Development and Application of Semi-automated ITK Tools For the Segmentation of Brain MR Images

By
Shilpa N. Kinkar

A Thesis
Submitted to the Faculty
Of
Worcester Polytechnic Institute

In partial fulfillment of the requirements for the
Degree of Master of Science
In
Computer Science

By

_____

May 2005

APPROVED:

_____
Professor John M. Sullivan, Jr.,
Thesis Advisor

_____
Professor Michael A. Gennert
Thesis Reader

_____
Professor Michael A. Gennert
Head of Department

Abstract

Image segmentation is a process to identify regions of interest from digital images. Image segmentation plays an important role in medical image processing which enables a variety of clinical applications. It is also a tool to facilitate the detection of abnormalities such as cancerous lesions in the brain.

Although numerous efforts in recent years have advanced this technique, no single approach solves the problem of segmentation for the large variety of image modalities existing today. Consequently, brain MRI segmentation remains a challenging task.

The purpose of this thesis is to demonstrate brain MRI segmentation for delineation of tumors, ventricles and other anatomical structures using Insight Segmentation and Registration Toolkit (ITK) routines as the foundation. ITK is an open-source software system to support the Visible Human Project. Visible Human Project is the creation of complete, anatomically detailed, three-dimensional representations of the normal male and female human bodies. Currently under active development, ITK employs leading-edge segmentation and registration algorithms in two, three, and more dimensions. A goal of this thesis is to implement those algorithms to facilitate brain segmentation for a brain cancer research scientist.

Acknowledgment

I am extremely grateful for the opportunity to have Professor John M. Sullivan Jr., as my thesis advisor for the past two years. He has been the motivation behind this thesis and always inspired me in our research talk. I learned a lot from him not only on our project, but on the method to do research as well.

I would like to thank Professor Michael A. Gennert for being the reader of this thesis and for his invaluable feedback on the thesis.

I would like to express my greatest gratitude to my parents, parents-in-laws, my brother and brother in-law and my husband for their constant and strongest support and utmost encouragement for my graduate studies and this thesis. I would also like to thank my uncle Subhash Dandage and aunt Kanchan Dandage who have encouraged me for the studies and have been our local guardians in USA.

I would also like to thank my team members, Praveen, Hamid, Murali, James and Wei for their co-operation in past two years. I would like to thank Praveen especially for sharing the knowledge and helping me in general during past 2 years.

I would also like to thank WPI Computer Science department, WPI Mechanical department and CCNI (Center for Comparative NeuroImaging) for providing me with this opportunity to work.

I acknowledge and appreciate the strongest support of my husband Shirish without whom I would be lost.

This work was supported, in part, from NCI P01-CA80139

Table of Contents

Table of Figures

Chapter 1: Introduction

Image segmentation is a process to identify and classify regions of interest from digital images. Typically, digital images are acquired from medical instrumentation as CT (Computer Tomography) or MRI (Magnetic Resonance Image) scanners, digital mammograms etc. Image segmentation plays an important role in medical image processing which enables a variety of clinical applications [1, 2, 6, 7]. It is also a tool to facilitate the detection of abnormalities such as cancerous lesions in the brain. Segmentation of medical images is a challenging task [3, 6]. A variety of different methods have been proposed and implemented in recent years [3, 9]. Although numerous efforts have advanced this technique, there is no single approach that can generally solve the problem of segmentation for the large variety of image modalities existing today [3].

Various segmentation techniques can be classified, for example, as classical techniques such as thresholding, boundary based technique, region based technique, or statistical technique [2, 8]. Depending on the level of interactivity, segmentation can be classified as manual, semiautomatic, or automatic [2]. Manual segmentation is time consuming [2, 6], costly, and non-repeatable [2]. Inconsistencies in the segmented extent of various structures are common among qualified experts. Moreover, it does not use the full multi-dimensional image data [4]. Automatic segmentation methods are sensitive to noise and unexpected situations, leading to errors [2]. Their advantages are a minimum time commitment from the user and results that are highly reproducible, albeit potentially erroneous [2]. These methods are usually problem-specific, and an image-processing expert is needed to determine which image-processing functions are best suited for a given segmentation task [4].

Semi-automatic image segmentation combines manual interaction with automated sub-components to solve segmentation problems. Semi-automatic segmentation is faster than a

manual strategy with more reproducible results compared to both manual and fully automatic. It uses the complete image data set, and is minimally affected by human inconsistency and error. In addition, the user can harness the power of automated image-processing algorithms without being an image-processing expert [4].

1.1 Purpose of Thesis

Medical image segmentation has become an important diagnostic tool in the practice of modern medicine. Segmentation of MRI Brain images is the delineation of neuro-anatomical structures such as the Cerebrum, Cerebellum, Hippocampus, etc. as well as abnormalities such as tumors [11]. Although numerous methods have been proposed during the past two decades for brain MRI segmentation, it remains a challenging task [3, 6]. In this thesis brain MRI segmentation is demonstrated for delineation of a tumor, ventricles and other anatomical structures using Insight Segmentation and Registration Toolkit (ITK) routines as the foundation. This public software provides callable routines only. It does not provide graphics, visualization, or other interactive tools specific to a given task, such as segmentation. We have implemented a variety of semiautomatic and automatic segmentation routines as well as preprocessing algorithms provided by ITK. This implementation facilitates interactive adjustments of algorithm parameters and their consequences on the segmentation results via an intuitive GUI incorporating the ITK segmentation routines.

1.2 Outline of Thesis

Chapter 1 briefly introduced Image Segmentation and the thesis objectives and outline. Chapter 2 provides background on MRI and ITK. The MRI background briefly explains the principle of MRI. Subsequently, the history of ITK, its features, advantages and rationale for using ITK for this thesis is discussed. The compilation and build process of ITK carried out on Windows platform, using CMake build process is also discussed. Chapter 3 describes different segmentation methods such as Region Growth, Watershed and Level Set Segmentation and their details as implemented in ITK. Chapter 4 discusses the programs developed using ITK's basic set of algorithms. It also discusses results obtained using these programs applied upon Brain MR Images. Chapter 5 demonstrates and discusses results of various segmentation methods described previously. This chapter evaluates different segmentation methods and finally provides different pathways to efficiently delineate the different anatomical structures and abnormalities inside brain MRI using these programs. The conclusions are in chapter 6 followed by references.

Chapter 2: Background

There are multiple medical imaging strategies such as MRI (Magnetic Resonance Imaging), PET (Positron emission Tomography), CT (Computer Tomography), Ultrasound, SPECT (Single Photon Emission Computed Tomography and many more. This thesis has concentrated on MRI imaging due to the research focus of our laboratory, the Center for Comparative NeuroImaging.

Magnetic resonance imaging (MRI) is an imaging technique used primarily in medical settings to produce high quality images of soft tissue structures. Unlike conventional X-ray imaging or Computed Tomography, which produce images that show the X-ray attenuation of tissues, MRI measures the amount of hydrogenous materials (water and lipids) in tissues. The nucleus of the hydrogen atom is a spinning charged proton with magnetic properties which the MR imaging strategy utilizes.

Two magnetic fields are used in MRI. The first being a strong static magnetic field which causes the hydrogen atoms in the body to align in a direction parallel to the field. A second magnetic field (radio-frequency pulse) is applied at right angle to the first field causing the hydrogen atoms to change their alignments. When radio-frequency pulse is turned off, the hydrogen atoms return to their alignment along the static magnetic field direction. The time rate of recovery differs with tissue properties. Longitudinal relaxation (T1) depends on the recovery time of the hydrogen atoms to return to the axis of the primary magnetic field. The transverse relaxation time (T2) measures the rate of decay of the hydrogen proton alignment due to the RF pulse.

2.1 MR Image Segmentation

2.1.1 Basic components in Computer Vision system

The figure 2-1 shows representative diagram of basic components in a computer vision system.

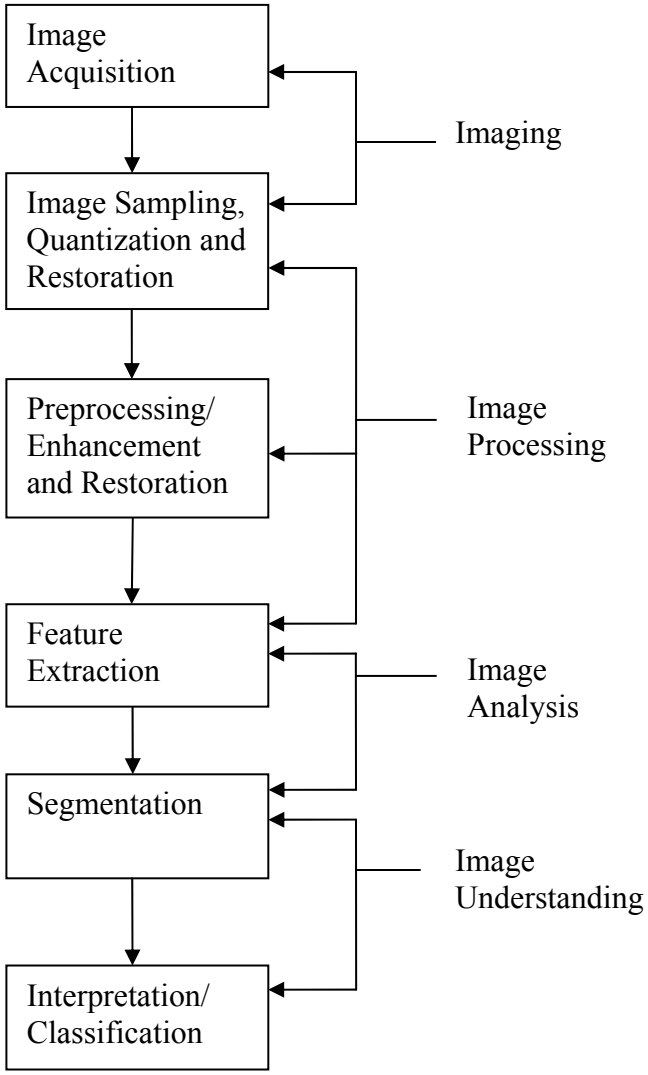


Figure 2-1 Basic Steps in Computer Vision

Image acquisition is done using MRI, CT etc. Image preprocessing such as smoothing improves the quality of image by removing noise, image artifacts etc. Feature Extraction is typically used for guiding segmentation methods, preparing data for registration methods or as a mechanism for recognizing anatomical structures in an image. It extracts the features such as edge and texture. Segmentation groups pixels into regions and hence defines boundaries of tissue regions. Selection of delineable features is the key to successful segmentation.

2.1.2 Segmentation Methods

Several methods of image segmentation have been proposed in literature. The following figure 2-2 arranges these methods.

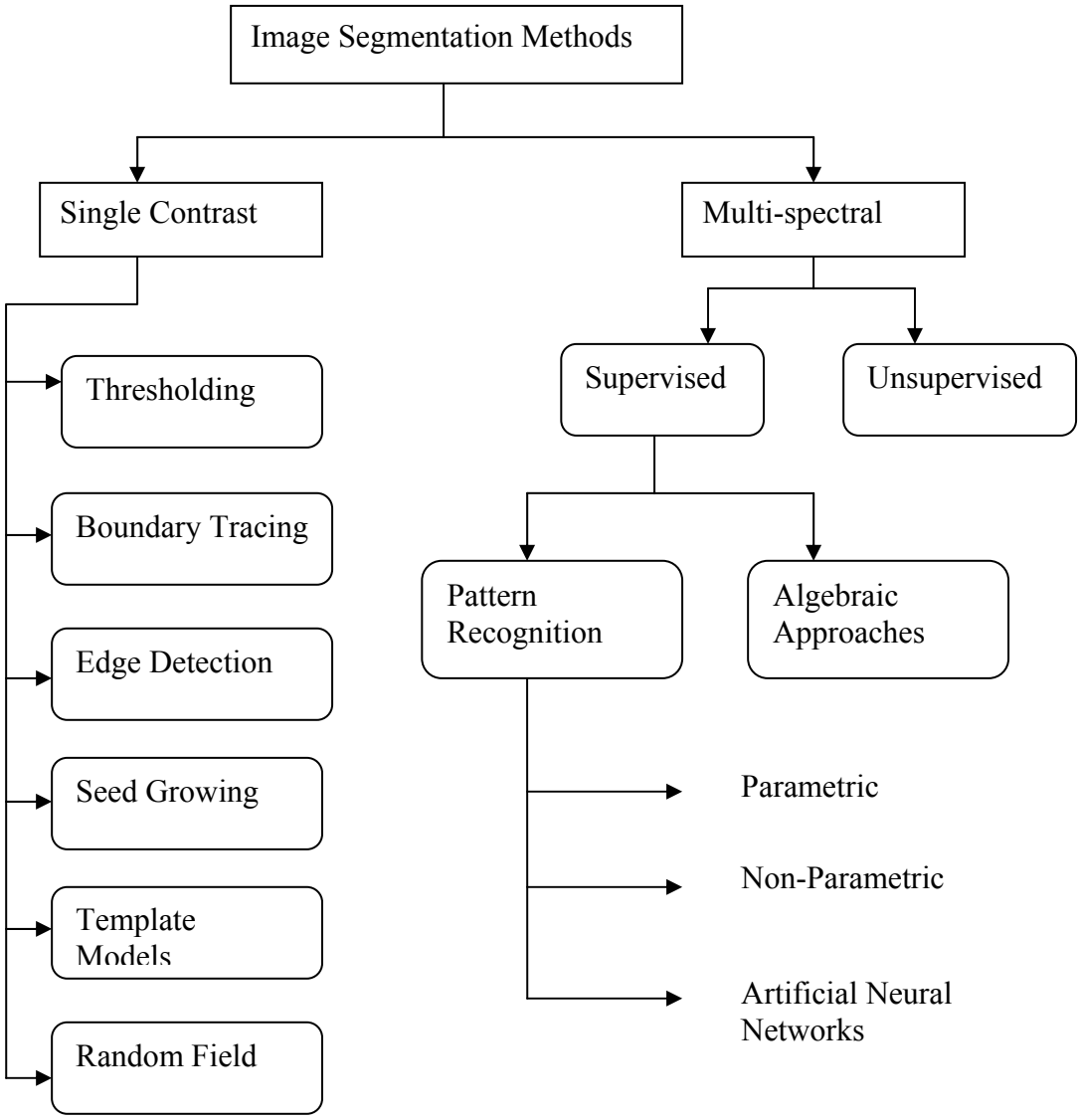


Figure 2-2 Image Segmentation Methods

2.1.2.1 Single Contrast Methods:

Thresholding: Thresholding is the most intuitive approach to segmentation where values of upper and lower thresholds are provided by the user. This method is limited and successful application for clinical use is hindered by the variability of anatomy and image artifacts.

Edge Based Segmentation: Edges are usually detected by rapid changes in intensity values observed while traversing a set of pixels. Operators such as first and second derivatives play significant role in edge detection. Sometimes, insufficient intensity gradients, presence of noise or artifacts, or poor guess of threshold may cause over or under segmentation.

Boundary Tracing: In this method, user selects the pixel on the boundary of the region for outlining purposes. Then the method follows the boundary from this user selected seed point. This method is useful for cases where good definition of an edge boundary exists. This method can cause problems for images with tissue variance.

Seed Growing Segmentation: In this method, user selects a seed or multiple seeds and specifies threshold value. Using this data, the method examines neighboring pixels of the selected seed(s) and includes them in the region if those neighbors also satisfy the criteria.

2.1.2.2 Multi-Spectral Segmentation:

These methods are classified as supervised and un-supervised.

Supervised methods require operator input for segmentation. This is done by selecting training pixels or training regions in the images. Un-supervised methods define regions in the image without operator input.

2.1.2.2.1 Supervised Methods:

Pattern Recognition Methods: This is the most common approach for multi-spectral segmentation. There are several pattern recognition techniques and many of them assume particular distribution of the features based upon parametric models. Non-parametric models do not rely on predefined distributions but on the actual distribution of the training samples.

Algebraic Methods: For images that clearly identify signature vectors, these methods provide an elegant solution to deal with the partial volume effect, which might have influence on volume

measurements. Algebraic approaches can become impractical for images showing complex pathology.

2.1.2.2.2 Unsupervised methods:

Unsupervised methods, also called "clustering" find the structure in data automatically. A cluster is an area in feature space with high density and can be promising for tumor volume determination. However, the initialization is very important for meaningful clustering and reasonable computation time. These methods are reproducible but may not necessarily arrive at meaningful segmentation and often require computation time.

ITK (Insight Segmentation and Registration Toolkit) has implemented some segmentation methods discussed. ITK is an open-source software system to support Visible Human Project and is discussed in the next chapter.

2.2 Background of ITK

2.2.1 What is ITK?

Insight Segmentation and Registration Toolkit (ITK) is an open-source software system to support the Visible Human Project. ITK is sponsored by National Library of Medicine at the National Institutes of Health and developed by six principal organizations, three commercial (Kitware, GE Corporate R&D, and Insightful) and three academic (UNC Chapel Hill, University of Utah, and University of Pennsylvania). ITK is currently advancing leading-edge segmentation and registration algorithms in two, three, and more dimensions. ITK does not provide GUI or Visualization. ITK is cross-platform package. A common build environment called CMake is used to manage the compilation process.

2.2.2 What is CMake?

CMake is a cross-platform open source software system. It was developed by Kitware, as part of the ITK project. CMake is a build process which is independent of the operating system and compiler. Currently it supports UNIX and Windows platforms and produces native build files appropriate to these OS. On Unix CMake produces makefiles and on Windows, it generates projects and workspaces. CMake allows the user to control the software compilation process using simple platform and configuration files (contain pre-defined CMake commands as well as user-defined commands) that are compiler independent. In addition, an automated wrapping process generates interfaces between C++ and interpreted programming languages such as Tcl, Java, and Python. Wrapping process is achieved using CableSwig. This ITK utility enables developers to create software using a variety of programming languages.

2.2.2.1 Advantages of CMake

Following are the advantages of using CMake.

- Compiles source code
- Creates libraries
- Generates wrappers
- Supports static and dynamic library
- CMake provides portable configuration Management (Windows & UNIX)
- Supports complex directory hierarchies and applications dependent on several libraries (consisting complex directory hierarchies) plus additional code

ITK is implemented in C++ programming language. It uses generic programming style of C++ that uses templated code extensively. Templated code is highly efficient. It also helps to discover many software problems at compile-time, rather than at run-time during program execution.

ITK uses a model of software development called as *extreme programming*, which is a simultaneous and iterative process of design-implement-test-release. This approach helps to manage the rapid evolution of the software. ITK is also tested everyday using Dart. Dart is an open-source, distributed, software quality system that allows software projects to be tested at multiple sites in multiple configurations (hardware, operating systems, compilers, etc.).

2.2.3 Why ITK?

ITK was chosen for this thesis since ITK is specifically developed for segmentation and registration of medical images. It was developed by a team with expertise in a wide range of areas related with registration and segmentation.

Some of the advantages of ITK are:

- Since ITK provides automated wrapping process, it is possible to develop and customize the application using variety of programming languages such as TCL, Java, and Python.
- ITK is organized around data-flow architecture: Data is represented using data objects which are in turn processed by process objects (filters). Data objects and process objects are connected together into pipelines. This loosely coupled architecture allows efficient code extension and re-usability as well as variety of data formats. We can add support for our own data format to develop the application using ITK.
- ITK can process images of N-dimension.
- ITK facilitates efficient memory management through the use of "smart-pointers" by using garbage collection in an efficient manner. That is, "smart pointers" can be allocated on the stack, and when scope is exited, the smart pointers disappear and decrement their reference count to the object that they refer to.

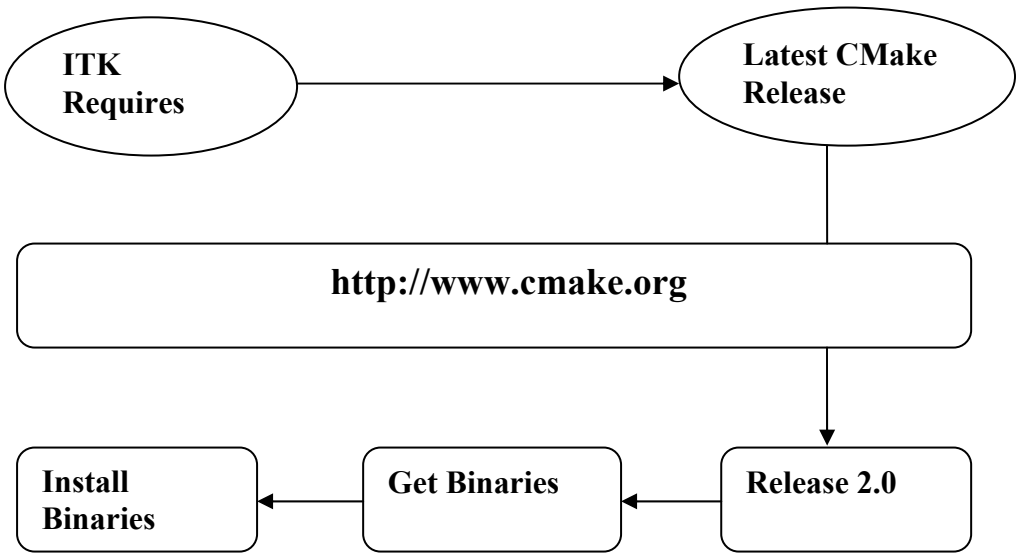
- It supports multi-threading, which allows us to develop programs to run on parallel processors to facilitate data parallelism.

2.3 Compilation of ITK

This thesis work started with ITK version 1.0. This version and its successive version releases were downloaded from ITK Website: http://www.itk.org/HTML/Download.htm.

Following pages describes the compilation and build process of ITK version 1.8 which was released in August 2004.

2.3.1 Building ITK: What do we need?



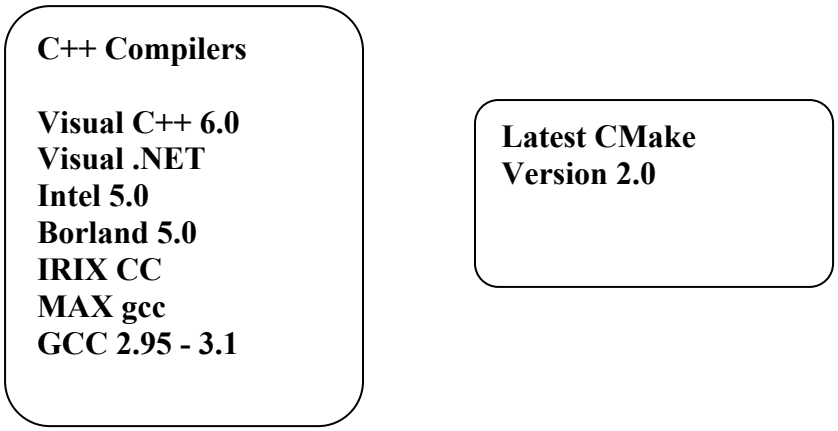


Figure 2-3 Building ITK

2.3.2 Building ITK using CMake

The build process of ITK is controlled by creating one or more CMakeLists.txt files in ITK's main directory and subdirectories that make up a project. Each CMakeLists.txt consists of one or more commands. Each command has the form COMMAND (args...) where COMMAND is the name of the command, and 'args' is a white-space separated list of arguments.

CMake generates a cache file which is designed to be used with a graphical editor. This cache file contains the information about include directories, libraries, executables and other optional build directives. This cache file can be changed by the user prior to generating native build files. Figure 2-4 shows CMake cache GUI in Windows MSVC environment.

ITK 1.8 was compiled and built on Windows XP, using Microsoft Visual C++ 6.0 as follows: Default configuration that was used for building ITK to create workspace files using CMake is shown in figure 2-4. Advanced Variables were used with their default values.

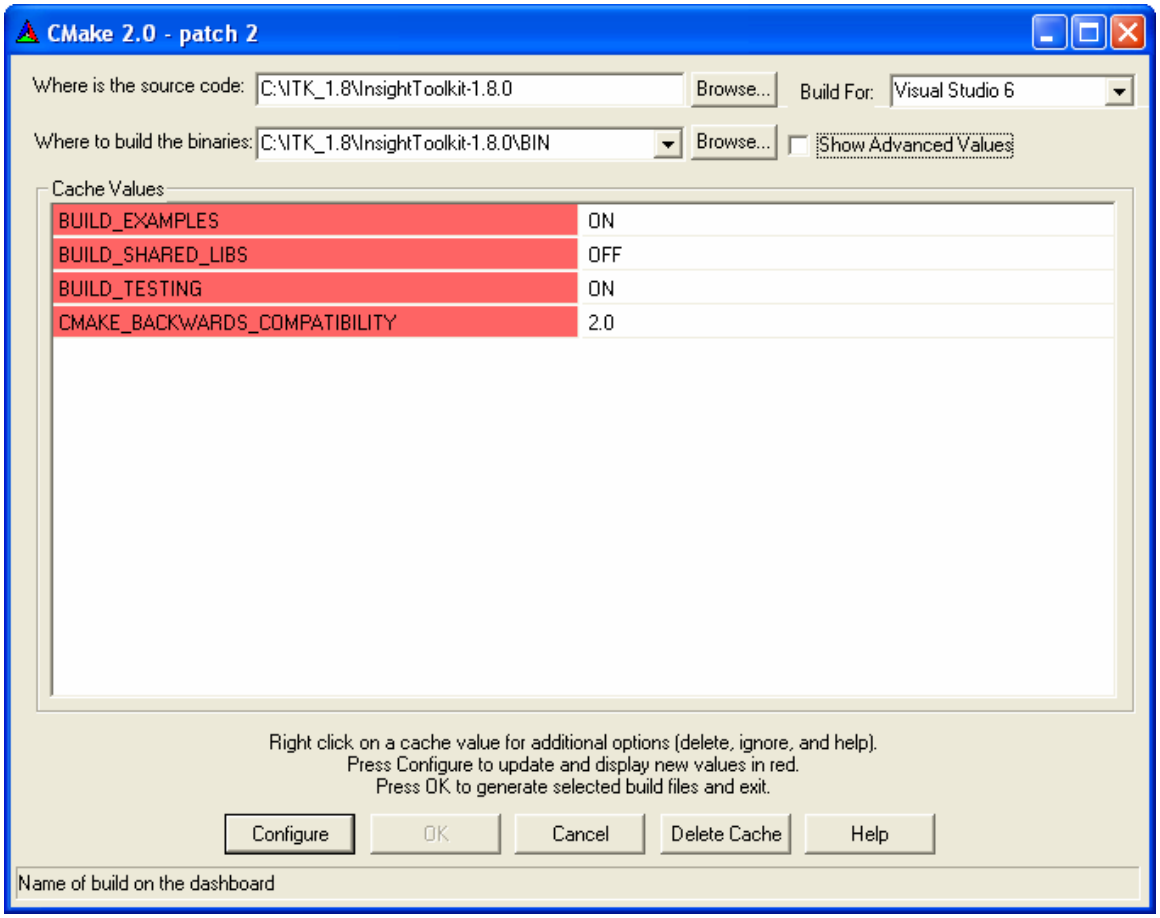


Figure 2-4 Using CMake

After clicking "Configure" button in above GUI, CMake will configure ITK project and create ITK.dsw workspace on Windows platform. This workspace is then opened in Windows native compiler, Microsoft Visual C++ 6.0 and run the project to create ITK binaries. Following configuration was used to build ITK binaries:

Debug

RelWithDebugInfo

MinSizeRel

ITK binaries were tested for proper functioning, for ex.

itkBasicFiltersTests.exe itkGradientImageFilter

2.3.2.1 Executing Tests

Following steps were used to execute the tests:

STEP 1: Go to BINARY directory (as set in CMake)

STEP 2: ctest –R itkGradientImageFilter

(-R says any test matching this string is executed). Here, without –R, all tests will be executed. 'cdtest' is companion program to CMake

2.3.3 Using ITK from external project

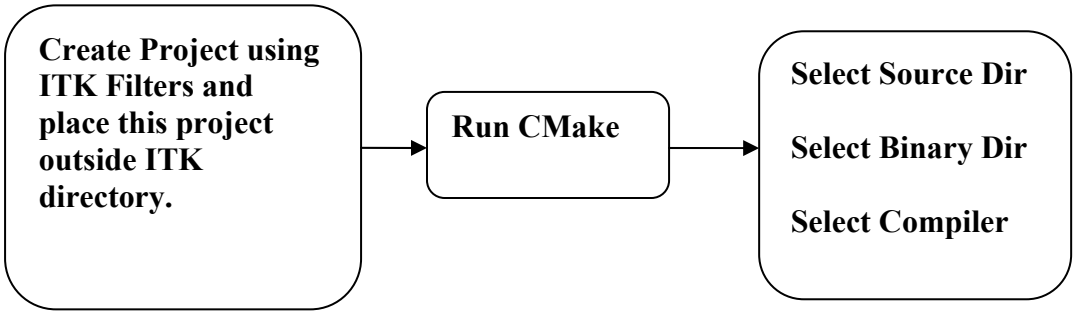


Figure 2-5 Using ITK from external project

Following configuration file (CMakeList.txt) was used for a sample project to read and write RAW Images using ITK.

```
PROJECT (BasicRawIO)

# Find ITK
INCLUDE (${CMAKE_ROOT}/Modules/FindITK.cmake)
 IF (USE_ITK_FILE)
   INCLUDE (${USE_ITK_FILE})
 ENDIF (USE_ITK_FILE)

#Specify executables
ADD_EXECUTABLE ( ReadRAWWrite   ReadRAWWrite.cpp )
TARGET_LINK_LIBRARIES ( ReadRAWWrite ITKCommon ITKIO
ITKNumerics  ITKStatistics  )
TARGET_LINK_LIBRARIES ( ReadRAWWrite itkpng itkzlib
ITKAlgorithms ITKBasicFilters ITKMetaIO ITKFEM )
```

2.4 Integration of ITK with CWBench

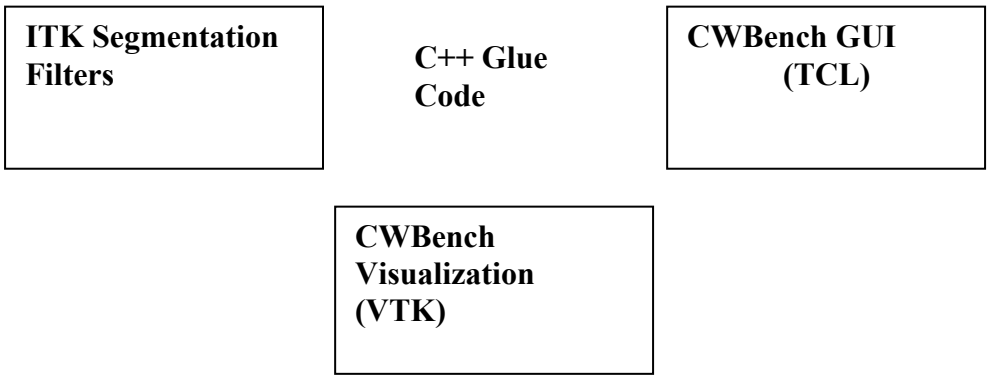


Figure 2-6 Integrating ITK with CWBench

Chapter 3: ITK Segmentation Methods

3.1 Introduction

Image segmentation plays an important role in medical image processing. The goal of segmentation is to extract one or several regions of interest in an image. Depending on the context, a region of interest can be characterized based on a variety of attributes, such as gray-scale level, contrast, texture, shape, size etc. Selection of good features is the key to successful segmentation. There are a variety of techniques for segmentation, ranging from simple ones such as thresholding, to more elaborate strategies including region-growing, edge-detection, morphological methods, artificial neural networks and much more.

Although numerous methods have been proposed in recent years, there is no single approach that can solve segmentation problems for many existing imaging modalities today.

The Insight Toolkit provides a basic set of algorithms that can be used to develop and customize a full segmentation application. The most effective segmentation algorithms can be obtained by carefully customizing combinations of the basic segmentation algorithms and pre-processing and post-processing algorithms implemented in ITK for images of specific modality. The parameters of these algorithms can be tuned for the characteristics of the image modality used as input and the features of the anatomical structure or abnormalities to be segmented.

Some of the most commonly used segmentation components are described in the following sections.

3.2 Region-growing segmentation

Region-based segmentation algorithms postulate that neighboring pixels within the same region have similar intensity values. The general approach is to compare a pixel with its immediate surrounding neighbors and to group pixels or sub-regions into larger regions based on predefined criteria [12, 13]. If a criterion of homogeneity is satisfied, the pixel can be classified

into the same class as one or more of its neighbors [12, 13]. The choice of homogeneity criterion is therefore critical to the success of the segmentation.

The basic approach starts with a seed point (or multiple seed points). From these seed points, region grows by appending to each seed those neighboring pixels that have properties similar to the seed. The process continues as long as new pixels are added to the region. Criteria can be, gray-scale level, color, texture etc. The selection of criteria for the similarities depends not only on the problem under consideration but also the type of image to be segmented [13].

Region growing algorithms vary depending on the criteria used to decide whether a pixel should be included in the region or not, the type connectivity used to determine neighbors, and the strategy used to visit neighboring pixels. Different implementations of region growing segmentation are available in ITK such as Connected Threshold, Neighborhood Connected, Confidence Connected. Details of these algorithms and their implementation are described in the next chapter.

3.3 Segmentation Based On Watershed

Watershed algorithm is one of the most reliable automatic and unsupervised segmentation strategies [15-16]. It is one of the most popular methods due to its capability to adapt itself to very different types of images [19, 20]. Watershed segmentation classifies pixels into regions (catchment basins) [6, 14]. These regions are formed by using local geometric structure to associate points in the image domain with local extrema in some feature measurement such as curvature or gradient magnitude [6, 17, 14]. The basic idea is: Image is viewed in three dimensions, 2 spatial coordinates versus gray levels [14]. With this topographic interpretation, suppose that there is hole punched in each region and entire topography is flooded from below by letting water rise through the holes at a uniform rate [14]. When rising water in distinct catchment basins is about to merge, a dam is built to prevent merging. These dam boundaries

correspond to boundaries extracted by watershed algorithm [14]. The important property of this algorithm is that it gives continuous boundaries between regions [14].

Watershed algorithms are sensitive to the noise and contrast in image and the watershed lines are poorly localized if smoothing is not used [22]. Several algorithms such as Edge Preserving Anisotropic Diffusion filter are proposed for improving the performance of watersheds.

Figure 3-1 illustrates the watershed transformation to be performed on a topographic surface of Visible Human female head and neck cryosection data.

Figure 3-1 Original Image of Visible Human female head and neck cryosection data [17]

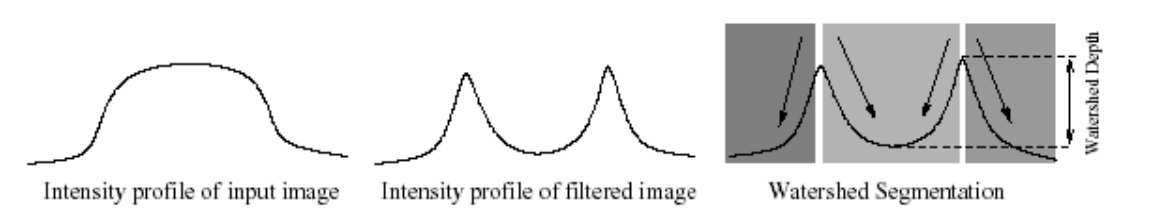


Figure 3-2 Intensity Profiles

Figure 3-3 Segmented sections of Visible Human female head and neck cryosection data [17]

Figure 3-2 illustrates intensity profiles of input image, filtered image and catchment basins with watershed depth, respectively. Figure 3-3 represents the result of watershed segmentation.

A typical approach for segmenting a gray-scale image with the watershed method is to make use of its gradient image as an input to the transformation since high gradients constitute watershed lines that correspond to the region boundaries of the gray-scale image, as shown is rightmost figure 3-2. This method is called the *gradient watershed*. ITK implements this type of watershed algorithm, which is discussed in the next chapter.

3.4 Level Set Methods

These methods started with the work of Osher and Sethian (Front Propagating with Curvature-dependent speed: Algorithms based on Hamilton-jacobi formulations", Journal of Computational Physics 79, 12-49, 1988) who showed how deformation could be modeled on discrete grids using level sets.

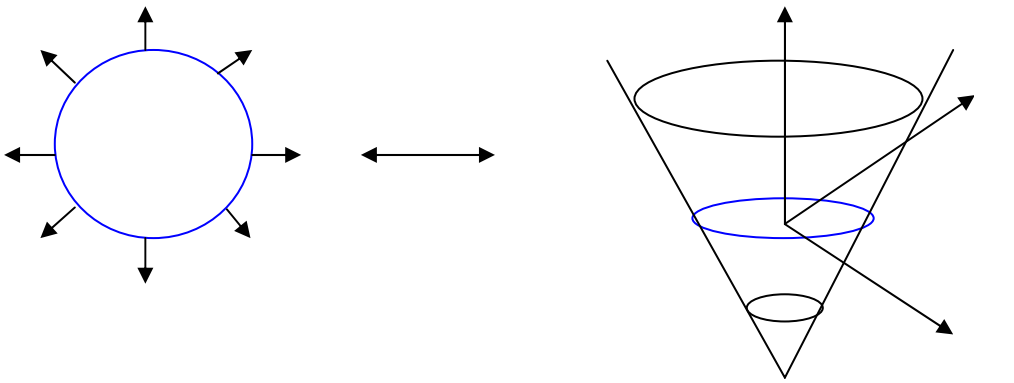
Level set techniques, also known as implicit active contours have been the subject of active research in the last few years.

The idea behind active contours (deformable models) is that the user specifies an initial guess for the contour, which is then moved by image driven forces to the boundaries of the desired objects. In such models, two types of forces are considered - the internal forces, defined within the curve, are designed to keep the model smooth during the deformation process, while the external forces, which are computed from the underlying image data, are defined to move the model toward an object boundary or other desired features within the image.

There are two forms of deformable models. In the parametric form, also referred to as snakes, an explicit parametric representation of the curve is used. In contrast, the implicit deformable models, (also called implicit active contours or level sets), are designed to handle topological changes naturally. In a Level Set approach, instead of following the interface/contour directly, the contour is built into a surface. That is, contour is embedded as the zero level set of a higher dimensional function called the level-set function [18]. The level-set function is then evolved under the control of a partial differential equation. The motion of the interface is matched with the zero level set of the level set function, and the resulting initial value partial differential equation for the evolution of the level set function resembles a Hamilton-Jacobi equation. In this setting, curvatures and normals may be easily evaluated, topological changes occur in a natural manner, and the technique extends trivially to 3 dimensions. At any time, the evolving contour can be obtained by extracting the zero level-set from the output.

3.4.1 Following section describes the basics of Level Set Function.

Concept of Zero set in a level Set function



Original Front:
Front lies in X-Y plane

Level Set Function:
Front is intersection of Surface And X-Y plane
(b)

Figure 3-4 Concept of Zero set in Level Set Function

Figure 3-4 (a) shows the circle, as original front and (b) shows its level set function in the form of cone surface. This cone surface has a great property: It intersects the X-Y plane exactly where the curve (shown in blue color) is. Entire black surface is called Level Set function, because it accepts as input any point in the plane and hands back its height as output. This level set function expands, rises, falls and does all the work. This method is also called as initial value formulation because initial position of the front gives initial data for time-dependent problem. That is, solution starts at a given position and evolves in time. The surface at zero-height (blue surface intersecting with X-Y plane) is called Zero Level Set.

Level Set techniques are powerful mathematical tools to deal with various applications in imaging and computer vision. The main advantage of using level sets is that arbitrarily complex shapes can be modeled and topological changes such as merging and splitting are handled

implicitly [18]. Level Set models are topologically flexible and can split and merge as necessary in the course of deformation without the need for re-parameterization. Using Osher and Sethian Level Set approach, complex curves can be detected and tracked and topological changes for the evolving curves are naturally managed. Level sets can be used for image segmentation by using image-based features such as mean intensity, gradient and edges in the governing differential equation [18]. In this method, a contour is initialized by a user and is then evolved until it fits the form of an anatomical structure in the image.

Chapter 4: Programs developed and used for thesis

This section describes the programs developed within this thesis and the ITK routines used within the programs. All programs were written in C and C++ compatible on Windows operating systems.

Configuration files (Cmakelists.txt) were written and programs compiled and built using CMake 2.0 and Microsoft Visual C++ 6.0. Typical inputs for these programs were 2-D brain slices with the following data characteristics:

Endian Type = Big Endian

Element Spacing = (X = 1.01563, Y = 1.01563)

Dimension Size = 256 * 256

Element Type = 16-bit Signed Short

4.1 Basic Input-Output

Original brain MRI slices, with details mentioned previously were converted into MetaImage format. This format uses an ASCII text MetaImage header file and a binary raw data file for each slice. This format was directly compatible with the ITK Input Reader (itk::ImageFileReader). Results (at intermediary times) were written to a file using the ITK Output Writer (itk::ImageFileWriter). Both ITK routines handle multiple file I/O file formats.

The actual low level task of reading and writing specific file formats was done by super class itk::ImageIO. Input images to the programs were specified with .mhd or .mha extension.

Figure 4-1 Collaboration Diagram for MetaImage Object Factory

MetaImageIOFactory was registered with objectFactoryBase. This allowed run-time instantiation of the classes that supported MetaImage format, based on the extension specified for an input file.

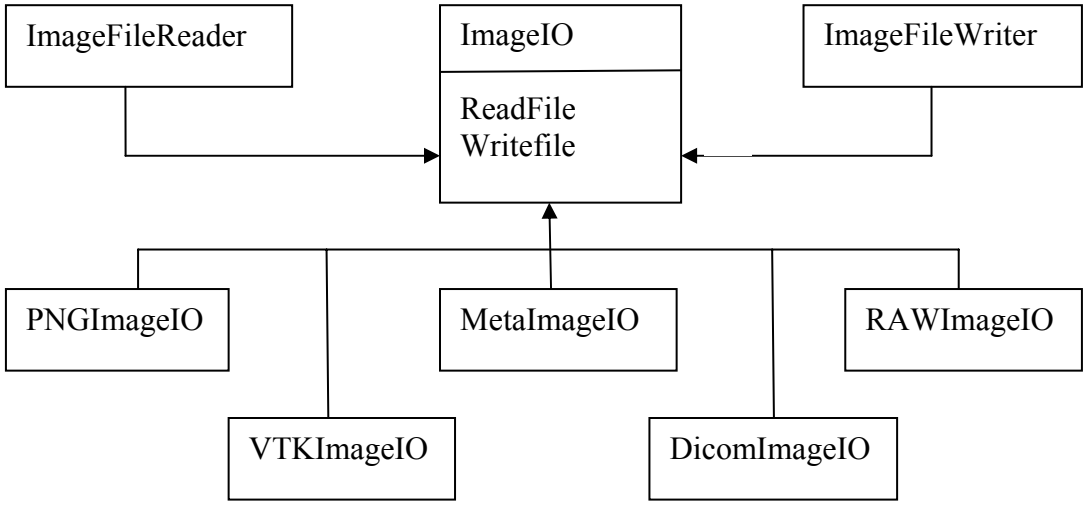


Figure 4-2 Collaboration Diagram of the ImageIO Classes

The IO architecture of the ITK makes it possible to avoid explicit specification of the file format used to read or write images. The object factory mechanism enables the ImageFileReader and ImageFileWriter to determine the file format at run-time. Typically, file formats are chosen based on the filename extension, but the architecture supports arbitrarily complex processes to determine whether a file can be read or written. Alternatively, the user can specify the data file format by explicit instantiation and assignment of the appropriate itk::ImageIO subclass. Following example file shows the header files associated with brain MRIs as input.

```
ObjectType = Image
NDims = 2
BinaryData = True
BinaryDataByteOrderMSB = True
Color = 1 0 0 1
ElementSpacing = 1.01563 1.01563
DimSize = 256 256
ElementType = MET_SHORT
ElementDataFile = tumour75.raw
```

Figure 4-3 Example MetaImage header file for tumor75.raw

Figure 4-4 represents the data pipeline for basic IO of brain MRIs.

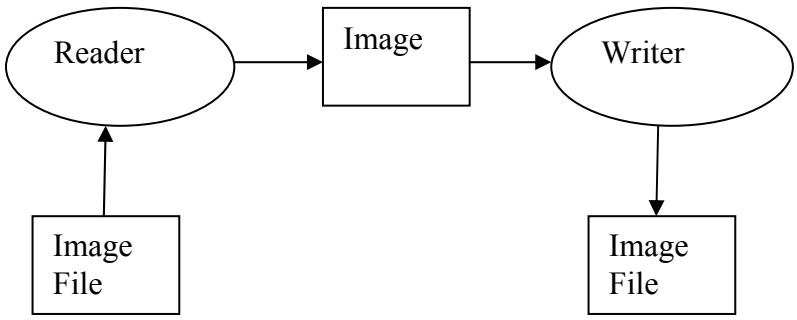


Figure 4-4 Data Pipeline for Basic IO

All output files were written as RAW files along with their MetaImage header files that consisted all the information as shown in figure 4-3.

4.2 Region-Growing Segmentation Program

This program used 3 region-growing algorithms implemented by ITK, which are:

- Connected Threshold
- Confidence Connected
- Neighborhood Connected

4.2.1 Connected Threshold

This program uses itk::ConnectedThresholdImageFilter. This filter uses the flood fill iterator. Most of the algorithmic complexity of a region growing method comes from visiting neighboring pixels. The flood fill iterator greatly simplifies the implementation of the region growing algorithm. The criterion used by the ConnectedThresholdImageFilter, to decide whether a particular pixel should be included in the current region or not, is based on an interval of intensity values provided by the user. The region growing algorithm includes those pixels whose intensities are inside the interval defined by [Upper, Lower].

Noise present in the image can reduce the capacity of this filter to grow large regions. Hence we pre-process the image by using an edge-preserving smoothing filter such as itk::CurvatureFlowImageFilter. Here, input images were read as pixel type as “float” due to the requirement of Curvature Flow Image Filter. This filter uses a level set formulation where the iso-intensity contours in a image are viewed as level sets, where pixels of a particular intensity form one level set. The level set function is then evolved under the control of a diffusion equation where the speed is proportional to the curvature of the contour:

$(I)t = K * (\Delta I)$, where K is the curvature.

Areas of high curvature will diffuse faster than areas of low curvature. Hence, small jagged noise artifacts will disappear quickly, while large scale interfaces will be slow to evolve, thereby preserving sharp boundaries between objects. However, although the evolution at the

boundary is slow, some diffusion still occurs. Thus, continual application of this curvature flow scheme will eventually result is the removal of information as each contour shrinks to a point and disappears.

The PDE update equation in discrete form is:

dI/dt = K * magnitude(gradient(I))

Where 'K' is the curvature of contours in the level set (isocontours in the image)

The continuous equation is then discretized as:

I[n+1] = I[n] + delta*(k[n]*magnitude(gradient(I[n]))

Here, I[n] is the image at n^{th} time step and delta is the user specified time-step to control stability of the solution.

'K' is curvature which is computed at each pixel from the current level set/image(I[n]) and not set externally.

Gradient of an image f(x,y) at location (x,y) is defined as vector

$\Delta f = [Gx,Gy] = [df/dx, df,dy]$.

- Gradient Vector points in the direction of maximum rate of change of f at co-ordinates (x,y)

- Magnitude of Gradient Vector $|\Delta f| = \sqrt{Gx\ 2\ +\ Gy\ 2}$

This quantity gives the maximum rate of increase of f(x,y) per unit distance in the direction of Delta(f)

- Direction of Gradient Vector is given by => $\tan^{-1}(Gy/Gx)$ where angle is measured w.r.t. X-axis)

Parameters used for smoothing:

TimeStep: TimeStep is used in the computation of level-set evolution. Typical values for the time step are 0.125 in 2-D images and 0.0625 in 3D images.

Number of iterations: Number of Iterations can be usually around 10; more iterations will result in further smoothing and will increase linearly the computing time. The edge-preserving is not an absolute on this filter, some degradation will occur on the edges and will accentuate as the number of iterations is increased. The number of iterations (for Segmentation after smoothing) is specified based on the homogeneity of the intensities of the anatomical structure to be segmented. Highly homogeneous regions may only require a couple of iterations. Regions with ramp effects, like MRI images with inhomogeneous fields, may require more iteration.

Parameters used for Connected Threshold filter

Inside Intensity: The intensity value to be set inside the region is selected as 255.

Seed Index: The location of seed index inside the region to be segmented.

Lower Threshold: Minimum value of threshold required for segmenting desired object

Upper Threshold: Minimum value of threshold required for segmenting desired object

Output: The output of this filter is a binary image (unsigned char type) with zero-value pixels everywhere except on the extracted region (which has all pixels with value 255).

4.2.2 Confidence Connected

Program uses itk::ConfidenceConnectedImageFilter. The criterion used by this filter is based on simple statistics of the current region. First, the algorithm computes the mean and standard deviation of intensity values for all the pixels currently included in the region. A user-provided factor is used to multiply the standard deviation and define a range around the mean. Neighbor pixels whose intensity values fall inside the range are accepted and included in the region. When no more neighbor pixels are found that satisfy the criterion, the algorithm is considered to have finished its first iteration. At that point, the mean and standard deviation of the intensity levels are recomputed using all the pixels currently included in the region. This mean and standard deviation defines a new intensity range that is used to visit current region neighbors

and evaluate whether their intensity falls inside the range. This iterative process is repeated until no more pixels are added or the maximum number of iterations is reached. The following equation illustrates the inclusion criterion used by this filter,

$$\mathbf{I(X) \in [m-f\sigma , m+f\sigma]}$$

Here, **m** and σ are the mean and standard deviation of the region intensities, **f** is a factor defined by the user, **I** is the image and **X** is the position of the particular neighbor pixel being considered for inclusion in the region.

Here also we pre-process the image by using an edge-preserving smoothing filter which reads input image as float type. Curvature Flow filter is described in detail in section 4.2.1

Parameters used for Confidence Connected filter

Inside Intensity: The intensity value to be set inside the region is selected as 255.

Seed Index: The initialization of the algorithm requires the user to provide a seed point. It is convenient to select this point to be placed in a typical region of the anatomical structure to be segmented. A small neighborhood around the seed point is used to compute the initial mean and standard deviation for the inclusion criterion.

Multiplier(f): Factor *f* defines how large the range of intensities will be. Small values of the multiplier will restrict the inclusion of pixels to those having very similar intensities to those in the current region. Larger values of the multiplier will relax the accepting condition and will result in more generous growth of the region. Values that are too large will cause the region to grow into neighboring regions that may actually belong to separate anatomical structures.

Number of Iterations: The number of iterations for this filter is specified based on the homogeneity of the intensities of the anatomical structure to be segmented. Highly homogeneous regions may only require a couple of iterations. Regions with ramp effects, like MRI images with inhomogeneous fields, may require more iteration. In practice, it seems to be more important to

carefully select the multiplier factor than the number of iterations. By letting the algorithm run for more iterations the region may possibly end up engulfing the entire image.

Output: The output of this filter is a binary image (unsigned char type) with zero-value pixels everywhere except on the extracted region (which has all pixels with value 255).

4.2.3 Neighborhood Connected

Program uses NeighborhoodConnectedImageFilter . This filter is a close variant of the itk::ConnectedThresholdImageFilter. The main difference between these two filters is that the Connected Threshold Image Filter accepts a pixel in the region if its intensity is in the interval defined by two user-provided threshold values. The NeighborhoodConnectedImageFilter, on the other hand, will only accept a pixel if all its neighbors have intensities that fit in the interval. The size of the neighborhood to be considered around each pixel is defined by a user-provided integer radius. The reason for considering the neighborhood intensities instead of only the current pixel intensity is that isolated pixels are less likely to be accepted in the region. This can be seen as a preemptive mathematical morphology operation that is similar to using the connectedThresholdImageFilter and then applying a combination of erosion and dilation with a structuring element of the same radius used for the neighborhood provided to the NeighborhoodConnectedImageFilter.

Parameters used for Neighborhood Connected Filter

Inside Intensity: The intensity value to be set inside the region is selected as 255.

Seed Index: The location of seed index inside the region to be segmented.

Lower Threshold: Minimum value of threshold required for segmenting desired object

Upper Threshold: Minimum value of threshold required for segmenting desired object

Setting these two threshold values too close will not allow enough flexibility for the region to grow. Setting them too far apart will result in a region that engulfs the image.

Neighborhood Radius: Neighborhood size is used to determine whether a pixel lies in the region. The larger the neighborhood, the more stable the filter against noise in the input image, and the longer the compute time. This radius parameter is user specified. A radius value of 2 along each dimension results in a neighborhood of (5*5) pixels. To set the radius of arbitrary size, radius in X and Y are entered separately.

Output: The output of this filter is a binary image (unsigned char type) with zero-value pixels everywhere except on the extracted region (which has all pixels with value 255).

4.3 Watershed Segmentation Program

Watershed segmentation is one of the most popular segmentation methods because it is non-parametric and computationally efficient [21]. As described in section 3.3, a typical approach for segmenting a gray-scale image with the watershed transformation is to make use of its gradient image as an input to the transformation since high gradients constitute watershed lines that correspond to the region boundaries of the gray-scale image. Contours generally correspond to crest lines of the gradient norm of the original image.

Since its use for segmentation by Lantuejoul in 1979, several definitions of the watershed have been proposed by different authors [19, 23, 24, 25]. There are two different algorithms commonly used to implement watersheds: top-down and bottom-up. The bottom-up strategy starts with seeds at the local minima in the image and grows regions outward and upward at discrete intensity levels. This approach is equivalent to a sequence of morphological operations and called morphological watersheds [26]. This limits the accuracy by enforcing a set of discrete gray levels on the image.

ITK has implemented top-down approach (i.e. a gradient descent strategy). This approach allows us to consider the output of multi-scale differential operators, and hence the function f in question will have floating point values [17].

Watershed algorithm implemented in ITK treats an image f as a height function, i.e., the surface formed by graphing f as a function of its independent parameters, $\vec{x} \in U$. The image f is often not the original input data, but is derived from that data through some filtering, graded (or fuzzy) feature extraction, or gradient filter.

Gradient descent associates regions with local minima of f (interior points) using the watersheds of the graph of f. That is, a segment consists of all points in *U* whose paths of steepest descent on the graph of f terminate at the same minimum in f. Thus, there are as many segments in an image as there are minima in f. In the 1-D case, the watershed boundaries are the local maxima of f, and the results of the watershed segmentation is trivial. For higher dimensional image domains, the watershed boundaries are not simply local phenomena; they depend on the shape of the entire watershed.

This program uses itk::WatershedImageFilter for segmentation. Input image is read as float type. Since watershed segmentation is sensitive to noise and high contrast, we obtain input for watershed filter by preprocessing the original image with an edge-preserving diffusion filter, such as anisotropic diffusion filter. The height function used as input should be created such that higher positive values correspond to object boundaries. A suitable height function for many applications is generated as the gradient magnitude of the image to be segmented. Hence, itk::GradientMagnitudeImageFilter is applied on the output of smoothing.

Parameters used by Watershed filter

Watershed Scale Level (Flood Level): Controls watershed depth

Threshold: Controls the lower thresholding of the input.

Both parameters are set as a percentage (0.0 - 1.0) of the maximum depth in the input image.

The drawback of watershed segmentation is that it produces a region for each local minimum. In practice there are too many regions and hence it causes an over segmentation as a

result. To alleviate this, we can establish a minimum watershed depth. The watershed depth is the difference in height between the watershed minimum and the lowest boundary point. That is, it is the maximum depth of water a region could hold without flowing into any of its neighbors. Thus, a watershed segmentation algorithm can sequentially combine watersheds whose depths fall below the minimum until all of the watersheds are of sufficient depth.

Following figure 4-5 shows the pipeline of filters used for this segmentation.

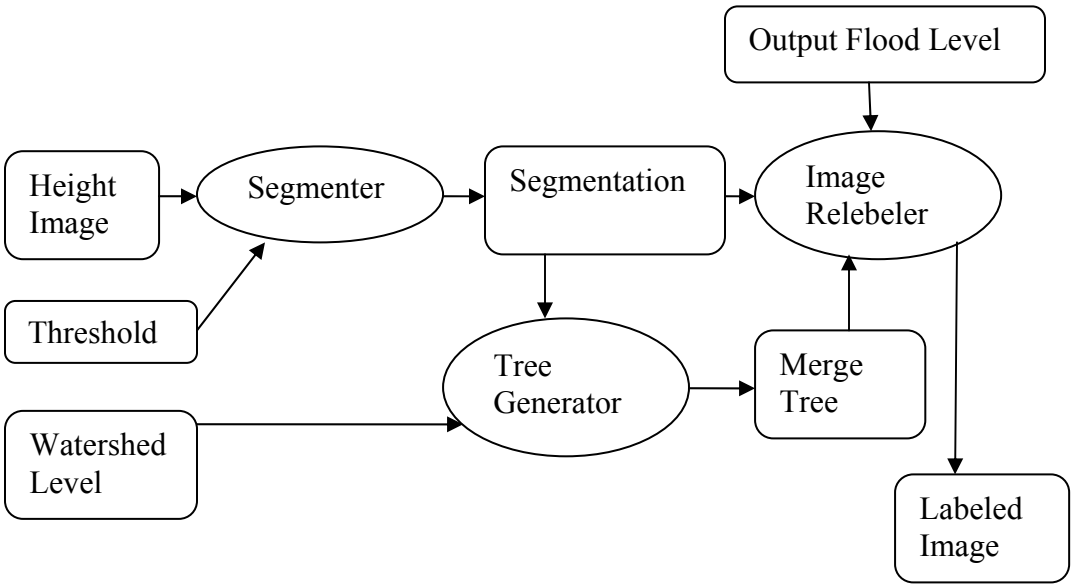
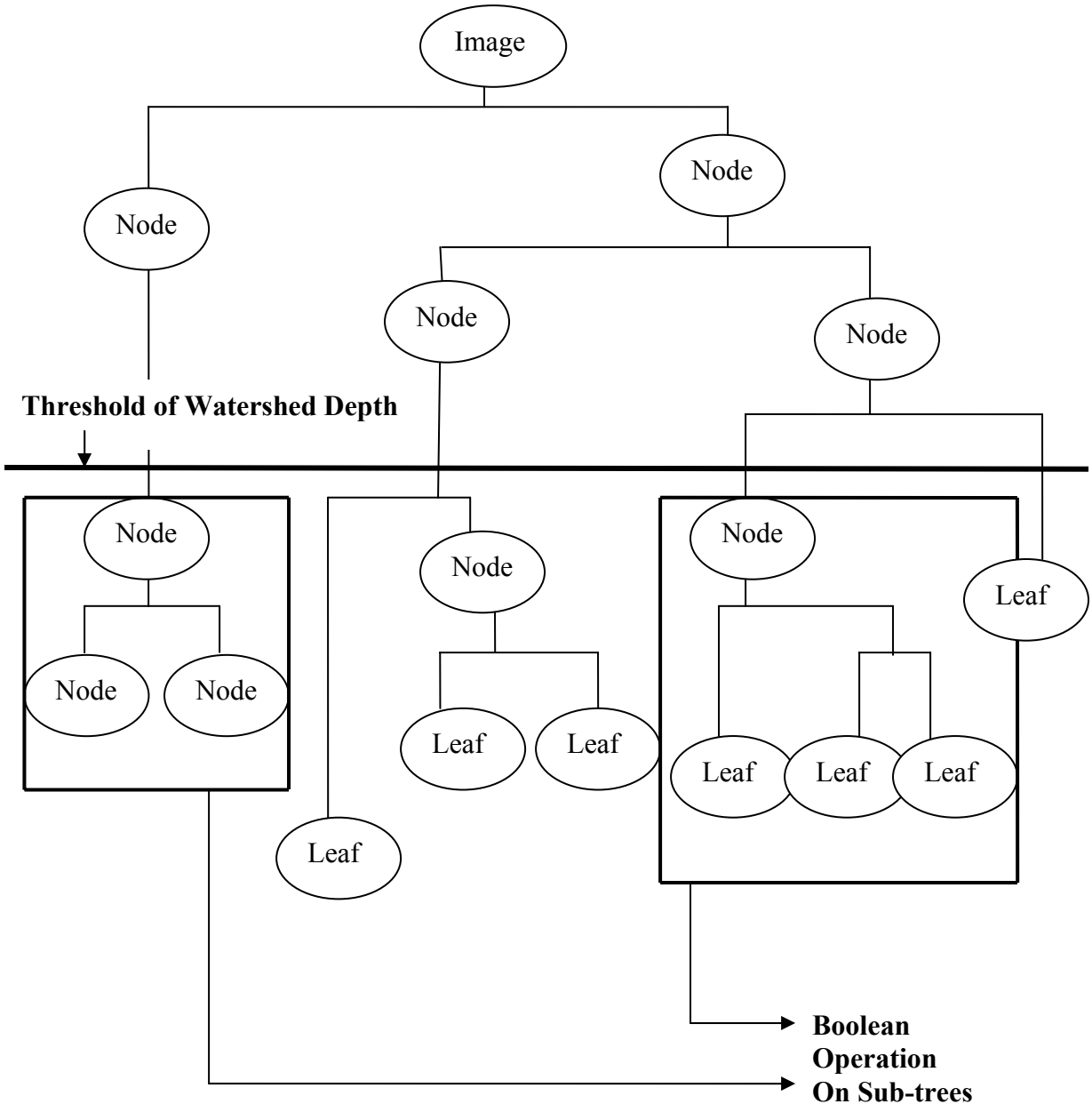


Figure 4-5 Parameters and Pipeline of filters used in Watershed Segmentation



Watershed Segmentation combined with Watershed Level creates hierarchy of regions. Structures are derived from images combining sub-tress with hierarchy or using threshold value (given as input by the user).

Figure 4-6 Hierarchy of regions in Watershed Segmentation

Threshold parameter is used to control over-segmentation of the image. Raising the threshold generally reduces computation time and produces output with fewer and larger regions. When tuning parameters, the scale level of the objects being segmented needs to be considered. The best trade-off between time and quality can be achieved by smoothing the image to remove noise and high contrast. Subsequent thresholding eliminates features just below the desired scale to avoid over-segmentation [17].

For smoothing, linear pre-filtering methods such as a Gaussian filter are not suitable for this application since they blur surfaces and edges equally. Instead, edge preserving techniques such as anisotropic diffusion and bilateral filtering can be used prior to applying watershed filter. These approaches smooth across surfaces while treating edges as outliers and thus preserving them [28],[29]. Anisotropic diffusion is an iterative procedure based on a nonlinear anisotropic version of the heat diffusion equations proposed by Perona and Malik [3]. ITK has implemented Anisotropic diffusion in itk:: AnisotropicDiffusionImagefilter that is used for the application.

The output of WatershedImageFilter is an image of unsigned long integer labels, where a label denotes membership of a pixel in a particular segmented region. Since this format is not practical for visualization, labeled output is converted to RGB pixels and output file is saved in PNG image format. The itk::ScalarToRGBPixelFunctor class is used to hash a scalar value into an itk::RGBPixel. This class is plugged into the itk::UnaryFunctorImageFilter to create an image filter for that converts scalar images to RGB images.

4.4 Level Set Segmentation Program

As explained in section 3.4, Level Set Segmentation is a numerical method for tracking the evolution of contours and surfaces by embedding the contour as the zero level set of a higher dimensional function called as level-set function $\Psi(X, t)$.

Level Set filters implemented in ITK make use of following **general level set equation:**

$$\frac{d}{dt}\psi = -\alpha \mathbf{A}(\mathbf{x}) \cdot \nabla\psi - \beta P(\mathbf{x}) \mid \nabla\psi \mid + \gamma Z(\mathbf{x}) \kappa \mid \nabla\psi \mid$$

Here, **A** is an advection term, **P** is a propagation (expansion) term, and **Z** is a spatial modifier term for the mean curvature k. The scalar constants **α**, **β** and **γ** weight the relative influence of each of the terms on the movement of the interface. Different level set segmentation filters use some or all of these terms in its calculations. If a specific term is not desired in the filter, then the corresponding scalar constant weighting is set to zero.

Most filters require two images as input, an initial model **Ψ(X, t = 0)**, and a feature image, which is either the image you wish to segment or some preprocessed version. You must specify the iso-value that represents the surface **Γ** in your initial model. The single image output of each level set filter is the function **Ψ** at the final time step. The contour representing the surface **Γ** is the zero level-set of the output image, and not the iso-value specified for the initial model. To represent **Ψ** using the original iso-value, we just simply add that value back to the output.

The solution **Γ** is calculated to sub pixel precision. The best discrete approximation of the surface is therefore the set of grid positions closest to the zero-crossings in the image, as shown in figure 4-7.

All of the level-set based segmentation filters operate with floating point precision to produce valid results.

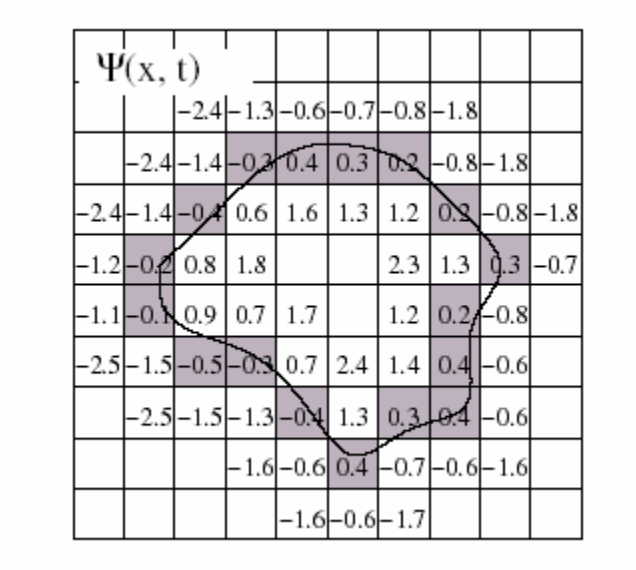


Figure 4-7 implicit level set surface Γ [27]

The implicit level set surface **Γ** is the black line superimposed over the image grid. The location of the surface is interpolated by the image pixel values. The grid pixels closest to the implicit surface are shown in gray.

In general, ITK level set filters implements a Level Set approach to the segmentation. A contour is represented in the form of a zero set of a function. The function evolves and carries with it the zero set. Evolution of the function is controlled by a partial differential equation in which a speed term is involved. This speed is computed based on an image provided by the user.

4.4.1 Introduction to Fast Marching Segmentation

The Fast Marching algorithm, introduced by Sethian (1996) is a numerical algorithm that is able to catch the viscosity solution of the Eikonal equation |grad(D)|=P. The level set {x | F(x) = t} can be seen as a front advancing with speed P(x). The resulting function D is a distance function, and if the speed P is constant, it can be seen as the distance function to a set of starting points.

The Fast Marching is very similar to the Dijkstra algorithm that finds shortest paths on graphs. Using a gradient descent of the distance function D, one is able to extract a good approximation of the shortest path (geodesic) in various settings (Euclidean for P constant and a weighted Riemannian manifold with P varying).

The central idea behind the Fast Marching Method is to systematically construct the solution in a downwind fashion to produce the solution *u*. Fast Marching Algorithm rests on solving following equation by building the solution outwards from the smallest *u* value.

$$\begin{bmatrix} \max(D_{ijk}^{-x}u, -D_{ijk}^{+x}u, 0)^2 \\ +\max(D_{ijk}^{-y}u, -D_{ijk}^{+y}u, 0)^2 \\ +\max(D_{ijk}^{-z}u, -D_{ijk}^{+z}u, 0)^2 \end{bmatrix}^{1/2} = F_{ijk},$$

This equation is the approximation to the following Eikonal equation

$$|\nabla u(x,y,z)| = F(x,y,z).$$

The solution is stepped outwards from the boundary condition in a downwind direction. The algorithm is made fast by confining the “building zone" to a narrow band around the front; this approach is motivated by the narrow band technology introduced by Chopp [30], used in recovering shapes in images by Malladi, Sethian, and Vemuri [31], and analyzed extensively by Adalsteinsson and Sethian in [29]. The idea is to sweep the front ahead in a downwind fashion by considering a set of points in a narrow band around the existing front, and to march this narrow band forward, freezing the values of existing points and bringing new ones into the narrow band structure. The key is in the selection of which grid point in the narrow band to update [32].

Consider a two-dimensional version of the Eikonal equation, in which the boundary value is known at the origin; this is shown schematically in Figure 4-8. The black sphere in (a) signifies a grid point where the value of u is known (in this case, the initial value), and the light gray spheres are grid points where the solution value is unknown. We may start the algorithm by marching downwind" from the known value, computing new values at each of the four

neighboring grid points, as shown in figure 4-9. This provides possible values for u at each grid point u(-1,0); u(1,0);

(0,-1); u(0,1). Now, we would like to march downwind from these values given at the dark gray spheres, but we do not know which one to choose. The answer lies in the observation that the smallest u value at these dark gray spheres must be correct. Because of upwinding, no point can be affected by grid points containing larger values of u. Thus, we may freeze the value of u at this smallest dark gray sphere,2 and proceed ahead with the algorithm; we can march the solution outwards, always selecting the narrow band grid point with minimum trial value for u and readjusting downwind neighbors. When a point is accepted, its neighbors are updated, and their u values may change. Thus, only a small subset of the structure must be reordered in order to regain the ordering.

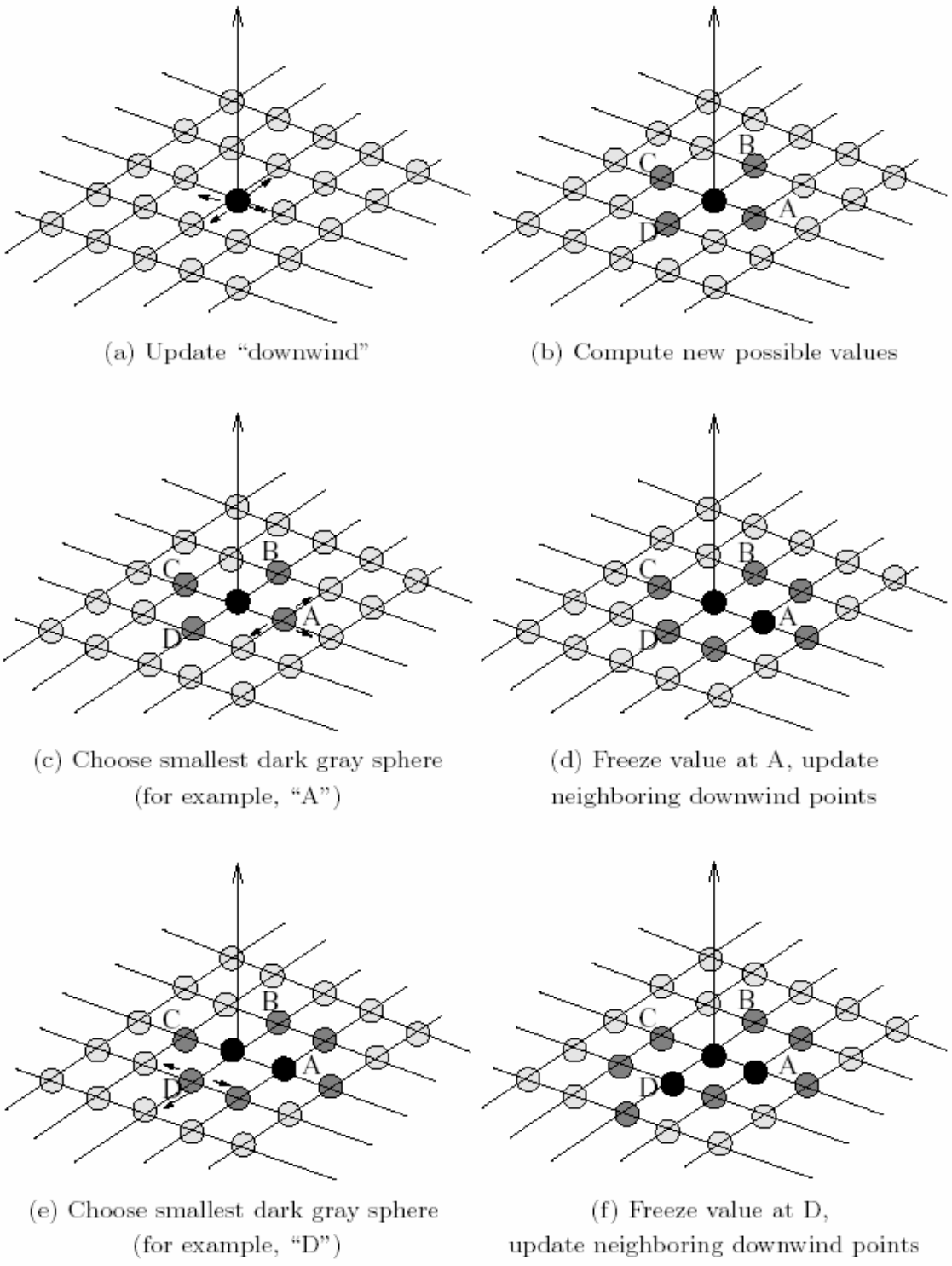


Figure 4-8 Beginning and update of Fast Marching method [32]

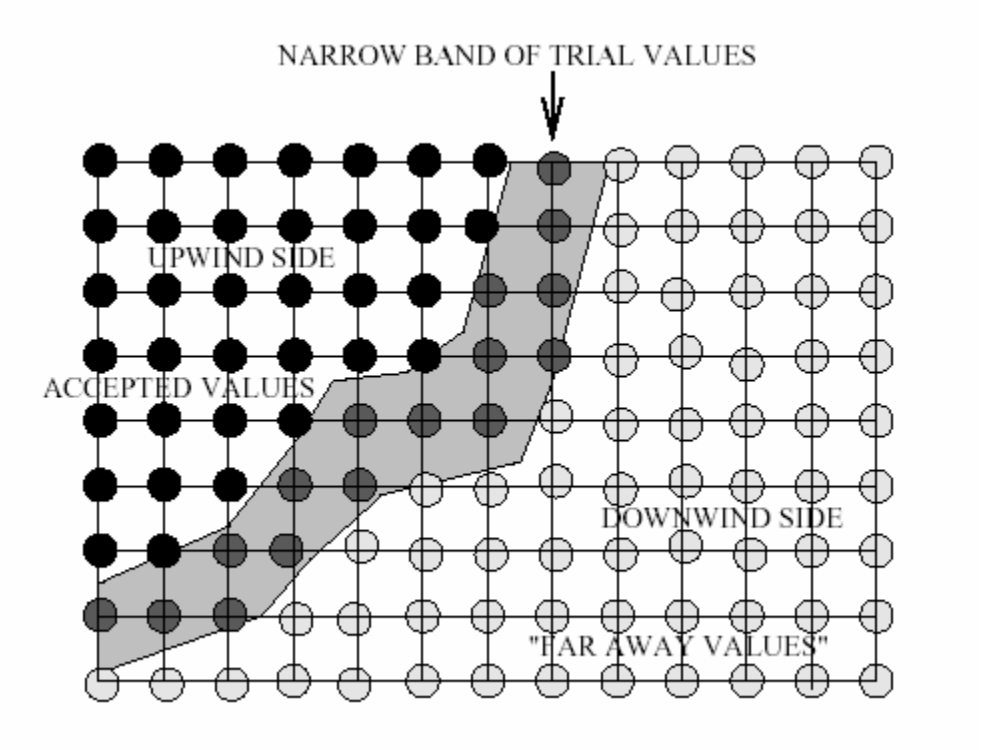


Figure 4-9 Upwind construction of accepted values [32]

4.4.1.1 Algorithm of Fast Marching Method [32]

Thus, the Fast Marching Method is as follows: First, tag points in the initial conditions as Alive. Then tag as Close all points one grid point away. Finally, tag as Far all other grid points. Then the loop is as follows:

1. Begin Loop: let Trial be the point in Close with the smallest value of u.

2. Tag as ‘Close’ all neighbors of Trial that are not ‘Alive’: If the neighbor is in Far, remove it from that list and add it to the set Close.

3. Re-compute the values of u at all Close neighbors of Trial by solving the piecewise quadratic equation according to the equation.

4. Add the point Trial to Alive; remove it from Close.

5. Return to top of Loop.

For the thesis itk::FastMarchingImageFilter filter is used to generate the model Ψ(X, t = 0) as a distance map generator. In this case, it does not require a speed image as input. Instead, we pass the constant value 1.0 using the SetSpeedConstant() method to this filter.

The output of the FastMarchingImageFilter is a time-crossing map that indicates, for each pixel, how much time it would take for the front to arrive at the pixel location. The threshold is applied on the output image to take a snapshot of the contour at a particular time during its evolution. This snapshot represents the initial model which evolved further by using variations of level set equations such as Shape Detection Level Set, Geodesic Level Set, Laplacian Level set etc.

4.4.2 Threshold Level Set Segmentation Program

This program uses itk::ThresholdSegmentationLevelSetImageFilter which is an extension of the threshold connected-component segmentation to the level set framework. The goal is to define a range of intensity values that classify the tissue type of interest and then base the propagation term on the level set equation for that intensity range.

The propagation term P from general level set equation is calculated from the Feature Image input *g* with Upper Threshold *U* and Lower Threshold *L* according to the following formula.

$$P(\mathbf{x}) = \begin{cases} g(\mathbf{x}) - L & \text{if } g(\mathbf{x}) < (U-L)/2 + L \\ U - g(\mathbf{x}) & \text{otherwise} \end{cases}$$

Intensity values in *g* between L and H yield positive values in P, while outside intensities yield negative values in P.

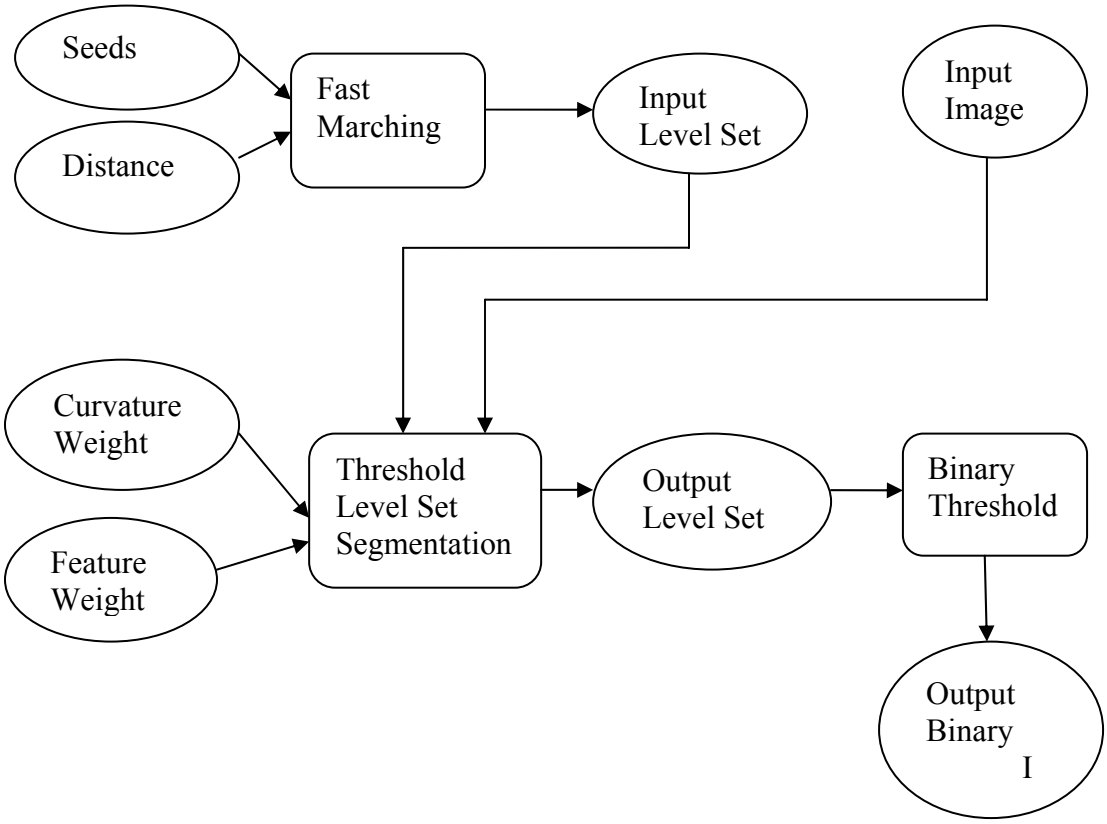


Figure 4-10 Collaboration diagram for the Threshold Level Set Segmentation

The threshold segmentation filter accepts two inputs. The first is an initial level set in the form of an itk::Image. The second input is the feature image *g*. Figure 4-10 shows how the image processing pipeline is constructed. The initial surface is generated using the fast marching filter.

Multiple seed points are first passed to the Fast marching filter. In this program, fast marching is used only to generate the initial level set and hence does not require speed image as input. Scaling parameters are used to balance the influence of the propagation (inflation) and the curvature (surface smoothing) terms from general level set equation. The advection term is not used in this filter. The values for upper threshold and lower threshold are accepted by the user.

The output of the segmentation filter is passed to a itk::BinaryThresholdImageFilter to create a binary representation of the segmented object. This output represents the zero set of the resulting level set. For obtaining this zero-level set, the upper threshold of the

BinaryThresholdImageFilter is set to 0.0 and the lower threshold is set to a large negative number to ensure that the interior of the segmented object will appear inside the binary region.

4.4.3 Shape Detection Level Set Segmentation Program

The implementation of this filter in ITK is based on the work of Malladi, Sethian and Vemuri [33]. In this implementation, the governing differential equation has an additional curvature-based term. This term acts as a smoothing term where areas of high curvature, assumed to be due to noise, are smoothed out. Scaling parameters are used to control the tradeoff between the expansion term and the smoothing term. In this level set function, contour is no longer guaranteed to be always be expanding. Instead, the level set function is updated iteratively.

The goal of this filter is to define a speed function from the image data that can be applied on the propagating front as a halting criterion. Speed function F is split as $F = F_A + F_G$.

F_G is the part that depends on local curvature. This term acts as a smoothing term. Thus evolution equation is given by

$$\psi_t + F_A |\nabla \psi| + F_G |\nabla \psi| = 0.$$

With a front propagating with a constant speed, negative speed F_I is defined as

$$F_I(x,y) = \frac{-F_A}{(M_1 - M_2)} \left\{ \left| \nabla G_\sigma * I(x,y) \right| - M_2 \right\}$$

Where M1 and M2 are maximum and minimum values of the magnitude of gradient of smoothed image and expression (Gσ * I) is the image convolved with Gaussian smoothing with sigma. When image gradient approaches the maximum M1 at the object boundaries, then the front gradually attains zero speed as it gets closer to the object boundaries and eventually comes to a stop. Other stopping criteria can also be used as a speed function. By updating the level set

function on a grid, level sets are moved without constructing them explicitly, by advancing from one time step to the next time step.

The ShapeDetectionLevelSetImageFilter expects two inputs, the first being an initial level set in the form of an Image, and the second being a feature/speed image. For this algorithm, the feature image is an edge potential image. A typical speed image is produced by a mapping of the gradient magnitude of the original image. The mapping is selected in such a way that regions with high contrast will have low speeds while homogeneous regions will have high speed. In this application the mapping of the gradient magnitude is made with a sigmoid function. This is provided by itk::SigmoidImageFilter. This is very convenient since the sigmoid offers parameters that can be tuned in order to select the features of interest from the input image. In order to initialize the Shape Detection filter, an input level set is required. It is desirable for this initial contour to be relatively close to the final edges of the anatomical structure to be segmented. For this program, FastMarchingImageFilter is used to produce the initial level set as the distance function to a set of user-provided seeds. The FastMarchingImageFilter is run with a constant speed value which enables us to employ this filter as a distance map calculator.

In order to initialize the Fast Marching filter, the user selects a set of seed points in the viewer of the input image. The fast marching filter will propagate a front starting from the seeds and traveling at constant speed until the front reaches a user-specified distance. At that point the front is passed to the Shape Detection filter which will use the partial differential equation in order to continue the evolution of the contour. The Shape Detection filter also uses a speed computed from the speed image. This results in the contour slowing down close to object edges. The final output of the filter is a level set in which the zero set represents the object borders. The number of iterations to run this filter is a critical parameter since too many iterations will always result in the contour leaking through the regions of low contrast of the anatomical object borders.

Once activated, the level set evolution will stop if the convergence criteria or the maximum number of iterations is reached. The convergence criteria are defined in terms of the

root mean squared (RMS) change in the level set function. The evolution is said to have converged if the RMS change is below a user-specified threshold.

Following collaboration diagram shows the different filters used in this program.

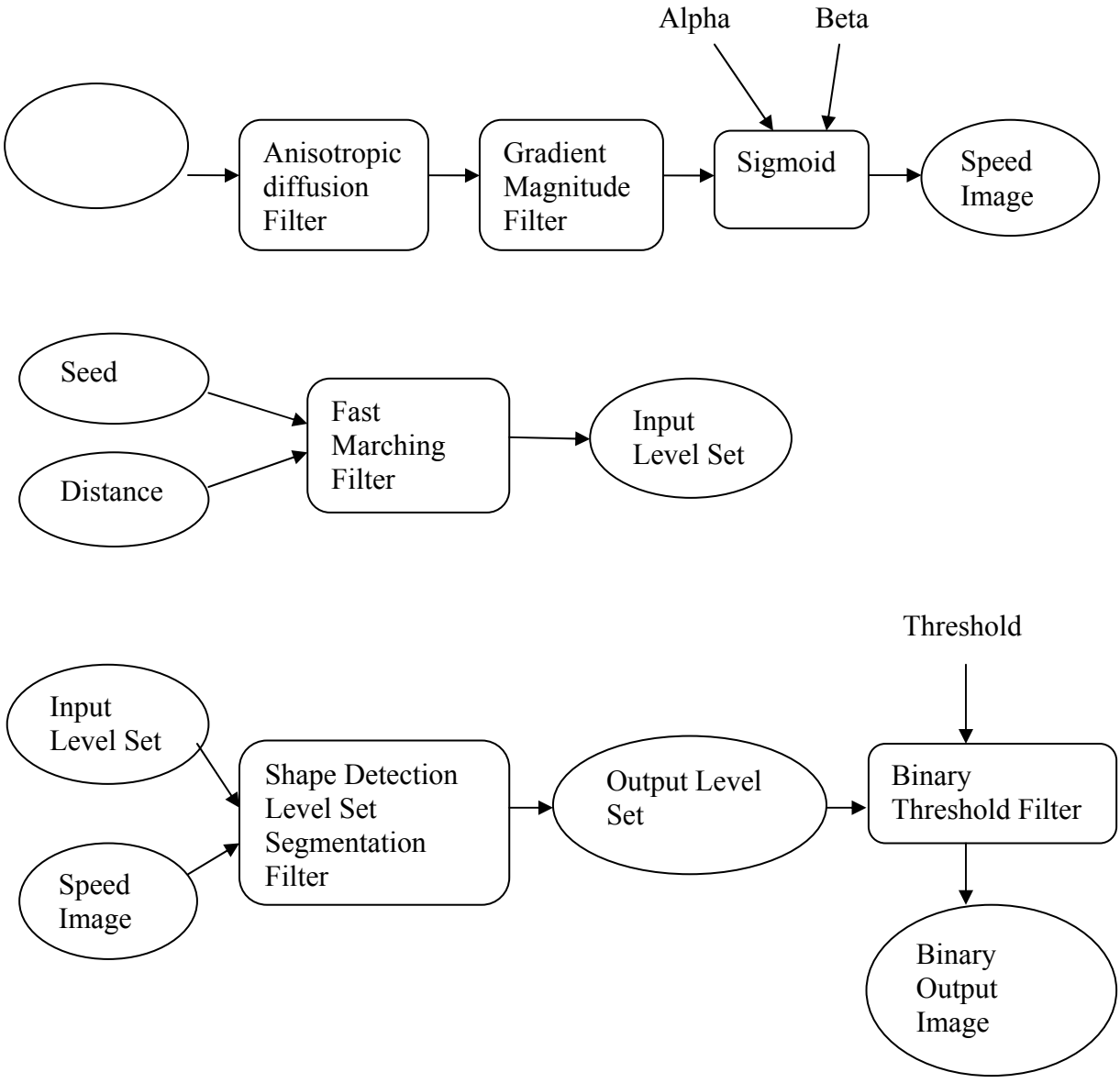


Figure 4-11 Collaboration diagram for Shape Detection Segmentation

4.4.4 Geodesic Level Set Segmentation Program

The implementation of this filter in ITK is based on the work of Kimmel, et. al. [34] The Geodesic Active Contour technique is based on active contours evolving in time according to intrinsic geometric measures of the image. This approach is based on the relation between active contours and the computation of geodesics (minimal distance curves) which lies in a Riemannian space. This geodesic approach for object segmentation allows to connect classical "snakes" based on energy minimization and geometric active contours based on the theory of curve evolution.

The classical "snake" approach is based on deforming an initial contour C_0 towards the boundary of the object to be detected. The deformation is obtained by trying to minimize a functional designed so that its (local) minimum is obtained at the boundary of the object. Geometric models (Level Set approaches) are based on the theory of curve evolution and geometric flows (PDE) based on mean curvature motion. In these type of active contours models, the curve is propagating (deforming) by means of a velocity that contains two terms as well, one related to the regularity of the curve (curvature term) controls the smoothness of the curve and the other (propagation term) shrinks or expands it towards the boundary.

While implicit active contour models avoid several of the difficulties known from explicit models such as inability of splitting and merging of objects, their main disadvantage is additional computational complexity. First, in their simplest implementation, the partial differential equation must be evaluated on the complete image domain. Second, most approaches are based on explicit updating schemes which demand very small time steps.

Geodesic Active Contour approach improves previous models of geometric (PDE) active contours allowing stable boundary detection when their gradients suffer from large variations, including gaps.

Following level set representation which is derived from energy functional constitutes the general Geodesic Active contour model proposed by Kimmel [34].

$$\frac{\partial u}{\partial t} = |\nabla u| \operatorname{div}\left(g(I) \frac{\nabla u}{|\nabla u|} \right) + c g(I) |\nabla u|.$$

The solution to the object detection problem is then given by the zero level-set of the steady state (u_t=0) of this flow. As described in the experimental results, it is possible to choose c = 0 (no constant velocity), and the model still converges (in a slower motion). This model has less parameter (with C = 0) as compared to level set equation.

Above geodesic flow includes a new component in the curve velocity that improves those models. The new velocity component allows us to accurately track boundaries with high variation in their gradient, including small gaps, a task that was difficult to accomplish with the previous curve evolution models.

ITK implementation of this approach extends the functionality of the Shape Detection Level Set Image Filter by the addition of a third advection term which attracts the level set to the object boundaries.

This filter requires two inputs. The first input is an initial level set. The initial level set is a real image which contains the initial contour surface as the zero level set. For example, a signed distance function from the initial contour surface is typically used. Unlike the simpler Shape Detection Level Set Image Filter the initial contour does not have to lie wholly within the shape to be segmented. The initial contour is allowed to overlap the shape boundary. The extra advection term in the update equation behaves like a doublet and attracts the contour to the boundary.

The second input is the feature image which is the edge potential map. General characteristics of an edge potential map are that it has values close to zero in regions near the edges and values close to one inside the shape itself. Typically, the edge potential map is computed from the image gradient

The Gradient Magnitude Recursive Gaussian Image Filter is used that performs the equivalent of a convolution with a Gaussian kernel, followed by a derivative operator. The sigma of this Gaussian is used to control the range of influence of the image edges. Output of this filter is given to Sigmoid Image Filter. This filter requires two parameters that define the linear transformation to be applied to the sigmoid argument. For this filter, we set Min and Max values to be 0.0 and 1.0 respectively in order to get a nice speed image to feed to the Fast Marching Image Filter. This arrangement will make the contour propagate until it reaches the edges of anatomical structures in the image and then slow down in front of those edges.

Fast Marching Image Filter is used here as a Distance Map generator, i.e. to generate initial level set. A set of user-provided seeds is passed to a Fast Marching Image Filter in order to compute the distance map. The fast marching filter will propagate a front starting from the seeds and traveling at constant speed until the front reaches a user-specified distance. At that point the front is passed to the Geodesic Active Contour filter which will use the partial differential equation in order to continue the evolution of the contour. A constant value is subtracted from the distance map generated by Fast Marching filter to obtain a level set in which the {zero set} represents the initial contour.

The Geodesic Active Contour filter also uses a speed computed from the speed image. This results in the contour slowing down close to object edges. The output of Geodesic Active contour Filter is passed to a Binary Threshold Image Filter to produce a binary mask representing the segmented object. This final output is a level set in which the zero set represents the object borders.

The number of iterations to run this filter is a critical parameter since too many iterations will always result in the contour leaking through the regions of low contrast of the anatomical object borders. Scaling parameters are used to trade off between the propagation (inflation), the curvature (smoothing) and the advection terms. Advection Image of this filter is a function of gradient. Geodesic Active Contour Level Set Function calculates Advection Image by computing the gradient of the feature (input) image using Recursive Gaussian Image Filter. To follow the implementation in Caselles et al paper, we set the Propagation Scaling to c (the inflation OR balloon force) and Advection Scaling and Curvature Scaling both to 1.0

Following collaboration diagram in figure 4-12 shows the different filters used in this program.

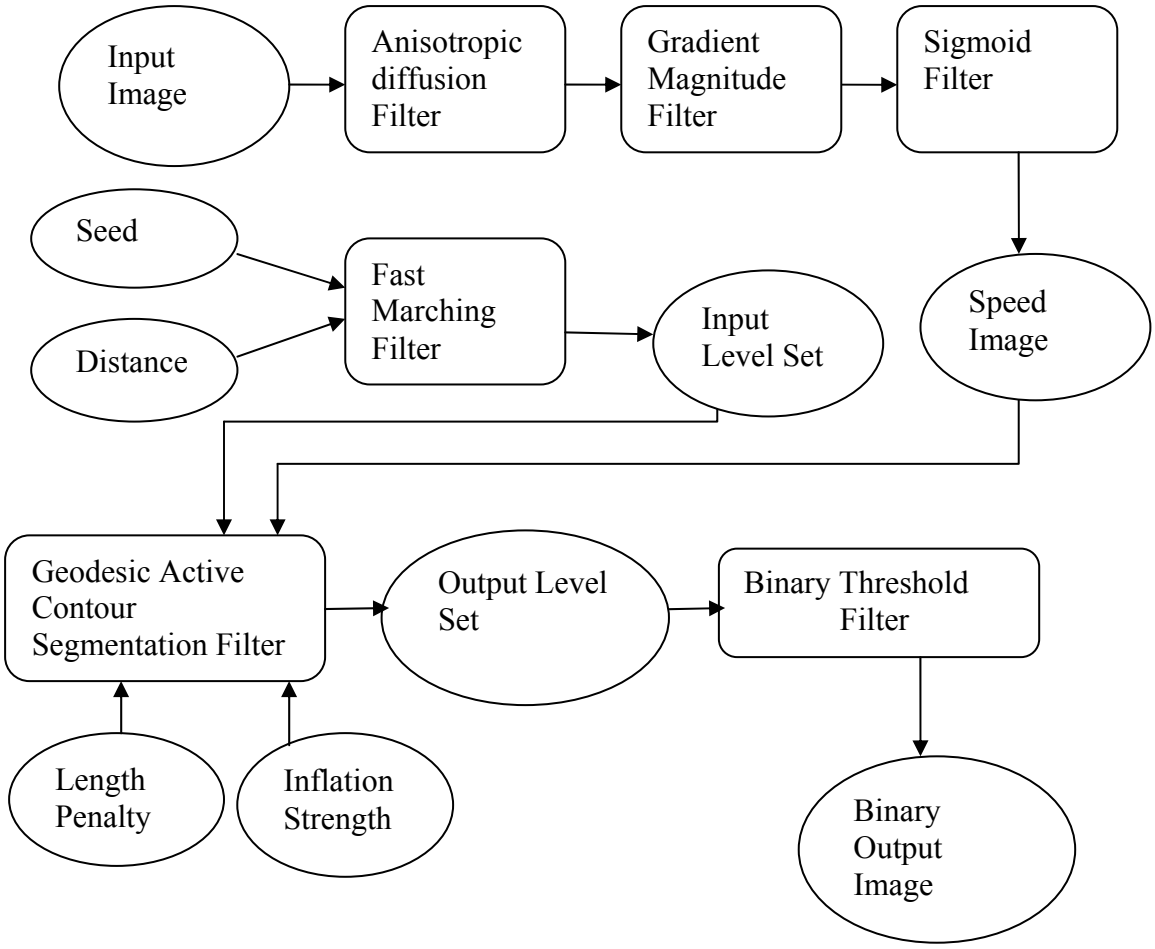


Figure 4-12 Collaboration diagram for Geodesic Active contour Segmentation

4.4.5 Laplacian Level Set Program

The itk::LaplacianSegmentationLevelSetImageFilter defines a speed term based on second derivative features in the image. The speed term is calculated as the Laplacian of the image values by applying the itk::LaplacianImageFilter to the input feature image.

This filter computes the Laplacian of a scalar-valued image. The Laplacian is an isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection.

The goal is to attract the evolving level set surface to local zero-crossings in the Laplacian image. This segmentation filter is more suitable for refining existing segmentations than as a region-growing algorithm. It can be used to perform region growing segmentation, but be aware that the growing surface may tend to become "stuck" at local edges.

Following collaboration diagram in figure 4-13 shows the different filters and their parameters used.

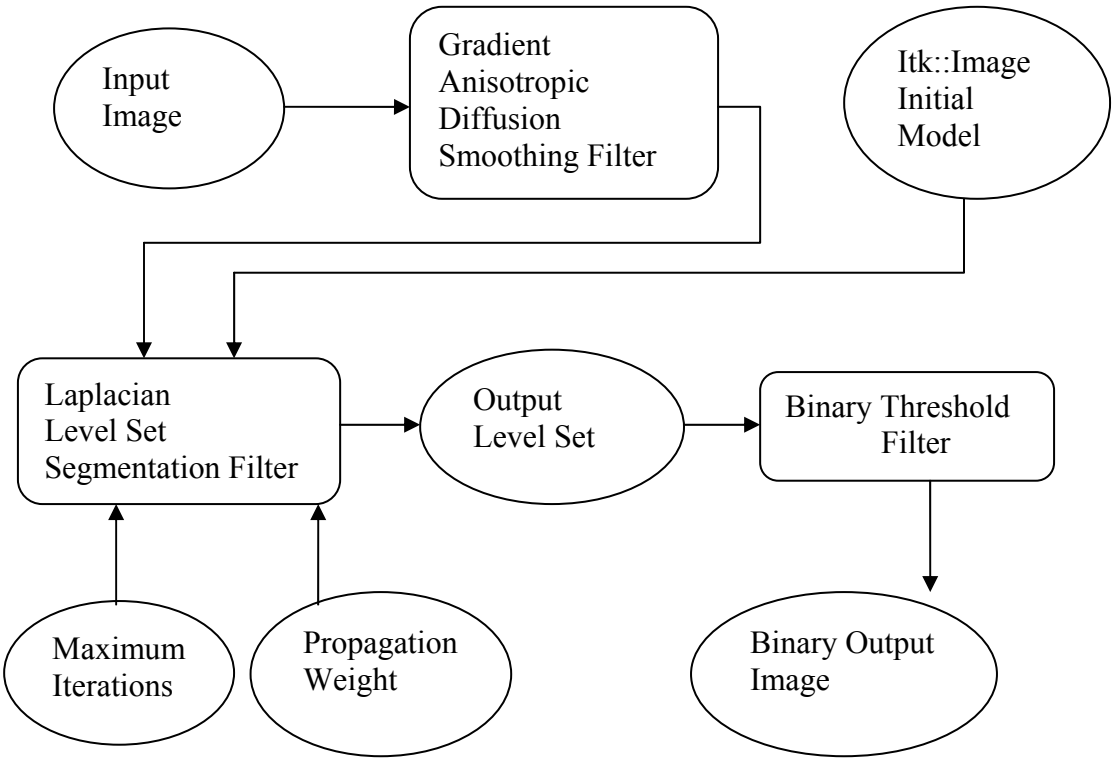


Figure 4-13 Collaboration diagram for Laplacian Level Set Segmentation

Since Laplacian filter performs second derivative, input image to be segmented is first smoothed in order to reduce its sensitive to noise. This smoothed image and initial model, i.e.

segmented image which is to be refined are given as input to the Laplacian Segmentation filter. Output of this filter is binary image which contains refined result of the initial model.

4.4.6 Canny Level Set Segmentation Program

The itk::CannySegmentationLevelSetImageFilter defines a speed term that minimizes distance to the Canny edges in an image. The initial level set model moves through a gradient advection field until it locks onto those edges. This filter is more suitable for refining existing segmentations than as a region-growing algorithm.

The two terms defined for the CannySegmentationLevelSetImageFilter are the advection term and the propagation term from Equation 9.3. The advection term is constructed by minimizing the squared distance transform from the canny edges.

$$\min \int D^2 \Rightarrow D\nabla D$$

4.4.6.1 Canny Edge-Detection

Canny edge detector is optimal for step edges corrupted by white noise. The idea has been described in a paper "*A Computational Approach to Edge Detection*", by Canny [35]. Canny listed following criteria to improve current methods of edge detection

The **detection** criterion expresses the fact that important edges should not be missed, and that there should be no spurious responses.

The **localization** criterion says that the distance between the actual and located position of the edge should be minimal.

The **one response** criterion minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

A last criterion was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

4.4.6.2 Algorithm: Canny edge detector

- Smooth the image with a Gaussian filter to reduce noise and unwanted details and textures.
- Compute gradient Magnitude using any of the gradient operators (such as Roberts, Sobel, Prewitt, etc)
- Estimate local edge normal directions for each pixel in the image.
- Find the location of the edges using non-maximal suppression. Non-maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.
- Compute the magnitude of the edge

Threshold edges in the image with hysteresis to eliminate spurious responses. Finally, hysteresis is used as a means of eliminating streaking: breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

Here, distance transform *D* is calculated using a itk::DanielssonDistanceMapImageFilter (which approximates distance transform to Euclidean distance) applied to the output of the itk::CannyEdgeDetectionImageFilter. DanielssonDistanceMapImageFilter uses the algorithm which is the N-dimensional version of the 4SED algorithm given for two dimensions in: **Paper:** Danielsson, Per-Erik. Euclidean Distance Mapping. Computer Graphics and Image Processing 14, 227-248 (1980).)

For cases in which some surface expansion is to be allowed, a non-zero value may be set for the propagation term. The propagation term is simply *D*. As with all ITK level set segmentation filters, the curvature term controls the smoothness of the surface.

Figure 4-14 shows collaboration diagram for Canny Level Set Filter

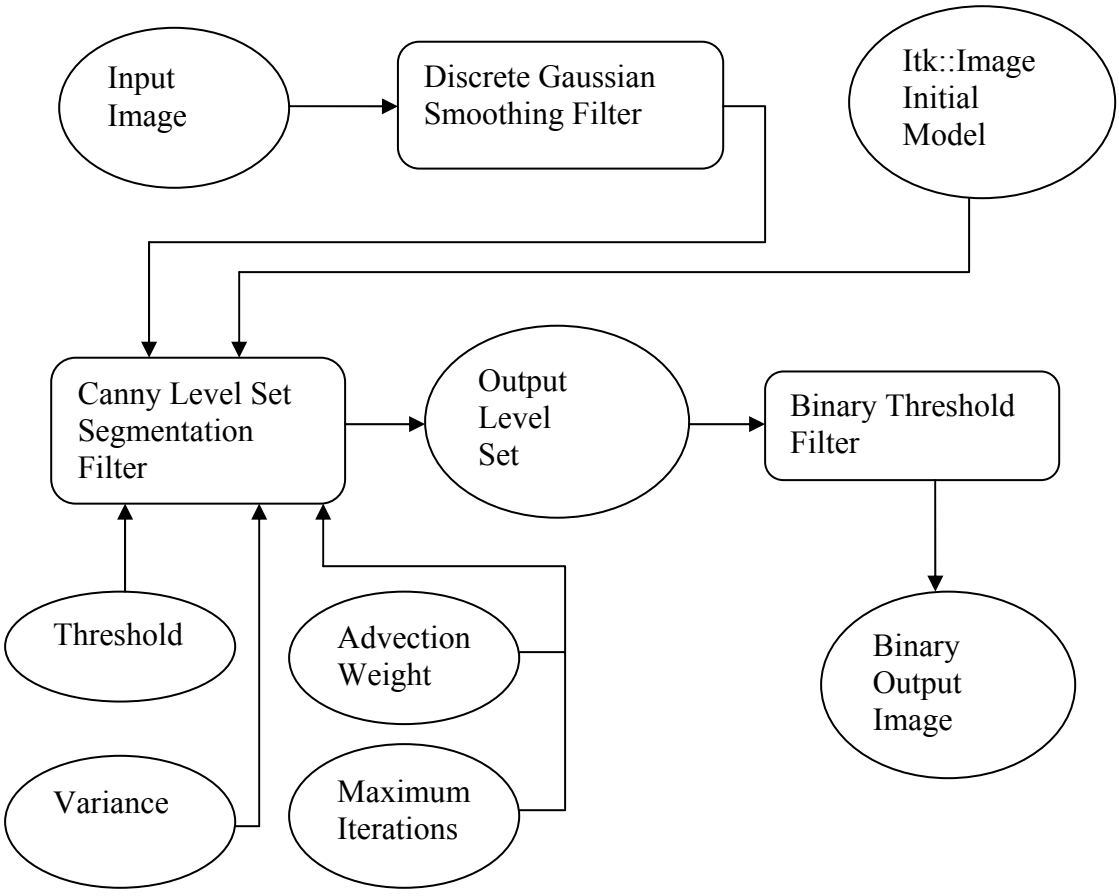


Figure 4-14 Collaboration diagram for Canny Edge Detection segmentation

CannySegmentationLevelSetImageFilter expects two inputs. The first is an initial level set in the form of an itk::Image. The second input is the feature image *g* from which propagation and advection terms are calculated. It is generally a good idea to do some preprocessing of the feature image to remove noise.

Filter reads two images: the image to segment and the image that contains the initial implicit surface. The goal is to refine the initial model from the second input and not to grow a new segmentation from seed points. The feature image is preprocessed with a few iterations of an anisotropic diffusion filter.

There are two important parameters in the CannySegmentationLevelSetImageFilter to control the behavior of the canny edge detection. The variance parameter controls the amount of Gaussian smoothing on the input image. The threshold parameter indicates the lowest allowed value in the output image. Thresholding is used to suppress canny edges whose gradient magnitudes fall below a certain value. Isovalue of the surface is specified in the initial model input image. In a binary image, for example, the isosurface is found midway between the foreground and background values. The free parameters of this filter can be adjusted to achieve a wide range of shape variations from the original model.

Chapter 5: Results and Discussion

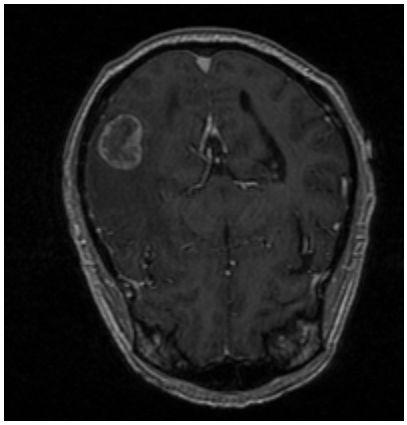
5.1 Results of Region Growing Segmentation

5.1.1 Connected Threshold

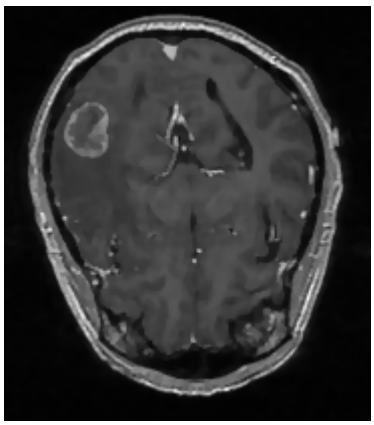
Following section presents the results of applying Connected Threshold Segmentation on Brain MRI to segment tumor, ventricles and white matter.

5.1.1.1 Segmenting Tumor

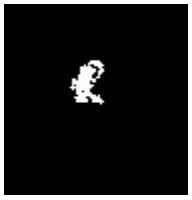
In following figures, image in figure 5-1 (a) represents original brain MRI. Figure 5-1 (b) shows the result of applying curvature flow smoothing with Time Step = 0.125, Number of Iterations = 5. In second row figure 5-1 (c) was obtained by applying Connected Threshold Segmentation on smoothed image with Lower Threshold = 169, Upper Threshold = 182 and Seed Index (76, 91). Second result in figure 5-1 (d) was obtained by applying Curvature Flow Smoothing twice, with Time Step = 0.125 each time and Number of Iterations = 5 and 10 respectively. Then Connected Threshold Segmentation was applied with Lower Threshold = 169, Upper Threshold = 182. Dilation was applied to this above result as shown in figure 5-1 (e). Figure 5-1 (f) was obtained by applying Edge preserving smoothing (Curvature Anisotropic diffusion with timestep = 0.125, conductance = 1, number of iterations = 5) and then applying Connected Threshold to this smoothed result with lower threshold = 50, upper threshold = 90 and Seed Index (76, 91). Figure 5-1 (g) shows the result of applying dilation to the segmented result in figure 5-1 (f).



(a)



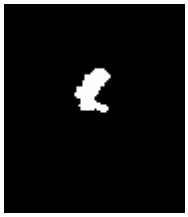
(b)



(c)

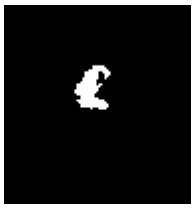
(d)

(e)

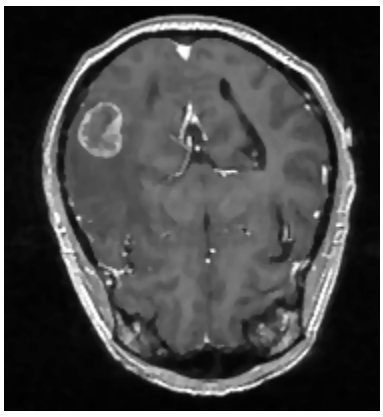


(f)

(g)



(i)



(h)

Figure 5-1 (a) – (h) Segmenting tumor with connected threshold filter

Now we apply Normalization to the image smoothed with curvature anisotropic diffusion, as shown in Figure 5-1 (h). But as seen from result figure 5-1 (i), normalization does not contribute to any significant improvement in the result of connected Threshold Segmentation. Hence, hereafter, we have used Normalization in some cases for display purpose only.

As we can see here, segmented result does not cross the external boundary of the tumor. Gray matter is not being completely segmented. This illustrates the vulnerability of the region growing methods when the anatomical structures to be segmented do not have a homogeneous statistical distribution over the image space.

One solution will be to apply level set segmentation to the above result. This result can be dilated so as to cross the internal boundary of the tumor, bring it close enough to the external boundary of the tumor and then apply the level set segmentation such as laplacian or canny edge detection to refine the result. Other level set segmentation algorithms such as shape detection, geodesic active contour may be applied to obtain the desired segmentation.

5.1.1.2 Segmenting brain (white matter)

Figure 5-2 (a) shows image smoothed original brain MRI of 256*256.. Figure 5-2 (b) shows brain area segmented with Connected Threshold with lower threshold = 45 and upper threshold = 55 and Seed Index (160,115). Here, smoothing was not applied.

As seen in the results, this filter produces the output with holes. There are different approaches that we can use to fill above holes:

1. Use dilation operation.

2. We can apply flood fill operation using itk::GrayscaleFillholeImageFilter

3. If size is enough for characterizing the holes we can use the connected components filter (itk::ConnectedComponentImageFilter) and then collect the number of pixels per component in order to identify the labeled regions those need to be set ON to fill the holes.

4. We can use the ConnectedThresholdImageFilter and use the first pixel of the image as seed point (assuming it is in the exterior of the shape) and segment the "exterior" of brain area.

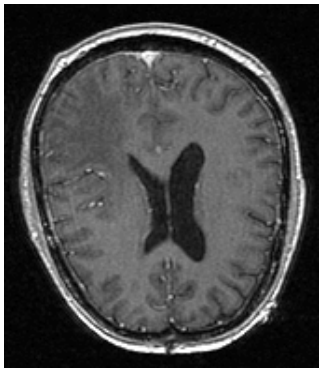
Then make a logical OR between the exterior and the shape, which will give the negative map of the holes. Negating this map and do an OR between the map and the original shape, you will end up with the shape without holes.

Figure 5-2 (c) shows the result after applying dilation with X-radius=1, Y-radius=1. Here, most of the holes disappear, but still some holes are left and it changes the segmentation result Figure 5-2 (d) shows the result of applying GrayScaleFillHoleImagefilter to 5-2 (b). With this filter, all holes are filled into ON region, including ventricles area.

Figure 5-2 (e) shows the result of Connected Threshold segmentation with lower threshold = 39 and upper threshold = 56 and figure 5-2 (f) is the result of applying GrayScaleFillHole filter to the result in figure 5-2 (e).

In all above cases, smoothing was not used. Noise present in the image can reduce the capacity of this filter to grow large regions. When faced with noisy images, it is usually

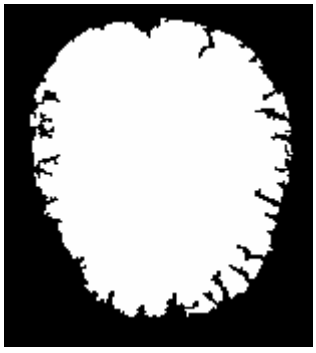
convenient to pre-process the image by using edge-preserving smoothing filters such as Curvature Anisotropic Diffusion Filter, Gradient Anisotropic Diffusion Filter.



(a)

(b)

(c)

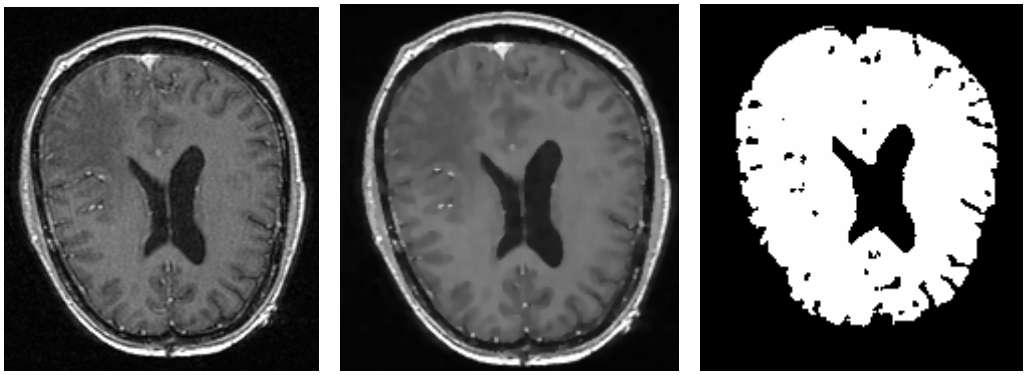


(d)

(e)

(f)

Following are the results of applying smoothing and then performing segmentation.



(g) (h) (i)

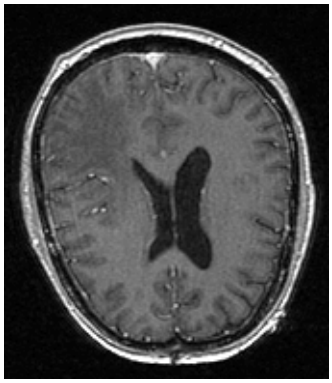
(k)

Figure 5-2 (a) - (k) Segmenting brain with connected threshold filter with smoothing

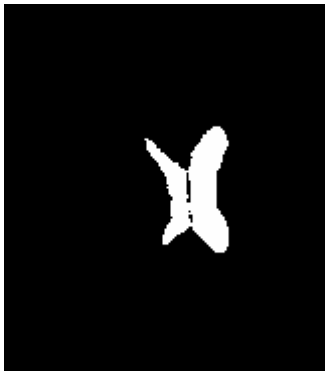
Here, figure 5-2 (g) represents original image, figure 5-2 (h) represents image smoothed with curvature anisotropic Diffusion with No. of Iterations = 5, Time step = 0.125 and Conductance = 1. Figure 5-2 (i) is the image segmented with Connected Threshold filter with Seed Index (160, 115) and Lower Threshold = 80 and Upper Threshold = 125. This result can be improved with GrayScaleFillHole filter as shown in figure 5-2 (k)

5.1.1.3 Segmenting Ventricles

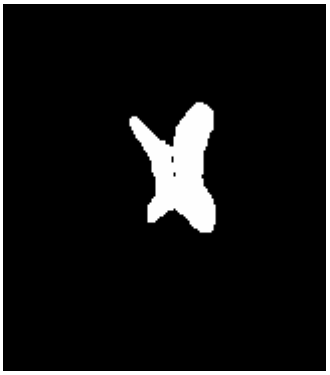
Figure 5-3 (a) shows the original brain MRI slice of 256*256. Figure 5-3 (b) shows the result of applying Connected Threshold to original image, without using smoothing. Here, seed point (138,138) was used and lower threshold = 1, upper threshold = 30. Figure 5-3 (c) shows the result of applying dilation to (b)



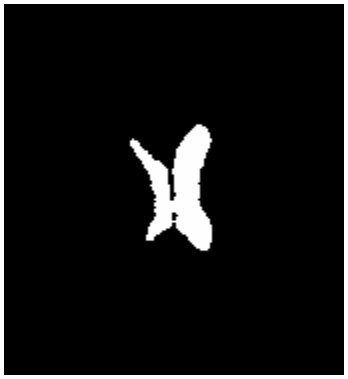
(a)



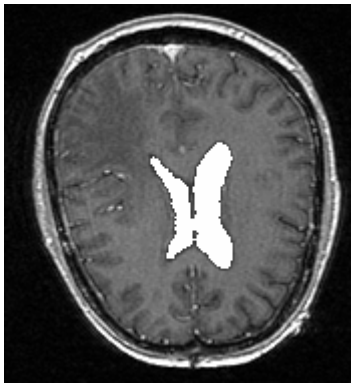
(b)



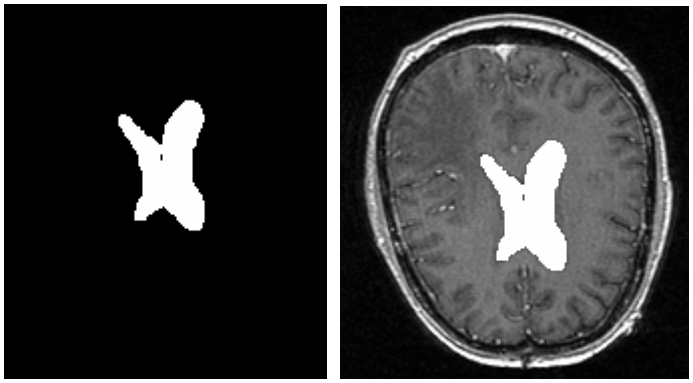
(c)



(d)



(e)



(f) (g)

Figure 5-3 (a)-(g) Segmenting ventricles with connected threshold filter

Figure 5-3 (d) shows the result of applying Connected Threshold to an image smoothed with Curvature Anisotropic Diffusion. Here, seed point (138,138) was used and lower threshold = 0, upper threshold = 30. As we can see from this result, smoothing does not change the result significantly here. This result can also be dilated with X-radius= 1 and Y-radius=1. Figure 5-3 (e) shows the result to match segmentation result with original image. Figures 5-3 (f) and 5-3 (g) show the dilated result with X-radius=1, Y-radius=1.

Thus smoothing is not needed for segmenting ventricles. With or without smoothing, output of connected threshold contains jagged edges which can be improved with the use of dilation. Here dilation also helps to achieve the ventricles segmented to their actual size in original image, as shown in figure 5-3 (g).

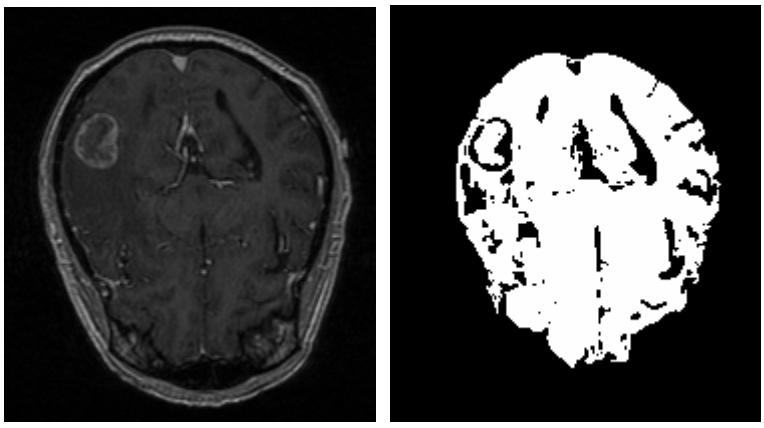
5.1.2 Confidence connected

5.1.2.1 Segmenting Tumor

Figure 5-4 (a) shows original 256*256 slice of brain MRI. Result in figure 5-4 (c) is obtained by first applying Curvature Flow Smoothing with Time Step = 0.125 and Iterations = 10 and Confidence Connected Filter is applied with Seed Index (80, 92), Multiplier = 2.4 and Iterations = 5. Here, changing Multiplier from 2.4 to 2.5 and keeping rest of the parameters same

changes the result slightly as shown in figure 5-4 (d) while changing the Number of Iterations = 10 causes the over-segmentation as shown in figure (b). Figure 5-4 (e) shows the result of Dilation with X-radius=3, Y-radius=1, on the result in figure 5-4 (d).

From above results it is clear that Confidence Connected Region Grow Segmentation is not helpful to delineate tumor fully. Further segmentation can be obtained by using Level Set Segmentation on these results.

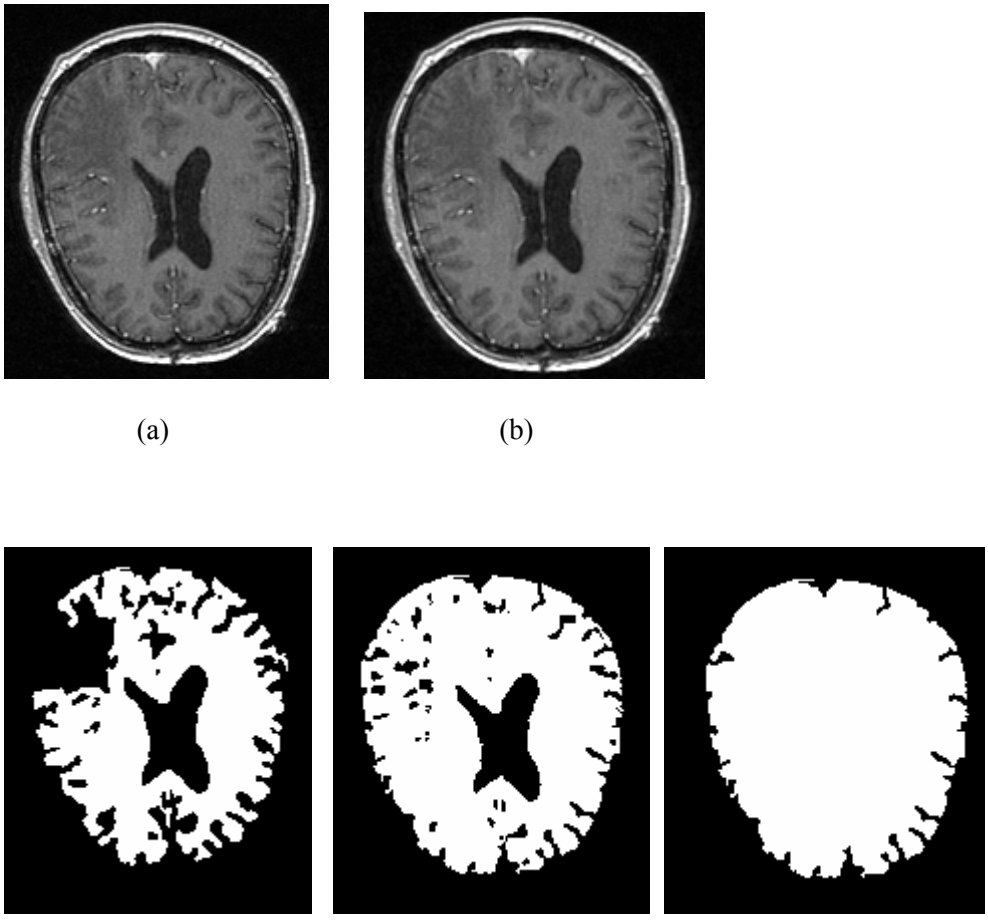


(a) (b)

(c) (d) (e)

Figure 5-4 (a) - (e) Segmenting tumor with confidence connected filter

5.1.2.2 Segmenting brain



(a) (b)

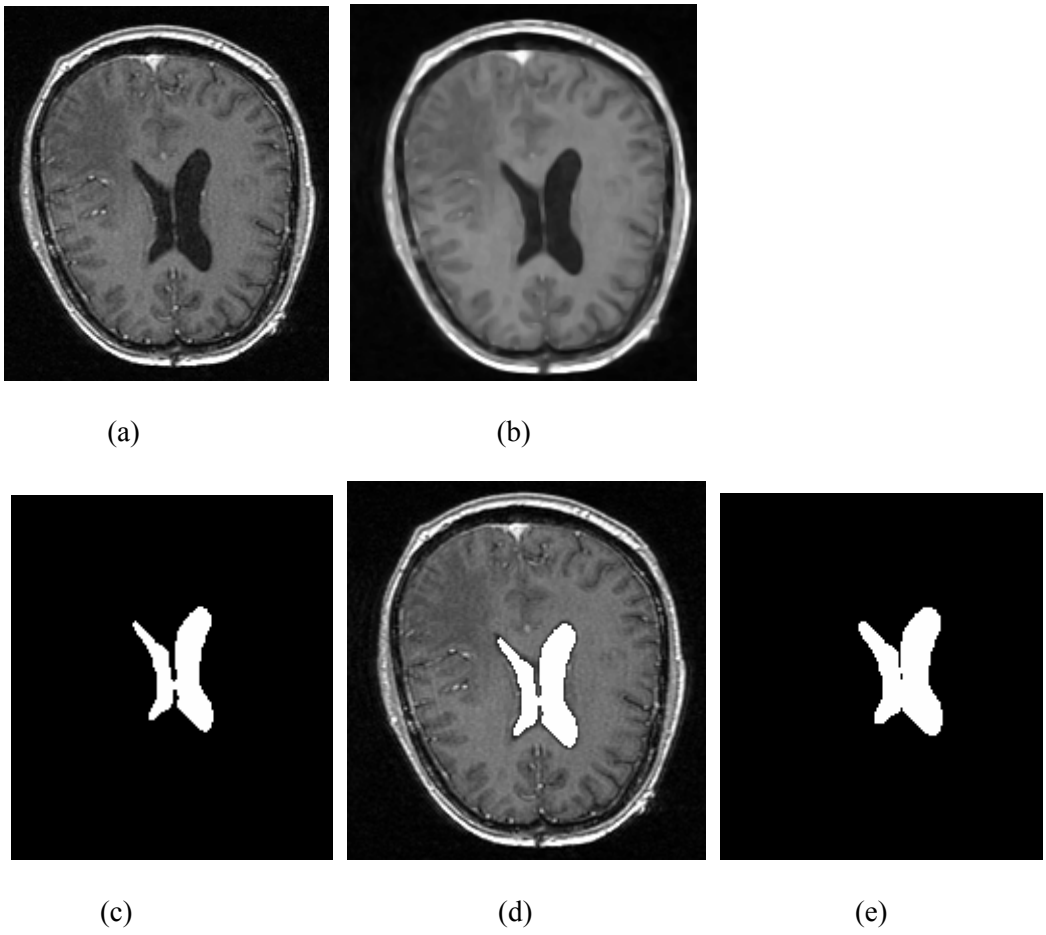
(c) (d) (e)

Figure 5-5 (a) - (e) Segmenting brain with confidence connected filter

Figure 5-5 (a) shows original 256*256 brain MRI slice in axial plane. Figure 5-5 (b) shows result of applying edge-preserving Curvature Anisotropic Diffusion Smoothing on original image with Time Step = 0.125, No. of Iterations = 5 and Conductance = 1.

Figure 5-5 (c) shows the result of applying Confidence Connected Segmentation with Multiplier = 3.2 and Number of Iterations = 10 and Seed Index (160, 155). To increase the segmentation, we increase multiplier to 3.5 and keep Number of Iterations = 10. The result of this is shown in figure 5-5 (d). This result can be improved with GrayScaleFillHole filter which removes the holes with TimeStep = 0.125 and Iterations =1 as shown in figure 5-5 (e).

5.1.2.3 Segmenting Ventricles



(a) (b)

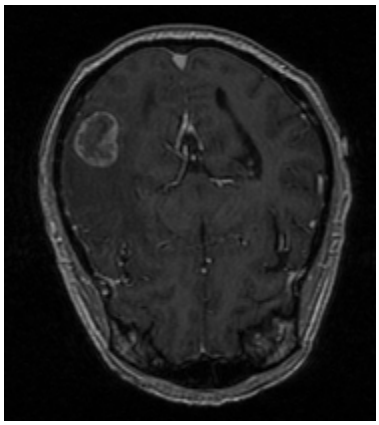
(c) (d) (e)

Figure 5-6 (a) – (e) Segmenting ventricles with confidence connected filter

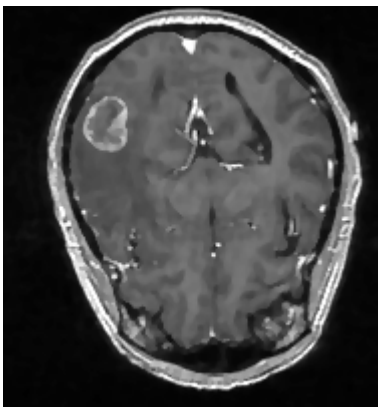
Figure 5-6 (a) shows original 256*256 brain MRI slice in axial plane. Figure 5-6 (b) shows result of applying Curvature Flow Smoothing on original image with Time Step = 0.125, Iterations = 10. Figure5-6 (c) shows the result of applying Confidence Connected filter with Multiplier = 3.0, No. of Iterations = 5 and Seed Index (138,138) on smoothed image. Above result can be further dilated as (X-radius=1, Y-radius=1) shown in figure 5-6 (e).

5.1.3 Neighborhood Connected

5.1.3.1 Segmenting Tumor



(a)

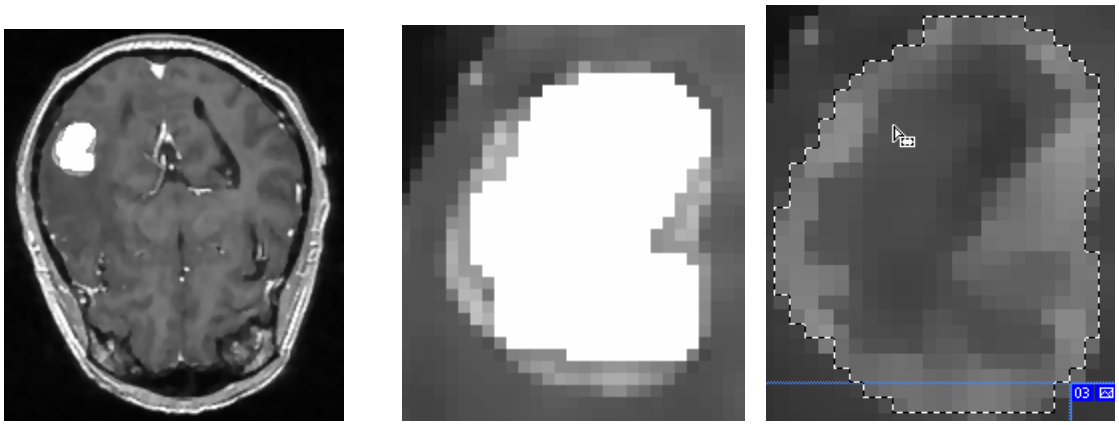


(b)

(c)

(d)

(e)



(f) (g) (h)

Figure 5-7 (a) – (h) Segmenting tumor with neighborhood connected filter

Figure 5-7 (a) shows original 256 * 256 MRI scan while figure 5-7 (b) shows original images smoothed with Curvature Anisotropic Diffusion (Timestep = 0.125, Iterations = 10 and Conductance =1). Figure 5-7 (c) shows the result of Neighborhood Connected segmentation with lower threshold = 33, upper threshold = 114, Neighborhood Radius (1, 1) and Seed Index (80, 91). Here neighborhood radius is the crucial parameter that defines the neighborhood size used to determine whether a pixel lies in the region. The larger the neighborhood, the more stable this filter will be against noise in the input image, but also the longer the computing time will be. A radius of 1 along each dimension results in a neighborhood of 3*3 pixels.

Figure 5-7 (d) shows the segmented result with lower threshold = 33, upper threshold = 113, Neighborhood Radius (1, 1) and Seed Index (80, 91).

Above result in figure 5-7 (d) can be improved by dilation as shown in figure 5-7 (e). In this result, segmented tumor still does not match with the tumor in the original image. This is because, as seen in figure 5-7 (c), tumor does not contain homogeneous intensity and it is difficult to segment such objects with intensity based segmentation method as this one.

Although we have applied this filter on smoothed image with edges preserved, it may not require any initial filtering to smooth the image since this filter is more resistant to the presence

of noise in the input image. Also, this filter considers neighborhood intensities instead of only the current pixel intensity, hence isolated pixels are less likely to be accepted in the region.

5.1.3.2 Segmenting Brain

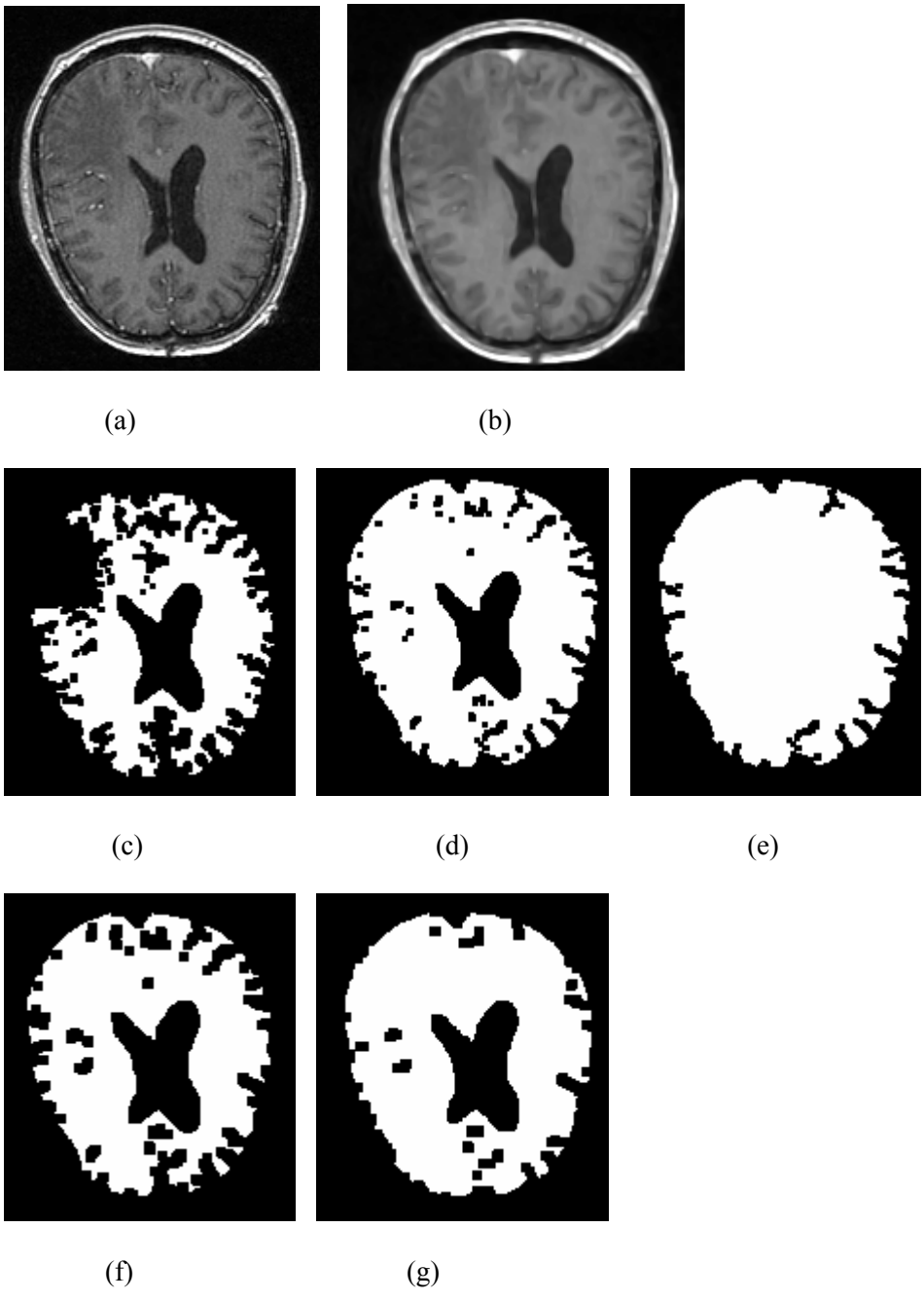


Figure 5-8 (a) – (g) Segmenting brain with neighborhood connected filter

Figure 5-8 (a) shows the original image while Figure 5-8 (b) shows image smoothed with Curvature Anisotropic Diffusion. Figure 5-82 (c), Figure 5-8 (d) shows the result of segmentation with Neighborhood Connected filter with Thresholds (50,120) and Threshold (40,110) respectively. Both of these use Radius 1 and Seed Index (160,115). Figure 5-8 (e) shows the result of applying Gray Scale Fill Hole filter to remove the holes. Here changing the radius affects the smoothness of the segmented object borders as well as size of the segmented region. Increased smoothness of the boundary shows the stability of this filter against noise with increased neighborhood. This may also cost in computing time. Following figures show the results with increased radius value, 2, while keeping Seed Index as (160, 115). Figure 5-8 (f) uses Threshold (42, 110) while figure 5-8 (g) uses Threshold (35, 120). These results show smoother boundaries of the segmented objects but results in increased size for segmented gray matter. Since it does not include isolated pixels, we also see the increased size of the holes in the result. Thus increased radius does not produce desired result even though it provides smoother boundaries.

5.1.3.3 Segmenting Ventricles

Figure 5-9 (a) shows original 256*256 brain MRI slice in axial plane. Figure 5-9 (b) shows result of applying curvature anisotropic diffusion with Iterations = 5, conductance = 1 and Timestep = 0.125.

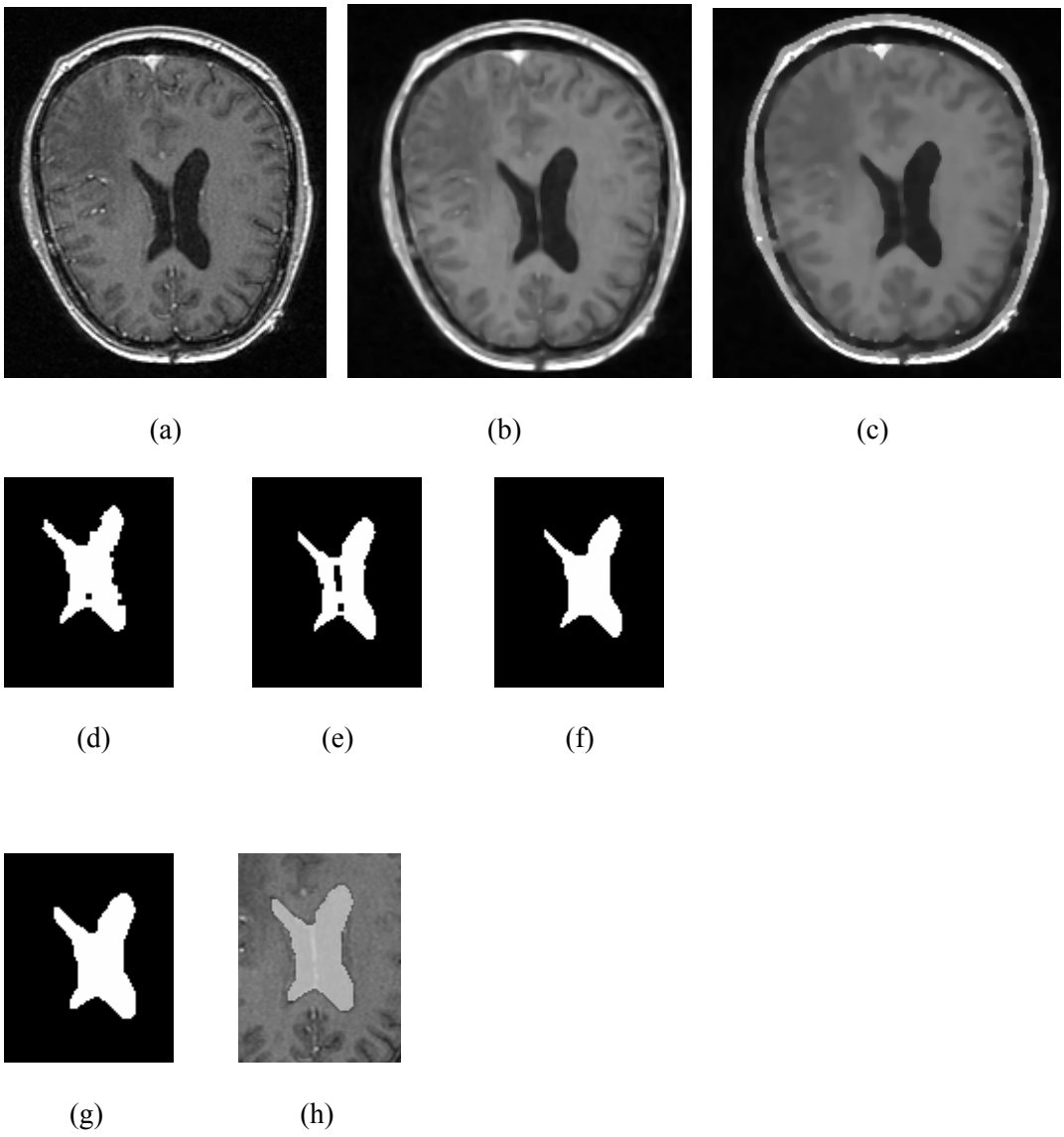


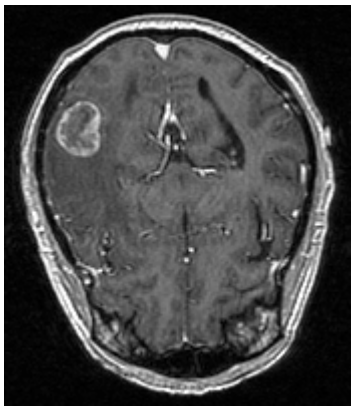
Figure 5-9 (a) – (h) Segmenting ventricles with neighborhood connected filter

Figure 5-9 (d) shows the result of applying Neighborhood connected filter to the original image with Seed Index (140, 131), Radius 1 and Threshold (7, 45). Here smoothing was not used. As seen above, result contains jagged edges and small spurs on upper side of the segmentation. Here filter shows little sensitivity to the noise. The result of segmentation on the smoothed image is shown in figure 5-9 (e). This result still contains some small spurs and edges are not enough smoothed out. Hence we increase the number of iterations of smoothing. Figure 5-9 (c) represents the result of curvature anisotropic diffusion smoothing with Timestep = 0.125,

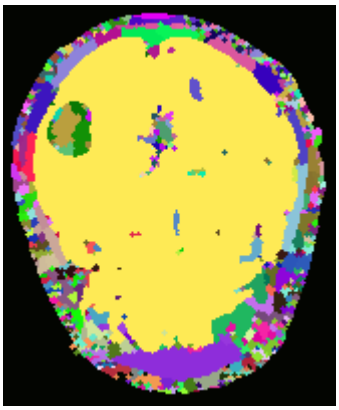
Conductance = 2.0 and Iterations = 10 and figure 5-9 (f) shows the result of segmentation on this smoothed image. This represents better result than in figure 5-9 (e). We can further enhance the result in figure 5-9 (f) by applying the dilation with X-radius=1 and Y-radius =1, as shown in figure 5-9 (g). This result matches with ventricles in original image as shown in figure 5-9 (h).

5.2 Results of Watershed Segmentation

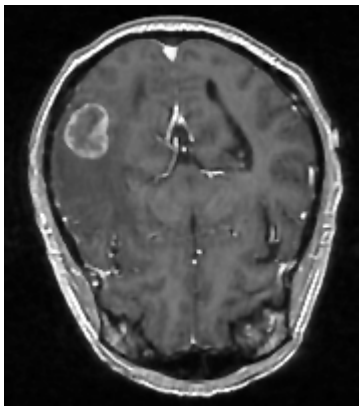
5.2.1 Segmenting Tumor



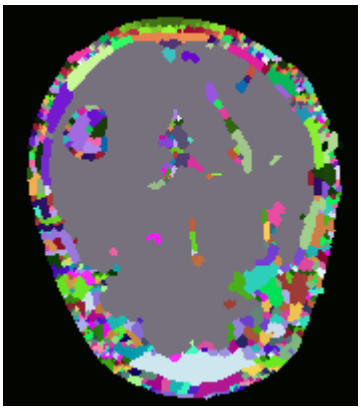
(a)



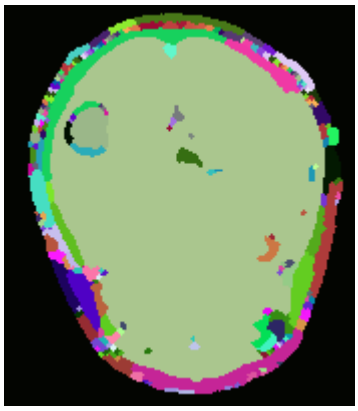
(b)



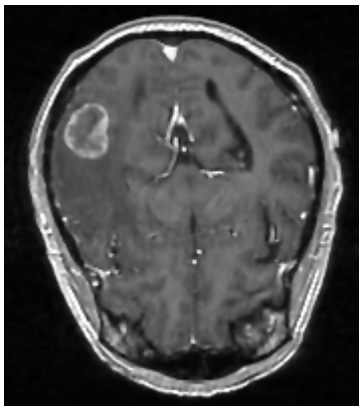
(c)



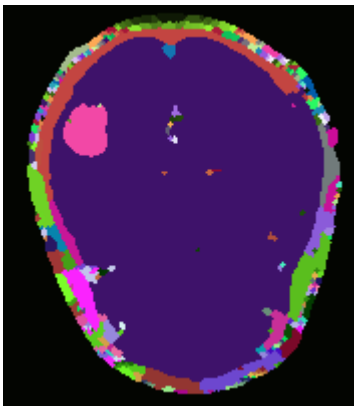
(d)



(e)



(f)



(g)

Figure 5-10 (a) – (g) Segmenting tumor with watershed segmentation filter

Figure 5-10 (a) represents original 256 * 256 brain MRI. Watershed Segmentation is highly sensitive to noise. This is shown in figure 5-10 (b) where applying watershed segmentation without smoothing causes segmentation with too many small regions in the result. Here we apply smoothing before segmentation to remove the noise. Figure 5-10 (c) shows original image smoothed with curvature anisotropic diffusion with Conductance = 2.0, Iterations = 5.

Output of watershed segmentation is an image of unsigned long integer labels, where a label denotes membership of a pixel in a particular segmented region. Since this format is not practical for visualization, we have converted the output to RGB pixels and each segment is represented by a different color.

Figure 5-10 (d) shows the result of applying watershed segmentation on result of curvature anisotropic diffusion. Here, result shows the over-segmentation. To decrease this over-segmentation, we increased the Iterations from 5 to 10 for smoothing with curvature anisotropic diffusion and used conductance = 2.0, and then applied watershed segmentation to it with Threshold = = 0.011 and OutputScaleLevel (Flood Level) = 0.232. Result of this is shown in figure 5-10 (e) with reduced number of segments. Scale Level greater than 0.232 for Threshold greater than 0.011 removes tumor from the result. Here, even though number of segments is reduced, tumor is not obtained as a single segment. It still shows small segments inside the tumor.

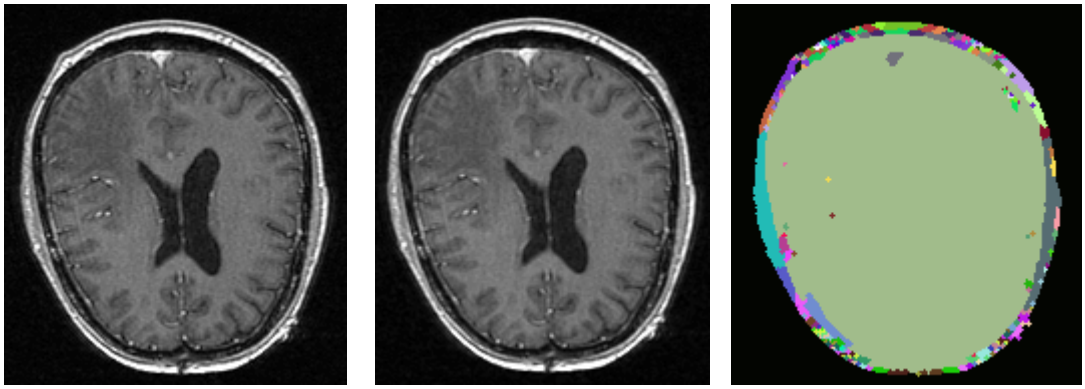
Figure 5-10 (f) shows original image smoothed with gradient anisotropic diffusion with Conductance = 2.0, Iterations = 5. Figure 5-10 (g) shows result of watershed on image smoothed with gradient anisotropic diffusion (figure 5.2.1 (f)). This result is obtained with Threshold = 0.011 and Scale Level = 0.29 as maximum values. With gradient anisotropic diffusion,

segmented tumor does not have multiple segments as with curvature anisotropic diffusion smoothing.

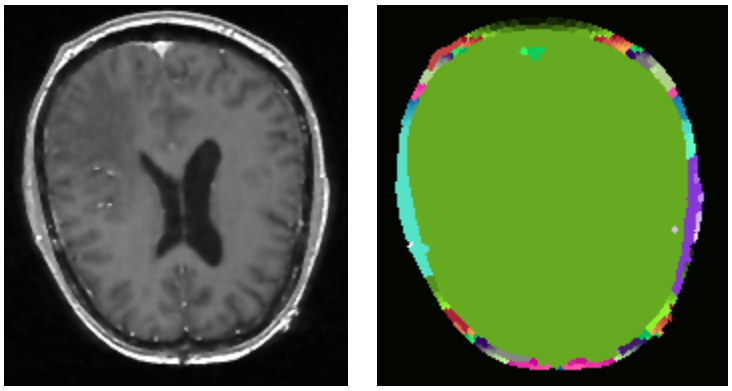
In general, smoothing with less number of iterations require more computation time during segmentation and larger value of threshold since watershed segmentation is highly sensitive to noise and creates too many smaller regions/segments which are not desired in many cases.

In many cases, tuning the filter parameters, in most techniques, including watershed segmentation, is a process of trial and error. The threshold parameter can be used to great effect in controlling over-segmentation of the image. Raising the threshold will generally reduce computation time and produce output with fewer and larger regions. Hence the trick in tuning parameters is to consider the scale level (Flood Level) of the objects that we are trying to segment in the image. The best time/quality trade-off may be achieved when the image is smoothed and thresholded to eliminate features just below the desired scale.

5.2.2 Segmenting Brain



(a) (b) (c)



(d) (e)

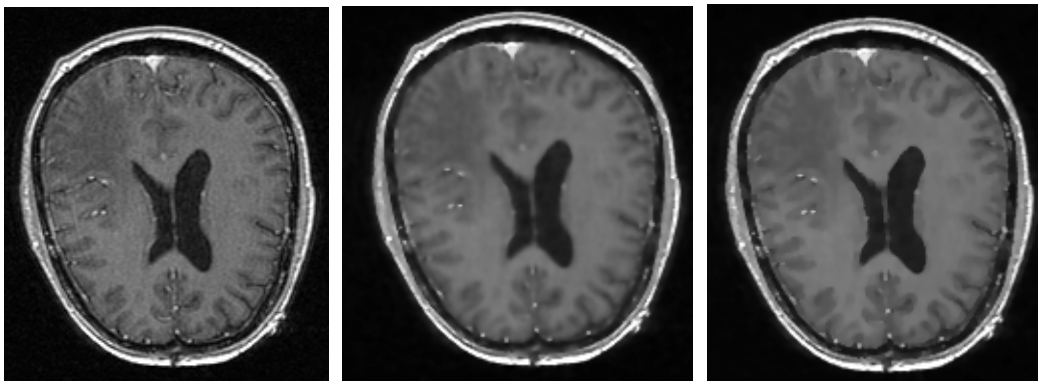
Figure 5-11 (a) – (e) Segmenting brain with watershed segmentation filter

Figure 5-11 (a) shows original 256 * 256 brain MRI slice. Figure 5-11 (b) shows image smoothed with curvature flow anisotropic diffusion with timestep = 0.125, iterations = 10 and conductance = 2.0 and figure 5-11 (c) shows the result of watershed segmentation on this smoothed image with scale level = 0.34 and threshold = 0.011.

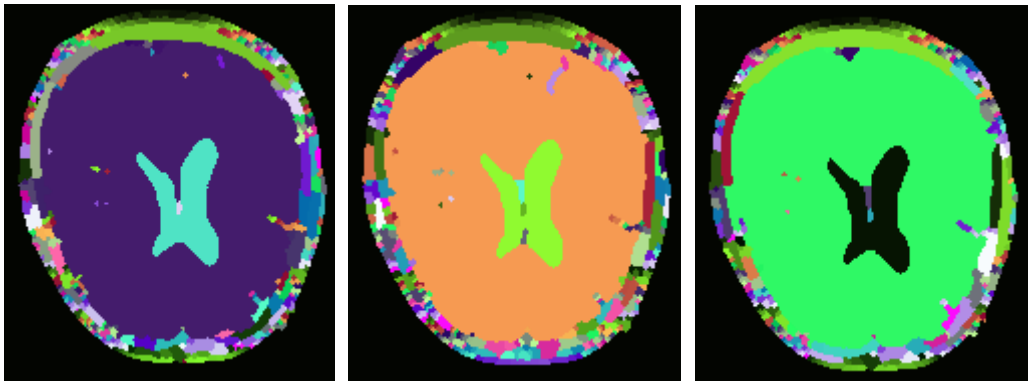
Figure 5-11 (d) shows image smoothed with gradient flow anisotropic diffusion with timestep = 0.125, iterations = 5 and conductance = 2.0 and figure 5-11 (e) shows the result of watershed segmentation on this smoothed image with scale level = 0.1 and threshold = 0.088.

This result shows better result than in figure 5-11 (c) due to fewer segments present inside the brain area.

5.2.3 Segmenting Ventricles



(a) (b) (c)



(d) (e) (f)

(g)

Figure 5-12 (a) – (g) Segmenting ventricles with watershed segmentation filter

Figure 5-12 (a) shows the original 256 * 256 MRI slice and figure 5-12 (b) shows the result of Gradient Anisotropic Diffusion smoothing on original image, with conductance = 2.0, iterations = 5 and TimeStep = 0.125. Figure 5-12 (d) shows the result of applying Watershed Segmentation on this smoothed image. As seen in this result, some very small segments are present which are undesirable.

Figure 5-12 (e) shows the result of applying Watershed segmentation on image smoothed with Curvature Anisotropic Diffusion with Conductance = 2.0, Iterations = 10 and TimeStep = 0.125. Figure 5-12 (f) shows the result of segmentation on image smoothed with gradient anisotropic diffusion with conductance = 3, Iterations = 20 and lower threshold = 0.03, OutputScaleLevel = 0.19 for watershed segmentation. Figure 5-12 (g) shows the result of applying Watershed segmentation with Threshold = 0.02 and OutputScaleLevel = 0.19, on image smoothed with Curvature Anisotropic Diffusion with Conductance = 2.0, Iterations = 10 and Timestep = 0.125.

As clearly seen from above results, figure 5-12 (g) represents better result of all with curvature anisotropic diffusion smoothing (figure 5-12 (c)). Curvature anisotropic diffusion smoothing is less sensitive to contrast than classic Perona-Malik style diffusion, and preserves finer detailed structures in images. Although using this function may slower the segmentation, as compared to ITK implemented Gradient Anisotropic Diffusion Function, fewer iterations may be required to reach an acceptable solution.

5.3 Results of Level Set Segmentation

5.3.1 Threshold Level Set Segmentation

5.3.1.1 Segmenting Tumor

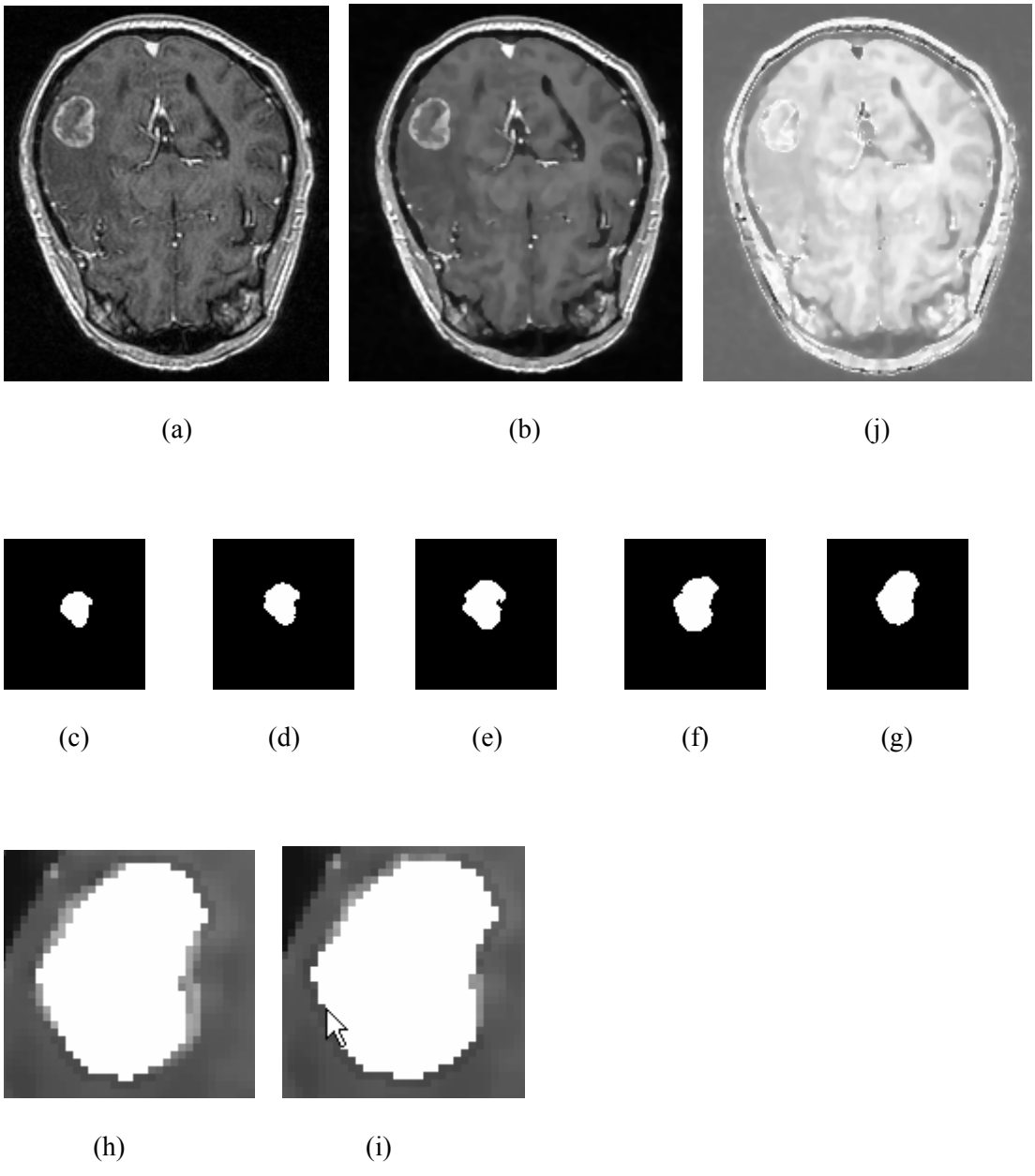


Figure 5-13 (a) – (j) Segmenting tumor with threshold level set segmentation filter

Figure 5-13 (a) shows original brain MRI slice of 256 * 256. Figure 5-13 (b) shows image smoothed with curvature anisotropic diffusion with iterations = 10, conductance = 2.0 and timestep = 0.125. Figure 5-131 (c) shows the result of threshold level set segmentation on smoothed image with seed (82, 86), initial distance = 5, RMS error = 0.02, iterations = 30, lower threshold = 45, upper threshold = 147, propagation scale = 1.0 and curvature scale = 3.0. Figure 5-13 (d) shows the result of threshold level set segmentation on smoothed image with seed (82, 86), initial distance = 5, RMS error = 0.02, iterations = 40, lower threshold = 45, upper threshold = 147, propagation scale = 5.0 and curvature scale = 15.0. Figure 5-13 (e) shows the result of threshold level set segmentation on smoothed image with seed (82, 86), initial distance = 5, RMS error = 0.02, iterations = 50, lower threshold = 45, upper threshold = 147, propagation scale = 5.0 and curvature scale = 15.0. As seen from these figures, increasing the number of iterations and propagation scale increase the size of the tumor while increased curvature scale smoothes the segmented tumor. But here result does not match the actual size and the shape of the tumor.

The shape of the initial level set generated by fast marching filter is totally dependent on the location of the seed point(s) and the time used for thresholding the time-crossing map. The process of the Fast Marching filter used in this case is equivalent to a distance map to the seed points.

With a single seed point, it may take longer for the front to propagate and cover the structure. Using multiple seed points in the initialization of the fast marching will generate an initial level set much closer in shape to the object to be segmented with appropriate selection of initial distance given as input to fast marching. In all cases, initial distance is the distance from the seed points at which initial contour would be and the rule of thumb for the user is to select this value in such a way that initial contour is close to the boundary of the object. This results in less number of iterations to fill and reach the edges of the anatomical structure.

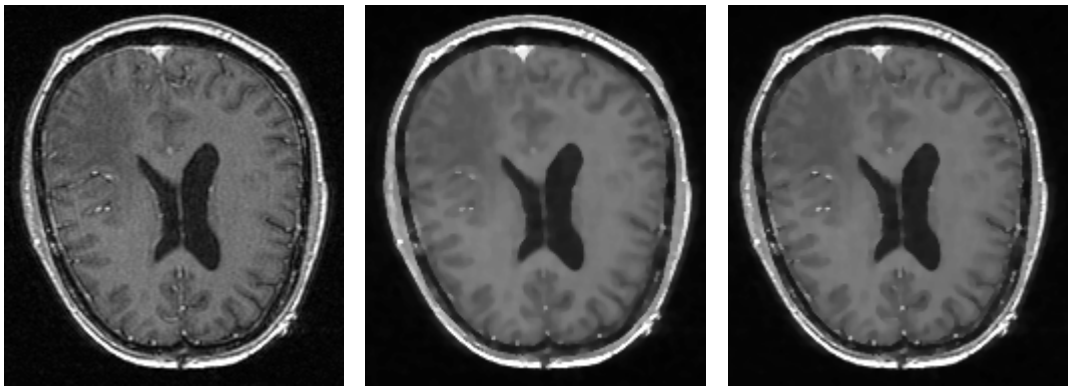
Figure 5-13 (f) shows the result of using multiple seed points with RMS error = 0.02, iterations = 25, lower threshold = 45, upper threshold = 147, initial distance = 5, propagation scale = 2.0, curvature scale = 10.0 and following seed points: With Seed #1(82 86), Seed #2(76 98), Seed #3(75 94), Seed #4(74 89), Seed #5(77 86), Seed #6(84 85), Seed #7(86 90), Seed #8(78 98). Although this result produces the basic structure of the tumor to be segmented, the result needs to be smoother and smaller in size. Larger value for curvature scaling parameter results in smoother segmentation result. However, the curvature scaling parameter should not be set too large, as it will draw the contour away from the shape boundaries. We need to increase the curvature scaling and decrease the propagation scale. We also need to change the values of seed points slightly.

Figure 5-13 (g) shows the result of using multiple seed points with RMS error = 0.02, iterations = 25, lower threshold = 45, upper threshold = 147, initial distance = 5, propagation scale = 1.0, curvature scale = 25.0 and following seed points: With Seed #1(82 86), Seed #2(76 94), Seed #3(75 94), Seed #4(74 89), Seed #5(77 86), Seed #6(82 83), Seed #7(86 90), Seed #8(78 96). This result clearly requires less number of iterations with multiple seed points for fast marching. From figure 5-13 (h) we can see that above result closely resembles the actual size of the tumor to be segmented. We can improve this result by further processing that result with dilation in x-axis. This result is shown in figure 5-13 (i). Figure 5-13 (j) shows the speed image for this result in figure 5-13 (i).

In general, selection of seed points and appropriate values for scaling parameters is critical to the segmentation. The curvature term will smooth the surface of the zero set. If the curvature weight is too low with respect to the propagation and advection terms then the zero-set will tend to expand and will try to enter every small protrusion of the anatomical structure. This will make the zero-set prone to leaking in regions of small contrast. Using a high curvature weight will prevent the level set from attempting to enter in small details: a large enough curvature may induce contractions of the zero-set.

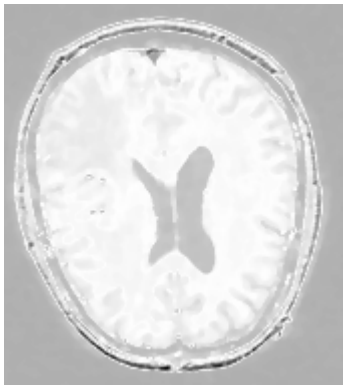
Because the level-set equations are usually solved only at pixels near the surface, the time taken at each iteration depends on the number of points on the surface. This means that as the surface grows, the solver will slow down proportionally. Because the surface must evolve slowly to prevent numerical instabilities in the solution, the distance the surface must travel in the image dictates the total number of iterations required. Once activated, the level set evolution will stop if the convergence criteria or the maximum number of iterations is reached. The convergence criteria are defined in terms of the root mean squared (RMS) change in the level set function. The evolution is said to have converged if the RMS change is below a user-specified threshold.

5.3.1.2 Segmenting Brain



(a) (b) (c)

(d) (e) (f)



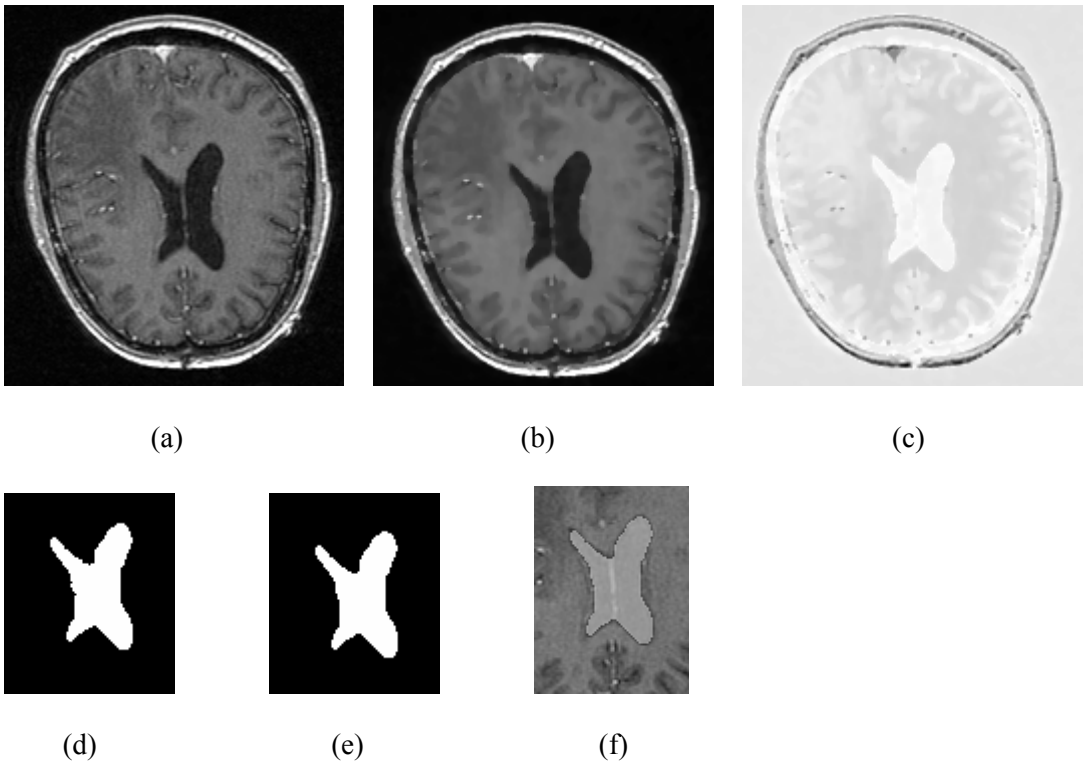
(g)

Figure 5-14 (a) – (g) Segmenting brain with threshold level set segmentation filter

Figure 5-14 (a) shows original 256 * 256 brain MRI slice. Figure 5-14 (b) shows original image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-14 (d) shows the result of segmentation with RMS error = 0.02, iterations = 1000, lower threshold = 43, upper threshold = 78, Seed Index # (160 115), initial distance = 15, propagation scale = 5.0 and curvature scale = 15.0. This result has a small spur on left side of the image. This spur shows the leaking of the segmentation outside the brain region. We can reduce this leak in several ways such as increasing the curvature term. Too small value of curvature term causes to enter the segmentation into every small protrusion of the anatomical structure and this makes the zero-set prone to leaking in regions of small contrast. We first reduced the conductance term used for smoothing the result and resulting into small contrast. By using conductance term = 1.0 and iterations = 5, we get the result in figure 5-14 (c). Then applying threshold level set segmentation with RMS error = 0.02, iterations = 1000, increasing lower threshold to 46, upper threshold = 78, Seed Index # (160 115), initial distance = 15, propagation scale = 5.0 and increasing the curvature scale to 25.0, we get better result as shown in figure 5-14 (e). Speed image for this result in shown in figure 5-14 (g). This result can be further improved by using GrayScaleFillHoleFilter to remove small holes inside the segmentation result as shown in figure 5-14 (f).

Figure 5-14 (e) shows how the smoothness constraint on the surface prevents leakage of the segmentation into both ventricles, but also localizes the segmentation to a smaller portion of the gray matter instead of entire gray matter.

5.3.1.3 Segmenting Ventricles



(a) (b) (c)

(d) (e) (f)

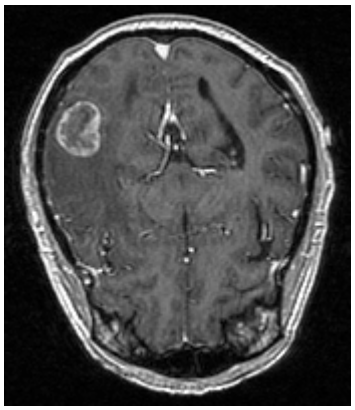
Figure 5-15 (a) – (f) Segmenting ventricles with threshold level set segmentation filter

Figure 5-15 (a) shows original 256 * 256 brain MRI slice and figure 5-15 (b) shows the image smoothed with curvature anisotropic diffusion. Figure 5-15 (d) and figure 5-15 (e) show the result of segmentation. Since ventricles has homogeneous region, its segmentation is straightforward. Figure 5-15 (d) shows the result of segmentation on image smoothed with iterations = 5 and conductance = 1.0. Segmentation used RMS error = 0.02, iterations = 525, lower threshold =15, upper threshold = 40, Seed Index # (138 139), initial distance = 15, propagation scale = 1.0 and curvature scale = 4.0. Figure 5-15 (e) shows the result of segmentation on image smoothed with iterations = 5 and conductance = 2.0 and segmentation

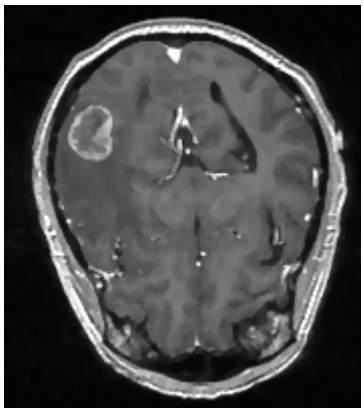
parameters with same values as for result in figure 5-15 (d). Here, increasing the smoothing with increased conductance parameter yields better result than previous result, with boundaries of the segmented object more smoothed out. This result did not need any post-processing as shown in figure 5-15 (f).

5.3.2 Shape Detection Level Set Segmentation

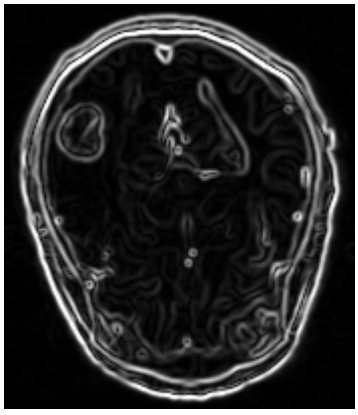
5.3.2.1 Segmenting Tumor



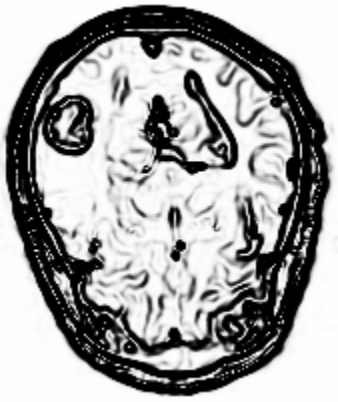
(a)



(b)



(c)

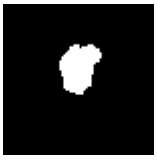


(d)

(e)

(f)

(g)



(h)

(i)

Figure 5-16 (a) – (i) Segmenting tumor with shape detection level set segmentation filter

Figure 5-16 (a) shows original brain MRI slice of 256*256 and figure 5-16 (b) shows image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-16 (c) shows the gradient magnitude with sigma = 1.0 and figure 5-16 (d) shows the result of sigmoid filter with alpha = -1.0 and beta = 5.0.

Figure 5-16 (e) shows the result of shape detection segmentation with Seed Index (80, 92), initial distance = 5, alpha = -1.0, beta = 5, curvature scale =1, propagation scale = 20 and iterations = 800. This result shows the over-segmentation in that result enters into the small protrusions and this is undesirable. Looking at the speed image, we can observe the speed. We want to stop the segmentation at the boundaries of the tumor. In the result, there are small protrusions which cause segmentation to cross the tumor and enter into the brain area.

Figure 5-16 (f) shows the result of shape detection segmentation with Seed Index (80, 92), initial distance = 5, alpha = -1.0, beta = 5, curvature scale = 4, propagation scale = 10 and iterations = 600. Here over-segmentation is significantly reduced but there is still small spur on top of the resulting segmented tumor. From this result it seems difficult to get tumor segmented properly. We would like to obtain initial level set closed to the tumor structure and then apply shape detection level set on it. Also with a single seed point, it may take longer for the front to propagate and cover the structure. So we use multiple seed points in the initialization of the fast marching that will generate an initial level set much closer in shape to the object to be segmented with appropriate selection of initial distance given as input to fast marching. Again in this case we select the value for initial distance in such a way that initial contour is close to the boundary of the object. This may result in less number of iterations to fill and reach the edges of the anatomical structure.

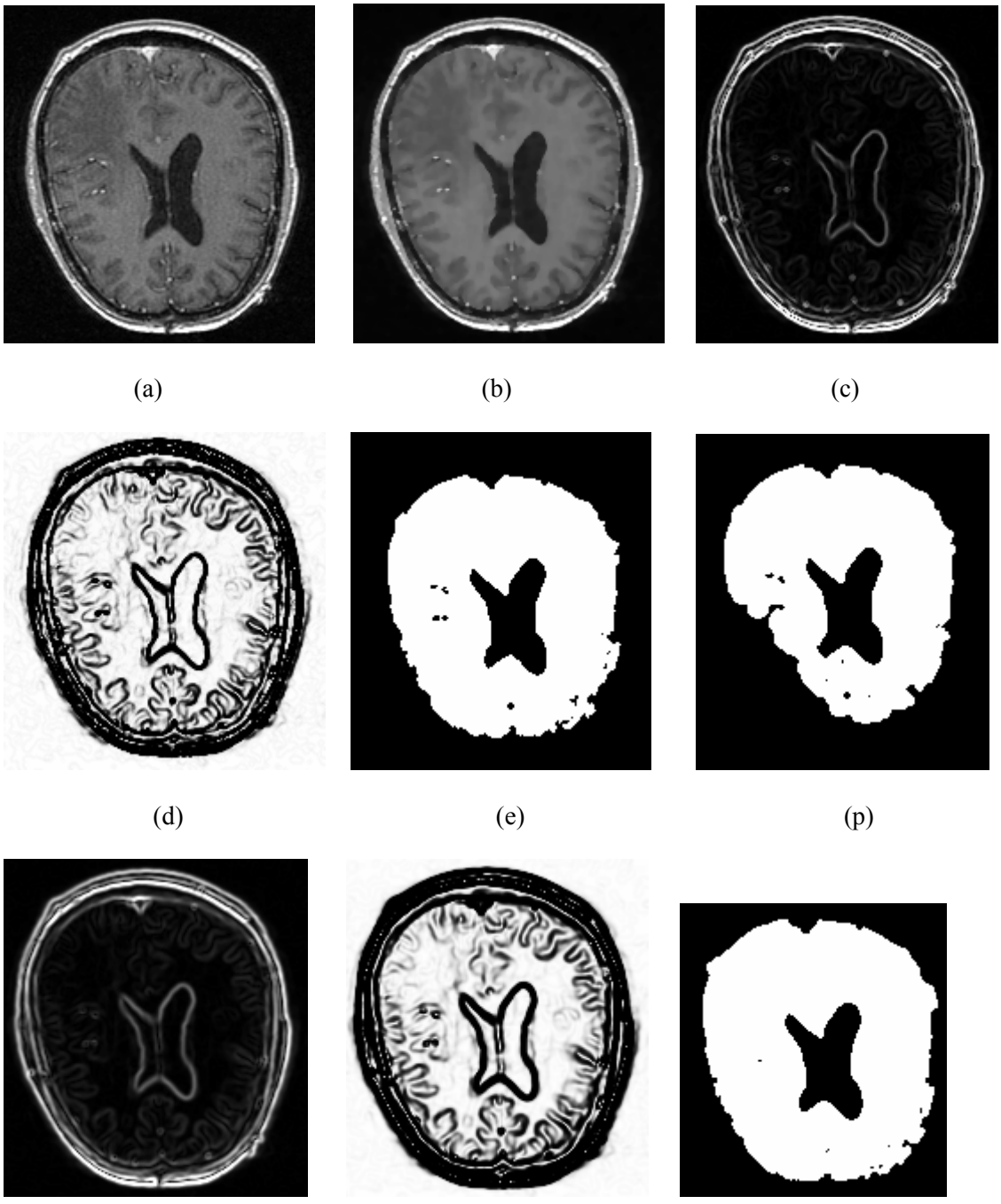
Figure 5-16 (g) shows the result of segmentation with multiple seed points: Seed #1(80, 92), Seed #2(76, 98), Seed #3(75, 94), Seed #3(74, 89), Seed #3(77, 86), Seed #3(84, 85) and Seed #3(78, 98). This result uses initial distance = 3, alpha = -0.9 and beta = 3, curvature scale = 13, propagation = 10 and iterations = 800.

Figure 5-16 (h) shows the result with initial distance= 4, alpha = -0.9, beta = 3, curvature scale = 0.5, propagation scale = 1 and iterations = 800. Seed points used are same as those for figure 5-16 (g).

Figure 5-16 (i) shows the result with initial distance= 3, alpha = -0.9, beta = 4, curvature scale = 0.5, propagation scale = 35 and iterations = 800. Seed points used are same as those for figure 5-16 (g). Here increasing propagation scale from 1 to 35 and increasing beta slightly makes significant change to the result. This result closely resembles the structure of the tumor, but border needs to be more smoothed out even though there is additional curvature term in the driving equation of the shape detection. In general, segmentation of tumor is problematic.

One approach is to get the basic structure with inner boundary as shown in figure 5-16 (g), dilate that result and then pass it as initial level set to shape detection again. Another approach to refine this segmentation is to use another level set methods such as laplacian level set. This will be demonstrated in section 5.3.4.1.

5.3.2.2 Segmenting Brain



(a) (b) (c)

(d) (e) (p)

(f) (g) (h)

(i) (j) (k)

Figure 5-17 (a) – (k) Segmenting brain with shape detection level set segmentation filter

Figure 5-17 (a) shows original brain MRI slice of 256*256 and figure 5-17 (b) shows image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-17 (c) shows the gradient magnitude with sigma = 0.5 and figure 5-17 (d) shows the result of sigmoid filter with alpha = -0.9 and beta = 3.0. Figure 5-17 (e) shows the result of segmentation with seed index # (160, 115), initial distance = 7, curvature scale = 6.0, propagation scale = 10 and number of iterations = 700. Speed image in figure 5-17 (d) does not produce the clear boundary with speed = 0 where we want segmentation to stop. Hence result in figure 5-17 produces the result with leakage. Here, curvature term also plays important role. If the curvature weight is too low with respect to the propagation and advection terms then the zero-set will tend to expand and will try to enter every small protrusion of the anatomical structure. This will make the zero-set prone to leaking in regions of small contrast. Using a high curvature weight will prevent the level set from attempting to enter in small details: a large enough curvature may induce contractions of the zero-set. This is shown in figure 5-17 (p).

Figure 5-17 (f) shows the gradient magnitude of smoothed image with sigma = 1 and figure 5-17 (g) shows the speed image, that is the speed image with alpha = -0.9 and beta = 3.0.

2.0. Figure 5-17 (h) shows the result of segmentation with above speed image and seed index # (160, 115), initial distance = 7, curvature scale = 6.0, propagation scale = 10 and number of iterations = 700.

Figure 5-17 (i) shows the gradient magnitude with sigma = 2.0 and figure 5-17 (j) shows the result of sigmoid filter with alpha = -0.9 and beta = 3.0. Figure 5-17 (k) shows the result of segmentation with seed index # (160, 115), initial distance = 7, curvature scale = 6.0, propagation scale = 10 and number of iterations = 700.

Figure 5-17 (k) is more smoothed out result as compared to figure 5-17 (h) and figure 5-17 (e). Shape detection level set also produces more smoothed out result compared to threshold level set segmentation.

In all above case, 700 is the minimum number of iterations; iterations less than 700 cause under-segmentation of the brain area. This result shows the over-segmentation in that result enters into the small protrusions and this is undesirable. Looking at the speed image, we can clearly see the speed is not zero at the right side and there are small protrusions which cause segmentation to cross the right side boundary with propagation scale of 10. As discussed earlier in chapter 4, level set segmentation programs take 2 inputs: Initial model which we have produced using fast marching in this case, and a feature image. A feature image is either the image we wish to segment or some preprocessed version. We use input image to be segmented as a feature image from which speed image is produced using gradient magnitude and sigmoid filters respectively. Thus for shape detection algorithm, the feature image is an edge potential image. This feature image is what the level set filter actually sees. Hence it is critical to get a high quality feature image if we expect to get reasonable results from the level set methods. The weights for curvature, propagation also play a role in shaping the zero set. But the fact of the matter is that no combination of these parameters can compensate for a bad feature image. Conversely, a good feature image will produce reasonable level sets even if we provide bad combinations of the weights for curvature, propagation.

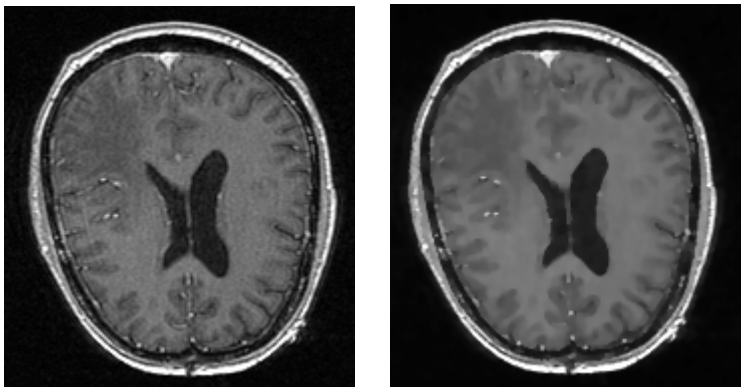
The rule of thumb is that the feature image should almost look like a fuzzy segmentation of the anatomical structure that we are looking for. Figures 5-18 (l)-(m) show the final results. Result in figure 5-18 (n) contains some leakage outside brain area. This is due to some gaps in gradient magnitude image which results into non-zero speed terms in sigmoid image and this causes the leakage.

(l) (m) (n)

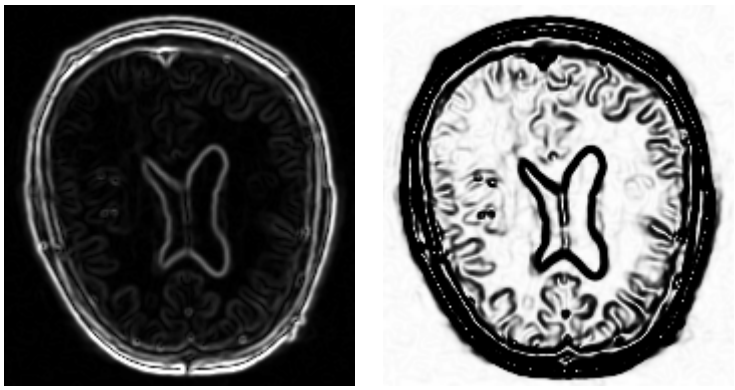
Figure 5-18 (l) – (n) Segmenting brain with shape detection level set segmentation filter

In general, a larger number of iterations is required for segmenting large structures since it takes longer for the front to propagate and cover the structure. This drawback can be easily mitigated by setting any seed points in the initialization of the FastMarchingImageFilter. This will generate an initial level set much closer in shape to the object to be segmented and hence require fewer iterations to fill and reach the edges of the anatomical structure.

5.3.2.3 Segmenting Ventricles



(a) (b)



(c) (d)

(e) (f) (g) (h)

Figure 5-19 (a) – (h) Segmenting ventricles with shape detection level set segmentation filter

Figure 5-19 (a) show original 256 * 256 brain MRI slice and figure 5-19 (b) shows the image smoothed with curvature anisotropic diffusion with iterations = 5 and conductance = 2.0. Figure 5-19 (c) show the gradient magnitude of the smoothed image, with sigma = 1 and figure 5-19 (d) shows the sigmoid of the gradient magnitude with alpha = -0.9 and beta = 3.0.

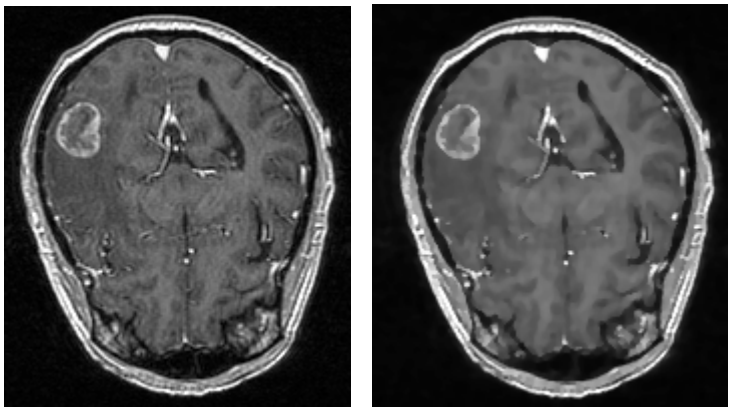
Figure 5-19 (e) shows the result of segmentation with Seed Index (138, 139), initial distance=6, curvature scale = 4.0, propagation scale = 10, iterations = 400 and RMS error = 0.001.

As shown in figure 5-19 (g), result is slightly under-segmented in that left corners are not covered under the segmentation. This result can be slightly improved by increasing the number of iterations for shape detection level set from 400 to 600 and increasing the y-radius for dilation. This result is shown in figure 5-19 (h). The small spur on the right side center can be removed by tweaking the parameters with more trials.

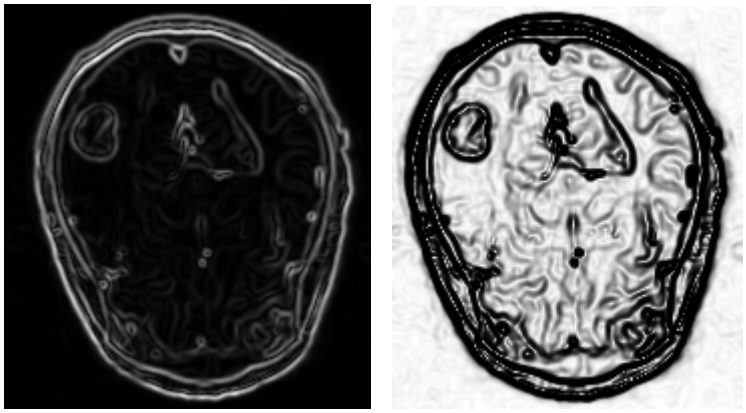
5.3.3 Geodesic Level Set Segmentation

For the GeodesicActiveContourLevelSetImageFilter, scaling parameters are used to trade off between the propagation (inflation), the curvature (smoothing) and the advection terms.

5.3.3.1 Segmenting Tumor



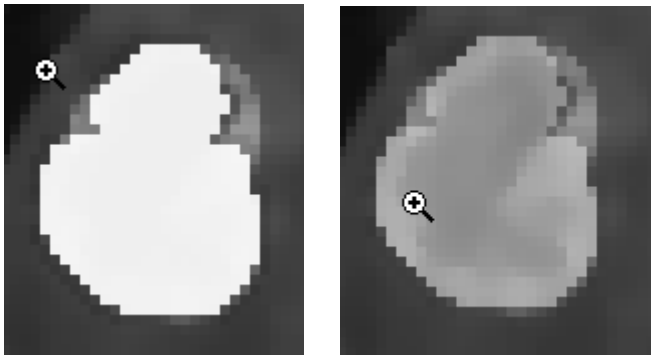
(a) (b)



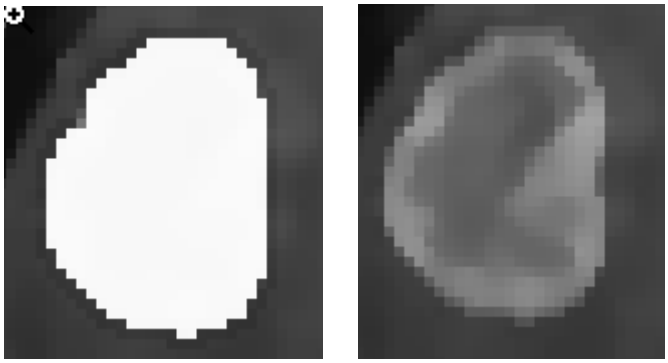
(c) (d)

(e) (f) (g) (h)

(i) (j) (k)



(l) (m)



(n) (p)

Figure 5-20 (a) – (p) Segmenting tumor with geodesic active contour level set segmentation filter

Figure 5-20 (a) shows original brain MRI slice of 256*256 and figure 5-20 (b) shows image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-20 (c) shows the gradient magnitude with sigma = 1.0 and figure 5-

20 (d) shows speed image which is the sigmoid of gradient magnitude with alpha = -2.9 and beta = 3.0.

Figure 5-20 (e) shows the result of geodesic level set segmentation with Seed Index (80, 92), initial distance = 4, alpha = -2.9, beta = 3.0, curvature scale = 2, propagation scale = 1, advection = 1, iterations = 500 and RMS error = 0.02. Here, level set evolution stops since the convergence criterion of 0.02 is reached before the 500 iterations completed. It stops after 214 iterations. This result closely resembles the shape of inner boundary of the tumor. Here, we want to attract it towards the outer boundary. One nice property of geodesic level set is that new term "advection term" attracts the level set towards the boundary and it can easily cross the inner boundary. So, we need to adjust the values for advection term and adjust curvature and propagation terms. We also need to reduce RMS error and adjust number of iterations accordingly to get desired result.

Figure 5-20 (f) shows the result of segmentation with RMS error = 0.01, iterations = 350, seed index (80, 92), initial distance = 4, alpha = -2.9, beta = 3.0, propagation scale = 1 and curvature scale = 2. Here, reducing the RMS error let the level set to evolve after completing 350 iterations and brings the result closer to the outer boundary of the tumor. But this result shows small spur on top and it needs to be smoothed out more. Increasing the propagation scale to 3 for more inflation of the segmentation causes over-segmentation due to low value of curvature and causes segmentation to leak into outer region. This is shown in figure 5-20 (g).

For Geodesic Active Contour Level Set, scaling parameters are used to trade off between the propagation (inflation), the curvature (smoothing) and the advection terms. Hence it is important to use appropriate values of these values with respect to each other. Hence we need to increase the curvature scale now. Figure 5-20 (h) shows the result with seed index (76, 92), RMS error = 0.01, iterations = 350, seed index (80, 92), initial distance = 4, alpha = -2.9, beta = 3.0, propagation scale = 2 and curvature scale = 3, and advection = 1. Although the parameters in this

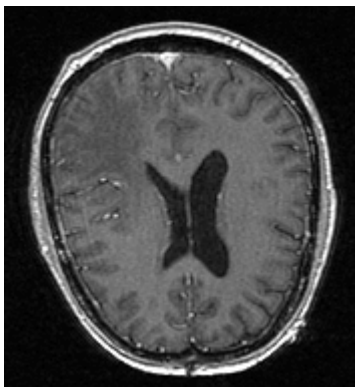
result prevent the leakage of the segmentation into the brain area, it does not match the shape of the tumor, as shown in figures 5-20 (l) and 5-20 (m).

Figure 5-20 (i) shows the result with seed index (76, 92), RMS error = 0.01, iterations = 350, seed index (80, 92), initial distance = 4, alpha = -2.9, beta = 3.0, propagation scale = 2 and curvature scale = 5, and advection = 1. Figure 5-20 (j) shows the result of applying dilation with y-radius = 1 and x-radius = 0 to the result of figure 5-20 (i). This result closely matches with tumor shape and size but slightly more inflated to match the tumor exactly. Finally, tweaking scaling parameters give the result in figure 5-20 (k). This result requires RMS error = 0.01, iterations = 500, seed index (76, 92), initial distance = 4, alpha = -2.9, beta = 3.0, propagation scale = 2 and curvature scale = 2, and advection = 1. This result matches exactly to the tumor as shown in figures 5-20 (n) and 5-20 (p).

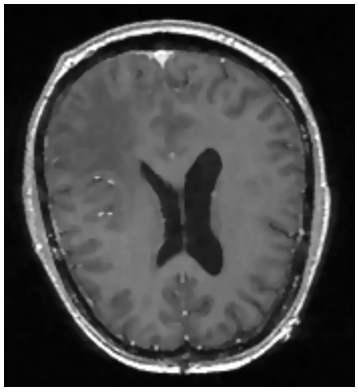
Figure 5-20 (p) shows the original tumor and segmented tumor overlapped with increased opacity. From this figure we can clearly see how geodesic has crossed the inner boundary with just single seed point and only 350 iterations as compared to the threshold level set and shape detection level set segmentation.

5.3.3.2 Segmenting Brain

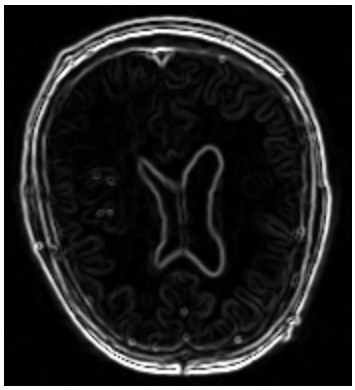
Figure 5-21 (a) shows original brain MRI slice of 256*256 and figure 5-21 (b) shows image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-21 (c) shows the gradient magnitude with sigma = 0.5 and figure 5-21 (d) shows the result of sigmoid filter with alpha = -0.9 and beta = 3.0. Figure 5-21 (e) shows the result of segmentation with seed index # (160, 115), initial distance = 7, curvature scale = 6.0, propagation scale = 10 and number of iterations = 700.



(a)

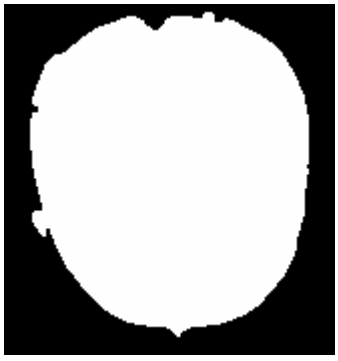


(b)



(c)

(d)



(e)

(f)

(g)

(h) (i)

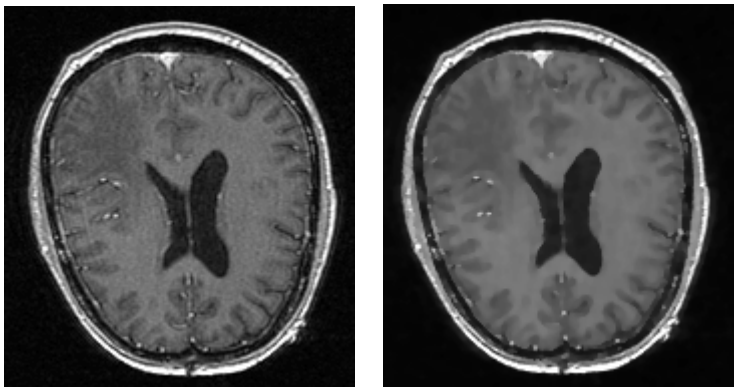
Figure 5-21 (a) – (i) Segmenting brain with geodesic active contour level set segmentation filter

Figure 5-21 (g) shows sigmoid of gradient magnitude with sigma = 0.5, alpha = -2.0 and beta = 2.5. Figure 5-21 (c) shows the gradient magnitude with sigma = 1.5 and figure 5-21 (d) shows the sigmoid of gradient magnitude with alpha = -2.5 and beta = 2.0. Figure 5.3.3.2 (e) shows the result of segmentation with RMS error = 0.02, iterations = 600, seed index # (160, 115), initial distance = 4, propagation scale = 3, curvature scale = 1 and advection scale = 1. With these parameter values, level set crosses the boundaries of the ventricles and enters into it. To restrict this propagation into ventricles, we need to reduce the propagation scale. Figure 5-21 (f) shows the result with propagation scale = 1. But it does not solve the purpose completely. Finally, figure 5-21 (g) shows the result with RMS error = 0.02, iterations = 600, seed index # (160, 115), initial distance = 4, propagation scale = 3, curvature scale = 3 and advection scale = 1, sigma = 1.5 for gradient magnitude and alpha = -1.2 and beta = 2.0 for sigmoid of the gradient magnitude. Here, segmentation does not enter into the ventricles at all. Increased curvature scale prevents the segmentation to enter into ventricles region, but segmentation still enters outside the brain boundary due to non-zero speed at the locations while crossing the inner boundary. This is shown by small spurs on left and right sides.

Note that a relatively larger propagation scaling value was required to segment the white matter. This is due to two factors: the lower contrast at the border of the white matter and the complex shape of the structure. Unfortunately the optimal value of these scaling parameters can only be determined by experimentation. To get the correct values we need an interactive mechanism by which a user supervises the contour evolution and adjusts these parameters accordingly to stop the evolution of level set at the desired boundary.

A one more possible workaround is to use multiple seeds distributed along the elongated object. Figure 5-21 (i) shows the result of segmentation with RMS error = 0.02, iterations = 600, seed index # (160, 115), initial distance = 4, propagation scale = 3, curvature scale = 1 and advection scale = 1. Level set evolves at RMS change = 0.0166861 and elapsed iterations are 586. Here, boundaries are not quite smoothed and segmentation slightly enters into outer region of the brain area. This result uses multiple seeds: (160, 160), (115, 185), (100, 155), (103, 90)

5.3.3.3 Segmenting ventricles



(a) (b)

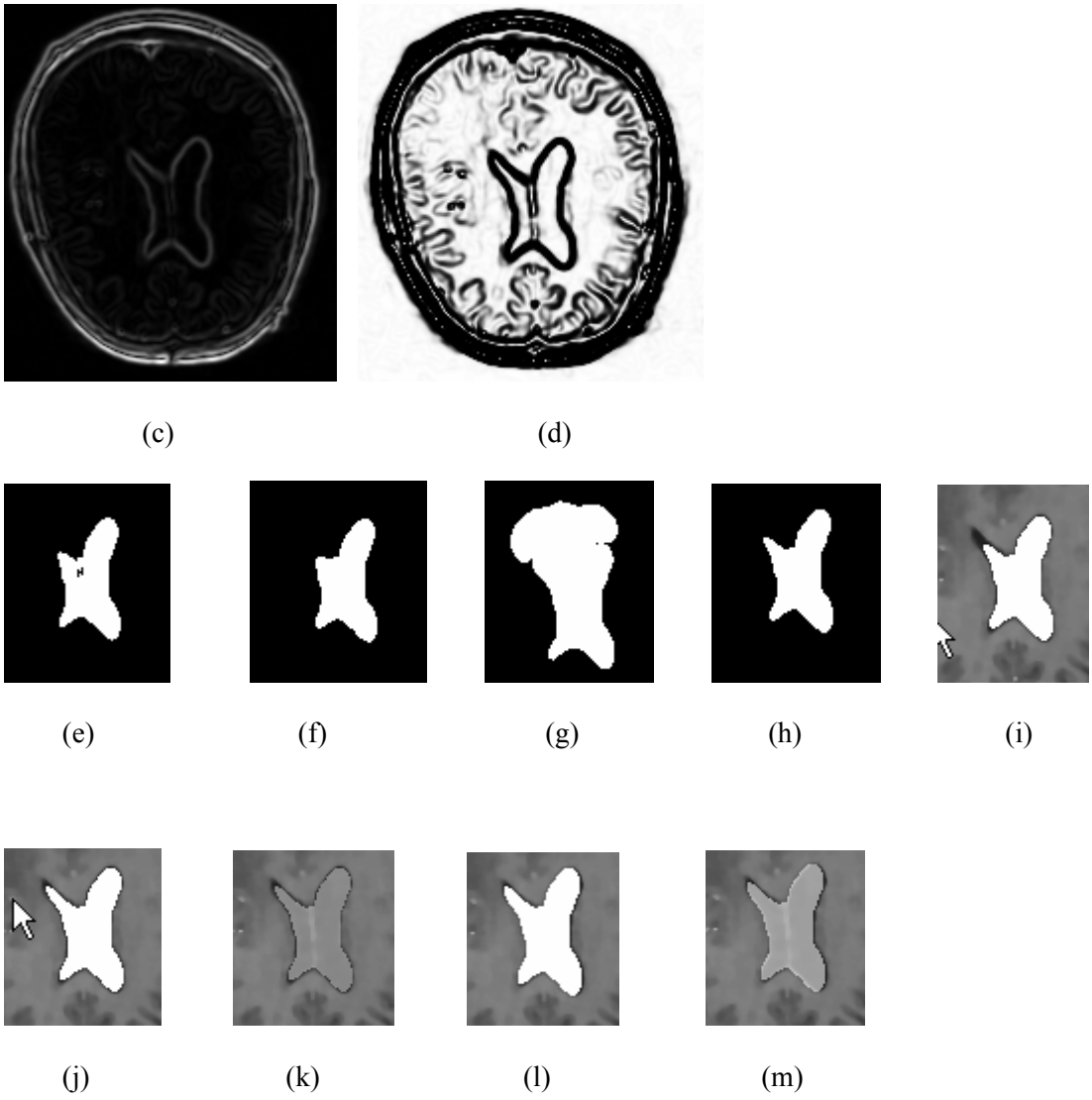


Figure 5-22 (a) – (m) Segmenting ventricles with geodesic active contour level set segmentation filter

Figure 5-22 (a) shows original 256 * 256 brain MRI slice and figure 5-22 (b) shows the image smoothed with curvature anisotropic diffusion with iterations = 5 and conductance = 2.0. Figure 5-22 (c) shows the gradient magnitude of smoothed image with sigma = 1.0 and figure 5-22 (d) shows the sigmoid of the gradient magnitude with alpha = -1.4 and beta = 3.0.

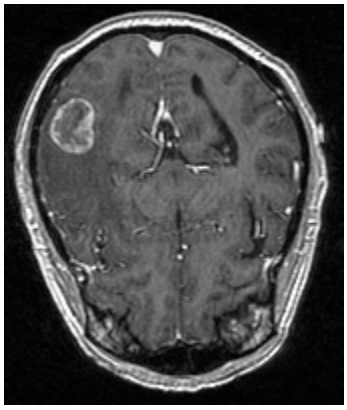
Figure 5-22 (e) shows the result of segmentation with RSM error = 0.02, iterations = 600, seed index (138, 139), initial distance = 5, propagation scale = 3, curvature scale = 3 and advection scale = 1.0. Figure 5-22 (f) shows the result of segmentation with RSM error = 0.02, iterations = 600, seed index (138, 139), initial distance = 5, propagation scale = 3, curvature scale = 3 and advection scale = 2.0. Here, increasing the advection term does not help to attract the segmentation towards the boundary of the top corner of left ventricle. With these parameters values, number of elapsed iterations is 215 with RMS change = 0.0192324, so we reduce RMS error value so that number of iterations are increased. Reducing the RMS error to 0.01 causes over-segmentation as shown in figure 5-22 (g). Reducing the advection term to 1.0 and keeping RMS error = 0.01 gives result shown in figure 5-22 (h). Figure 5-22 (i) shows that above values give better result but still it is under-segmented in that top corner of left ventricle is not completely segmented. Here, we need to increase the propagation scale so that segmentation will reach to the top left corner. Finally, using RMS error = 0.0049, iterations = 600, seed index (138, 139), initial distance = 5, alpha = -1.0 and beta = 3.0 for speed image (sigmoid), propagation scale = 6, curvature scale = 1 and advection scale = 1.0 gives result in figure 5-22 (j) where result matches very well with ventricles in original image. Figure 5-22 (k) shows the previous result with increased opacity. This result can further be improved slightly with dilation using y-radius = 1, as shown in figures 5-22 (l) and (m).

5.3.4 Laplacian Level Set Segmentation

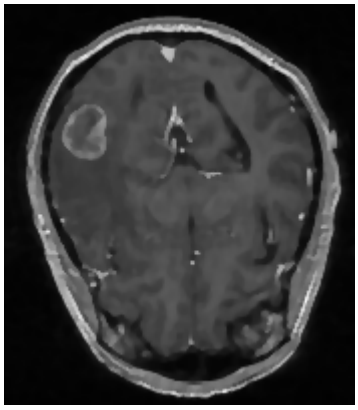
In this section, we will see how this filter can be used to improve the results of other segmentation filters such as shape detection level set filter.

In this algorithm, speed is computed as the Laplacian of the image values. The goal is to attract the evolving level set surface to local zero-crossings in the Laplacian image. Since Laplacian level set filter tends to stuck at the local edges, it is more suitable for refining existing segmentations than as a stand-alone region growing algorithm

5.3.4.1 Refining Tumor Segmentation



(a)



(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

Figure 5-23 (a) – (i) Refining tumor with laplacian level set segmentation filter

Figure 5-23 (a) shows original brain MRI slice of 256*256 and figure 5-23 (b) shows image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-23 (c) shows the speed image obtained using Laplacian, second derivative features in the image. Figure 5-23 (d) shows the result of shape detection (shown in figure 5-23 (i)) as an initial model, which we want to refine using Laplacian level set filter. Our goal is to refine this initial model from the second input (smoothed image in figure 5-23 (b)) to better match the structure represented of the tumor.

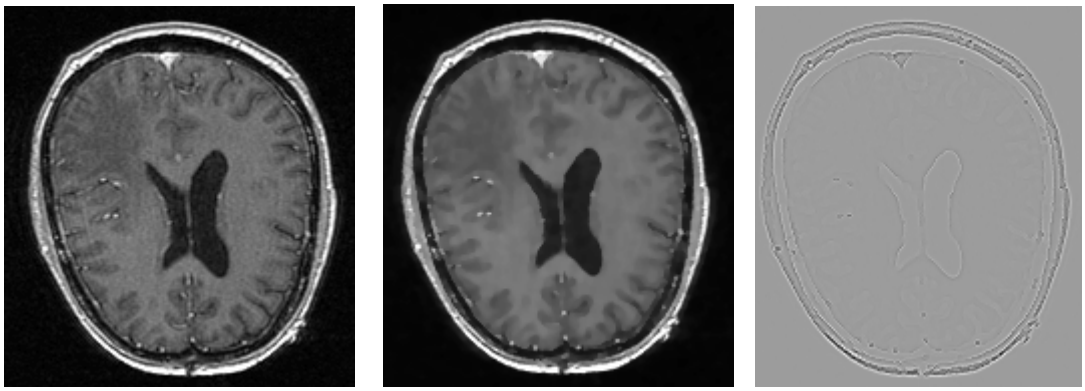
Figure 5-23 (e) shows the result of applying Laplacian on this already segmented image with iterations = 15, RMS error = 0.02, Iso-value = 127.5, propagation scale = 5, curvature scale = 15. The filter runs until convergence and RMS change is 0.0535672. Here, iso-value is the iso-value of the surface in the initial model input image. Since the initial model of the tumor is binary image, in that binary image, the isosurface is found, for example, midway between the foreground and background values. This initial isosurface should ideally be very close to the segmentation boundary of interest. The idea is that this rough segmentation can be refined by allowing the isosurface to deform slightly to achieve a better fit to the edge features of an image.

The result in figure 5-23 (e) is somewhat close to the desired segmentation (outer boundary of the tumor), but it contains jagged edges and the result should have smoother boundaries But increasing curvature scale to 25 does not smooth the result enough, as shown in figure 5-23 (f). Also increasing the propagation scale to 10 does not provide desired result as shown in figure 5-23 (g). We will increase the number of iterations so that above result propagates further to remove small notch shown on left side of the segmented tumor while smoothing the result further. Increasing the iterations to 30 improves the result significantly as shown in figure 5-23 (h). But this result contains small spurs on the center of top and the bottom

which can be removed by reducing the propagation scale slightly. Finally, result in figure 5-23 (i) shows the desired segmentation with iterations = 30, RMS error = 0.02, propagation scale = 1 and curvature scale = 25.

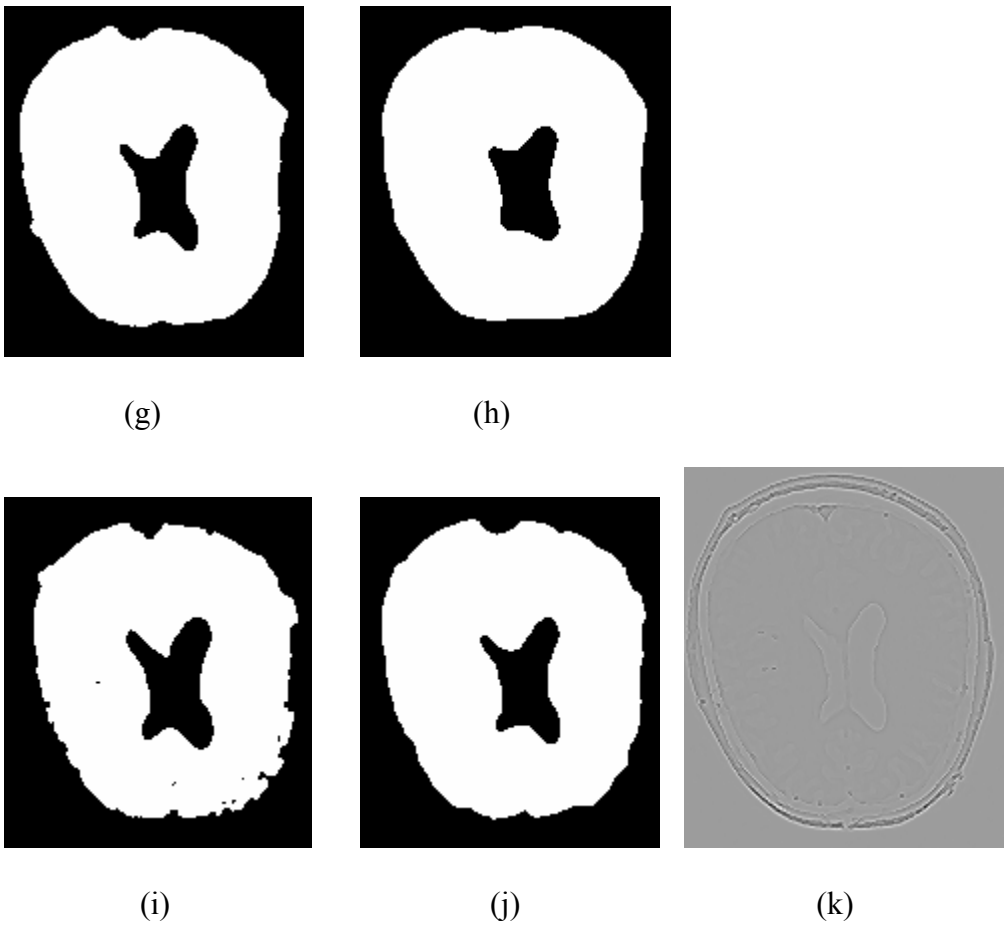
Now if we compare the results in the images of Figure 5-23 (d) and (i), we can clearly see that jagged edges are straightened and the small spurs at the upper as well as lower right-hand side of the mask has been removed. Thus Laplacian level set filter can also be very useful to refine the output of a hand segmented image.

5.3.4.2 Refining Brain Segmentation



(a) (b) (c)

(d) (e) (f)



(g) (h)

(i) (j) (k)

Figure 5-24 (a) – (k) Refining brain with laplacian level set segmentation filter

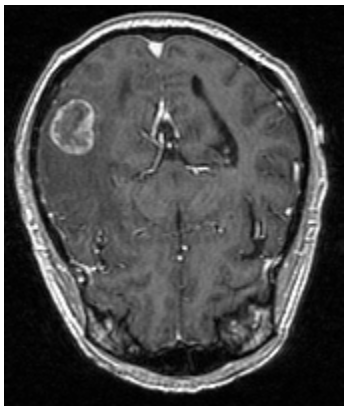
Figure 5-24 (a) show original 256 * 256 brain MRI slice and figure 5-24 (b) shows the image smoothed with curvature anisotropic diffusion with iterations = 5 and conductance = 2.0. Figure 5-24 (d) shows the initial model which needs to be refined. This initial model is the result of geodesic active contour level set filter. Figure 5-24 (e) shows the result of laplacian level set filter with iterations = 10, RMS error = 0.02, Iso-value = 127.5, propagation scale = 1, curvature scale = 40. In this result small spurs on left side of initial model are removed, but the contour in the result is not quite straightened out and needs to be smoother. Figure 5-24 (f) shows the result of laplacian with iterations = 10, RMS error = 0.02, Iso-value = 127.5, propagation scale = 0.1, curvature scale = 40. Figure 5-24 (g) shows the result of laplacian with iterations = 30, RMS error = 0.02, Iso-value = 127.5, propagation scale = 0.1, curvature scale = 8. The result contains

boundaries that are smoothed and straightened enough but there is still a spur on right side of the segmentation. If we try to straighten the boundary more to remove this spur, ventricles also get affected with increased curvature scale. This is shown in figure 5-24 (h). Thus laplacian filter was not very much useful for refining the segmentation of geodesic active contour level set.

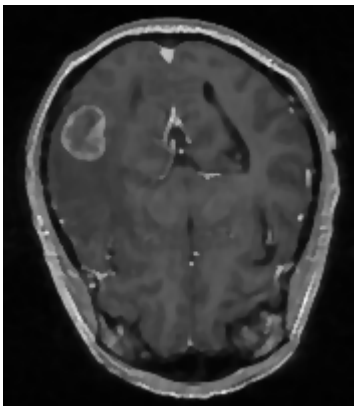
Figure 5-24 (c) shows the speed image for the result in figure 5-24 (h). But compared to the refinement of segmentation of geodesic active contour level set, laplacian on the result of shape detection level set filter gives better result. This is shown in figures 5-24 (i), (j) and (k). Here, figure 5-24 (i) is the initial model which is the result of shape detection level set segmentation filter. Figure 5-24 (j) is the result of laplacian level set with iterations = 60, RMS error = 0.01, Iso-value = 127.5, propagation scale = 0.41, curvature scale = 24. Figure 5-24 (k) shows the speed image for the result in figure 5-24 (j).

5.3.5 Canny Edge Detection Level Set Segmentation

5.3.5.1 Refining brain segmentation



(a)



(b)

(c)

(d)

(e)

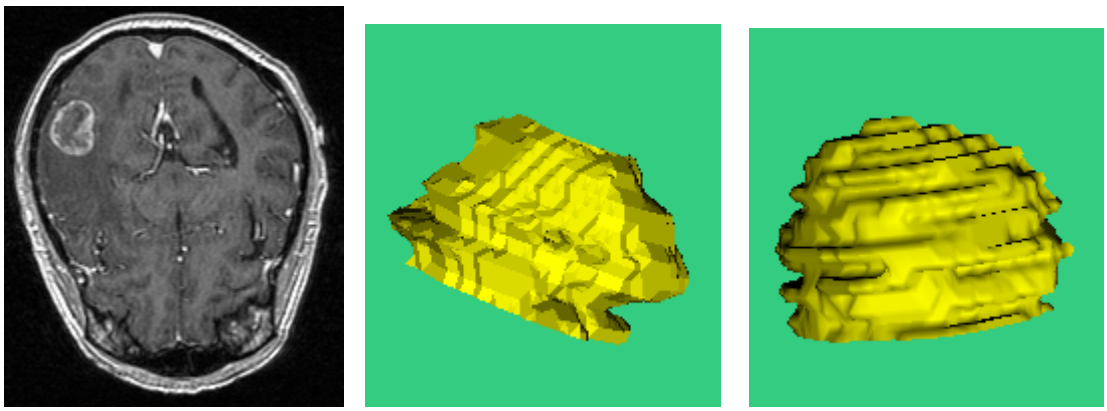
(f)

Figure 5-25 (a) – (e) Refining tumor with canny edge detection level set segmentation filter

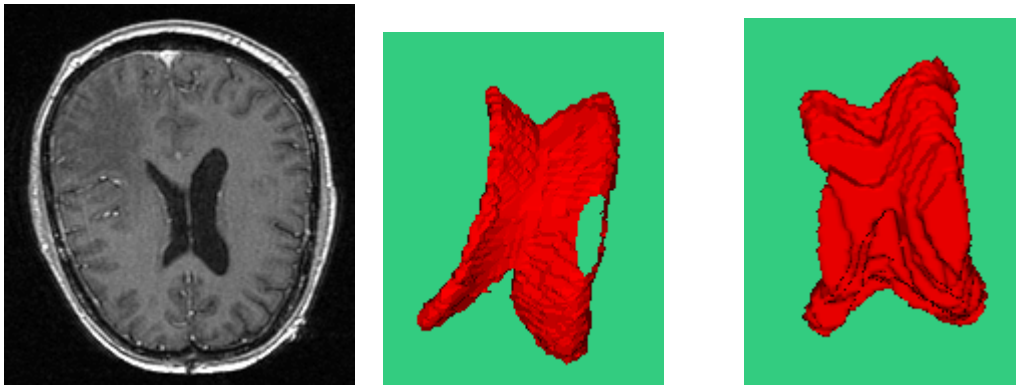
Figure 5-25 (a) shows original brain MRI slice of 256*256 and figure 5-25 (b) shows image smoothed with curvature anisotropic diffusion smoothing with iterations = 5 and conductance = 2.0. Figure 5-25 (c) shows the result of shape detection (shown in figure 5-25 (i)) as an initial model, which we want to refine using Canny Edge Detection Level Set Filter.

Figure 5-25 (d) and Figure 5-25 (e) shows the result of applying canny edge detection level set filter on initial model in figure 5-25 (c) which is to be refined.

Here, exact shape of tumor is not achieved as with laplacian level set filter. But free parameters of this filter can be adjusted to achieve a wide range of shape variations and requires a process of trial and error for finding the right parameters for this particular case.



Data - Tumor **ITK Segmentation** **Hand Contour**



Data - Ventricles **ITK Segmentation** **Hand Contour**

Figure 5-26 Volumes of Tumor and Ventricles

Figure 5.26 shows the ITK Segmented as well as hand contoured volume of ventricles and tumor. ITK segmented volumes were generated by applying ITK filters in pipeline, such as shape curvature anisotropic diffusion filter, fast marching filter, shape detection filter, dilation filter and then laplacian filter. Some of the slices needed different suite of filters to achieve desired level of segmentation in 2-D. After segmenting all tumor and ventricle slices, volume was constructed using Marching Cube algorithm for single material with linearization. Hand contoured volume was generated using slices that were segmented manually using ROI by drawing splines.

Chapter 6: Conclusion

Image segmentation plays an important role to facilitate the detection of abnormalities such as cancerous lesions in the brain MRI. Although numerous efforts in recent years have advanced this technique, there is no single approach that can generally solve the problem of segmentation for the large variety of image modalities existing today.

We have demonstrated the segmentation of brain MRI for detection of abnormalities as well as ventricles and white matter in the brain. Different basic segmentation filters implemented in ITK were used in pipeline for brain MRI segmentation. These filters included region-growing methods, watershed algorithm and various implementations of level set methods such as shape detection, geodesic active contour, threshold level set, canny edge detection level set and laplacian level set methods. Canny edge detection and laplacian methods were used to refine the already segmented objects rather than using them as standalone segmentation methods.

Region growing methods, which are intensity based methods, were useful for segmenting homogeneous structures such as ventricles but they failed to segment tumor effectively since it did not have homogeneous statistical distribution over image space.

White matter in the brain segmented with these separate and combination of these methods required post-processing using Fill Hole and dilation filters.

Watershed segmentation was effective in segmenting tumor, ventricles as well as brain area provided that edge-preserving smoothing was used with appropriate number of iterations and diffusion to remove the noise. Without smoothing, watershed segmentation proved to be very sensitive to noise and resulted in over-segmentation with too many undesirable small regions. Smoothing with less number of iterations required more computation time during segmentation and larger value of threshold.

In general, tuning the filter parameters for using watershed segmentation was a process of trial and error. The threshold parameter was used to great effect in controlling over-segmentation

of the image. Raising the threshold reduced computation time and produced output with fewer and larger regions. Hence the trick in tuning parameters was to consider the scale level (Flood Level) of the objects to be segmented. The best time/quality trade-off was achieved when the image was smoothed and thresholded to eliminate features just below the desired scale.

Level set methods were able to cross the internal boundaries of the tumor and helpful in achieving the complete size and shape of the tumor. But these methods were vulnerable for segmenting larger structures. In general, a larger number of iterations were required for segmenting large structures since it takes longer for the front to propagate and cover the structure. This drawback was mitigated by setting many seed points in the initialization of the FastMarchingImageFilter that generated initial contour which was evolved further with different implementations of level set methods. A rule of thumb for the user was to select value for initial distance (A distance from the seed points at which initial contour would be) in such a way that initial contour is close to the boundary of the object. This resulted in less number of iterations to fill and reach the edges of the anatomical structure.

In general, for level set methods, selection of seed points and appropriate values for curvature, advection and propagation scaling parameters was critical to the segmentation. The curvature term smoothes the surface of the zero set. Too low curvature weight with respect to the propagation and advection terms resulted in the expansion of the zero-set and result entered into every small protrusions of the anatomical structure. This made the zero-set prone to leaking in regions of small contrast. Using a high curvature weight prevented the level set from attempting to enter in small details: a large enough curvature could induce contractions of the zero-set.

Because the level-set equations are usually solved only at pixels near the surface, the time taken at each iteration was dependent on the number of points on the surface. The distance the surface must travel in the image dictated the total number of iterations required for the evolution of the contour. The result was also dependent on the construction of a proper speed image during the preprocessing process. Approximation to the segmentation can be achieved by Fast Marching

or region-growing segmentation and result can further be enhanced by the application of level sets.

In general, presented results showed that the application of region growing, watershed and combination of region-growing and level set techniques implemented in ITK proved to be efficient for the task of tumor, ventricles and white matter segmentation. Setting the seed points for the fast marching segmentations for initial contour generation provided good results. There was a large improvement concerning quality with level set methods, compared to using only basic segmentation techniques like region growing.

In conclusion, ITK segmentation filters offered a large number of state of the art algorithms implemented in a very consistent and meaningful coding style. With the combination of techniques discussed above, it is important to get some experience in setting the parameters correct in order to get the best results for brain MRI segmentation using the ITK filters.

References

[1] Pham, DL, Xu, CY, and Prince, JL, "Current methods in medical image segmentation," *Annual Review of Biomedical Engineering,* Vol. 2, pp. 1, 2000.

[2] Tomi Heinonena, Prasun Dastidarb, Harry Freyc, Hannu Eskolaa, "Applications of MR Image Segmentation", *International Journal of Bioelectromagnetism*, Vol. 1, No. 1, 35-46, 1999.

[3] Luis Lbanez and William Schroeder with Lydia Ng and Josh Cates, *The ITK software Guide ITK 1.4*, Kitware Inc 2003, pg 315.

[4] A. Bazzani, D. Bollini, R. Brancaccio, R. Campanini, N. Lanconelli and D. Romani, "System of Automatic Detection Of Clustered Microcalcifications In Digital Mammograms", *International Journal of Modern Physics C*, Vol. 11, No. 5, 1–12, 2000.

[5] K. Preston White, Jr., *Senior Member, IEEE*, Tonya L. Hutson, and Thomas E. Hutchinson, "Modeling Human Eye Behavior during Mammographic Scanning: Preliminary Results", *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans*, Vol. 27, No. 4, July 1997.

[6] J. Sijbers, M. Verhoye, P. Scheunders, A. Van der Linden, D. Van Dyck, E. Raman, "Watershed-based Segmentation of 3D MR Data for Volume Quantization", *Magnetic Resonance Imaging*, Vol. 15, No. 6, p. 679-688, 1997.

[7] Paul R. Hill, C. Nishan Canagarajah, and David R. Bull, "Image Segmentation Using a Texture Gradient Based Watershed Transform", *IEEE Transactions on Image Processing*, Vol. 12, No. 12, December 2003.

[8] Jagath C. Rajapakse, Member, IEEE, Jay N. Giedd, and Judith L. Rapoport, "Statistical Approach to Segmentation of Single-Channel Cerebral MR Images", *IEEE Transactions on Medical Imaging*, Col. 16, No. 2, April 1997.

[9] Faguo Yang and Tianzi Jiang*, Member, IEEE,* "Pixon-Based Image Segmentation with Markov Random Fields", *IEEE Transactions on Image Processing*, Vol. 12, No. 12, December 2003.

[10] Chenyang Xu, Dzung L. Pham, Maryam E. Rettmann, Daphne N. Yu, and Jerry L. Prince, " *"*Reconstruction of the Human Cerebral Cortex from Magnetic Resonance Images", *IEEE Transactions on Medical Imaging, Vol. 18, No. 6, June 1999 467*

[11] Brain Facts, A primer on the brain and nervous system, *Society for Neuroscience*, pg 3.

[12] Luis Lbanez and William Schroeder with Lydia Ng and Josh Cates, *The ITK software Guide ITK 1.4*, Kitware Inc 2003, pgs 315.

[13] Rafael Gonzalez and Richard E. Woods, *Digital Image processing*, second edition, Pearson Education Inc. 2003, pgs 613.

[14] Rafael Gonzalez and Richard E. Woods, *Digital Image processing*, second edition, Pearson Education Inc. 2003, pgs 617-620.

[15] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Machine Intell*., vol. 13, no. 6, pp. 583–598, June 1991.

[16] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Trans. Pattern Anal.Machine Intell*., vol. 18, no. 12, pp. 1163–1173, 1996.

[17] Luis Lbanez and William Schroeder with Lydia Ng and Josh Cates, *The ITK software Guide ITK 1.4*, Kitware Inc 2003, pgs 330,331, 332.

[18] Luis Lbanez and William Schroeder with Lydia Ng and Josh Cates, *The ITK software Guide ITK 1.4*, Kitware Inc 2003, pg 337.

[19] F. Meyer and S. Beucher, "Morphological segmentation", *Journal of Visual Communication and Image Representation,* 11, No 1:21{46, 1990}.

[20] F. Meyer, A. Oliveras, P. Salembier, and C. Vachier "Morphological tools for segmentations: connected Filters and watershed", *Annals of telecommunications,* 1997.

[21] F. Meyer, C. Vachier, "Image Segmentation based on Viscous Flooding Simulation", *Proc. of the VIth International Symposium of Mathematical Morphology*, Hugues Talbot and Richard Bear Ed., CSIRO publishing, April 2002, pp 69-77

[22] Beucher S. Yu X., Bilodeau M., "Road Segmentation by Watershed Algorithm", *Proceeding of the Workshop in Vision held in Sophia Antipolis France*, April 19-20, 1990.

[23]. S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. Proc. Int. *Workshop on image processing, Rennes (France)*, pages 17{21, Sept. 1979.

[24] F. Meyer. *Topographic distance and watershed lines*. *Signal processing*, pages 113-125, 1994.

[25] L. Najman and M. Schmitt, *Watershed for a continuous function*, *Signal Processing*, pages 99{112, Juil. 1994}

[26] J. P. Serra. "Image Analysis and Mathematical Morphology*". Academic Press Inc*., 1982, 6.4.3, 9.2.1

[27] Luis Lbanez and William Schroeder with Lydia Ng and Josh Cates, *The ITK software Guide ITK 1.4*, Kitware Inc 2003, pgs 338.

[28] Black, M. J,"Robust Anisotropic Diffusion", *IEEE Trans. on Image Proc*., 7(3),1998.

[29] Tomasi, C. and Manduchi, R., "Bilateral Filtering for gray and color images", *Proc. of the IEEE International Conference on Computer Vision*, 1998.

[29] D. Adalsteinsson and J.A. Sethian, *A fast level set method for propagating interfaces, J. Comput. Phys.*, 118 (1995), pp. 269{277.

[30] D.L. Chopp, "*Computing minimal surfaces via level set curvature flow*", J. Comput. Phys., 106, (1993), pp. 77{91.

[31] R. Malladi, J.A. Sethian, and B.C. Vemuri, “Evolutionary fronts for topology-independent shape modeling and recovery”, in Proceedings of *Third European Conference on Computer Vision, Stockholm, Sweden*, Lecture Notes in Computer Science, 800, Springer-Verlag, New York, 1994, pp. 3

[32] J.A Sethian, “Fast Marching Method”, SIAM Review, *1999 Society for Industrial and Applied Mathematics*, Vol. 41, No. 2, pp. 199–235

[33] Malladi, R., Sethian, J., and B. Vemuri. “Shape modeling with front propagation: A level set approach, *IEEE Transactions on Pattern analysis and Machine Intelligence 17(2) (1995): 158-175*

[34] R. Kimmel V. Caselles and G. Sapiro. Geodesic active contours. *International Journal on Computer Vision*, 22(1):61–97, 1997.

[35] J. Canny. A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 8, No. 6, Nov 1986.