k)\}
\end{aligned} \tag{7.44}$$

At each stage, a row in the energy matrix is calculated corresponding to the values of the optimal function  $S_k$ , and a row in the position matrix corresponding to the values  $v_k$  that minimize (9.39). Convergence is guaranteed since the configuration of the points does not change unless the energy of the contour is reduced by a new configuration.

### 7.4.3 Experimental Results

With the coefficients values  $\alpha = 0.2$  and  $\beta = 0.2$ , 50 iterations were computed and the elapsed time was measured.

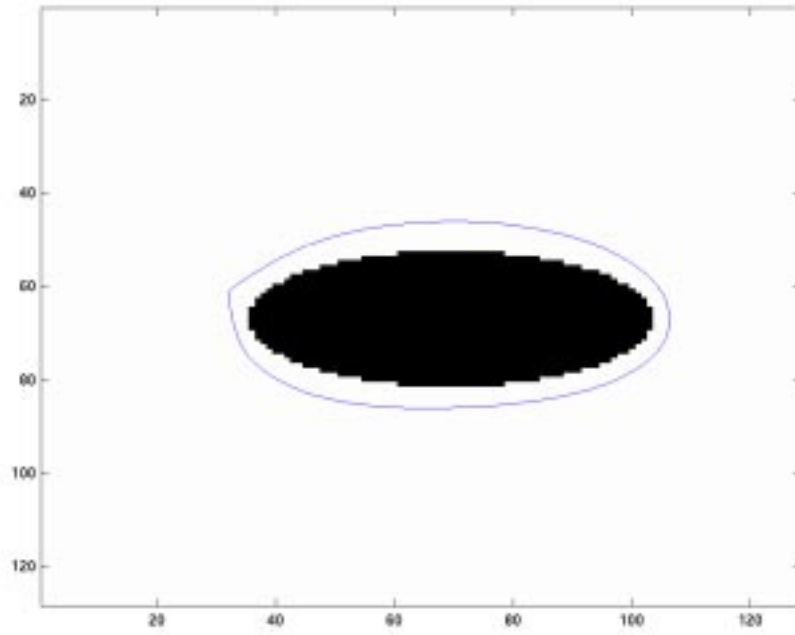


Figure 95: Implicit finite difference method: Iteration 1

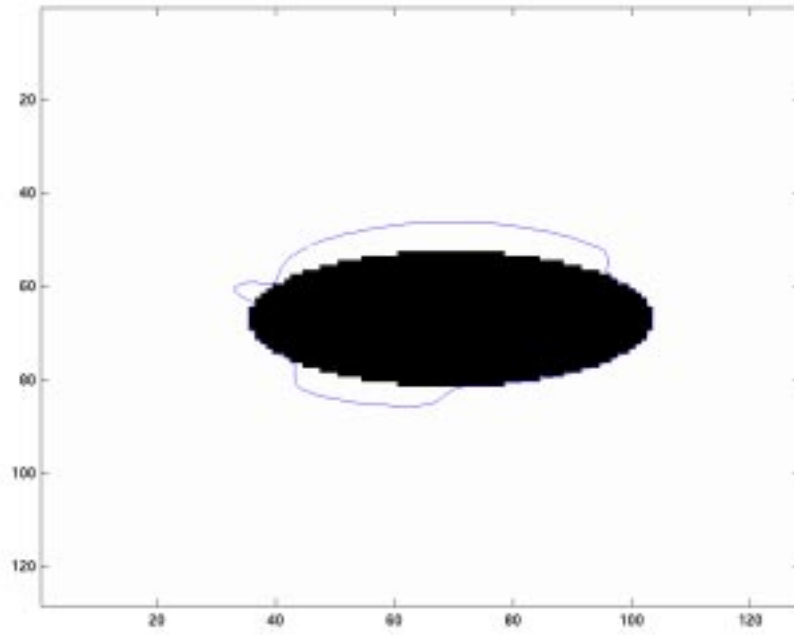


Figure 96: Implicit finite difference method: Iteration 50

The time elapsed was 45 seconds.

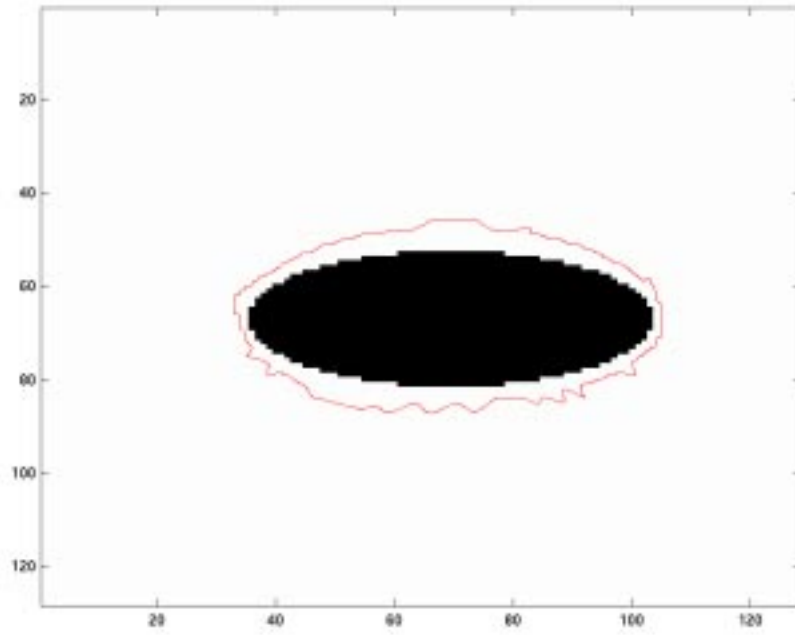


Figure 97: Dynamic programming: Iteration 1

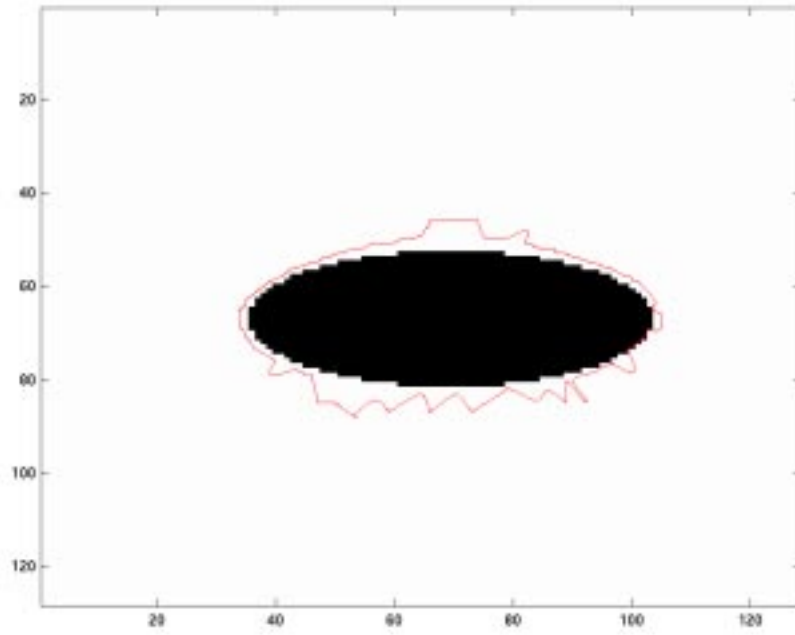


Figure 98: Dynamic programming: Iteration 3

The time elapsed was 220 seconds (note that it is only for 3 iterations).

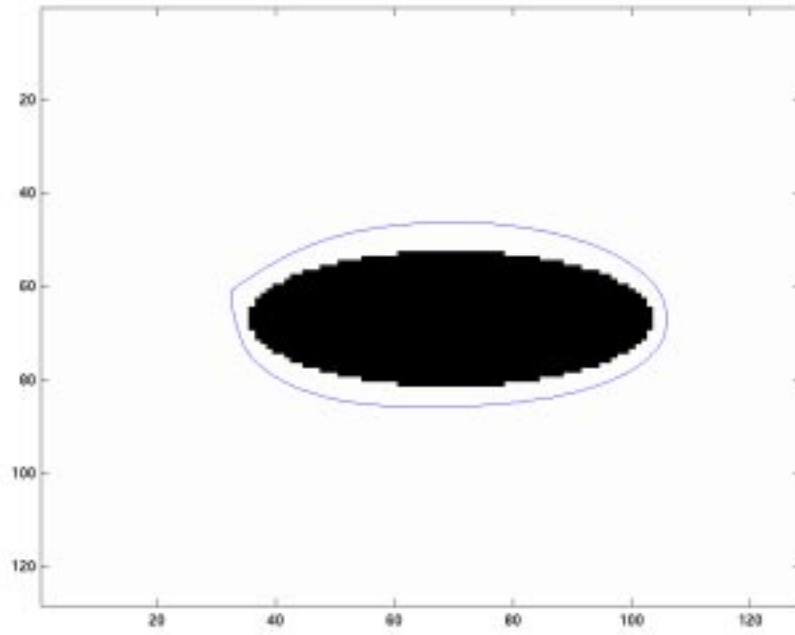


Figure 99: Fourier spectral method: Iteration 1

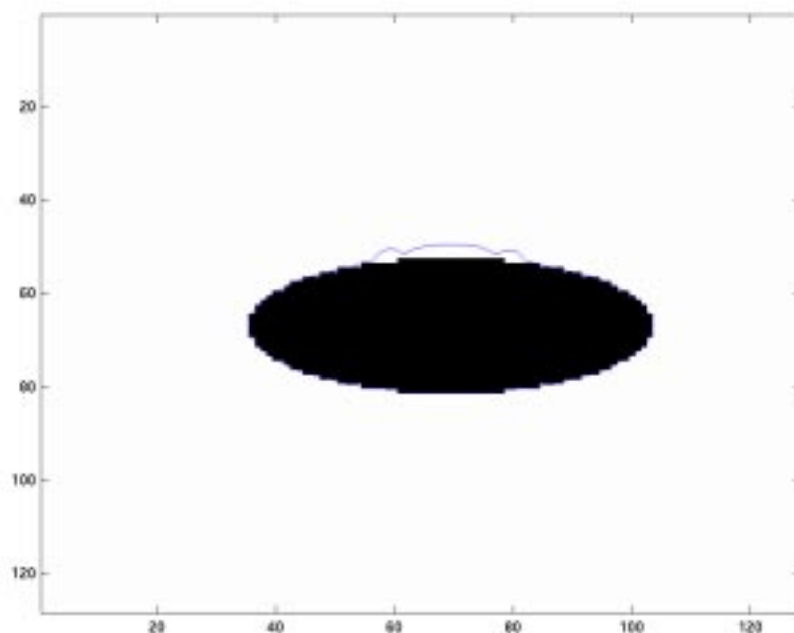


Figure 100: Fourier spectral method: Iteration 50

The time elapsed was 31 seconds.

#### 7.4.4 Comparison and Conclusion

We will start by examining properties of the dynamic programming approach. The use of dynamic programming for minimizing the energy functional in the snake model was proposed by Amini [16]. They pointed out some of the weaknesses of the classical variational approach and suggested dynamic programming as a way of overcoming these obstacles. We will take a closer look at the concerns about the classical variational model raised by Amini.

They pointed out that the Euler condition for a functional minimum is a necessary but a sufficient condition. Compared with finding the minimum of a one-dimensional function, it corresponds to the requirement that the first derivative is equal zero. If it is satisfied, the point is a function extremum, yet we do not know if it is minimum or a maximum. A way to ensure that the solution is a minimum is to that other stronger necessary conditions are satisfied. Such conditions are Jacobi condition, Legendre condition or Weierstrass condition for strong relative minimal-

ity. These conditions are often difficult to test.

The dynamic programming, on the other hand, ensures that the Legendre, Weierstrass and also the Euler condition are satisfied. All these conditions can be derived from the fundamental equation for dynamic programming.

This remark made by Amini is valid but not directly relevant for our applications. The function over which we search the minimum not the image intensity function. This function is rarely convex and both maxima and minima may arise. We search the minima over the negative edge map computed from the image intensity function. This negative edge-map function is, on the other hand, a convex function. The function may even be transformed into a binary image, where boundaries have the value 0 and rest of the image the value 1. For that reason, every extremum will be a minimum and the Euler condition is sufficient.

Another important observation is that the evolution equation is a dynamic model searching the equilibrium which is the lowest energy position. Such a model can never stay long in a maximum energy position even if it gets there. The model can be compared with a pendulum rotating in the plane vertical to the ground level around a fixed point. The pendulum will lose energy and seek the equilibrium (the point closest to the ground) when not acted upon by other forces than gravitational force and the frictional forces. Such a pendulum can not stay long in the highest potential energy point (top of the circle it describes) which is an unstable extremum for such a system.

Another concern raised by Amini was about the issue of numerical stability and accuracy. When using the variational approach, there is a need for estimates of higher order derivatives of discrete data. This process may lead to numerical instabilities. Dynamic programming on the other hand, can be directly applied to the discrete grid with no need for approximation. In addition, orders of derivatives are generally lower since functionals are directly optimized and necessary conditions are not used.

The issue of numerical instabilities is a concern one should pay attention to if for example the explicit finite difference method is used to iterate the evolution equation. If the implicit finite difference method or the Fourier spectral method is used, stability is not an issue. As we pointed out in chapters () and (), both of these methods are stable for any choice of  $\Delta x$  and  $\Delta t$ .

In addition to that, the classical variational approach provides more accurate solutions because of its higher order derivative terms. Furthermore, both the implicit

finite difference and the Fourier spectral method are significantly more time efficient. The implementation algorithms are simpler and easier to debug. A drawback these methods have compared with the dynamic programming is that they require more mathematical calculation prior to the implementation.

It remains to compare the Fourier spectral and implicit finite difference method.

Generally, the spectral methods are much more accurate than the finite difference methods. The standard central difference approximation is

$$\frac{f(x+h) - f(x-h)}{(2h)} + O(h^2) \quad (7.45)$$

As more grid points are used in the approximation, higher accuracy is obtained.

Since the spectral methods are based on evaluating the residual function only at the selected points,  $s_i$ , we can take the grid point values of the approximate solution, the set  $\mathbf{v}_N(s_i)$ , as the unknowns instead of series coefficients. Given the value of the function at  $(N+1)$  points, we can compute the  $(N+1)$  series coefficients  $a_n$  through polynomial or trigonometric interpolation.

As  $N$  is increased, the spectral method benefits in two ways. First, the interval  $h$  between grid points becomes smaller - this would cause the error to rapidly decrease even if the order of the method were fixed. Unlike the finite difference method, however, the order is not fixed. The error is of order

$$O((1/N)^N) \quad (7.46)$$

The error is decreasing faster than any finite power of  $N$  because the power in the error formula is always increasing. This is the so called exponential convergence or spectral precision.

The Fourier spectral method has also proved to be more time efficient in our simulations.

## 8 A Statistical Approach to Snakes

We have so far viewed the snake curve and its evolution model as a simulation of physical properties and processes. It has given us an intuitive understanding and tools from mathematical physics to find solutions of the encountered problems.

There is yet another useful approach to the snakes which will make some of the things we have done “make even more sense”: the Bayesian estimation.

A Bayesian approach is very useful when a priori knowledge of a process is available which needs to be combined with sensed data to make statistical inferences about the parameters of the process.

Before we start interpreting the snakes from the viewpoint of Bayesian estimation, we will define the most important theoretical notions which we will use.

## 8.1 Maximum Likelihood Estimators

Let  $\mathbf{X}$  denote a random variable whose probability density function  $f_\theta(\mathbf{x})$  is parameterized by the unknown parameter  $\theta$ . Suppose the value  $\hat{\mathbf{x}}$  is observed. There is a value of  $\theta$  for which the value of  $\hat{\mathbf{x}}$  is more probably observed than for any other. We call this value of  $\theta$  the maximum likelihood estimate and denote it by  $\hat{\theta}$  [44]. The function

$$l(\theta, \hat{\mathbf{x}}) = f_\theta(\hat{\mathbf{x}}) \quad (8.1)$$

is called the likelihood function and the function

$$L(\theta, \hat{\mathbf{x}}) = \ln f_\theta(\hat{\mathbf{x}}) \quad (8.2)$$

is the log-likelihood function. Let us now assume that the random variable  $\mathbf{x}$  and  $\theta$  are jointly distributed according to the joint density function  $f(\mathbf{x}, \theta)$ . We write it as follows

$$f(\mathbf{x}, \theta) = f(\theta|\mathbf{x})f(\mathbf{x}) \quad (8.3)$$

where  $f(\theta|\mathbf{x})$  is the posteriori density of  $\theta$  and  $f(\mathbf{x})$  is the marginal density of  $\mathbf{x}$

$$f(\mathbf{x}) = \int f(\mathbf{x}, \theta)d\theta \quad (8.4)$$

The log-likelihood function is

$$L(\theta, \hat{\mathbf{x}}) = \ln f(\hat{\mathbf{x}}, \theta) = \ln f(\theta|\hat{\mathbf{x}}) + \ln f(\hat{\mathbf{x}}) \quad (8.5)$$

When this function is continuously differentiable in  $\theta$ , the maximum likelihood estimate may be determined by differentiating the log-likelihood function.

$$\frac{\partial}{\partial \theta} L(\theta, \hat{\mathbf{x}}) = \frac{\partial}{\partial \theta} \ln f(\theta|\hat{\mathbf{x}}) = \mathbf{0} \quad (8.6)$$

The maximum likelihood is often called the maximum a posteriori probability estimate of  $\theta$ .

## 8.2 Snake Within the Bayesian Framework

We will now define the entities in the snake model and show that they can be derived from knowledge about the observation model. We consider both the contour parameters and the observation model parameters to be unknown.

A discrete matrix representation of the energy functional (8.1) can be written as

$$E(\mathbf{v}) = \frac{1}{2} \mathbf{v}^T \mathbf{K} \mathbf{v} + P(\mathbf{v}) \quad (8.7)$$

where  $\mathbf{K}$  is called the stiffness matrix and  $P(\mathbf{v})$  is the discrete version of the image potential. We can now interpret the snake equation within the Bayesian framework. The posterior density  $p(\mathbf{v}|\mathbf{I})$  of the unknown boundary  $\mathbf{v}$  conditioned on the image data  $\mathbf{I}$  is computed using Bayes rule

$$p(\mathbf{v}|\mathbf{I}) = \frac{p(\mathbf{I}|\mathbf{v})p(\mathbf{v})}{p(\mathbf{I})} \quad (8.8)$$

with the normalizing denominator

$$p(\mathbf{I}) = \sum_{\mathbf{v}} p(\mathbf{I}|\mathbf{v}) \quad (8.9)$$

The prior model  $p(\mathbf{v})$  is a probabilistic description of the information we have prior to imaging. The sensor model  $p(\mathbf{I}|\mathbf{v})$  is a description of the relationship between the unknown state  $\mathbf{v}$  and the input image  $\mathbf{I}$ . Bayes rule combines these two probabilistic models into a posterior model  $p(\mathbf{v}|\mathbf{I})$  which describes the probability of the  $\mathbf{v}$  being the best estimate given the image data  $\mathbf{I}$ .

The next step is integrating the knowledge of internal snake energy into the probabilistic model. This is done using Gibbs (Boltzmann) distribution of the form

$$p(\mathbf{v}) = \frac{1}{Z_p} e^{\{-E_p(\mathbf{v})\}} \quad (8.10)$$

where  $E_p(\mathbf{v})$  is a discrete version of the internal energy and  $Z_p$  is a normalizing constant. According to (10.7),  $E_p(\mathbf{v})$  is a quadratic energy of the form

$$E(\mathbf{v}) = \frac{1}{2} \mathbf{v}^T \mathbf{K} \mathbf{v} \quad (8.11)$$

which means that the prior distribution is zero-mean Gaussian with a covariance matrix  $\mathbf{P} = \mathbf{K}^{-1}$ . This prior model is now combined with a sensor model

$$p(\mathbf{I}|\mathbf{v}) = \frac{1}{Z_s} e^{\{-P(\mathbf{v})\}} \quad (8.12)$$

which is generated using Gibbs distribution.

Combining the prior (10.10) and the sensor model (10.12) using Bayes rule, we obtain the posteriori distribution

$$p(\mathbf{v}|\mathbf{I}) = \frac{p(\mathbf{I}|\mathbf{v})p(\mathbf{v})}{p(\mathbf{I})} = \frac{1}{Z}e^{\{-E(\mathbf{v})\}} \quad (8.13)$$

where

$$E(\mathbf{v}) = E_p(\mathbf{v}) + P(\mathbf{v}) \quad (8.14)$$

We note that this is the same equation as in (9.39). This shows that there is an equivalence between the physical and the statistical approach. Computing the maximum a posteriori estimate, i.e. the value of  $\mathbf{v}$  that maximizes the conditional probability  $p(\mathbf{v}|\mathbf{I})$ , provides the same result as finding the energy configuration of the snake.

There is, however, one advantage in using the statistical model. The external force field can be derived taking into account the known noise characteristic of the sensor and in that way improve the detection.

## 9 Deformable Templates

In this section we will take a closer look at the so called deformable templates for object localization and identification. Deformable templates localization is a method which integrates prior information about the shape of the object of interest in a direct manner. The knowledge about the shape is encapsulated in a sketch of the object or a parameter vector describing the shape of the object. Deformed versions of the original template are obtained by applying parametric transforms to the prototype. Among all such admissible deformations, the one that matches the object best is chosen.

There are many deformable template models proposed in the literature. They use often different approaches and are based on different theoretical backgrounds. We will now give structured introduction to principles behind these models and try to determine which ones are applicable in medical imaging.

## 9.1 Principles Behind Deformable Templates

### 9.1.1 Representation of Deformable Templates

One way of representing deformable templates is by a suitable parameterization such as Fourier series or spline parameterization. The advantage is shorter computational time since we use few parameters to represent a whole boundary.

As we discussed in chapter , there are several parameterization models one can choose among. Each parameterization has properties that suit it for different purposes. It is important to choose a parameterization which can express such a class of shapes which can capture the shape we want to localize. For fitting purposes, it is important that there is one-to-one, continuous mapping between the shape and its parameterization if we want to operate only in parameter space. It is also desirable to choose such a parameterization that produces a concise parameter vector because that determines the complexity of the optimization process.

Both Staib et.al. [47] and Figueiredo [23] used Fourier Parameterization, with applications in medical imaging. Yuille [18] have used circles and parabolic curves to draw eye and mouth templates, and the parameters that control the shape of a template are the center and the radius of the circle and the characteristic parameters of the parabola. In Point Distribution Model (PDM), shapes are represented using a labeled set of points that are carefully placed at the strategically important places. It is important choose an appropriate set of points which one can reproducibly place on different samples. Cootes [25] used this model for locating structures in medical images and the parameter set was the set of labeled points.

Another way of representing templates is by a sketch in the form of a bitmap with e.g. bright pixels (gray level of 255) lying on the contour and dark pixels (grey level of 0) elsewhere. Such a scheme captures the global structure of a shape without specifying its parametric form and was used by Jain [17].

### 9.1.2 Model Definition Framework

When studying snakes, we showed that the snake model can be represented both with analogy to physics and within the Bayesian estimation framework. The former one has been most frequently used through the literature.

In the same way, deformable templates can be represented within both frameworks. Chuang [10] proposed a potential-based approach for shape matching. The

matching process involves minimization of a scalar potential function. The proposed potential model assumes that the border of any 2D region is uniformly charged. If a shape template is small in size and it can be placed inside a region which is to be located, the template will experience repulsive force and torque arising from the potential field. The matching is done by translating, re-orienting and scaling the template along the above force and torque directions toward the configuration of the lowest potential. The template is expanded and its position re-adjusted until the template almost touches the border of the given region. In this work, the Newtonian potential model was adopted where the potential is inversely proportional to the distance between any two point charges. This method is computationally efficient since the main computation associated with the minimization of the potential is the calculation of the repulsive force and torque between the shape contours. These quantities can be derived analytically, avoiding expensive numeric implementation of the discretization of the shape contours. Nevertheless, the model allows only rigid transformations (scaling, translation and rotation) and requires a template with an exact shape of the object of interest. Because of the complexity and high local variability of the medical images, it is difficult to produce such templates. The model, is therefore, not very well suited for that kind of applications.

The Bayesian framework has been most frequently used to model the matching process. This approach is very useful when we wish to combine prior knowledge with sensed data to make statistical inferences about the parameters of the process. This approach has been very popular through the literature because of its capability to integrate low-level image analysis and high-level decision tasks. We will now review some of the notions from Bayes estimation theory.

The goal is to develop a classification strategy, based on classifiers which attempt to integrate all available problem information, such as measurements and a priori probabilities [43]. Decision rule may be formulated in several interrelated ways

- By converting an a priori class probability  $P(\mathbf{v})$  into a measurement conditioned (a posteriori probability)  $P(\mathbf{v}|\mathbf{I})$
- By formulating a measure of expected classification risk and choosing a decision rule that minimizes this measure

We will use the first approach in deformable template matching.

The prior estimate of the probability of a certain class or event is converted to a posteriori or measurement conditioned probability using the Bayes rule

$$P(\mathbf{v}|\mathbf{I}) = \frac{P(\mathbf{I}|\mathbf{v})P(\mathbf{v})}{P(\mathbf{I})} \quad (9.1)$$

where

$$P(\mathbf{I}) = \sum_i P(\mathbf{I}|\mathbf{v})P(\mathbf{v}) \quad (9.2)$$

Intuitively, classification which minimizes the error is the one that maximizes the posteriori probability  $P(\mathbf{v}|\mathbf{I})$ .

This formulation assumes knowledge of  $P(\mathbf{I}|\mathbf{v})$  and  $P(\mathbf{v})$ . Generally, there are two approaches in calculating these two expressions. If we assume that the parameters describing  $\mathbf{v}$  are fixed but unknown, the maximum likelihood approach gives the solution. The ML approach seeks the 'best' parameter estimate in the sense that 'best' means the set of parameters that maximize the probability of obtaining the given set. (see Chapter 10). Bayesian estimation models the parameters to be estimated as random variables with some (assumed) known a priori distribution. This approach uses so called training sets which are image sets similar to the image a hand, and they are used to collect information about the object which is to be segmented. The Bayesian approach uses the training set to update the training set conditioned density function of the unknown parameters. The parameters in the deformable template model are random, which means that the Bayesian approach is the one we will use.

Depending on what kind of information we have, and which form does that information have, there are several ways of generating a priori probabilities in the deformable template model. To simplify, discussion we will assume that the template is given parametrically by the parameter  $\mathbf{p}$ ,

- The probability distributions associated with the parameters are used to bias the model toward a particular range of shapes based on the prior knowledge. This prior knowledge comes from experience with a sample of images if it is available. When that kind of information is not available, uniform distributions are used for the prior probabilities
- If a sample is available, the prior probability distribution can be estimated from the shapes determined from the sample by decomposing the boundaries

into their model parameters and collecting statistics. In order to calculate these statistics, the boundaries of the object must be determined either by manual segmentation or alternatively, this method can be run on a set of example images with manual initialization and uniform distribution.

- If a particular distribution is known to govern the parameters it can be used as the prior. Otherwise if mean and variance are known, an independent multivariate Gaussian distribution can be used for N parameters

$$P(\mathbf{p}) = \prod_{i=1}^N P(\mathbf{p}_i) = \prod_{i=1}^N \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(\mathbf{p}_i - \mathbf{m}_i)^2}{2\sigma_i^2}} \quad (9.3)$$

where  $\mathbf{m}_i$  is the mean of  $\mathbf{p}_i$  and  $\sigma_i^2$  is the variance.

Gaussian is the natural form for a probability density within the information theory. The reason is that among the probability densities with a given variance, the Gaussian is the one with the maximum entropy. The entropy is the average information per information output. As its magnitude increases, more information is associated with the source. Thus, the Gaussian density follows directly from knowing no information other than a mean and a variance.

### 9.1.3 Deformation Methods

In order to localize the object of interest, we have to transform the original template in such a manner that will increase the probability of matching the object by some of the obtained transformations. The transformations can be divided into rigid and non-rigid. The rigid transformations are isotropic scaling, translation, rotation, and perspective approximation which are all known from the classical computer vision literature [31],[21].

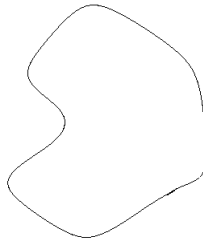


Figure 101: Original template

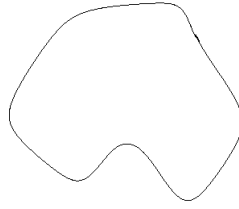


Figure 102: Rotated template



Figure 103: Scaled template

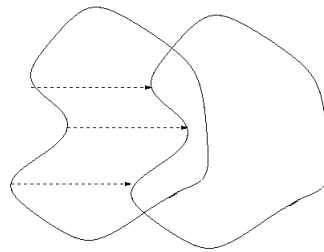


Figure 104: Translated template

The non-rigid transformations involve stretching, squeezing and twisting locally. Usually it is implemented by adding displacement offsets to the boundary points [17], [26]. The displacement offsets are realizations of the displacement functions which are spanned by an orthogonal basis.

Whether to choose rigid, non-rigid or both transformations is dependent on the applications. In traffic monitoring, the rigid transformations of the original car template will be satisfactory since the object to be localized is rigid. In our application, the anatomic structure segmentation, both rigid and non-rigid transformations are needed. There are several reasons for that.

The high complexity and local variability of the structures makes it difficult to “draw” a good original template. It is therefore desired to capture the lack of the local accuracy by local elastic deforming. A good original template is of course of

crucial importance. It is often desired to pre-process the image in order get a close match to the object of interest which can be used as the deformable template. This is called training.

In automated image segmentation and object localization, it is desired to use one template on several image sequences. In our application, one original template for corpus callosum should be used on several patients. Even though the form of anatomic structures is similar by different individuals, there are both size and form deviations from patient to patient. It is desired to use one original template and capture the differences by rigid and non-rigid transformations. For these reasons we will discuss the non-rigid deformations in more details.

There are several approaches to generating the non-rigid transformations of the original template. They usually differ in the degree of freedom. Methods with high degree of freedom are methods that allow high degree of local deformation [17].

Among the low degree methods (low meaning that the deformations are more global than local and rigid to some extent) is the work by Lakshmanan [37] who used a parametric template model to locate the road boundary in radar images and the Yuille [18] who used a template consisting of circles and parabolic curves to locate eyes and mouths in real images.

In the former work, the templates are straight two parallel lines. They are rotated and the distance between them is adjusted until they match a road. In the latter, the parabola and circle are varied until the contour matches the mouth. Varying these parameters changes the global shape of the the paraboles and circles and thereby the active contour. It does not generate local deformations in small size neighborhood of the contour points.

High degree deformations can be generated using approaches proposed by Jain [17] and [26]. Fourier and Wavelet bases, respectively, have been used to span the space of the so called deformation functions. These displacement functions  $D(x, y)$  where  $x$  and  $y$  are Cartesian coordinates of the prototype templates, are added to the contour. We will take a closer look at these methods in section 8.3. It will be shown that they can be used to generate a high degree local deformations. However, some of these deformed templates will not resemble the original template and will be useless in the applications.

In the next section, the theoretical background for another approach will be studied. It is the Principal Component Analysis and it is a method of analysis of variance. Cootes [25] used this statistical method to deform templates in the so called



The mean of the score on any given  $P_i$  is

$$\bar{P}_i = b_{i,1}\bar{x}_1 + b_{i,2}\bar{x}_2 + \dots + b_{i,m}\bar{x}_m \quad (9.7)$$

Letting

$$\tilde{x}_i = x_i - \bar{x} \quad (9.8)$$

we have

$$\begin{aligned} s_P^2 &= \frac{1}{N-1} \sum (b_{i,1}\tilde{x}_1 + b_{i,2}\tilde{x}_2 + \dots + b_{i,m}\tilde{x}_m)^2 \\ &= \mathbf{b}_i^T \mathbf{S}_x \mathbf{b}_i \end{aligned} \quad (9.9)$$

where  $\mathbf{S}_x$  is the variance-covariance matrix of the variables  $x_i$ .

We want to find  $\mathbf{b}_i$  that maximizes  $s_P^2$ . Using the method of Lagrangian multipliers, we find that, for  $\mathbf{b}_1$

$$L = \mathbf{b}_1^T \mathbf{S}_x \mathbf{b}_1 - \lambda(\mathbf{b}_1^T \mathbf{b}_1 - 1) \quad (9.10)$$

The derivative is

$$\frac{dL}{d\mathbf{b}_1} = 2\mathbf{S}_x \mathbf{b}_1 - 2\lambda \mathbf{b}_1 = 0 \quad (9.11)$$

and the system of equations we have to solve is

$$[\mathbf{S}_x - \lambda \mathbf{I}] \mathbf{b}_1 = 0 \quad (9.12)$$

This is a set of  $m$  homogeneous equations with  $m$  unknowns. We recognize this equation as an eigenvalue equation from matrix theory. In order for this set to have nontrivial solution, the determinant must be equal 0

$$|\mathbf{S}_x - \lambda \mathbf{I}| = 0 \quad (9.13)$$

Expanding this determinant with  $\lambda$  as unknown, produces a  $m$ th order polynomial in  $\lambda$  (the characteristic equation) which can be solved to yield  $m$  roots  $\lambda_i$ ,  $i = 1, 2, \dots, m$ . Inserting  $\lambda_i$  back into equation (11.12) gives a homogeneous system which can be solved for  $\mathbf{b}_i$ . With analogy to the eigenvalue problem,  $\lambda_i$  and  $\mathbf{b}_i$  are the  $i$ th eigenvalue and eigenvector.

Further, we can rewrite the equation (11.12) to get

$$\mathbf{b}_i^T \mathbf{S}_x \mathbf{b}_i = \lambda \mathbf{b}_i^T \mathbf{b}_i \quad (9.14)$$

Because of the constraint

$$\mathbf{b}_i^T \mathbf{b}_i = 1 \quad (9.15)$$

we have

$$\lambda_i = \mathbf{b}_i^T \mathbf{S}_x \mathbf{b}_i \quad (9.16)$$

It means that the eigenvectors of the covariance matrix corresponding to the largest eigenvalue describe the most significant modes of variation in the variables used to derive the covariance matrix. Moreover, the sum of variances of scores on  $m$  different  $P_i$  will be exactly equal the sum of the variances of the original variables. It means that the proportion of the total variance explained by each vector is equal to the corresponding eigenvalue.

Most of the variation can be expressed by a small number of modes  $t$ ,  $t < m$ . One method of calculating  $t$  is to choose the smallest number of modes such that the sum of their variances can explain a sufficiently large proportion of  $\lambda_T$ , the total variance of all variables

$$\lambda_T = \sum_{k=1}^m \lambda_k \quad (9.17)$$

In the next section, a short introduction and analysis of the most frequently used deformable templates will be given. We will shortly examine their properties and, based on that, try to determine if they can be used in automated medical image analysis.

## 9.2 An Overview of the Deformable Templates

### 9.2.1 The Hough Transform

The earliest example of deformable template matching is the Hough Transform for detecting lines and circles [31] which has been extended to detect arbitrary shapes [4].

The principle behind the Hough Transform is quite simple. For points in an image we want to find subsets of those points that are on straight lines. Consider a point  $(x_i, y_i)$ . The lines that pass through this point must satisfy the equation

$$y_i = ax_i + b \quad (9.18)$$

writing this equation as

$$b = -x_i a + y_i \tag{9.19}$$

and considering the  $ab$  plane (the parameter space) yields the equation of a single line for a fixed pair  $(x_i, y_i)$ . A second point  $(x_j, y_j)$  has also a line in parameter space and these two lines intersect at  $(a', b')$  where  $a'$  is the slope and  $b'$  is the intercept of the line containing both  $(x_i, y_i)$  and  $(x_j, y_j)$ . In fact, all points contained on this line have a line in parameter space that intersect at  $(a', b')$ .

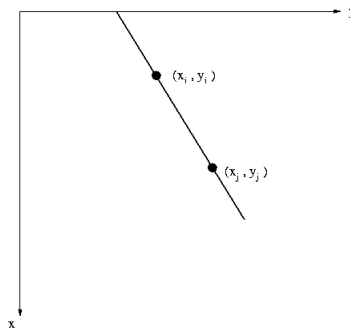


Figure 105: xy plane

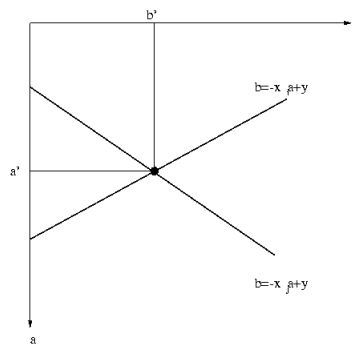


Figure 106: Parameter space

The method is implemented using the so called accumulator cells [31]

**Algorithm 1** -Quantize parameter space between appropriate maximum and minimum values for  $a$  and  $b$ .

- Form an accumulator array  $A(a, b)$  whose elements are initially zero.
- For each point  $(x, y)$  in a gradient image such that the strength of the gradient exceeds some threshold, increment all points in the accumulator array along the appropriate line, i.e.

$$A(a, b) = A(a, b) + 1 \tag{9.20}$$

-Local maxima in the accumulator array now corresponds to collinear points in the image array.

Using a similar approach, the Hough transform is applicable to any function of the form  $g(\mathbf{v}, \mathbf{c}) = 0$ , where  $\mathbf{v}$  is a vector of coordinates and  $\mathbf{c}$  is a vector of coefficients [31]. For example, the points lying on the circle

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2 \tag{9.21}$$

can be detected using the approach just discussed. []

Dramatic reductions in amount of computations can be achieved if the gradient direction is integrated. Without gradient information, all values  $c_1, c_2$  lying on the circle given by (11.21) are incremented. With the gradient information, only points on an arc centered at  $(c_1, c_2)$

$$\begin{aligned} c_1 &= x + c_3 \cos\phi \\ c_2 &= y + c_3 \sin\phi \end{aligned} \tag{9.22}$$

where  $\phi(x)$  is the gradient angle returned by an edge detector, must be incremented.

Since the HT is so closely related to template matching, and template matching can handle the curves with no simple analytical form too, the HT can be generalized to handle this case also. To see how it is done, consider locating points that lie on the curve in Fig.

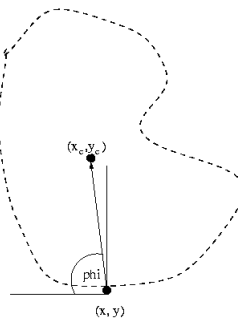


Figure 107: Geometry for generalized Hough transform

Pick a reference point in the silhouette and draw a line to the boundary. At the boundary point, compute the gradient direction and store it as a function of this direction. Then, compute the location of the reference point from boundary points given the gradient angle. The set of all such locations indexed by gradient angle comprises the R-table

Angle measured from figure boundary to reference point	Set of radii $\mathbf{r}^k$ where $\mathbf{r} = (r, \alpha)$
$\phi_1$	$\mathbf{r}_1^1, \mathbf{r}_1^1, \dots, \mathbf{r}_{n_1}^1$
$\phi_2$	$\mathbf{r}_1^2, \mathbf{r}_1^2, \dots, \mathbf{r}_{n_2}^2$
$\vdots$	$\vdots$
$\phi_m$	$\mathbf{r}_1^m, \mathbf{r}_1^m, \dots, \mathbf{r}_{n_m}^m$

Table 2:

As in the analytical case, we compute the possible loci of reference points in parameter space from edge point data and increment the parameter points in the accumulator array.

An edge point  $(x, y)$  with gradient orientation  $\phi$  constraints the possible reference points to be at

$$\begin{aligned} x_c &= x + r\phi \cos(\alpha(\phi)) \\ x_c &= y + r\phi \sin(\alpha(\phi)) \end{aligned} \tag{9.23}$$

The Hough transform is relatively unaffected by gaps in curve and noise. It is simple to implement but its application is limited because of its excessive requirement for memory and computation time especially as the number of parameters increases. In the case of detecting a line, we have two parameters, in the case of circle three, but in the case of detecting a shape with no analytical expression, there are as many parameters as number of points to match. Biologic anatomic structures have usually complex shapes with no analytical expression. The Hough transform is, therefore, not well suited for segmenting anatomic structures.

As it was mentioned earlier, boundary templates with more degrees of freedom were proposed by Staib [47] to detect objects in medical images and [15] used both boundary strength information and region homogeneity to match the templates. Cootes [25] proposed an approach which is based on learning shapes from training data. He used a point distribution model to describe the curve.

All these models are parametric, in the sense that few parameters are used to describe the contour. As in the case of snakes, the applicability of the  $p$  parametric deformable models is limited because the shapes under investigation have to be rather smooth and well-defined if few parameters are used to capture the shape. In many segmenting anatomic structures, this is often not the case.

These parametric models are, however, often used in the literature and give satisfactory results in a number of applications. We will, therefore, take a short look at the principles behind them.

### 9.2.2 Parametrically Deformable Models

When referring to parametrically deformable models, two notions are often confused with one another. All deformable template models are parametric in the sense that we use a set of parameters to deform the original template (scaling factor, angle of rotation, local deformation parameters such as number of frequencies in the Fourier series based deformation). On the other hand, not all deformable templates are parametric in the sense that the original template is represented in the parametric form (spline nodes or Fourier descriptors). One of these was developed by Staib [47] and applied to medical image segmentation. We will take a short look at the idea behind this model.

The shapes are parameterized using Fourier descriptor parameterization

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} a_0 \\ c_0 \end{bmatrix} + \sum_{k=1}^N \begin{bmatrix} a_k & b_k \\ c_k & d_k \end{bmatrix} \begin{bmatrix} \cos(kt) \\ \sin(kt) \end{bmatrix} \quad (9.24)$$

which is the the Fourier series representation discussed in section 4.5. The model was fitted to the images by optimization in the parameter space.

Let  $I(x, y)$  denote the data in an image, and  $t_p(x, y)$  is an image template corresponding to a particular value of the parameter vector  $p$ . What we want, in terms of probabilities, is to evaluate the probability of the template given the image, denoted by  $P(t_p|I)$ , and find the maximum over  $p$ . This will give us the most probable  $t_p$  based on the prior information and the image information. Using Bayes rule, we have

$$P(t_{mas}|I) = \max_p P(t_p|I) = \max_p \frac{P(I|t_p)P(t_p)}{P(I)} \quad (9.25)$$

Where  $t_{mas}$  is the maximum a posteriori solution,  $P(t_p)$  is the prior probability of the template and  $P(I|t_p)$  is the conditional probability of the image given a tem-

plate.  $P(I)$  is the prior probability of image data and it is equal for all  $p$ . Therefore, we can eliminate it and calculate the log-likelihood

$$L(I, t_{mas}) = \max_p L(I, t_p) = \max_p [\ln P(t_p) + \ln P(I|t_p)] \quad (9.26)$$

The function  $L$  is the general form of the objective function that will be optimized to find the maximum a posteriori solution.

What we need to calculate, is the likelihood  $P(I|t_p)$ . If we consider the image  $I$  to be noise corrupted version of one of these templates with noise that is independent and additive

$$I = t_p + n \quad (9.27)$$

then

$$P(I|t_p) = P(I = t_p + n|t_p) = P(I = t_p + n) = P(n = I - t_p) = P(n) \quad (9.28)$$

The noise at each pixel is

$$n(x, y) = I(x, y) - t_p(x, y) \quad (9.29)$$

Since the noise is independent from pixel to pixel, we have

$$P(I|t_p) = P(n) = \prod_I P(P(n(x, y))) \quad (9.30)$$

We assume that (explained in section ()) the noise is Gaussian with zero mean and standard deviation  $\sigma_n$

$$P(I|t_p) = \prod_I \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(I(x, y) - t_p(x, y))^2}{2\sigma_n^2}} \quad (9.31)$$

The objective function is then

$$L(I, t_p) = \ln P(t_p) + \sum_I \ln \frac{1}{\sqrt{2\pi}\sigma_n} - \sum_I \frac{(I(x, y) - t_p(x, y))^2}{2\sigma_n^2} \quad (9.32)$$

This is the maximum a posteriori function for images with assumption of independent Gaussian noise at each pixel.

Since it is the boundaries we are interested in, we formulate the objective function in such a way that we calculate the match only along the boundary of the template instead of the whole template image: If we now assume that  $B(x, y)$  is the

image obtained by applying an edge detector to a raw image, the objective function becomes

$$L(I, t_p) = \ln P(t_p) + \sum \ln \frac{1}{\sqrt{2\pi}\sigma_n} + \left[ \sum_I B^2(x, y) + \sum_{b_p} (-2B(x, y)t_p(x, y) + t_p^2(x, y)) \right] \quad (9.33)$$

where  $b_p$  is the boundary defined by  $(v(p), y(p))$  in template  $t_p$ . Since the templates is often a binary image, it has a constant magnitude along the curve

$$L(I, t_p) = \ln P(t_i) + \sum_I \ln \frac{1}{\sqrt{2\pi}\sigma_n} - \frac{B^2(x, y)}{2\sigma_n} + \frac{1}{2\sigma_n} \sum_{b_p} (2B(x, y)k - k^2) \quad (9.34)$$

where  $k$  is the magnitude of the template point. We remove the terms that do not depend on  $p$  and  $k^2$  which does not vary significantly from template to template and get

$$L(I, t_p) = \ln P(t_p) + \frac{1}{\sigma_n^2} \sum_{b_p} B(x, y)k \quad (9.35)$$

This equation is the maximum a posteriori objective function

If we want to exploit all information present in the image, we can reformulate the boundary to be a vector quantity, where the magnitude of a boundary is one component while the direction (tangent) of the boundary is another component. The corresponding  $\mathbf{k}$  is now a function of position along the curve

$$\mathbf{k}(x, y) = k \begin{bmatrix} \frac{\partial x(\mathbf{p}, s)}{\partial s} \\ \frac{\partial y(\mathbf{p}, s)}{\partial s} \end{bmatrix} \quad (9.36)$$

The edge image  $\mathbf{B}$  is a measure of both boundary magnitude and direction. Therefore, equation (9.35) can be interpreted as vector valued and rewritten using the dot product.

### 9.2.3 Active Shape Point Distribution Model

Cootes [25] suggested an active shape model which relies on representing objects by sets of labeled points where each point is placed on a particular part of object.

By examining the statistics of the positions of the labeled points, a Point Distribution Model is derived. The model gives the average positions of the points and a description of the main modes of variation found in the training set. We will take a closer look at the mathematics behind the model.

In order to model the shape, we represent it by a set of points. The chosen points must be labeled on each shape in training set. The labelling of the points is important. Each point represents a particular part of the object or its boundary and therefore the points must be placed at the exact same locations on every training example.

The modeling method works by examining the statistics of the coordinates of the labeled points over the training set. In order to be able to compare equivalent points from different shapes, they must be aligned in the same way with respect to a set of axes. The required alignment is achieved by scaling rotating and translating the training shapes so that they correspond as closely as possible

Once a set of  $N$  aligned shapes is available, the mean shape and variability can be found. We let  $\mathbf{x}_i$  denote the vector describing the  $n$  points if the  $i$ th shape in the set

$$\mathbf{x}_i = (x_{i0}, y_{i0}, x_{i1}, y_{i1}, \dots, x_{ik}, y_{ik}, \dots, x_{i(n-1)}, y_{i(n-1)})^T \quad (9.37)$$

where  $(x_{ij}, y_{ij})$  is the  $j$ th point of the  $i$ th shape. The mean shape is calculated by

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \quad (9.38)$$

The ways in which the points tend to move together are called the modes of variation. They can be found by applying principal component analysis to the deviations from the mean. For each shape in the training set the deviation from the mean is calculated as follows

$$d\mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}} \quad (9.39)$$

The covariance matrix is calculated as follows

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^N d\mathbf{x}_i d\mathbf{x}_i^T \quad (9.40)$$

The modes of variation of the points of the shape described by eigenvectors  $\mathbf{b}_k$  are solution to the equation

$$\mathbf{S}\mathbf{b}_k = \lambda_k \mathbf{b}_k \quad (9.41)$$

where

$$\mathbf{b}_k^T \mathbf{b}_k = 1 \quad (9.42)$$

The  $k$ th vector effects the  $l$ th point in the model by moving it along a vector parallel to  $(dx_{kl}, dy_{kl})$  which is obtained from the  $l$ th pair of elements in  $\mathbf{p}_k$

$$(dkx_{k0}, dky_{k0}, \dots, dkx_{kl}, dky_{kl}, \dots, dkx_{k(n-1)}, dky_{k(n-1)}) \quad (9.43)$$

Any shape in the training set can be approximated using the mean shape and a weighted sum of these deviations obtained from the first  $t$  modes

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b} \quad (9.44)$$

where  $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_t)$  is the matrix of first  $t$  eigenvectors and  $\mathbf{b} = (b_1, b_2, \dots, b_t)$  is a vector of weights, one for each eigenvector.

The eigenvectors are orthogonal

$$\mathbf{P}^T \mathbf{P} = \mathbf{1} \quad (9.45)$$

which gives us

$$\mathbf{b} = \mathbf{P}^T (\mathbf{x} - \bar{\mathbf{x}}) \quad (9.46)$$

The above equation allows us to generate new examples of the shapes by varying the parameters  $\mathbf{b}$  within suitable limits, so the new shapes will be similar to those in the training set.

In order to generate “reasonably” deformed shapes from the mean, we have to limit each element of  $\mathbf{b}$ ,  $b_k$ . Cootes did this by examining the distribution of the parameter values required to generate the training set. He assumed the Gaussian distribution of the parameters and chose a set  $\{b_1, \dots, b_t\}$  such that Mahalanobis distance  $D_m$  from the mean is less than a suitable value  $D_{max}$

$$D_m^2 = \sum_{k=1}^t \left( \frac{b_k^2}{\lambda_k} \right) \leq D_{max}^2 \quad (9.47)$$

It remains to find a method of adjusting the model to the object of interest. The approach used in the methods discussed so far was to calculate an image potential from the image intensity. Then, an objective function is defined incorporating the image potential and the model parameters. The “best” match is achieved adjusting

the parameters until the objective function is minimized. Cootes, on the other hand, used a matching procedure proposed by Bailes and Taylor [] where the location of model points in images is improved by incorporating each point's grey-level environment into the model.

For every model point  $i$  in each image  $j$ , the profile  $\mathbf{g}_{ij}$  of length  $n_p$  (pixels) centered at the point, was extracted. Following Bailes [3], he chose to sample the derivative of grey-levels along the profile in the image and normalize it. This gives invariance to uniform scaling of the grey-levels and the addition of a constant.

If the profile runs from  $\mathbf{p}_{start}$  to  $\mathbf{p}_{end}$  and is of length  $n_p$  (pixels), the  $k$ th element of the derivative profile is

$$g_{ijk} = I_j(\mathbf{y}_{i(k+1)}) - I_j(\mathbf{y}_{i(k-1)}) \quad (9.48)$$

where  $\mathbf{y}_{ik}$  is the  $k$ th point along the  $i$ th profile

$$\mathbf{y}_{ik} = \mathbf{p}_{start} + \frac{k-1}{n_p-1}(\mathbf{p}_{end} - \mathbf{p}_{start}) \quad (9.49)$$

and  $I_j(x, y)$  is the grey-level in image  $j$  at that point.

The profile is then normalized to

$$\mathbf{g}'_{ij} = \frac{\mathbf{g}_{ij}}{\sum_{k=1}^{n_p} |g_{ijk}|} \quad (9.50)$$

For each point  $i$  we calculate a mean normalized derivative profile

$$\bar{\mathbf{g}}_i = \frac{1}{N_s} \sum_{j=1}^{N_s} \mathbf{g}'_{ij} \quad (9.51)$$

then, an  $n_p \times n_p$  covariance matrix  $\mathbf{S}_{\mathbf{g}_i}$  is calculated giving a statistical description of the expected profiles about the point.

If we have a profile model for each point, the matching involves finding a nearby region which best matches the profile model. At a particular point, we extract a derivative profile  $\mathbf{g}$  of some length  $l$  centered at the point and aligned parallel with the normal to the boundary. We run then the profile model along this sampled profile and find the point at the model best matches. In order to that best match, we calculate the fit of the model at a point  $d$  as follows

$$f_{prof}(d) = (\mathbf{h}(d) - \bar{\mathbf{g}})^T \mathbf{S}_{\mathbf{g}}^{-1} (\mathbf{h}(d) - \bar{\mathbf{g}}) \quad (9.52)$$

where  $\mathbf{h}(d)$  is a sub-interval of  $\mathbf{g}$  of length  $n_p$  (pixels) centered at  $d$ . This is the square of the Mahalanobis distance of the sample from mean grey level model and

the normally distributed data is proportional to the log of the probability of obtaining  $h(d)$  from the measured distribution of grey-levels. The value of  $f_{prof}$  decreases as the fit improves. The point of best fit is thus the point at which  $f_{prof}(d)$  is minimal.

#### 9.2.4 Deformable Template Approach to Some Specific Applications

Lakshmanan and Grimmer [37] have developed a deformable template based method for locating two straight and parallel road edges in images that are acquired from a stationary millimeter-wave radar platform positioned near ground-level. They developed a robust and completely data-driven Bayesian solution to this problem. The method is designed as following

1. Deformable template a priori models express the rigid geometric constraints regarding the location of road boundaries
2. Two parameter log-normal models embody the relationship between the backscatter characteristics of the roads and their sides and account for the noise characteristics of the radar receiver
3. A posteriori density maximization algorithms are designed. They find precise locations of road boundaries in an iterative manner.

The road detection problem is posed in the Bayesian framework where the deformable templates play the role of prior and the log-normal probability density functions play the role of a likelihood. They exploited the geometric rigidness of the object in interest and developed a priori model that is effective for the problem at hand. In the same way, the likelihood model has been carefully designed for that particular problem and it accounts for deterministic and random degradations associated with the ground-level radar imaging process. The result is the model that can be applied effectively to detection of straight and parallel road edges. It is however, incapable of detecting curved or multiple roads.

Jolly [24] proposed a specific deformable template model with applications in the domain of Intelligent Transportation Systems. They developed a model for segmentation and classification of vehicles both from stationary complex background and moving vehicles from in an image sequence. The model was formulated in a Bayesian framework.

This type of segmentation is difficult due to the complex nature of the background. The background may contain other moving vehicles strong region boundaries that do not pertain to the vehicle of interest and the vehicle of interest is moving on that complex background.

They gathered all available domain-specific knowledge and integrated it in the segmentation model in order to improve it. These domain-specific constraints were

1. the object of interest is a vehicle
2. it is located in the lane closest to the camera
3. it is moving
4. its edges in the image are well defined

In order to design prototype templates, they studied a set of images containing typical vehicles and came up with a classification which categorizes vehicles into one of the following classes: sedans, pickup trucks, hatch-backs, station wagons and vans. For each one of these vehicle classes, they constructed a prototype template as a polygon characterized by  $N$  vertices. Because of the rather well-behaved geometrical nature of the vehicle shapes and the fact that we see all vehicles from one angle, they were able to represent the templates with few vertices (up to ten).

When deforming the templates to match the variations among the vehicles, they studied the interclass variability of the vehicles and designed a set of rules on the template parameters (polygon vertices) to constraint the deformed shapes.

Furthermore, they integrated the motion information by preprocessing the images and extracting moving areas in order to make sure that the deformable template was contained within these moving areas. This information was comprised in the likelihood function which contained a motion detection term and an edge directional term.

### **9.3 Deformable Template Model Proposed by Jain et.al**

In this model, the prior shape information about the object of interest is specified as a sketch of the boundary. The prototype template is not parametrized but contains the boundary information in the form of a bitmap.

Deformed templates are obtained by applying a trigonometric parametric transform as follows: Assume that the template is drawn on a unit square  $S = [0, 1]^2$ . It

is locally deformed by mapping

$$(x, y) \mapsto (x, y) + D(x, y) \quad (9.53)$$

where

$$D(x, y) = (D^x(x, y), D^y(x, y)) = \sum_{m=1}^M \sum_{n=1}^N \frac{\xi_{mn}^x \mathbf{e}_{mn}^x + \xi_{mn}^y \mathbf{e}_{mn}^y}{\lambda_{mn}} \quad (9.54)$$

The function  $D(x, y)$  is called the displacement function and it deforms the prototype template locally. The vectors

$$\begin{aligned} \mathbf{e}_{mn}^x(x, y) &= (2\sin(\pi nx)\cos(\pi my), 0) \\ \mathbf{e}_{mn}^y(x, y) &= (0, 2\cos(\pi mx)\sin(\pi ny)) \\ m, n &= 1, 2, \dots \end{aligned} \quad (9.55)$$

are the orthogonal bases that span the space of displacement functions. The parameters

$$\xi = \{(\xi_{mn}^x, \xi_{mn}^y), m, n = 1, 2, \dots\} \quad (9.56)$$

are the projections of the displacement function on the orthogonal basis, and

$$\lambda_{mn} = \alpha\pi^2(n^2 + m^2) \quad (9.57)$$

are the normalizing constants.

This continuous displacement function preserves the connectedness of the prototype template. It also preserves the smoothness of the template when  $M$  and  $N$  are not too large.

It should be noted that the length of the deformed template varies depending on the deformation.

### 9.3.1 A Probabilistic Model of Deformation

Since not all the transformations result in a deformed template that visually resembles the prototype template, we want to bias the templates obtained as a result of these transformations. As all the prior shape information is contained in the prototype template, we assume that this template exemplifies the most likely a priori shape of the object. Small deformations (small  $\xi_{mn}$ ,  $M$  and  $N$ ) that leave templates similar to the original shape are, therefore, more likely than large displacements.

We impose now a probability density on the family of functions in () in such a way that deformations generated using small values of the parameters have largest probabilities. We assume that  $\xi_{mn}$  are mutually independent and Gaussian distributed with zero mean and variance  $\sigma^2$

$$\begin{aligned} P(\xi) &= \prod_{m,n=1}^{M,N} P(\xi_{mn}) = \prod_{m,n=1}^{M,N} \frac{1}{2\pi\sigma^2} e^{-\frac{\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{MN}} e^{-\frac{1}{2\sigma^2} \sum_{m,n} (\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2)} \end{aligned} \quad (9.58)$$

The value of  $\sigma^2$  reflects the confidence about the prototype template, with large values of  $\sigma^2$  allowing more deformation.

### 9.3.2 A Priori Probability Function

Let  $T_0$  denote the prototype template. A deformation of the template is denoted by  $T_{s,\theta,\mathbf{d},\xi}$ , where  $s$  is the scaling factor,  $\theta$  is the angle of rotation,  $\mathbf{d} = (d^x, d^y)$  is the translation vector and  $\xi$  is a set of parameters that define local transformations

$$T_{s,\theta,\mathbf{d},\xi}(x, y) = T_0\{s \cdot [(x, y) + D(R_\theta(x, y))] + (d^x, d^y)\} \quad (9.59)$$

where  $D$  is the displacement function given in () and  $R_\theta(x, y)$  is the rotation of a point by an angle  $\theta$ . Assuming that the parameters are mutually independent, we have

$$P(s, \theta, \mathbf{d}, \xi) = P(s) \cdot P(\theta) \cdot P(\mathbf{d}) \cdot P(\xi) \quad (9.60)$$

We assume further that transitions, rotations and scalings are equally possible. Such an assumption is leads to a prior distribution that depends explicitly on the parameters of local deformations while rigid deformations contribute with a multiplicative constant

$$P(s, \theta, \mathbf{d}, \xi) = \mathcal{K} \frac{1}{(2\pi\sigma^2)^{MN}} e^{-\frac{1}{2\sigma^2} \sum_{m,n} (\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2)} \quad (9.61)$$

This formulation of the a priori distribution favors deformable templates that have a geometric shape similar to the prototype, regardless of its size orientation or location.

Now, as we have a priori probability density function, we design a likelihood function that specifies the probability of observing the input image, given a deformed template at a specific position, orientation and scale.

### 9.3.3 Likelihood Function

In Chapter 10, we explained the maximum likelihood principle.

We have observed an image  $I$  and defined a priori probability for the model parameters. We wish to design a function that connects the observed image with the a priori parameter model in form of a probability function. The likelihood function proposed in this work is designed as a directional potential field. First, a potential field is computed from the positions of the pixels in the image

$$\Phi(x, y) = -e^{-\rho(\delta_x^2 + \delta_y^2)^{1/2}} \quad (9.62)$$

where  $(\delta_x, \delta_y)$  is the displacement to the nearest edge in the image and  $\rho$  is the smoothing factor that controls the degree of smoothness of the potential field. This is a function with that takes values in the interval  $[-1, 0)$ . If a point lies on the edge, the function has value  $-1$ . We would be therefore looking for the points that minimize the edge potential function. We modify this function further by adding it a directional component and design an energy function that a deformed template  $T_{s,\theta,\mathbf{d},\xi}$  to the edges in the input image

$$\mathcal{E}(T_{s,\theta,\mathbf{d},\xi}, I) = \frac{1}{n_T} \sum (1 + \Phi(x, y) \cdot |\cos(\beta(x, y))|) \quad (9.63)$$

where the summation is over all pixels on the deformed template,  $n_T$  is the number of pixels on the template and  $\beta(x, y)$  is the angle between the tangent of the nearest edge and the tangent direction of the template at  $(x, y)$ . The constant 1 is added for the purpose of making the potential positive so that it takes the values between 0 and 1.

If a boundary in the image and the template boundary agree in the direction, and position, this function will have small values, so the goal is to minimize the energy function in order to get a good match. It remains to convert this function into a priori distribution over expected shapes, with lower energy shapes being more likely. We have already studied this problem in Chapter 10 and will use the same approach as in the statistical snake case. The Gibbs distribution produces the following likelihood function

$$P(I|s, \theta, \mathbf{d}, \xi) = \alpha e^{-\mathcal{E}(T_{s,\theta,\mathbf{d},\xi}, I)} \quad (9.64)$$

where  $\alpha$  is the normalizing constant.

### 9.3.4 A Posteriori Probability Function

We use now the Bayes rule to obtain the a posteriori probability density function of the deformed template given the input image

$$\begin{aligned}
 P(s, \theta, \mathbf{d}, \xi | I) &= \\
 \frac{P(I | s, \theta, \mathbf{d}, \xi) P(s, \theta, \mathbf{d}, \xi)}{P(I)} &= \\
 C_1 e^{-\mathcal{E}(T_{s, \theta, \mathbf{d}, \xi}, I)} \prod_{m=1}^M \prod_{n=1}^N \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\pi\sigma^2}(\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2)} &= \\
 C_2 e^{-\mathcal{E}(T_{s, \theta, \mathbf{d}, \xi}, I) - \sum_{m=1}^M \sum_{n=1}^N -\frac{1}{2\pi\sigma^2}(\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2)} &
 \end{aligned} \tag{9.65}$$

where  $C_1$  and  $C_2$  are normalizing constants. We take the natural logarithms on both sides of () and get

$$\ln P(s, \theta, \mathbf{d}, \xi | I) = \ln C_2 - \mathcal{E}(T_{s, \theta, \mathbf{d}, \xi}, I) - \sum_{m=1}^M \sum_{n=1}^N \frac{1}{2\pi\sigma^2} (\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2) \tag{9.66}$$

We seek to minimize the following objective function with respect to  $s, \theta, \mathbf{d}, \xi$

$$L(T_{s, \theta, \mathbf{d}, \xi}, I) = \mathcal{E}(T_{s, \theta, \mathbf{d}, \xi}, I) + \frac{1}{2\pi\sigma^2} \sum_{m=1}^M \sum_{n=1}^N (\xi_{mn}^x{}^2 + \xi_{mn}^y{}^2) \tag{9.67}$$

## 10 Discussion

There is a considerable amount of approaches to active contour boundary estimation, as we have seen through the work. When applying these models to specific problems, one has to consider certain issues that are significant for the application at hand. Generally we evaluate methods by their accuracy, compactness, robustness etc. Since all methods have pros and cons, specific requirements of the application weight some properties more than others. In the clinical and research medical imaging, we are concerned with following issues:

- **Automation**

Large scale applications requiring processing of large amounts of data from many different patients makes automation one of the most important requirements. Full automation is still a scientific goal and none of the methods presented, can operate without some user interaction. In a situation like

that, methods that require less specific-knowledge-knowledge-based interaction than others, are preferred. It means that a method requiring specification of some control points on one image in the image set, with acceptable error-margins are preferred compared with methods requiring accurate specification of control points on several images. It is also important to note that knowledge-based-specification is a somewhat ambiguous term. Both making the medical expert specify mathematical model parameters (elasticity coefficient, deformation variance etc.) and making the technical expert specify strategically important points on an anatomic structure, are not desirable.

We have seen that the snake models require a specification of control points. It is however a visual task that anyone can be trained to execute with satisfactory results. Local specification of elasticity and rigidity parameters may require more complicated software and some specific knowledge, but global specification is easily implemented and done. Furthermore, if we have a set of images that are registered (aligned according to some coordinate or landmark system), the set of control points specified on one image can be used on the rest of the images in the set.

Most of deformable template models discussed in this work, require much more preprocessing. The most generic models, like the ones proposed by Jain [17] and Staib [47], require drawing a sketch and specifying Fourier descriptors for the original template, respectively. Jain's method is certainly the most generic one. Except for an initial sketch and tuning the variance of the Gaussian probability, which specifies the confidence about the original template, no other user interaction is required.

- **Generality**

The objects of segmentation in medical images (anatomic structures) are highly variable, as opposed to objects in traffic monitoring or other rigid motion tracking applications. Such strong and unpredictable variations in the shape of the anatomic structures sets strong generality requirements.

Free form active contours (snakes) have necessary generality. They are capable of adapting to capture any shape since the only constraints imposed on them are those of continuity and smoothness. They are not not constrained to a limited shape space.

None of the deformable template models have the generality which is comparable with that of snakes. That is of course rooted in the model nature. Deformable templates are strongly constrained by a priori shape knowledge and it is not allowed to take any arbitrary shape. We can however distinguish between deformable template models of different degrees of generality.

The model proposed by Jain [17] has a high degree of generality since only a sketch of the object of interest is required. It can be generated by visual “copying” of the shape from an image containing the object. Alternatively, we can specify control points along the boundary of the object using some interactive software (such as the control point determination MATLAB program implemented by the author) and then approximate them with a smooth curve.

The models proposed by Staib [47] and Figueiredo [23] are general in the sense that they are not hand-crafted for a specific problem. They are, however, limited because they describe shapes by limited (preferably small) number of parameters. As we have pointed out, anatomic structures are often too complex to be accurately captured by a small number of parameters.

The models that are hand-crafted for some specific problems are of course not general.

There is yet another type of models like the one proposed by Cootes [25]. These models are “tailored” for the problem at hand. They can be “tailored” for any specific problem but they can not be automatically applied without any preprocessing and training.

- **Accuracy**

When images are not too noisy and objects do not overlap, snakes have a high degree of accuracy, as it was shown in this work. As far as the implementation of the snake evolution equation is concerned, the Fourier spectral method has the best spatial accuracy. Problems arise when the object of interest overlaps with another object. Alternatively it may have a “tail”, as in the case of corpus callosum segmentation, which is a part of the object but we do not want to include it in the result. Since the shape space snake contours is not limited, the snake follows the potential function derived from the image.

This problem can be controlled to some extent by adjusting the parameters,

as it was shown earlier.

Accuracy of the deformable template models depends highly on the degree of deformation allowed by the model. Deformation models proposed by Jain [17], produces a large number of deformed templates with both local and global deformations. On the other hand, we have to impose a probability distribution on these templates to bias the templates that significantly resemble the original template since many of the deformed templates are useless from the practical point of view.

Deformation method proposed by Cootes [25] has given has shown good performance. Other models mentioned in this work, deform templates in a rigid and random way which is not always suited for medical images.

## **11 Conclusion**

The complexity and variability of anatomic structures, poses many challenging problems for computer vision community in search for a good segmentation model. Snakes have proven to be very attractive approach and have produced good results in many medical imaging applications. Deformable templates proposed by Jain [17], Cootes [25] and Staib [47] are the ones that posses properties significant for applications in large scale medical imaging applications.

However, continued development, improvement and refinement of these methods is an important research area with the goal of producing automated, accurate and robust segmentation models.

## **12 Acknowledgments**

I would like to thank my supervisors, Arvid Lundervold and Frank Melandsø, for sharing their deep understanding of image analysis and numerical mathematics with me and having the patience to supervise me.

When it comes to patience, I would like to thank Kristine Sørgaard too.

## References

- [1] W.F. Ames. *Numerical Methods for Partial Differential Equations*. Academic Press, 1977.
- [2] R. Aris. *Discrete Dynamic Programming*. Blaisdel, 1964.
- [3] D.R. Bailes and C.J. Taylor. The Use of Symmetry Chords for Expressing Grey Level Constraints. In *Proc. British Machine Vision Conference*, pages 296–305. Springer, 1992.
- [4] D.H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 13:111–122, 1981.
- [5] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [6] A. Blake and M. Isard. *Active Contours*. Springer, 1998.
- [7] G.J. Borse. *Numerical Methods with Matlab*. PWS, 1997.
- [8] J.P. Boyd. *Chebyshev and Fourier Spectral Methods*. Springer, 1985.
- [9] W. Cheney and D. Kincaid. *Numerical Mathematics and Computing*. Brooks/Cole, 1985.
- [10] J.H. Chuang. Potential-Based Approach for Shape Matching and Recognition. *Pattern Recognition*, 29:463–470, 1996.
- [11] P.G. Ciarlet and J.L. Lions. *Handbook of Numerical Analysis, Volume 5*. Elsevier, 1997.
- [12] L.D. Cohen. NOTE On Active Conctour Models and Baloons. *Computer Vision and Image Processing: Image Understanding*, 53:211–218, March 1991.
- [13] L.D. Cohen and I.Cohen. Finit-Element Methods for Active Contour Models and Baloons for 2-D and 3-D Images. *IEEE Transactions on Pattern Analysis and Machine Intellingence*, 15:1131–1147, November 1993.
- [14] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.
- [15] A. Chakraborty et.al. Deformable boundary finding influenced by region homogeneity. Available at the web site of the Yale University, March 1994.

- [16] A.A. Amini et.al. Using Dynamic Programming for Solving Variational Problems in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:855–867, September 1990.
- [17] A.K. Jain et.al. Object Matching Using Deformable Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:267–278, March 1996.
- [18] A.L. Yuille et.al. Feature Extraction from Faces Using Deformable Templates. In *Conference on Computer Vision and Pattern Recognition*, pages 104–109. IEEE, 1989.
- [19] C. Canuto et.al. *Spectral Methods in Fluid Dynamics*. Springer, 1998.
- [20] G. Szekely et.al. Segmentation of 2-D and 3-D Objects from MRI Volume Data using Constrained Elastic Deformations of Flexible Fourier Contour and Surface Models. *Medical Image Analysis*, 1:19–34, January 1996.
- [21] J.D. Foley et.al. *Computer Graphics*. Addison-Wesley, 1997.
- [22] M. Kass et.al. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1:321–331, 1987.
- [23] M.A.T. Figueriedo et.al. Adaptive Parametrically Deformable Contours. In M. Pelillo and E.R.Hancock, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition, International Workshop EMMCVPR'97, Venice, Italy, May 21-23, 1997, Proceedings*, pages 35–50. Springer, 1997.
- [24] M.P.D. Jolly et.al. Vehicle Segmentation and Classification Using Deformable Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:293–308, March 1996.
- [25] T.F. Cootes et.al. Active Shape Models-Their Training and Application. *Computer Vision and Image Understanding*, 61:38–59, 1995.
- [26] T.R. Downie et.al. A wavelet based approach to deformable templates. Available at the web site of the University of Bristol, April 1996.
- [27] Y. Amit et.al. Structural Image Restoration Through Deformable Templates. *Journal of the American Statistical Association*, 86:376–387, June 1991.

- [28] Y.Y. Wong et.al. Segmented Snake for Contour Detection. *Pattern Recognition*, 31:1669–1679, 1998.
- [29] M.J. Forray. *Variational Calculus in Science and Engineering*. McGraw-Hill Book Company, 1968.
- [30] A. Garrido and N. Perez De La Blanca. Physically-Based Active Shape Models: Initialization and Optimization. *Pattern Recognition*, 31:1003–1017, 1998.
- [31] R.C. Gonzales and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1993.
- [32] U. Grenander. *Pattern synthesis: Lectures in pattern theory*. Springer, 1976.
- [33] S.R. Gunn and M.S. Nixon. Global and Local Active Contours for Head Boundary Extraction. *Internation Journal of Computer Vision*, 30:43–54, 1998.
- [34] R.J. Harris. *A Primer of Multivariate Statistics*. Academic Press, 1975.
- [35] J.Stewart. *Multivariable Calculus*. Brooks/Cole, 1995.
- [36] N. Kiryati and D. Maydan. Calculation Geometric Properties from Fourier Representation. *Pattern Recognition*, 22:469–475, 1989.
- [37] S. Lakshmanan and D. Grimmer. A Deformable Template Approach to Detecting Straight Edges in Radar Images. *IEEE Transactions on Pattern Analysis and Machine Intellingence*, 18:438–442, April 1996.
- [38] C.S. Lin and C.L. Hwang. New Forms of Shape Invariance from Elliptic Fourier Descriptors. *Pattern Recognition*, 20:535–545, 1987.
- [39] S. Loncaric. A Survey of Shape Analysis Techniques. *Pattern Recognition*, 31:983–1001, 1998.
- [40] T. McInerney and D. Terzopoulos. Deformable Models in Medical Image Analysis: a survey. *Medical Image Analysis*, 1:91–108, March 1996.
- [41] N. Peterfreund. The Velocity Snake: Deformable Contour for Tracking in Spatio-Velocity Space. *Computer Vision and Image Understanding*, 73:346–356, March 1999.

- [42] R. Ronfard. Region-Based Strategies for Active Contour Models. *International Journal of Computer Vision*, 13:229–251, 1994.
- [43] R. Scharf. *Pattern Recognition*. John Wiley and Sons, 1992.
- [44] L.L. Scharf. *Statistical Signal Processing*. Addison-Wesley, 1991.
- [45] S. Fejes and A. Rosenfeld. Discrete Active Models and Applications. *Pattern Recognition*, 30:817–835, 1997.
- [46] G.D. Smith. *Numerical Solution of Partial Differential Equations*. Oxford University Press, 1965.
- [47] L.H. Staib and J.S. Duncan. Boundary Finding with Parametrically Deformable Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:1061–1075, November 1992.
- [48] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- [49] C. Xu and J.L. Prince. Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on Image Processing*, 7:359–369, March 1998.