2007

Segmentation of images with low-contrast edges

Matthew J. Madden
West Virginia University

Segmentation of Images with Low-Contrast Edges

Matthew J. Madden

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Tim McGraw, Ph.D., Chair
Donald Adjeroh, Ph.D.
Susan Lemieux, Ph.D.
Xin Li, Ph.D.
Aina Puce, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, WV
2007

Keywords: Image Segmentation, Active Contours, Shape prior.

ABSTRACT

Segmentation of Images with Low-Contrast Edges

Matthew J. Madden

A vast amount of the current research in medical image analysis has aimed to develop improved techniques of image segmentation. Of the existing approaches, active contour methods have proven effective by incorporating edge or region information from the image into a level set formulation. However, complications arise in images containing regions of low-contrast due to noise, occlusions, or partial volume effects, which are often unavoidable in practical applications. Incorporating prior shape information into the segmentation framework provides a more accurate and robust solution by constraining the evolving contour to resemble a target shape. Two methods are presented to incorporate a shape prior into existing active contour segmentation methods, including the edge-based geodesic active contours model and a fast update implementation of the region-based Chan-Vese model. Applying these methods to synthetic and real images demonstrates that an improved result can be obtained for images containing low-contrast edge regions.

ACKNOWLEDGEMENTS

I would like to express my deep and sincere appreciation to my advisor, Professor Tim McGraw, for his guidance and support through the course of my graduate studies at WVU. Without his advice, enthusiasm, and patience, I would not have been able to achieve my goals.

I would like to specially thank Susan Lemieux for her valuable advice, and for provoking my interest in the field of Medical Imaging.

I would also like to thank the other members of my committee, Donald Adjeroh, Xin Li, and Aina Puce, for their time, valuable suggestions, and encouragement.

Finally, I'd like to thank my colleagues, Hemalatha Sampath and Lierong Zhu for their support and discussions on a number of subjects during my graduate studies.

TABLE OF CONTENTS

Abstract: ii
Acknowledgements: iii
Table of Contents: iv
List of Tables: vi
List of Figures: vii
Chapter 1: Introduction 1
1.1 Overview 1
1.2 Motivation 12
1.3 Problem Definition 12
1.4 Contributions 13
1.5 Outline 14
Chapter 2: Background 15
2.1 Medical Image Segmentation 15
2.1.1 Active Contours 15
2.1.2 Level Set Segmentation Methods 18
2.1.3 Atlas-guided Segmentation 30
2.2 Medical Image Registration 30
2.2.1 Interpolator 31
2.2.2 Transformation 32
2.2.3 Metric 32
2.2.4 Optimization Schemes 34
2.3 Insight Segmentation and Registration Toolkit (ITK) 34
2.3.1 Overview of Segmentation in ITK 35
2.3.2 Overview of Registration in ITK 37
Chapter 3: A New Speed Function for Shape Prior Segmentation 39
3.1 Overview 39
3.2 Atlas Construction 40
3.3 Registration Algorithm 41
3.3.1 Preprocessing 42
3.3.2 Initialization 43
3.3.3 Registration Framework Components 43
3.3.4 Registration Optimization 46
3.3.5 Postprocessing 47
3.4 Geodesic Active Contour Level Set Segmentation Method 48
3.4.1 Preprocessing 48
3.4.2 A New Speed Function for Shape Prior Segmentation 49
Chapter 4: Fast Level Set Segmentation using a Shape Prior 53
4.1 Image Moments 54
4.2 Transformation 56
4.3 Fast Algorithm for Level Set Based Optimization 59
4.3.1 Initialization 59
4.3.2 Segmentation Procedure 59
4.3.3 Variations to the Segmentation Procedure 63
4.3.4 Labeling Function 64
Chapter 5: Experimental Results 65

5.1 Registration 65
5.1.1 Atlas Construction 65
5.1.2 Novel Image Registration 68
5.2 Geodesic Active Contour Level Set Segmentation Method 69
5.2.1 Synthetic Data 69
5.2.2 Real Image Data 75
5.3 Fast Level Set Segmentation using Shape Constraints 77
5.3.1 Synthetic Data 77
5.3.2 Real Data 84
Chapter 6: Concluding Remarks and Future Work 87
Appendix: 90
Bibliography: 99

LIST OF TABLES

Table 1.1. The intensity characteristics of the white matter, gray matter, and CSF tissue classes associated with T1 and T2 weighted images. 9

Table 4.1. Tilt angle θ for the different sign cases of the second order central moments. 55

Table 5.1. Parameters of the filters involved in the registration process used in registering the training data..... 66

Table 5.2. Parameters used in the ITK segmentation of the synthetic data. 71

Table 5.3. Parameters used in the ITK segmentation of the synthetic data shown in Figure 5.9 using the traditional speed function. 72

Table 5.4. Parameters used in the ITK segmentation of the synthetic data shown in Figure 5.10 using the traditional speed function. 73

Table 5.5. Parameters used in the ITK segmentation of the synthetic data using the new speed function. 74

Table 5.6. Parameters used to obtain the results shown in Figure 5.14 from the ITK segmentation of the real data shown in Figure 5.12. 76

Table 5.7. Parameters used in the segmentation of the hand image using the shape constraint..... 78

Table 5.8. Parameters used in the segmentation of the data pertaining to the segmentation of the image shown in Figure 5.19 (a) to provide the results shown in Figure 5.20. 80

Table 5.9. Parameters used in the segmentation of the synthetic image shown in Figure 5.21..... 81

Table 5.10. Parameters used in the segmentation of the data pertaining to Figure 5.22. . 82

Table 5.11. Parameters used in the segmentation of the data pertaining to Figure 5.23. . 83

Table 5.12. Parameters used in the segmentation of the real image data shown in Figure 5.25 to obtain the results shown in Figure 5.26. 85

LIST OF FIGURES

Figure 1.1. Example of the segmentation of MRI brain data into different tissue classes in which (a) the original brain image is segmented into the regions of: (b) white matter (white), gray matter (gray), and the ventricles and background (black)..... 2

Figure 1.2. The application of an external magnetic field, B_0, to a group of protons which are: (a) initially in a natural random state. (b) The application of B_0 causes precessional motion about the direction of B_0. 5

Figure 1.3. A RF pulse applied to a proton in the presence of an external magnetic field B_0 forces the spin axis of the proton from the direction of B_0 into the transverse plane. ... 6

Figure 1.4. After the RF pulse ends, the spin of the proton realigns with the direction of B_0 as indicated. 6

Figure 1.5. The MRI magnet with the axes directions indicated. The transverse plane lies in the x-y plane while the direction of B_0 is in the direction of the z-axis. 7

Figure 1.6. Example of (a) T1- and (b) T2- weighted image simulation models generated from Brainweb: Simulated Brain Database [8, 16]. 8

Figure 2.1. A 2D example of using active contours techniques to find the boundaries of a shape object in an image. (a). The initial contour (red) located inside the target object (light). (b) and (c) Intermediate steps of the evolving contour. (d) The final segmentation result. 16

Figure 2.2. A parametric representation of a curve (red) represented as a vector of points (white). 16

Figure 2.3. A 3D embedding function, ϕ, the level sets of which are indicated by the 2D contour plot found below the mesh plot. Values inside the zero level set are negative, while values outside are positive. 18

Figure 2.4. The intermediate steps of geodesic active contours segmentation. (a) The image I. (b) Smoothed image. (c) The gradient magnitude image. (d) The speed image g computed using (2.17). 22

Figure 2.5. The effects of varying the parameters of (2.17), the sigmoid function. (a) Varying α. Here, α = -4 (*dotted line*), α = -2 (*solid line*), and α = -1 (*dashed line*). (b) Varying β. Here, β = -2 (*dotted line*), β = 0 (*solid line*), and β = 2 (*dashed line*). 22

Figure 2.6. An example of the results of the geodesic active contours for the segmentation of the right ventricle. (a) The original image showing the initial embedding function ϕ_0 where $\phi_0 < 0$. (b) The final segmentation result. 23

Figure 2.7. The segmentation of the brain region of a slice of MRI rabbit brain data with weak edge characteristics. (a) A slice of the rabbit brain MRI data. (b) The speed image. (c) A poor segmentation result in which the evolving curve (white) has passed beyond the boundary of the brain as indicated by the arrows (red). 24

Figure 2.8. An example of image segmentation using an implementation of the fast update algorithm found in [61]. (a) The image I. (b) The initial embedding function ϕ_0 where $\phi_0 > 0$. (c) The final segmentation result ϕ where $\phi > 0$ after 2 sweeps of the algorithm. 27

Figure 2.9. The effect of the smoothness constraint in the segmentation of an image with a high level of noise. (a) The image I. (b) The final segmentation result, $\phi > 0$, when the smoothness constraint is not enforced. (c) The final segmentation result, $\phi > 0$, when applying the smoothness constraint. 27

Figure 3.1. The flow chart of the ITK registration framework used in the registration step of the brain extraction algorithm. Inputs I_{atlas}. I, and ϕ_{atlas} are used to obtain ϕ_0, which is then used in the segmentation step..... 42

Figure 3.2. Computing the g_{High} and g_{Low} terms. (a) The original image I is used to compute (b) the gradient magnitude image. This is then used to compute the speed terms: (c) g_{High} and (d) g_{Low}. 50

Figure 3.3. The difference between the speed terms g_{High} and g_{Low} computed as $g = g_{High} - g_{Low}$. 50

Figure 3.4. The new speed term. (a) The binary atlas image used to compute ϕ_0, and (b) the new speed term..... 51

Figure 3.5. Diagram of the segmentation application using the geodesic active contour filter in ITK. The level set input ϕ_0 is provided by the registration step as indicated in Figure 3.1. The new speed term is computed as described..... 52

Figure 4.1. An example of the transformation **T** applied to I_{shape} (upper left) used to map this shape prior to $H(\phi)$ (upper right) to obtain the result used as the shape constraint (bottom). The directions of the principal axes, computed from the image moments, are indicated in the upper two images. 58

Figure 5.1. Various slices of the training data used to construct the atlas image I_{atlas} (a) A slice of the reference training data volume, $I_{TrainingReference}$. (b) and (c) the corresponding slices from the training data, $I_{Training}$. 66

Figure 5.2. Results of registering $I_{Training}$ to $I_{TrainingReference}$. (a) A slice of the training data, $I_{Training}$. (b) The difference image between corresponding slices in $I_{TrainingReference}$ and the

unregistered $I_{Training}$. (c) The final difference image between corresponding slices in $I_{TrainingReference}$ and the registered $I_{Training}$. 67

Figure 5.3. Computing I_{atlas} (a)-(e) Several slices of $I'_{Training}$ resulting from the registration step. (f) The corresponding slices of I_{atlas} computed as the median of these registered training samples. 67

Figure 5.4. The result of the manual segmentation of I_{atlas}. (a) and (b) Slices of I_{atlas} with the respective manual segmentation results indicated (red). 68

Figure 5.5. Results of the registration of I_{atlas} to I. (a)-(c) Slices of I_{atlas} with the respective manual segmentation. (d)-(f) The corresponding slices of a novel image I with the transformed signed distance ϕ_0, where $\phi_0 < 0$, which provides the initial embedding function of the segmentation step. 69

Figure 5.6. The intermediate steps involved in computing the proposed speed term. (a) The synthetic image I. (b) The gradient magnitude image. (c) The g_{Low} speed image. (d) The g_{High} speed image. (e) The difference $g_{High} - g_{Low}$. (f) The new speed map g. 70

Figure 5.7. Segmentation results of the synthetic image using: (a) The speed map computed using (2.17). (b) The proposed speed map computed using the methods presented in section 3.4. 71

Figure 5.8. The input image data including: (a) The synthetic image I. (b) The shape image I_{shape}. 72

Figure 5.9. Intermediate steps involved in the segmentation as well as the result of using the traditional speed function found in (2.17). (a) The gradient magnitude image. (b) The speed image. (c) The initialization, $\phi_0 < 0$. (d) The segmentation result. 73

Figure 5.10. Segmentation result using a second example of the traditional speed function found in (2.17). (a) The speed image. (b) The segmentation result. 73

Figure 5.11. The segmentation result obtained using the proposed speed function and initialization ϕ_0 shown in Figure 5.9 (c). (a) The g_{Low} speed image. (b) The g_{High} speed image. (c) The new speed map g. (d) The segmentation result obtained using the parameters provided in Table 5.5 74

Figure 5.12. Input data used in the implementation of geodesic active contours segmentation. (a) The original image. (b) The speed map image of MRI rabbit brain data computed using the Sigmoid Filter provided by ITK with β = 35 and α = -5. Edge information is missing as indicated by arrows. 75

Figure 5.13. Input data used in the implementation of geodesic active contours segmentation. (a) A slice of the Rabbit brain MRI data I. (b) The gradient magnitude

image. (c) The g_{Low} speed image. (d) The g_{High} speed image. (e) The difference $g_{High} - g_{Low}$. (f) The proposed speed map g. 76

Figure 5.14. Segmentation of MRI data using: (a) A typical speed map. (b) The proposed speed map..... 77

Figure 5.15. The synthetic hand image I. 78

Figure 5.16. The segmentation result using an implementation [61]. (a) The binary segmentation. (b) Final segmentation result in which the hand is not entirely segmented from the background. 78

Figure 5.17. The image segmented using the method proposed in Chapter 4. (a) The original image I. (b) The binary image shape representation I_{shape}. (c) The binary result $\phi > 0$. (d) The segmented image I. 79

Figure 5.18. The segmentation result using the proposed method. (a) The original image I. (b) The binary image shape representation I_{shape}. (c) The binary segmentation result $\phi > 0$. (d) The segmented image. 79

Figure 5.19. The image segmentation using an implementation of [61]. (a). The synthetic brain shape image I. (b). The binary segmentation result $\phi > 0$. 80

Figure 5.20. The image segmentation of the synthetic image using the proposed method. (a). The binary image shape representation I_{shape}. (b). The binary segmentation result $\phi > 0$. (c). The segmented image. (d). The difference image computed between the segmentation result without the shape term (Figure 5.19 (b)) and the segmentation result using the shape term (Figure 5.20 (b)). 81

Figure 5.21. Segmentation results without the shape constraint. (a) The original image I with the initial segmentation $\phi_0 > 0$ indicated as the white circle. (b)-(d) The segmentation results using different values of σ_1^2 and σ_2^2 provided in Table 5.9 above.. 82

Figure 5.22. Segmentation results using the shape constraint and the region growing approach. (a) The original image I with the initial segmentation $\phi_0 > 0$ indicated as the white circle. (b) The shape image (c) The binary segmentation result $\phi > 0$. (d) The final segmentation result. 83

Figure 5.23. Segmentation results using the shape constraint along with the labeling function. (a) The region of the image to be considered in the shape comparison as indicated by L. (b) The shape image (c) The binary segmentation result $\phi > 0$. (d) The final segmentation result. 84

Figure 5.24. A real data aircraft image I. 85
Figure 5.25. The segmentation result using an implementation [61]. (a) The binary segmentation. (b) Final segmentation result. 85

Figure 5.26. The image segmented using the method proposed in Chapter 4. (a) The original image I in the regions to be used in the shape constraint as indicated by L. (b) The binary image shape representation I_{shape}. (c) The binary result $\phi > 0$. (d) The segmented image I. 86

Chapter 1: Introduction

The chapter begins by providing a brief overview of the current research topics being explored in the field of medical image processing. This is followed by a discussion of the key terminology and concepts necessary in understanding the task at hand. The motivation behind this work as well as the definition of the problem explored throughout the course of this research is then given. The chapter concludes by outlining the remainder of the thesis.

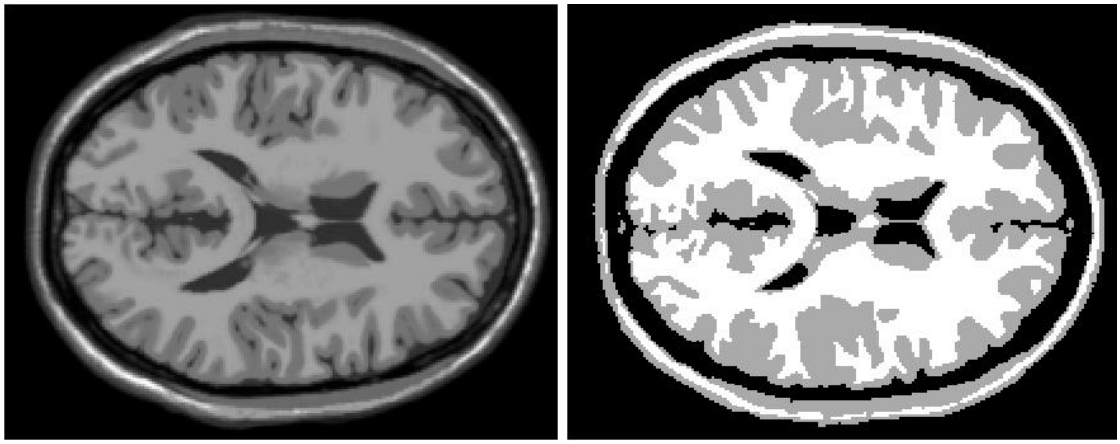
1.1 Overview

Recently, the focus of a vast amount of research in the field of medical image analysis has aimed to develop improved methods of image segmentation, image registration, and the quantification of anatomical and physiological parameters of images. These techniques provide the key components involved in systems such as those designed for image-guided surgery, the construction of atlas-based descriptions of anatomical regions, and the visualization of anatomy and their underlying physiological processes [24]. Recent improvements in image acquisition techniques and the processing power of modern computers have allowed for significant advancements in these technologies. The field of image analysis has proven itself a powerful tool in the field of medicine, stimulating an incredible demand for faster and more accurate techniques and algorithms that require less human interaction. As a result, the amount of research invested in these topics is increasing dramatically. This section provides a brief overview of image segmentation, magnetic resonance imaging (MRI), and image registration. Finally, the Insight Registration and Segmentation Toolkit (ITK) is introduced.

Image Segmentation

Image segmentation refers to the process of partitioning an image into regions which are homogeneous according to some property. In medical image analysis, segmentation is a critical step used in the identification of region boundaries and the separation of tissue classes [52]. Figure 1.1 illustrates an example segmentation of a slice of MRI brain data

into the different tissue classes, which include the white matter, gray matter, and ventricles.



(a) **(b)**

Figure 1.1. Example of the segmentation of MRI brain data into different tissue classes in which (a) the original brain image is segmented into the regions of: (b) white matter (white), gray matter (gray), and the ventricles and background (black).

An accurate segmentation result allows for improved accuracy in quantification [34] and morphological analysis [71], and leads to enhancements in the visualization [6] of data. Thus, accuracy during the segmentation process is important to provide a more useful visual rendering to aid radiologists and clinicians in diagnostics and surgical planning [3, 26]. Manual segmentations may be performed by an expert through the careful labeling of tissue classes at each pixel in the image data. While this may yield desirable results, it is often time consuming, expensive, and the subjective nature of the process does not allow for reproducibility. As a result, a number of computational methods for image segmentation have been investigated.

The image segmentation problem can be classified into two primary approaches. The first approach involves clustering pixels into different regions and subsequently providing a label for each region [11, 51]. A second approach seeks to extract the boundaries of the image which separate these regions [10, 32, 42]. In either approach, many variations exist. First, these algorithms may be formulated to solve the segmentation problem in the case of 2 or 3 dimensional (2D or 3D) images. Also, the application may either require user initialization or may be fully automatic. Similarly, the number of region labels to be

applied may be user-specified or may be left to be determined by the algorithm. Finally, the nature of the tissue classification at each pixel may be strictly limited to a single label, or may contain a fractional probability associated with each label.

Many of the existing segmentation techniques are based on image intensity [51], intensity gradient [10, 32, 42], and region statistics [11]. However, complications occur such as the object of interest may be occluded, the image may be noisy, or there may be partial volume effects. In the latter case, multiple tissue classes contribute to the intensity value of a single pixel or voxel of the image, an effect that may be attributed to choice of slice thickness or pixel size. In low-contrast regions, the boundaries between contiguous regions may become blurred, causing the two regions to appear to be connected when an actual boundary exists. Similarly, different objects in the image may take on similar intensity characteristics, making it more difficult to segment these tissues. It then becomes necessary to include a combination of information into the segmentation algorithm in order to more accurately extract the target object. Methods have been proposed which incorporate prior information in terms of shape [12, 13, 20, 21, 25, 35, 36], topology [43], atlas information [2, 14, 19, 22, 33, 59, 64, 70, 73] (see [52] for details), and statistical information [44]. The exact technique and information to be incorporated should be chosen to best suit the specific application. A more complete overview of medical image segmentation techniques can be found in [52]. Chapter 2 provides a more detailed description of active contour segmentation techniques. In particular, level set methods, including geodesic active contours [10], the Chan-Vese image segmentation model [11], and a fast algorithm for level set optimization [61] are discussed.

Magnetic Resonance Imaging (MRI)

Medical image segmentation, as applied to magnetic resonance (MR) or computed tomography (CT) images, is used to map the anatomy of a subject [52]. An important use of these images includes the segmentation of the brain scans of normal subjects. The extraction of the brain tissue from the data may then lead to the classification of gray

matter, white matter, and cerebrospinal fluid (CSF), or may lead to the segmentation of specific structures in the brain.

Magnetic resonance imaging (MRI) is an imaging technique based upon the principles of nuclear magnetic resonance (NMR) [28]. NMR, which has origins in the fields of chemistry and physics, is a spectroscopic technique used to observe the molecular properties of materials. MRI is a Hydrogen-based imaging technique used primarily in the medical field to observe the physiology of living tissue inside the body. MRI has shown to be especially useful in brain imaging due to its ability to provide images which exhibit superior soft tissue contrast, high resolution, and high signal to noise ratio (SNR). Also, MRI has flexibility in acquiring, or re-slicing images in the orientation best suited for visualizing and quantifying a specific anatomical structure. It is therefore effective in demonstrating a broad range of pathologies making it an extremely valuable tool in medical diagnostics.

MRI has demonstrated superiority over computed tomography (CT) in several respects. First, CT uses an ionizing radiation in the form of x-rays to perform the image acquisition. MRI employs a non-ionizing radio frequency signal, leading to the non-invasive nature of this technology. Also, because MRI has the ability to provide greater soft tissue contrast, it allows for greater differentiation of white and gray matter of the brain. Another desirable property of MRI lies in its ability to identify morphological anomalies of the brain associated with disorders such as schizophrenia [37] and Alzheimer's disease [49]. These irregularities are often very subtle. In fact, diagnosing such disease can not be accomplished through an individual subject, but through differences which are apparent in comparing group data only. Accuracy in the imaging techniques used in the visualization and quantification of the brain is therefore of high importance.

To provide a summary of the basic principles of MRI, the following overview is given. Consider the human body, which is primarily fat and water. The hydrogen content within these components leads to the body being composed of roughly 63% hydrogen atoms

[28]. The nuclei of these hydrogen atoms consist of a single proton, which continuously spins around an axis. This property of rotating charge is therefore referred to as 'spin', and may be interpreted as a small magnetic field. As illustrated in Figure 1.2 (a), a group of protons in a natural state do not possess a net magnetic field due to the random orientation of these magnetic moments. The application of a fixed external magnetic field, B_0, causes the spin axes of protons to precess about B_0, leading to a net magnetization in the direction parallel or anti-parallel to that of B_0 [28]. Figure 1.2 (b) illustrates the effect of applying B_0 to a group of protons originally in a natural random state.

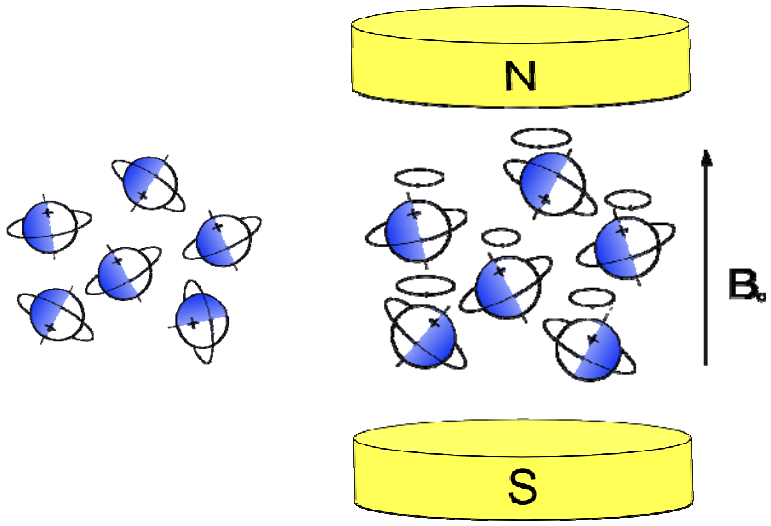


Figure 1.2. The application of an external magnetic field, B_0, to a group of protons which are: (a) initially in a natural random state. (b) The application of B_0 causes precessional motion about the direction of B_0.

The precessional motion occurs at a frequency which is proportional to B_0. This is referred to as the Larmor frequency given as

$$\omega_0 = \gamma \cdot B_0, \qquad \textbf{(1.1)}$$

where γ is the gyromagnetic constant. If the direction of B_0 lies in the z-plane, the net magnetic field will remain zero in the transverse plane, or x-y plane in this case.

The application of electromagnetic radiofrequency (RF) pulses of an appropriate frequency has the ability to force the spin axis of the protons from the z-axis into the x-y plane, as illustrated in Figure 1.3.

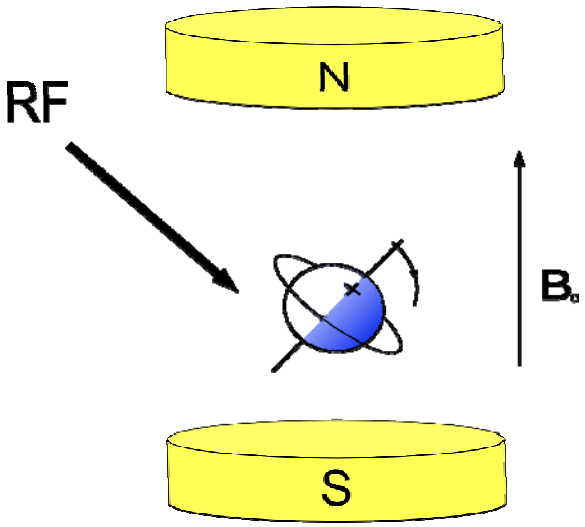


Figure 1.3. A RF pulse applied to a proton in the presence of an external magnetic field B_0 forces the spin axis of the proton from the direction of B_0 into the transverse plane.

This process, referred to as RF excitation, is accomplished by passing a current through an RF coil, which acts as a transmitter, to produce an electromagnetic field. This leads to a net magnetization in the transverse plane. Upon completion of this RF pulse, the spin axis is able to realign to the original direction given by the magnetic field, B_0, as shown in Figure 1.4.

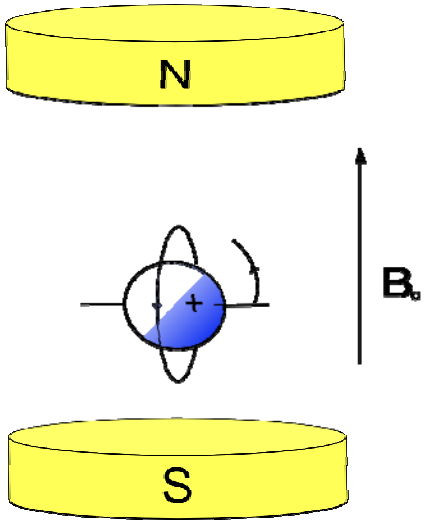


Figure 1.4. After the RF pulse ends, the spin of the proton realigns with the direction of B_0 as indicated.

As a result of the changes in the magnetic properties associated with the spins realigning following the RF pulse, the nuclei emit RF energy. This effect produces a current in the RF coil, which now acts as the receiver. This current is the signal that is measured during

the acquisition process. Figure 1.5 illustrates the MRI magnet and the direction of the magnetic field. In order to acquire image data, the subject is placed in the bore of the magnet.

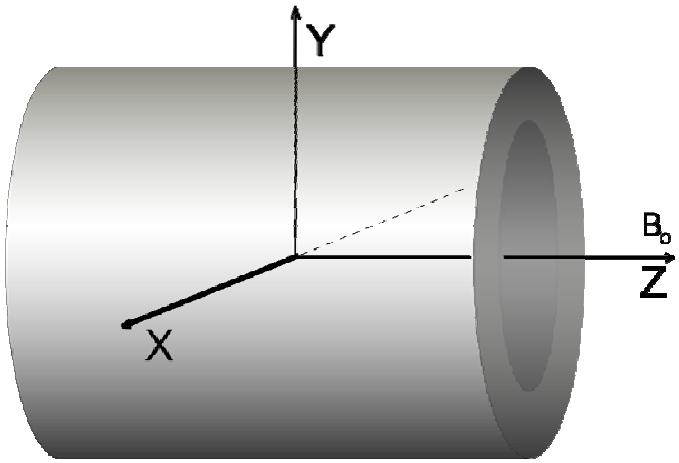


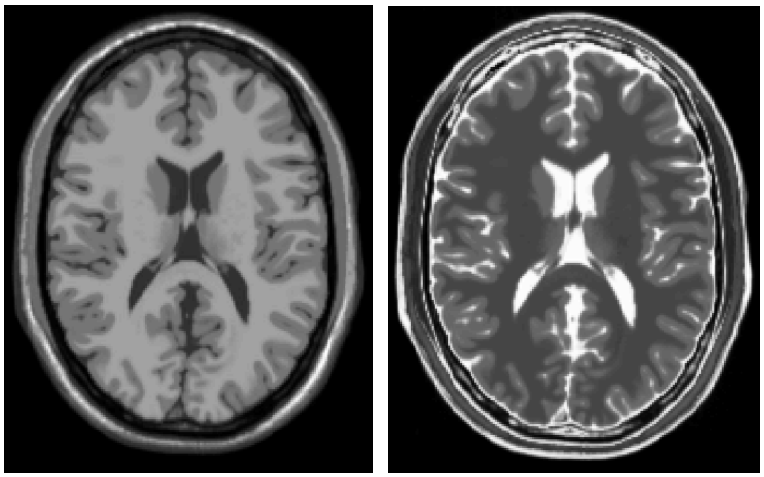
Figure 1.5. The MRI magnet with the axes directions indicated. The transverse plane lies in the x-y plane while the direction of B_0 is in the direction of the z-axis.

In order to obtain data for a single slice, a magnetic field gradient, referred to as the slice-select gradient, is applied to provide a magnetic field which is unique to each slice of the volume [28]. If the magnitude gradient field in the z-direction is applied, an RF pulse can then be emitted to bring about a net magnetization in the transverse plane of a desired slice. The removal of the z-component of the magnetization and then applying this gradient magnitude concept in the x and y directions of the magnetic field allows for the relative coordinate positions to be encoded by the frequency of the obtained signal. This process is known as frequency encoding. Similarly, in phase encoding gradient pulses are applied separately along the orthogonal direction to the frequency encoding to allow for the phase to be a function of the spatial position of the emitted signal. The specific order and timing associated with applying the gradient magnitudes in each direction is referred to as the pulse sequence. Reconstructing the raw data into useful images is then accomplished through an inverse 2D Fourier transform.

The intensity values of the acquired image are generally influenced by the underlying properties of the tissues from which the RF signal is generated. However, one of the primary strengths of MRI is that the contrast between different biological tissues may be tuned by varying the parameters involved in the image acquisition [37]. The spin lattice

relaxation time, T1, refers to the time constant related to the spin realigning with the magnetic field, B_0. Similarly, the spin-spin relaxation time, T2, refers to the decay of the net magnetization in the x-y direction. Proton density (PD) is a measure based upon the number of nuclei stimulated. The time between RF pulses (TR) and the time when the signal is acquired following this pulse (TE), determines the contribution from PD, T1 and T2 in the resulting image.

The ability to alter the mentioned parameters has given rise to a wide variety of methods used to acquire images of differing characteristics. For example, a 3D acquisition protocol commonly used includes spoiled gradient recall (SPGR) imaging [37]. SPGR is a T1-weighted imaging technique which therefore provides images with relatively good white and gray matter contrast. This modality can acquire data from an entire brain with 1.5 mm or thinner slices in 10 minutes or less. Also, Fast Spin Echo (FSE) imaging is a T2-weighted imaging sequence, which provides a better contrast between brain tissue and bone. Figure 1.6 illustrates examples of these MRI images. Table 1.1 contrasts the intensity value characteristics associated with white matter, gray matter, and CSF for T1- and T2-weighted images. A more thorough treatment of the principles of MRI physics is provided by [28, 63].



(a) **(b)**

Figure 1.6. Example of (a) T1- and (b) T2- weighted image simulation models generated from Brainweb: Simulated Brain Database [8, 16].

Table 1.1. The intensity characteristics of the white matter, gray matter, and CSF tissue classes associated with T1 and T2 weighted images.

	White Matter	Gray Matter	CSF
T1	Bright	Light	Dark
T2	Dark	Light	Bright

More recently, variations to the pulse sequence involved in MRI acquisition have been used to produce images containing information specific to oxygenation levels or water diffusion, as in the case of functional MRI (fMRI) and diffusion tensor MRI (DT-MRI), respectively [28]. fMRI is often used to determine which brain regions are being used in performing functions such as thought, speech, movement, and sensation. DT-MRI is used to generate detailed 3D maps of nerve pathways in the brain, heart muscle fibers, and other soft tissues.

Consider two categories of MRI images: (1) anatomical images, which depict information based mainly on morphology, and (2) functional images, which illustrate information based upon metabolism or blood flow. The process of image registration of functional medical data to the associated anatomical data can be used to align the two images, thus generating an image which displays this metabolic activity superimposed over the anatomical region responsible for this activity [41]. This may provide useful insight that may aid radiologists and surgeons in the diagnosis and treatment of a range of conditions.

Image Registration

Image registration denotes the process of determining the coordinate transformation that brings two images into spatial alignment. As in image segmentation, an expert can perform a manual registration by aligning images according to visual clues. However, the process is neither time nor cost efficient. Accuracy in the registration process is necessary in applications such as the construction of atlas-based descriptors of anatomical regions, comparison of pre- and post-operation images, and tumor growth tracking.

To summarize the process involved in performing a typical registration approach, consider two images which are to be registered. Assume one image is designated as the ‘fixed’ image and the second image is referred to as the ‘moving’ image. One convention

stipulates that the coordinate system of the fixed image is moved relative to the coordinate system of the moving image to find the best alignment between the two images [30]. This searching process characterizes the registration as an optimization problem. The primary components of the registration include: (1) an interpolator, which determines the voxel intensities at non-grid positions, (2) a similarity metric, which provides a measure of how well the images are matched, (3) an optimization scheme, which uses the metric as a cost function to guide the search for the optimal transformation parameters, and (4) the type of transform, which specifies the parameters to be used in the optimizer search space [30].

The variety of applications involving registration has contributed to the large range of techniques. Therefore, a number of criteria have been identified to be used in determining which techniques to incorporate given a specific task [41]. For example, the problem may involve images acquired from multiple subjects, or from an individual subject as in the case of longitudinal time studies. Another criterion to consider includes the types of imaging modalities involved. While unimodal registration techniques exist to register images of the same modality, multimodal registration techniques are also available to provide a correspondence between images of differing imaging modalities. In yet another example, the characteristics of the data for which the registration process is to be performed must be determined. This may include the use of known anatomical landmarks within the data, or the use of information provided by the image intensities of the full image [41]. The consideration of these criteria, among others described further in [41], is necessary in choosing the techniques and methods to be incorporated for the specific registration application. A more complete overview of medical image registration can be found in [41, 75].

Chapter 2 provides a more detailed description of the components involved in the registration process. Also, registration methods which incorporate mutual information as the metric component are discussed in more detail in Chapter 2 as well as in the methods of Chapter 3.

Insight Segmentation and Registration Toolkit (ITK)

The Insight Segmentation and Registration Toolkit (ITK) is an object-oriented, open source software system currently under active development [30]. The toolkit, provided in a template C++ style, is available in multiple platforms, including Windows, UNIX, and Linux. ITK was developed to support the Visible Human Project and to provide the medical imaging community with tools for segmentation and registration in data of two, three, or higher dimensions. Within the toolkit there exists a vast repository of algorithms well-known within the medical imaging community. ITK allows for applications used in research to be quickly implemented without the need for reproducing and debugging code fundamental to the field of medical image analysis.

The data within ITK are represented as objects, and process objects are referred to as filters. These data objects are connected to filters in pipelines in order to perform the desired process. The use of "smart pointers" within the implementation of ITK provides efficient memory management. ITK does not, however, provide graphical user interfaces (GUIs), nor does it provide tools for visualization of the data. However it is not difficult to incorporate other tools such as the Visualization Toolkit (VTK) and the Fast Light Toolkit (FLTK) in order to provide this functionality. ITK also provides wrappers for interfacing with interpreted languages such as Tcl or Java.

Assembling an application in ITK involves connecting a string of filters, each of which is used to perform a precise function [30]. Data, usually in the form of an image, are specified as the input to the first filter to be processed. Each subsequent filter in the pipeline is connected by specifying its input as the output of the previous filter. The parameters associated with each of the filters are also to be specified prior to processing the data. Upon executing the pipeline, this data sequentially undergo the process associated with each filter, generating the desired output at the final filter. The filters involved in an ITK implementation of an affine registration and geodesic active contours application are discussed in more detail in Chapter 3.

1.2 Motivation

Alzheimer's disease (AD) is a form of dementia usually occurring in the elderly population [1]. It has been estimated that roughly 4.5 million Americans may suffer from AD. Although the disease is not recognized as a normal part of the aging process, AD usually begins around the age of 60 and the risk of disease increases with age. The disorder first affects the regions of the brain responsible for controlling memory, thought, and speech. Initial symptoms slowly become apparent and may include difficulty in retrieving recent memories or the names of familiar people. As AD progresses, symptoms include the inability to recognize family members, difficulty speaking, forgetting how to perform common daily tasks, aggression, and extreme anxiety. Extreme cases require the individual to be given full-time care.

Although no cure exist, early and accurate diagnosis of AD is important to provide the best treatment for the symptoms [1]. Diagnosing AD is a difficult task and no single test exists which can confirm the disease with great certainty. Therefore accurate diagnosis relies on a combination of a number of tools, such as a review of family history, physical examinations, and brain scans. Morphological scans such as CT or MRI are frequently used to ensure that the symptoms mimicking those of AD are not being caused by a tumor, hemorrhage, blood clot, or hydrocephalus. These scans are also used to show loss of brain mass, particularly in the hippocampus, associated with AD. Researchers are working to develop techniques which provide an accurate measure for this brain atrophy from imaging data to aid physicians in diagnosing AD with greater success [1].

1.3 Problem Definition

The overall aim of this research is to investigate techniques for the segmentation of gray scale images containing edges with regions of low-contrast. More specifically, a goal is to provide a method adept in extracting the brain volume from a population of real MRI rabbit brain data. These data volumes were acquired for use in an experiment designed to analyze the effects of AD on the relative volumes of the tissue classes in the brain. The brain extraction step is necessary for a subsequent classification step which determines the white matter, gray matter, and CSF content within the brain volume. This

classification then allows for the quantification of the relative tissue volumes which are compared between a normal animal group and an animal model of AD.

An exploration into the literature of existing techniques in medical image analysis was completed in order to gain an understanding of the steps that have been taken in order to arrive at the desired result. Through this research, several approaches were investigated as a solution for the segmentation of images containing edges with low-contrast regions, in which the real MRI rabbit data is an example. These approaches incorporate concepts from image registration and segmentation in order to perform the task at hand.

In the initial stages of this research, the MRI rabbit head data sets were obtained. During the preliminary experiments aimed towards extracting the brain volume, a variety of segmentation approaches were visited. However, several characteristics of the data complicated the process. These include the irregular shape of the rabbit brain, the similarity of intensity values of the brain to tissues outside of the brain region, and also partial volume effects apparent in several slices of each data volume. The standard approaches found in the existing literature were unable to produce adequate segmentation results. These include intensity thresholding, k-means, snakes active contours, geodesic active contours, and the Chan-Vese segmentation model. Although active contour methods were among the most promising, these methods would require improvements in order to provide an adequate segmentation of the MRI rabbit brain data.

1.4 Contributions

Information provided by a shape prior can be incorporated into image segmentation techniques to make the process more accurate and robust. In this thesis, several methods are proposed which incorporate shape information into existing level set based techniques to provide an improved solution for the segmentation of low-contrast images.

First, a method is presented for using shape information within many existing curve-evolution based segmentation algorithms by encoding this information into the speed term. This makes it possible to perform segmentation based on a shape prior using

existing algorithms and code by simply substituting the speed image. An algorithm is presented which uses a pre-segmented atlas image. The new speed function slows curve evolution near edges in the image to be segmented as well as near edges in the registered atlas image. This technique is demonstrated using 2D synthetic images as well as real data.

Second, a method is proposed which uses moment invariants to provide a shape constraint within the segmentation technique proposed by Song and Chan in [61]. The algorithm presented incorporates an additional shape energy term into their fast level set framework. This shape term influences the evolving segmentation to maintain the shape prior. The correspondence between these two shapes is provided by a set of moment invariants computed from each. This technique is demonstrated using 2D synthetic and real image data.

1.5 Outline

The remainder of this thesis is outlined as follows. Chapter 2 presents a background of the literature which provides a foundation for the concepts and methodology described throughout the thesis. Also to be discussed in Chapter 2 is the ITK toolkit used in implementing and testing a number of these concepts. Chapter 3 presents the methods surrounding a new speed function used in a geodesic active contours segmentation approach. Chapter 4 presents a method which incorporates image moments to provide a shape prior based Chan-Vese segmentation model. Chapter 5 provides the details and results of these experiments using the methods presented in Chapters 3 and 4. Concluding remarks as well as a discussion of possible future work are included in Chapter 6.

Chapter 2: Background

The following chapter provides an overview of the literature that was examined, along with a background of the ITK software used in implementing several of the concepts employed through the course of this research. Methods of image segmentation, image registration, as well as the use of an atlas or shape prior to assist the segmentation process, were of primary focus. Also discussed are the concepts of image moments and the use of moment invariants in pattern recognition.

2.1 Medical Image Segmentation

A number of approaches used in the segmentation of images exist. These methods may be classified into a number of categories including thresholding [51], clustering [5, 17], region growing [27], active contours [10, 11, 32, 42, 60], graph cuts [7], Markov Random Field models [44], and atlas guided approaches [2, 14, 19, 22, 33, 59, 64, 70, 73]. A more complete summary of many of these approaches can be found in [52].

2.1.1 Active Contours

Of these techniques, active contour methods have been proven highly effective in medical image segmentation. These methods operate by first initiating a user-specified curve in an image. The curve then evolves according to the minimization of an energy function which drives the curve to the boundaries of a target object. Figure 2.1 demonstrates the use of active contour segmentation techniques in segmenting a target object from the background.

Kass, Witkin, and Terzopolous introduced the active contour approach with their classic snakes model [32]. To describe this model, consider the parametric representation of a curve as $c(s) = (x(s), y(s))$, in which x and y represent coordinates of the image grid at locations in which the curve lies. This parametric representation is illustrated in Figure 2.2.

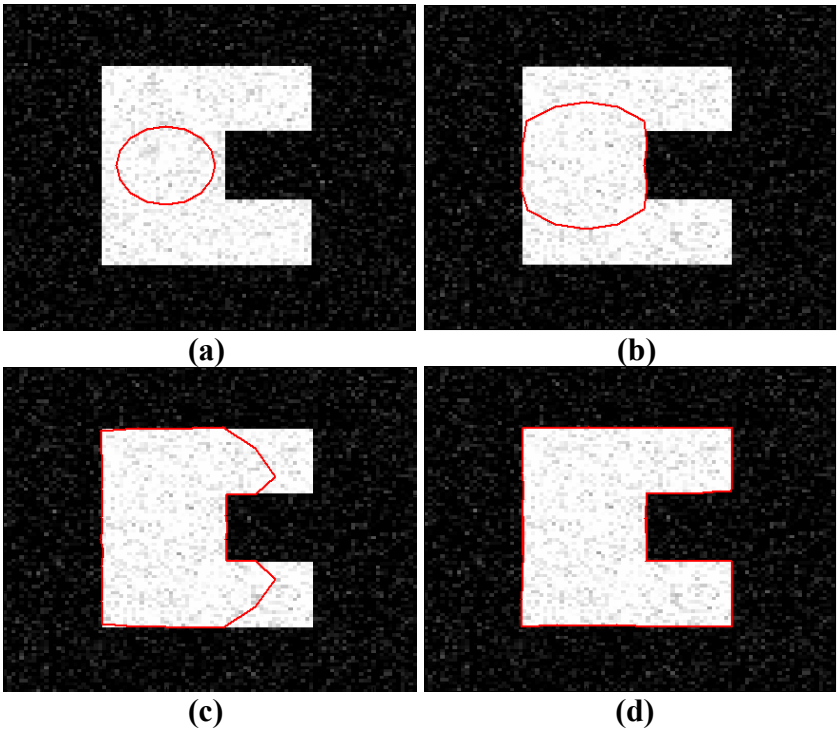


Figure 2.1. A 2D example of using active contours techniques to find the boundaries of a shape object in an image. (a). The initial contour (red) located inside the target object (light). (b) and (c) Intermediate steps of the evolving contour. (d) The final segmentation result.

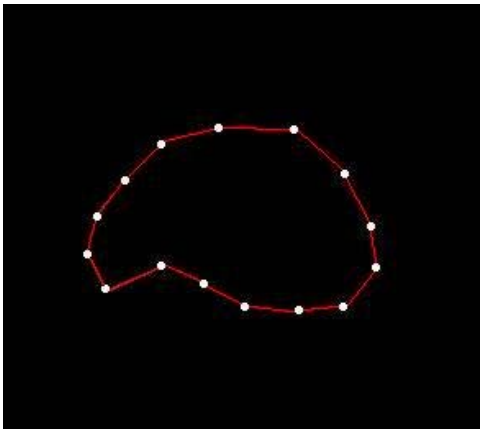


Figure 2.2. A parametric representation of a curve (red) represented as a vector of points (white).

In this snakes model, the energy function to be minimized is written as [32]

$$E^*_{snake} = \int_0^1 E_{\text{int}}(c(s)) + E_{image}(c(s)) + E_{con}(c(s))ds \,. \qquad \textbf{(2.1)}$$

This energy controls the curve evolution by incorporating (1) an internal energy term which controls the smoothness of the evolving curve, (2) an image-based energy term to

pull a curve to the desired image feature, and (3) a user provided constraint term which allows a user to influence the curve toward or away from certain regions in the image.

The internal energy term, E_{int}, is represented in [32] by a combination of a first order membrane spline energy term and a second order thin plate spline energy term written as

$$E_{\text{int}} = \left(\alpha \left\| c_s(s) \right\|^2 + \beta \left\| c_{ss}(s) \right\|^2 \right) / 2, \qquad \textbf{(2.2)}$$

where α and β are weights controlling the influence of each of these energy components. Minimizing the membrane spline energy of a curve minimizes the curve's length, while minimizing the thin plate spline energy minimizes the curve's squared curvature.

The image energy, E_{image}, presented in [32] is used to attract the curve to salient features in the image such as lines and edges. Here, lines refer to dark or light contours in an image, while edges refer to contours with large image gradients. This energy term can be written as a combination of the two features

$$E_{image} = w_{line} E_{line} + w_{edge} E_{edge}, \qquad \textbf{(2.3)}$$

where w_{line} and w_{edge} represent the constants weighting these terms. If the first term is set as $E_{line} = I(x,y)$, the image intensity at the pixel (x,y), then E_{line} attracts the curve to dark or light intensity lines, depending upon the sign of w_{line}. If the second term is set as $E_{edge} = -|\nabla I(x,y)|^2$, the curve is attracted to contours with large image gradients.

The user constraint term of the energy, E_{con}, consists of simple repulsion or attracting forces, described further in [32], which push or pull the curve towards user specified locations in the image. If the last two energy terms of (2.1) are combined to give $E_{ext} = E_{image} + E_{con}$, the evolution equation derived in [32] is written as

$$\frac{\partial c}{\partial t} = \alpha c_{ss} - \beta c_{ssss} - \nabla E_{ext}. \qquad \textbf{(2.4)}$$

An example of the segmentation results obtained using this model is shown above in Figure 2.1. Despite the potential of this model to obtain desirable segmentation results, it

fails to accommodate situations in which the evolving interface develops sharp corners or cusps, or experiences topological changes. Therefore, the result is extremely dependent upon the initialization of the active contour.

Level set techniques overcome the problems associated with discontinuities or topological changes within the parametric representation. During level set curve evolution, the interface is able to split and merge, thus accommodating for complex topologies. Another advantage of this property over previous active contour methods is that the topology is not required to be specified in advance and there is no additional complexity in handling the occurrence of such situations.

2.1.2 Level Set Segmentation Methods

Level set techniques, originating in the fields of fluid mechanics and material science to track dynamic boundaries, were introduced by Osher and Sethian [48]. In this model, a curve $c(t)$, where t is a spatial parameter, is represented as the zero level set of a higher dimensional embedding function, ϕ. Figure 2.3 illustrates such an embedding function which follows the conventions described below.

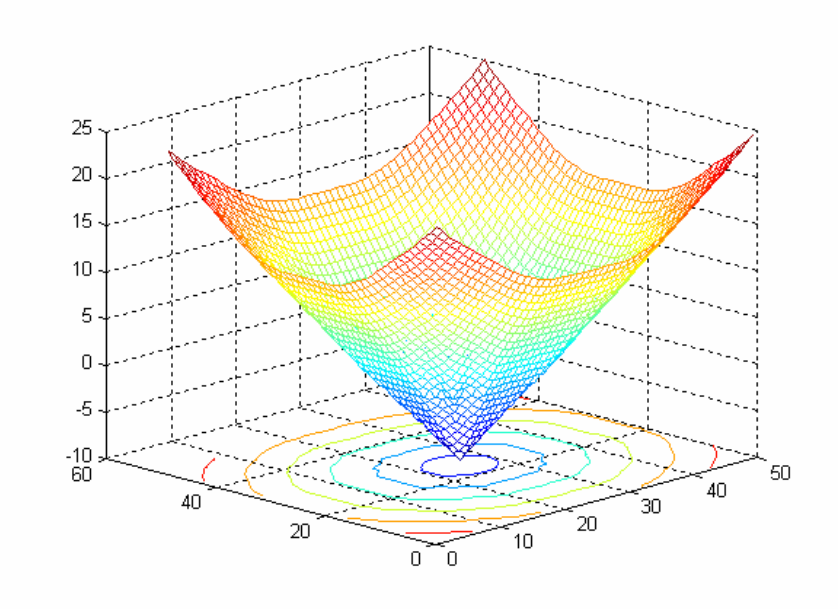


Figure 2.3. A 3D embedding function, ϕ, the level sets of which are indicated by the 2D contour plot found below the mesh plot. Values inside the zero level set are negative, while values outside are positive.

One possible technique to initialize ϕ is by applying the signed distance function to an initial contour placed in the image as specified by a user. Here, ϕ(x,y) is determined by the distance from the *(x,y)* location in the image grid to the curve. One convention stipulates that the region inside the curve is represented by values of ϕ less than zero, and the region outside of the curve is denoted by values of ϕ greater than zero, which is summarized in the following:

$$\begin{aligned} \phi(x,y) &= 0 \text{ , if } (x,y) \text{ is on the curve,} \\ \phi(x,y) &< 0 \text{ , if } (x,y) \text{ is inside the curve,} \\ \phi(x,y) &> 0 \text{ , if } (x,y) \text{ is outide the curve.} \end{aligned} \qquad \textbf{(2.5)}$$

The use of this implicit model allows for geometric properties of the curve to be easily determined. For instance, the gradient of ϕ is perpendicular to the level curve, and therefore the normal of the embedded curve can be computed as

$$N(x,y) = \frac{\nabla \phi(x,y)}{\|\nabla \phi(x,y)\|}. \qquad \textbf{(2.6)}$$

Also, the curvature of the level curve is determined as the rate of change of this normal vector as

$$\kappa = div\left(\frac{\nabla \phi(x,y)}{\|\nabla \phi(x,y)\|}\right). \qquad \textbf{(2.7)}$$

In order to derive an evolution equation for ϕ, consider *c(t)* the zero level set curve, where ϕ *(x(t), y(t), t)* = 0, for all *t*. This implies that

$$\frac{d\phi}{dt}(x(t), y(t), t) = 0, \qquad \textbf{(2.8)}$$

which, by the chain rule, can be rewritten as

$$\frac{d\phi}{dt} = \frac{\partial \phi}{\partial x}\frac{dx}{dt} + \frac{\partial \phi}{\partial y}\frac{dy}{dt} + \frac{\partial \phi}{\partial t} = (\nabla \phi \cdot c'(t)) + \frac{\partial \phi}{\partial t}. \qquad \textbf{(2.9)}$$

The *c'(t)* term in (2.9) can be decomposed into its tangent and normal components to give

$$0 = (\nabla\phi \cdot (v_N N(t) + v_T T(t)) + \frac{\partial\phi}{\partial t}. \quad \textbf{(2.10)}$$

Because the gradient of the embedding function is perpendicular to the tangent of *c(t)*, the tangential component can be removed and (2.10) is rewritten as

$$0 = (\nabla\phi \cdot (v_N N(t)) + \frac{\partial\phi}{\partial t}. \quad \textbf{(2.11)}$$

Substituting Equation (2.6) for the normal component *N(t)* gives

$$0 = \left(\nabla\phi \cdot \left(v_N \frac{\nabla\phi}{\|\nabla\phi\|}\right)\right) + \frac{\partial\phi}{\partial t}, \quad \textbf{(2.12)}$$

which may be rearranged as

$$0 = v_N \|\nabla\phi\| + \frac{\partial\phi}{\partial t}. \quad \textbf{(2.13)}$$

In curvature based evolution schemes, v_N is a function of the curvature κ as written in (2.7). Equation (2.13) can therefore be written as

$$0 = -F(\kappa) \cdot \|\nabla\phi\| + \frac{\partial\phi}{\partial t}. \quad \textbf{(2.14)}$$

The details of an example of the function *F(κ)* used in the geodesic active contour level set approach are provided in the following section.

Geodesic Active Contours

Geodesic active contours is a level set segmentation approach in which an embedding function ϕ, which is initialized by the user, evolves according to an energy function consisting of three terms: (1) an internal smoothness constraint term which minimizes curvature, (2) an image dependent speed term which slows the curve near image region boundaries, and (3) an inflation force used to expand or contract the curve. Similar to the derivation shown in (2.8) through (2.14), the evolution equation for ϕ was derived by Sapiro et al [10] to give

$$\frac{\partial \phi}{\partial t} = -g(I)\kappa\|\nabla\phi\| + \nabla g \cdot \nabla \phi . \qquad \textbf{(2.15)}$$

Here, ϕ represents the embedding function; g represents the image dependent speed term; and κ represents the curvature.

In determining the speed term g used in the level set formulation, a typical approach begins with the computation of the image intensity gradient. The negative potential of the intensity gradient is then computed using a speed function, resulting in an edge potential image. This function can take rational and exponential forms, examples of which are shown in Equations (2.16) and (2.17), respectively as

$$g(s) = \frac{1}{\left(1 + \left(\frac{s}{k}\right)^{1+\alpha}\right)}, \qquad \textbf{(2.16)}$$

$$g(s) = \frac{1}{\left(1 + \exp\left[-\left(\frac{s-\beta}{\alpha}\right)\right]\right)}. \qquad \textbf{(2.17)}$$

These speed functions provide a mapping which assigns strictly positive values close to 1 in homogeneous regions and near 0 where edges are located. Therefore, the evolving curve is slowed to a near stop close to the boundaries of homogeneous regions. Figure 2.4 illustrates the intermediate images, including the speed image produced using (2.17), involved in a segmentation application using the geodesic active contour model. Figure 2.4 (d) shows that the pixels located near the edge of Figure 2.4 (a) are mapped to low values while pixels in the homogeneous region are mapped to higher values between 0 and 1.

In (2.17), commonly referred to as the sigmoid function, the constant α defines the center of the gradient intensity range and β defines the width of this range. Properties of this function are illustrated in Figure 2.5. These constants are chosen by the user to indicate the range of gradient values which are to be considered image edges and the gradient values which are to correspond to homogeneous regions.

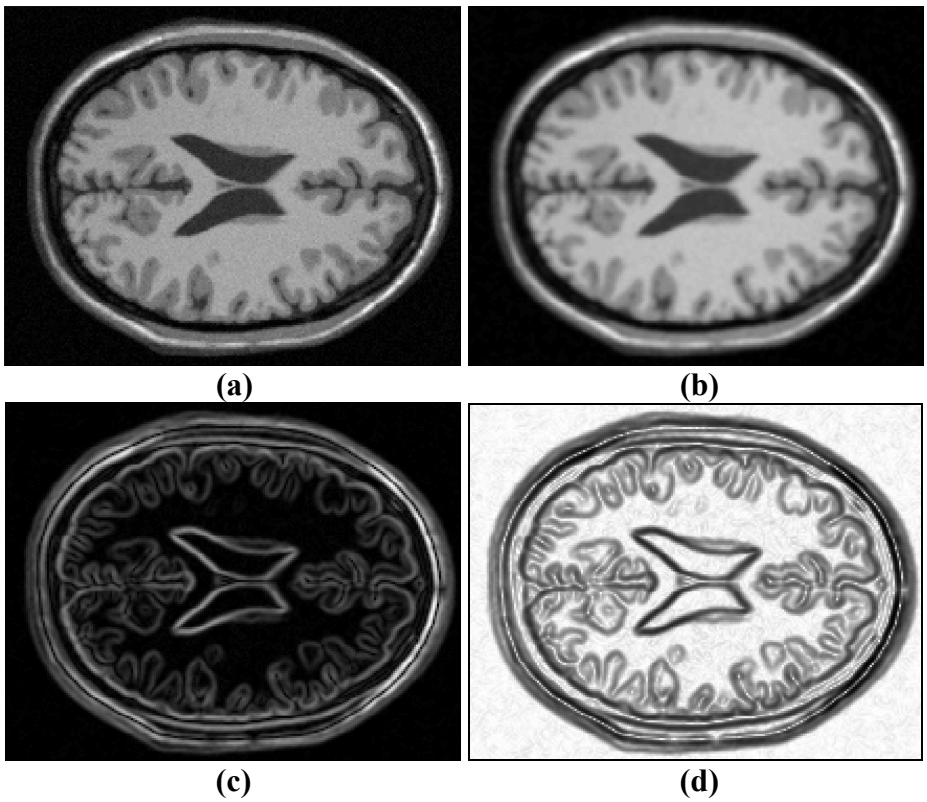


Figure 2.4. The intermediate steps of geodesic active contours segmentation. (a) The image *I*. (b) Smoothed image. (c) The gradient magnitude image. (d) The speed image *g* computed using (2.17).

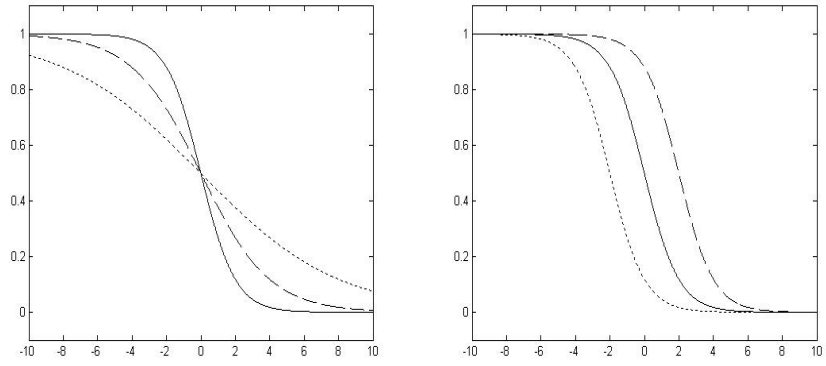
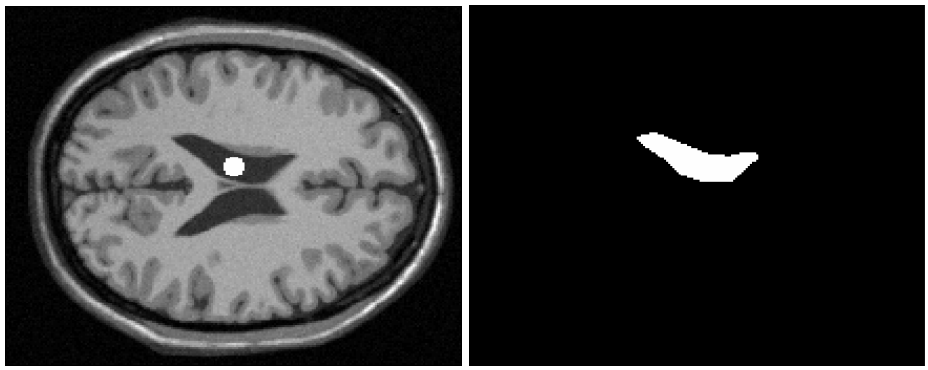


Figure 2.5. The effects of varying the parameters of (2.17), the sigmoid function. (a) Varying α. Here, α = -4 (*dotted line*), α = -2 (*solid line*), and α = -1 (*dashed line*). (b) Varying β. Here, β = -2 (*dotted line*), β = 0 (*solid line*), and β = 2 (*dashed line*).

Other methods to generate the speed term have also been proposed, which use local homogeneity measurements, as described in [31] e.g., to provide a measure of edge strength and homogeneity at each location within the image.

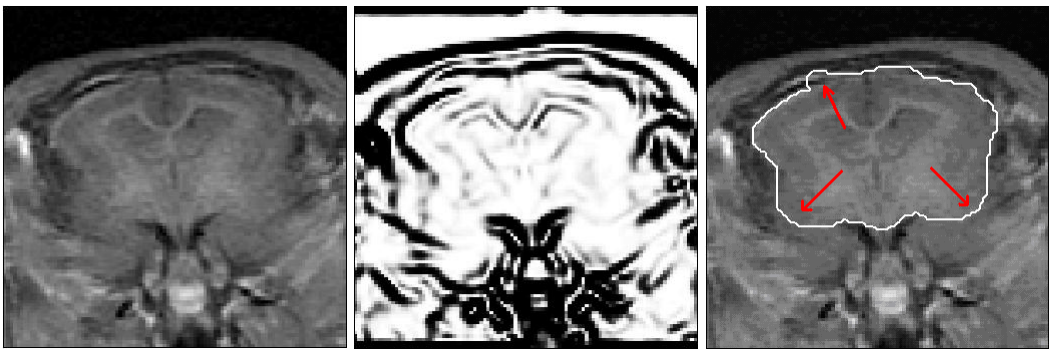
Figure 2.6 illustrates the segmentation of the right ventricle of the brain image of Figure 2.4. Figure 2.6 (a) shows the initialization of ϕ, specified by a user as the white circle. Applying the signed distance function to this circle provides the initial embedding function. Figure 2.6 shows the final segmentation result of the right ventricle.



(a) **(b)**

Figure 2.6. An example of the results of the geodesic active contours for the segmentation of the right ventricle. (a) The original image showing the initial embedding function ϕ_0 where $\phi_0 < 0$. (b) The final segmentation result.

This technique is able to produce adequate segmentation results in the limited case that a quality initialization of ϕ is provided and the image contains detectable edges between regions. In regards to the latter case, areas of low-contrast may prevent the detection of discernable edges in a number of regions. Therefore, the speed image did not have the effect of slowing the evolving contour at the location of the true boundary during the segmentation process. This contour is then able to pass through into the non-brain regions of the head. Figure 2.7 illustrates this 'leaking' effect in a single slice of data resulting from an attempt to extract the rabbit brain tissue from the surrounding tissue in an MRI image using an implementation of geodesic active contours. Compare the speed image and segmentation result provided in Figure 2.7 with the speed image of Figure 2.4 (d) and segmentation result provided in 2.6 (b). The speed image pictured in Figure 2.4 (d) shows distinct edges at the boundary of the ventricles. The corresponding segmentation result shown in Figure 2.6 (b) illustrates the ability of the speed image to slow the evolving curve to a stop at this boundary, producing a desirable result. This is in contrast to the speed image pictured in Figure 2.7 (b), which shows no apparent edge in several locations along the boundary of the brain. The evolving curve is therefore able to 'leak' beyond the boundary as indicated by the arrows in Figure 2.7 (c).



(a) **(b)** **(c)**

Figure 2.7. The segmentation of the brain region of a slice of MRI rabbit brain data with weak edge characteristics. (a) A slice of the rabbit brain MRI data. (b) The speed image. (c) A poor segmentation result in which the evolving curve (white) has passed beyond the boundary of the brain as indicated by the arrows (red).

Active Contours without Edges

Chan and Vese [11] proposed another level set based segmentation model which is governed by the properties of the region enclosed by the evolving curve. Therefore, the segmentation allows for the detection of objects containing no discernable edges as provided by the intensity gradient. Another benefit of using this model is that the result is less dependent upon the initialization. This implementation involves the minimization of a Mumford-Shah function [47] which defines the energy

$$E(u,c) = \mu \int_0^1 \|c'(s)\| ds + \lambda \int_{\Omega} |u_0(x,y) - u(x,y)|^2 dxdy + \int_{\Omega/c} |\nabla u(x,y)|^2 dxdy, \qquad \textbf{(2.18)}$$

where $u_0(x,y)$ represents the image, $u(x,y)$ represents an image model, Ω is the image domain, and the curve is denoted as c. The terms of (2.18) include (1) a boundary smoothness term, (2) a term describing the model fit error, and (3) a term constraining the smoothness of the model $u(x,y)$. The implementation proposed by Chan and Vese uses a piecewise constant image model, resulting in the energy

$$E(c_1,c_2,c) = \mu \int_0^1 \|c'(s)\| ds + \lambda_1 \int_{inside(c)} |u_0(x,y) - c_1|^2 dxdy + \lambda_2 \int_{outside(c)} |u_0(x,y) - c_2|^2 dxdy, \qquad \textbf{(2.19)}$$

where c_1 is the average of $u_0(x,y)$ inside c, and c_2 denotes the average of $u_0(x,y)$ outside c computed as

$$c_1 = \frac{\int_\Omega uH(\phi)dxdy}{\int_\Omega H(\phi)dxdy}, \qquad c_2 = \frac{\int_\Omega u(1-H(\phi))dxdy}{\int_\Omega (1-H(\phi))dxdy}. \tag{2.20}$$

Through the use of the regularized heaviside *H(z)* and Dirac measure *δ(z)* functions described by Chan and Vese in [11], the energy function in Equation (2.19) is cast into the level set framework. The minimization of this function using the Euler Lagrange framework leads to the evolution equation written as

$$\frac{\partial \phi}{\partial t} = \delta(\phi)\left[\mu\left(div\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right)\right) - \lambda_1(u-c_1)^2 - \lambda_2(u-c_2)^2\right]. \tag{2.21}$$

Fast Update for Minimization of the Chan-Vese Model

Chan and Song [61] proposed a fast algorithm for solving optimization problems formulated by level sets. In the Chan-Vese segmentation model, ϕ is updated by solving the Euler-Lagrange equation shown in (2.21). However, the methods presented in [61] dramatically reduce the computational speed by computing the energy directly and tracking only the sign of ϕ.

The algorithm of [61] begins by initializing ϕ as $\phi = 1$ inside the level set and $\phi = -1$ outside, which may be randomly assigned at each pixel. Next, the segmentation process begins by sweeping over the entire image. In doing so, each pixel of the image is visited and the energy change associated with switching each pixel from inside the curve to outside, or vice versa, is computed. According to the energy function in Equation (2.19), the direct solution is presented in [61] as follows. Assume at the current location *(x, y)*, $\phi = 1$. The change in energy associated with changing $\phi = 1$ to $\phi = -1$ (neglecting the length term) is computed as

$$\Delta F_{12} = (I(x,y) - c_2)^2 \frac{n}{n+1} - (I(x,y) - c_1)^2 \frac{m}{m-1}, \qquad \textbf{(2.22)}$$

where *I(x,y)* is the pixel intensity value at location *(x,y)*, and *m* and *n* are the number of pixels for which $\phi = 1$ and $\phi = -1$, respectively. Similarly, if at the current pixel location, $\phi = -1$, the change in energy is given as

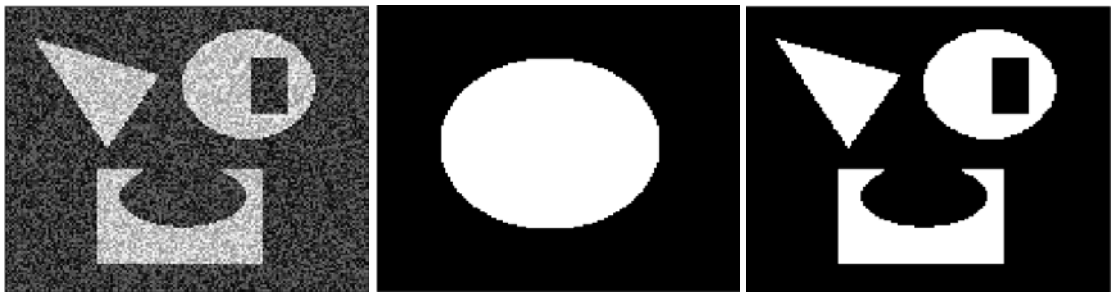
$$\Delta F_{21} = (I(x,y) - c_1)^2 \frac{m}{m+1} - (I(x,y) - c_2)^2 \frac{n}{n-1}. \qquad \textbf{(2.23)}$$

In [61], the length term of the energy (2.19) in the Chan-Vese model is approximated by

$$\int |\nabla H(\phi)| \partial x = \sum_{i,j} \sqrt{\left(H(\phi_{i+1,j)}) - H(\phi_{i,j})\right)^2 + \left(H(\phi_{i,j+1}) - H(\phi_{i,j})\right)^2}, \qquad \textbf{(2.24)}$$

where $\phi_{i,j}$ is the value of ϕ at the i,jth pixel. The result of (2.24) can only take values of 0, 1, or √2, depending on whether the set $\{\phi_{i,j}, \phi_{i+1,j}, \phi_{i,j+1}\}$ belong to the same regions. If this length term is to be considered, the change of length associated with switching ϕ is easily computed since only four neighbor points are altered in changing the value of ϕ.

In the occurrence of a negative change in energy, the sign of ϕ is changed. Thus, the region label of the pixel is switched to minimize the overall energy. When the ϕ value of a pixel is switched, the values of c_1 and c_2 are recomputed. This process is repeated until the total energy remains unchanged. This algorithm has demonstrated the ability to converge in only a few sweeps for two phase images, which refers to images containing two regions. An in depth analysis of this technique is presented in [61]. An example of the results obtained using the segmentation algorithm is illustrated in Figure 2.8. Figure 2.9 illustrates the effect of using this smoothness constraint in an image with a high level of noise. In the case that the smoothness constraint is not enforced, the result of the segmentation includes spurious pixels throughout the image as shown in Figure 2.9 (b). However, the application of the smoothness constraint is able to more accurately segment the objects from the background as shown in Figure 2.9 (c).



(a) **(b)** **(c)**

Figure 2.8. An example of image segmentation using an implementation of the fast update algorithm found in [61]. (a) The image I. (b) The initial embedding function ϕ_0 where $\phi_0 > 0$. (c) The final segmentation result ϕ where $\phi > 0$ after 2 sweeps of the algorithm.

(a) **(b)** **(c)**

Figure 2.9. The effect of the smoothness constraint in the segmentation of an image with a high level of noise. (a) The image I. (b) The final segmentation result, $\phi > 0$, when the smoothness constraint is not enforced. (c) The final segmentation result, $\phi > 0$, when applying the smoothness constraint.

If the smoothness constraint of this fast Chan-Vese algorithm is omitted, it can be compared to the k-means segmentation algorithm presented in [39]. The k-means procedure is initialized by randomly placing two centers of mass, m_1 and m_2. This is analogous to the initialization of the fast update algorithm in which ϕ is initialized and the values of c_1 and c_2 are computed. In separating the clusters in the k-means procedure, a minimum distance classifier is used to update the segmentation result. This step corresponds to the comparison of each pixel value to the mean value of each region using (2.22) and (2.23). Following this classification step in the k-means algorithm, m_1 and m_2 are recomputed for each cluster. Similarly, the values of c_1 and c_2 are updated according to the new segmentation result given by ϕ.

Shape Prior Segmentation Techniques

While variations of level set techniques have been proposed to improve computation time [61], others have presented methods for improving upon the segmentation accuracy. In

doing so, many techniques have been proposed which include a shape prior into the active contour segmentation framework [12, 13, 20, 21, 25, 35, 36, 58]. In [35, 36], Leventon et al. presented a method which includes statistical shape information into the geometric active contours energy function. Another approach presented by Chen et al. [13] incorporated a different shape term into this model. Also, several approaches have been introduced which incorporate shape priors into the Chan-Vese model [12, 21, 25]. Cremers et al. [21] proposed a variational approach that incorporates a shape term into the Chan-Vese model along with a labeling level set function L to indicate the region in which the shape prior is to be enforced. However, the reference shape must be aligned with the target object. Chan and Zhu [12] improved upon this work by allowing translation, scaling, and rotational differences of the prior shapes in accordance with the image object. Chan and Zhu also applied this concept to the fast update technique, excluding the length term, proposed in [61]. Foulonneau et al. [25] incorporated shape descriptors based on Legendre moments into the energy function, thus minimizing a function of the distance between the evolving contour and a target shape. The use of such moments allows for misalignment between the target object and shape prior and also allows for geometric variability between the object and reference shape.

Shape Descriptors based on Image Moments

In 2D pattern recognition, the forms of distortion often encountered include translation, rotation, scaling, and skew [38]. A number of methods have been developed, which are geared towards transforming a pattern into a form which is invariant under such distortions. In 1962, Hu first proposed a method which uses moments for 2D pattern recognition applications in [29], which allows for object recognition in 2D image patterns regardless of distortions caused by translation, rotation, and scaling.

In [29], Hu derived the set of 2D moment invariants based on the theory of algebraic invariants. The set of second- and lower-order invariants are summarized as follows. In [29], a 2D $(p + q)$th order moment m_{pq} of a density function $\rho(x, y)$ is defined in terms of Riemann integrals as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q \rho(x,y)dxdy\,, \tag{2.25}$$

where p, q = 0, 1, 2, … The first order moments are used to compute the image centroids, or the centers of mass, using

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}\,. \tag{2.26}$$

Using these centroids, the central moments μ_{pq} are defined in [29] as

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q \rho(x,y)dxdy\,. \tag{2.27}$$

The central moments do not change with a change in coordinates, and therefore, these moments are invariant to translation. These central moments may also be expressed in terms of the ordinary moments, the equations of which, corresponding to the first four orders, are provided in [29]. The first few orthogonal invariants which use the second order centralized moments are given as

$$\mu_{20} + \mu_{02}\,,$$
$$\left(\mu_{20} - \mu_{02}\right)^2 + 4\mu_{11}{}^2\,. \tag{2.28}$$

An extended list of these invariants is provided in [29].

In the specific example of 2D image patterns, methods which use moment invariants [29], Fourier descriptors [50, 74], Hough transforms [4], shape matrices [65], and principle axis methods [56] have been investigated as solutions to providing a normalized image which is invariant under the mentioned distortions. However, difficulties in accommodating for skew remain in these methods. Wang and Zhao [38] proposed a method which makes use of moment invariants to produce a compact image representation which accommodates for this type of distortion. Such a representation may be used to provide a measure of similarity between two images despite differences in translation, rotation, scaling, and skew.

2.1.3 Atlas-guided Segmentation

Information such as shape may also be provided by an atlas [2, 14, 19, 22, 33, 59, 64, 70, 73]. The atlas refers to a reference image, which provides a model of the anatomy or general morphology expected to be contained within the population of image data. Construction of an atlas can be accomplished by sampling from a large data bank, registering these samples to a defined coordinate system, and then averaging these samples to form a single data volume. In one possible representation of the anatomical structure of interest, careful manual or semi-automatic segmentation of the atlas may be conducted in order to arrive at a binary mask. By incorporating the atlas binary mask into a segmentation procedure, one introduces prior knowledge, such as the shape and geometry of the anatomical structure of interest.

Atlas-guided segmentation approaches are generally treated as an image registration problem, a process which is described in the following section, and are commonly referred to as atlas-warping. In this approach, one finds the transformation which maps the target image space to the atlas space to compare to one another on a level playing field. As a result, any labeling which identifies anatomical structures within the atlas is transferred to the novel image. While approaches have been proposed which use linear transformations [2, 33, 64], mappings formed by a sequence of linear and non-linear transformations [14, 19, 22, 59] allows for variability in the anatomy between subjects. Techniques have also been proposed which estimate a non-rigid deformation field between an atlas and the target image [73]. The deformation is then applied to the atlas to achieve the segmentation of the anatomical structure of interest. Also, methods have been proposed which combine non-rigid registration and segmentation into a unified framework, thus achieving the atlas-assisted segmentation result in a single step [70].

2.2 Medical Image Registration

As introduced in Chapter 1, image registration is the process used to determine

$$\min_T dist\left(I_1(x), I_2(x)\right), \tag{2.29}$$

the spatial transformation T which provides the minimum distance between homologous points of two images, I_1 and I_2. This distance refers to a measure of dissimilarity computed as a function of the image intensities of the images. A minimum of this distance is found using an optimization process. The optimization is accomplished by traversing through a given transformation parameter space with the moving image and computing the distance between the transformed moving image and the fixed image. As previously mentioned, the main components required within this registration framework include an interpolator, the type of transform, a similarity metric, and an optimization scheme. Existing techniques used in implementing each component are discussed in the following sections.

2.2.1 Interpolator

While performing the search for the optimal transformation parameters, an image is often mapped onto non-grid positions, and therefore an estimation of the image intensities at the sub-pixel level is required. This estimation can be accomplished through a number of interpolation techniques. The choice of interpolator to be used through the course of the registration process necessitates a tradeoff between computational speed and accuracy. Due to the numerous iterations in which the interpolation is performed during a typical registration, the time required in the registering two images may dramatically increase [30].

The computational efficiency of techniques such as linear interpolation or nearest-neighbor techniques has greatly popularized their use in image registration [41]. Linear interpolation is based upon the assumption that the image intensities vary linearly between integer pixel positions. The intensities at the non-grid positions are continuous, and the gradient of the intensity values is discontinuous at grid positions. Other methods, such as B-splines or Windowed Sinc interpolation, provide for greater sub-voxel accuracy, but at a computational cost [30]. B-spline interpolation methods represent image intensities using B-spline basis functions [67, 68, 69]. Upon inputting an image, the coefficients of the basis functions are computed using recursive filtering. The

intensities at non-grid positions are then determined by multiplying these coefficients with shifted B-spline kernels in a small support region.

2.2.2 Transformation

In choosing the type of transformation to be used in the registration framework, the nature of the spatial transformation required to align the images of interest must be considered. Typical transformations used in image registration include rigid, affine, projective, and curved transformations [41]. Rigid transformations allow for translations, rotations, and reflections, and are best suited for applications in which the object of interest is 'rigid' and exhibits only small difference in shape between the images. Affine transformations extend rigid transformations to include scaling and shearing, in which straight lines and parallelism between lines is preserved, but not the lengths and angles between these lines. Affine transformations are therefore most appropriate in the case that the scaling factors between images are not precisely known. Projection transformations do not preserve parallelism, and are most common in registering 2D images to 3D image volumes [41, 53]. While the above transformations can be represented in the form of a matrix, curved transformations require a displacement field representation [53]. Curved transformations are performed on a local basis, and are useful when the required mapping must have the ability to transform straight lines into curves.

2.2.3 Metric

Perhaps the most significant of the components to consider given a specific application is the choice of metric [30]. Certain metrics are less effective in the case that the images to be registered are of different modalities. Typical examples of unimodal registration metrics include the sum of squared-differences between image intensities and normalized cross-correlation [30]. In the case of multimodal registration, any given tissue present in the data may generate very different intensity values when acquired using another modality. Concepts from information theory, such as mutual information [18, 40, 53] and the Kullback-Leibler measure [15], have provided a solution to the multimodal registration application.

Mutual Information has proven to be an extremely useful metric, and may be interpreted as a measure of how much information in one image is contained in another image, in terms of the image intensities [40]. The exact correspondence between these intensities is not necessarily required to be known, and therefore this metric is suitable for multimodal registration applications.

In order to describe mutual information between two random variables, the Shannon Entropy for a random variable A and event a is first defined as

$$H(A) = -\sum_{a} p_A(a)\log(p_A(a)). \tag{2.30}$$

Given another random variable B for which the entropy is similarly defined, the joint entropy of A and B can be written as

$$H(A,B) = -\sum_{a,b} p_{AB}(a,b)\log(p_{AB}(a,b)), \tag{2.31}$$

which is equal to the sum of the individual entropies of A and B in the case that A and B are independent. The mutual information defined in terms of the joint entropy as

$$I(A,B) = H(A) + H(B) - H(A,B), \tag{2.32}$$

which serves as a measure of the difference in the joint entropy associated with the degree of dependence between two random variables A and B [30, 53]. As a result, maximum value of mutual information denotes an optimal correspondence between two random variables.

Mutual information can be used in order to provide a similarity metric between two images, A and B. This can be accomplished by first estimating the marginal and joint probability densities. One way to provide this estimation is through the normalization of the joint and marginal histograms of the image [30]. The joint histograms are constructed by initializing a 2D array, the rows and columns of which correspond to each image intensity bin, and each dimension is associated with an image. The value of each 2D histogram bin is therefore determined by the intensities of both images. Besides histogram normalization, continuous density estimation techniques such as Parzen

windowing may also be used to estimate the densities [23]. Following the density estimation, the entropies can be computed using Equations (2.30) and (2.31) and the mutual information, *I(A,B)*, can be obtained using (2.32). The mutual information is assumed to be maximal when the images are geometrically aligned. Therefore, when using this metric within the registration framework, the distance as described in Equation (2.29) must be maximized.

2.2.4 Optimization Schemes

Lastly, numerous optimization algorithms exist which may be incorporated into the registration framework. The optimizer is used to find a minimum or maximum of a function based on the metric. This is accomplished by traversing the transformation space with the fixed image, the parameters of which are defined by the type of transformation. The optimizer therefore provides the search algorithm which ultimately finds the transformation providing the best match between the images according to some metric.

Because this metric function may not be differentiable, or because the computational complexity in differentiating the function may be expensive, techniques have arisen to simplify this task. These optimization methods include techniques such as gradient decent, Newton's method, and Quasi-Newton [9] methods, which are based on Taylor's series expansion. Also, a number of search methods exist such as Brent's Line Search [55], Simplex Search [62], and Powell's Method [54], which are relatively general, simple to implement, and do not require the function derivatives to be computed.

2.3 Insight Segmentation and Registration Toolkit (ITK)

Many of the medical image analysis concepts visited throughout this chapter have been previously implemented as ITK classes. Therefore, this software was used extensively through the course of this research to test the ability of these methods to perform the required registration and segmentation tasks. This section provides a background of the segmentation and registration capabilities of ITK.

2.3.1 Overview of Segmentation in ITK

ITK provides a number of fundamental segmentation algorithms which can be incorporated into a complete segmentation application [30]. The components involved in a segmentation application can be tuned to accommodate the type of data and the characteristics of the anatomical region to be segmented. Typically, several filters in the pipeline may be used in pre-processing the data prior to the segmentation. These may include denoising filters or filters used in extracting image features such as edges. Filters may also be used to refine the segmentation result in a post-processing step.

The types of segmentation filters available in ITK fall into three main categories, including: (1) region growing segmentation techniques, (2) segmentation methods based on watersheds, and (3) level set segmentation algorithms [30]. A number of hybrid approaches are also available, which make use of combinations of these region-based and boundary-based techniques. This approach exploits the strengths and weaknesses in each technique in order to improve the quality of the segmentation.

Region growing segmentation filters detect object regions based on the similarity of intensity values of adjacent pixels [30]. The process of region growing begins through the user specification of initial seed pixels, which are indicated as inside or outside the object of interest. These filters then evaluate neighboring pixels according to a criterion which varies depending on the exact region growing filter used. These neighboring pixels are then determined to be inside or outside of the object. This process continues until all pixels have been visited and labeled.

Watershed techniques have acquired its name from the manner in which the segmentation is performed [30]. Consider a feature image f, which serves as a height function of the image, and which is typically formed from the gradient magnitude or curvature measures within the image. The segmentation is accomplished using gradient decent on this function f to identify the paths of steepest descent terminating at the same local minimum of f. The result determines the basins which provide the segmentation. Watershed segmentation techniques are less sensitive to thresholds than the region-growing

techniques. Also, a hierarchy of segmentations is determined which allows a user to assist in the segmentation process.

ITK Level Set Segmentation Techniques

The level set method filters available in ITK make use of a generic level set equation used to update the embedding function ϕ according to the partial differential equation [30]

$$\frac{\partial \phi}{\partial t} = -\alpha A(x) \cdot \nabla \phi - \beta P(x)\left|\nabla \phi\right| + \lambda Z(x)\kappa\left|\nabla \phi\right|. \qquad \textbf{(2.33)}$$

Here, A represents the advection term used to attract the curve to the edges, P is the propagation term, and Z represents the spatial modifier associated with the curvature κ. α, β, and γ are constants which provide the weight of influence for each of the three terms in the evolution equation. The terms to be included in the application will vary depending upon the specific level set filter used.

ITK provides an implementation of geodesic active contours based upon the paper proposed by Sapiro et al in [10]. This implementation extends the evolution equation in Equation (2.33) to include a curvature term as well as an additional advection term. These terms provide a smoothness constraint to the evolving contour, and attract the curve to the object boundaries, respectively.

As mentioned, the inputs to this filter include the initial embedding function and the feature image, which in this case refers to the speed image computed as a function of the gradient magnitude image. In a typical application, a user specifies an initial contour in the form of the zero level set of an embedding function ϕ [10]. The result of this segmentation method is relatively sensitive to the initialization, and therefore, the initial contour should be placed inside the boundaries of the target object. In order to produce the speed image, the original image is typically processed in the following manner. The image is first smoothed using an edge-preserving smoothing filter, such as an anisotropic diffusion filter, to remove noise. This smoothed image is then passed to a second filter used to compute the gradient magnitude image. Finally, the output of the gradient

magnitude filter is passed to a filter such as the sigmoid filter which makes use of Equation (2.17) to provide the speed image.

Upon executing the segmentation process for a number of iterations, the output of the segmentation filter is a single image representing ϕ at the final time step. The zero level set of the evolved embedding function provides the surface of the segmented object computed at subpixel accuracy. This result is typically passed through a threshold filter which sets the values denoting the inside of the curve to 1 and outside to 0. The methods presented in Chapter 3 provide a detailed description of an ITK application which uses the implementation of geodesic active contours.

2.3.2 Overview of Registration in ITK

ITK treats the registration problem as an optimization process used to determine the spatial correspondence between two images. The basic input to the registration method in ITK includes two images, one of which is designated as the fixed image, and the other the moving image. The registration framework consists of four primary components, each of which has a number of interchangeable options. These components include an interpolator, the type of transform, a similarity metric, and an optimization scheme.
The variety of types available for each of these components allows for a large range of possible combinations, leading to many unique registration methods. Interpolation functions existing in ITK include nearest neighbor, linear, B-spline, and Windowed Sinc interpolation methods. Many transforms are available, including translation, rigid, affine, and B-spline deformations, to name a few. ITK metrics include mutual information, mean squares, normalized correlation, and Kullback-Leibler distance. Finally, several optimization methods offered by ITK include gradient descent, one-plus-one evolutionary, Powell method, and the Nelder-Meade downhill simplex.

Upon plugging each of the components into the pipeline and specifying parameters for each, the process may be executed. The output of the registration process includes the parameters of the transformation required to map the fixed image to the moving image. Post-processing steps may include applying this transformation to the moving image in

order to provide the aligned image. The methods presented in Chapter 3 provide a detailed description of an ITK registration application which makes use of an affine transformation, mutual information based metric, and a Powell optimization filter.

Chapter 3: A New Speed Function for Shape Prior Segmentation

The approach discussed in this chapter incorporates shape information into the speed term used in many curve-evolution based segmentation algorithms. This makes it possible to perform segmentation based on a shape prior using existing algorithms and code by simply substituting the speed image. The new speed function slows curve evolution near edges in the image to be segmented as well as near edges in a registered atlas image. In this chapter, the details of the atlas construction and registration step, which only applies to the segmentation of the real MRI data, are first provided. This is followed by a discussion of the methods used to compute the new speed term, as well as the segmentation step which uses an implementation of geodesic active contours.

3.1 Overview

To overcome the complications associated with extracting the MRI rabbit brain data, an atlas image I_{atlas} was constructed and used to guide the process. This segmentation procedure includes the following steps:

Step 1: Register the novel data set with I_{atlas}.
Step 2: Compute the new speed term.
Step 3: Perform the segmentation using an embedding function initialization provided by the registration step, and by replacing a new speed term for the traditional speed in an implementation of geodesic active contours.

The atlas was constructed using a twelve-member subset of the MRI data sampled from the population of available data. This was then hand segmented to produce a mask labeling the brain and non-brain tissue regions. A signed distance function was computed from this binary image to give ϕ_{atlas} representing the brain surface of I_{atlas}. Each novel data volume I to be segmented is registered to I_{atlas}. This process finds a transformation matrix T_{affine} which is used to map ϕ_{atlas} onto I.

Within the population of data, anatomical variations do not allow for such a registration step to provide an adequate segmentation result. However, the step may be used to produce a close approximation to the final segmentation by using ϕ_{atlas} as an initial embedding function in a final level set segmentation step. The input to the segmentation step also includes the speed image g, which is typically computed from the gradient magnitude of I using an edge potential function such as (2.17). Because of the low-contrast edges at the boundary of the brain, a new speed function, which also incorporates information provided by I_{atlas}, is used to compute the speed image g. Upon producing the initial embedding function ϕ_0 and the new speed map g, several iterations of geodesic active contours segmentation are performed to achieve the final segmentation result.

3.2 Atlas Construction

In order to construct I_{atlas}, 12 data sets were sampled from the population of existing MRI data. Of these, a single data set $I_{TrainingReference}$ was chosen to provide a basis for which the remaining training samples $I_{Training}(k)$, where k = 1, 2, …11, were to be registered. Each data set in $I_{Training}$ was then registered to $I_{TrainingReference}$ using the registration framework of ITK.

The two-step algorithm used to register $I_{TrainingReference}$ and $I_{Training}(k)$, for each k, finds the affine transformation T_{affine}. This transformation registration solves for translation, rotation, scaling, and shearing. Components used in this ITK registration application include Mattes mutual information image metric and the Powell optimizer. A complete description of the registration process used in aligning these data sets is provided in section 3.3.

Following the registration, the transformed training data sets $I'_{Training}(k)$ were then used to give the atlas image, I_{atlas}, by computing the median of the intensity values at each voxel. This result was then manually segmented by identifying and labeling the voxels in the brain to produce a binary image.

The binary image is then used as input to the itk::SignedDanielIsonDistanceMapImageFilter. This filter takes a binary image as input and computes a distance map approximated to pixel accuracy. The signed distance refers to the Euclidean distance between a current pixel and the closest point of the closest object while using the sign of the value to indicate whether the pixel is located inside or outside the binary object. ITK convention stipulates that the values of the inside pixels are negative, while the values of the outside pixels are positive. The output of this filter is ϕ_{atlas}, the signed distance map representation of the atlas brain surface.

The results of the intermediate steps involved in constructing the atlas, including the registration step, the averaging process used to obtain I_{atlas}, and the manual brain segmentation, are provided in Chapter 5.

3.3 Registration Algorithm

As mentioned in section 3.2, the registration procedure described in this section is used in the construction of I_{atlas}. Also, this procedure is used in the initial registration step involved in the segmentation of I.

This two-step registration algorithm performs an initial registration step to find $T_{translation}$. This quickly corrects for any large translational misalignment between the two images. $T_{translation}$ is then used as the initial transform in a second registration step, which finds the affine transformation, T_{affine}. In the registration algorithm involved in registering a novel image, T_{affine} is applied to ϕ_{atlas} to obtain ϕ_0, the initial embedding function used in the geodesic active contours segmentation. Figure 3.1 depicts the registration framework in relation to the inputs, outputs, and ITK registration components.

The registration procedures used in the atlas construction and in the initial stages of the segmentation are essentially the same. However, minor variations in the preprocessing and the postprocessing stages exist. The procedure described in this section emphasizes the registration of I_{atlas} to I used in determining ϕ_0, although any discrepancies are

specified. This section provides implementation details of the preprocessing, initialization, optimization, and postprocessing stages of the registration procedure.

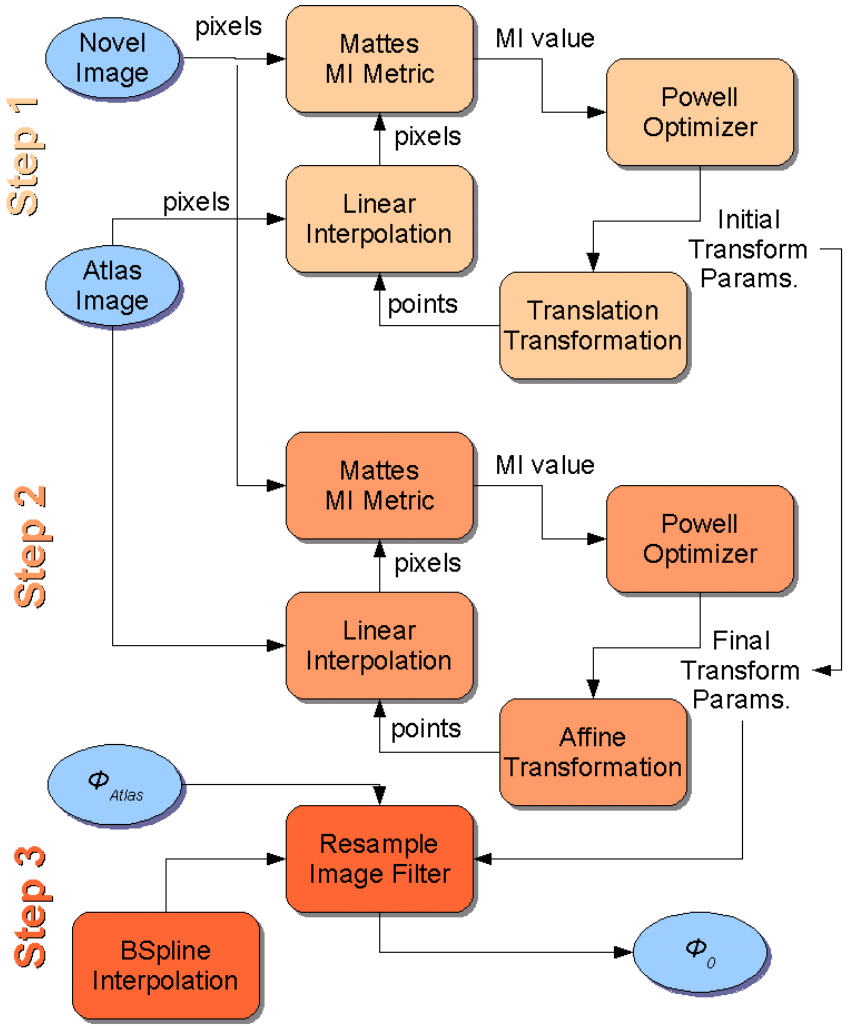


Figure 3.1. The flow chart of the ITK registration framework used in the registration step of the brain extraction algorithm. Inputs I_{atlas}, I, and ϕ_{atlas} are used to obtain ϕ_0, which is then used in the segmentation step.

3.3.1 Preprocessing

In registering I_{atlas} to I, the two images are first read from file and cast to type float. The statistical distributions of the images' intensity values are then normalized to zero mean and unit variance using the itk::NormalizeImageFilter class in order to simplify the computation of the mutual information metric within ITK. A low-pass filter, provided in

the itk::DiscreteGaussianFilter class, is then used to smooth I. Performing this filtering step improves the registration process in terms of robustness to noise. The variance parameter of the Gaussian kernel used in the itk::DiscreteGaussianFilter class is specified using the SetSigma( ) method. This filter is not applied to I_{atlas} due to the assumption that the averaging process involved in the atlas construction has the effect of smoothing the noise components of the data.

3.3.2 Initialization

A generic registration interface is provided by the base class, itk::ImageRegistrationMethod [30]. This class expects two input images, which are designated as the fixed and the moving image using the methods SetFixedImage( ) and SetMovingImage( ). The coordinate system of the fixed image is used by this class as a reference. In the construction of I_{atlas}, $I_{Training}(k)$ is designated as the moving image while $I_{TrainingReference}$ is denoted as the fixed image. In the registration of I_{atlas} to I, I_{atlas} is set as the moving image and I is designated as the fixed image. Note that the actual inputs to the registration filter are the preprocessed versions of the images as described in the previous section. The registration process then determines the coordinate transform that maps points from the space of the fixed image to the space of the moving image.

3.3.3 Registration Framework Components

The itk::ImageRegistrationMethod base class is used to connect a generic interpolator, transform, metric, and optimizer, thus allowing for the run-time selection of the exact types of these components to be used. Upon initializing a registration application, an object of this base class is instantiated. Once the types of the registration framework components are chosen, an object corresponding to each of these classes is instantiated. The components are then connected through the itk::ImageRegistrationMethod base class, using the methods SetInterpolator( ), SetTransform( ), etc. The parameters corresponding to each of these components can be set at any time prior to executing the registration process.

Interpolation

An interpolator is used during the registration process to estimate image intensities of the moving image when the transformation maps pixels to non-grid positions of the coordinate system. To minimize the computation time, a linear interpolation scheme provided by the itk::LinearInterpolateImageFunction class was used. No parameters are required in implementing this class.

Transformation

As mentioned, the transform parameters define the search space for the optimizer within the registration framework [30]. The more parameters this transform contains, the longer the computation time associated with performing the registration. In order to reduce the time required, the registration is broken up into two stages: (1) resolve the translational misalignment to provide a quick initial result, and then (2) solve for rotation, scaling, and shearing to achieve a refined alignment result.

The itk::TranslationTransform class is plugged into the transform component of the registration filter in the initial registration step. The transform matrix associated with this class represents a simple translation of points in each of the image dimensions. Therefore, each component of the transform matrix output is associated with a dimension of the input image. This registration step is used to quickly solve for $T_{translation}$.

The affine transform, provided by the itk::AffineTransform class, is a transformation composed of rotation, scale, translation, and shearing [30]. It is represented by an N x N and an N x 1 vector, where N is the number of dimensions in the image space. The number of parameters in the transformation matrix is therefore (N + 1) x N. Here, the first N x N parameters define the matrix, corresponding to scaling, rotation, and shearing, in column major order. The last N parameters define the translation associated with each dimension of the image space. The result of the first registration step, $T_{translation}$, is used as the initial translation in the final affine registration step which finds T_{affine}. The initial translation is set using the SetInitialTransformParameters( ) method of the registration base class. No other parameters are required to be set for this component.

Metric

Mattes mutual information measure is a variation to the mutual information metric as proposed by Mattes et. al in [45]. The primary difference in Mattes implementation is that only a single subset of the intensities are drawn from the image, instead of using every pixel intensity. This sample set is used to compute the joint and marginal probabilities at the discrete histogram bins. The entropy is then computed by summing over these bins. This implementation results in a smoother cost function to be used within the optimization process, as well as a faster computation with only a minimal loss in accuracy [30].

The implementation provided in the itk::MattesMutualInformationImageToImageMetric class is based on [45, 46, 66]. Using this class requires that only a set of the intensity values are used to compute the metric value [30]. The method SetNumberOfSpatialSamples( ) is used to specify the number of pixel intensities to sample from the image. This sample set is used to compute the joint and marginal probability distribution function (PDF) at the discrete histogram bins spread evenly throughout the range of image intensity values using Parzen histograms. The number of bins to be used is set using the SetNumberOfHistogramBins( ) method. The entropy is then computed by doubly summing over these bins. The value returned is the negative mutual information.

The metric classes available in ITK support region based evaluation to allow a user to specify the region of the image from which the metric is to be computed [30]. Although the region is not specified in the registration procedure used in constructing the atlas, a dilated version of the manually segmented atlas is used as an itk::SpatialObject to specify this region. First, a structuring element is instantiated in the form of a binary ball with radius of 20 using the itk::BinaryBallStucturingElement class in ITK. This structuring element is set as the kernel for the itk::BinaryDilateImageFilter using the SetKernal( ) method. Upon inputting the atlas binary mask, it is dilated using this filter to produce a binary image indicating the region which contains the brain volume but excludes a large portion of the surrounding data. The SetFixedImageMask( ) and SetMovingImageMask( ) methods of the

metric class are used to specify that the dilated binary mask marks the image region which is to contribute to the value of the metric.

Optimizer

The optimizer traverses a transformation space specified by the type of transform in search of a minimum (or maximum) of the cost function defined by the metric [30]. Because the mutual information metric is employed, the optimizer must be suited for handling a single valued cost function. In the registration methods implemented, the itk::PowellOptimizer class is used as the optimizer component. This class provides an implementation of Powell optimization which uses a Brent line search [30]. To briefly describe this optimization scheme, consider the *N*-dimensional parameter space characterizing the transform. At each iteration of the optimization procedure, the optimizer seeks to minimize the Mattes mutual information value in each of the *N* dimensions [54].

Within this optimizer class, the method SetStepLength( ) is used to set the initial step distance that is to be taken in each line direction of the parameter space. The StepTolerance( ) method is used to terminate the optimization process when the current parameter values are known to be within the distance specified. Similarly, the ValueTolerance( ) method is used to terminate the optimization when the value of the metric at the current parameters is known to be within the tolerance specified. Because the parameters of the transform may have different dynamic ranges, angles versus translations i.e., the scale to be used for each parameter of the transform is set using the SetScales( ) method. Also, the SetMaximize( ) method is called with the parameter false to indicate that the Mattes mutual information metric should be minimized. Finally, the SetMaximumIteration( ) method is used to limit the number of iterations to be performed.

3.3.4 Registration Optimization

The registration procedure begins by reading in the images to be registered and performing the preprocessing steps. This is followed by initializing each of the registration components, setting the necessary parameters associated with each of the

components, and connecting the data and process objects into a pipeline using the registration base class, as shown in Figure 3.1.

Upon construction of the pipeline, the registration begins by performing the initial translation registration. Optimization terminates upon arriving at a tolerance value indicated by the optimizer, or by reaching the maximum iteration number indicated. The output of the registration filter, $T_{translation}$, is then used as input to the second registration step which performs the affine registration. The optimization of the affine registration step proceeds in an analogous manner, with the exception of the transformation space in which the optimizer traverses. The result of this step gives T_{affine}, which can be then used to map the fixed image to the moving image.

3.3.5 Postprocessing

In the case of the registration between the atlas I_{atlas} and a novel data set I, T_{affine} is used to provide a transformation which maps I_{atlas} onto I. This transformation is applied to the signed distance map of the binary atlas mask ϕ_{atlas} to get

$$\phi_0(x) = \phi_{atlas}\left(T_{affine}(x)\right), \qquad \textbf{(3.1)}$$

which is to be used as the initial embedding function in the segmentation step. This is accomplished using an itk::ResampleImageFilter, an ITK class used to resample an image according to a specified coordinate transform [30]. Here, the image ϕ_{atlas} is set as the input to be transformed according to T_{affine}. The type of interpolation to be used by this filter is specified as the itk::BSplineInterpolateImageFunction, the details of which are provided in [30]. Finally, the spacing, size, and direction for the output image are specified by setting I as the reference image and using the UseReferenceImage( ) method available in the itk::ResampleImageFilter class. The output of this filter is ϕ_0.

Similarly, in the registration procedure used in constructing the atlas, the transformation T_{affine} computed using $I_{Training}(k)$ is used to align $I_{Training}(k)$ to $I_{TrainingReference}$. As performed in the transformation of the embedding function described above, this is accomplished using itk::ResampleImageFilter and the itk::BSplineInterpolateImageFunction interpolator. Here, the image $I_{Training}(k)$ is set as the input to be transformed according to T_{affine}. Finally, the

spacing, size, and direction for the output image are specified by setting $I_{TrainingReference}$ as the reference image and using the UseReferenceImage( ) method. The output of this filter is the aligned training data image, $I'_{Training}(k)$. Examples of the results obtained by performing the registration step required in both the construction of I_{atlas}, and in mapping ϕ_{atlas} to a novel data set, are provided in Chapter 5.

3.4 Geodesic Active Contour Level Set Segmentation Method

The segmentation approach used in extracting the brain volume from the image I uses a segmentation filter in ITK which implements the geodesic active contours. The pipeline which performs the segmentation includes a series of filters. These components are used to perform image smoothing, compute the gradient magnitude image, compute the speed image, and perform the segmentation. This process is followed by a thresholding step to isolate the region of interest located inside the evolved interface, which is represented by the negative values of the embedding function. This section describes the segmentation procedure and provides insight into key parameters used within the components of the segmentation pipeline. Details of the preprocessing stage, speed image computation, level set evolution, and post-processing stage of the segmentation are also provided in this section. The code corresponding to an implementation of this method is provided in the Appendix.

3.4.1 Preprocessing

In order to begin the segmentation process, the image I is smoothed in order to denoise the image. This is accomplished using an edge-preserving, anisotropic diffusion filter implemented in by the itk::CurvatureAnisotropicDiffusionImageFilter class. The time step, conductance, and the number of iterations are set using the SetTimeStep( ), SetConductanceParameter( ), and SetNumberOfIterations( ) methods of this class. An additional class method UseImageSpacingOn( ) is used to indicate that the image spacing is to be accounted for while undergoing the smoothing process. Next, the smoothed image is passed to an itk::GradientMagnitudeRecursiveGaussianImageFilter, which computes the gradient magnitude image through the convolution with the first derivative of a Gaussian. The variance σ^2 parameter of the Gaussian is set using the SetSigma( ) method.

3.4.2 A New Speed Function for Shape Prior Segmentation

The traditional speed image is computed by passing the output of the gradient magnitude filter to an itk::SigmoidFilter, which results in an edge potential image. This filter provides a pixel-wise implementation of Equation (2.17). The key parameters of this filter include α andβ, the implications of which are described in section 2.1.2. These values are set using the SetAlpha( ) and SetBeta( ) methods of the itk::SigmoidFilter class.

In cases in which the edges are not detected by the gradient magnitude, edge information from the atlas may be used in these locations. The proposed speed function therefore combines the gradient information with atlas information. Computing the proposed speed term is summarized in the following steps:

> **Step 1:** Compute two speed map terms, referred to as g_{High} and g_{Low}, using the traditional speed function given by Equation (2.17). The two terms are computed using different values of the constant β.
> **Step 2:** Take the difference of the two speed terms, but apply the Dirac measure, $\delta\varepsilon$, to restrict the effect of the difference of g_{High} and g_{Low}, to the surface of the atlas shape.

In the first step, two speed map terms, referred to as g_{High} and g_{Low}, are computed using (2.17). Here, β_{High} and β_{Low} are substituted for β in (2.17) to compute the g_{High} and g_{Low} terms of the new speed map. The values of β_{High} and β_{Low} are tuned by the user to generate speed maps g_{High} and g_{Low} that will remain equal in regions of strong edges and homogeneity, but will differ in value where there is intermediate edge information. β_{Low} is chosen to be the value of the gradient magnitude, below which, the image at that location is to be considered homogeneous. β_{High} is chosen to be the value of the gradient magnitude, above which, the image at that location is to be considered an edge. Figure 3.2 shows the original synthetic image in Figure 3.2 (a) from which the gradient magnitude image of Figure 3.2 (b) is generated. From the gradient magnitude, the two speed maps are computed as shown in Figure 3.2 (c) and (d). Once these speed terms are found, their difference can then computed as $g = g_{High} - g_{Low}$, as shown in Figure 3.3.

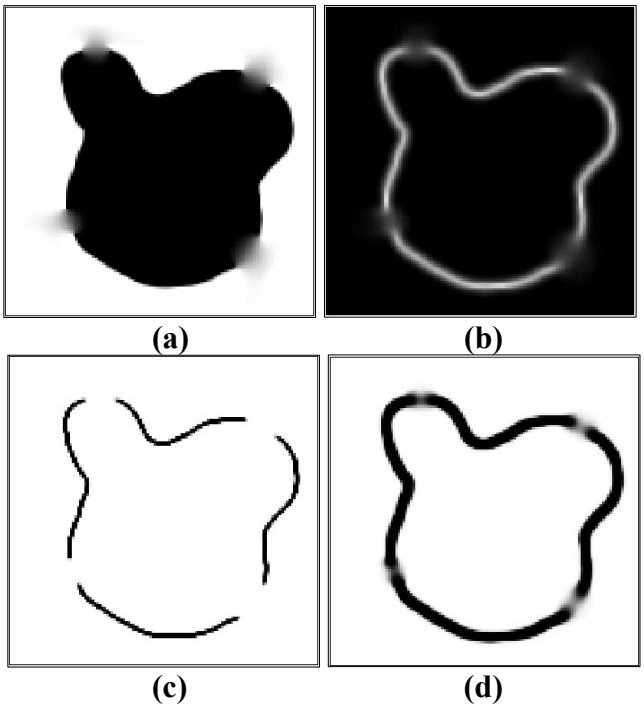


Figure 3.2. Computing the g_{High} and g_{Low} terms. (a) The original image I is used to compute (b) the gradient magnitude image. This is then used to compute the speed terms: (c) g_{High} and (d) g_{Low}.

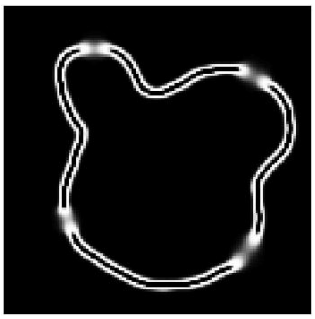


Figure 3.3. The difference between the speed terms g_{High} and g_{Low} computed as $g = g_{High} - g_{Low}$.

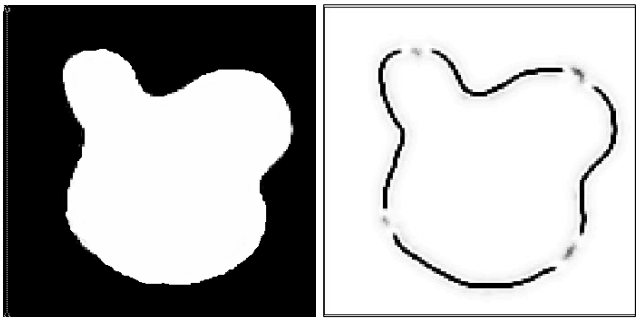
However, the result shown in Figure 3.3 is isolated to the region immediately surrounding the atlas brain surface, as indicated by the zero level set of ϕ_0, by formulating the new speed function as

$$g = g_{High} - \delta_\varepsilon(\phi_0) g_{Low}, \tag{3.2}$$

where ϕ_0 denotes the initial embedding function, and δ_ε denotes the regularized Dirac measure as proposed in [11], given by

$$\delta_\varepsilon(z) = \frac{1}{\pi}\left(\frac{\varepsilon}{\varepsilon^2 + z^2}\right), \tag{3.3}$$

where ε represents the width of the function. Using ϕ_0 computed in (3.1), Equation (3.3) is then used with ε = 1.0 to get the $\delta_\varepsilon(\phi_0)$ term of the new speed function. Finally, the speed map is computed using (3.2). The binary atlas, from which the signed distance map is computed to give ϕ_0, is shown in Figure 3.4 (a). The speed map computed using the new speed function (3.2) is shown in Figure 3.4 (b). Here, it can be observed that the apparent gaps in edge information are filled by edge information provided by the atlas.



(a) **(b)**

Figure 3.4. The new speed term. (a) The binary atlas image used to compute ϕ_0, and (b) the new speed term.

Upon computing the new speed term, the segmentation is performed by replacing this speed term for the traditional speed in an implementation of geodesic active contours.

Level Set Segmentation Method

The itk::GeodesicActiveContourLevelSetImageFilter class used to perform the segmentation expects two inputs including: (1) an initial level set embedding function ϕ_0, and (2) the speed image g. The initial level set ϕ_0 provided by the preceding registration stage is initialized using the SetInput( ) method. The proposed speed image g computed using the methods described in the previous section is input using the SetFeatureImage( ). The weights given to the advection, propagation, and curvature terms of the energy are set using SetAdvectionScaling( ), SetPropagationScaling( ), and SetCurvatureScaling( ) methods, respectively. The components of the segmentation pipeline which implements geodesic active contours are depicted in Figure 3.5.

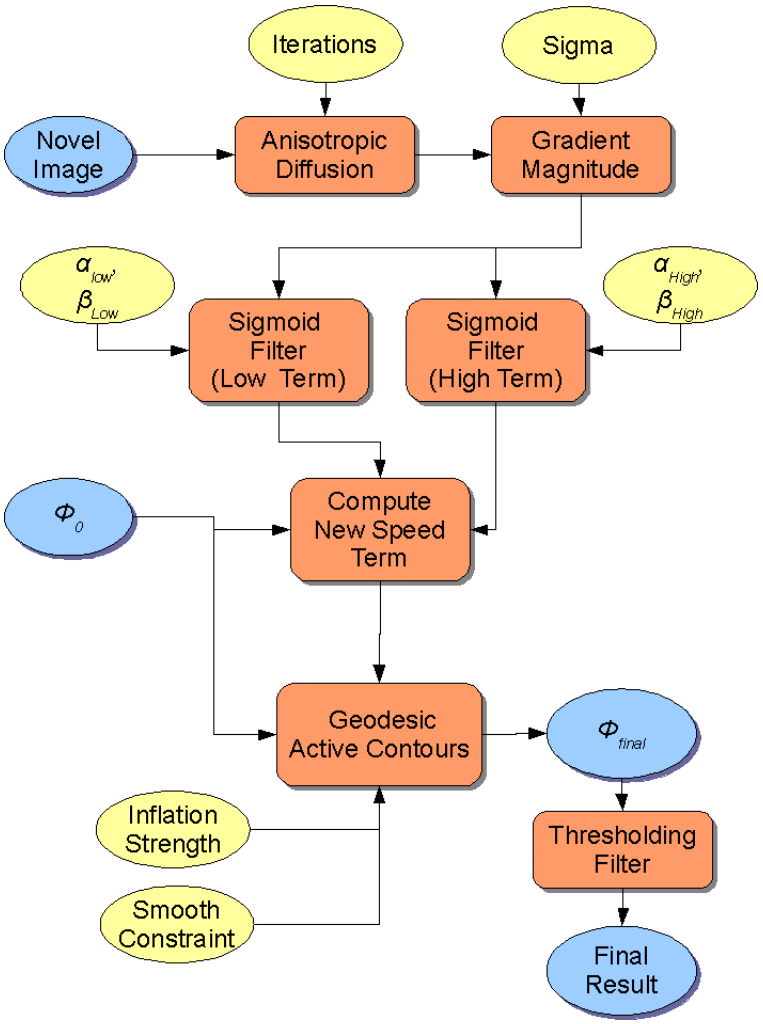


Figure 3.5. Diagram of the segmentation application using the geodesic active contour filter in ITK. The level set input ϕ_0 is provided by the registration step as indicated in Figure 3.1. The new speed term is computed as described.

The user terminates the segmentation process upon visual inspection of a close segmentation provided by the evolving curve represented by ϕ. The segmentation will also terminate after a predefined number of iterations. When the segmentation completes, the final result ϕ_{final} is obtained from the itk::GeodesicActiveContourLevelSetImageFilter. This is then passed to a thresholding filter, itk::BinaryThresholdImageFilter, which sets the negative values denoting the inside of the curve to 1. The positive values representing pixels on the outside of the curve are set to 0. This is accomplished by setting the upper threshold, lower threshold, inside value, and outside value parameters of this filter. The result is a binary mask which labels the brain volume and non brain tissues of the data set.

Chapter 4: Fast Level Set Segmentation using a Shape Prior

In another segmentation approach, a method is proposed which uses concepts of image moments to provide a shape constraint to be incorporated into the fast algorithm for Chan-Vese level set optimization presented in [61]. This method is similar to the method proposed by Chan and Zhu [12]. The main difference is that the transformation linking the shape prior to the segmentation is determined by the set of moments computed from the evolving segmentation and the shape prior.

The proposed method first determines a set of moments from shape image I_{shape}. At each sweep of the set of image pixels during the segmentation process, a set of moments are computed from the current segmentation specified by the embedding function, ϕ. In some situations, it is useful to build a hierarchy of these moments which correspond to the sub-regions of the images. This is accomplished by first computing the moments of the complete image. The image is then split into two regions along the principle axes, and the moment computation process is repeated for each of these regions.

From the moments, a transformation matrix is determined which maps I_{shape} onto the shape represented by the current segmentation as $H(\phi)$, where H is the Heaviside function. The transformation allows for translation, rotation, and scaling differences between the shape and image. The shape comparison term used in the energy is computed as the difference between I_{shape} and $H(\phi)$. The fast level set principle described in [61], implemented with the additional shape term, is then used to perform the update of ϕ. The outline of the segmentation algorithm can be summarized as follows:

> **Step 1:** Initialization of the embedding function, ϕ_0. The initial segmentation is constructed as $\phi = +1$ for the initial inside region and $\phi = -1$ for the initial outside region. The set of moments for the shape prior are also computed.

Step 2: Advance. Compute the region means c_1 and c_2, the set of moments corresponding to $H(\phi)$, and the transformation matrix. The segmentation process then advances through the image, pixel by pixel. At each pixel, the change in energy associated with changing $\phi(x,y) = -\phi(x,y)$ is computed.

Step 3: Update. Switch $\phi(x,y) = -\phi(x,y)$ for each of the pixels (or a fraction of these pixels) that demonstrate a negative change in energy. Repeat Step 2 until the segmentation converges or as indicated by a maximum number of iterations.

The details of the implementation used to accomplish each of these steps are discussed in this section. The code corresponding to an implementation of this method is provided in the Appendix.

4.1 Image Moments

Prior to the segmentation process, the set of moments are computed from the shape prior image I_{shape}. Generally, a binary image serves as the shape representation. Applying Equation (2.24) to the 2D N x M digital image $I_{shape}(x, y)$ gives the 2D moments as

$$m_{pq} = \sum_{y=0}^{M-1}\sum_{x=0}^{N-1} x^p y^q I_{shaope}(x,y)dxdy\,, \tag{4.1}$$

where p, q = 0, 1, and 2. Upon computing moments, the centroids are computed using

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}\,. \tag{4.2}$$

The second order central moments are then computed as

$$\begin{aligned} \mu_{11} &= \frac{m_{11}}{m_{00}} - \bar{x}\cdot\bar{y}, \\ \mu_{02} &= \frac{m_{02}}{m_{00}} - \bar{y}^2, \\ \mu_{20} &= \frac{m_{20}}{m_{00}} - \bar{x}^2. \end{aligned} \tag{4.3}$$

The three second-order central moments, μ_{20}, μ_{02}, and μ_{11}, characterize the size and orientation of the image [72]. Considering only these moments, the original image is equal to the image ellipse of the same size, orientation, and centroid location. The orientation of the image is determined as the orientation of the principal axes of the corresponding image ellipse, referred to as the tilt angle, according to the equation

$$\theta = \left(\frac{1}{2}\right)\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right). \qquad \textbf{(4.4)}$$

The conventions of [38] require the angle to be chosen as the angle between the x-axis and the semi-major axis of the image ellipse. Also, the principle axis of the arc tangent function is chosen so that $-\pi/2 \leq \tan^{-1}x \leq \pi/2$. Table 4.1 summarizes the choice of the tilt angle to be chosen according to these conventions [72].

Table 4.1. Tilt angle θ for the different sign cases of the second order central moments.

μ_{20} - μ_{02}	μ_{11}	$\theta\left(\xi = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right)$
0	0	0
0	+	+ 45°
0	−	- 45°
+	0	0
−	0	- 90°
+	+	$\left(\frac{1}{2}\right)\tan^{-1}(\xi)$
+	−	$\left(\frac{1}{2}\right)\tan^{-1}(\xi)$
−	+	$\left(\frac{1}{2}\right)\tan^{-1}(\xi)+90^{\circ}$
−	−	$\left(\frac{1}{2}\right)\tan^{-1}(\xi)-90^{\circ}$

In the case that the moments are to be computed for the image sub-regions, the image is subdivided along the principal axis, resulting in two child images. The process is then repeated for each of these child images for a specified number of subdivision levels. Generally, one or two levels are used.

The moments are similarly computed for the current segmentation prior to each sweep of the segmentation algorithm over the image pixels. The difference is that I_{shape} of Equation (4.1) is replaced by $H(\phi)$, the heaviside function of the current segmentation. Applying the Heaviside function to ϕ results in a binary image.

4.2 Transformation

From the moments and tilt angle computed from both I_{shape} and $H(\phi)$, a transformation matrix **T** is constructed which maps the pixels of I_{shape} to the corresponding pixels of the current segmentation. In order to accomplish this transformation, the transformation matrix is constructed as the product of several components. These include a (1) centering matrix, (2) translation matrix, (3) rotation matrix, and (4) scaling matrix. The centering matrix **cen** translates the centroids of $H(\phi)$ to the origin, and is constructed using the image centroids as follows.

$$\mathbf{cen} = \begin{bmatrix} 1 & 0 & -\bar{x}_{image} \\ 0 & 1 & -\bar{y}_{image} \\ 0 & 0 & 1 \end{bmatrix}, \qquad \textbf{(4.5)}$$

The centering component of the transform in combination with the translation component maps the centroids of I_{shape} to the location of the centroids of $H(\phi)$. The translation matrix **trans** is constructed using the centroids of I_{shape} as

$$\mathbf{trans} = \begin{bmatrix} 1 & 0 & \bar{x}_{shape} \\ 0 & 1 & \bar{y}_{shape} \\ 0 & 0 & 1 \end{bmatrix}, \qquad \textbf{(4.6)}$$

Next, the covariance matrices are first constructed using the central moments computed from I_{shape} and $H(\phi)$. The covariance matrix of a pattern is used to decouple correlated features and to scale the features to make the pattern compact [38]. The covariance matrix **M** of each image pattern is given in [38] as

$$\mathbf{M} = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix}, \qquad \textbf{(4.7)}$$

to give $\mathbf{M}_{\mathbf{shape}}$ and $\mathbf{M}_{\mathbf{image}}$, where μ_{ij} are the central moments. The eigenvalues of these covariance matrices are computed from the central moments as

$$\lambda_1 = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4{\mu_{11}}^2}}{2}, \tag{4.8}$$

$$\lambda_2 = \frac{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4{\mu_{11}}^2}}{2}. \tag{4.9}$$

The eigenvectors associated with these eigenvectors are computed as

$$\begin{bmatrix} e_{ix} \\ e_{iy} \end{bmatrix} = \begin{bmatrix} \frac{\mu_{11}}{\sqrt{(\lambda_i - \mu_{20})^2 + {\mu_{11}}^2}} \\ \frac{\lambda_i - \mu_{20}}{\sqrt{(\lambda_i - \mu_{20})^2 + {\mu_{11}}^2}} \end{bmatrix}. \tag{4.10}$$

The dot product of the eigenvector e_i corresponding to I_{shape} and $H(\phi)$ is computed and compared with the dot product computed for $-e_i$ corresponding to $H(\phi)$. The sign of e_i for $H(\phi)$ is chosen based on which dot product is greater.

The rotation matrix **E** rotates the eigenvectors of I_{shape} to align with the eigenvectors of $H(\phi)$. This is computed as the product of the eigenvectors matrix of the shape prior and the inverse of the eigenvector matrix of the image. From the eigenvectors, the rotation matrix **E** can be constructed as

$$\mathbf{E} = \mathbf{M}_{\text{shape}}\, \mathbf{M}_{\text{image}}{}^{\text{T}}. \tag{4.11}$$

Each component of the covariance matrix of I_{shape} can be scaled independently in order to arrive at a covariance matrix which is scaled to the covariance matrix of $H(\phi)$. This is accomplished by first constructing the scaling matrices **W** for each image, I_{shape} and $H(\phi)$ as

$$\mathbf{S} = \begin{bmatrix} 1/\sqrt{\lambda}_1 & 0 \\ 0 & 1/\sqrt{\lambda_2} \end{bmatrix}, \tag{4.12}$$

resulting in $\mathbf{S}_{shape}$ and $\mathbf{S}_{image}$. The scaling matrix **W** used in computing the transformation matrix is given by

$$\mathbf{W} = \mathbf{S}_{image}\,\mathbf{S}_{shape}^{-1}. \qquad \textbf{(4.13)}$$

In computing the final transformation matrix, the 2x2 rotation and scaling matrices, **E** and **W,** are first extended to 3x3 matrices to give a uniform matrix size between the various transformation components. This is accomplished by replacing the first 2x2 components of a 3x3 identity matrix with the matrix given by **E** or **W**. The final transformation matrix **T** is computed as the product of the components found using Equations (4.4) through (4.13) as

$$\mathbf{T} = \mathbf{cen}*\mathbf{trans}*\mathbf{E}*\mathbf{W}. \qquad \textbf{(4.14)}$$

Figure 4.1 illustrates the transformation, computed using the image moments of I_{shape} and $H(\phi)$, which is used to map I_{shape} to the current segmentation. In the case that the moments have been computed for the image sub-regions, a transformation matrix **T** is similarly computed using the moments corresponding to each of these sub-regions.

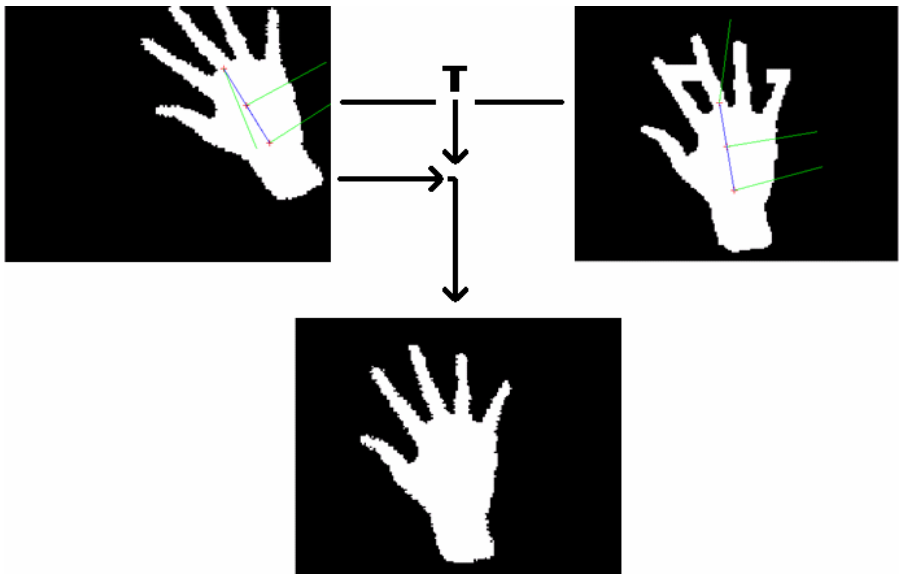


Figure 4.1. An example of the transformation **T** applied to I_{shape} (upper left) used to map this shape prior to $H(\phi)$ (upper right) to obtain the result used as the shape constraint (bottom). The directions of the principal axes, computed from the image moments, are indicated in the upper two images.

4.3 Fast Algorithm for Level Set Based Optimization

4.3.1 Initialization

The first step of this segmentation algorithm requires the initialization of ϕ by setting $\phi = +1$ in the initial region denoting the object and $\phi = -1$ in the initial region denoting the background. This is either accomplished by randomly assigning values of -1 or +1 to each pixel, or by initializing ϕ with an arbitrary shape such as a circle, with values of -1 outside and +1 inside. Also, the set of moments and tilt angle described in section 4.1 are computed for I_{shape}.

4.3.2 Segmentation Procedure

Prior to performing a sweep of the segmentation through the pixels of the image, the mean of the inside and outside regions, c_1 and c_2, as specified by the value of ϕ are computed using Equation (2.20). Also, the set of moments and tilt angle are computed for $H(\phi)$ using the procedure described in section 4.1. Using the set of moments of $H(\phi)$ and I_{shape}, **T** is computed using the methods presented in section 4.2. The segmentation then begins through the sweeping process over the image pixels. During this process, the energy change associated with switching the class of each pixel is computed. This energy is composed of three terms, including (1) the Chan-Vese model fitting term, (2) the smoothness constraint term, and (3) a shape constraint term.

Chan-Vese Model Energy Term

Assume at the current location (x, y), $\phi = 1$. The change in energy associated with switching $\phi = 1$ to $\phi = -1$, as proposed in [61], is computed as given in (2.22). Similarly, if at the current pixel location, $\phi = -1$, the change in energy is given in (2.23). The variance of the image intensity in the respective regions is taken into account as in [57] to give

$$\Delta F_{12} = \left(\frac{(I(x,y) - c_2)^2}{{\sigma_2}^2} \right) \frac{n}{n+1} - \left(\frac{(I(x,y) - c_1)^2}{{\sigma_1}^2} \right) \frac{m}{m-1}, \qquad \textbf{(4.15)}$$

$$\Delta F_{21} = \left(\frac{(I(x,y) - c_1)^2}{\sigma_1^{\,2}} \right) \frac{m}{m+1} - \left(\frac{(I(x,y) - c_2)^2}{\sigma_2^{\,2}} \right) \frac{n}{n-1}, \qquad \textbf{(4.16)}$$

where σ_1^2 and σ_2^2 represent the expected variance of the object and background regions.

Smoothness Constraint

As mentioned, Chan and Song approximate the length term of the Chan-Vese model energy in [61] using Equation (2.24). This constraint is used to minimize the length of the curve and therefore provide a smooth segmentation result. Equation (2.24) only provides values of 0, 1, or √2, depending on whether the set $\{\phi_{i,j}, \phi_{i+1,j}, \phi_{i,j+1}\}$ belong to the same regions. In this implementation, (2.24) is extended to include all elements in a 3x3 neighborhood within the summand as well to get a more accurate approximation.

Because the value of $H(\phi_{i,j})$ for a given point (i,j) is {0, 1}, there are 2^9 possible combinations for the 3x3 neighborhood considered. The change in length, ΔL, associated with switching $\phi_{i,j}$ to $-\phi_{i,j}$ for each 3x3 neighborhood combination can therefore be pre-computed. This will allow for ΔL to be determined and a look-up table constructed in a single step. This table then provides ΔL for all possible neighborhood combinations, minimizing the computation time and accuracy of the length approximation for any subsequent segmentation procedure.

The ΔL look-up table described is constructed as follows. Each neighborhood combination is assigned an index, which is computed according to its content as follows. Consider the 3x3 neighborhood $H(\phi_{i,j})$ matrix

$$\mathbf{nhood} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}, \qquad \textbf{(4.17)}$$

where $a_i = 0$ or 1 depending on the values of ϕ. The index is computed as

$$index = 2^0 a_1 + 2^1 a_2 + \ldots + 2^8 a_9. \qquad \textbf{(4.18)}$$

The length term L_1 is then computed for **nhood** using the extension to Equation (2.24) as described. The a_5 component of **nhood** is then switched, and from this new matrix the length term L_2 is computed. The change in length ΔL is then computed as

$$\Delta L = L_2 - L_1, \tag{4.19}$$

which is stored in the look-up table along with the index. This table can then used during the segmentation process to find ΔL for a given pixel location *(i,j)*. This is accomplished at each pixel by computing the index value from the 3x3 neighborhood of $H(\phi)$ surrounding the pixel location *(i,j)* using (4.18). The ΔL value is then found at the location of the look-up table provided by the index. The change in energy when applying this length constraint then becomes

$$\Delta E = \Delta F + \alpha \Delta L, \tag{4.20}$$

where ΔF is the change in energy computed from (4.15) or (4.16) and α is a positive constant used to weight the influence of the length constraint.

Shape Constraint

In addition to the change in energy found through Equations (4.15), (4.16), or (4.20), the shape energy term is incorporated after completing several sweeps of the segmentation procedure. Chan and Zhu define the shape comparison term in [12] as

$$E_{shape}(\phi,\psi) = \int_{\Omega} (H(\phi) - H(\psi))^2 \, dx, \tag{4.21}$$

where ψ represents the signed distance function of the shape prior. In this implementation, however, $H(\psi)$ is replaced by I_{shape}. In order to compute E_{shape}, the location $(\tilde{x}, \tilde{y})$ of the pixel in I_{shape} corresponding to the current pixel is determined using the transformation matrix **T** as follows:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = \mathbf{T} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{4.22}$$

In the case that a different **T** was computed for the sub-regions of the images, the **T** that is used in (4.22) is determined by first determining which region of the image the current

pixel is found in relation to the principle axis. The energy change can then be written for $\phi(x,y) = +1$ as

$$\Delta E_{12} = \Delta F_{12} + \alpha \Delta L + \lambda (H(\phi(x,y)) - I_{shape}(\tilde{x},\tilde{y})), \tag{4.23}$$

and similarly for the case that $\phi(x,y) = -1$

$$\Delta E_{21} = \Delta F_{21} + \alpha \Delta L + \lambda (H(\phi(x,y)) - I_{shape}(\tilde{x},y)), \tag{4.24}$$

where λ is a constant weighting the effect of the shape constraint. The additional energy term encourages switching the value of ϕ in locations where the shape provided by the current segmentation ϕ does not match I_{shape} .

Updating

Once ΔE has been computed at each pixel during this sweep, a threshold value ΔE_{thresh} is computed as a specified percent of the minimum negative value of ΔE. At the pixel locations that contain a negative value of ΔE which is less than ΔE_{thresh}, the value of ϕ is set to $-\phi$. The values of c_1 and c_2 are then updated, along with the set moments and transformation matrix.

Instead of recomputing the set of moments by summing over the entire image as in (4.1), these values can be quickly updated using the following procedure. Following each sweep over the image pixels, ϕ is switched as specified above according to ΔE. In the case that a pixel (x,y) has been switched from outside to inside ϕ, the moments computed in (4.1) may be updated as

$$\begin{aligned} m_{00} &= m_{00} + 1, \\ m_{10} &= m_{10} + x, \\ m_{01} &= m_{01} + y, \\ m_{11} &= m_{11} + x * y, \\ m_{20} &= m_{20} + x^2, \\ m_{02} &= m_{02} + y^2. \end{aligned} \tag{4.25}$$

Similarly, for each pixel (x,y) that is removed from inside ϕ, the moments computed in (4.1) may be updated as

$$
\begin{aligned}
m_{00} &= m_{00} - 1, \\
m_{10} &= m_{10} - x, \\
m_{01} &= m_{01} - y, \\
m_{11} &= m_{11} - x * y, \\
m_{20} &= m_{20} - x^2, \\
m_{02} &= m_{02} - y^2.
\end{aligned}
\tag{4.26}
$$

Following this update of the moments, the image centroids, second order central moments, eigenvalues λ_1 and λ_2, and the angle θ are recomputed using Equations (4.2), (4.3), (4.4), (4.8), and (4.9), respectively.

The above segmentation process is then repeated for a number of iterations or until the segmentation converges as indicated by no change in the energy.

4.3.3 Variations to the Segmentation Procedure

Several variations to the shape prior-based fast level set algorithm have been employed in situations which fail to provide a desirable segmentation result using the methods described above. These variations are found to be most useful in the case that objects in the background have an exceptionally negative impact while attempting to segment the target object. The region growing concept described avoids including background object pixels that are not connected to the target image. Similarly, a labeling function allows the user to discriminate against objects with similar intensity or shape characteristics and arrive at the desired result.

Update via Region Growing

The first variation limits the switching of the value of ϕ following each sweep of the image pixels to include only the pixels connected to edge of the current segmentation ϕ as in a region growing implementation. In [61], the smoothness term of the energy in the Chan-Vese model, the first term in Equation (2.19), is approximated by (2.24). The result of (2.24) can only take values of 0, 1, or $\sqrt{2}$, depending on whether the set $\{\phi_{i,j}, \phi_{i+1,j}, \phi_{i,j+1}\}$ belong to the same regions. As mentioned, (2.24) is extended to include the $\phi_{i-1,j}$ and $\phi_{i,j-1}$ terms within the summand. Equation (2.24) is computed prior to each sweep of

the image pixels. The change in energy at each pixel is initially set to a value of 0 and is only computed in the case that the smoothness computed using this extension to (2.24) is greater than 0. This in turn restricts the possibility of ϕ being switched to include only the pixels adjacent to the current segmentation boundary.

4.3.4 Labeling Function

The second variation to the algorithm makes use of a user-specified labeling function L. In the methods presented in [12, 21], L is used to indicate the region of the image in which the shape constraint is to be enforced. In this implementation, L is used to indicate the region of $H(\phi)$ to be used in computing the image moments. This in turn will allow for I_{shape} to be correctly mapped to $H(\phi)$ by excluding regions similar in image intensity to the object to segment. This concept is incorporated into the implementation by replacing Equation (4.1) used to compute the moments of the current segmentation with

$$m_{pq} = \sum_{y=0}^{M-1}\sum_{x=0}^{N-1} x^p y^q L \cdot H(\phi(x,y))dxdy\,, \qquad \textbf{(4.27)}$$

where L is represented as a binary image with values of +1 in the region used to compute the image moments from the current segmentation, and 0 elsewhere.

Results are provided in Chapter 5 to illustrate the ability of the proposed method to segment various objects in the presence of noise and object occlusion. Examples are also illustrated which make use of these variations to the implementation. These are contrasted with the segmentation results obtained using the fast level set algorithm presented in [61].

Chapter 5: Experimental Results

This chapter provides a description of the experimental results obtained through the course of this research. First, the details surrounding the real image data are given. The intermediate results obtained during the construction of an atlas image from the real data are shown and slices of the atlas image are then provided. Finally, the experimental results of using the segmentation methods described in Chapter 3 and 4 are presented for a number of images. The experiments demonstrate the ability of the shape prior segmentation techniques to overcome the complications of image segmentation in the presence of object occlusion, noise, and partial volume effects.

5.1 Registration

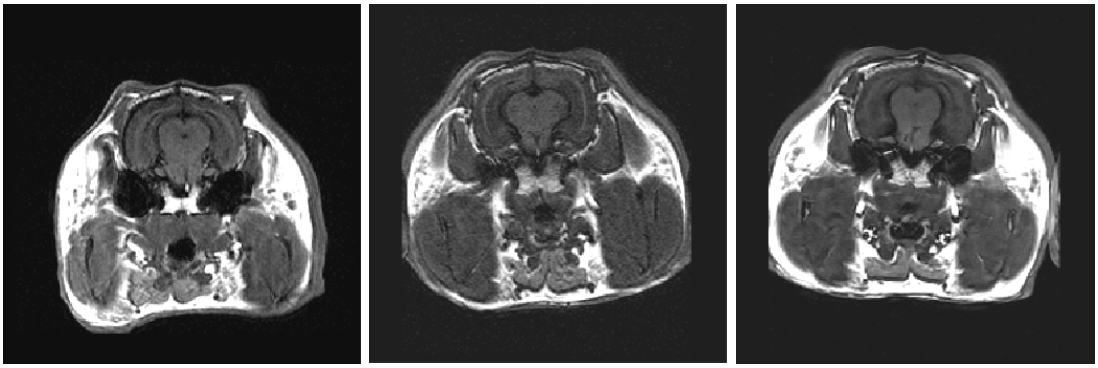
In regards to the real MRI data, results of the registration process used in constructing the atlas and in aligning the atlas to a novel image are provided in this section. The intermediate results obtained during the construction of an atlas are shown and slices of the resulting atlas image are then provided. In registering the atlas to novel image data, results are shown to illustrate the methods ability to align the atlas mask to the brain volume of the new image to provide a close initialization for a final segmentation step.

Real Image Data

The 3D fast spoiled gradient echo (FSPGR) images of brain data were acquired from rabbit subjects by Susan Lemieux, Ph.D., and Bernard Schreurs, Ph.D., at West Virginia University. This was accomplished using a General Electric 3T Signa Excite MR scanner and a Nova Medical 12 cm Quadrature Coil. The parameters include TR = 10.4, TE = 2.3 ms, and FOV = 80x60 mm. The resulting images are each composed of a 16 bit grayscale matrix = 256x256x72 with a slice thickness = 0.8mm.

5.1.1 Atlas Construction

Figure 5.1 illustrates a slice of the 3D data set, $I_{TrainingReference}$, used as the reference image in the registration of the remaining training samples $I_{Training}(k)$, for k = 1,2,…11. A few corresponding slices of $I_{Training}(k)$ are also pictured to provide insight into the variation that exists between subjects.



(a) **(b)** **(c)**

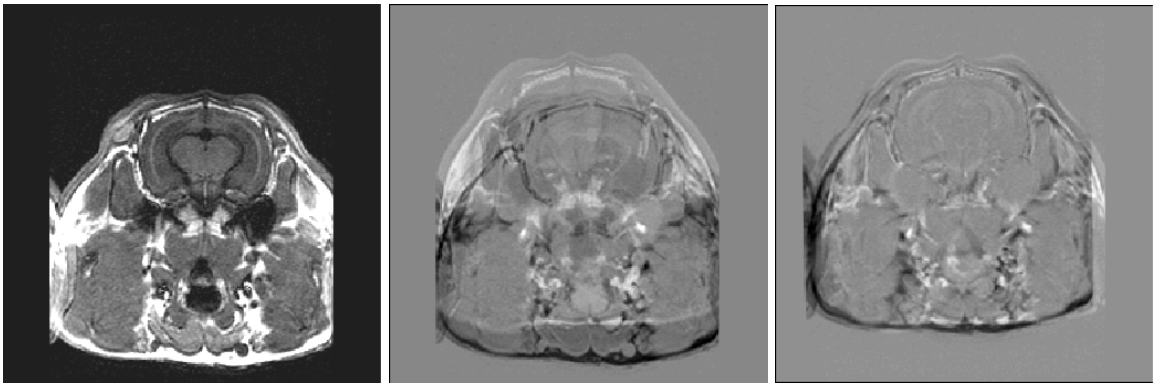
Figure 5.1 Various slices of the training data used to construct the atlas image I_{atlas}. (a) A slice of the reference training data volume, $I_{TrainingReference}$. (b) and (c) the corresponding slices from the training data, $I_{Training}$.

Table 5.1 provides implementation details, in terms of the parameters, used in performing the 3D registration of the training data $I_{Training}$ to the training data reference image $I_{TrainingReference}$. Figure 5.2 illustrates an example of the results obtained through this registration process, including the initial difference image as well as the final transformed result. Figure 5.2(b) shows the initial misalignment between the two images through the difference image produced by subtracting $I_{TrainingReference}$ by $I_{Training}$. The difference image in Figure 5.2(c) illustrates the aligned images provided by the registration process.

Table 5.1. Parameters of the filters involved in the registration process used in registering the training data.

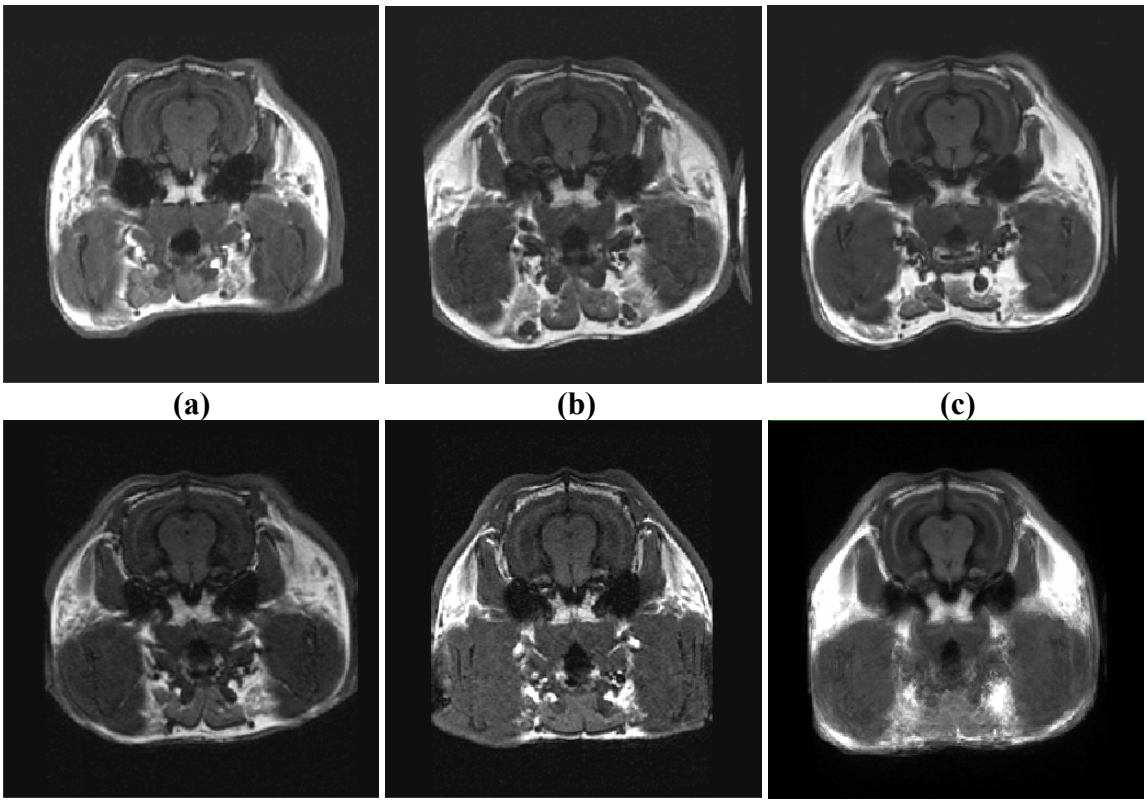
Filter	Parameter	Translation Registration	Affine Registration
Mattes MI Metric	**Number of Histogram Bins**	64	64
	Number of Spatial Samples	20000	20000
Powell Optimizer	**Step Length**	0.01	0.01
	Step Tolerance	0.001	0.0005
	Value Tolerance	0.001	0.0005
	Max Iterations	100	100
	Translation Scaling	0.01	0.01
	Matrix Scaling	NA	10

Upon registering each training data set $I_{Training}(k)$ to $I_{TrainingReference}$, the atlas image I_{atlas} is computed as the median intensity value at each pixel location of these training sets. Figure 5.3(a)-(e) illustrate corresponding slices found within the aligned training data $I'_{Training}$. Figure 5.3(f) shows the resulting slice within I_{atlas}, which can be seen to provide an average representation of the data with the same resolution, particularly in the brain region.



(a) **(b)** **(c)**

Figure 5.2 Results of registering $I_{Training}$ to $I_{TrainingReference}$. (a) A slice of the training data, $I_{Training}$. (b) The difference image between corresponding slices in $I_{TrainingReference}$ and the unregistered $I_{Training}$. (c) The final difference image between corresponding slices in $I_{TrainingReference}$ and the registered $I_{Training}$.

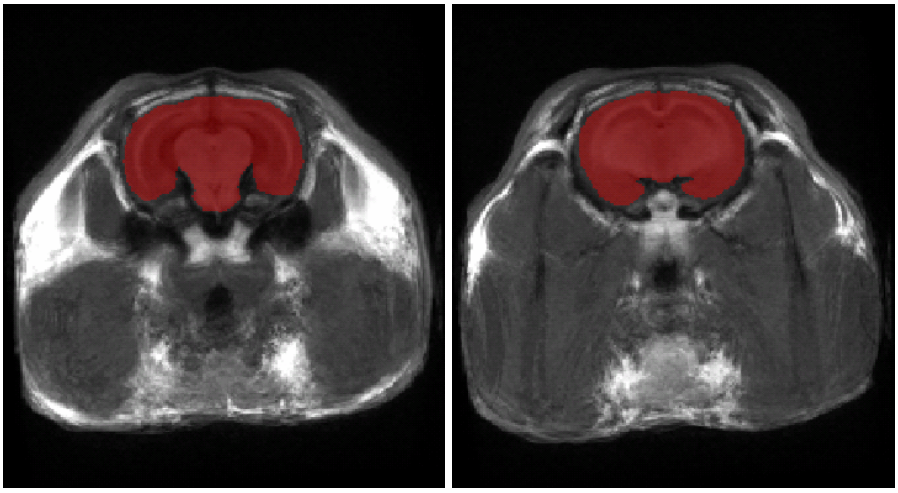


(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 5.3 Computing I_{atlas} (a)-(e) Several slices of $I'_{Training}$ resulting from the registration step. (f) The corresponding slices of I_{atlas} computed as the median of these registered training samples.

Following the computation of I_{atlas}, the atlas binary mask was formed by manually segmenting each slice of I_{atlas}. Figure 5.4 illustrates slices of I_{atlas} with the binary manual segmentation result superimposed over the brain region.



(a) **(b)**

Figure 5.4 The result of the manual segmentation of I_{atlas}. (a) and (b) Slices of I_{atlas} with the respective manual segmentation results indicated (red).

5.1.2 Novel Image Registration

In the initial steps involved in extracting the brain volume from a novel data set, the 3D atlas I_{atlas} is registered to the novel data set I. The parameters of the filters used in the ITK implementation of this registration step are the same as in constructing I_{atlas}, which are provided in Table 5.1. The transformation T_{affine} found in this registration step is used to map the signed distance of the manually segmented atlas ϕ_{atlas} onto the corresponding brain region of I. Figure 5.5(a)-(c) illustrates slices of I_{atlas} with the manual segmentation overlay. Figure 5.5(d)-(f) shows the transformed binary $\phi_0 < 0$ over slices of I resulting from T_{affine} found in the registration of I_{atlas} to I. This illustrates the ability of the method to accurately register the two data volumes to provide a close initialization for the final segmentation step. The transformed result ϕ_0 appears to be well aligned with the brain region of I. This signed distance map is then used as the initial embedding function ϕ_0 for the level set segmentation step.

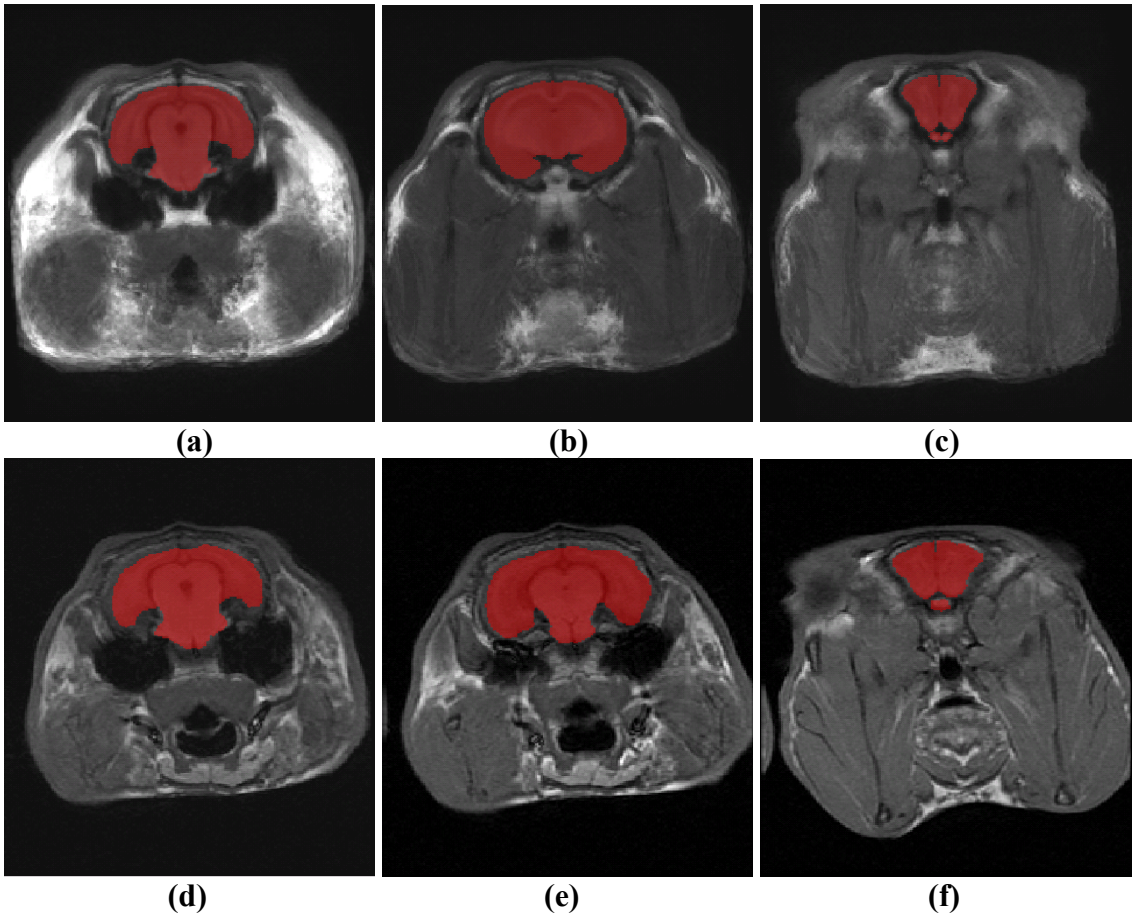


Figure 5.5 Results of the registration of I_{atlas} to I. (a)-(c) Slices of I_{atlas} with the respective manual segmentation. (d)-(f) The corresponding slices of a novel image I with the transformed signed distance ϕ_0, where $\phi_0 < 0$, which provides the initial embedding function of the segmentation step.

5.2 Geodesic Active Contour Level Set Segmentation Method

To illustrate the use of the proposed speed function, the new speed function described in Chapter 3 was first applied to synthetic images and then to the real MRI rabbit brain data. This was accomplished by replacing the speed map g described in [10] with the proposed speed map to be used in an implementation of the geodesic active contours method.

5.2.1 Synthetic Data

A 2D synthetic image of size 128 x 128 with 256 gray levels was generated by taking an arbitrary shape and then blurring the edges in various locations around the perimeter. For this experiment, the image without the blurred edges serves as I_{atlas}. This synthetic image is illustrated in Figure 5.6, along with the intermediate steps involved in determining the

proposed speed map. This new speed map, shown in Figure 5.6(f) provides information of the edges found in the atlas image when edge information is missing.

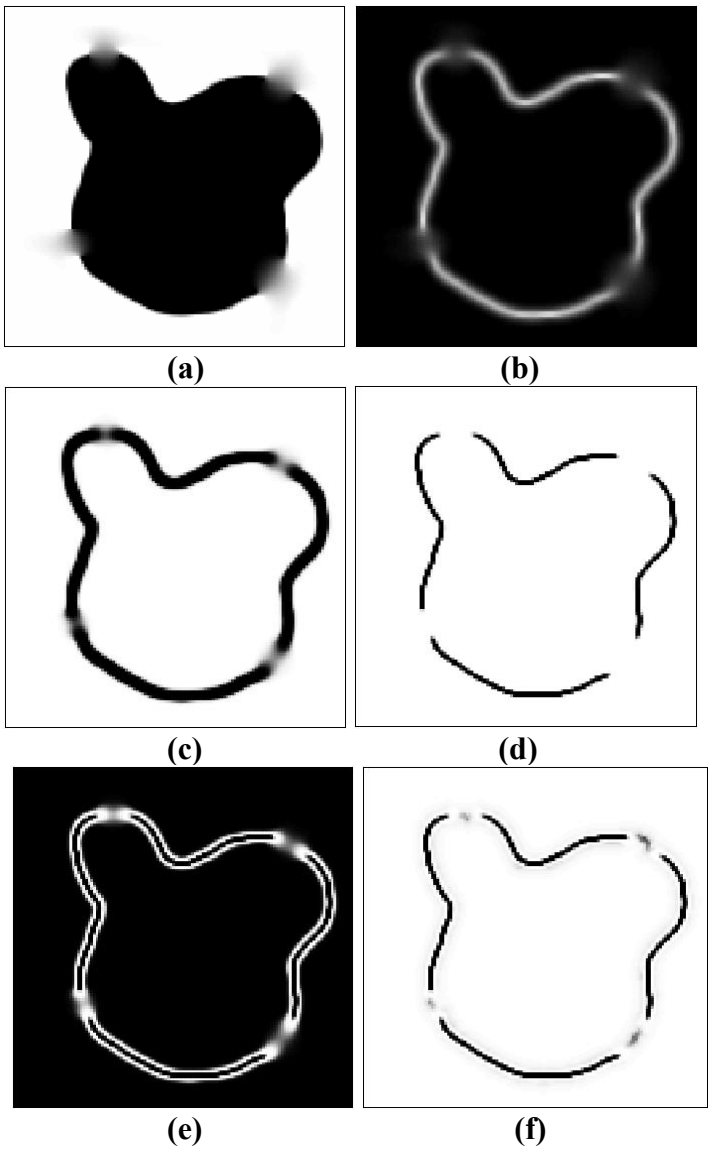


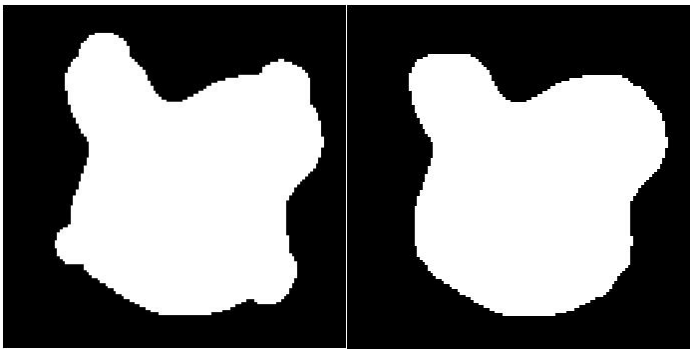
Figure 5.6 The intermediate steps involved in computing the proposed speed term. (a) The synthetic image *I*. (b) The gradient magnitude image. (c) The g_{Low} speed image. (d) The g_{High} speed image. (e) The difference $g_{High} - g_{Low}$. (f) The new speed map *g*.

Segmentation results were obtained for the synthetic image shown in Figure 5.6 using both the traditional speed function in (2.17) as well as the proposed difference speed function. This was accomplished using the geodesic level set implementation available in ITK, as described in section 3.4. The initial embedding function ϕ_0 was initialized as a signed distance function of a circle located in the center of the shape.

In using the proposed speed function, the speed image was simply replaced by the image generated using the methods in section 3.4.2. The g_{High} speed term was used as the speed term computed using (2.17) to produce the results of using the traditional approach. No parameters involved in the level set evolution equation were altered when comparing the use of the traditional and proposed speed maps. Table 5.2 provides the parameter values used to obtain the results illustrated in Figure 5.7. These results show that the evolving curve passes through regions of missing edge information when a traditional speed map is used. However, a more accurate result is obtained with the proposed speed function.

Table 5.2 Parameters used in the ITK segmentation of the synthetic data.

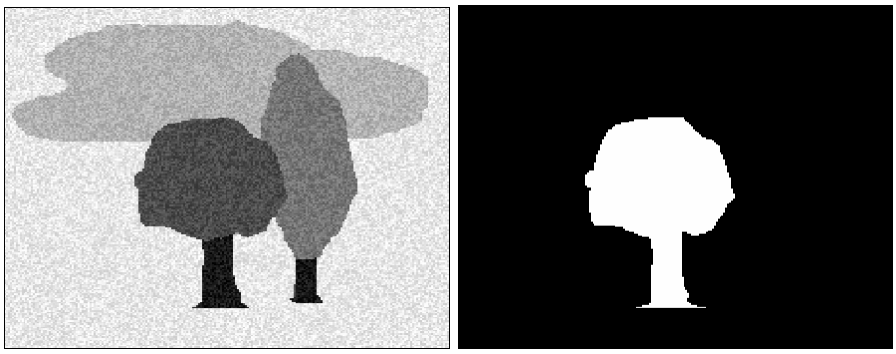
Smoothing Filter	**Sigmoid Filter**				**Geodesic Active Contours Filter**		
σ	**α_{Low}**	**β_{Low}**	**α_{High}**	**β_{High}**	**Advection**	**Curvature**	**Propagation**
1.0	-4.0	25.0	-6.0	60.0	0.9	0.4	0.2



(a) **(b)**

Figure 5.7. Segmentation results of the synthetic image using: (a) The speed map computed using (2.17). (b) The proposed speed map computed using the methods presented in section 3.4.

This concept was also tested using another synthetic image in which the target object is surrounded by background objects of similar intensity characteristics. The 2D synthetic image I of size 207 x 229 contains 256 gray levels and an added Gaussian noise of 25% variance. This image, along with the shape image I_{shape}, is illustrated in Figure 5.8. Results were obtained using an implementation of geodesic active contours written with MATLAB 7.1.



(a) **(b)**

Figure 5.8 The input image data including: (a) The synthetic image I. (b) The shape image I_{shape}.

Segmentation results were first obtained using two different speed images both computed using the traditional speed function in Equation (2.17). In the first example, the parameters of (2.17) are set to produce a speed map which should prevent the evolving curve from passing through the object boundary into the surrounding regions. The gradient magnitude image, the speed map, the initial ϕ, and the segmentation result are found in Figure 5.9. Table 5.3 provides the parameter values used to obtain the segmentation result.

In a second example using the traditional speed function, the parameters of (2.17) are set to allow for the evolving curve to proceed through the entire object without permitting the evolving curve to leak into the surrounding objects. Figure 5.10 illustrates the speed map and final segmentation result. The gradient magnitude image and initialization are the same as in the previous example, and are pictured in Figure 5.9 (a) and (c). Table 5.4 provides the parameter values used during the segmentation of this image.

Table 5.3 Parameters used in the ITK segmentation of the synthetic data shown in Figure 5.9 using the traditional speed function.

Smoothing Filter	**Sigmoid Filter**		**Geodesic Active Contours Filter**		
σ	**α**	**β**	**Advection**	**Curvature**	**Propagation**
0.6	-1.0	6.0	1.0	0.75	0.75

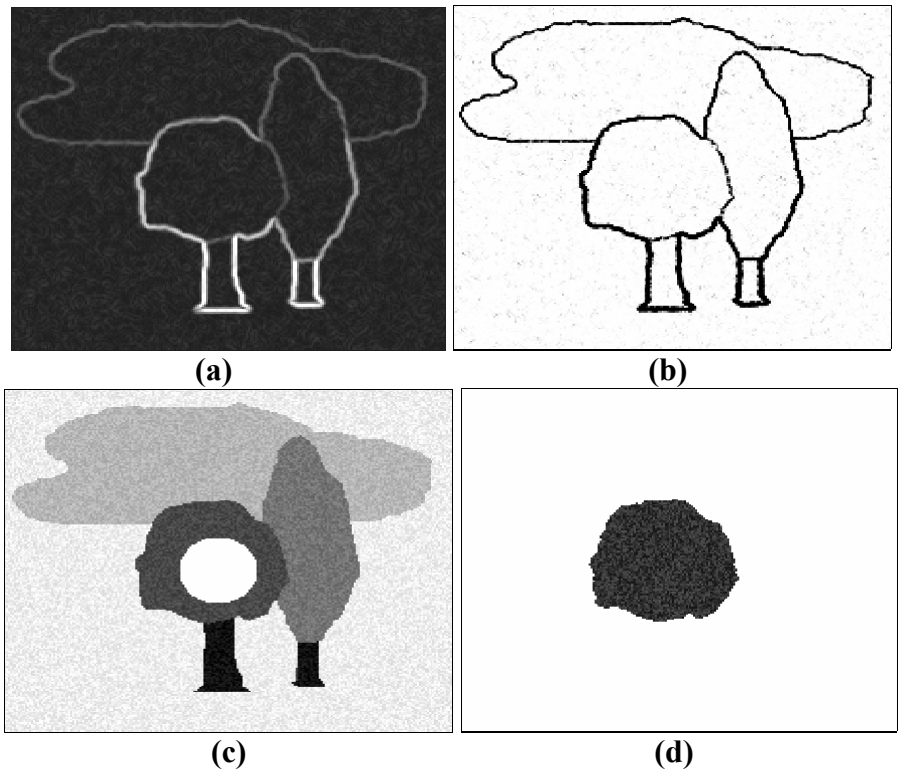


Figure 5.9. Intermediate steps involved in the segmentation as well as the result of using the traditional speed function found in (2.17). (a) The gradient magnitude image. (b) The speed image. (c) The initialization, $\phi_0 < 0$. (d) The segmentation result.

Table 5.4 Parameters used in the ITK segmentation of the synthetic data shown in Figure 5.10 using the traditional speed function.

Smoothing Filter	Sigmoid Filter		Geodesic Active Contours Filter		
σ	α	B	Advection	Curvature	Propagation
0.6	-1.0	12.0	1.0	0.75	0.75

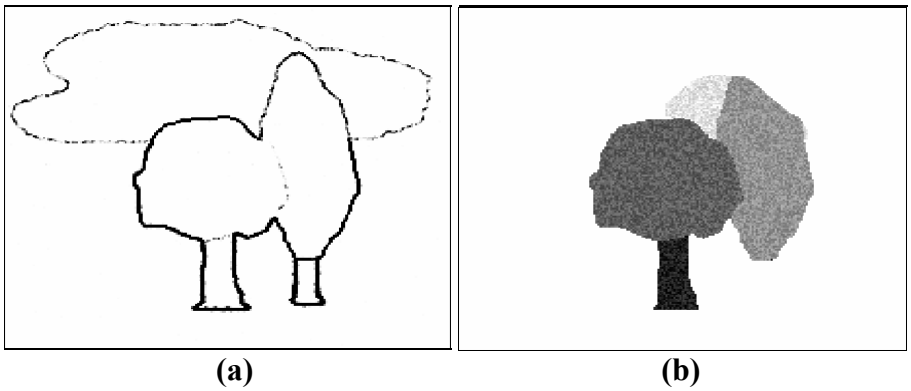


Figure 5.10. Segmentation result using a second example of the traditional speed function found in (2.17). (a) The speed image. (b) The segmentation result.

From the results shown in Figure 5.9, it becomes apparent that by adjusting the parameters of the speed function to segment the object from the background objects, the speed map does not allow for the entire image to be segmented, given the initialization ϕ_0. The results shown in Figure 5.10 further illustrate the difficulty in choosing the parameters that would allow for an accurate segmentation. Despite these complications, using the proposed speed function described in section 3.4.2 provides a more useful speed map. Figure 5.11 illustrates the intermediate steps used in computing the proposed speed map. The segmentation result illustrated in Figure 5.11(d) is obtained using the proposed speed function in place of the traditional function (2.17). Table 5.5 provides the parameter values used in the implementation.

Table 5.5 Parameters used in the ITK segmentation of the synthetic data using the new speed function.

Smoothing Filter	**Sigmoid Filter**				**Geodesic Active Contours Filter**		
σ	**α_{Low}**	**β_{Low}**	**α_{High}**	**β_{High}**	**Advection**	**Curvature**	**Propagation**
0.6	-1.0	12.0	-1.0	16.0	1.0	0.75	0.85

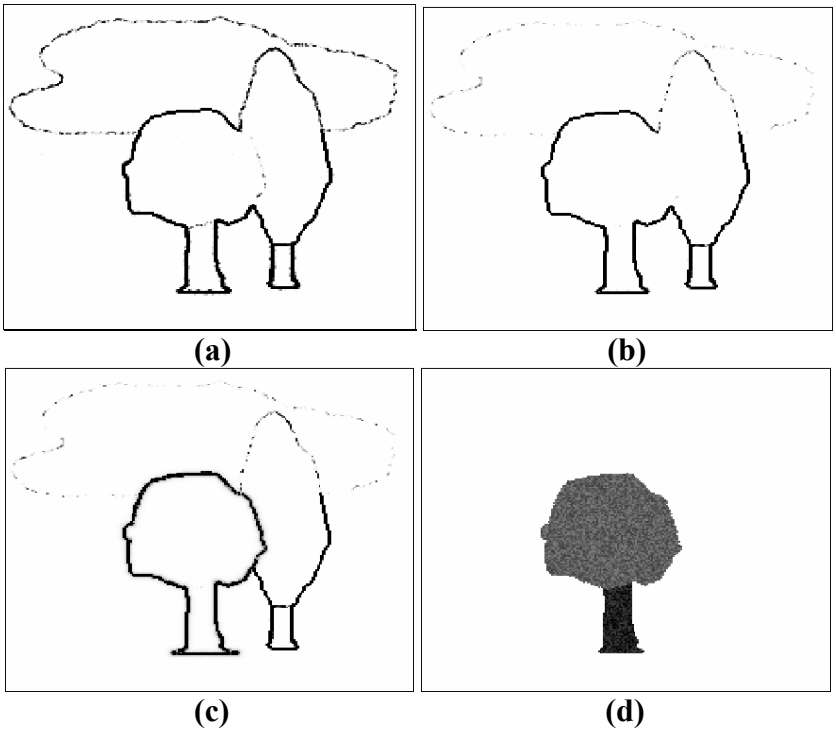
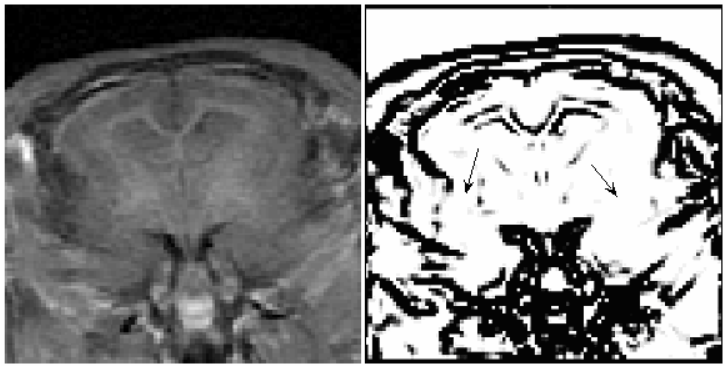


Figure 5.11 The segmentation result obtained using the proposed speed function and initialization ϕ_0 shown in Figure 5.9 (c). (a) The g_{Low} speed image. (b) The g_{High} speed image. (c) The new speed map g. (d) The segmentation result obtained using the parameters provided in Table 5.5.

5.2.2 Real Image Data

As mentioned, the real image data contains regions of no discernable edges, as indicated by arrows in Figure 5.12(b), which creates difficulties in obtaining an accurate segmentation result. Therefore, the speed function proposed in section 3.4.2 was used in place of Equation (2.17), which is applied to produce the traditional speed map used in level set segmentation techniques. The intermediate steps of computing this speed map for a slice of the real MRI data are illustrated in Figure 5.13.



(a) **(b)**

Figure 5.12. Input data used in the implementation of geodesic active contours segmentation. (a) The original image. (b) The speed map image of MRI rabbit brain data computed using the Sigmoid Filter provided by ITK with $\beta = 35$ and $\alpha = -5$. Edge information is missing as indicated by arrows.

As with the synthetic data, segmentation results were obtained for the real 2D MRI data using both the traditional speed map computed with Equation (2.17) as well as the proposed speed map obtained as described in section 3.4.2. The segmentation was then performed using the geodesic level set implementation available in ITK as described in Chapter 3. Table 5.6 provides the parameter values used to obtain the results shown in Figure 5.14. Figure 5.14(a) illustrates the traditional speed map's inability to prevent the contour from passing through the boundary of the brain. This occurs in the regions of missing edge information. Figure 5.14(b) shows an improved segmentation result obtained using the proposed speed map. The edge information provided by the atlas is used in locations where edge information provided by the image gradient is not found.

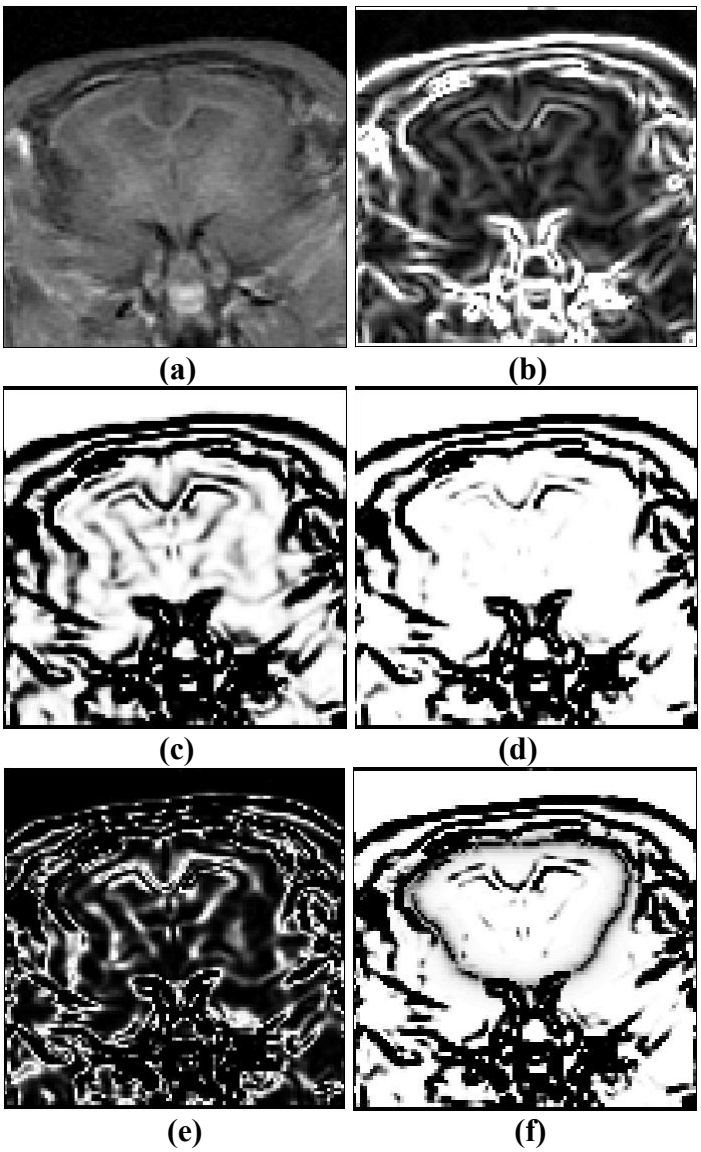
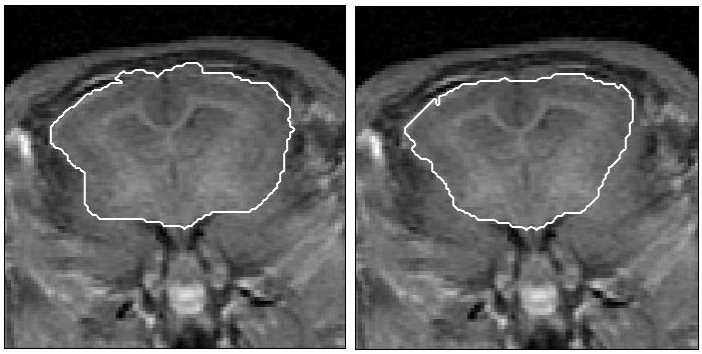


Figure 5.13. Input data used in the implementation of geodesic active contours segmentation. (a) A slice of the Rabbit brain MRI data I. (b) The gradient magnitude image. (c) The g_{Low} speed image. (d) The g_{High} speed image. (e) The difference $g_{High} - g_{Low}$. (f) The proposed speed map g.

Table 5.6. Parameters used to obtain the results shown in Figure 5.14 from the ITK segmentation of the real data shown in Figure 5.12.

Smoothing Filter	Sigmoid Filter				Geodesic Active Contours Filter		
σ	α_{Low}	β_{Low}	α_{High}	β_{High}	Advection	Curvature	Propagation
1.5	-7.0	30.0	-7.0	42.5	0.75	2.0	0.25



(a) **(b)**

Figure 5.14. Segmentation of MRI data using: (a) A typical speed map. (b) The proposed speed map.

5.3 Fast Level Set Segmentation using Shape Constraints

The fast level set shape prior segmentation methods proposed in Chapter 4 were implemented using MATLAB 7.1. This implementation was applied to several 2D images to illustrate the ability of this method to quickly segment an object from the background in the presence of occlusions and noise. The results are contrasted with results obtained using an implementation of the method presented in [62] to show that the incorporation of the shape prior provides a more accurate and robust solution.

5.3.1 Synthetic Data

A 150 x 175 pixel size synthetic image was generated in order to illustrate the ability of the method to segment an object that is occluded by another object. The first synthetic image consisting of 256 gray levels with added Gaussian noise of 10% variance is illustrated in Figure 5.15. ϕ was randomly initialized at each pixel of the image. Figure 5.16 illustrates the result of segmenting the image I using the fast level set implementation of [61] without the shape constraint. The result, which was obtained in 23 sweeps of the segmentation algorithm in 11.05 seconds, shows that the approach presented in [61] is unable to segment the hand object from all surrounding objects.

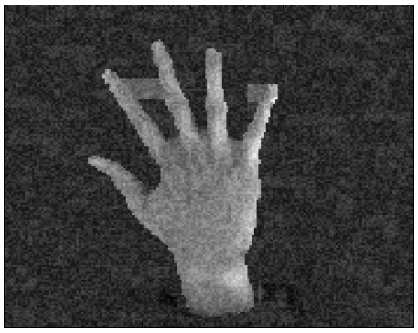
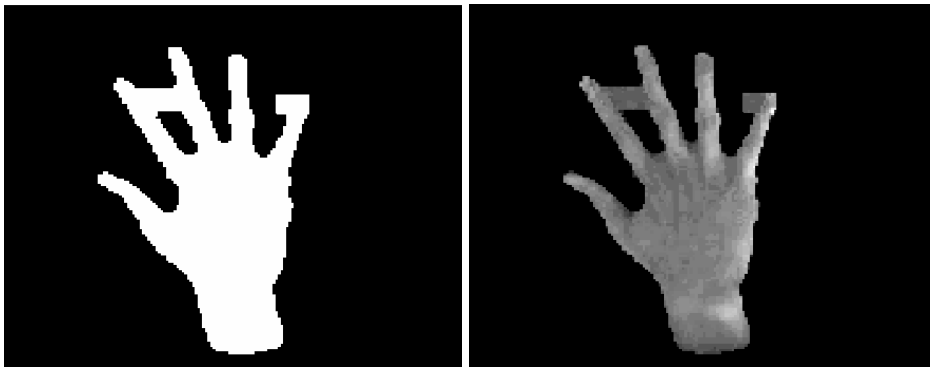


Figure 5.15. The synthetic hand image *I*.



(a) **(b)**

Figure 5.16. The segmentation result using an implementation [61]. (a) The binary segmentation. (b) Final segmentation result in which the hand is not entirely segmented from the background.

Figures 5.17 and 5.18 illustrate the segmentation result using the proposed method described in Chapter 4 with two different shape priors. The incorporation of these shape priors, represented as the binary images illustrated in 5.17(b) and 5.18(b), allows for a more accurate segmentation of the hand object from the surrounding background. The parameters used to obtain these results are shown in Table 5.7.

Table 5.7. Parameters used in the segmentation of the hand image using the shape constraint.

Result Figure	Size	σ_1^2	σ_2^2	λ	% Min. Pixel Switched	Iterations (before shape constraint)	Total Iterations	Computational Time (secs)
5.17, 5.18	150x175	1.50	1.00	2.0	75	23	30	4.32

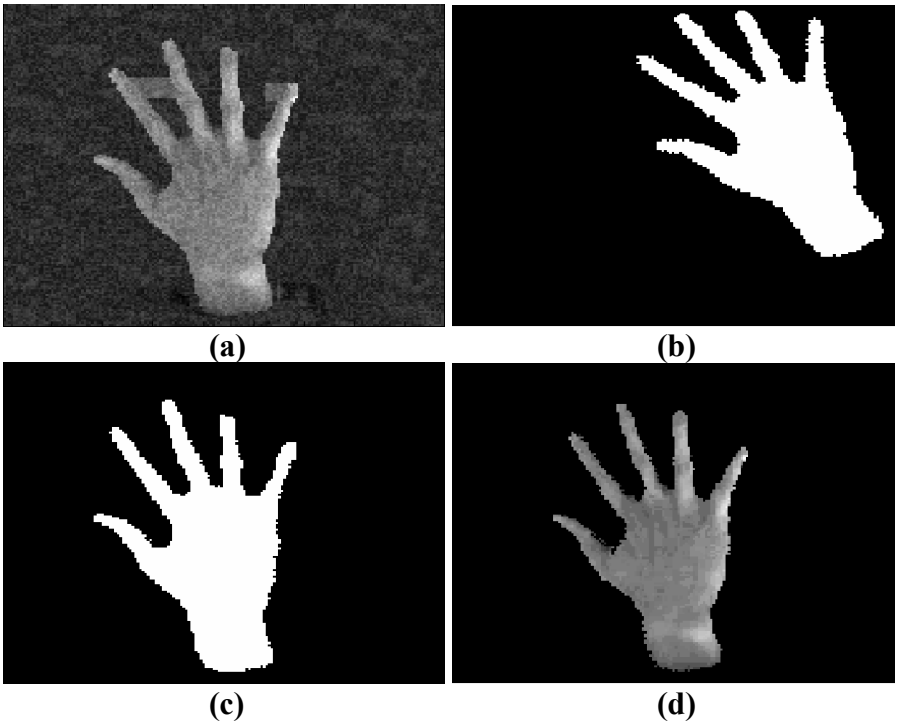


Figure 5.17. The image segmented using the method proposed in Chapter 4. (a) The original image *I*. (b) The binary image shape representation I_{shape}. (c) The binary result $\phi > 0$. (d) The segmented image *I*.

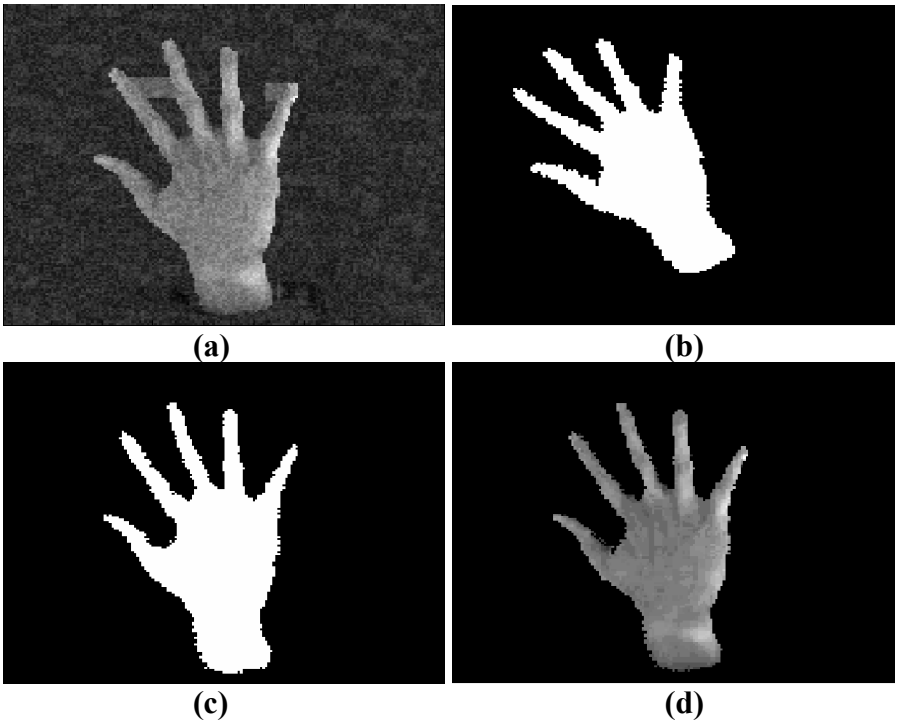
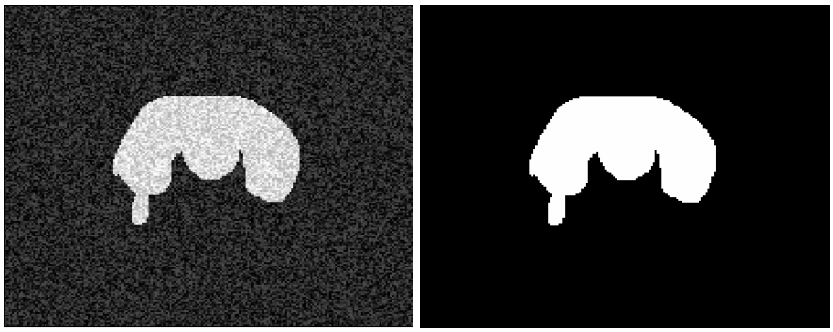


Figure 5.18. The segmentation result using the proposed method. (a) The original image *I*. (b) The binary image shape representation I_{shape}. (c) The binary segmentation result $\phi > 0$. (d) The segmented image.

A second synthetic image was generated to mimic the low-contrast between contiguous regions of the real rabbit brain data. This image, of size 186 x 195 pixels, consists of 256 gray levels with added Gaussian noise with 25% variance, and is illustrated in Figure 5.19(a). A projection is attached to the brain shape which does not adhere to a typical brain shape, an example of which can be seen in Figure 5.20 (a). Figure 5.19(b) shows the segmentation result obtained from 10 sweeps of the fast level set algorithm of [61]. These results show that this method provides no shape constraints to avoid such a result.



(a) **(b)**

Figure 5.19. The image segmentation using an implementation of [61]. (a). The synthetic brain shape image I. (b). The binary segmentation result $\phi > 0$.

Figure 5.20 illustrates the segmentation results using the fast level set shape prior segmentation implementation which make use of the shape image I_{shape} found in Figure 5.20(a). The results were obtained using the parameters indicated in Table 5.8. The results indicate that the proposed method has the ability to segment the brain shape object from such a projection in order to obtain a result which exhibits an acceptable shape. Figure 5.20(d) shows the difference in segmentation results provided by the original method [61] and the proposed method including a shape constraint. The most significant difference appears in the location of the region projecting from the brain shape.

Table 5.8. Parameters used in the segmentation of the data pertaining to the segmentation of the image shown in Figure 5.19(a) to provide the results shown in Figure 5.20.

Result Figure	Size	σ_1^2	σ_2^2	λ	% Min. Pixel Switched	Iterations (before shape constraint)	Total Iterations	Computational Time (secs)
5.20	186x195	1.00	1.00	3.0	75	10	12	5.34

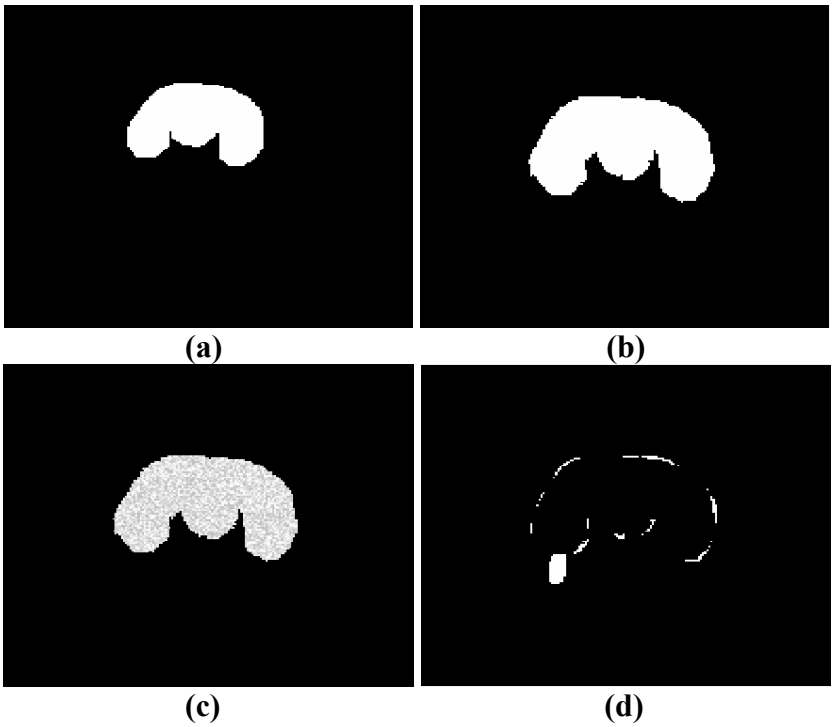


Figure 5.20. The image segmentation of the synthetic image using the proposed method. (a). The binary image shape representation I_{shape}. (b). The binary segmentation result $\phi > 0$. (c). The segmented image. (d). The difference image computed between the segmentation result without the shape term (Figure 5.19 (b)) and the segmentation result using the shape term (Figure 5.20 (b)).

This concept was also tested using the synthetic image of the tree object found in section 5.3.1. As mentioned, the synthetic image I of size 207 x 229 contains 256 gray levels and an added Gaussian noise of 25% variance. This image, along with the shape image I_{shape}, is illustrated in Figure 5.8 of section 5.3.1. Segmentation results were first obtained using the fast level set algorithm implementation of [61] without the shape constraint term. Using different values of σ_1^2 and σ_2^2, different regions of the image may be segmented as shown in Figure 5.21. The parameters used to obtain these results are found in Table 5.9.

Table 5.9. Parameters used in the segmentation of the synthetic image shown in Figure 5.21.

Result Figure	Size	σ_1^2	σ_2^2	λ	% Min. Pixel Switched	# Iterations	Computational Time (secs)
5.21 b.	209x229	2.75	1.00	0.00	75	24	1.02
5.21 c.	209x229	1.00	1.00	0.00	75	20	0.99
5.21 d.	209x229	0.35	1.00	0.00	95	34	1.69

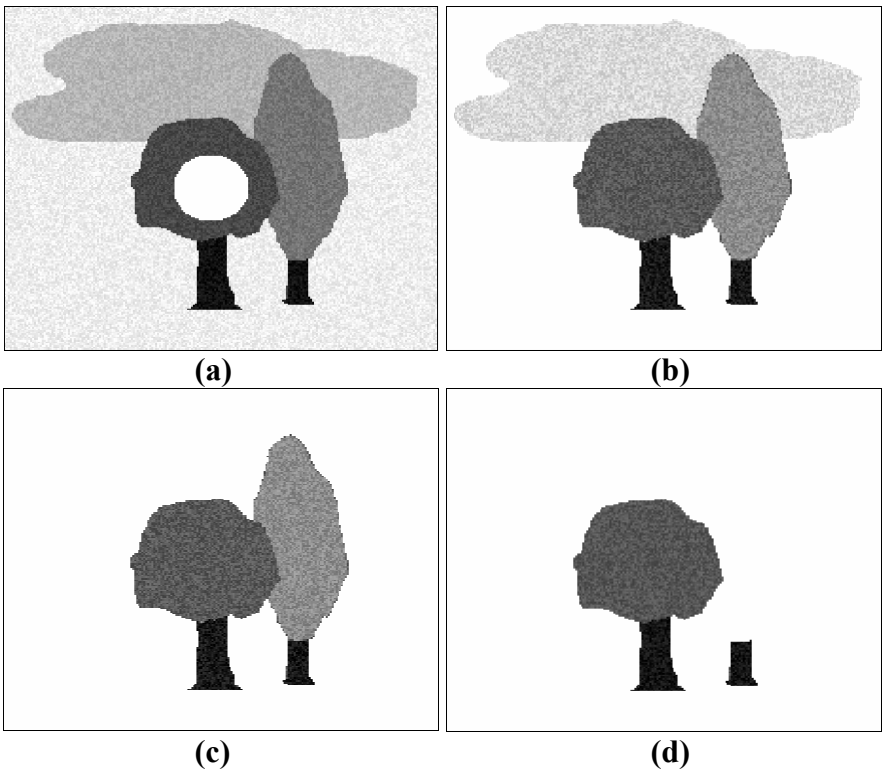


Figure 5.21. Segmentation results without the shape constraint. (a) The original image I with the initial segmentation $\phi_0 > 0$ indicated as the white circle. (b)-(d) The segmentation results using different values of σ_1^2 and σ_2^2 provided in Table 5.9 above.

The two variations described in section 4.3.3 were imposed on the fast level set algorithm used to segment the object while imposing the shape constraint. The first variation includes limiting the switching of the value of the segmentation to only pixels connected to the current segmentation as in a region growing implementation. Table 5.10 provides details of the parameters and computation time involved. Results are illustrated in Figure 5.22.

Table 5.10. Parameters used in the segmentation of the data pertaining to Figure 5.22.

Result Figure	Size	σ_1^2	σ_2^2	λ	% Min. Pixel Switched	# Iterations (before shape constraint)	# Total Iterations	Computational Time (secs)
5.22	209x229	1.00	1.00	2.0	75	65	70	34.14

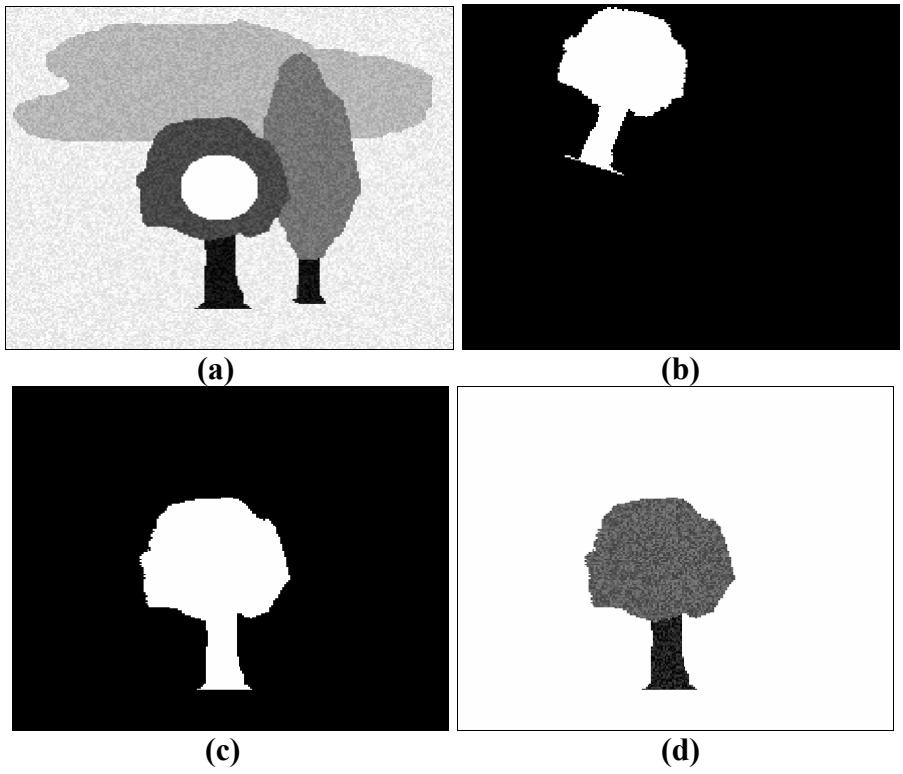


Figure 5.22. Segmentation results using the shape constraint and the region growing approach. (a) The original image I with the initial segmentation $\phi_0 > 0$ indicated as the white circle. (b) The shape image (c) The binary segmentation result $\phi > 0$. (d) The final segmentation result.

The second variation to the algorithm makes use of the labeling function L as described in 4.3.3. The initialization of ϕ was accomplished by randomly assigning values of 1 and -1 throughout the image space. Table 5.11 provides details of the parameters and computation time involved and the results are illustrated in Figure 5.23. The results shown in Figures 5.22 and 5.23 illustrate the ability of the proposed method to segment the tree object using the variations presented in section 4.3.3. As shown in Figure 5.21, difficulties exist in segmenting this object which do not allow for the method presented in [61] to obtain such a result.

Table 5.11. Parameters used in the segmentation of the data pertaining to Figure 5.23.

Result Figure	Size	σ_1^2	σ_2^2	λ	% Min. Pixel Switched	Iterations (before shape constraint)	Total Iterations	Computational Time (secs)
5.23	207x229	0.35	1.00	2.0	95	34	42	7.26

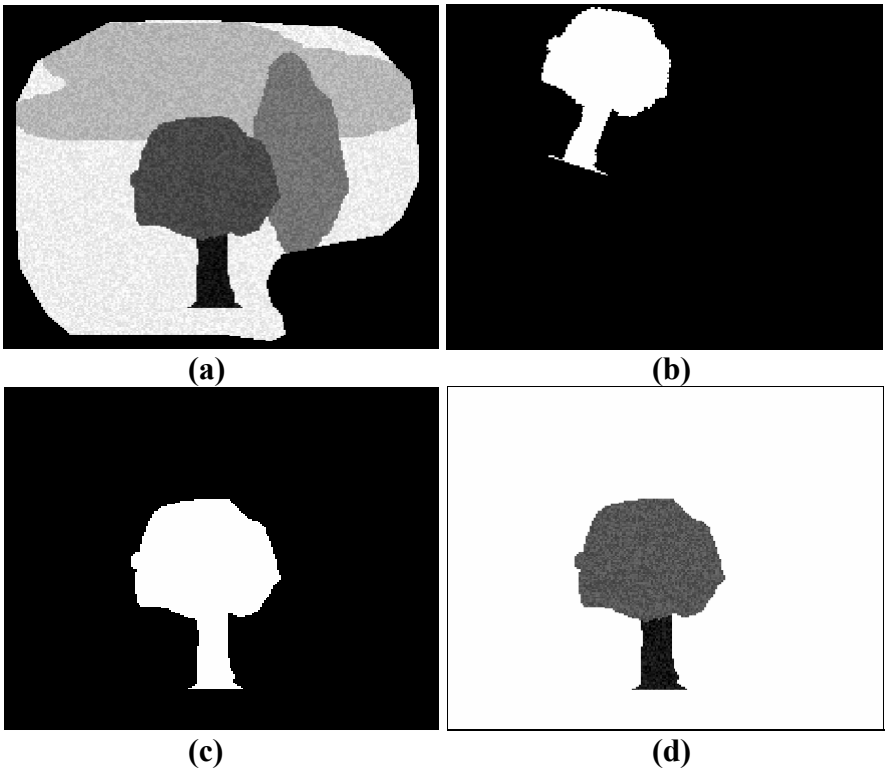
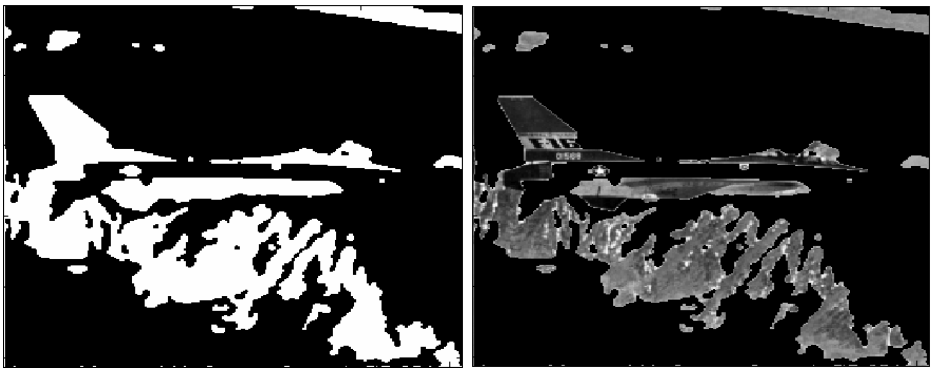


Figure 5.23. Segmentation results using the shape constraint along with the labeling function. (a) The region of the image to be considered in the shape comparison as indicated by L. (b) The shape image (c) The binary segmentation result $\phi > 0$. (d) The final segmentation result.

5.3.2 Real Data

The proposed fast level set segmentation method was also applied to a 256 x 256 sized 8 bit real image shown in Figure 5.24. In segmenting this aircraft image, ϕ was randomly initialized at each pixel of the image. Figure 5.25 illustrates the result of segmenting the image I using the fast level set implementation of [61] without the shape constraint. The result, which was obtained in 15 sweeps of the segmentation algorithm in 18.90 seconds, shows that this approach presented in [61] is unable to segment the aircraft from the background, which contains regions similar in intensity characteristics. Here, regions of the background are included in the segmentation while regions of the aircraft are missing. The parameters used, excluding the shape constraint, are the same as those indicated in Table 5.12.

Figure 5.24. A real data aircraft image *I*.



(a) **(b)**

Figure 5.25. The segmentation result using an implementation [61]. (a) The binary segmentation. (b) Final segmentation result.

Figures 5.26 illustrates the segmentation result using the proposed method described in Chapter 4 with the use of the labeling function *L* as described in 4.3.3. The labeling function is necessary in excluding a portion of the darker regions of the background in order for the shape image to be accurately transformed to the image object. The incorporation of these shape priors, represented as the binary image illustrated in 5.26(b), allows for the entire segmentation of the aircraft from the surrounding background. The parameters used to obtain these results are shown in Table 5.12.

Table 5.12. Parameters used in the segmentation of the real image data shown in Figure 5.25 to obtain the results shown in Figure 5.26.

Result Figure	Size	σ_1^2	σ_2^2	λ	% Min. Pixel Switched	Iterations (before shape constraint)	Total Iterations	Computational Time (secs)
5.26	256x256	1.00	1.00	5.0	25	5	15	20.06

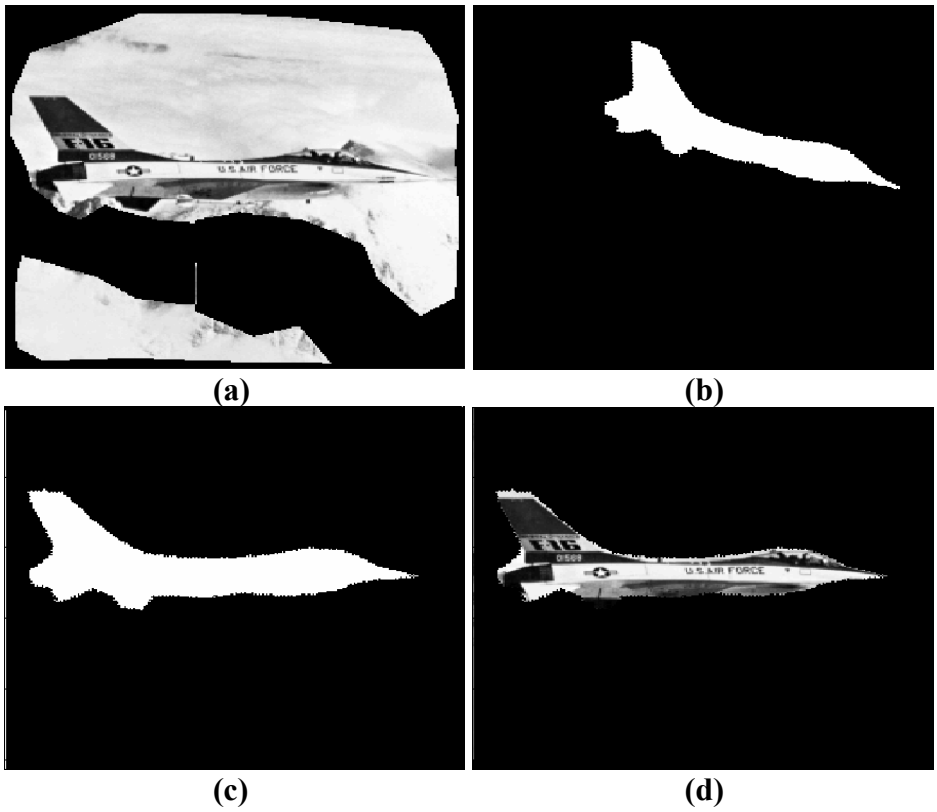


Figure 5.26. The image segmented using the method proposed in Chapter 4. (a) The original image I in the regions to be used in the shape constraint as indicated by L. (b) The binary image shape representation I_{shape}. (c) The binary result $\phi > 0$. (d) The segmented image I.

Chapter 6: Concluding Remarks and Future Work

Level set techniques such as geodesic active contours [10] have proven an effective solution to the segmentation problem of images with well defined boundaries. In implementing such segmentation methods, a crucial step in arriving at a desirable segmentation is the computation of the speed map to be used in the evolution equation. The speed map, typically derived from the gradient information of the image, is designed to allow an evolving curve to move freely in regions of homogeneity but slow the curve to a stop at edges. However, boundary information may become distorted due to occlusions, noise, or partial volume effects. The speed map computed from this image may not have the ability to slow the evolving curve in these low-contrast regions, thus allowing the curve to pass beyond the boundary and into other regions of the image. The accuracy of the segmentation method is compromised as a result. Assuming that an atlas or shape prior is available, this information could be incorporated into the segmentation framework in order to avoid such occurrences.

A New Speed Function for Shape Prior Segmentation

A new speed function has been proposed that combines edge and atlas information to compute the speed map in atlas-assisted level set segmentation. Subtracting two traditional speed terms, which differ only in the value of the constants in Equation (2.17), separates regions of homogeneity and strong edges from regions of intermediate edge information. The proposed speed function is formulated to rely on atlas information when the edge information in unavailable. This concept was incorporated into an implementation of geodesic active contour segmentation by simply replacing the traditional speed term computed using (2.17) with the proposed speed term.

Results of the segmentation of synthetic and real MRI data illustrate the ability of the new speed map to prevent the ‘leaking’ of the evolving curve in regions of the image containing missing edge information which would otherwise occur. This leads to a more accurate and robust segmentation than a typical speed map may provide, without a significant increase in computation time.

Possible future work for the new speed function concept includes:

- Applying the proposed speed function to 3D data.
- Determining a method to automatically determine the parameter values, α and β, of Equation (2.17).
- Investigating its potential in graph cut segmentation methods.
- Casting this concept into a Bayesian Framework.

In another level set segmentation model [11], Chan and Vese propose a technique which partitions an image based on the region properties such as intensity mean, rather than including a speed term based on the gradient of the image. This allows for the detection of objects even in the presence of smooth or discontinuous boundaries. However, as in the case of the geometric active contours, the Chan-Vese model fails to provide an accurate segmentation in regions of high noise or when the target object is occluded by other objects. Such occurrences are often unavoidable in practical applications. Therefore, prior shape information may once again be incorporated into the segmentation method in order to provide a more accurate and robust solution.

Fast Level Set Segmentation using a Shape Prior

A method is proposed which embeds a shape prior into region-based active contour segmentation models through an additional energy term which minimizes a function of the distance between the evolving active contour and the shape prior. This shape term therefore constrains the evolving curve to resemble a target shape. This allows for a more accurate segmentation despite noise and occlusions when information regarding the shape of the target object is available. This concept is implemented by extending the fast algorithm applied to the Chan-Vese model presented in [61] to include the shape prior term. This method is similar to that presented by Chan and Zhu in [12]. However, the approach for incorporating the shape prior is based on image moments. Experimental results obtained from 2D image data demonstrate the ability of the geometric shape prior to provide a more accurate segmentation of the target object in images of low-contrast. Also, it is shown that objects can be segmented from an image containing similar background objects and occlusions.

Possible future work for this concept includes:

- Extending to a multiphase piecewise constant segmentation Chan-Vese model.
- Incorporating a linear model as opposed to the piecewise constant model.
- Investigating this concept as applied to texture based segmentation methods.
- Extending this work to 3D.
- Experimenting with higher order moments as shape descriptors.
- Providing a more accurate look-up table regarding the smoothness constraint of the energy.

Although a technique could not be determined which can be applied to the real MRI data used in this research, possible future work also includes providing a solution to this problem using the proposed fast shape prior segmentation concept.

APPENDIX

The following section of code found in 'SegmentData_GAC_Base.cxx' uses ITK to perform the geodesic active contours segmentation. This was implemented and compiled using Microsoft Visual C++. It is assumed that the input image 'input_image' and ϕ_0 'initial_phi', provided by the registration step, have previously been read from file. The final segmentation result is assigned to the image variable 'segmented_binary_image' which is previously defined. Also, the parameters for the various filters are previously defined and the values have been assigned.

/* The first step is to instantiate the anisotropic diffusion filter and set the corresponding parameters. */

```
m_smoothing_filter = AnisotropicDiffusionFilterType::New();
m_smoothing_filter ->UseImageSpacingOn();
m_smoothing_filter ->SetTimeStep(time_step);
m_smoothing_filter ->SetNumberOfIterations(number_of_iters);
m_smoothing_filter ->SetConductanceParameter(conductance_param);
```

/* Next, instantiate a gradient magnitude filter and set the parameters. */

```
m_gradient_magnitude = GradientMagnitudeFilterType::New();
m_gradient_magnitude ->SetSigma(sigma);
```

/* For the high speed term, instantiate a sigmoid filter and set parameters. */

```
m_sigmoid_high = SigmoidFilterType::New();
m_sigmoid_high ->SetOutputMinimum(0.0);
m_sigmoid_high ->SetOutputMaximum(1.0);
m_sigmoid_high ->SetAlpha(alpha_high);
m_sigmoid_high ->SetBeta(beta_high);
```

/* Similarly, for the high speed term, instantiate a sigmoid filter and set parameters. */

```
m_sigmoid_low = SigmoidFilterType::New();
m_sigmoid_low ->SetOutputMinimum(0.0);
m_sigmoid_low ->SetOutputMaximum(1.0);
m_sigmoid_low ->SetAlpha(alpha_low);
m_sigmoid_low ->SetBeta(beta_low);
```

/* Now, create the new speed term with the following steps:
(1) Apply Dirac measure to initial embedding function, initial_phi (Equation (3.3)). */

```
for(int i=0, i<x_dim, i++)
      for(int j=0, j<y_dim, j++)
            delta_phi(i,j) = (1/pi)*(1 /(1+initial_phi(i,j)));
      end
end
```

/* (2) Multiply the low speed term by the delta_phi computed above (Equation (3.2)).*/

```
m_multiplier = MultiplyImageFilterType::New();
m_multiplier->SetInput1(m_sigmoid_low->GetOutput());
m_multiplier->SetInput2(delta_phi);

/* (3) Subtract the result of step (2) from the high speed term (Equation (3.2)). */

m_subtractor = SubtractionFilterType::New();
m_subtractor->SetInput1(m_sigmoid_high);
m_subtractor->SetInput2(m_multiplier->GetOutput());

/* Next, instantiate the geodesic active contour filter and set parameters. */

m_geodesicActiveContour = GeodesicActiveContourFilterType::New();
m_geodesicActiveContour->SetCurvatureScaling(curvature_scaling);
m_geodesicActiveContour->SetPropagationScaling(propagation scaling);
m_geodesicActiveContour->SetAdvectionScaling(advection_scaling);
m_geodesicActiveContour->UseImageSpacingOn();

/* Instantiate threshold image filter and set parameters. */

m_thresholder = ThresholdFilterType::New();
m_thresholder->SetUpperThreshold(itk::NumericTraits<float>::Zero);
m_thresholder->SetLowerThreshold(
      itk::NumericTraits<float>::NonpositiveMin());
m_thresholder->SetInsideValue(1);
m_thresholder->SetOutsideValue(0);

/* Set up the pipeline which connects all of the data and process filters. */

m_smoothing_filter->SetInput(input_image);
m_gradient_magnitude->SetInput(m_smoothing filter->GetOutput());
m_sigmoid_high->SetInput(m_gradient_magnitude->GetOutput());
m_sigmoid_low->SetInput(m_gradient_magnitude->GetOutput());
m_geodesicActiveContour->SetInput(initial_phi);
m_geodesicActiveContour->SetFeatureImage(m_subtractor->GetOutput());
m_thresholder->SetInput(m_geodesicActiveContour->GetOutput());

/* Execute the segmentation filter to perform the segmentation. */

m_geodesicActiveContour->Update();

/* Finally, execute the thresholding filter to get the final binary segmentation result. */

m_thresholder->Update();
segmented_binary_image = m_thresholder->GetOutput();
```

The following is a basic version of the MATLAB function 'bingsong_shape.m' used to perform the fast update Chan-Vese segmentation technique with the additional shape constraint term. This function takes an input image 'image', shape prior image 'atlas', and smoothness constraint look-up table 'delta_length', as parameters. The output includes the final segmentation result 'seg'. Several other MATLAB functions are called within this code, and are therefore provided as well. All MATLAB code was implemented using MATLAB 7.1.

```
function [ seg ] = bingsongshape6( image, shape, dL_lookup )

[rows, cols] = size(image);
% Transformed shape.
shape_reg = 0*shape;
% Change in energy at each pixel.
dE = zeros(rows, cols);
% Randomly assign initial embedding function at each pixel.
seg = 2*(rand(rows, cols)>=0.5)-1;
seg = max(0,seg);

% Assign a weight of influence for each term of the energy.
alpha  = 1.0;       % Region smoothness contraint.
lambda = 1.0;       % Shape constraint.
delta  = 1.0;       % Chan-Vese model-fitting term.
% Assign the variance for each region.
sigma1 = 1.0;       % Inside variance.
sigma2 = 1.0;       % Outside variance.

smooth_itr = 10;    % Iteration to start smooth constraint.
shape_itr  = 20;    % Iteration to start shape constraint.
max_itr    = 100;   % Maximum iterations.

% Weights given to 3x3 neighborhood locations. Used to find the
% index for the smooothness constraint lookup table.
index_wt = [(2^8) (2^5) (2^2);
    (2^7) (2^4) (2^1);
    (2^6) (2^3) (2^0)];

% Compute the image moments for the shape prior.
tree_level = 1;
shape_moment_tree = build_moment_tree(shape, tree_level);

% Begin the segmentation process.
for n=1:max_itr

    % Compute the mean for the inside region.
    inside = image(seg > 0);
    n_inside = size(inside,1);
    mean_inside = sum(sum(inside))/n_inside;

    % Compute the mean for the outside region.
    outside = image(seg <= 0);
    n_outside = size(outside,1);
```

```
mean_outside = sum(sum(outside))/n_outside;
 h = seg>0;

% Compute the image moments for the current segmentation.
[image_moment_tree] = build_moment_tree(h, tree_level);

% Compute the tranformation from the image moments.
[transform_tree] = build_transform_tree(shape_moment_tree,
  image_moment_tree);

% Begin iterating through the image, pixel by pixel.
for i=1:rows
    for j=1:cols
        % Value of the image at the current location.
        I = image(i,j);
        % Value of the current segmentation class.
        class = seg(i,j);
        x=i;y=j;
        % Use the lookup table to find the change in length.
        dL=0;
        if((i>1 && i<rows)&&(j>1 && j<cols))&&(n>=smooth_itr)
            % Current 3x3 neighborhood.
            nhood = h(i-1:i+1,j-1:j+1);
            % Find the index for the current neighborhood.
            ind = sum(sum(nhood.*index_wt)) + 1;
            % Value of smoothness constraint.
            dL = dL_lookup(ind);
        end

        % Find the value of the transformed shape at the current
        % pixel.
        if n>=shape_itr
            % Tranformed pixel location of the shape image.
            x_reg = int16(transform_tree.xform*[x; y; 1]);
            % Value of the transformed shape at current location.
            if x_reg(1)>0 && x_reg(2)>0 && x_reg(1)<=size(shape,1)
              && x_reg(2)<=size(shape,2)
                shape_reg(x,y) = shape(x_reg(1),x_reg(2));
            else
                shape_reg(x,y) = 0;
            end
        end

        %Compute dE if swap pixel from inside to outside.
        if class > 0
            dE(i,j) = delta*(n_outside/(n_outside+1))*((I-
              mean_outside)/sigma2)^2 - delta*(n_inside/(n_inside-
              1))*((I-mean_inside)/sigma1)^2 ;
            if (n > shape_itr)
                %Apply shape contraint.
                dE(i,j) = dE(i,j) + lambda*(shape_reg(x,y)-class);
            end
            % Apply smoothness constraint.
            dE(i,j) = dE(i,j)+alpha*dL;
            %Compute dE if swap pixel from outside to inside.
        else
```

```
                dE(i,j) = delta*(n_inside/(n_inside+1))*((I-
                  mean_inside)/sigma1)^2 - delta*(n_outside/(n_outside-
                  1))*((I-mean_outside)/sigma2)^2;
                if (n > shape_itr)
                    %Apply shape contraint.
                    dE(i,j) = dE(i,j) + lambda*(shape_reg(x,y)-class);
                end
                % Apply smoothness constraint.
                dE(i,j) = dE(i,j)+alpha*dL;
            end
        end
    end

    % Perform the update on pixels with lowest dE values.
    min_dE = min(min(dE));
    if min_dE < 0
        thresh = 0.75*min_dE;
        for i=1:rows
            for j=1:cols
                if dE(i,j) <= thresh
                    seg(i,j) = ~seg(i,j);
                end
            end
        end
    end

end
end
end
```

The following is a basic version of the MATLAB function `'build_moment_tree.m'` used to compute the image moments for a binary image. This function takes an input image `'image'`, and number of image subdivisions `'level'` as parameters. The output includes a tree structure `'tree'`, which contains the image moment information.

```
function [ tree ] = build_moment_tree(image,level)

[R C] = ndgrid(1:size(image,1),1:size(image,2));

% Compute the regular moments.
tree.M00 = sum(sum(image));
tree.M10 = sum(sum(image.*R));
tree.M01 = sum(sum(image.*C));
tree.M11 = sum(sum(image.*R.*C));
tree.M20 = sum(sum(image.*R.*R));
tree.M02 = sum(sum(image.*C.*C));

% Compute the image centroids.
tree.x_bar = tree.M10/tree.M00;
tree.y_bar = tree.M01/tree.M00;

% Compute the second-order central moments.
tree.mu11c = tree.M11/tree.M00 - tree.x_bar*tree.y_bar;
```

```
tree.mu02c = tree.M02/tree.M00 - tree.y_bar^2;
tree.mu20c = tree.M20/tree.M00 - tree.x_bar^2;

% Compute the eigenvalues.
tree.lambda1 = 0.5*(tree.mu20c + tree.mu02c) +
      0.5*sqrt(4*tree.mu11c^2+(tree.mu20c - tree.mu02c)^2);
tree.lambda2 = 0.5*(tree.mu20c + tree.mu02c) -
      0.5*sqrt(4*tree.mu11c^2+(tree.mu20c - tree.mu02c)^2);

% Compute the principal axis direction.
if (tree.mu20c - tree.mu02c) == 0
    if tree.mu11c == 0
        tree.theta = 0;
    elseif tree.mu11c > 0
        tree.theta = pi/4;
    else
        tree.theta = -pi/4;
    end
elseif (tree.mu20c - tree.mu02c) > 0
    if tree.mu11c == 0
        tree.theta = 0;
    else
        tree.theta = 0.5*atan2(2*tree.mu11c,(tree.mu20c - tree.mu02c));
    end
else
    if tree.mu11c == 0
        tree.theta = -pi/2;
    else
        tree.theta = 0.5*atan2(2*tree.mu11c,(tree.mu20c - tree.mu02c));
    end
end

% If the image is to be subdivided, continue computing the image
% moments for the next level of the image.
if level == 1
    tree.child1 = [];
    tree.child2 = [];
else
    % Subdivide the image based on principal axes.
    R = R - tree.x_bar;
    C = C - tree.y_bar;
    n = [cos(tree.theta); sin(tree.theta)];
    region1 = R.*n(1) + C.*n(2) > 0;
    region2 = 1-region1;

    % Get image moments at the next level.
    [tree.child1] = build_moment_tree_1(image.*region1,level-1);
    [tree.child2] = build_moment_tree_1(image.*region2,level-1);
end
end
```

The following is a basic version of the MATLAB function 'build_transform_tree.m' used to compute the transformation which maps the shape image to the current segmentation. This function takes an input image, 'image'; the image moment trees, 'node1' and 'node2', computed from the shape image and the current segmentation, respectfully; and the number of image subdivisions, 'level', as parameters. The output includes a tree structure 'tree', which contains the transformation matrix.

```
function [ tree ] = build_transform_tree(image, node1, node2, level)

image_node = node1;
shape_node = node2;

% The covariance matrix of the shape image.
shape_cov = [shape_node.mu20c shape_node.mu11c;shape_node.mu11c
shape_node.mu02c];

e1_shape = [(shape_node.mu11c/sqrt((shape_node.lambda1-
      shape_node.mu20c)^2+shape_node.mu11c^2)) ((shape_node.lambda1-
      shape_node.mu20c)/sqrt((shape_node.lambda1-shape_node.mu20c)^2 +
      shape_node.mu11c^2))]';
e2_shape = [(shape_node.mu11c/sqrt((shape_node.lambda2-
      shape_node.mu20c)^2+shape_node.mu11c^2)) ((shape_node.lambda2-
      shape_node.mu20c)/sqrt((shape_node.lambda2-shape_node.mu20c)^2 +
      shape_node.mu11c^2))]';

Vshape = [e1_shape(1) -e1_shape(2); e1_shape(2)  e1_shape(1)];
Dshape = Vshape*shape_cov*Vshape';

% The covariance matrix of the image.
image_cov = [image_node.mu20c image_node.mu11c;image_node.mu11c
image_node.mu02c];

e1_image = [(image_node.mu11c/sqrt((image_node.lambda1-
      image_node.mu20c)^2+image_node.mu11c^2)) ((image_node.lambda1-
      image_node.mu20c)/sqrt((image_node.lambda1-image_node.mu20c)^2 +
      image_node.mu11c^2))]';
e2_image = [(image_node.mu11c/sqrt((image_node.lambda2-
      image_node.mu20c)^2+image_node.mu11c^2)) ((image_node.lambda2-
      image_node.mu20c)/sqrt((image_node.lambda2-image_node.mu20c)^2 +
      image_node.mu11c^2))]';

Vimage = [e1_image(1) -e1_image(2);e1_image(2)  e1_image(1)];
Dimage = Vimage*image_cov*Vimage';

% Compute the rotation matrix.
xrot = eye(3,3);
xrot(1:2, 1:2) = Vshape*Vimage';

% Construct the centering matrix.
xcen = [ 1 0 -image_node.x_bar;
    0 1 -image_node.y_bar;
    0 0 1];
```

```
% Construct the translation matrix.
xtrans = [ 1 0 shape_node.x_bar;
    0 1  shape_node.y_bar;
    0 0 1];

% Compute the scaling matrix.
c_shape = 1;
W_shape = [c_shape/sqrt(shape_node.lambda1) 0;
    0 c_shape/sqrt(shape_node.lambda2)];

c_image = 1;
W_image = [c_image/sqrt(image_node.lambda1) 0;
    0 c_image/sqrt(image_node.lambda2)];

scale = eye(3,3);
scale(1:2, 1:2) = W_image*W_shape^-1;

% Compute the final tranformation matrix.
tree.xform = xtrans*xrot*scale*xcen;

% Continue computing the transformation matrix for the next level if
% necessary.
[R C] = ndgrid(1:size(image,1),1:size(image,2));
R = R - image_node.x_bar;
C = C - image_node.y_bar;
n = [cos(image_node.theta2); sin(image_node.theta2)];
tree.region = R.*n(1) + C.*n(2) > 0;

if isempty(image_node.child1)
    tree.child1 = [];
    tree.child2 = [];
else
    tree.child1 = build_transform_tree_1(image, node1.child1,
node2.child1);
    tree.child2 = build_transform_tree_1(image, node1.child2,
node2.child2);
end
end
```

The following is the MATLAB code, `'build_lookup_table.m'`, used to construct the smoothness constraint lookup table. This table contains the change in length `'dL'` associated with each possible 3 x 3 neighborhood combination, which is denoted by a unique index.

```
% Go through the indexes of each possible neighborhood combination.
for n=0:(2^9)-1
    % Determine the 9-element binary string that gives this index.
    bin_str = dec2bin(n,9)
    % Reshape into a 3x3 matrix which represents the neighborhood.
    tmp = reshape(bin_str,3,3);

    % Reassign the values from type character to type integer.
    neigbr = zeros(3,3);
    for i=1:3
```

```
        for j=1:3
            if tmp(i,j) == '1'
                neigbr(i,j)=1;
            end
        end
    end

    % Compute the length for the neighborhood by incrementing for each
    % difference found between 2 neighboring pixels.
    len1(1,1) = sqrt( (neigbr(1,2)-neigbr(1,1))^2 + (neigbr(2,1)-
      neigbr(1,1))^2);
    len1(1,2) = sqrt( (neigbr(2,2)-neigbr(1,2))^2 + (neigbr(1,3)-
      neigbr(1,2))^2);
    len1(1,3) = sqrt( (neigbr(2,3)-neigbr(1,3))^2);

    len1(2,1) = sqrt( (neigbr(2,2)-neigbr(2,1))^2 + (neigbr(3,1)-
      neigbr(2,1))^2);
    len1(2,2) = sqrt( (neigbr(2,3)-neigbr(2,2))^2 + (neigbr(3,2)-
      neigbr(2,2))^2);
    len1(2,3) = sqrt( (neigbr(3,3)-neigbr(2,3))^2);

    len1(3,1) = sqrt( (neigbr(3,2)-neigbr(3,1))^2);
    len1(3,2) = sqrt( (neigbr(3,3)-neigbr(3,2))^2);
    len1(3,3) = 0;

    % Switch the center pixel class value from 0 to 1, or vice versa.
    neigbr(2,2)=~neigbr(2,2);

    % Recompute the length according to the swapped center pixel.
    len2(1,1) = sqrt( (neigbr(1,2)-neigbr(1,1))^2 + (neigbr(2,1)-
      neigbr(1,1))^2);
    len2(1,2) = sqrt( (neigbr(2,2)-neigbr(1,2))^2 + (neigbr(1,3)-
      neigbr(1,2))^2);
    len2(1,3) = sqrt( (neigbr(2,3)-neigbr(1,3))^2);

    len2(2,1) = sqrt( (neigbr(2,2)-neigbr(2,1))^2 + (neigbr(3,1)-
      neigbr(2,1))^2);
    len2(2,2) = sqrt( (neigbr(2,3)-neigbr(2,2))^2 + (neigbr(3,2)-
      neigbr(2,2))^2);
    len2(2,3) = sqrt( (neigbr(3,3)-neigbr(2,3))^2);

    len2(3,1) = sqrt( (neigbr(3,2)-neigbr(3,1))^2);
    len2(3,2) = sqrt( (neigbr(3,3)-neigbr(3,2))^2);
    len2(3,3) = 0;

    % Compute the change in length that corresponds to the given
    % neighborhood combination.
    length_original = sum(len1(:));
    length_final = sum(len2(:));
    dL(n+1) = length_final - length_original;

end
% Save the lookup table.
save 'lookup_table.mat' dL;
```

BIBLIOGRAPHY

1. “Alzheimers Disease Factsheet”. 2006. National Institute of Health/National Institutes on Aging. July 2007. <www.nia.nih.gov/Alzheimers/Publications/adfact.htm.>

2. Andreasen, N.C., et al. “Automatic Atlas-Based Volume Estimation of Human Brain Regions from MR Images.” *Journal of Computer Assisted Tomography* 20 (1996): 98-106.

3. Ayache, N., et al. “Segmentation of Complex Three Dimensional Medical Objects: A Challenge and a Requirement for Computer-Assisted Surgery Planning and Performance.” In R.H. Taylor, S. Lavallee, G.C. Burdea, and R. Mosges, editors, *Computer Integrated Surgery: Technology and Clinical Applications*. MIT Press (1996): 59-74.

4. Ballard, D. H. “Generalizing the Hough Transform to Detect Arbitrary Shape.” *Pattern Recognition* 13 (1981): 111-122.

5. Bezdek, J. C., Hall, L. O. and Clarke, L. P. “Review of MR Image Segmentation Techniques using Pattern Recognition.” *Medical Physics* 20 (1993): 1033-1048.

6. Bomans, M., Hohne, K., Tiede, U., and Riemer, M. “3-D Segmentation of MR Images of the Head for 3-D Display.” *IEEE Transactions on Medical Imaging* 9 June 1990: 253-277.

7. Boykov, Y., and Jolly, M. “Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images.” *ICCV* (2001).

8. “Brainweb: Simulated Brain Database.” July 2007 <http://www.bic.mni.mcgill.ca/brainweb/ >.

9. Broyden, C. G. “A New Double-Rank Minimization Algorithm.” *Notices of the American Mathematical Society* 16 (1969): 670.

10. Caselles, V., Kimmel, R., and Sapiro, G. “Geodesic Active Contours.” *ICCV* (1995): 694-699.

11. Chan, T., and Vese, L. “Active Contours without Edges.” *IEEE TIP* (2001).

12. Chan, T., and Zhu, W. “Level Set Based Shape Prior Segmentation.” *IEEE CVPR'05* 2 (2005): 1164-1170.

13. Chen, Y., et al. “Using Prior Shapes in Geometric Active Contours in a Variational Framework.” *International Journal of Computer Vision* 50(3) (2002): 315-328.

14. Christensen, G. E., Joshi, S. C., and Miller, M. I. “Volumetric Transformation of Brain Anatomy.” *IEEE Transactions on Medical Imaging* 16 (1997): 864-877.

15. Chung, A., Wells, W., Norbash, A., and Grimson, W. “Multi-Modal Image Registration by Minimizing Kullback-Leibler Distance.” *In MICCAI’02 Medical Image Computing and Computer-Assisted Intervention, Lecture Notes in Computer Science* (2002): 525-532.

16. Cocosco, C. A., Kollokian, V., Kwan, R. K.-S., and Evans, A. C. “BrainWeb: Online Interface to a 3D MRI Simulated Brain Database. NeuroImage” vol.5, no.4, part 2/4, S425, 1997 -- *Proceedings of 3-rd International Conference on Functional Mapping of the Human Brain*. Copenhagen, May 1997.

17. Coleman, G.B., and Andrews, H.C. “Image Segmentation by Clustering.” *Proc. IEEE* 5 (1979): 773-785.

18. Collignon, Maes, F., Delaere, D., Vandermeulen, D., Suetens, P., and Marchal, G. “Automated Multi-Modality Image Registration Based on Information Theory.” *Information Processing in Medical Imaging* (1995): 263-274.

19. Collins, D. L., Holmes, C. J., Peters, T. M., and Evans, A. C. “Automatic 3-D Model-Based Neuroanatomical Segmentation.” *Human Brain Mapping* 3 (1995):190-208.

20. Cremers, D., Tischhauser, F., Weickert, J., and Schnorr, C. “Diffusion Snakes: Introducing Statistical Shape Knowledge into the Mumford-Shah Functional.” *International Journal of Computer Vision* 50(3) (2002): 295-313.

21. Cremers, D., Sochen, N., and Schnorr, C. “Towards Recognition-Based Variational Segmentation using Shape Priors and Dynamic Labeling.” In L. Griffith, editor, *International Conference on Scale Space Theories in Computer Vision*, volume 2695 of LNCS: 388-400, Isle of Skye, 2003.

22. Davatzikos, C. “Spatial Normalization of 3D Images using Deformable Models.” *Journal on Computer Assisted Tomography* 20 (1996): 656-665.

23. Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification,* 2nd ed. John Wiley & Sons, Inc. 2000.

24. Duncan, J., and Ayache, N. “Medical Image Analysis: Progress over Two Decades and the Challenges Ahead.” *IEEE PAMI* (2000).

25. Foulonneau, A., Charbonnier, P., and Heitz, F. “Geometric Shape Priors for Region-Based Active Contours.” *In Proc. IEEE Conference on Image Processing* 3 Sept. 2003: 413-416.

26. Grimson, W. E. L., et al. “Utilizing Segmented MRI Data in Image-Guided Surgery.” *International Journal on Pattern Recognition and Artificial Intelligence* 11 (1997): 1367-1397.

27. Haralick, R. M., and Shapiro, L. G. “Image Segmentation Techniques.” *Computer Vision and Graphic Imaging Proceedings* 29 (1985): 100-132.

28. Hornak, J.P. “Basics of MRI.” Sept. 1996. <http://www.cis.rit.edu/htbooks/mri.htm>.

29. Hu, M. K. “Visual Pattern Recognition by Moment Invariants.” *IRE Transactions on Information Theory IT* 8 (1962): 179-187.

30. Ibanez, L., Schroeder, W., Ng, L. and Cates. The ITK Software Guide, the Insight Consortium, <http://www.itk.org>.

31. Jin, Y., Laine, A., and Imielinska, C. “An Adaptive Speed Term Based on Homogeneity for Level-Set Segmentation.” *SPIE* Feb. 2002: 25-28.

32. Kass, M., Witkin, A., and Terzopoulos, D. “Snakes: Active Contour Models.” *International Journal of Computer Vision* (1988).

33. Lancaster, J. L., et al. “Automated Labeling of the Human Brain: A Preliminary Report on the Development and Evaluation of a Forward-Transform Method.” *Human Brain Mapping* 5 (1997): 238-242.

34. Larie, S. M., and Abukmeil, S. S. “Brain Abnormality in Schizophrenia: A Systematic and Quantitative Review of Volumetric Magnetic Resonance Imaging Studies.” *Journal on Psych.* 172 (1998): 110-120.

35. Leventon, M., Faugeraus, O., Grimson, W., and W. W. III. “Level Set Based Segmentation with Intensity and Curvature Priors.” *Mathematical Methods in Biomedical Image Analysis* (2000).

36. Leventon, M., Faugeraus, O., and Grimson, W. “Statistical Shape Influence in Geodesic Active Contours.” *CVPR* (2000): 316-323.

37. Lim, K., Rosenbloom, M., and Pfefferbaum, A. “In Vivo Structural Brain Assessment.” 2000. <http://www.acnp.org/g4/GN401000089/CH.html>.

38. Lo, C.-H., Don, H.-S. “3D Moment Forms: Their Construction and Application to Object Identification and Positioning.” *IEEE PAMI* 11 Oct. 1989: 1053-1064.

39. MacQueen, J. “Some Methods for Classification and Analysis of Multivariate Observation.” *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (1967): 281-297.

40. Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., and Suetens, P. "Multimodality Image Registration by Maximization of Mutual Information." *IEEE Transactions on Medical Imaging* 16(2) (1997): 187-198.

41. Maintz, J., and Viergever, M. "A Survey of Medical Image Registration." *Medical Image Analysis* (1998).

42. Malladi, R., Sethian, J. A., Vemuri, B.C. "Shape Modeling with Front Propagation: A Level Set Approach." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(2) (1995): 158-175.

43. Mangin, J.F., Frouin, V., Bloch, I., Regis, J., and Lopez-Krahe, J. "From 3D Magnetic Resonance Images to Structural Representations of the Cortex Topography using Topology Preserving Deformations." *Journal of Mathematical Imaging and Vision* 5 (1995): 297-318.

44. Marroquin, J., Santana, E., Botello, S. "Hidden Markov Measure Field Models for Image Segmentation." *IEEE PAMI* (2003).

45. Mattes, D., Haynor, D. R., Vesselle, H., Lewellen, T., and Eubank, W. "Non-rigid Multimodality Image Registration." *In Medical Imaging 2001: Image Processing* (2001): 1609-1620.

46. Mattes, D., Haynor, D. R., Vesselle, H., Lewellen, T., and Eubank, W. "PET-CT Image Registration in the Chest Using Free-form Deformations." *IEEE Transactions on Medical Imaging* 22(1) Jan. 2003: 120-128.

47. Mumford, D., and Shah, J. "Optimal Approximation by Piecewise Smooth Functions and Associated Variational Problems." *Comm. Pure Appl. Math.* 42 (1989): 577-685.

48. Osher, S., and Sethian, J.A. "Fronts Propagating with Curvature Dependent Speed: Algorithms based on Hamilton-Jacobi Formulation." *J. Comput. Phys.* 79 (1988): 12-49.

49. Pearlson, G.D., Harris, G.J., Powers, R.E., Barta, P.E., Camargo, E.E., Chase, G.A., Noga, J.T. and Tune, L.E. "Quantitative Changes in Mesial Temporal Volume, Regional Cerebral Blood Flow, and Cognition in Alzheimer's Disease." *Arch Gen Psychiatry* 49 (1992): 402-408.

50. Persoon, E. and Fu, K. S. "Shape Discrimination using Fourier Descriptors." *IEEE Transactions on Systems, Man, and Cybernetics* 7 (1977): 170-179.

51. Prasanna K. Sahoo, Soltani, S., Wong, A. K. C. "A Survey of Thresholding Techniques." *Computer Vision, Graphics, and Image Processing* 41(2) (1988): 233-260.

52. Pham, D. L., Xu, C., and Prince, J. L. "Current Methods in Medical Image Segmentation." *Annual Review of Biomedical Engineering* 2 (2000): 315-337.

53. Pluim, J. P., Maintz, J. B. A., and Viergever, M. A. “Mutual-Information-Based Registration of Medical Images: A Survey.” *IEEE Transactions on Medical Imaging* 22(8) Aug. 2003: 986-1004.

54. Powell, M. J. D. “An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives.” *Computer Journal* 7 (1964): 155-162.

55. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes in C.* Cambridge University Press, 2nd ed. 1992.

56. Rosenfeld, A., and Kak, A. C. *Digital Picture Processing*. Academic Press, New York, 1982.

57. Rousson, M., and Deriche, R. “A Variational Framework for Active and Adaptive Segmentation of Vector Valued Images.” *In Proc. IEEE Workshop on Motion and Video Computing* Dec. 2002: 56-61.

58. Rousson, M., and Paragios, N. “Shape Priors for Level Set Representations.” *In Proc. Seventh European Conf. Computer Vision* May 2002: 78-93.

59. Sandor, S., and Leahy, R. “Surface-based Labeling of Cortical Anatomy using a Deformable Atlas.” *IEEE Transactions on Medical Imaging* 16 (1997): 41-54.

60. Sethian, J.A. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1996.

61. Song, B., and Chan, T. “A Fast Algorithm for Level Set Based Optimization.” *CAM-UCLA* 68 (2002).

62. Spendley, W., Hext, G. R., and Himsworth, F. U. “Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation.” *Technometrics* 4 (1962): 441-461.

63. Sprawls, P. *Physical Principles of Medical Imaging*. Aspen Publishers, 1993.

64. Talairach, J., and Tournoux, P. *Co-Planar Stereotaxic Atlas of the Human Brain. 3-Dimensional Proportional System: An Approach to Cerebral Imaging.* Thieme Medical Publisher, Inc., Stuttgart, New York, 1988.

65. Taza, A., and Suen, C. Y. “Discrimination of Planar Shapes using Shape Matrices.” *IEEE Transactions on Systems, Man, and Cybernetics* 19(5) (1989): 1281-1289.

66. Thevenaz, P., and Unser, M. “Optimization of Mutual Information for Multi-Resolution Image Registration.” *IEEE Transactions on Image Processing* 9(12) Dec. 2000.

67. Unser, M. “Splines: A Perfect Fit for Signal and Image Processing.” *IEEE Signal Processing Magazine* 16(6) Nov. 1999: 22-38.

68. Unser, M., Aldroubi, A. and Eden, M. “B-Spline Signal Processing: Part I-Theory.” *IEEE Transactions on Signal Processing* 41(2) Feb. 1993: 821-832.

69. Unser, M., Aldroubi, A. and Eden, M. “B-Spline Signal Processing: Part II-Efficient Design and Applications.” *IEEE Transactions on Signal Processing* 41(2) Feb. 1993: 834-848.

70. Vemuri, B. C., Chen, Y., and Wang, Z. “Registration Assisted Image Smoothing and Segmentation.” *ECCV* 4 (2002): 546-559.

71. Worth, A. J., Makris, N., Caviness, V. S., and Kennedy, D. N. “Neuroanatomical Segmentation in MRI: Technological Objectives. *International Journal on Pattern Recognition and Artificial Intelligence* 11 (1997): 1161-1187.

72. Xiaohong, W., and Rongchun, Z. “A New Method for Image Normalization.” *Intelligent Multimedia, Video and Speech Processing* (2001): 356-359.

73. Xu, C., Pham, D.L., Prince, J.L. *Image Segmentation using Deformable Models.* In Sonka, M., Fitzpatrick, J.M., eds.: Handbook of Medical Imaging. Volume 2. SPIE Press. 129-174, 2000.

74. Zahn, C. T., and Roskies, R. S. “Fourier Descriptors for Plane Closed Curves.” *IEEE Transactions on Computing* 2(1) (1972): 269-281.

75. Zitova, B., and Flusser, J. “Image Registration Methods: A Survey.” *Image and Vision Computing* 21(11) (2003).