Preface

Trondheim, June 2005

This thesis is part of the grade *Master of Technology in Computer Science* at the Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU). The thesis was prepared in cooperation with SINTEF Health Research – part of SINTEF, the largest independent research group in Scandinavia.

The authors of this thesis are Gunn Marie Navestad and Stian Dalene Gjedrem. Teaching supervisors from the Department of Computer and Information Science were Ole Christian Eidheim and prof. Richard E. Blake. Teaching supervisor from SINTEF Health Research was Frank Lindseth.

We would like to thank Ole Christian Eidheim, Frank Lindseth and prof. Richard E. Blake for valuable guidance during our work with this thesis. We would also like to thank Geirmund Unsgård and Ola Rygh at St.Olavs Hospital in Trondheim for valuable help during the evaluation of our results.

Gunn Marie Navestad

Stian Dalene Gjedrem

Abstract

We have implemented and tested segmentation methods for segmenting brain tumours from magnetic resonance (MR) and ultrasound data.

Our work in this thesis mainly focuses on active contours, both parametric (snakes) and geometric contours (level set). Active contours have the advantage over simpler segmentation methods that they are able to take both high- and low-level information into consideration. This means that the result they produce both depends on shape as well as intensity information from the input image. Our work is based on the results from an earlier completed depth study which investigated different segmentation methods.

We have implemented and tested one simplified gradient vector flow snake model and four level set approaches: fast marching level set, geodesic level set, canny edge level set, and Laplacian level set. The methods are evaluated based on precision of the region boundary, sensitivity to noise, the effort needed to adjust parameters and the time to perform the segmentation. We have also compared the results with the result from a region growing method.

We achieved promising results for active contour segmentation methods compared with other, simpler segmentation methods. The simplified snake model has given promising results, but has to be subject to more testing. Furthermore, tests with four variants of the level set method have given good results in most cases with MR data and in some cases with ultrasound data.

Contents

List of Figures

List of Tables

Chapter 1

Background

This chapter will give an overview of the background for our thesis. We will elaborate the intention and motivation for this thesis. Section 1.1 will give a definition of the problem we are going to solve. Section 1.2 will present the motivation for our work. We will explain in which context our work is supposed to be used, and how our work can contribute to improve medical treatment of brain tumours. We will then in section 1.3 and 1.4 give a short introduction to the imaging technology and types of brain tumour we have been working with. Section 1.5 presents the datasets we have tested our segmentation approaches on, and in section 1.6 we shortly arguments the choice of segmentation methods. Our work is based on a previous performed depth study of medical image segmentation [23]. Section 1.7 gives a presentation on existing work on this field and section 1.8 of how we have evaluated the results.

1.1 Problem definition

A set of methods for segmentation of neuro tumours (brain tumours) should be implemented and tested. Image segmentation is the process where image objects or areas in the image are separated from the background and each other. In the context of segmentation of neuro tumours, segmentation is the task of finding regions in the image representing the brain tumour. The focus should be on implementing and testing deformable active contours (snake- and level set algorithms), but also compare their performance with a simpler segmentation algorithm.

The segmentation methods should be tested on both ultrasound (US) and magnetic resonance (MR) images. All testdata are going to be three dimensional volumes and represent a variation of tumour types and tumour characteristics.

The goal is to agree on a selection of methods that perform well on general basis, and the methods should be evaluated based on:

- Precision of the region boundary.
- Sensitivity to noise.
- The effort needed to adjust parameters.
- Time to perform the segmentation.

1.2 Motivation

The main motivation of our work is to make a small contribution to improve method of treatment for brain tumours. One of the paramount objectives for research of treatment of brain tumours is to be able to perform surgery with minimal incisions. This will contribute to reduce the risk attended with surgery and make the operation less straining for the patient.

To elaborate the motivation of our thesis and to put our work into a greater perspective we will give a short introduction to how brain tumours can be diagnosed and treated and to the field of image guided surgery.

1.2.1 Diagnose and treatment of brain tumours

There are several different methods to diagnose and treat patients with brain tumours. The choice of method depends on the type of brain tumour and how far the brain tumour has developed. The most common methods to diagnose the patient are physical exams, neurological exams, CT-scans, MRI and skull X-ray. The patient may be treated with surgery, radiation therapy, chemotherapy or a combination of the preceding methods.

When a physician suspects that the patient may have a brain tumour he often starts with a physical and neurological exam where he among other things checks the patient's general health, coordination and reflexes. After this procedure is done he may continue with either a CT scan, MRI or skull X-ray to generate pictures of the brain. From these pictures it is possible to verify and localize a tumour. If a tumour is discovered, the physician can use these pictures to get an idea of the size of the tumour, degree of malignancy, type of tumour and other important facts to help him decide how to treat the tumour.

The most common treatment of brain tumours is surgery. The surgeon then attempts to remove as much of the tumour tissue as possible. Sometimes surgery can be difficult, it may be impossible to remove the tumour without damaging the surrounding brain tissue or the tumour may be located in an area of the brain that is unreachable for the surgeon. During surgery the surgeon can use imaging techniques to get a better view of the situation. Ultrasound images can be used to give the surgeon real-time images of the relevant organs. Radiation therapy or chemotherapy is often used after surgery to kill remaining tumour cells in the affected area or as an alternative to surgery if surgery is not possible. Radiation therapy, or radiotherapy, uses high-energy rays to kill cancer cells and reduce the size of the tumour. X-rays, gamma-ray and neutrons are the most common form of radiation source. Chemotherapy uses drugs to kill the cancer cells. The drugs follow the bloodstream into the brain. Chemotherapy may be used as a standalone method of treatment, but may also follow surgery and radiation therapy.

To monitor the patients' condition after the surgery, the imaging techniques mentioned over can be used to examine if the tumour was successfully removed. If residual tumour cells exists, it is necessary with further treatment.

1.2.2 Image guided surgery

Image guided surgery is a term used to describe surgical procedures that use medical imaging techniques like CT, MRI and Ultrasound to help surgeons navigate through a particular part of the body. This often involve that the images is used to plan a surgical

procedure, guide the surgeon during surgical intervention and to control the result of the surgery. During surgery it is desirable to view the location of surgical tools on real-time updates of the anatomy of the patient. We will now briefly outline the development of image guided surgery. A more thorough description of the field of image guided surgery can be found in Lindseth [19].

One of the first approaches used was so-called frame-based stereotactic systems. This approach is based on a technique where the patient is fastened to a special designed frame during image examinations done both prior to surgery and during the actual surgery. This ensures that the head is placed at the same position all the time, and mapping between images is an easy task. This is a very accurate procedure, but it also has some evident drawbacks. First of all, the frame interferes with the surgical procedure. There are also problems with the efficiency and lack of real-time feedback.

Frame-based stereotactic systems were gradually replaced by frame-less stereotactic systems. The frame-less approach differs from the frame-based approach in the way it integrates preoperative images with physical space and the type of tracking technology they use to follow the surgical tools that are used [19]. The traditional frame-less approach has an obvious drawback as the technique is based only on images taken before surgery, and cannot reflect changes in the anatomy of the patient that occurs during surgery.

A common way to integrate images taken during surgery with preoperative images is to transport the patient into a CT or MRI scanner a couple of times during surgery to update the images, this has the obvious drawback that the patient must be moved in and out of the scanner during surgery. The scanners can also be moved over the patient. The scanners are rather large, and this is not a very practical approach. An alternative approach is for the surgeon to operate inside the magnet of an MRI machine. This makes it possible to get almost real-time images. This approach, however, is a very expensive solution. A third approach that combines pre- and intra operative imaging is use of ultrasound during surgery. This is a more flexible and cost-effective approach, but the ultrasound images often covers only a small part of the surgical field, making it harder to get an overview of the situation. Intra operative ultrasound is often used in combination with preoperative CT- or MRI- images to get a better view of the situation. The ultrasound and CT- or MRI- images can be combined by using the intra operative ultrasound images to deform the CT- or MRI- images taken before surgery and navigation is done based on the manipulated CT- or MRI- images. Another alternative is to use the ultrasound images in a more direct manner where the intra operative images are used as maps for navigation.

During the last couple of years surgery has to a larger extent been performed with minimal invasive therapy. This leads to a increasing demand for good visualization techniques of the surgical area. The surgeon needs images showing both the surface of the organs and beyond. To help guide the surgeon in the best possible way, the desirable situation is a system which gives real-time 3D vision of the patient's anatomy.

It is still a lot of work to be done to improve the methods for image guided surgery. In our project we will focus on the process to locate and extract objects of interest from the rest of the image, and particularly focus on segmentation of brain tumours. This will help the surgeon in both the planning phase of an operation, as well as during surgery. For instance, when an operation is performed in order to remove a tumour, a good method to localize and view the tumour in real time can be very helpful for the

surgeon.

1.3 Medical Imaging Technologies

In this section we will give an introduction to two of the key technologies in the medical imaging area, Magnetic Resonance Imaging and Ultrasound.

1.3.1 Magnetic Resonance Imaging

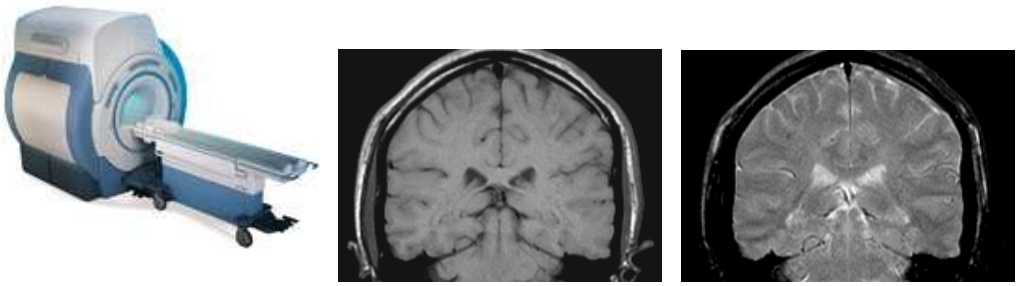
Magnetic Resonance Imaging (MRI) produces high quality images of the inside of the human body. Among many others areas, MRI is useful for diagnosing multiple sclerosis, tumours and strokes.

MRI is based on the principle of nuclear magnetic resonance (NMR), a technique used by scientists to investigate the characteristics of molecules [13]. The technique is relatively young, in 1952 the magnetic resonance phenomenon was discovered, and in 1977 the first MRI exam ever was performed. As late as 1980 there were only a handful MRI scanners in the entire United States, while there in 2003 were approximately 10,000 MRI scanners worldwide and approximately 75 million MRI scans per year performed [15].

While the first versions of MRI scanners only produced one thin slice of the body at each scan (tomographic imaging), and took hours to produce, todays MRI scanners produce volume visualizations in much shorter time.

The scanner

A modern MRI scanner can be seen in figure 1.1(a). The size of a typical MRI scanner



(a) (b) (c)

Figure 1.1: **(a)** A modern MRI scanner [2]. **(b)** T_1 and **(c)** T_2 weighted MR images of a human brain [1].

is 2 meters tall by 2 meters wide by 3 meters long. A horizontal tube, known as the bore of the magnet, is running through the scanner. The patient lies on a special table that slides into the bore. When the bodypart to be examined is in the exact isocenter of the magnetic field, the scan can begin. The magnets used in the MRI scanner varies from 0.5 Tesla to 3.0 Tesla, or 5,000 to 30,000 gauss (The magnetic field of the Earth is only 0.5 gauss).

To examine a specific part of the body, Radio frequency (RF) pulses specific only to hydrogen are directed to that part of the body. The pulses cause hydrogen protons to spin in a particular direction. Turning off the RF pulses cause the protons to return to their previous alignment relative to the magnetic field in the MRI scanner. Meanwhile, their stored energy is released, causing RF pulses to be sent out from the protons. These pulses are picked up by RF coils and forwarded to a computer system that performs calculations on the signals [15].

Image characteristics

By weightening different parameters, different MR images can be produced. Two common MR image variants are the T_1 and T_2 weighted images, see figure 1.1(b) and 1.1(c). T_1 weighted images are characterized by the dark cerebrospinal fluid (CSF) and that white matter is brighter than gray matter. T_2 images have bright CSF and gray matter is brighter than white matter.

Although the MRI technology provides images of good quality, there are some limitations: Due to noise, physiological factors, partial volume effects and non-uniform RF fields the MR signal intensity is non-uniform. Problems related to the RF fields are probably of the greatest influence, and depends on many factors, such as the subject, slice orientation, RF coil design and the pulse sequence of RF waves [22].

1.3.2 Ultrasound

Ultrasound, or ultrasonography, uses high frequency sound to produce images. Ultrasound has a wide area of applications, among these are diagnosing tumours, analyzing bone structures and examining a foetus inside the uterus.

During the period 1910-1930 sonar and radar, equipment that use sound waves to detect objects that cannot be seen, was developed. Later this technology found its way to the medical area, in 1942 Dr. Dussik first used ultrasound to locate brain tumours [9].

The ultrasound machine

A modern ultrasound machine can be seen in figure 1.2(a). Ultrasound waves are mechanical waves with a frequency above what humans can hear. Sound frequencies used in medical ultrasound are in the range of approx. 2 to 10 MHz (or wavelengths in the range of approx. 0,15 to 0,75 mm), while frequencies hearable by humans lie in the range of 20 to 20 000 Hz [14].

An ultrasound machine uses the transducer probe to emit the high-frequency sound pulses into the body. When encountering a boundary between two tissues that conduct sound differently, some of the sound energy get reflected, causing an echo. Other sound waves travel further, and some of these again get reflected back. The transducer probe picks up the echoes and passes the incoming information on to a computer. The computer then calculates an image based on knowledge of the speed of sound in tissue along with the time it takes for each echo to return to the probe.

Image characteristics

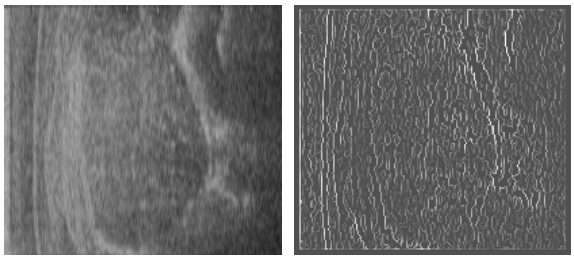
An example of an ultrasound image can be seen in figure 1.2(b). An ultrasound machine can operate in different modes: The standard mode is the B-mode (B for brightness).

(a) (b)

Figure 1.2: **(a)** A modern ultrasound machine [4]. **(b)** Ultrasound image of intracranial anatomy and facial features of a foetus [3].

The stronger the echo is, the brighter the image. Among other modes are the M-mode (M for motion), which works in a similar way as an echo sounder; one axis of the image represents image data while the other represents time. The Doppler effect can also be used to calculate blood flow to and from the probe. 3D ultrasound is a relatively new technique that combines multiple 2D ultrasound images to produce 3D images.

The lateral resolution (perpendicular to the sound waves) is poorer than the radial resolution (parallel to the sound waves). Also, a typical ultrasound image is relatively noisy, with a high influence of *ultrasound speckle*. The speckle is often modelled as multiplicative noise, since its magnitude is proportional to the signal strength. This leads to ultrasound images with significant noise even in bright areas. Furthermore, extraction of edges in an ultrasound image is not an easy task, this is illustrated in figure 1.3. In this figure a simple Laplacian of Gaussian (LoG) operator detects the edges surrounding the object of interest correctly, but a large number of incorrect edges are also produced [30].



(a) (b)

Figure 1.3: Edge detection in ultrasound images. **(a)** Original image **(b)** Edge image (Laplacian of Gaussian with $\sigma = 1.0$) [30].

1.4 Brain tumours

A tumour is a mass of excess tissue that results from abnormal cell division [17]. Brain tumours are often characterized to be either benign or malignant. In this section we will try to explain the differences between these two types of tumour. We will also try to explain the difference between the two main types of brain tumours: primary brain tumours and metastatic brain tumours. Furthermore, we will give examples of the most common primary brain tumours. For a more thorough description of the brain and brain tumours, see [10], [5], [17], [24] and [11].

1.4.1 Benign and malignant brain tumours

A brain tumour can be characterized as either benign or malignant. This will indicate the aggressiveness of a brain tumour.

A benign brain tumour is growing slowly. The tumour cells do not usually invade surrounding tissue, and the tumour has rather distinct borders. A benign tumour does not contain cancer cells and is often less threatening than malignant brain tumours, but it can do a lot of damage if it puts a lot of pressure on other brain tissue.

A malignant brain tumour is a more aggressive tumour than a benign tumour. It consists mostly of cancer cells. Malignant tumours infect surrounding brain tissue and are more difficult to delimit. It usually grows more rapidly than a benign tumour and is in most cases life-threatening.

The World Health Organization (WHO) has developed a classification system to indicate the degree of malignancy. A brain tumour can be classified as grade I, II, III or IV. Grade III and grade IV tumours are, by definition, malignant. Grade IV is the most malignant tumour. It can often be hard to determine which grade a tumour has, specially if it is on the margin between two tumour grades.

1.4.2 Metastatic brain tumours

When the cells in a tumour are rapidly growing and infect healthy tissue, they are called cancer cells. Sometimes, when cancer cells are growing, it happens that some cells detach from the tumour and travel to other parts of the body. In most cases the cancer cells penetrate through the connectiv tissue and into the bloodstream. If the cancer cell survives the trip, it takes home in healthy tissue in other parts of the body. What kind of tissue the cancer cell attacks is often dependent on the type of cancer cell [24]. The source of metastasis in the brain is often cancer from the lung, breast, colon or skin. A metastatic brain tumour is malignant, due to the fact that the cells are cancer cells.

1.4.3 Primary brain tumours

A primary brain tumour starts in the brain. A primary brain tumour can be benign or malignant. A primary brain tumour is named according to the type of cells or the part of the brain in which it begins. The rest of this subsection will give an overview over the tumours we have worked with.

Gliomas

This is the most common type of brain tumour. The tumour starts in the glial cells. Glia is the supportive tissue which helps keep the neurons in place and functioning well [5]. Gliomas has diffuse infiltrating nature, this makes total surgical removal difficult. The most common types of gliomas are:

- **Astrocytoma:** This tumour arises from astrocytes (star-shaped glial cells). Astrocytomas can further be divided into pilocytic-, low grade-, anaplastic astrocytoma and glioblastoma depending on degree of malignancy. Pilocytic astrocytoma is the least malignant tumour (grade I) and glioblastoma is the most malignant tumour (grade IV).

- **Oligodendroglioma:** This tumour arises from oligodendrocytes (glial cells with shape like a fried-egg). Oligodendrogliomas can be of degree II and III and the most common location is the cerebral hemisphere.

- **Ependymoma:** This tumour arises from ependymal cells. Ependymoma can have degree I - IV, and is a relatively rare tumour.

Meningioma

Meningioma arises in the meninges (cortex) and is a slowly growing tumour with distinct borders. When the tumour grows, it puts pressure on surrounding brain tissue and can cause displacement of the tissue. Because meningioma is growing very slowly, the brain can adjust the growth, and the tumour can grow into the size of an orange before it is discovered.

Acoustic Neuroma

Acoustic Neuroma, also called Schwannoma, is located in the angle between the cerebellum and the pons in the back of the skull [5]. The tumour arises from a schwann cell, and affects the nerve of hearing. The tumour is benign, and usually grows very slowly.

1.5 Datasets

In this section we will provide a description of the five datasets that we have tested the segmentation filters on. Information about the data format will also be included.

1.5.1 Data format

All datasets are in the MetaHeader data format, that consists of two parts: the image data itself and a header with meta data. Typical information in the header is image dimensions, data type, orientation and position. These two parts may either be located in the same file or in two separate files. All the datasets we have used in this thesis are of type 8 bit unsigned char, with grey level intensities from 0 to 255.

1.5.2 Composition of a dataset

All of the datasets consist of two series with MRI and US data, where the series are aligned with each other. Each series consists of data that is sorted by the time the data was recorded. When we in this thesis refer to "first MRI" or "first US" we therefore mean the first MRI or the first US recorded of the patient, respectively. The difference between the two series is that in one of them MRI is used as master and in the other one US is master. The meaning of "master" is that either the MRI or the US data has been used as basis for the alignment, or registration[1] process. The series with MRI as master typically contain data for the entire head, while the others typically contains data surrounding the area of interest. Hence, the size of the tumours in the MRI series are typically much smaller than the US series, thus leading to a shorter computation time. In most cases we have used the US series to get as good resolution of the tumour as possible, but in some cases where the tumour is very large we have used data with MRI as master to speed up the computation time.

1.5.3 Extraction of subsets

Another way of decreasing the computation time needed for the segmentation filters is to extract subsets of the data. A subset should typically include the tumour along with some surrounding tissue. This way the filters only have to process a relatively small part of the image data, without loosing any resolution of the tumour itself, as the case is with using data with MRI as master. In addition, this method does not prevent the filter parameters from being used on the entire dataset afterwards since the conditions are the same.

1.5.4 Description of the datasets

In this section a short description of the five datasets will be provided. Each dataset is illustrated with an axial[2] , coronal[3] and sagittal[4] slice from the series of the dataset with MRI as master.

Dataset N241

The tumour in this dataset is a *Metastatic* brain tumour (see 1.4.2), and its location is seen in figure 1.4. The tumour has an intensity distribution that makes it relatively easy to distinguish from the surrounding tissue. However, we see that it lies close to other brain structures that makes the segmentation process more difficult.

Dataset N351

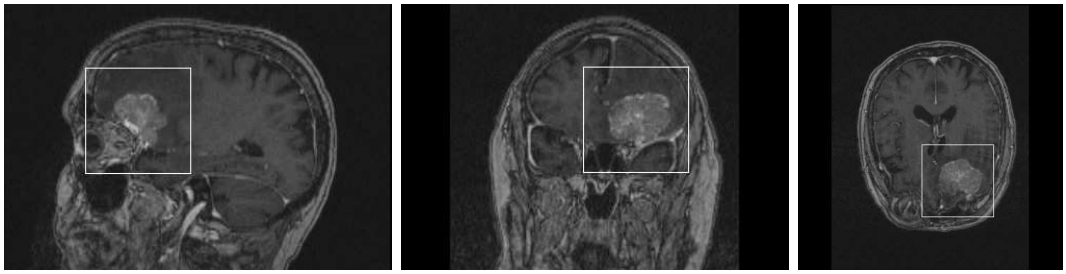
This dataset contains a tumour of type *Acoustic Neuroma* (see 1.4.3). Its location is seen in figure 1.5. This tumour is probably the easiest one to separate from the surrounding tissue due to the bright and uniform intensity distribution.

[1] Registration is the task of mapping one data set onto another. This can be done using position information from navigation equipment or from using image processing filters, among others.

[2] An axial slice is a horiozontal cross section through the head (when the patient is standing)

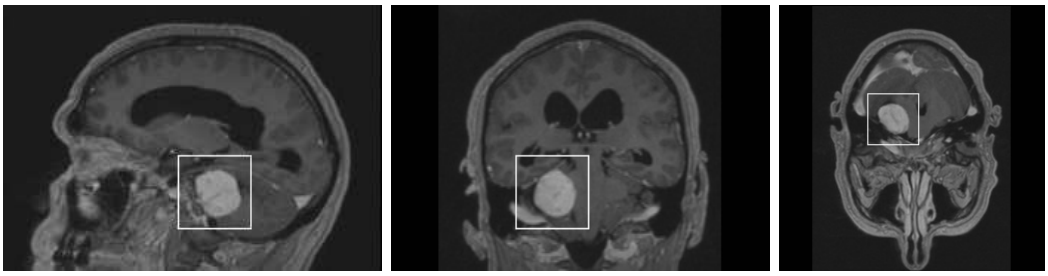
[3] A coronal slice is a vertical cross section, parallell to the face (when the patient is standing)

[4] A sagittal slice is a vertical cross section, perpendicular to the coronal cross section.



(a) (b) (c)

Figure 1.4: Dataset N241 (size: 373 × 373 × 255) with highlighted subset (size: 102 × 102 × 102). **(a)** Sagittal, **(b)** coronal and **(c)** axial slice of the dataset.

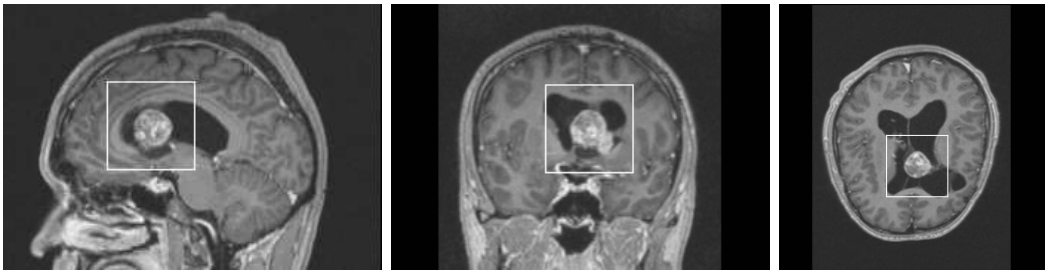


(a) (b) (c)

Figure 1.5: Dataset N351 (size: 256 × 256 × 179) with highlighted subset (size: 50 × 50 × 50). **(a)** Sagittal, **(b)** coronal and **(c)** axial slice of the dataset.

Dataset N359

The location of the tumour, which is of an unknown type, is shown in figure 1.6. Although the tumour lies relatively free from the surrounding tissue, its non-uniform intensity distribution makes the segmentation process more difficult.

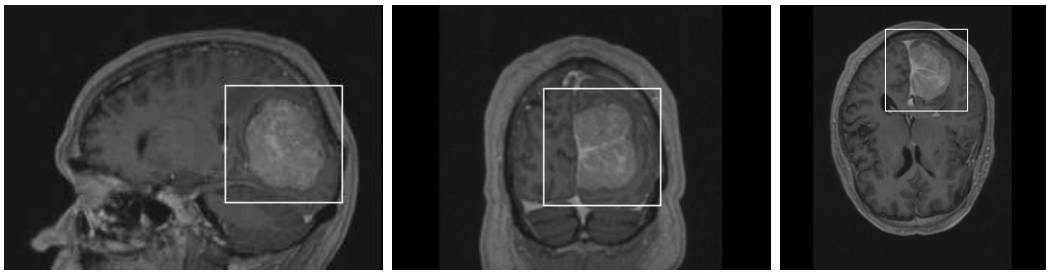


(a) (b) (c)

Figure 1.6: Dataset N359 (size: 256 × 256 × 179) with highlighted subset (size: 60 × 60 × 60). **(a)** Sagittal, **(b)** coronal and **(c)** axial slice of the dataset.

Dataset N360

The tumour in this dataset is of type *Meningioma* (see 1.4.3) and the dataset is shown in figure 1.7. We see that the intensity distribution of this large tumour is relatively non-uniform and quite similar to the surrounding tissue, making the segmentation task more difficult.

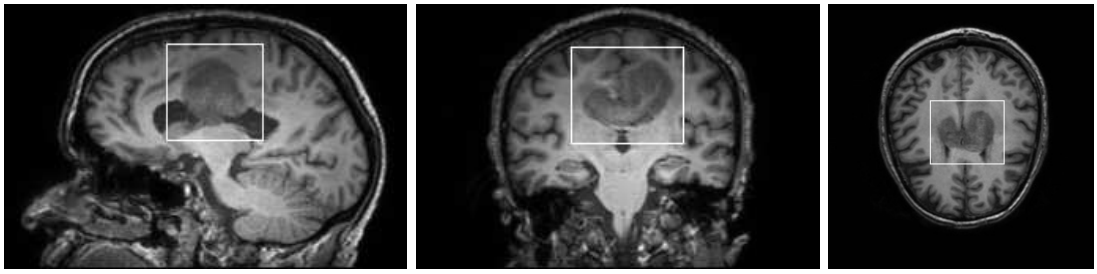


(a) (b) (c)

Figure 1.7: Dataset N360 (size: 256 × 256 × 179) with highlighted subset (size: 80 × 80 × 100). **(a)** Sagittal, **(b)** coronal and **(c)** axial slice of the dataset.

Dataset N378

This dataset contains a tumour of type *Glioblastoma*, which belongs to the Gliomas category (see 1.4.3). The location of the tumour is shown in figure 1.8. With its dark and diffuse characteristics, this tumour is probably one of the most difficult cases among the datasets we have available.



(a) (b) (c)

Figure 1.8: Dataset N378 (size: 256 × 256 × 168) with highlighted subset (size: 72 × 72 × 62). **(a)** Sagittal, **(b)** coronal and **(c)** axial slice of the dataset.

1.6 Choice of segmentation method

Our work is based on a literature study carried out prior to this thesis [23]. In the literature study we examined existing segmentation approaches and to what extent they have been tested within medical image segmentation, and in particular segmentation

of brain tumours. We considered active contours to be the most suitable segmentation methods for our usage. Active contours are capable of combining both knowledge of greylevel intensities in the image as well as apriori knowledge about size, shape and location. This is a desirable quality because medical image data often is very noisy, this makes segmentation based solely on greylevel intensities difficult. We found many promising results from use within medical segmentation in the literature.

The first step in an active contour algorithm is to let the user define a starting point for the algorithm. This can for instance be a rough boundary of the object of interest, or a seed point inside it. The contour is evolving through time, the evolution is described through minimization of a mathematical formulation. The formulation govern which image qualities the contour strain to find, and the contour shape which is most preferable. The algorithm converges towards a local minimum in the image, i.e. the contour evolves to a position in the neighbourhood of the initializing contour that corresponds best to the properties described by the mathematical formulation.

There are two main groups of active contours: **Snakes** and **level set**. We have tested the performance of one snake formulation and four level set formulations. In addition, we have compared their performance with region growing segmentation.

1.7 Existing work

We have based our work on the deformable active contour models developed from the snake model presented by Kass [18] and the level set approach presented by Osher and Sethian [25]. Various active contour algorithms have previously been tested on segmentation of different biological structures such as the liver, blood vessels, and structures from the brain. Chan et. al have used a refined version of the snake method presented by Kass [18] to find the boundaries of a brain tumour in a MR image. They achieved very good results on this particular tumour.

Yim et. al. [32] compare the performance of two snake approaches and one watershed approach on segmentation of a MR image of metastasis (a tumour type) in the brain. They found that the active contour approach gave good results in this particular case.

There are many of examples from successful use of active contours for segmentation of other structures than the brain. Chang et. al [8] used a 3-D snake to extract breast tumour in ultrasound images. They achieved fairly good results, but experienced some problems due to the high level of noise in the image.

We have not been able to find any existing work where the behaviour of various active contour algorithms used on various tumour types are compared. The active contour approach are only tested on ultrasound images to a limited extent.

1.8 Evaluation of results

To be able to measure the quality of a segmentation algorithm, we need a good procedure for evaluation of the results. In the ideal situation we would have access to the "ground truth", a perfect segmentation to compare the results with. The "ground truth" segmentation is usually generated from a constructed image, where the exact border of the object of interest is known in advance. Before the segmentation is performed, noise is added to the image. If a perfect segmentation is not available, as often is the

case, a manually expert segmentation of the datasets can be used as reference. If access to a perfectly segmented or a manually segmented image can be provided, all results can be compared to this and it is possible to get a numerical measure on how good each segmentation result is. Udupa et. al. [29] suggested a methodology for evaluating the performance of image segmentation algorithms. They take three factors into account:

- **Precision:** the reproducibility of the segmentation.
- **Accuracy:** the degree of resemblance with the truth (i.e. the "ground truth").
- **Efficiency:** the time taken to run the algorithm.

It is a very time consuming task to perform manually segmentation of images, particularly with three dimensional images. During our work we have not been provided with segmentation results we can use as reference when evaluating the performance of the segmentation methods. This makes the task of evaluating the results very difficult. A selection of our results has been evaluated by a group of experts consisting of two surgeons and one researcher. Expert 1 is the head surgeon of the neuro clinic at St. Olavs hospital and professor II at NTNU. He has also produced many publications regarding ultrasound and navigation during neuro surgery. Expert 2 is assistant doctor at the neuro clinic at St. Olavs hospital and a PhD student. He has also participated in research in the ultrasound area. Expert 3 is a researcher at SINTEF Health Research, and has published work on image guided surgery.

The criteria for evaluation are listed in table 1.1. For the first five and the last criteria each member of the expert team has graded the results (setting points between 1 and 5) based on their opinion of the quality. 1 corresponds to very good quality of the result, 5 to very low quality. The second last criteria is a true-false evaluation. Appendix A contains the evaluation form handed out to the expert team prior to the evaluation.

Criterion	Description
Misplacement	Does the resulting boundary not approximately correspond to the boundary tumour.
Oversegmentation	The amount of pixels not belonging to the tumour, being classified as tumour tissue.
Undersegmentation	The amount of pixels belonging to the tumour, not being classified as tumour tissue.
Wrong edges traced	The amount of the final contour following edges in the image not belonging to the edges of the tumour.
Details missing	The amount of details missing in the image, i.e. the contour is not able to follow convexities, bumps or other minor details of the tumour boundary.
Is the result usable?	Is the results usable for a surgeon?
Manual adjustment required	The degree of manually segmentation is required to get a usable result (if the answer is no on the last question).

Table 1.1: Evaluation criteria

Because we have limited opportunity to evaluate the results and most of the evaluation is based on our subjective opinion, it is difficult to compare the performance of the various segmentation methods. We have therefore focused on trying to extract strengths and weaknesses with each method separately. The objective is to find out when a particular method can be used, and to agree on one or more methods that is producing good results on a regular basis. We will therefore look at one and one method at a time. First, we will present the theoretical background of the method, followed by a description of the implementation and finally the results of running the method on the datasets we have been provided with by SINTEF.

All of the segmentation methods we have tested was run on cropped versions of the original datasets provided by SINTEF. The cropping was done in order to speed up the computation time.

Chapter 2

Software design

In this chapter we will present the design behind our segmentation software. We will also give an introduction to the various tools used in the implementation of the software. The purpose of this chapter is to make our solution easier available for further usage.

2.1 Requirements

The software should be designed in such a way that is easy to integrate the application or part of it into other applications. More specifically, the software should be designed so it can be integrated into the CustusX software currently being developed by SINTEF (see section 2.3.3) This means that the code should be structured in packages and modules that easily can be replaced or reused. As a consequence of this, all segmentation filters should be designed in such a way that they all have a similar structure and are independent of the GUI. Furthermore, the software should seek to make use of other, existing imaging software as much as possible. This way the software does not have to include basic imaging functionality from scratch, thus saving a significant amount of time during the development.

2.2 Top-level design

Our code is structured in two main modules: Segmentation and Visualization. As seen in figure 2.1, the segmentation part is separated from the visualization part. This is done in order to make our software more suitable to be included in software developed by others.

2.2.1 The segmentation module

All segmentation filters are currently implemented as stand-alone programs that take inputs from the command line, but the filter classes are designed to be included in other classes as part of a larger application. The different segmentation filter classes are shown in figure 2.2.

To be able to execute one of the segmentation filters it is necessary to set input- and output filenames through the methods SetInputFilename() and SetOutputFilename(). It is also necessary to set other parameters specific to the particular filter. After all the parameters are given, the filter is executed through the method Run().

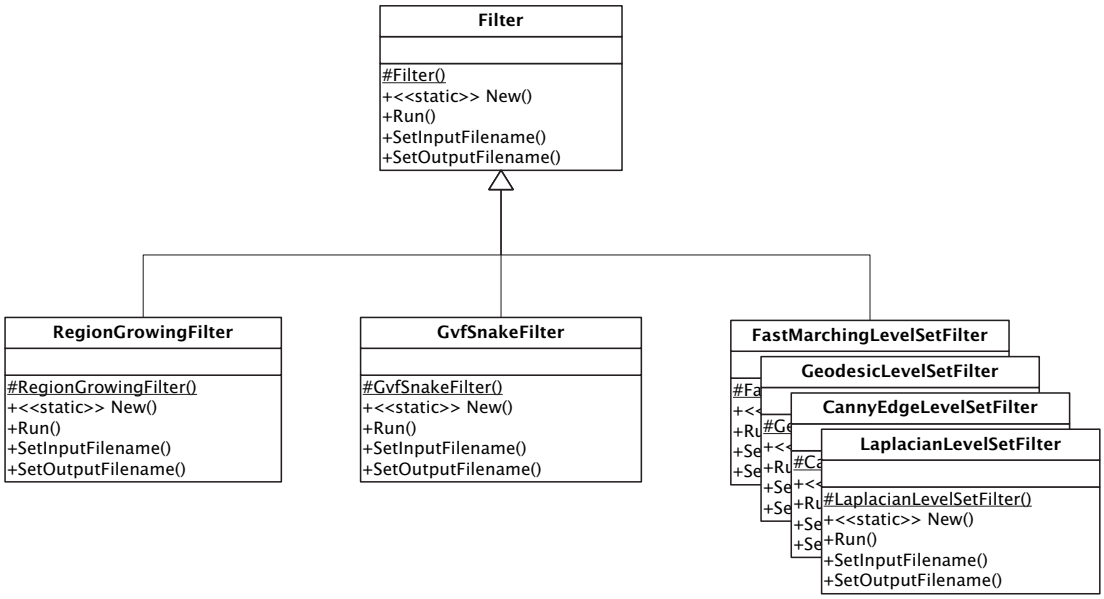


Figure 2.1: A simple top-level view of the visualization and segmentation parts of our software.

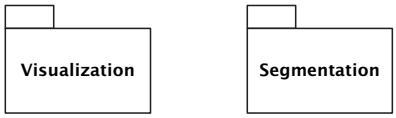


Figure 2.2: The different segmentation classes.

2.2.2 The visualization module

To present results from segmentation filters we needed some sort of visualization software. Although there exists many commercial as well as freely available visualization software, we couldn't find any that fully suited our needs. In particular, we needed software that could be helpful during the documentation of the segmentation results in this thesis. This included features such as extracting slices from the dataset with the segmentation result overlayed, a feature missing in most existing software such as CustusX.

We have implemented a basic visualization tool and sketched out a more advanced visualization tool. The basic visualizer is not directly connected with the filters. It reads the original data and the result from the segmentation from file and visualize them together. The advanced visualizer is supposed to offer a graphical user interface for the filters. This is not implemented yet. We will describe the details of the design in chapter 7.

2.3 Software tools

In this section we will give a brief presentation of some of the software we have used in our implementation. We will also give a short presentation of CustusX which our application is intended to be integrated with in the future.

2.3.1 ITK

The Insight Segmentation and Registration Toolkit (ITK) is an open-source, object-oriented software library for image processing, segmentation and registration. ITK is mainly developed to be used within medical image processing, and it is developed by three commercial (Kitware, GE Corporate R&D, and Insightful), and three academic (UNC Chapel Hill, University of Utah, and University of Pennsylvania) organizations [16]. The toolkit is meant to be used for commercial, research and teaching purposes. This section will give a brief introduction to ITK, for a more thorough description see the itk software guide [16]. Throughout all of this thesis, we refer to all classes in ITK through writing itk::classname.

Essential system concepts

In order to use ITK classes with different data types, the toolkit is built on a template programming mechanism. This mechanism allows classes to be initialized with parameters of unknown type, and methods to take parameters of unknown type etc. The ITK library is therefore organized in *containers* to hold data, *iterators* to iterate through the data, and *generic algorithms* that use the *containers* and *iterators* to make efficient algorithms.

Another important concept in ITK is the use of object factories. This enables the user to be able to control run-time initializations of classes.

To ensure a more effective use of memory, ITK is based on a concept called reference count. Every reference to an instance is counted, when the instance no longer has any references to it, it destroys itself. This way we ensure that data is not kept in memory longer than necessary and that no data is removed too early. In ITK this functionality is represented in the class itk::SmartPointer. This class contains functionality to count references made to an instance (in the Register() method) and to delete an instance from memory (in the Unregister() method).

Data representation

In ITK, data is represented as either an image or a mesh. The itk::Image class represents a multidimensional regular sampling of data where the sampling direction is parallel to the coordinate axes. The itk::Mesh class represents an adaptive, evolving structure consisting of points and cells. A mesh can contain information of structure. An itk::PointSet is a simpler version of the itk::Mesh class, and it can be used to represent a point cloud.

Data Processing Pipeline

A data object in ITK is passed through a pipeline of process objects. There are three types of process objects in ITK: *sources*, *filter objects* and *mappers*. The source objects produce data (typically reads from or writes to file), the filter objects take in, and

processes a data object to produce new data and the mapper objects are used to accept data to output.

All process objects have a method Update() that makes the filter "up-to-date". If the filter is taking the output from another filter as input, the Update() method checks to see if this filter has produced any output yet. If not, Update() is called on this filter. When Update() is called on the last process object in a pipeline, this can trigger execution of all process objects in the pipeline. An example of an ITK pipeline can be seen in figure 2.3.

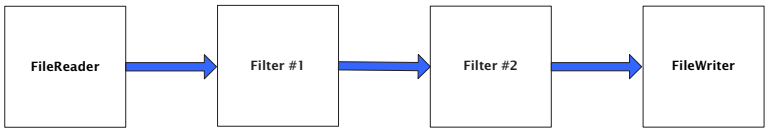


Figure 2.3: Example of an ITK pipeline.

2.3.2 VTK

The Visualization Toolkit (VTK) is an open source, object oriented toolkit designed for 3D computer graphics, image processing and visualization. The graphics model in VTK is at a higher level than other rendering libraries like OpenGL. VTK offers an easy and fast way to create graphics and visualization applications. It is however not a particularly fast graphics engine.

VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modelling techniques like implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. For a complete description of the toolkit we refer to the Visualization Toolkit Users Guide [16].

We have used VTK in our implementation of the visualization part of our application.

2.3.3 CustusX

CustusX is an application currently under development by SINTEF for managing medical data. It offers functionality for visualizing several data volumes simultaneous and it is employed to track and visualize the location of surgical tools.

We chose not to integrate our application directly into CustusX because it is not completely developed yet. In addition we need additional functionality for visualization than what is currently offered in CustusX.

Chapter 3

Preprocessing

In this chapter we will give a presentation of how images may be preprocessed in order to extract and enhance important information.

Figure 3.1 shows an overview of typical tasks that are performed prior to segmentation. First, smoothing is performed in order to remove noise from the input image. Next, an edge-extraction filter extracts edges of interest. Finally, specific ranges of intensities in the image are enhanced.

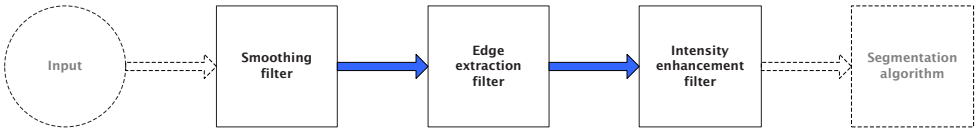


Figure 3.1: A typical preprocessing pipeline.

3.1 Smoothing

The amount of noise present in an image may vary, depending on the type of image source as well as many other parameters. MR images typically contain low amounts of noise, while ultrasound images typically have great amounts of noise. Smoothing an image may very well remove the noise, but often other information such as e.g. edge information is lost. Fortunately, there exists methods that both smooth an image and preserve most of the important information in the image as well.

3.1.1 Simple smoothing

One of the simplest ways of smoothing an image is to convolve an image with a Gaussian. For a 2D image the Gaussian operator is:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}},$$

where x, y are coordinates in the image and σ is the standard deviation [28]. Computers usually use a discrete approximation to the continuous Gaussian distribution. The

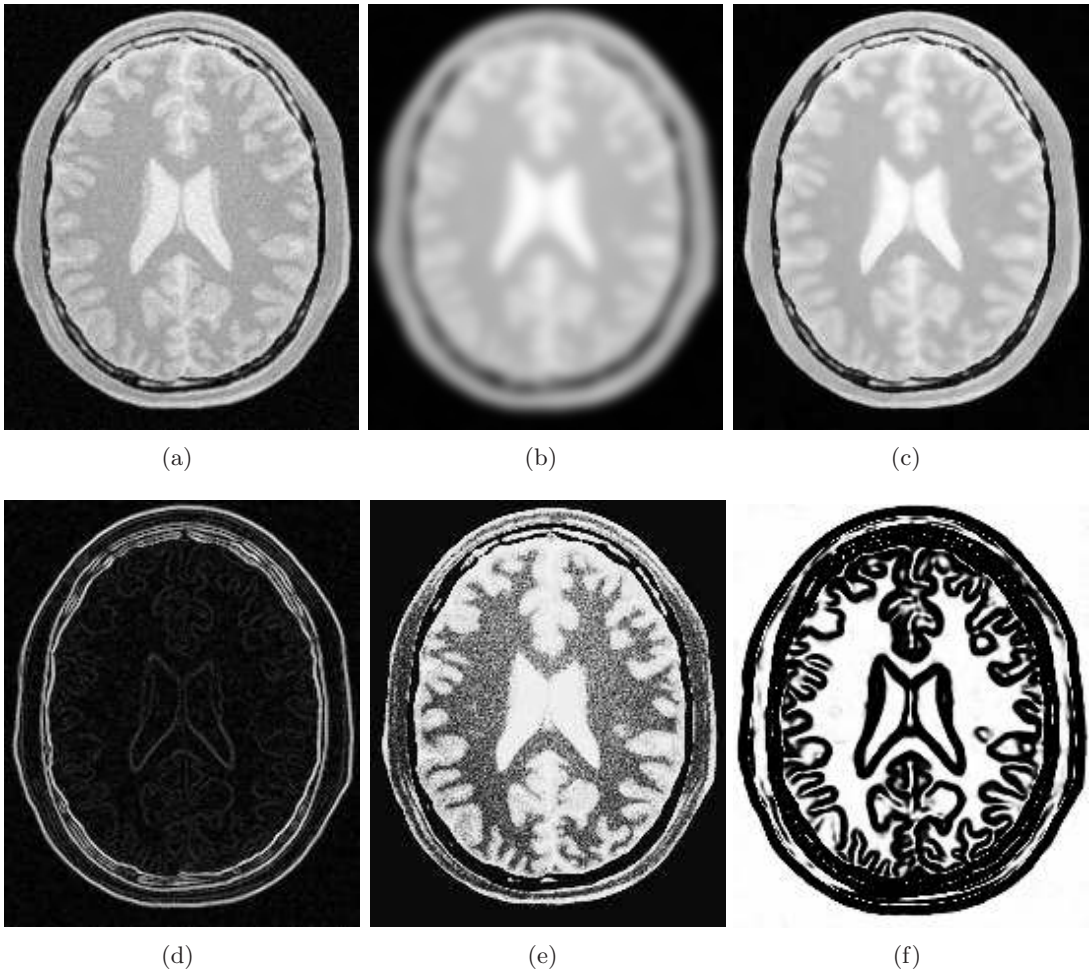


Figure 3.2: Results of different preprocessing steps **(a)** Original MRI slice of a human brain **(b)** Simple Gaussian blurring **(c)** Anisotropic diffusion blurring **(d)** Edge information extracted using gradient filter **(e)** Result of sigmoid filtering **(f)** Result of sigmoid filtering the gradient image (d) [16].

following matrix is an example of this:

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

The advantages of this type of smoothing are that it is a fairly fast and easy to implement technique, but it does not take edge information into consideration. Therefore, such information is lost during the process. Figure 3.2(b) is an example of Gaussian smoothing (with σ=3).

3.1.2 Edge-preserving smoothing

A more sophisticated and useful technique is to vary the amount of smoothing, depending on other information present in an image. One way of accomplishing this is to minimize smoothing close to strong edges in the image and maximize the smoothing elsewhere. *Anisotrophic diffusion*, introduced by Perona et al [26], is a solution to this problem. As opposed to Gaussian blurring, anisotropic diffusion take edge information into consideration.

The anisotropic diffusion uses a *conductance term* to vary the amount of smoothing of each pixel in an image. There are many ways to define such a term, but a desirable characteristic is that the term should be equal to 0 on edges and 1 in regions. The implementation of anisotropic diffusion in ITK uses the following conductance term:

$$C(\mathbf{x}) = e^{-(\frac{||\nabla U(\mathbf{x})||}{K})^2},$$

where $\mathbf{x}$ is an N-dimensional array representing pixel values in an N-dimensional image and K is a constant. The conductance term is then used as a guidance to a smoothing algorithm, resulting in an image with edge information preserved. For an example of anisotropic diffusion, see figure 3.2(c) (time step is 0.25 and number of iterations is 5).

The edge-preserving filters used in our implementations, are the itk::GradientAnisotropicDiffusionImageFilter and the itk::CurvatureAnisotropicDiffusionImageFilter. These provide similar functionality, but the main difference is that the last variant uses a more robust variant of the diffusion equation. This makes this variant less sensitive to contrast and it also preserves finer detailed structures.

3.2 Edge extraction

The position of edges in an image and their strength is information that is very helpful during segmentation tasks. The job of a gradient filter is to extract this edge information. One way of accomplishing this is to convolve the image with a binary mask. For a 2-D dimensional image the followings matrices are examples of such masks:

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}.$$

An example of edge extraction using a gradient filter is shown in figure 3.2. The edge-extracting ITK-filters we have used in our implementations are the itk::GradientMagnitudeImageFilter and the itk::GradientMagnitudeRecursiveGaussianImageFilter. They provide very similar functionality, but the main difference is that the latter smooths with a Gaussian kernel prior to the edge-extraction process.

3.3 Intensity enhancement

The purpose of the sigmoid filter is to enhance specific ranges of image intensities and thus making the segmentation task easier. The filter works by using a *non-linear mapping* between input and output image intensities. The mapping is given by the sigmoid equation

$$I_{out} = (max - min) \cdot \frac{1}{1 + e^{-\left(\frac{I_{in}-\beta}{\alpha}\right)}} + min,$$

where I_{in} denotes an input intensity and I_{out} denotes an output. max and min denote the maximum and minimum intensity of the output image, respectively. The *alpha* (α) and *beta* (β) parameters alter the intensity window of the mapping. As shown in figure 3.3, adjusting the alpha parameter will change the width of the intensity window, while changing the beta parameter will alter the center of the intensity window. Another useful feature with the sigmoid filter is that negative alpha values inverts an image. An example of use of the sigmoid filter can be seen in figure 3.2(e) ($min = 10$, $max = 240$, $\alpha = 10$ and $\beta = 170$) and 3.2(f) ($min = 0.0$, $max = 1.0$, $\alpha = -0.5$ and $\beta = 3.0$). The sigmoid filter provided by ITK is the itk::SigmoidImageFilter.

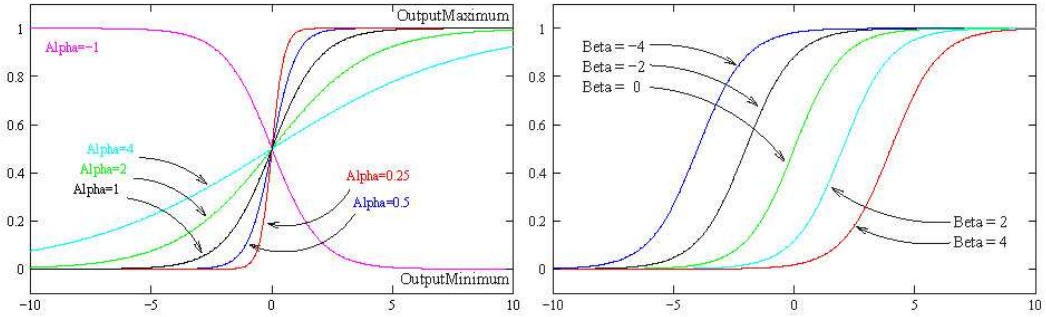


Figure 3.3: How the *alpha* and *beta* parameters affect the non-linear mapping in the Sigmoid filter [16].

Chapter 4

Region growing segmentation

4.1 Theory behind region growing

The basic principle behind a region growing algorithm is to let a region grow from a seed point within the region of interest. To expand the current region, the pixels adjacent to the region are examined, and the ones similar to the pixels already in the region are included in the region.

The behaviour of region growing algorithms depends on three factors:

- choice of similarity criteria
- definition of adjacency
- stopping criteria

Which region growing method to choose depends on the segmentation problem to solve.

4.1.1 Similarity criteria

The similarity criterion is used to determine if the pixel under examination should be included in the region or not. The most common criterion to use is similarity in colour or grey level intensities. A simple way is to include pixels if the intensity is inside an intensity interval. This can be expressed:

$$I(X) \in [lower, upper] \tag{4.1}$$

where $I(X)$ is the intensity of pixel X, *lower* and *upper* is the minimum and maximum intensity values.

4.1.2 Definition of adjacency

When expanding the region, the region growing algorithm evaluates the neighbouring pixels of the current region. The most commonly used adjacency definitions for two dimesions, 4-adjacency and 8-adjacency, can be found in Gonzalez and Woods [12].

4.1.3 Stopping criteria

The stopping criteria defines when the region is complete and the entire region has been found. The stopping criteria is usually set to be when there are no longer any neighbours that fulfil the similarity criterion.

4.2 Implementation

There are several variants of region growing algorithm implemented in ITK. We have tested the itk::ConnectedThresholdImageFilter. The itk::ConnectedThresholdImageFilter uses the similarity criterion defined in equation 4.1 and it accepts a pixel if the intensity is within the defined intensity interval.

Figure 4.1 shows the pipeline for the RegionGrowingFilter class, located in Filters/RegionGrowing. The filter interface can be found in figure 4.2. The first step in the pipeline is to pass the image through an edge preserving smoothing filter, represented by the itk::GradientAnisotropicDiffusionImageFilter. The image is then passed through the itk::ConnectedThresholdImageFilter.

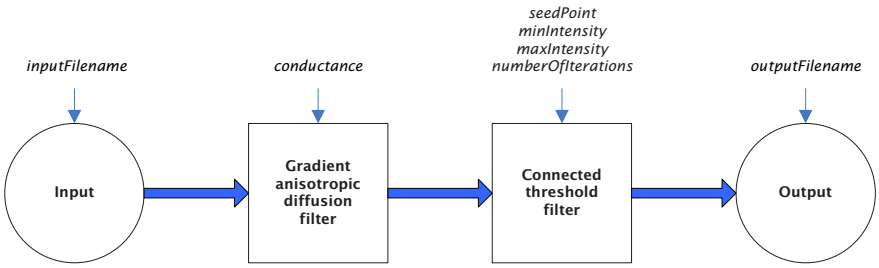


Figure 4.1: Flowchart of the RegionGrowingFilter class

The RegionGrowingFilter class provides functions to set the most important parameters of the filter:

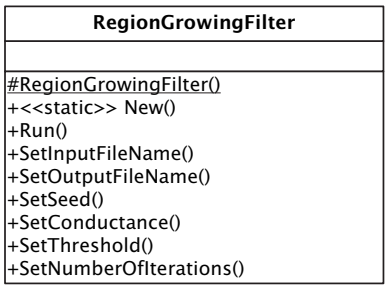


Figure 4.2: The RegionGrowingFilter class.

- Conductance: This parameter varies the strength of the diffusion.
- NumberOfIterations: The number of iterations the filter will run.
- Seed: Sets the seed point.
- MinIntensity, MaxIntensity: The minimum and maximum intensity.
- InputFilename, OutputFilename: These parameters sets the filename to read from and write to, respectively.

4.3 Results

The RegionGrowingFilter was tested on several data sets, without giving very good results. We will therefore present only a small selection of the results. Our purpose is to illustrate the behaviour of the filter.

4.3.1 Results from dataset N359 - MRI

Figure 4.3 shows the results after running the RegionGrowingFilter on the first MRI of the N359 series. The master is MRI and the parameters used are: seed point = (26, 25, 27), conductance = 2.8, lower threshold = 158 and upper threshold = 250. As we can see from Figure 4.3(b) the tumour in this dataset lies inside the ventricle, the tumour therefore has a different grey level than the surrounding tissue. This makes this data set suitable for the connected threshold region growing algorithm.

A volume representation of the result can be found in figure 4.3(a), and two different slices through the result in figure 4.3(c) and figure 4.3(e). As we can see from the result, the tumour is strongly undersegmented. The lower threshold is set to 158, and the tumour consists of many pixels with lower intensities than that.

The results from running the region growing algorithm with lower threshold = 157, can be found in figure 4.4. The master is MRI and the parameters used are: seed point = (26, 25, 27), conductance = 2.8, lower threshold = 157 and upper threshold = 250. As we can see the tumour is still very undersegmented, and in addition it is leaking to other parts of the image. The slice throw the initial picture, figure 4.3(b) shows that the tumour is in connection with tissue of high intensity at the lower right border of the tumour, this makes the algorithm fail.

4.3.2 Results from dataset N359 - US

When we tested the RegionGrowingFilter on ultrasound images, the result was worse than in the MRI case. Figure 4.5 shows the results achieved on the first ultrasound image of the N359 series. The master is MRI and the parameters are: seed point = (27, 28, 24), conductance = 20.0, lower threshold = 134/135 and upper threshold = 250.

The results are extremely sensitive to the value of the parameters. Figure 4.5(a) shows the result after setting the lower threshold to 135. As we can see, the result is heavily oversegmented and is leaking to other parts of the image. If we set lower threshold to 134, the region is not able to grow at all. The reason for the large variation of the results is the large amount of noise in the image.

4.3.3 Results from dataset N351 - MRI

Figure 4.6 shows the results after running region growing algorithm on the first MRI of dataset N351. The master is MRI and the parameters used are: seed point = (36, 26, 27), conductance = 1.0, lower threshold = 186 and upper threshold = 250.

The tumour has a more uniform intensity distribution than in the previous cases, and it is not in contact with bone structures or other tissue with intensity levels similar to the tumour. This makes the segmentation problem easier. The results from this segmentation are acceptable. The tumour is slightly undersegmented, but not nearly as much as in the previous cases. There is also a few locations in the middle of the

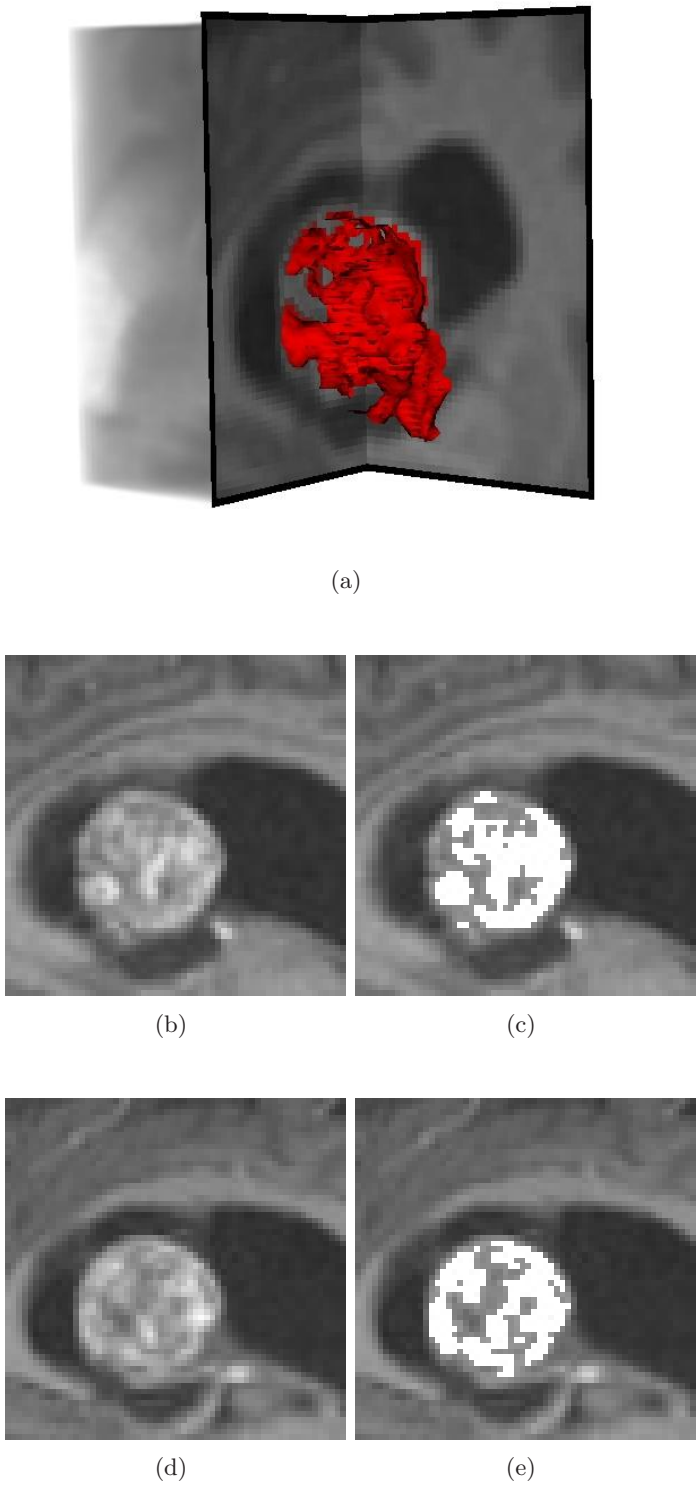
tumour that are not considered as being inside the region, as seen in figure 4.6(e). This is because these areas have a lower intensity value than the rest of the tumour.

4.4 Discussion

The region growing algorithm only takes pixel intensity level in consideration when solving a segmentation problem. This makes the performance of the algorithm very sensitive to noise in the image. MRI and particularly Ultrasound images are very noisy, and the different structures of the brain are not always clearly separated (in grey levels). For instance, a brain tumour can be placed close to a bone structure, making it difficult to find the boarder between the tumour and bone structure solely based on difference in grey level.

The region growing algorithm does not take shape into consideration. The resulting region may therefore have a shape that is rough and with many protruding elements. This shape is not likely the shape of the tumour.

In cases where the tumour has an almost uniform intensity level, and clearly differs from the surrounding tissue, the region growing algorithm gives good results. As we can see from figure 4.6, the results are acceptable, but the region growing algorithm is producing holes in the tumour. This can however easily be solved by not allowing regions to be smaller than some limit.

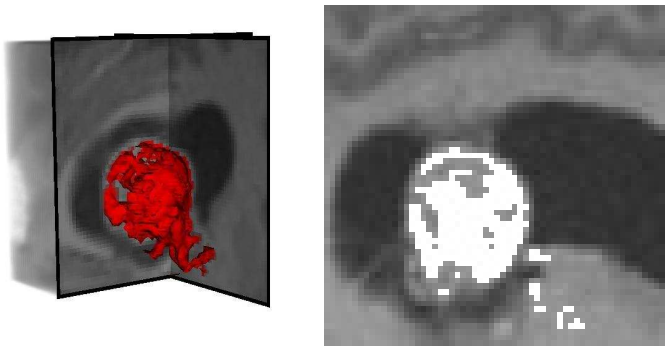


(a)

(b) (c)

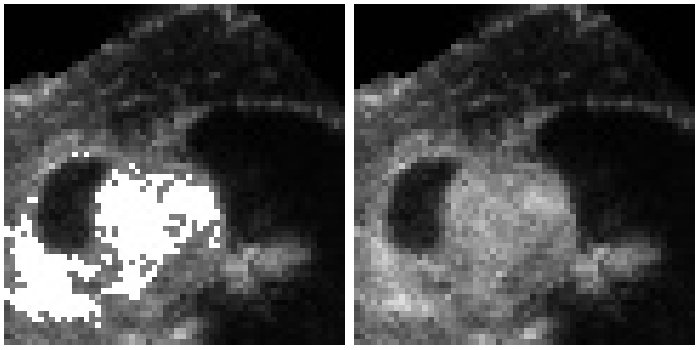
(d) (e)

Figure 4.3: Result of running RegionGrowingFilter on dataset N359 - MRI **(a)** Volume representation. **(b)** Slice of original data set (slice x = 26). **(b)** Slice of result after segmentation (slice x=26). **(b)** Slice of result after segmentation (slice x=24). **(d)** Slice of result after segmentation (slice x = 24).



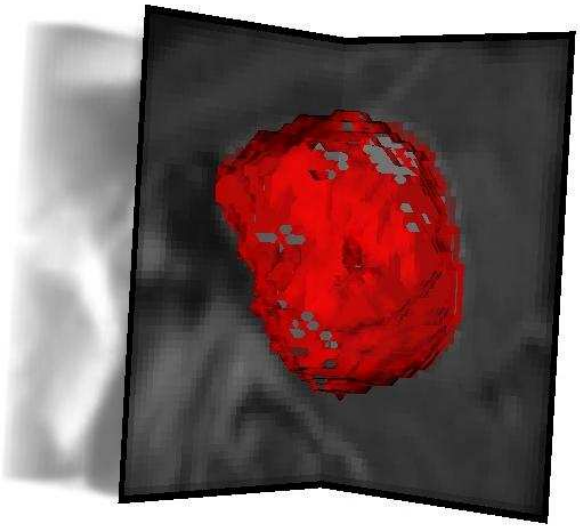
(a) (b)

Figure 4.4: Result of running RegionGrowingFilter on dataset N359 - MRI **(a)** Volume representation. **(b)** Slice of result after segmentation (slice x = 23).

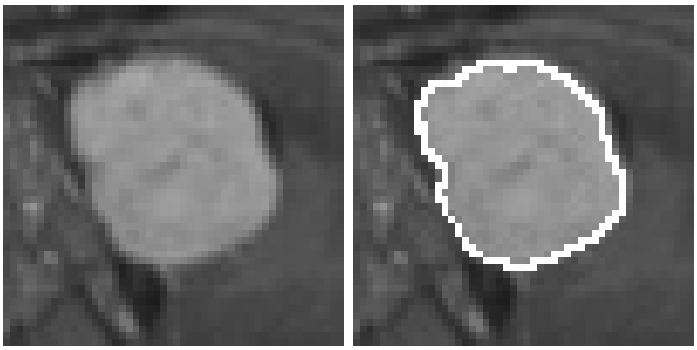


(a) (b)

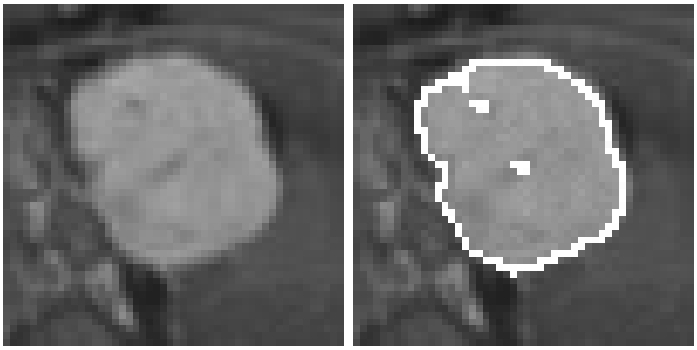
Figure 4.5: Result of running RegionGrowingFilter on dataset N359 - US **(a)** Slice of result after segmentation with lower threshold 134 (slice x = 33). **(b)** Slice of result after segmentation with lower threshold 135 (slice x = 33).



(a)



(b) (c)



(d) (e)

Figure 4.6: Result of running RegionGrowingFilter on dataset N351 - MRI **(a)** Volume representation. (b) Slice of original data set (slice x = 30). **(c)** Slice of result after segmentation (slice x = 30). **(d)** Slice of original data set (slice x = 29). **(e)** Slice of result after segmentation (slice x = 29).

Chapter 5

Snake segmentation

In this chapter we will present the results we have achieved from segmentation of brain tumours using the snake approach. Unlike the other segmentation algorithms we have looked at, ITK does not provide a working implementation of snakes[1]. We have therefore implemented a simplified snake algorithm. The purpose is to get indications of in which extent the snake approach is able to produce satisfactory results on brain tumour segmentation. We have implemented a version of the snake formulation defined by Xu and Price [31], from now on referred to as the Gradient Vector Flow (GVF) snake. We will start by giving an introduction to the theory behind the active contour algorithms, then present our snake implementation and finally look at the results we have achieved.

5.1 Theoretical background

This section will give a brief introduction to the theory behind the snake formulation. We will start by presenting the original snake model defined by Kass et. al. [18], followed by a presentation of the gradient vector flow approach.

5.1.1 The original snake model

The traditional snake model described by Kass et al [18] is a parametric closed contour where each point of the contour can be represented by:

$$v(s) = (x(s), y(s)),$$

where x and y represent coordinates and $s \in [0, 1]$. The snake goes through a deformation process which transforms the snake to a new parametric representation, i.e. a new contour. This transformation process is carried out by minimizing energy functionals using Langrangian mechanics. The snake moves towards a state of local energy minima. The minimization process can be viewed as a time-variable deformation of the curve. The energy of the snake can be defined:

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds \tag{5.1}$$

[1]There exists two snake implementations in itk, but according to the developers of ITK they contain bugs and do not behave properly.

E_{con} represents high-level information of the image. Usually, this energy is involved through the user initialising a contour near the object of interest and this is used as a starting point for the snake and is not usually expressed implicit in the energy function. We will in the following subsections give a description of the internal and image energy used in the snake model suggested by Kass et al. [18].

5.1.2 Internal forces

In the original snake model, the internal energy can be written:

$$E_{int} = \frac{1}{2}\Big(\alpha(s)\left|\frac{\delta v}{\delta s}\right|^2 + \beta(s)\frac{\delta^2 v^2}{\delta s^2}\Big)$$

A circular contour is represented by the terms $x(s) = cos(2\pi s)$ and $y(s) = sin(2\pi s)$. We can see that $x'(s)$ and $y'(s)$ point in the direction of the tangent of the circle and that the direction of $x''(s)$ and $y''(s)$ is towards the centre of the circle. From this we can see that a larger circle has a larger energy and hence when minimizing the energy we are reducing the size of the circle. The first term controls the tension, elasticity and stretching. The second term controls the rigidity and bending. The α and β coefficients are used to weight the influence of each term on the total energy. The internal energy makes the snake contract and assume a circular shape.

5.1.3 Image forces

Kass et al. [18] suggested an image energy formulation that contains three weighted terms:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}$$

The weights w_{line}, w_{edge} and w_{term} can be used to create different behaviours of the snake. They suggest different functions to attract the snake to salient features in the image of interest.

The simplest one is to let the snake be attracted to either dark or bright areas of the image. This can be done by defining:

$$E_{line} = I(x, y)$$

The sign of w_{line} will decide whether the snake moves towards the bright or dark pixels in the image. Another fairly simple approach is to let a large image gradient attract the snake. The snake will then move towards features in the image with the largest variation in pixel-value. This can be expressed:

$$E_{edge} = -|\nabla(x, y)|^2$$

These two approaches are very simple and are indeed very sensitive to noise.

A slightly more sophisticated and a much more widely used approach called scale space takes noise into consideration. This behaviour can be achieved by setting:

$$E_{line} = -(G_\sigma * \nabla^2 I)^2$$

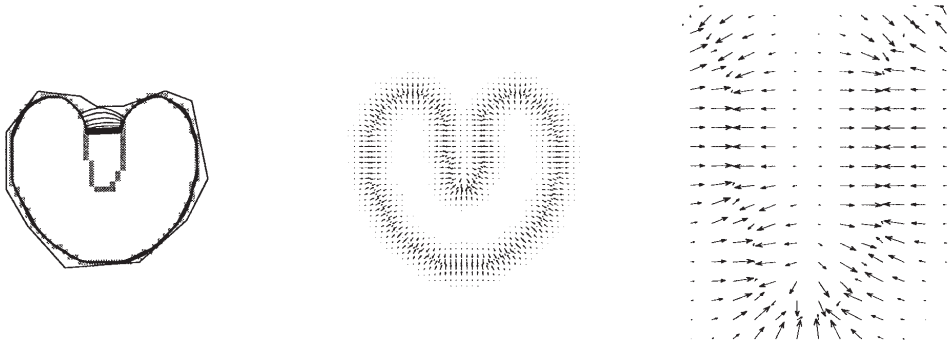
G_σ is a Gaussian with standard deviation σ. By using this external energy it is possible to increase the capture range of the active contour, but this will cause blurring of the

image edges. A result of this will be that the snake will not find the object contour exact.

Kass et al. [18] also describes a termination functional E_{term} that finds terminations of line segments and corners. When setting up an image energy function for the image we can use one or a combination of the functions described above. The choice is dependent on the properties of the image we want to segment.

5.1.4 Gradient Vector Flow

The original snake formulation has some weaknesses. One of the most important problems is sensitivity to the placement of the initial contour. The initial contour must be placed very close to the boundary of the object of interest, if not the snake has a tendency to converge toward the wrong result. The snake also has problems when dealing with objects with concave shapes. Figure 5.1 shows the problem with this approach. The first figure shows how the snake is converging when used on a U-shaped object. The figure in the middle shows the vector field of the shape. As we can see, the vector field has a limited range and consequently, the snake must be initialised close to the object. The rightmost figure shows the vector field close up, and this reveals the problem regarding concave shapes.



(a) (b) (c)

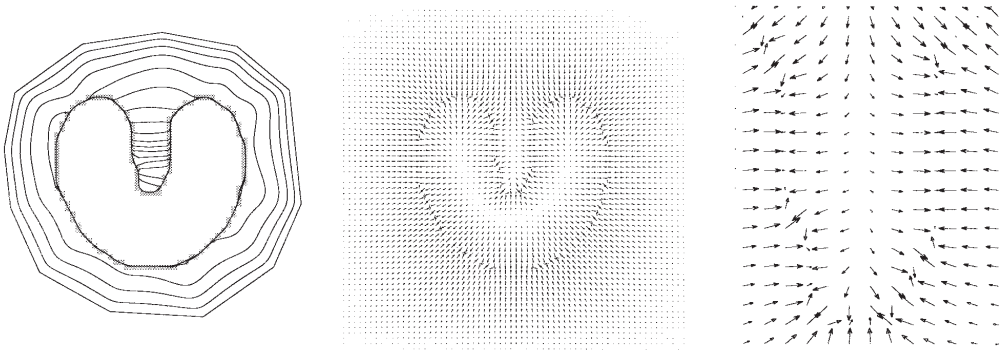
Figure 5.1: **(a)** Convergence of the original snake model. **(b)** Vector field of the snake. **(c)** Close up view of the vector field.[31]

Xu and Price [31] suggested a solution that uses vector diffusion to calculate the potential field, called **gradient vector flow**. They defined an edge map with the property that it is larger near edges in the image. Generally, the gradients of the edge map points towards edges, and are normal to an edge when being close to one. This is a desirable property, because it will make the snake converge into a stable configuration. The problem is that the gradients have large magnitudes only in the immediate neighbourhood of the edges, and in regions with approximately homogeneous intensity the gradient is close to zero. This will make the capture range small and homogenous regions will have no external forces. Xu and Price suggested a solution which objective is to dispose these two last properties of the edge map approach and keep the property of the gradient near the edges. They introduced a gradient vector

flow field and defined it to be the solution to the following equation:

$$\frac{\delta u}{\delta t} = g(|\nabla f|)\nabla^2 u - h(|\nabla f|)(u - \nabla f),$$

where $u(x, 0) = \nabla f(x)$, u is the vector field, f is the edge map derived from the image. The first term is called smoothing term and the second term is called the data term. The image force in equation 5.1 is replaced with the gradient vector flow. As we can see from figure 5.2, the gradient vector flow (GVF) snake deals with both the problems mentioned above. The vector field has a wider capture range and the direction of the vectors in the concavity is pointing downwards, pulling the snake into the concavity.



(a) (b) (c)

Figure 5.2: **(a)** Convergence of the GVF snake model. **(b)** Vector field of the snake. **(c)** Close up view of the vector field.[31]

5.2 Implementation

In this section we will describe how we have implemented the snake model, and also present a more sophisticated approach on how to implement a three dimensional snake described by McInerney and Terzopoulos [21]. We will first present a two dimensional snake implementation, and then try to extend it into three dimensions.

5.2.1 A two dimensional discrete solution

In this section we will present a two dimensional discrete snake model, based on the model described by Lobregt and Viergever [20]. We will explain the most important principles in the solution, and later use this as a basis to extend the solution to three dimensions. A snake model is described through a continuous energy function. We approximate the continuous energy function as a set of discrete nodes in space.

Principal structure of the model

The model represents the deformable contour as a collection of vertices and edges between the vertices, see figure 5.3. The position of a vertex V_i is represented by a vector p_i, and the edge between the vertices V_i and V_{i+1} by a vector d_i. The internal

and external forces of the snake make the vertices move. This is represented in the figure by the acceleration vector a_i. The vertices V_i and V_{i+1} are neighbours, and they therefore have an edge between them. The points form a closed contour, the first vertex of the structure, V_0, and the last vertex, V_{n-1}, is therefore also neighbours. The length of the vector d_i represents the local resolution of the model. If the length is too large, the contour will not be able to follow details of the contour, if it is too small, the computation time will increase.

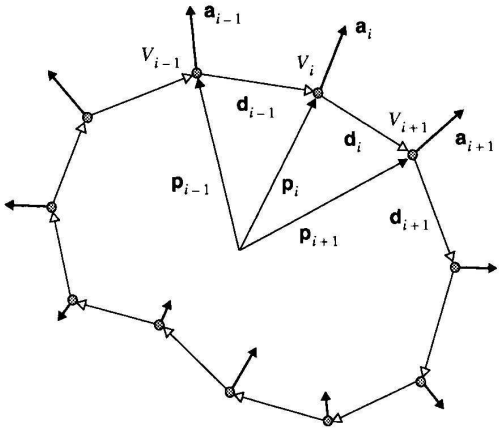


Figure 5.3: A discrete two dimensional contour model with a set of vertices V_i which are connected by edges d_i. Deformation is caused by acceleration forces a_i [20]

Definition of driving forces

The forces governing the deformation of the contour only influence the vertices of the contour. In terms of local curvature at a given position of the contour, Lobregt and Viergever [20] define the internal force to be the difference between the two edge segments that join at that position. This can be expressed:

$$c_i = \hat{d}_i - \hat{d_{i-1}}$$

where the unit vector $\hat{d}_i$ is the direction of the edge vector d_i. The vector c_i represents the second derivative of the contour at the position of vertex V_i. As we can see from figure 5.4, the vector c_i points in direction of smaller local curvature.

In addition to the force corresponding to the second derivative, we also use a force corresponding to the fourth derivative of the contour. We define the fourth derivative at point V_i to be the difference between the second derivatives at the points V_{i+1} and V_{i-1}, see figure 5.5. The fourth derivative controls the bending of the contour. This force is not used in the model described by Lobregt and Viergever [20].

We will describe the computation of the image forces f_{im} when we are describing the implementation in three dimensions.

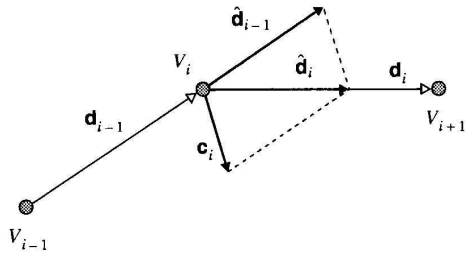


Figure 5.4: Local curvature c_i at the position of a vertex V_i[20].

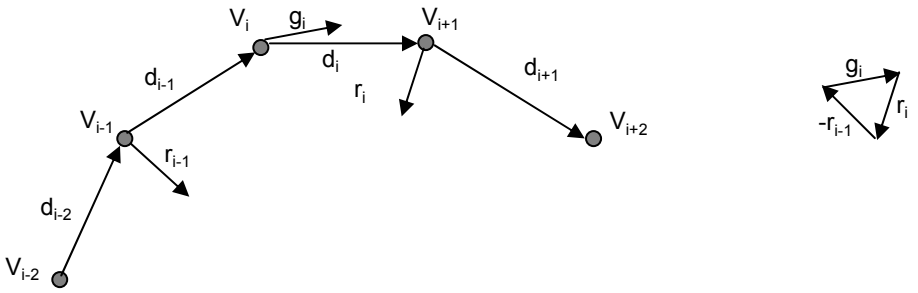


Figure 5.5: Fourth derivative, g_i, at the position of a vertex V_i.

The total force f_i acting on vertex V_i can be expressed:

$$f_i = \alpha c_i + \beta g_i + \lambda f_{im,i}$$

9where α, β and λ are user-defined weighting factors. c_i and g_i are the internal forces, $f_{im,i}$ the image forces acting on vertex V_i.

Deformation and resampling

The deformation process is implemented by making the energy term time-dependent, the deformation process now can be described by the expression:

$$p_i(t + \Delta t) = p_i(t) + f(t + \Delta t)$$

where t represents a point in time, and $t + \Delta t$ a time Δt later.

When the vertices in the contour are moved around, the distance between the points and consequently, the local resolution, are changing. It is desirable to have a fairly equal local resolution of the snake at all points of the evolution. To control that the distances between the points in the contour do not vary too much, we require that

the distance between two points must be inside a predefined interval. The following relationship between the desired distance, l_{des}, the minimum distance, l_{min}, and the maximum distance, l_{max}, has proven to give satisfactory results:

$$l_{min} = \frac{1}{2} l_{des}$$

$$l_{max} = \frac{3}{2} l_{des}$$

In our implementation we defined l_{des} to be 4 pixels.

If the distance between two vertices V_i and V_{i+1} is smaller than l_{min}, the two vertices are merged to a new vertex with position halfway between the to old vertices, as shown in figure 5.6(a). If the distance is larger than l_{max}, a new vertex is inserted between V_i and V_{i+1} as shown in figure 5.6(b).

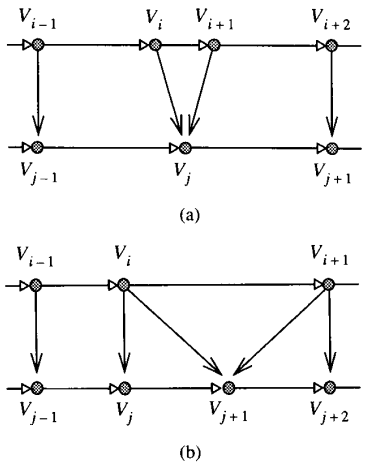


Figure 5.6: **(a)** Removal of a point in the contour. **(b)**Insertion of a point in the contour.

5.2.2 Implementing a three dimensional snake

Extending the two dimensional model into three dimensions may at first glance seem like an easy task, but this leads to some difficulties. In this section we will explain some of the difficulties an extra dimension involves and present our solution to the problem. We will also present an alternative approach, developed by McInerney and Terzopoulos [21].

Difficulties

Defining a three dimensional spherical structure does not lead to any difficulties, neither does defining the driving forces nor the neighbour relationships between the vertices. The neighbour relationships are important in order to be able to compute second and fourth derivatives of the sphere and to move the vertices towards local energy minima in the image. The second and fourth derivatives are now computed in two directions of

the image: the direction of the x-y- plane and one direction perpendicular to the x-y-plane.

The difficulties with a three dimensional spherical structure appear after the vertices have been moved around and the distance between the neighbouring vertices is either too small or too large. There is not always an intuitive way to define the neighbour relationships after a new vertex is inserted or a vertex is removed.

The most intuitive way to define a three dimensional initial contour is to define a spherical contour described by the following equation:

$$x(u_i, v_i) = \begin{pmatrix} \cos u_i \cos v_i \\ \cos u_i \sin v_i \\ \sin u_i \end{pmatrix} \tag{5.2}$$

In the initial contour, the vertex $x(u_i, v_i)$ has the following neighbours: $x(u_{i+1}, v_i)$, $x(u_{i-1}, v_i)$, $x(u_i, v_{i+1})$ and $x(u_i, v_{i-1})$ as shown in figure 5.7.

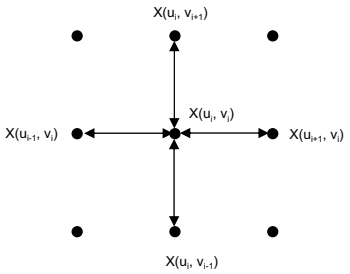
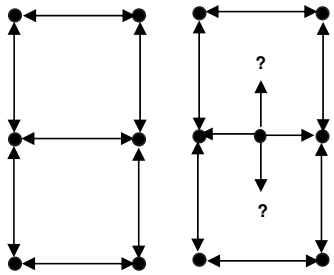


Figure 5.7: Definition of the point $x(u_i, v_i)$ neighbours.

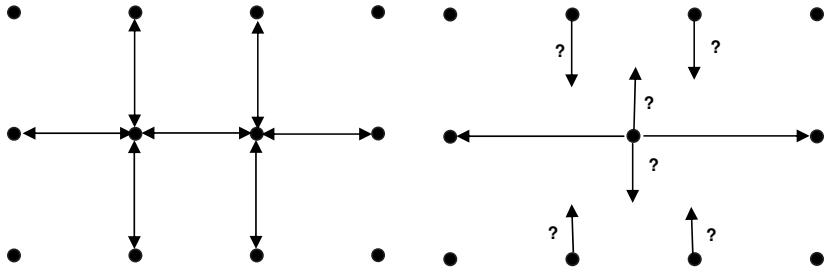
After the vertices have been influenced by the forces of the snake, they may be too close or too far away from each other. To make the snake model behave properly, it is necessary to insert and remove vertices from the sphere. Figure 5.8 shows the situation before and after inserting a vertex into a sphere. The problem in figure 5.8 is to define the neighbour relations marked with a question mark. One choice is to leave them open. This can however result in a lot of open spaces in the contour, and because of lack of neighbour relations no vertices will be inserted in these holes, resulting in an incomplete result.

Removing a vertex from the structure also results in a couple of difficulties. No intuitive way exists on how to define the new neighbours for the vertices previously being neighbours with the removed vertex. In figure 5.9, the new vertex inserted between the two removed vertices will have the same left neighbour as the vertex removed at left side of the new vertex, and same for the right vertex. But the two remaining neighbours are more difficult to decide. One choice is to let the neighbour references be empty, but this will not be a good solution because we can risk a lot of vertices being removed, leaving open spaces in the sphere. Another choice may be to choose the closest one of the two candidates as the new neighbour. This will neither be a good solution because when the contour evolves, the structure of the sphere can become very complex and difficult to handle. It is also extremely difficult to control that it is behaving correctly.



(a) (b)

Figure 5.8: Inserting a new vertex into a sphere. **(a)** Situation before inserting the vertex. **(b)** Situation after inserting the vertex.



(a) (b)

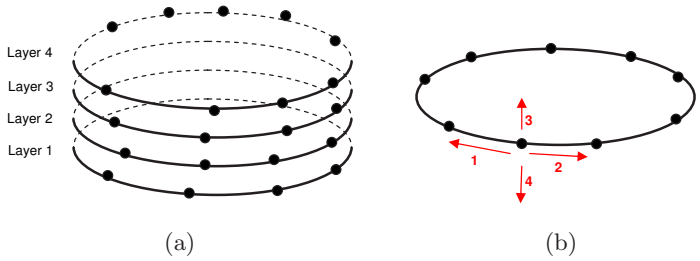
Figure 5.9: Removing a vertex from a sphere. **(a)** Situation before removing the vertex. **(b)** Situation after removing the vertex.

As we have seen, it is difficult to extend the two dimensional discrete snake model into three dimensions without making some modifications to it. The collection of vertices and edges constituting the contour have a tendency to get too complex and difficult to handle elsewhere.

A three dimensional solution

In our implementation of the snake model we made some simplifications of the model described in the previous section. Figure 5.10(a) shows the overall structure of the model. Our model is constructed from layers, with each layer containing a set of vertices. All the vertices have four neighbours, as shown in figure 5.10(b). Two of the neighbours are within the same layer (marked 1 and 2 in the figure), and are managed the same way as in the two dimensional model. This relationship is symmetric, the vertices are neighbours of each other. If the distance between two vertices in the same layer is too large, a new vertex is inserted, and if it is too small a vertex is removed. The last two neighbours (marked 3 and 4) are not symmetric relations. For every point in one layer, we find the closest point in the two neighbouring layers. Two points can be neighbours with the same point, and if a point is neighbour with another point, the other point is

not necessarily neighbour with the first point. The neighbour relations are recomputed for each iteration of the snake. This property of the sphere makes it possible to remove points from a layer without affecting the neighbour relationships of points in other layers.



(a) (b)

Figure 5.10: Structure of our 3D implementation. **(a)** A layered model. **(b)** Neighbour relationships. Relations marked 1 and 2 are relations to vertices in same layer, relations marked 3 and 4 are relations to points in neighbouring layers.

All vertices belonging to the same layer have the same z coordinate, i.e. all vertices can only move in x- and y-directions. The whole layer is moved depending on the resultant of the z-component of all forces acting on the vertices in that layer. The purpose of this restriction is to avoid problems when vertices are inserted between layers. Instead of inserting vertices between the layers or removing a single vertex from a layer if it is too close to a vertex in another layer, we insert and remove the entire layers.

Our implementation of the snake model is now manageable, and is not too different from the two dimensional solution. Our solution is more sophisticated than computing a two dimensional solution in a slice-by-slice manner. The computation of the forces influencing the movement of the vertices within a layer, takes under consideration the neighbour vertices in the neighbouring layers. This prevents the vertices from moving too far from their neighbours in the neighbouring layers.

Possible side effects

Our snake implementation contains some simplifications of a correct three dimensional implementation. This may affect the final result.

One of the side effects is that because we require that all vertices in a layer must have equal z-coordinates, the vertices move in the z-direction that is best for the entire layer, not what is necessarily best for that particular vertex. A movement of the entire layer may result in higher snake energy than what may be the case if all vertices were allowed to move freely.

To simplify the definition of the initial contour, we have the same radius on every layer. This results in a cylindrical shape of the initial contour. Most tumours have a spherical shape, and the snake may terminate in the wrong local minima, especially at the top and bottom of the contour.

Simple improvements to speed up the snake evolution

We have not focused on making a fast snake implementation, we will therefore not take computation time into consideration when evaluating the results of the snake. There are

some simple improvements that can make the computation time decrease considerably.

One improvement is to let all vertices remember their neighbour vertices in the neighbouring layers. If the vertex is not removed from the structure, the search for the closest neighbour starts at that vertex. This way it is not necessary to search through and compute the distance to all the vertices in the neighbour layers within the same iteration.

Another quite simple improvement of our implementation is to keep the distances and derivatives between all vertices in an array, instead of recomputing them several times.

5.2.3 Triangular Surface Model

McInerney and Terzopoulos [21] present a method for discretization of a three dimensional snake, where the initial points are defined in a sphere (see equation 5.2).

The sphere is subdivided into triangular elements. To increase the resolution, one triangle is subdivided into four new triangles. The model described by McInerney and Terzopoulos subdivides all triangles in the model to change the resolution. A better scheme is to locally subdivide the triangles only at places where the data varies considerably.

5.2.4 ITK pipeline

The pipeline for the snake model we have implemented and tested is shown in figure 5.11.

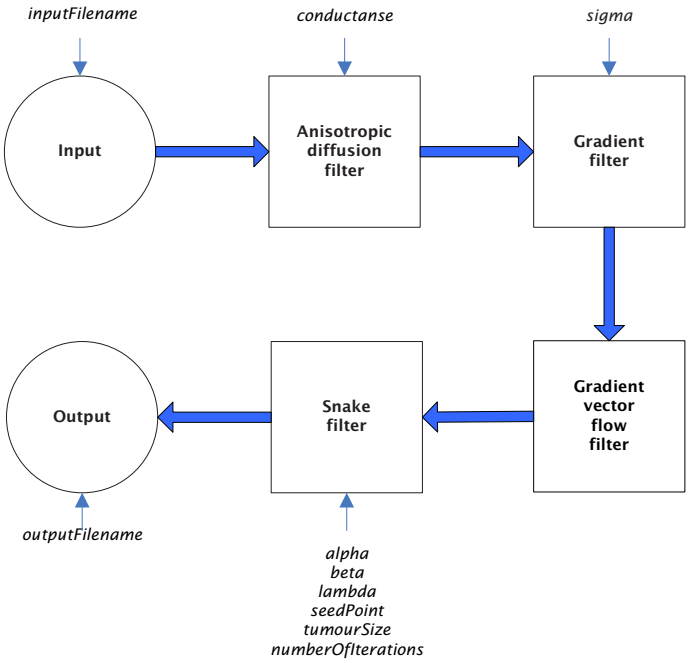


Figure 5.11: Flowchart of the GvfSnakeFilter class

The filter starts with an itk::GradientAnisotropicDiffusionImageFilter. This filter

smooths the image while preserving edge information. The smoothed image is then passed through an itk::GradientRecursiveGaussianImageFilter. This filter computes the gradient vectors of the smoothed image. The gradient image is then rescaled to make vectors at positions close to edges have small magnitudes, and vectors far away from edges have larger magnitudes. The rescaled gradient image is then passed through an itk::GradientVectorFlowFilter that enlarges the capture range of the gradient force. The GVF gradient image is then used as image forces for the snake implementation described in the previous section. An itk::PointSet is created with centre of the middle layer at the seedPoint, and radius equal to the tumourSize. The forces influencing the points of the pointset are then computed and the points are moved around as described in the previous section. The last step of the GVFSnakeFilter is to transform the pointset into an image using the itk::PointSetToImageFilter.

The interface for the GvfSnakeFilter can be found in figure 5.12. To use the filter, the following parameters must be given:

- Seed: Sets the initial seed point
- TumourSize: Sets the approximate radius of the tumour. This is used for creation of the initial contour.
- Conductance: This parameter varies the strength of the diffusion.
- Sigma (σ): Controls how sensitive the gradient filter should be.
- Alpha (α): Controls the snake tension, elasticity and stretching.
- Beta (β): Controls the rigidity and bending of the snake.
- Lambda (λ): Weights how much the image forces should influence the movement of the points.
- SnakeIterations: Maximum number of iterations of the snake.
- InputFilename and OutputFilename: These parameters set the filenames to read from and write to.

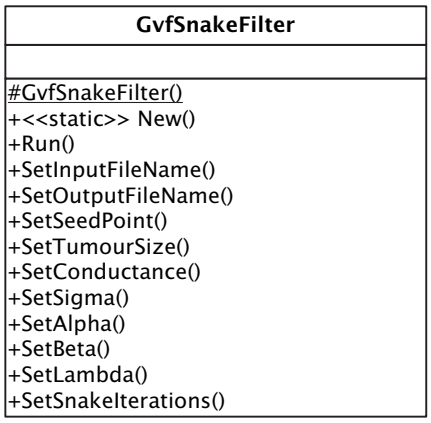


Figure 5.12: The GvfSnakeFilter class.

5.3 Results

In this section we will presents results from tests with the snake segmentation filter.

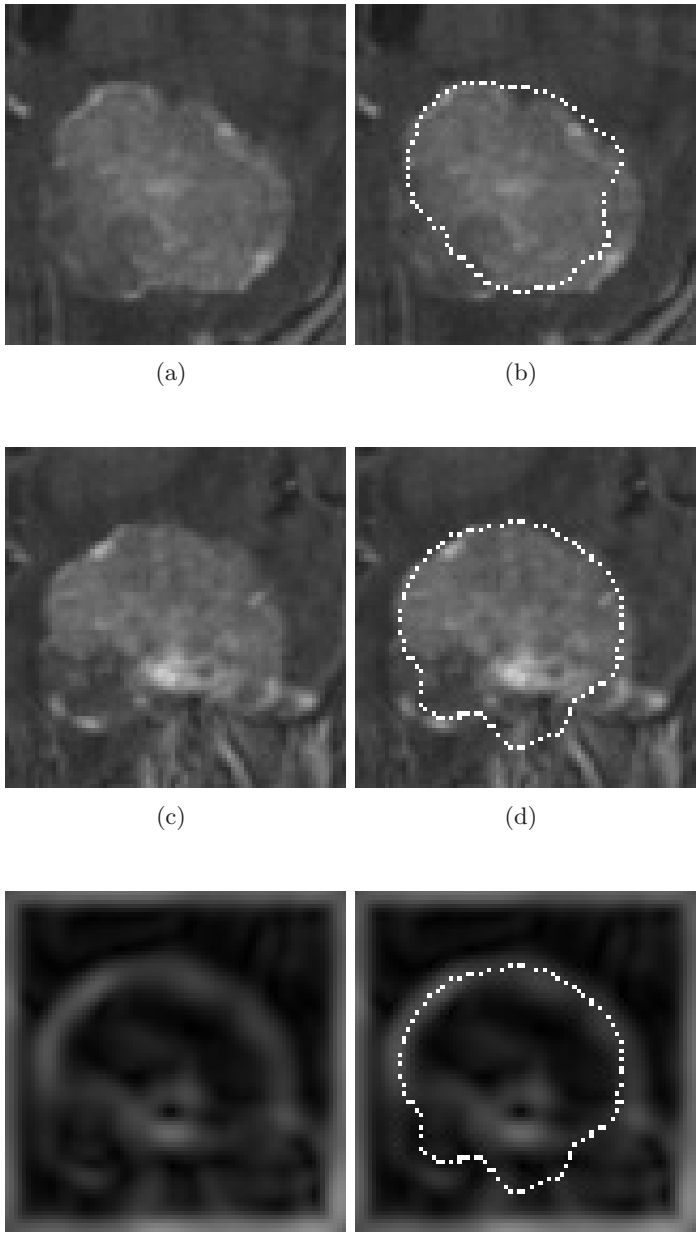
5.3.1 Dataset N241 - MRI

Figure 5.13 shows the result after running the GvfSnakeFilter on the first MR image of the N241 series. Master is US and the parameters are: seedpoint = (38, 39, 45), tumourSize = (29, 29, 15), conductance = 5.0, $\sigma = 3.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 2.0$.

As we can see from the slices 5.13(a) and (c), the tumour tissue does not have uniform graylevel intensities. The border at the lower left part of the tumour is difficult to separate from the surrounding tissue. As we can see from figure 5.13(b), the snake has failed to detect the lower left part of the tumour. It has also undersegmented the tumour at the lower right side. The degree of oversegmentation is low in this slice. The snake only has problems with a concavity on the top of the tumour. The edges are probably not sharp enough to pull the snake into the concavity.

Figure 5.13(d) shows a slice through the snake result at position z=36. We can see from figure 5.13(c) that the tumour lies close to some surrounding tissue with sharp edges at the lower left side. This results in a rather high degree of oversegmentation of the result. As we can see from figure 5.13(e) and (f), the edges of the surrounding tissue have almost the same intensity as the edges of the tumour, this is causing the snake to find the wrong boundary.

The total judgement of this result is that it is a middle good segmentation. At the parts of the tumour where the border separates clearly from the background the results are very good, but at parts where the transition between the tumour and surrounding tissue is less evident the results are of lower quality. The tumour is too much under- and oversegmented for the result to be valuable for tasks that require accurate segmentations. The slices where the tumour is not particularly oversegmented can be valuable as indication of tumour size, position etc. The degree of misclassification may be reduced if a better method for initialisation of the active contour is available.



(a) (b)

(c) (d)

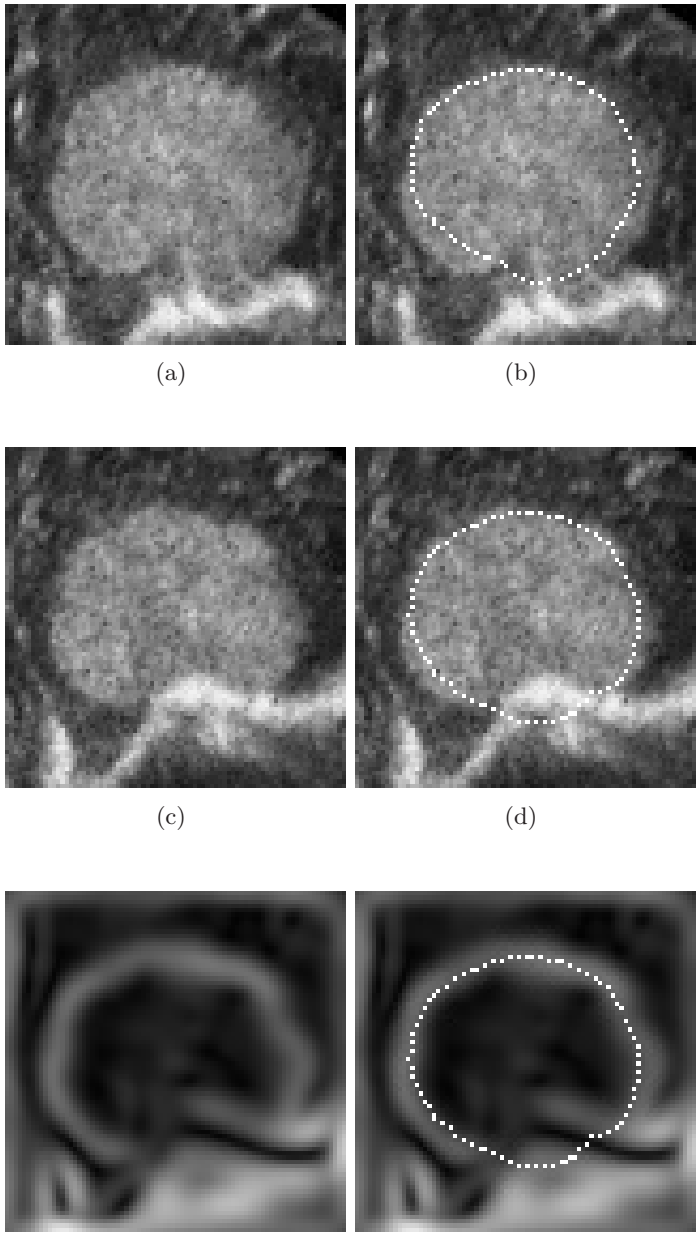
(e) (f)

Figure 5.13: Results of running GvfSnakeFilter on dataset N241 - MRI **(a)** Slice of original image data (slice z=49). **(b)** Slice of result on original data (slice z=49). **(c)** Slice of original image data (slice z=36). **(d)** Slice of result on original data (slice z=36). **(e)** Slice of gradient image (slice z=36). **(f)** Slice of result on gradient (slice z=36).

5.3.2 Dataset N241 - US

Figure 5.14 shows the result after running the GvfSnakeFilter on the first US image of the N241 series. Master is US and the parameters are: seedpoint = (41, 42, 45), tumourSize = (31, 25, 15), conductance = 6.0, $\sigma = 3.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 3.0$.

The results achieved for the US image is slightly better than for the for the MR image from the same series. The degree of undersegmentation is lower than for the MRI case. The algorithm still has problems regarding the the neighbouring tissue at the lower part of the tumour. The reason for the oversegmentation can be found in figure 5.14(e). The edges of the surrounding tissue is stronger than the edges of the tumour, dragging the contour to this position. The degree of undersegmentation is lower for the US image than what it was for the MRI case. One of the reasons for this is that the MR image is capable of capturing the variations of graylevel inside the tumour.



(a) (b)

(c) (d)

(e) (f)

Figure 5.14: Results of running GvfSnakeFilter on dataset N241 - US **(a)** Slice of original image data (slice z=44). **(b)** Slice of result on original data (slice z=44). **(c)** Slice of original image data (slice z=39). **(d)** Slice of result on original data (slice z=39). **(e)** Slice of gradient image (slice z=39). **(f)** Slice of result on gradient (slice z=39).

5.3.3 Dataset N351 - MRI

Figure 5.15 shows the result after segmentation of the first MR image of dataset N351 with the GvfSnakeFilter. The master is MRI and the parameters used are: seedpoint = (26, 26, 23), tumourSize = (16, 16, 14), conductance = 4.0, $\sigma = 3.0$, $\alpha = 0.05$, $\beta = 0.03$, $\lambda = 1.0$.

As we can see from figure 5.15(b), the tumour is slightly undersegmented (at the lower right part of the tumour). If we look at the image data in figure 5.15(a), there is a weak edge in the image data, causing the snake to fall into the wrong local minima. This problem can be solved by using a larger conductance and σ in the preprocessing phase, but this will also blur the edges of the tumour, making it difficult to find a precise position of the tumour edge.

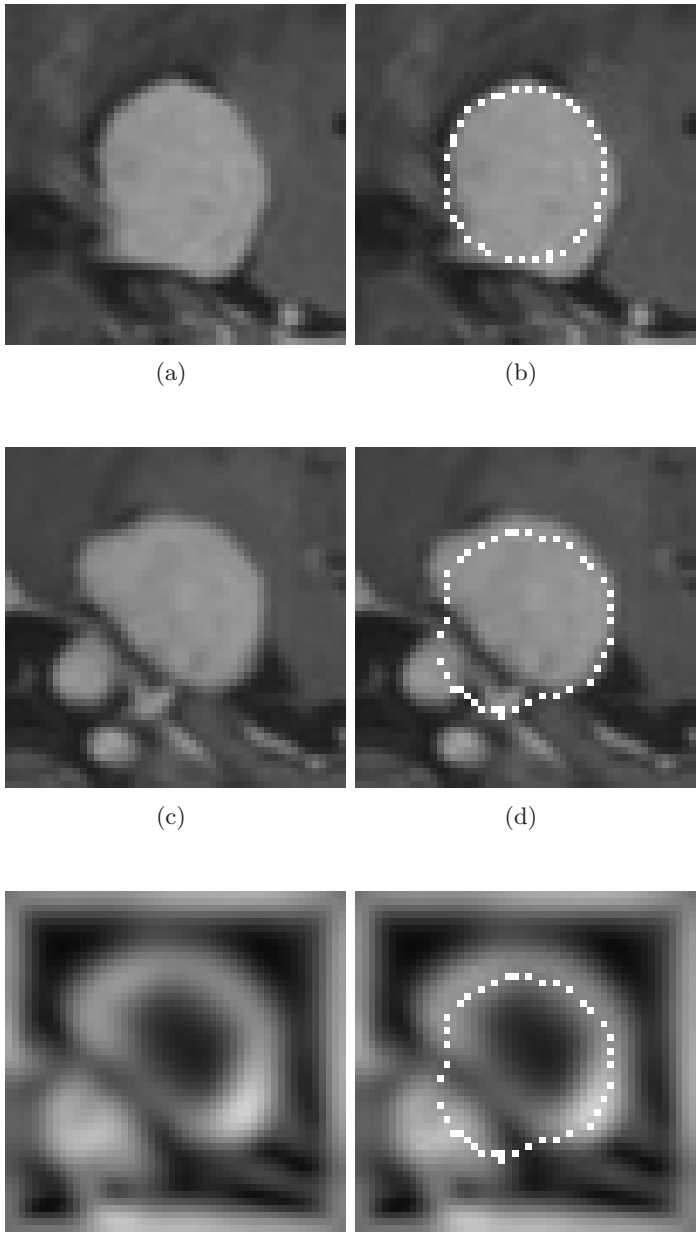
The tumour lies close to some surrounding tissue as shown in figure 5.15(c). The surrounding tissue has about the same graylevel as the tumour and the same contrast to other surrounding tissue as the tumour. These edged in the image pull the snake away from the edge of the tumour, and as we can see from the gradient image of figure 5.15(e), the edges of the surrounding tissue are stronger than the edges of the tumour. This leads to a oversegmentation of the tumour in this area.

The segmentation result is altogether slightly undersegmented, except from the positions where the tumour has neighbouring tissue with similar greylevel. The result is not good enough to be used in applications demanding high presission, because of the high degree of oversegmentation. This may be solved with a more sophisticated initialising method. The results give a good indication of the tumour boundary.

5.3.4 Dataset N351 - US

Figure 5.16 shows the results of running the GvfSnakeFilter on the first US image of the N351 series. Master is US and the parameters are: seedpoint = (69, 65, 56), tumourSize = (30, 31, 14), conductance = 25.0, $\sigma = 7.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 50.0$.

The US image is extremely noisy, making it very difficult to find the tumour on the image. Consequently the results are not very good. The result shown in figure 5.16(b) is achieved after performing heavy smoothing of the original US image. The GVF edge map has small magnitude because the edges of the image are rather weak, λ is therefore set to 50.0 to increase the magnitude of the gradient vector field to be about the same as the internal energies.

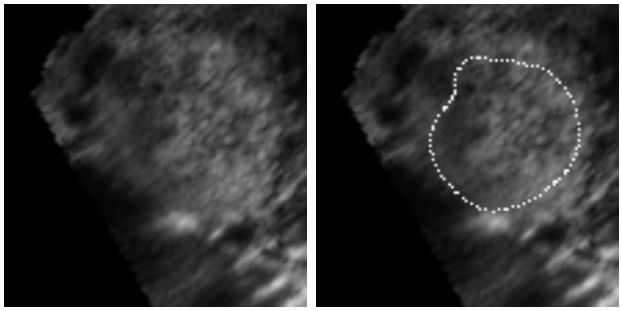


(a) (b)

(c) (d)

(e) (f)

Figure 5.15: Results of running GvfSnakeFilter on dataset N351 - MRI **(a)** Slice of original image data (slice z=34). **(b)** Slice of result on original data (slice z=34). **(c)** Slice of original image data (slice z=19). **(d)** Slice of result on original data (slice z=19). **(e)** Slice of gradient image (slice z=19). **(f)** Slice of result on gradient (slice z=19).



(a) (b)

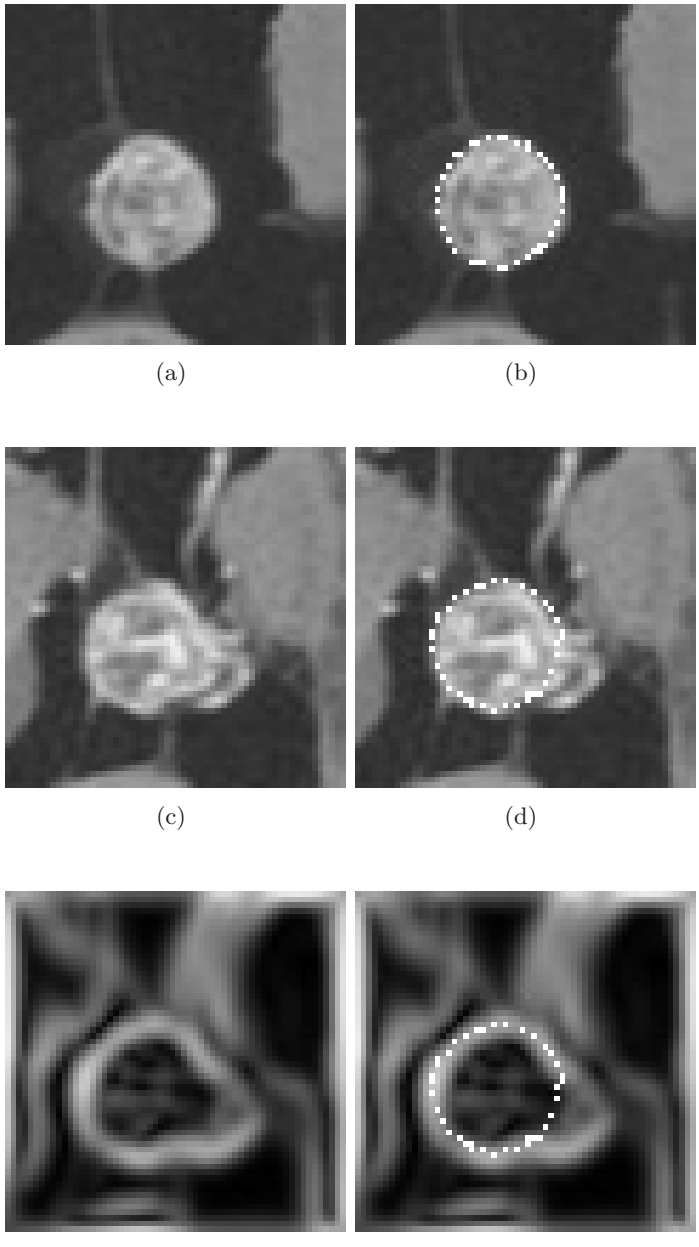
Figure 5.16: Result of running GvfSnakeFilter on dataset N351 - US **(a)** Slice of original image data (slice z=62). **(b)** Slice of result on original data (slice z=62).

5.3.5 Dataset N359 - MRI

Figure 5.17 shows the results of running the GvfSnakeFilter on the first MR image of the N359 series. The master is MRI and the parameters used are: seedpoint = (26, 26, 25), tumourSize = (13, 13, 12), conductance = 2.0, $\sigma = 2.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 2.0$.

Figure 5.17(b) shows that the snake has found the boarder of the tumour with high precision. Because the tumour tissue has high variance of grey level intensities, it is necessary to smooth the image to eliminate disturbing edge information. Figure 5.17(e) shows that there is a great deal of remaining edge information inside, and around the tumour. The result is that the snake has problems finding the lower left boundary of the snake. If we smooth the image more, we can eliminate this problem, but that will result in a less accurate result. The problem can also be solved by using a more sophisticated initialisation of the contour.

The result is very good, except from the place where the tumour is growing into the surrounding tissue as shown in figure 5.17(d). The result is useful as an indication of the size and scope of the tumour, but is not good enough in areas of application demanding extremely precise results.



(a) (b)

(c) (d)

(e) (f)

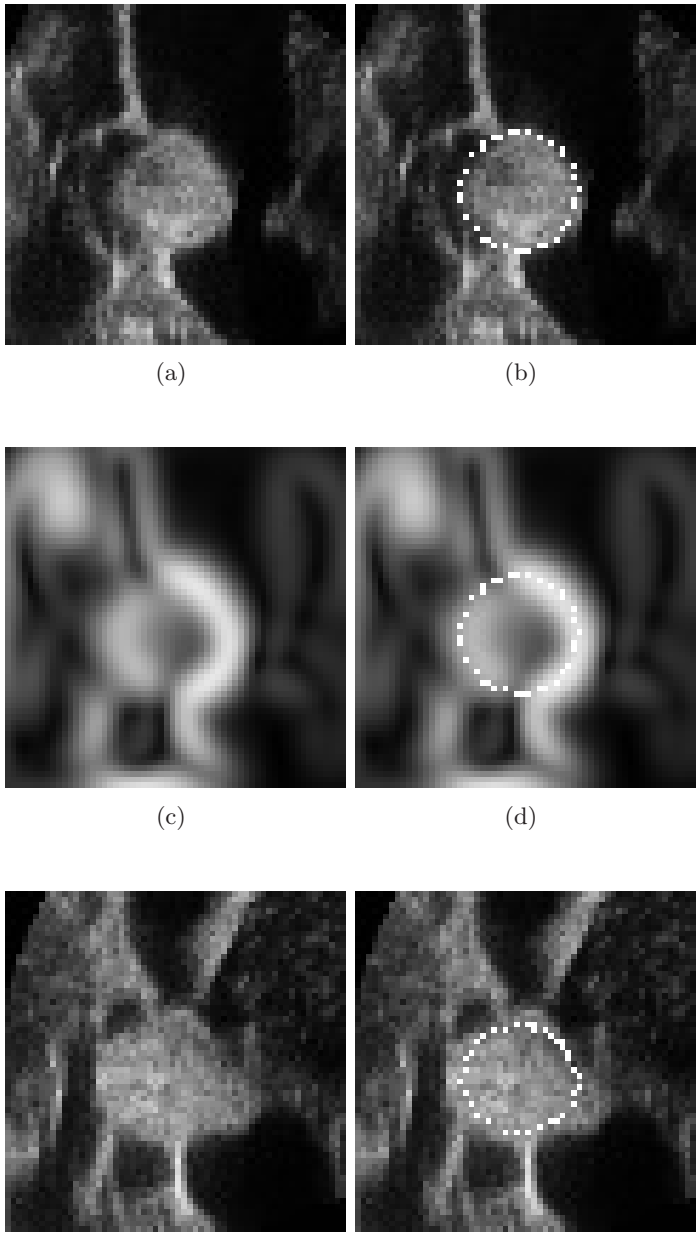
Figure 5.17: Results of running GvfSnakeFilter on dataset N359 - MRI **(a)** Slice of original image data (slice z=29). **(b)** Slice of result on original data (slice z=29). **(c)** Slice of original image data (slice z=19). **(d)** Slice of result on original data (slice z=19). **(e)** Slice of gradient image (slice z=19). **(f)** Slice of result on gradient (slice z=19).

5.3.6 Dataset N359 - US

The results of the segmentation of the US images are shown in figure 5.18. The master is MRI and the parameters used are: seedpoint = (30, 28, 14), tumourSize = (12, 12, 12), conductance = 7.0, $\sigma = 3.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 6.0$.

The results are not as good as the results of the MR image in the same dataset, and the results now have a higher degree of undersegmentation than what was the case for the MRI segmentation. The snake finds the tumour boundary relatively precise when the tumour is not surrounded by tissue with similar grey level as shown in figure 5.18(d). The results are not as good when the tumour merge into the surrounding tissue as shown in figure 5.18(b).

The result can be valuable for a surgeon when high precision is not necessary.



(a) (b)

(c) (d)

(e) (f)

Figure 5.18: Results of running GvfSnakeFilter on dataset N359 - US **(a)** Slice of original image data (slice z=29). **(b)** Slice of result on original data (slice z=29). **(c)** Slice of gradient image (slice z=29). **(d)** Slice of result on gradient (slice z=29). **(e)** Slice of original image data (slice z=19). **(f)** Slice of result on original data (slice z=19)

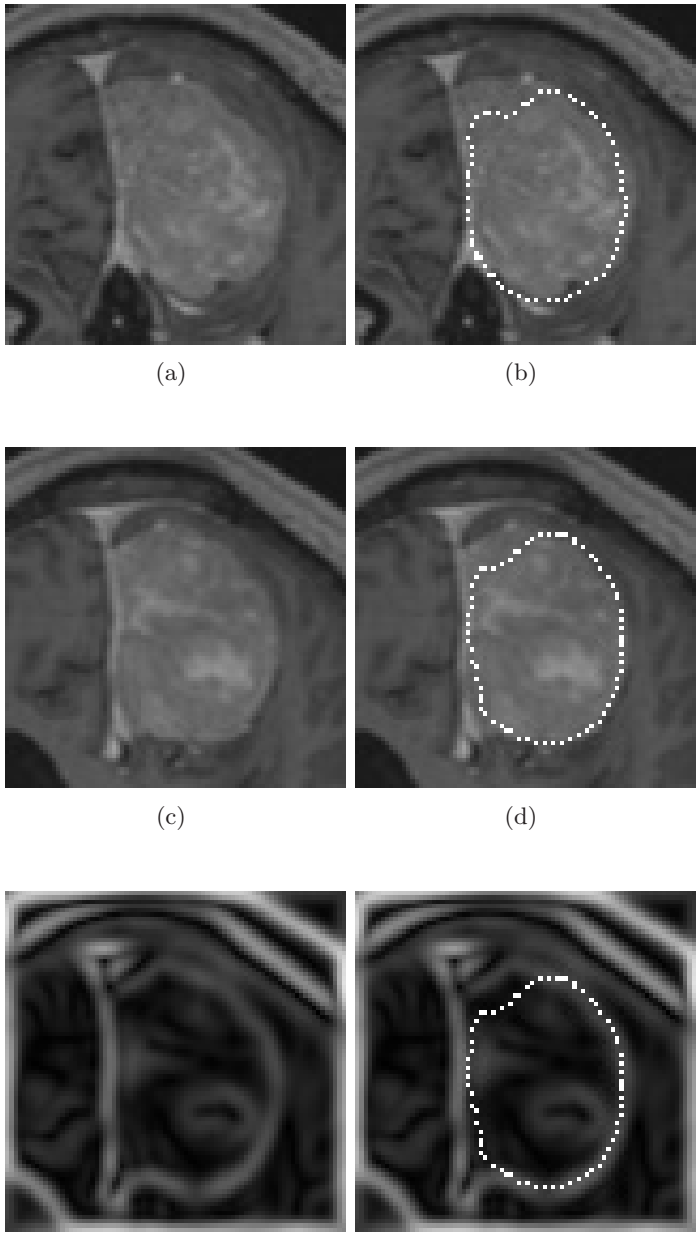
5.3.7 Dataset N360 - MRI

Figure 5.19 shows the snake contour after passing the first MRI of the dataset N360 through the GvfSnakeFilter. The master is MRI and the parameters used are: seedpoint = (46, 36, 50), tumourSize = (20, 27, 30), conductance = 6.0, $\sigma = 2.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 1.0$.

As we can see from figure 5.19 the GvfSnakeFilter gives very good results on this dataset. In figure 5.19(d) the contour is matching the contour of the tumour very good, the only exception is in the top left part of the tumour where the tumour is undersegmented. As we can see from the gradient image in figure 5.19(f), this problem could be avoided if the snake was initialised closer to the real boundary. The snake also succeeds in dividing the tumour from the nearby tissue at the left side of the tumour.

Figure 5.19(b) shows a slice where the results are not quite that good. The snake fails in detecting a concavity at the bottom of the tumour, and the neighbouring tissue at the left side of the tumour is not distinguished from the tumour.

The results of this segmentation are very good except from the problems at the top left of the tumour. If the snake implementation provides a method for better initialisation of the snake, we may be able to eliminate this problem.



(a) (b)

(c) (d)

(e) (f)

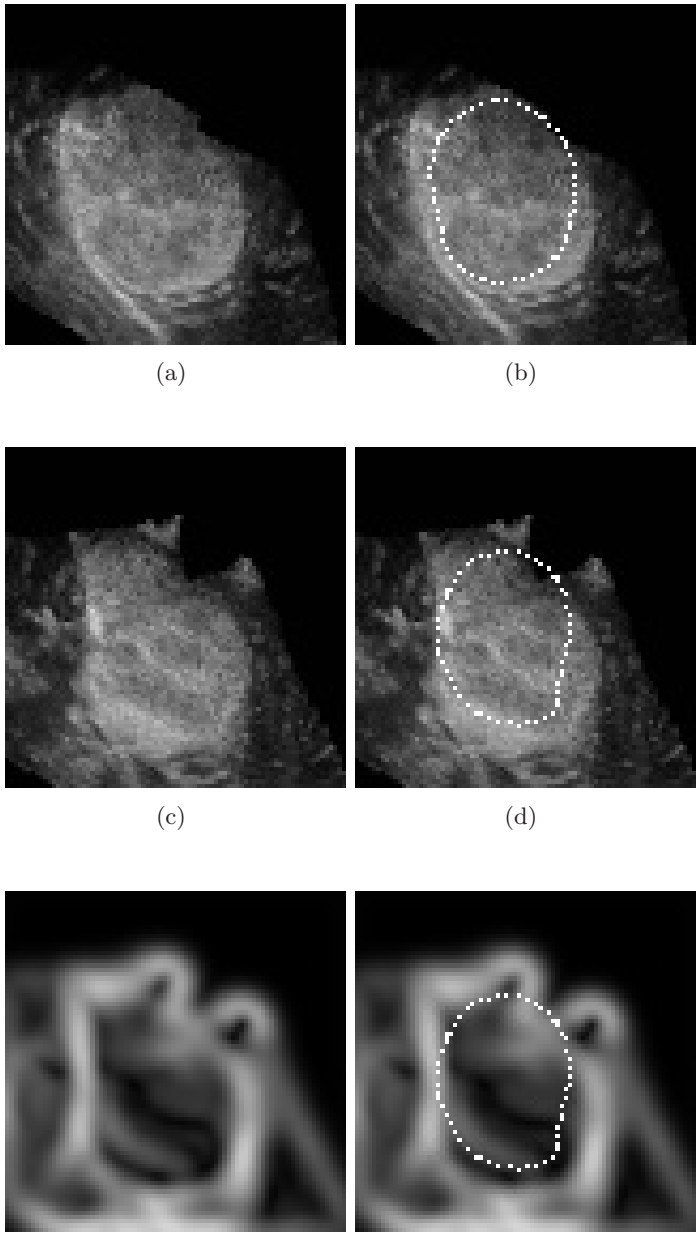
Figure 5.19: Results of running GvfSnakeFilter on dataset N360 - MRI **(a)** Slice of original image data (slice z=43). **(b)** Slice of result on original data (slice z=43). **(c)** Slice of original image data (slice z=54). **(d)** Slice of result on original data (slice z=54). **(e)** Slice of gradient image (slice z=54). **(f)** Slice of result on gradient (slice z=54).

5.3.8 Dataset N360 - US

Figure 5.20 shows the result after passing the first US image of the dataset N360 through the GvfSnakeFilter. The master is MRI and the parameters used are: seedpoint = (36, 38, 20), tumourSize = (20, 25, 30), conductance = 20.0, $\sigma = 3.0$, $\alpha = 0.04$, $\beta = 0.04$, $\lambda = 1000.0$.

The image is very noisy, and it is necessary to smooth the image heavily. This makes the edge information weaker and makes it more difficult to find the precise boundary. Figure 5.20(e) shows a slice through the gradient image, and as we can see the edge information is smoothed.

The results are very inaccurate, and can only be used as an indicator to the tumour boundary.



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

Figure 5.20: Results of running GvfSnakeFilter on dataset N360 - US **(a)** Slice of original image data (slice z=25). **(b)** Slice of result on original data (slice z=25). **(c)** Slice of original image data (slice z=38). **(d)** Slice of result on original data (slice z=38). **(e)** Slice of gradient image (slice z=38). **(f)** Slice of result on gradient (slice z=38).

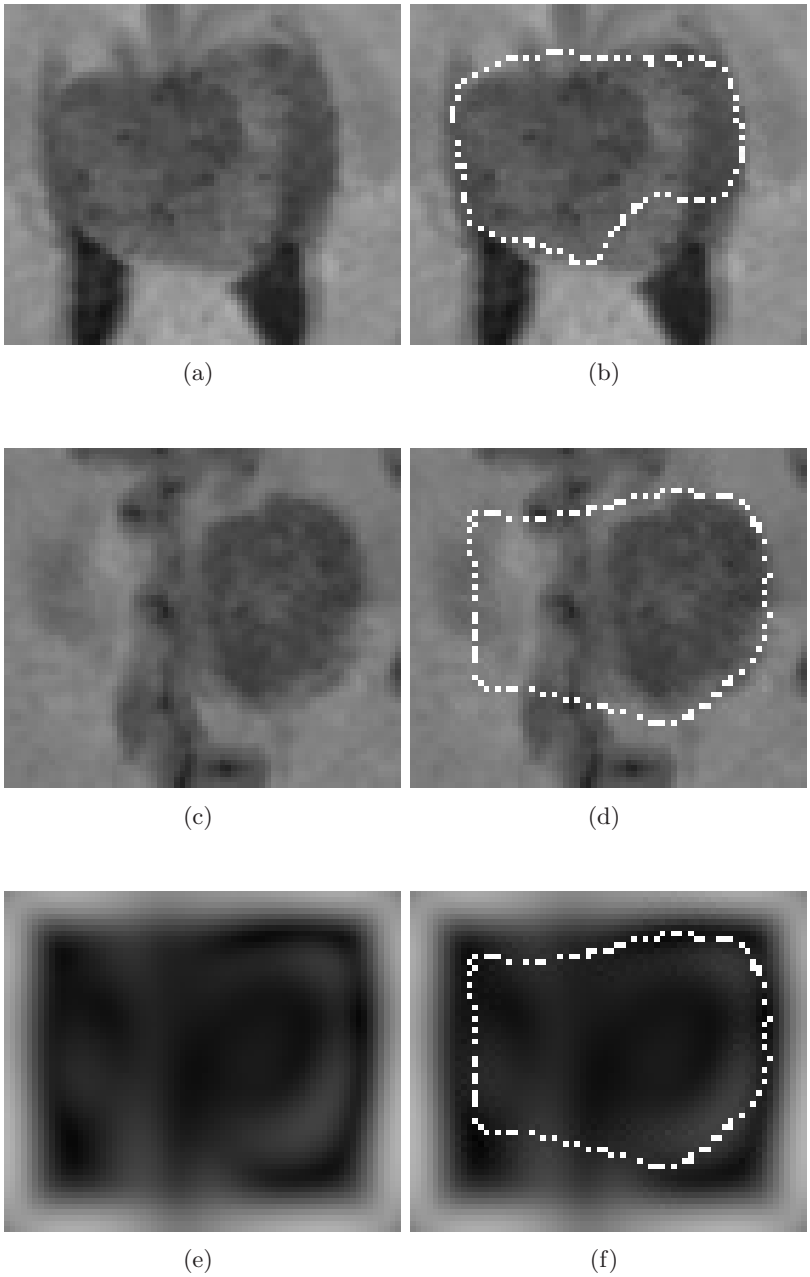
5.3.9 Dataset N378 - MRI

Figure 5.21 shows slices of the result after passing the first MR image of the N378 series through the GvfImageFilter. The master is MRI and the parameters used are: seedpoint = (36, 35, 30), tumourSize = (20, 15, 20), conductance = 5.0, $\sigma = 5.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 1.0$.

The best result is achieved on slice z=26, viewed in figure 5.21(b). The final contour is not able to capture the all the details of the tumour boundary. The reason is that the image is blurred during preprocessing. The tumour is non uniform and is very similar to its surroundings. This makes it difficult to remove the noise without reducing the quality of the edge information. This results in that the snake follows the wrong image edge at the lower left part of the tumour.

One disadvantage of the snake model is that it is not capable to split up and merge the contour. This results in erroneous segmentation results in cases like the one in figure 5.21(d).

The result is generally of low quality and is not usable for any practical applications.



(a) (b)

(c) (d)

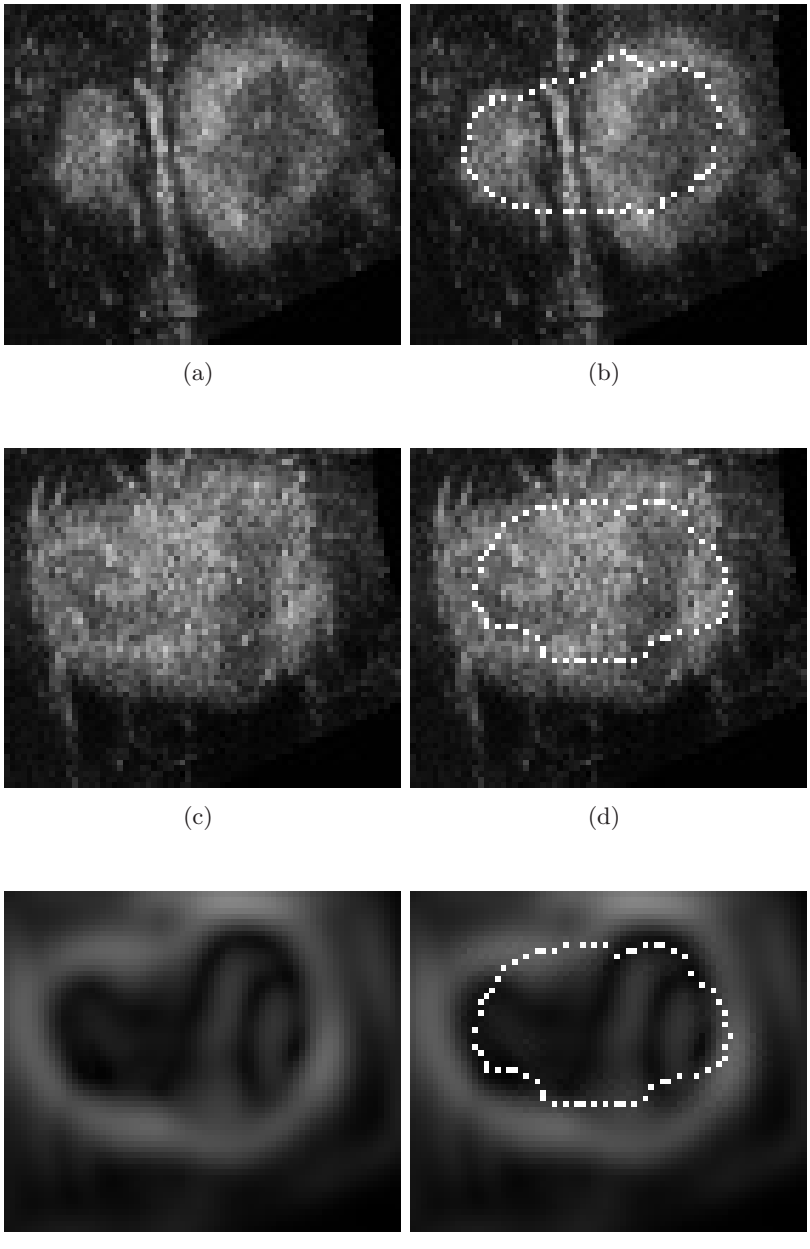
(e) (f)

Figure 5.21: Results of running GvfSnakeFilter on dataset N378 - MRI **(a)** Slice of original image data (slice z=26). **(b)** Slice of result on original data (slice z=26). **(c)** Slice of original image data (slice z=38). **(d)** Slice of result on original data (slice z=38). **(e)** Slice of gradient image (slice z=38). **(f)** Slice of result on gradient (slice z=38).

5.3.10 Dataset N378 - US

Figure 5.22 shows the result after passing the first US image of the N378 series through the GvfSnakeFilter. The master is MRI and the parameters used are: seedpoint = (34, 37, 25), tumourSize = (30, 20, 20), conductance = 8.0, $\sigma = 3.0$, $\alpha = 0.05$, $\beta = 0.05$, $\lambda = 2.0$.

The results are very similarly to the results we achieved with the corresponding MR image. The snake is not capable to adapt the topology changes of the tumour as shown in figure 5.22 **(b)**. The US image was smoothed with a higher conductance than the MR image, causing the edge information to be weaker and more diffuse. As we can see in figure 5.22**(d)** the image is heavily undersegmented.



(a) (b)

(c) (d)

(e) (f)

Figure 5.22: Results of running GvfSnakeFilter on dataset N378 - US **(a)** Slice of original image data (slice z=39). **(b)** Slice of result on original data (slice z=39). **(c)** Slice of original image data (slice z=31). **(d)** Slice of result on original data (slice z=31). **(e)** Slice of gradient image (slice z=31). **(f)** Slice of result on gradient (slice z=31).

5.4 Discussion

We have seen that the snake algorithm has located the tumour boundary with high precision in several cases. The algorithm typically fails at some parts of the tumour. This is often because the edges are weaker than other surrounding edges belonging to other structures or edges caused by graylevel variations within the tumour. This problem can be solved by offering a method where the user can initialise the contour more accurate. The snake algorithm could possibly perform even better if the user in addition is able to guide the snake during snake evolution by dragging the contour in the correct direction.

Finding the optimal segmentation requires a great deal of trial and error. There are several parameters that can be adjusted to get a better result. To simplify the task of producing a better result, it can be advantageous to keep some parameters constant. We have experienced that keeping $\alpha = 0.05$ and $\beta = 0.05$ and only adjusting λ gives good results. The parameters regulating the preprocessing are more difficult to find. The values of the conductance and σ depend a lot on characteristics of the image and the tumour.

Chapter 6

Level set segmentation

Level set methods are general numerical techniques for solving the problem of evolving interfaces, or boundaries. In addition to segmentation of images, level set methods are also being used to solve problems in physics, chemistry, fluid mechanics and optimal control theory, among many other areas.

In this section we will give a thorough description of the level set approach, starting out with a general description. Furthermore, four variants of the general level set method will be presented along with results from use of these methods.

6.1 Theoretical background

Presume an image, with a boundary separating two regions in the image. While most numerical techniques would use *marker points* to track this boundary, the principle of the level set approach is quite different: Instead of dealing with the boundary function itself, the level set approach converts this equation into a higher dimensional function, named *the level set function*, $\phi(x, y, t)$ (x, y are image coordinates, t denotes time). The level set function has the following property:

$$\phi(x,y,t) \begin{cases} < 0 \text{ if } (x,y) \in \Omega(t) \\ = 0 \text{ if } (x,y) \in \Gamma(t) \\ > 0 \text{ otherwise} \end{cases}$$

where $\Omega(t)$ denotes the region inside the boundary and $\Gamma(t)$ denotes the boundary itself at a given time t. An example of a level set function can be seen in figure 6.1. Evolving the boundary is done by manipulating the level set function ϕ with a differential equation ϕ_t. Extracting the boundary can easily be done from the *zero level* of the level set function: $\Gamma(t) = \{(x, y) | \phi(x, y, t) = 0\}$.

The level set equation is formulated as follows: First, the level set value of a particle on the boundary with path $x(t)$ must always must be zero,

$$\phi(x(t), t) = 0.$$

Furthermore, the chain rule then yields:

$$\phi_t + \nabla\phi((x(t), t) \cdot x'(t) = 0.$$

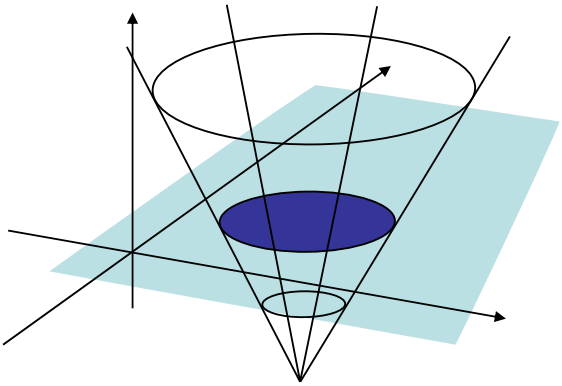


Figure 6.1: A level set function with the zero level set (dark circle).

The speed F is always in the outward normal direction and hence we can write $x'(t) \cdot n = F$, where $n = \nabla\phi/|\nabla\phi|$. This results in

$$\phi_t + F|\nabla\phi| = 0,$$
$$\text{given } \phi(x, t = 0),$$

which is the level set equation proposed by Osher and Sethian [25]. Here, F is the speed equation and it may depend on one or many factors, as will be explained in detail in the next section.

6.1.1 The speed equation

Determining a good speed equation is one of the key parts of level set segmentation. In its general form, the speed equation is made up of a *propagation*, a *curvature* and an *advection* term:

$$F = F_{prop} + F_{curv} + F_{adv},$$

the different terms are defined as follows:

The propagation term describes how the boundary expands. An example of this term is $F_{prop} = F_0$, where the boundary expands with a constant speed in its normal direction.

The curvature term describes the dependence of the speed on the boundary. An example of this term is $F_{curv} = -\varepsilon\kappa$, where ε is a constant and κ is the mean curvature. This results in a curvature term which value is proportional to the curvature.

The advection term describes how the boundary evolves based on the structure of the underlying image. An example of this term is an underlying velocity field calculated independent of the boundary itself and multiplied with a vector that is normal to the boundary: $F_{adv} = \phi(x, y, t) \cdot \mathbf{n}$.

Using the above example terms, the resulting level set equation would then become:

$$\phi_t + F|\nabla\phi| = 0,$$
$$\phi_t + (F_0 - \varepsilon\kappa + \phi(x,y,t)\cdot\mathbf{n})|\nabla\phi| = 0.$$

Since the normal $\mathbf{n}$ of the boundary is given by $\nabla\phi/|\nabla\phi|$, this yields:

$$\phi_t + F_0|\nabla\phi| + \phi(x,y,t)\cdot\nabla\phi = \varepsilon\kappa|\nabla\phi|.$$

While the speed equation defined here only relies on information from the curvature and the underlying image, additional information may also be included. An example of such information may be statistical models of the distribution of the various tissue types in a brain.

6.1.2 Narrow band

The straightforward way of solving a level set problem tries to compute answers for the entire computational domain. This means that all the level sets are updated, not only the zero level set that contains the boundary itself. Unfortunately, this leads to a slow solution. *The narrow band level set method*, on the other hand, narrows the computational domain down and thus leading to a much faster way of solving the problem.

The basic idea behind the narrow band approach is to compute solutions within a narrow band of user-specified width close to the moving boundary. This is carried out by calculating and freeze the level set values within the narrow band. Calculating new values of the boundary itself is then carried out in i normal way. Once the boundary moves close to the border of the narrow band the values within the narrow band is recalculated and frozen. An example of a boundary and a narrow band can be seen in figure 6.2.

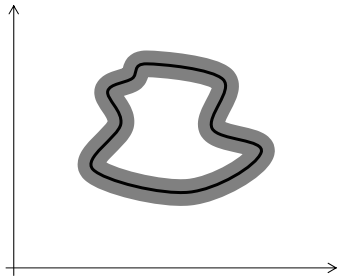


Figure 6.2: A boundary (black line) and the corresponding narrow band (dark grey colour).

The Narrow band approach provides a significant computational benefit: A calculation over the entire computational band would result in $O(N^3)$ operations per time step in three dimensions. Assuming that the boundary consists of $O(N^2)$ points in three dimensions would require $O(kN^2)$ operations in the narrow band approach, where k is the width of the narrow band. In addition, the general level set approach requires the speed function F to take into consideration the entire domain, not only the zero level set with the boundary itself. In this approach F only has to look at the points close to the boundary [27].

6.2 Fast marching level set

In this section we present the theory behind the Fast marching variant of the Level Set approach. We will also describe the implementation as well as presenting the results.

6.2.1 Theoretical background

In cases where the speed function $F > 0$, the fast marching approach may be used. While in the general level set approach the boundary may move both backward and forward and reach any point in the domain multiple times, this is not the case with a F that always is positive. As the boundary evolves, the boundary will only cross one point in the domain once, and the arrival time $T(x, y)$ for each point can be calculated. The problem now reduces to:

$$|\nabla T|F = 1, \quad T = 0 \text{ on } \Gamma,$$

where Γ is the initial location of the boundary. At time t, the boundary is now given by:

$$\Gamma(t) = \{(x, y)|T(x, y) = t\}.$$

The solution is calculated in an upwind scheme, starting out with low T values. A heap data structure is also used to store the T values. The computational benefit from this approach is substantial.

6.2.2 Implementation

The pipeline shown in figure 6.3 is implemented in our class FastMarchingLevelSetFilter. First, the input image is passed through a smoothing filter, the itk::CurvatureAnisotropicDiffusionImageFilter, that smoothes the image while preserving edge information. Next, the image is passed through a gradient filter, itk::GradientMagnitudeRecursiveGaussianImageFilter, that extracts edge information and passes this on to the itk::SigmoidImageFilter. The purpose of the sigmoid filter is to enhance intensity information of interest. Furthermore, the output from the sigmoid filter is passed on to the itk::FastMarchingImageFilter that calculates a *time crossing map*. The time crossing map describes how the propagating level set surface evolves through time. A time crossing map ideally varies slowly close to strong edges in the original image, and fast when no such information is present. To be able to extract the surface of interest, or the zero level set, the time crossing map is passed on to a threshold filter. The output of this filter is a binary image, representing the segmented surface.

Figure 6.4 shows an overview of the FastMarchingLevelSetFilter class, located in Filters/LevelSet. The class provides functions to set the most important parameters of the filter:

- TimeStep: Controls the amount of smoothing that is applied to the input image
- Sigma (σ): Controls how sensitive the gradient filter should be
- SigmoidAlpha (α): Controls the width of the intensity window
- SigmoidBeta (β): Controls the center of the intensity window

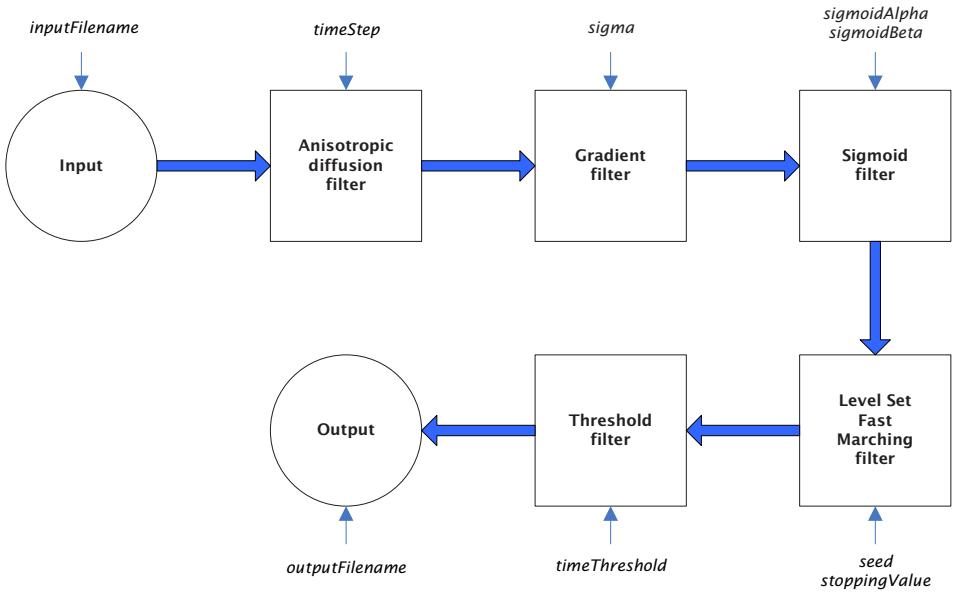


Figure 6.3: Flowchart of the FastMarchingLevelSetFilter.

- TimeThreshold: Sets the threshold level to be applied to the output from the level set filter
- Seed: Sets the initial seed point
- StoppingValue: Sets the range of the level set computation - the size of the time crossing map
- InputFilename and OutputFilename: These parameters set the filenames to read from and write to

Figure 6.4: The FastMarchingLevelSetFilter class.

6.2.3 Results

This section will present the results achieved with the FastMarchingLevelSetFilter. We will try to present the strengths and weaknesses with this segmentation approach.

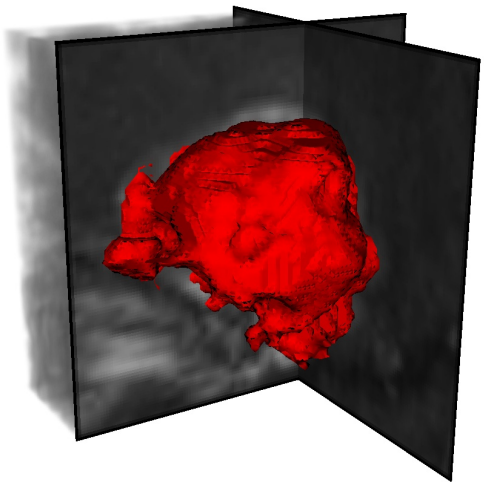
Some of the results were also presented to an expert panel. Their evaluation is presented along with the results.

Dataset N241 - MRI

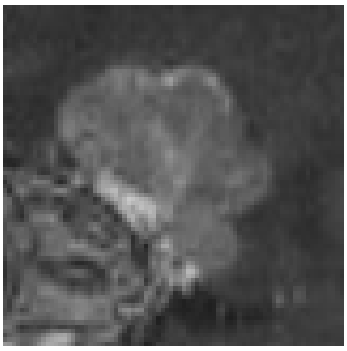
For results from the first MRI (T1) of the dataset N241, see figure 6.5. The master is US and the parameters used are: seed = (40, 40, 40), time step = 0.0625, σ = 1.5, α = -1.0, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 6,67 seconds to preprocess and 0,25 seconds to run the level set segmentation.

In terms of location, the segmentation result seems to fit the tumour well. As seen in figure 6.5 (c), the result seems to fit the boundary of the tumour fairly good on most parts, but it seems to "hang" up in small details of the tumour. The result has "holes" in the center of the tumour (figure 6.5 (c)), this is probably due to the light parts at these points. As seen in figure 6.5 (e), the result is both undersegmented at the upper part of the tumour, with some holes in the center. In the lower left corner of the tumour the level set filter has detected an arm, this is a false response. It seems that details have been picked up very well, but because of the fact that the tumour doesn't stand that well out from the rest of the brain data it is hard to adjust the sigmoid filter to enhance the right parts.

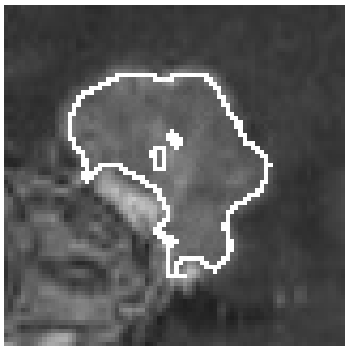
It seems like the result cannot be used directly in an operation, but it might be used for other purposes, such as navigation or to give a brief overview of the scope of the tumour. It should be possible to get a fairly good result with some manual modifications of the result.



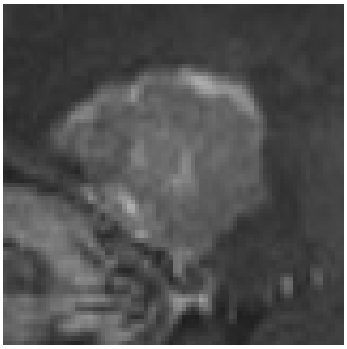
(a)



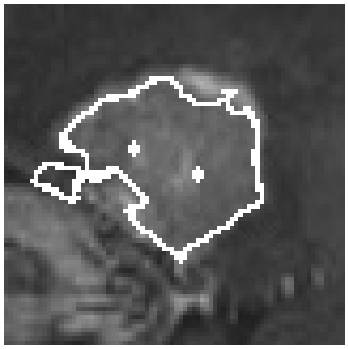
(b)



(c)



(d)



(e)

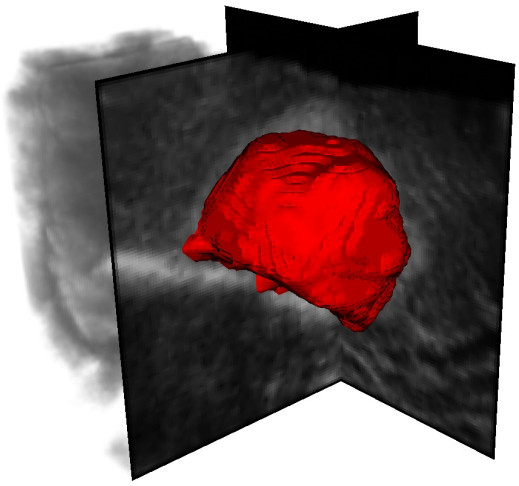
Figure 6.5: Results of running FastMarchingLevelSetFilter on dataset N241 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 41$) **(c)** Slice ($x = 41$) with result **(d)** Original slice ($x = 33$) **(e)** Slice ($x = 33$) with result

Dataset N241 - US

For results from the first US of the dataset N241, see figure 6.6. The master is US and the parameters are: seed = (40, 40, 40), time step = 0.0625, σ = 2.0, α = -1.5, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 6.66 seconds to preprocess and 0,20 seconds to run the level set segmentation.

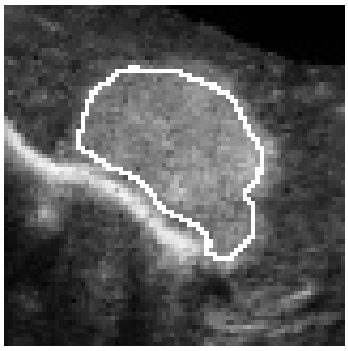
As seen from figure 6.6 (a) and (c), the ultrasound data is heavily disturbed by noise. In spite of this, the level set filter has produced good results. The main reason for this is probably that the tumour is relatively uniform in its intensity if the noise is disregarded. Although the result follows the shape of the tumour fairly good, it is slightly undersegmented along the entire contour. This is due to the initial smoothing of the input image, where the sigma is set to be larger than with the MR image.

Due to the overall undersegmentation, this result is probably useful. If a couple of iterations of a region growing filter is used to postprocess the result, this should compensate for the undersegmentation.



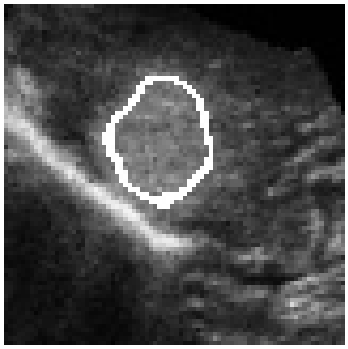
(a)

(b)



(c)

(d)



(e)

Figure 6.6: Results of running FastMarchingLevelSetFilter on dataset N241 - US **(a)** Volume representation of result **(b)** Original slice ($x = 48$) **(c)** Slice ($x = 48$) with result **(d)** Original slice ($x = 64$) **(e)** Slice ($x = 64$) with result

Dataset N351 - MRI

For results from the first MRI (T1) of dataset N351, see figure 6.7. The master is US and the parameters are: seed = (54, 58, 56), time step = 0.0625, σ = 1.0, α = -2.0, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 14,76 seconds to preprocess and 0,83 seconds to run the level set segmentation.

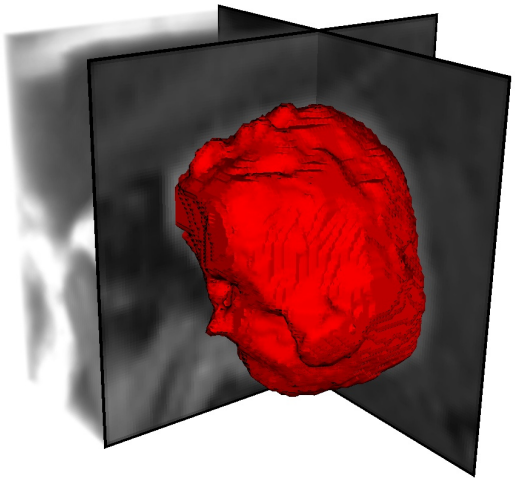
Compared with the MRI data from N241 (see figure 6.5), this tumour has a more uniform intensity distribution and stands more out from the surrounding intensities. This makes the job of the level set filter much easier and the result is very good, although a bit undersegmented. The resulting boundary tracks the contour of the tumour very well, and partly succeeds in finding small details of the tumour. This is seen in the upper part of figure 6.7 (e).

The result seems to be very useful in terms of medical application. In order to compensate for the small undersegmentation, one iteration of region growing may be run on the result

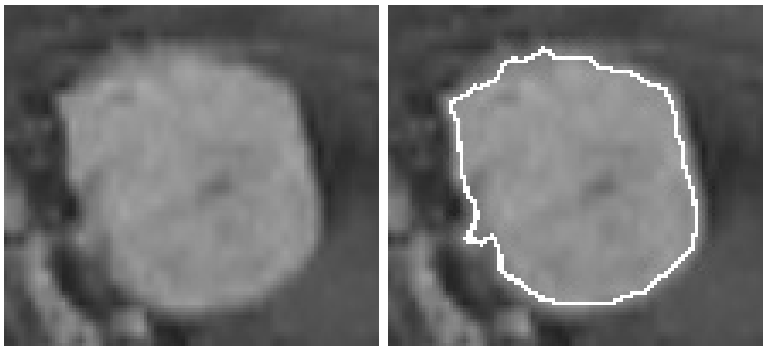
Comments from the expert panel may be seen in table 6.1.

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	1
Oversegmentation	1	1	1
Undersegmentation	2	2	2
Wrong edges traced	1	1	1
Details missing	1	1	-
Is the result usable?	yes	yes	yes
Manual adjustment required	-	1	2

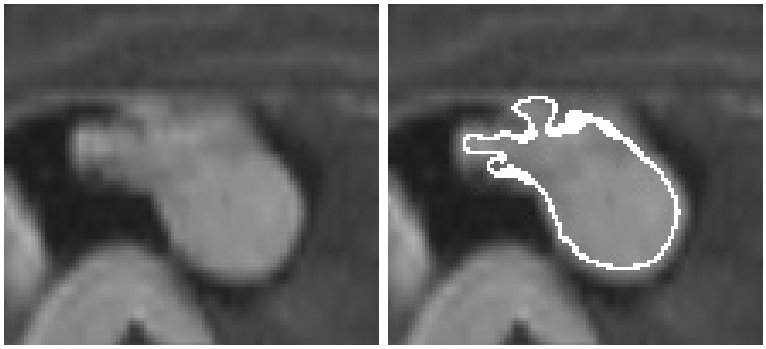
Table 6.1: Expert evaluation of segmentation result from dataset N351 - MRI (For a description of the evaluation criteria, see section 1.8)



(a)



(b) (c)



(d) (e)

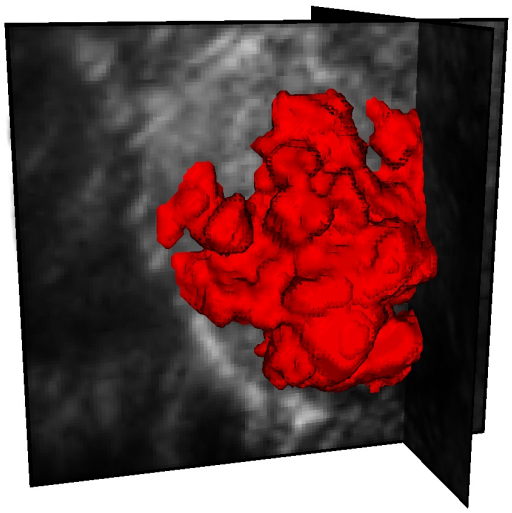
Figure 6.7: Results of running FastMarchingLevelSetFilter on dataset N351 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 59$) **(c)** Slice ($x = 59$) with result **(d)** Original slice ($x = 35$) **(e)** Slice ($x = 35$) with result

Dataset N351 - US

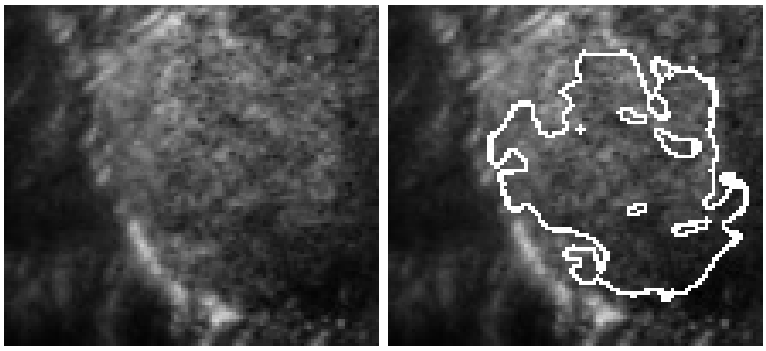
For results from the first US of dataset N351, see figure 6.8. The master is US and the parameters are: seed = (54, 58, 56), time step = 0.0625, σ = 2.0, α = -1.0, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 14,68 seconds to preprocess and 1,74 seconds to run the level set segmentation.

Due to the non-uniform intensity distribution of the tumour, the segmentation filter is facing great problems. Although the overall placement of the segmentation result partly fit the tumour, the result is not very good. The result both contains holes (figure 6.8 (b)) that should not have been detected, as well as a leaking edge on the left hand side (figure 6.8 (e)). The reason of the leaking edge is probably the diffuse/missing edge information. This problem is typical for this version of the level set filter, and as described in the geodesic level set section, improvements have been made to this model that takes especially care of this. Furthermore, the boundary often get stuck in local edge information, but more smoothing of the input image would probably result in other more important edge information being lost as well.

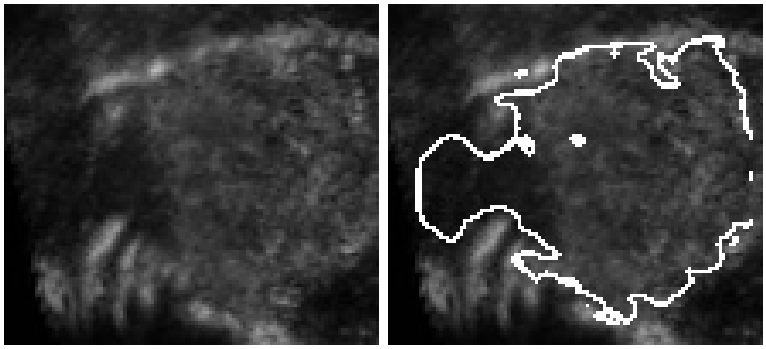
The result is not much useful for medical purposes, but may be used to get an indication of the scope of the tumour. Manual adjustment of the result would be very labour-intensive and would probably not be worth the effort.



(a)



(b) (c)



(d) (e)

Figure 6.8: Results of running FastMarchingLevelSetFilter on dataset N351 - US **(a)** Volume representation of result **(b)** Original slice ($x = 76$) **(c)** Slice ($x = 76$) with result **(d)** Original slice ($x = 54$) **(e)** Slice ($x = 54$) with result

Dataset N359 - MRI

For results from the first MRI (T1) of the dataset N359, see figure 6.9. The master is US and the parameters are: seed = (55, 50, 65), time step = 0.0625, σ = 2.5, α = -3.0, β = 3.0, time threshold = 200, stopping value = 100. Our reference computer used 13,49 seconds to preprocess and 0,53 seconds to run the level set segmentation

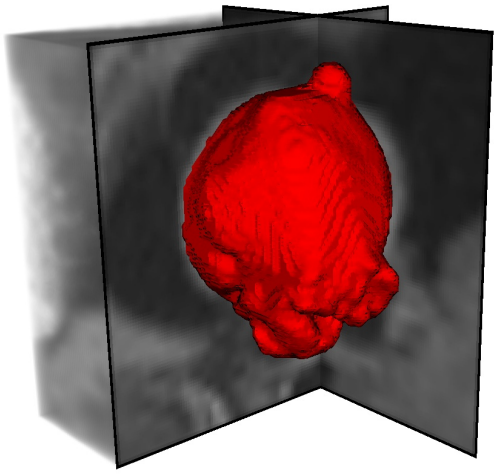
On the overall, these results approximates the tumour contour well. However, due to the heavy non-uniformity of the tumour intensities, the input image had to be heavily smoothed. The result of this is that a more uniform tumour area is obtained, but at the expense of that some edge information is lost. Furthermore, the resulting boundary is a bit undersegmented, especially on the right-hand and lower-right side of figure 6.9 (e). In figure 6.9 (c) and (e) we see that the result has hung up in small details of the tissue surrounding the tumour causing a small oversegmentation of the result.

The non-modified result should be useful in a medical context, at least as an indication of the size and scope of the tumour. Furthermore, due to the fact that the result is slightly oversegmented, it can not be used in navigation system during surgery.

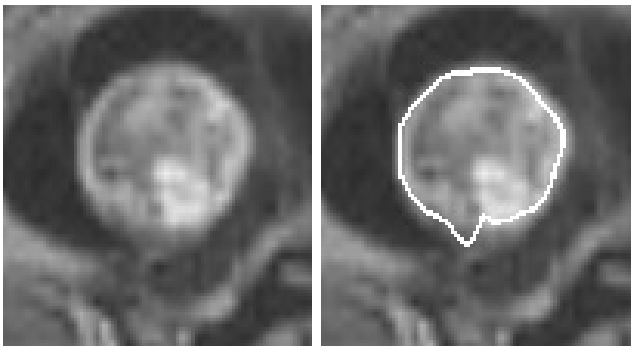
Comments from the expert panel may be seen in table 6.2. Expert 1 also stated that the "the risk of using this result is larger when it's oversegmented", and expert 2 stated "in my opinion a small part of the normal tissue was included. In addition, a part of the tumour was not segmented."

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	1
Oversegmentation	2	2	1
Undersegmentation	3	2	2
Wrong edges traced	2	2	3
Details missing	2	2	-
Is the result usable?	yes	yes	yes
Manual adjustment required	-	2	2

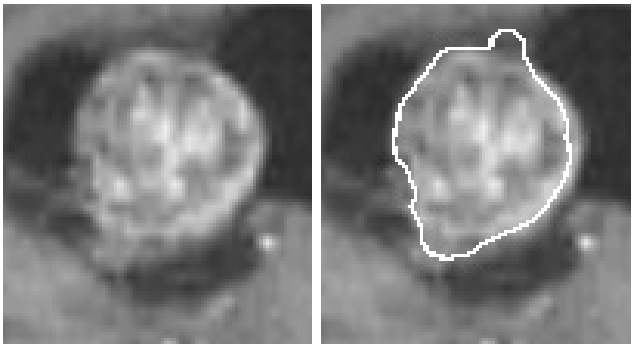
Table 6.2: Expert evaluation of segmentation result from dataset N359 - MRI (For a description of the evaluation criteria, see section 1.8)



(a)



(b) (c)



(d) (e)

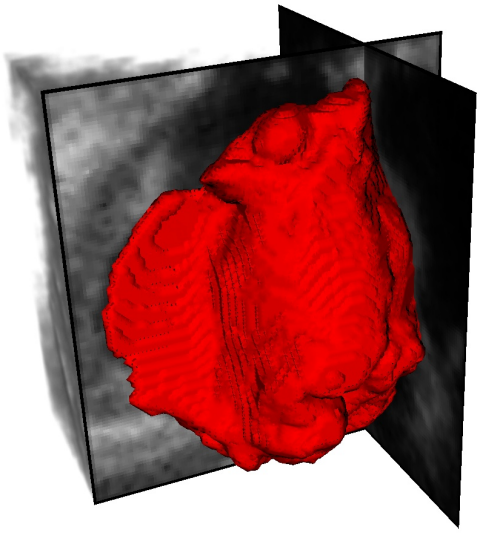
Figure 6.9: Results of running FastMarchingLevelSetFilter on dataset N359 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 41$) **(c)** Slice ($x = 41$) with result **(d)** Original slice ($x = 61$) **(e)** Slice ($x = 61$) with result

Dataset N359 - US

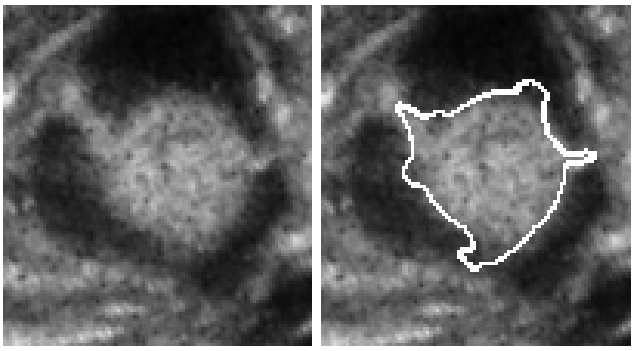
For results from the first US of the dataset N359, see figure 6.10. The master is US and the parameters are: seed = (55, 50, 65), time step = 0.0625, σ = 2.5, α = -2.5, β = 3.0, time threshold = 200, stopping value = 100. Our reference computer used 13,53 seconds to preprocess and 0,77 seconds to run the level set segmentation.

The ultrasound data had to be heavily smoothed due to the noise, thus resulting in some loss of edge information. Furthermore, although some parts of the tumour is relatively free from the surrounding tissue (figure 6.10 (c)), other parts lie close into other tissue (figure 6.10 (e)). This, along with the fact that the intensity of the tumour is similar to the one of the surrounding tissue causes the result to wander off. This effect is particularly visible in figure 6.10 (e), where the result is very oversegmented. This version of the level set filter, which is fairly simple compared with the on described in the next section, relies heavily on good results from the preprocessing stage in order to track the object of interest in an image. From the ultrasound data provided in this case, the preprocessing stages have failed to extract much useful information, thus leading to little help for the level set segmentation stage.

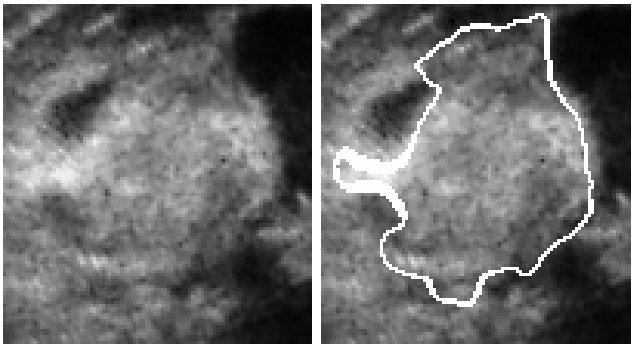
The result, as it stands is not very useful. At it's most it could probably be used by medical personnel to get an rough overview of the location of the tumour. Manual modifications to the result would probably not produce much better results.



(a)



(b) (c)



(d) (e)

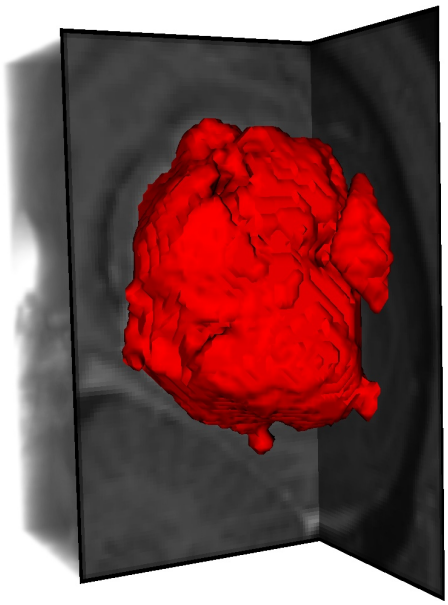
Figure 6.10: Results of running FastMarchingLevelSetFilter on dataset N359 - US **(a)** Volume representation of result **(b)** Original slice ($x = 45$) **(c)** Slice ($x = 45$) with result **(d)** Original slice ($x = 66$) **(e)** Slice ($x = 66$) with result

Dataset N360 - MRI

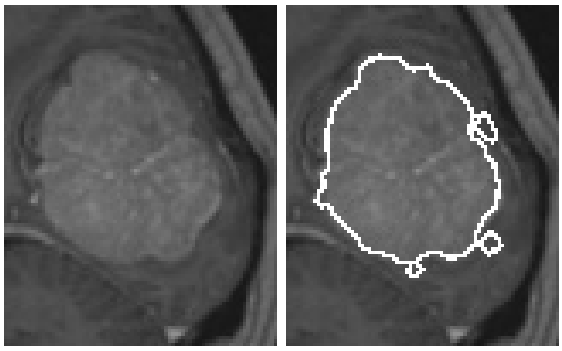
For results from the first MRI (T1) of the dataset N360, see figure 6.11. The master is MRI and the parameters: first MRI (T1), master: MRI, seed = (45, 45, 55), time step = 0.0625, σ = 1.0, α = -1.2, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 8,18 seconds to preprocess and 0,30 seconds to run the level set segmentation.

As can be seen from the results, the level set segmentation has caught the overall contour of the tumour very well. Even though the intensity of the tumour is not too different from the surrounding tissue, the tumour has a fairly uniform intensity distribution. As seen in figure 6.11 (c), the result covers three small areas that are not part of the tumour itself. This is probably due to low and diffuse edge information of some parts of the tumour boundary, and makes the level set segmentation "leak". Furthermore, as seen in figure 6.11 (e), the result is a bit undersegmented. Adjusting the sensitivity level in order to solve this problem would cause the result to include more of the brain structure seen on the lower left side of figure 6.11 (e). It is not easy to see from the lower slices, but the tumour narrows towards the upper left corner, and this is very well picked up by the level set filter.

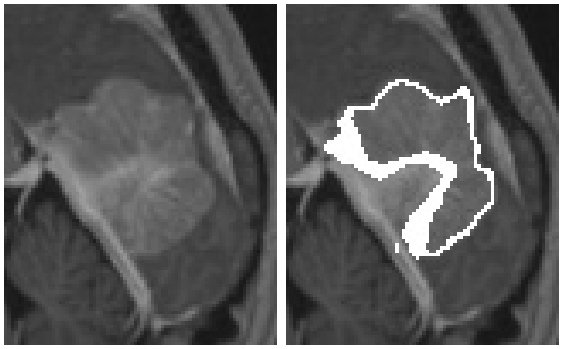
The result may very well be used for medical purposes. With some manual modifications, especially where the result has a tendency to "leak", the resulting contour will coincide with the tumour with a small error rate.



(a)



(b) (c)



(d) (e)

Figure 6.11: Results of running FastMarchingLevelSetFilter on dataset N360 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 42$) **(c)** Slice ($x = 42$) with result **(d)** Original slice ($x = 27$) **(e)** Slice ($x = 27$) with result

Dataset N360 - US

For results from the first US of the dataset N360, see figure 6.12. The master is MRI and the parameters are: seed = (40, 30, 40), time step = 0.0625, σ = 1.5, α = -2.0, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 0,22 seconds to preprocess and 9,16 seconds to run the level set segmentation.

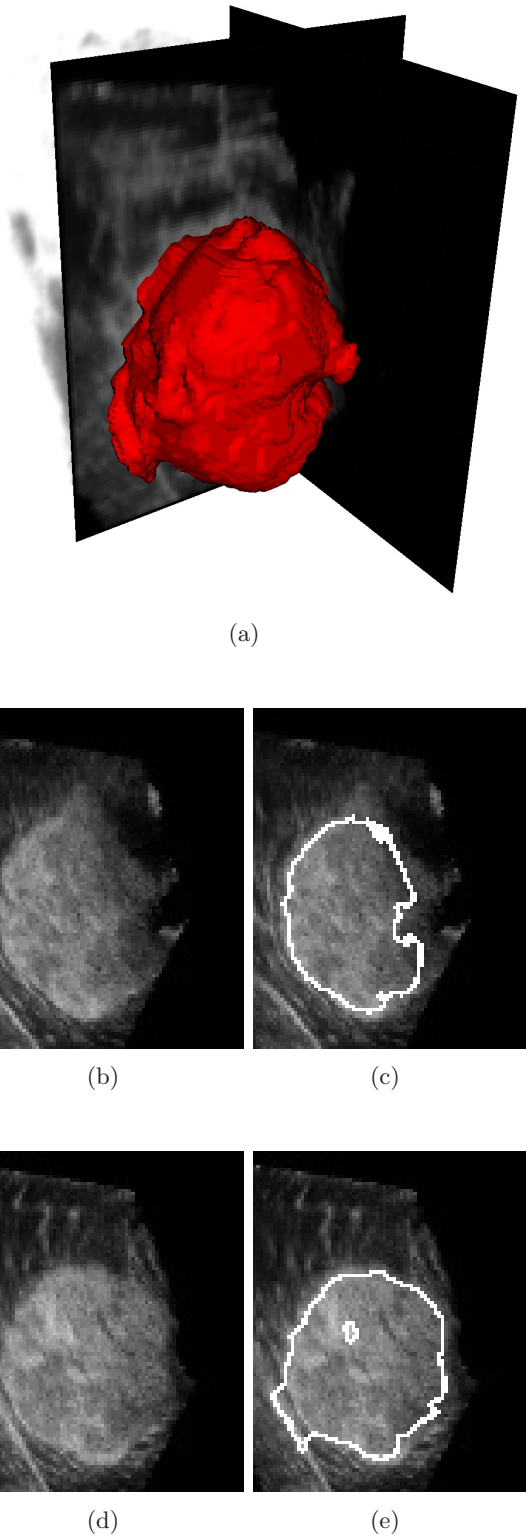
Figure 6.12 (c) shows that in tumour areas with strong edges the segmented boundary coincides very well with the tumour contour. On the other hand, when the edge information is weaker, which is the case with the right-hand side of figure 6.12 (c), the result is slightly undersegmented. Furthermore, as seen in figure 6.12 (e), the brain structure close to the lower left side of the tumour has influenced the result negatively and caused the result to be oversegmented. The hole in the center of the tumour is probably due to a light part of the tumour tissue.

The result covers the size and scope of the tumour very well but it is not useful in any medical applications where precision is required. Manual adjustment of the segmentation result could easily remove small, oversegmented parts but a lot of work would have to be done in order to make the resulting boundary fit the tumour in areas with weak edge information.

Comments from the expert panel may be seen in table 6.3. Expert 2 stated that the result was "quite good. Slightly undersegmented."

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	1
Oversegmentation	1	1	1
Undersegmentation	2	2	2
Wrong edges traced	1	2	2
Details missing	1	2	2
Is the result usable?	yes	yes	yes
Manual adjustment required	-	2	2

Table 6.3: Expert evaluation of segmentation result from dataset N360 - US (For a description of the evaluation criteria, see section 1.8)



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

Figure 6.12: Results of running FastMarchingLevelSetFilter on dataset N360 - US **(a)** Volume representation of result **(b)** Original slice ($x = 42$) **(c)** Slice ($x = 42$) with result **(d)** Original slice ($x = 31$) **(e)** Slice ($x = 31$) with result

Dataset N378 - MRI

For results from the first MRI (T1) of the dataset N378, see figure 6.13. The master is MRI and the parameters are: first MRI (T1), master: MRI, seed = (30, 30, 30), time step = 0.0625, σ = 1.0, α = -2.0, β = 1.0, time threshold = 200, stopping value = 100. Our reference computer used 3,44 seconds to preprocess and 0,13 seconds to run the level set segmentation.

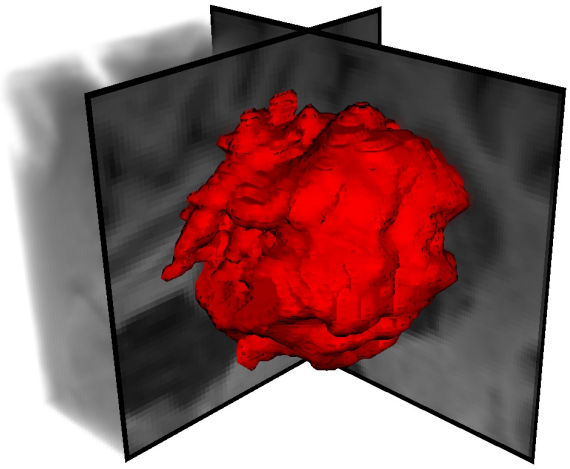
Although the intensity of the tumour is very low and it has diffuse edges, the level set filter manages to produce fairly good results on most parts of the tumour. As seen in figure 6.13 (c), the tumour undersegments in the lower right corner due to the slightly lighter tumour intensity. Altering the sigmoid filter in order to try to extend the segmentation would result in oversegmentation in other parts of the contour. Furthermore, the level set result is oversegmented, including areas not being part of the tumour as seen in the upper part of figure 6.13 (e). Small details of the tumour are captured fairly well by the level set filter on some parts of the tumour.

The result is not useful in any applications that demands high level of accuracy, but should be very useful as an overall guide to where the tumour is located. Furthermore, manual adjustment would be easy on some parts of the result, such as the part seen in figure 6.13 (c). Other parts of the result, such as those seen in figure 6.13 (e) is not so easy to improve manually.

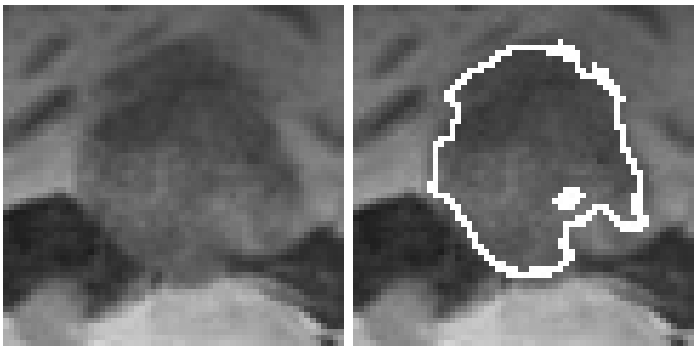
Comments from the expert panel may be seen in table 6.4. Expert 2 stated that "in my opinion the result seems to trace some of the normal tissue."

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	2	1
Oversegmentation	1	2	1
Undersegmentation	2	2	2
Wrong edges traced	2	3	2
Details missing	2	2	2
Is the result usable?	yes	yes	yes
Manual adjustment required	-	3	2

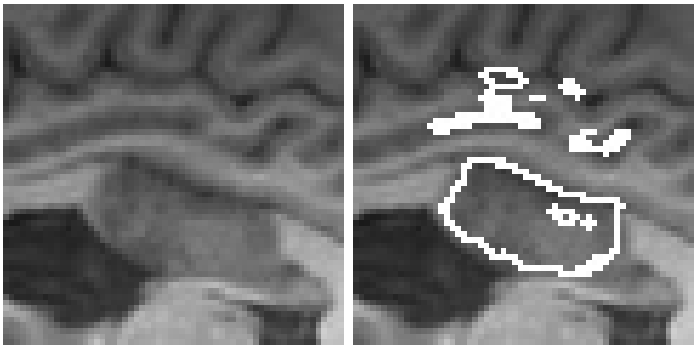
Table 6.4: Expert evaluation of segmentation result from dataset N378 - MRI (For a description of the evaluation criteria, see section 1.8)



(a)



(b) (c)



(d) (e)

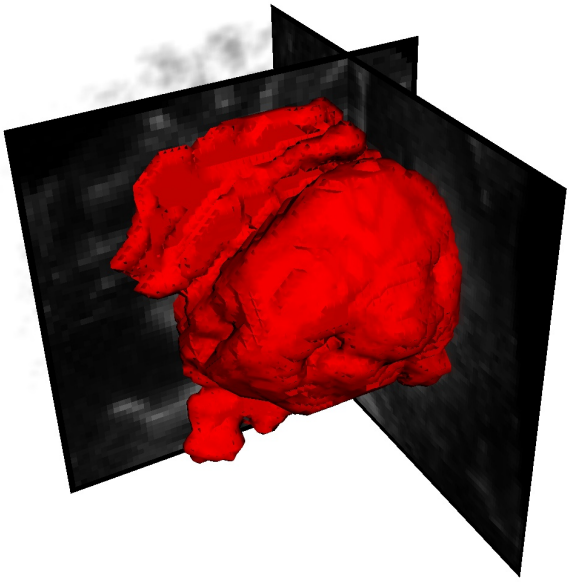
Figure 6.13: Results of running FastMarchingLevelSetFilter on dataset N378 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 44$) **(c)** Slice ($x = 44$) with result **(d)** Original slice ($x = 25$) **(e)** Slice ($x = 25$) with result

Dataset N378 - US

For results from the first US from the dataset N378, see figure 6.14. The master is MRI and the parameters are: first US, master: MRI, seed = (30, 30, 30), time step = 0.0625, σ = 2.0, α = -2.0, β = 2.0, time threshold = 200, stopping value = 100. Our reference computer used 3,44 seconds to preprocess and 0,14 seconds to run the level set segmentation.

The result is on the overall slightly undersegmented. This is probably due to the heavy smoothing of the ultrasound data. Furthermore, due to the some edges that lie on the upper side as well as the lower side of the tumour itself (see figure 6.14 (e)), the result has been oversegmented in these areas. Small details seems to be picked up well by the level set segmentation, this is seen in the lower part of the tumour area in figure 6.14 (e).

The result, as it stands, is not useful in any medical application where high precision is required. Removing false edges of the result manually would improve the result significantly and would probably make the result more useful.

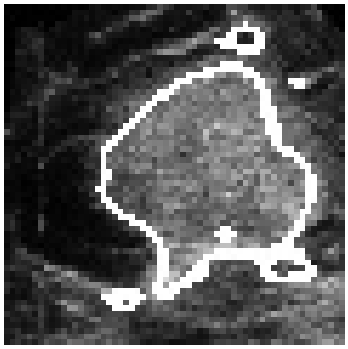


(a)

(b)

(c)

(d)



(e)

Figure 6.14: Results of running FastMarchingLevelSetFilter on dataset N378 - US **(a)** Volume representation of result **(b)** Original slice ($x = 40$) **(c)** Slice ($x = 40$) with result **(d)** Original slice ($x = 36$) **(e)** Slice ($x = 36$) with result

6.3 Geodesic level set segmentation

Most general level set approaches have proven to perform well when used on objects with a well defined contour. On the other hand, when used on objects with a more diffuse contour or objects that have gaps in their contour, the boundary tends to "leak", and thus producing incorrect results. The geodesic level set filter is based on the paper by Caselles et al [7], which presented improvements to the original level set method in order to solve the "leaking problem". This problem was related to cases where edges would "leak" and not be continuous in an image. Their approach was to take some of the ideas behind the energy minimization approach from the classic snake, and merge with the level set method. They presented results that proved that their model performed better than other level set models when dealing with images that had weak or missing edge information.

6.3.1 Theoretical background

According to Caselles et al. [7] the traditional snake model, that relies on energy minimization in order to detect a surface in an image, can be transformed into a geodesic curve. A geodesic curve is the (local) minimal distance between two given points.

We'll will present a short summary of how the snake model is transformed into the geodesic curve. First, consider the classic snake model[1] presented by Kass et al [18]:

$$E(C) = \alpha \int_0^1 |C'(q)|^2 dq + \beta \int_0^1 |C''(q)|^2 dq - \lambda \int_0^1 |\nabla I(C(q))| dq, \tag{6.1}$$

where α, β, λ are real, positive constants and I is the input image. The parameters α and β have control over the amount of smoothing of the snake (internal energy), while λ draws the snake towards edges in an image (external energy). Solving a segmentation problem using this snake model involves finding the curve C that minimizes the energy E for some given α, β and λ. One of the problems with this model is that it cannot segment more than one object at a time. This is not the case with the level set approach, where changes in topology are handled implicitly. An another problem with the snake model is that adjusting the parameters involves a trade-off between smoothness of the snake itself and proximity to the object in the image. Caselles et al. [7] showed that when by using the geodesic approach to solve the snake model, a regularization effect is introduced by the use of curvature based curve flows. That means that the smoothing is automatically handled by the way the problem is solved and there is no need for this to be handled explicitly. Due to this, the parameter β that controls the high-order smoothness of the curve, may be set equal to zero when transforming from the snake model to a geodesic model. The smoothness of the curve will instead be taken care of by the geodesic curve. With $\beta = 0$, (6.1) reduces to

$$E(C) = \alpha \int_0^1 |C'(q)|^2 dq - \lambda \int_0^1 |\nabla I(C(q))| dq. \tag{6.2}$$

[1] The notation used here can be transformed into the notation used in chapter 5 (equation 5.1) by replacing $C(q)$ with $v(s)$.

Furthermore, a curve evolution equation is derived from the classical snake model:

$$\frac{\partial C(t)}{\partial t} = g(I)\kappa\vec{N} - (\nabla g \cdot \vec{N})\vec{N}, \tag{6.3}$$

where κ is the Eucledian curvature and $\vec{N}$ is the unit inward normal to the curve. $g(r)$ is a function with the property that $g(r) \to 0$ when $r \to \infty$. The detected object may then be extracted when equation (6.3) has reached a steady solution. Next, assume that the curve C is the level-set of a function $\phi : [0, M] \times [0, N] \to \mathbf{R}$. This means that the curve C coincides with the set of points $\phi =$ constant in a *time crossing map* of size $[M, N]$, making ϕ an implicit representation of C. Each position in the time crossing map contains a number, telling when the evolving curve will reach that position. It can be shown [7], that if the planar curve C evolves according to

$$C_t = \beta\vec{N}, \tag{6.4}$$

for a given function β, then the embedding function ϕ should deform in the following way:

$$\phi_t = \beta|\nabla\phi|, \tag{6.5}$$

where β is computed on the level-sets. Furthermore, Caselles et al. proved that solving the geodesic problem is equivalent to finding the steady state solution $\frac{\partial\phi}{\partial t} = 0$ of the evolution equation:

$$\begin{aligned}\frac{\partial\phi}{\partial t} &= |\nabla\phi| \text{ div } \left(g(I)\frac{\nabla\phi}{|\nabla\phi|}\right) \\ &= g(I)|\nabla\phi| \text{ div } \left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \nabla g(I) \cdot \nabla\phi \\ &= g(I)|\nabla\phi|\kappa + \nabla g(I) \cdot \nabla\phi,\end{aligned} \tag{6.6}$$

with the starting condition $(\phi(0, C) = u_0(C))$. The curvature term κ is equivalent to div $\left(\frac{\nabla\phi}{|\nabla\phi|}\right)$, this is used in the above equation. The deduction has so far shown that there exists a connection between the original snake model and a geodesic curve equation. These results may then be used to find a level-set equation for use in segmenting objects.

First, consider the following level-set equation:

$$\begin{aligned}\frac{\partial\phi}{\partial t} &= g(I)|\nabla\phi| \text{ div } \left(\frac{\nabla\phi}{|\nabla\phi|}\right) + c\, g(I)|\nabla\phi| \\ &= g(I)(c + \kappa)|\nabla\phi|,\end{aligned} \tag{6.7}$$

where

$$g = \frac{1}{1 + |\nabla(\hat{I})|^p}.$$

Here, $\hat{I}$ is a Gaussian smoothed version of the input image I and $p = 1$ or 2. The equation (6.7) is an old model, proposed in an earlier paper by Caselles et al. [6]. The

problem with this model is that it works well on ideal edges ($g = 0$), but not so well on diffuse edges. In the newer paper [7], Caselles et. al proposed an extension to (6.7), based on the results from the deduced connection between the snake model and geodesic curve flow. This extension takes especially care of diffuse edges, enabling the level set segmentation to stop at these edges as well as the strong ones. Similar to the connection between (6.4) and (6.5), it can be shown [7] that the flow

$$\phi_t = (c + \kappa)|\nabla\phi|$$

is equal to the curve evolution

$$C_t = (c + \kappa)\vec{N},$$

where $\vec{N}$ is the inward normal to the curve. Using this result when comparing the old model (6.3) with the new (6.7), one sees that the new model has one extra term, $\nabla g \cdot \nabla\phi$. This term is responsible for attracting the curve towards the boundaries of objects. This works because ∇g points towards the middle of an edge. An example of how this attraction force works is shown in figure 6.15. In addition, the old model (6.3) doesn't take into consideration possible gaps that might appear in contours in an image. This is handled by the new model as well.

To summarize the theoretical background shown in this section, we present the new model for use with level set segmentation that were proposed by Caselles et al. [7]:

$$\frac{\partial\phi}{\partial t} = |\nabla\phi| \ \mathrm{div}\left(g(I)\frac{\nabla\phi}{|\nabla\phi|}\right) + c\,g(I)|\nabla\phi|, \tag{6.8}$$

which is equivalent to

$$\frac{\partial\phi}{\partial t} = g(c + \kappa)|\nabla\phi| + \nabla\phi \cdot \nabla g. \tag{6.9}$$

The extra term, $\nabla\phi \cdot \nabla g$, pulls the boundary back if it passes an edge in the image, thus leading to a more robust segmentation of edges in the given image. The advantage of this model is that it takes the best features from the snake model and combines with the automatic handling of topological changes of a level set. This model is implemented in the itk::GeodesicActiveContourLevelSetImageFilter.

6.3.2 Implementation

Figure 6.17 shows an overview of the different steps in the GeodesicLevelSetFilter. First, the input image is passed through the itk::CurvatureAnisotropicDiffusionImageFilter, that smoothes the image while preserving edge information. Next, the smoothed image is passed through the itk::GradientMagnitudeRecursiveGaussianImageFilter before it is send to the itk::SigmoidImageFilter. The former extracts edge information while the latter enhances specified ranges of image intensity. The itk::FastMarchingImageFilter creates a map, showing how the boundary should evolve based on the seed points. This map, along with the output from the sigmoid filter is processed in the itk::GeodesicActiveContourLevelSetImageFilter that performs the segmentation itself. Finally, the output is passed through a threshold filter, the itkBinaryThresholdImageFilter, that outputs the segmented boundary of the input image.

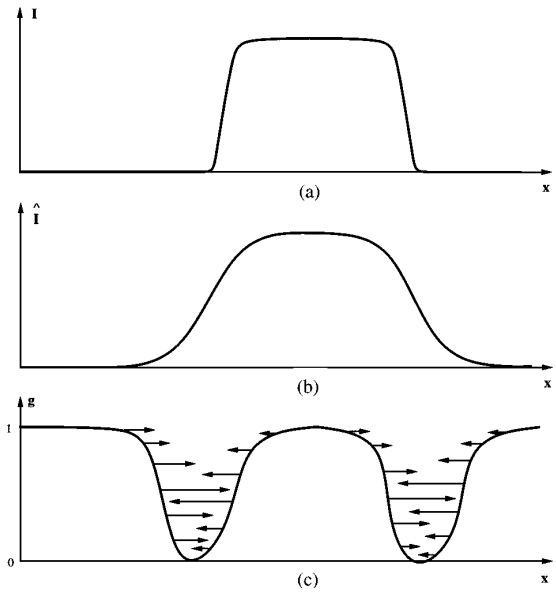


Figure 6.15: 1D example showing the effect of the attraction force **(a)** original 1D signal I **(b)** smoothed signal $\hat{I}$ **(c)** the derived stopping function g. The arrows show how the term $\nabla g \cdot \nabla \phi$ points toward the center of the edge [7].

An overview of the different methods can be seen in figure 6.16. Below is a list of the most important parameters along with a brief description of their purpose. The curvature term is fixed to be 1.0 in this implementation.

- Seed: Sets the initial seed point. This is typically in the center of the region to be segmented
- InitialDistance: The initial radius of the object to be segmented
- Sigma (σ): Controls the amount of smoothing of the input image
- SigmoidAlpha (α): Controls the width of the intensity window of the sigmoid filter
- SigmoidBeta (β): Controls the center of the intensity window of the sigmoid filter
- PropagationScaling: Controls how much the evolving boundary is allowed to propagate during the segmentation process
- InputFileName and OutputFileName: Set the filenames of the input and output image, respectively

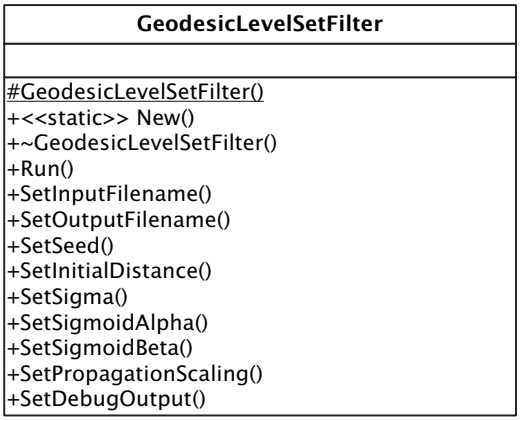


Figure 6.16: The GeodesicLevelSetFilter class.

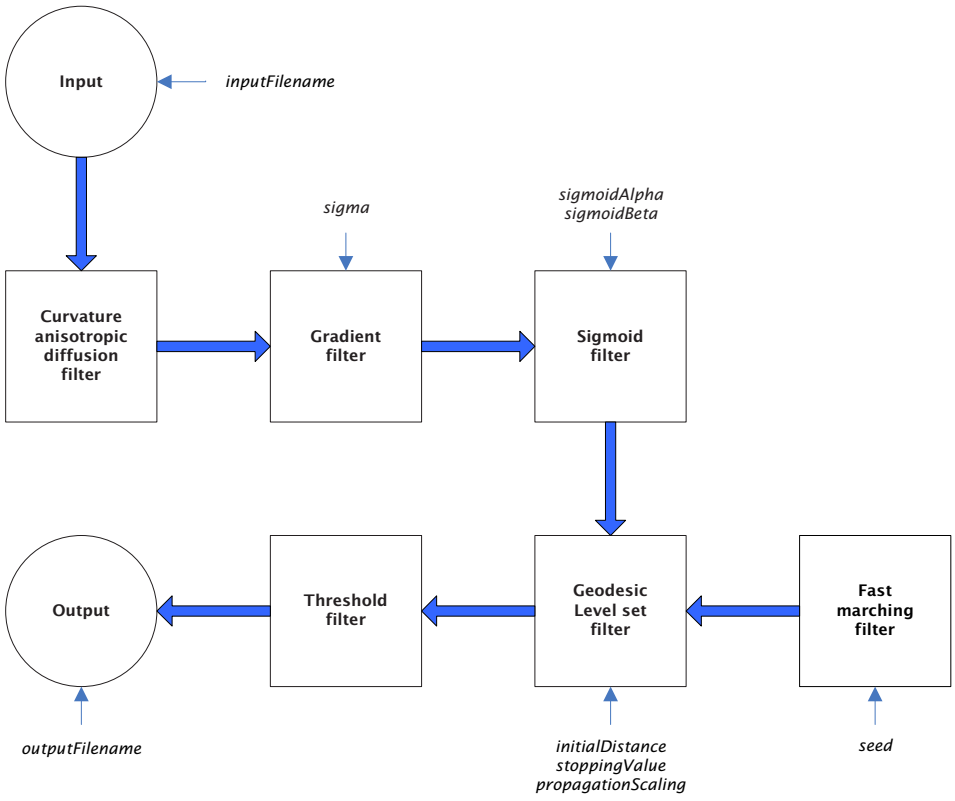


Figure 6.17: Flowchart of the GeodesicLevelSetFilter class.

6.3.3 Results

This section will present the results achieved with the GeodesicLevelSetFilter. We will try to present the strengths and weaknesses with this segmentation approach.

Dataset N241 - MRI

For results from the first MRI (T1) from the dataset N241, see figure 6.18. The master is US and the parameters are: seed = (40, 40, 40), initial distance = 10, σ = 1.0, α = -1.5, β = 2.0, propagation scaling = 1.0. Our reference computer used 6,53 seconds to preprocess and 9,56 seconds to run the level set segmentation.

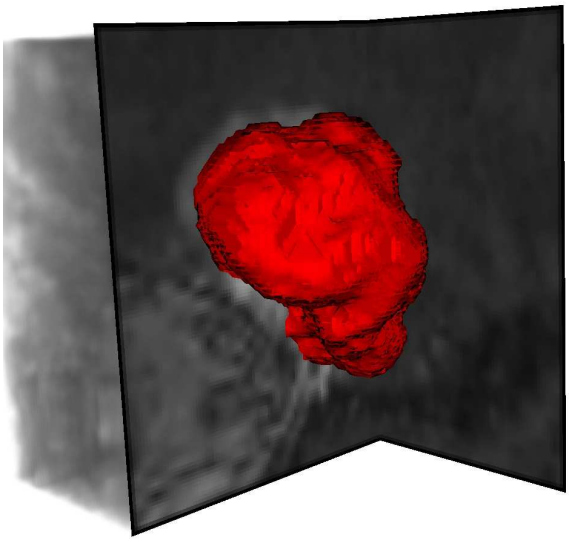
On some parts of the tumour, such as the on seen in figure 6.18 (c), the geodesic level set segmentation has managed to trace the tumour contour almost perfectly. On other parts, as in figure 6.18 (e), the result is undersegmented. This is due to the fact that the intensity distribution of the tumour lie close to the intensity distribution of the surrounding tissue. Adjusting the sigmoid filter to compensate for this would result in the filter to pick up more of the surrounding tissue not belonging to the tumour. Furthermore, the level set seems to have picked up small details of the tumour well.

The result gives a good overview of the scope of the tumour, but is not useful in medical applications where precision is required. Manual modification of the result would probably not yield much better result, either.

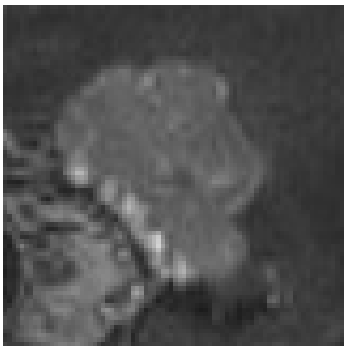
Comments from the expert panel may be seen in table 6.5.

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	2
Oversegmentation	1	1	1
Undersegmentation	2	3	2
Wrong edges traced	1	1	2
Details missing	3	3	-
Is the result usable?	yes	yes	yes
Manual adjustment required	2	2	2

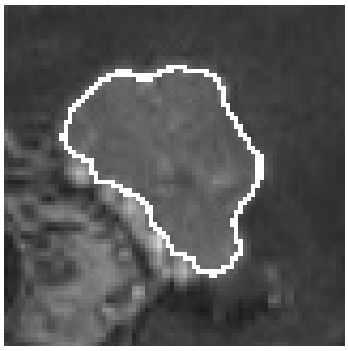
Table 6.5: Expert evaluation of segmentation result from dataset N241 - MRI (For a description of the evaluation criteria, see section 1.8)



(a)

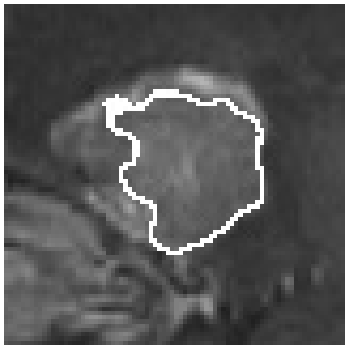


(b)



(c)

(d)



(e)

Figure 6.18: Results of running GeodesicLevelSetFilter on dataset N241 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 46$) **(c)** Slice ($x = 46$) with result **(d)** Original slice ($x = 32$) **(e)** Slice ($x = 32$) with result

Dataset N241 - US

For results from the first US of the dataset N241, see figure 6.19. The master is US and the parameters are: seed = (45, 50, 55), initial distance = 5, σ = 2.0, α = -1.7, β = 2.0, propagation scaling = 100.0. Our reference computer used 6,53 seconds to preprocess and 14,78 seconds to run the level set segmentation.

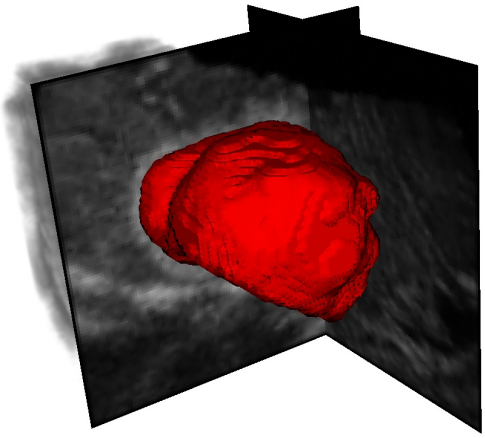
Even if the data is heavily disturbed by noise, the geodesic level set filter has managed to find the contour of the tumour. The segmentation result is slightly undersegmented along the entire contour due to the initial smoothing. Furthermore, the filter has been influenced by some irregularities in the tumour area, such as the hole in the lower part seen in figure 6.19 (e). This happens because the geodesic filter heavily relies on the edge information from the preprocessing stages.

The result should be very useful as it stands. Performing some manual modifications, such as guiding the result away from small intensity irregularities would certainly yield a much better result.

Comments from the expert panel may be seen in table 6.6.

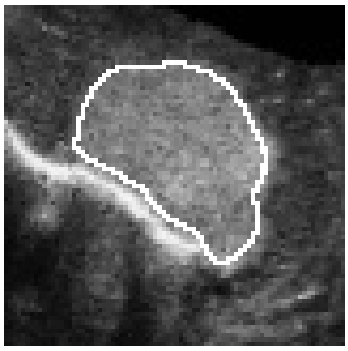
Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	1
Oversegmentation	1	1	1
Undersegmentation	1	2	3
Wrong edges traced	1	1	2
Details missing	-	1	-
Is the result usable?	yes	yes	yes
Manual adjustment required	1	1	2

Table 6.6: Expert evaluation of segmentation result from dataset N241 - US (For a description of the evaluation criteria, see section 1.8)



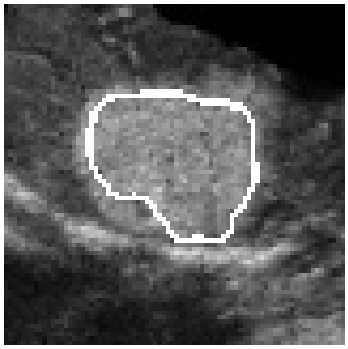
(a)

(b)



(c)

(d)



(e)

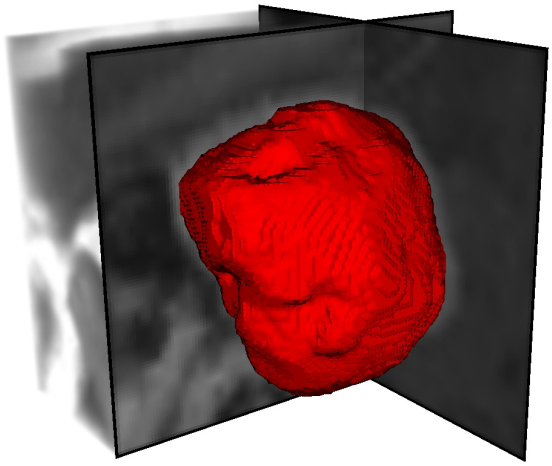
Figure 6.19: Results of running GeodesicLevelSetFilter on dataset N241 - US **(a)** Volume representation of result **(b)** Original slice ($x = 46$) **(c)** Slice ($x = 46$) with result **(d)** Original slice ($x = 32$) **(e)** Slice ($x = 32$) with result

Dataset N351 - MRI

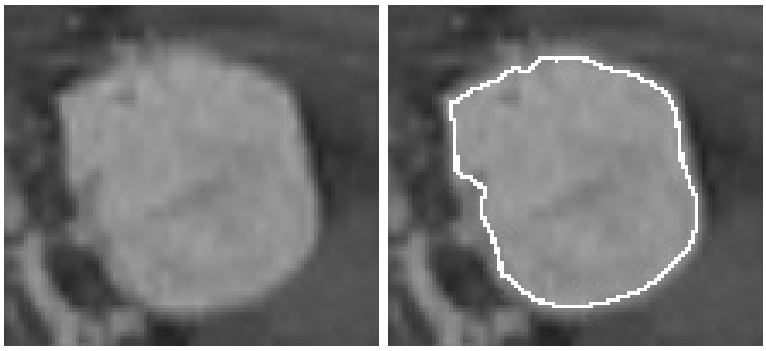
For results from the first MRI (T1) of the data set N351, see figure 6.20. The master is US and the parameters are: seed = (55, 55, 50), initial distance = 10, σ = 1.0, α = -1.5, β = 2.0, propagation scaling = 1.0. Our reference computer used 14,58 seconds to preprocess and 34,55 seconds to run the level set segmentation.

The relatively uniform distribution of intensities in the tumour, along with relatively sharp edges has resulted in a very good geodesic level set result. The contour is slightly undersegmented, this is probably due to the initial smoothing. We see that small details are picked up very well by the filter, this is best seen in the upper right part of the tumour in figure 6.20 (e).

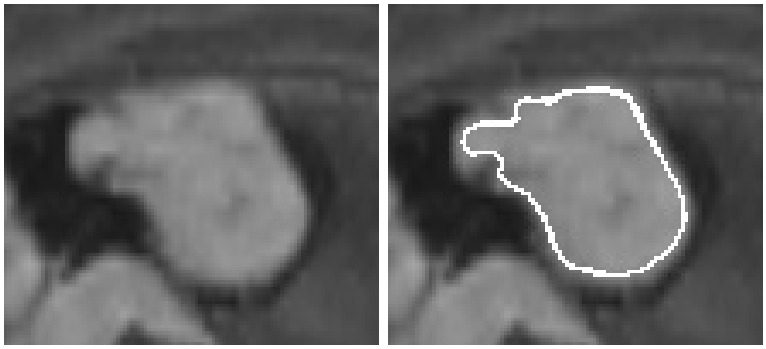
This result is one of the best results produced by the segmentation methods we have investigated. It is certainly useful in a medical perspective, particularly in situations where precision is required.



(a)



(b) (c)



(d) (e)

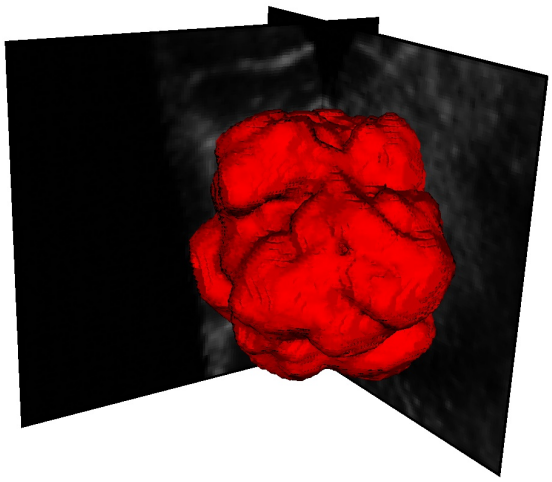
Figure 6.20: Results of running GeodesicLevelSetFilter on dataset N351 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 61$) **(c)** Slice ($x = 61$) with result **(d)** Original slice ($x = 40$) **(e)** Slice ($x = 40$) with result

Dataset N351 - US

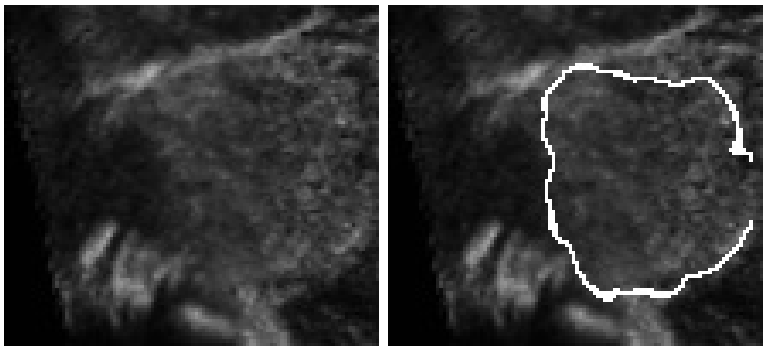
For results from the first US of the dataset N351, see figure 6.21. The master is US and the parameters are: seed = (60, 75, 50), initial distance = 5, σ = 2.5, α = -1.0, β = 1.0, propagation scaling = 100.0. Our reference computer used 14,41 seconds to preprocess and 41,55 seconds to run the level set segmentation.

Being one of the most corrupted ultrasound data in our set of test data, this is a real challenge to the level set filter. Some parts of the tumour has a relatively uniform distribution of intensity compared to the rest of the image, leading to the geodesic level set finding a good contour (as seen in figure 6.21 (c)). Other parts, such as the one seen in figure 6.21 (e), have lighter/non-uniform intensity distributions and the result is heavy undersegmented.

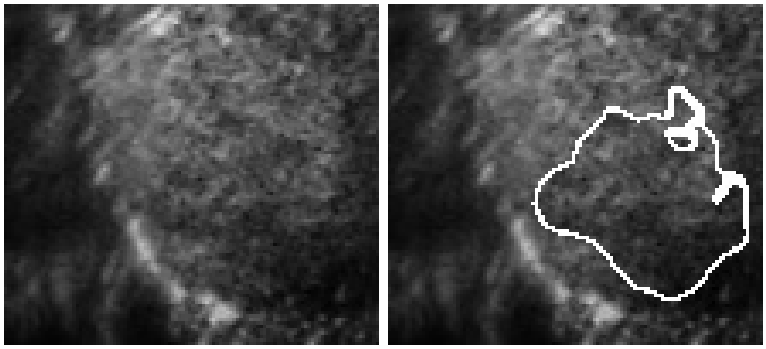
Although being undersegmented, the result gives an overview of the tumour. The result is not useful in any medical application where precision is required.



(a)



(b) (c)



(d) (e)

Figure 6.21: Results of running GeodesicLevelSetFilter on dataset N351 - US **(a)** Volume representation of result **(b)** Original slice ($x = 51$) **(c)** Slice ($x = 51$) with result **(d)** Original slice ($x = 75$) **(e)** Slice ($x = 75$) with result

Dataset N359 - MRI

For results from the first MRI (T1) of the dataset N359, see figure 6.22. The master is US and the parameters are: seed = (55, 50, 65), initial distance = 20, σ = 2.5, α = -2.0, β = 2.5, propagation scaling = 100.0. Our reference computer used 13,31 seconds to preprocess and 21,5 seconds to run the level set segmentation.

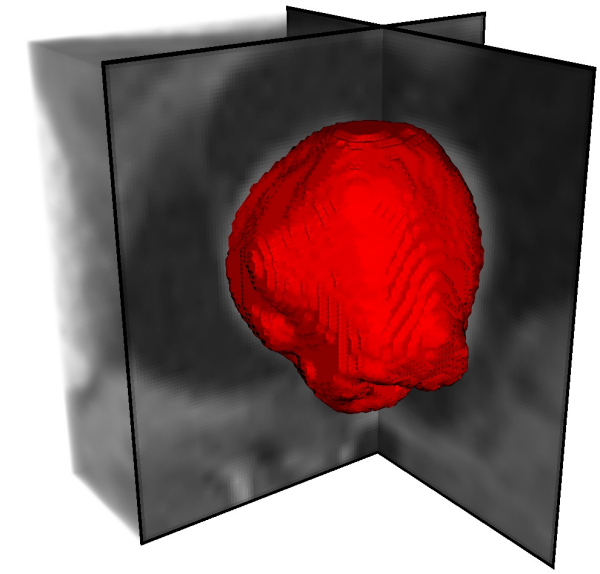
The non-uniform distribution of the tumour intensity makes the job of the geodesic level set real hard. The input data has to be heavily smoothed and some edge information is easily lost in this process. We see that the level set filter has managed to trace the contour fairly well in figure 6.22 (c), although being undersegmented on some parts. Furthermore, the bright area in figure 6.22 (c) has not been included by the filter, this is because this particular area has a strong edge. As seen in figure 6.22 (e), more diffuse parts of the tumour has not been picked up by the filter at all, this is due to the lack of edge information in the input image.

Although not including all parts of the tumour, this result gives an indication of the shape and scope of the tumour. It would probably take a lot of manual modifications in order to improve the result significantly.

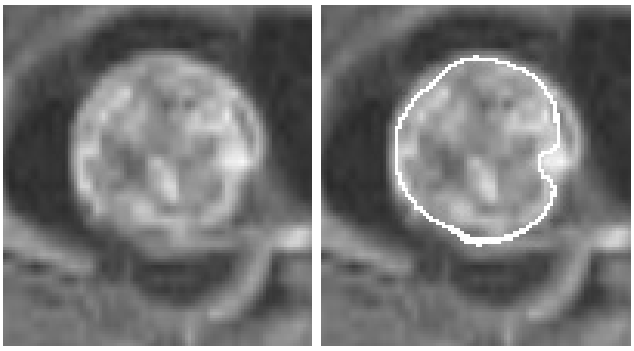
Comments from the expert panel may be seen in table 6.7. Expert 2 stated that "in the sense of usability, this result may very well be used during surgery planning and to some degree during guided surgery."

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	1
Oversegmentation	1	1	1
Undersegmentation	3	2	3
Wrong edges traced	2	2	-
Details missing	2	2	3
Is the result usable?	yes	yes	yes
Manual adjustment required	-	3	2

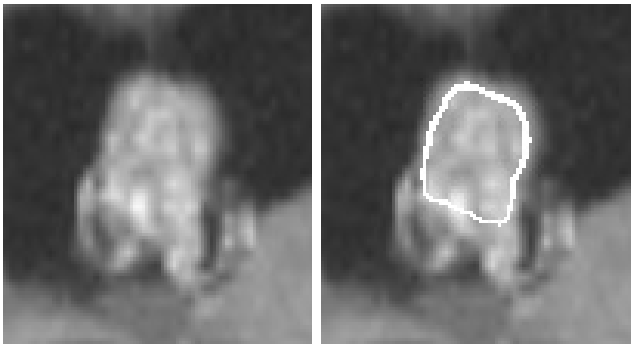
Table 6.7: Expert evaluation of segmentation result from dataset N359 - MRI (For a description of the evaluation criteria, see section 1.8)



(a)



(b) (c)



(d) (e)

Figure 6.22: Results of running GeodesicLevelSetFilter on dataset N359 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 48$) **(c)** Slice ($x = 48$) with result **(d)** Original slice ($x = 77$) **(e)** Slice ($x = 77$) with result

Dataset N359 - US

For results from the first US of the dataset N359, see figure 6.23. The master is US and the parameters are: first US, master: US, seed = (55, 50, 65), initial distance = 20, σ = 2.5, α = -1.5, β = 2.0, propagation scaling = 10.0. Our reference computer used 13,34 seconds to preprocess and 27,08 seconds to run the level set segmentation.

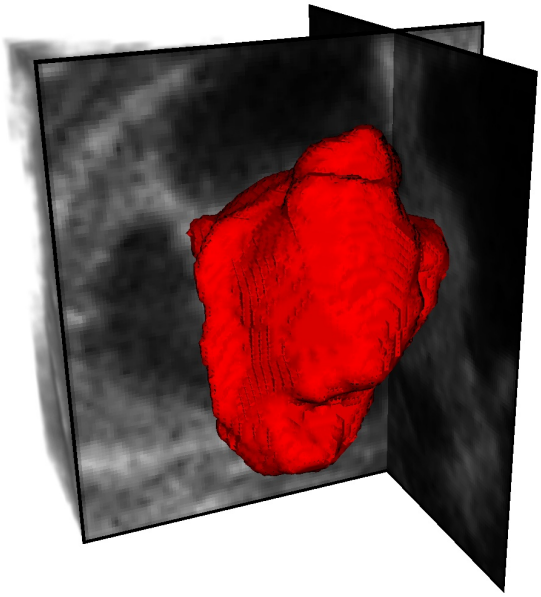
Due to the lack of edge information as well as the large amount of noise in the data, the geodesic level set has only partly succeeded in tracing the contour of the tumour. The result is partly oversegmented, as seen in figure 6.23 (c), as well as undersegmented in areas where the intensity of the tumour is larger than the overall.

The result is only useful as an indication of the scope of the tumour. Manual modifications would be very labour-intensive.

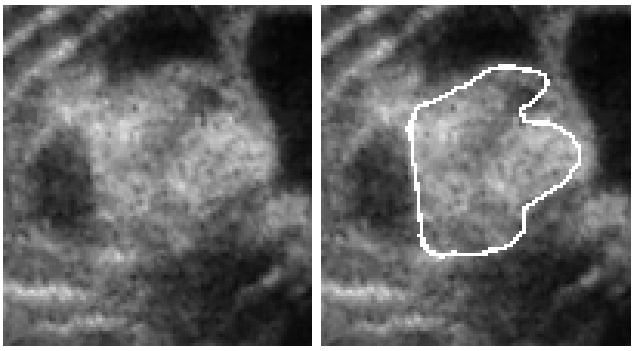
Comments from the expert panel may be seen in table 6.8. Expert 2 stated that the result was "slightly undersegmented. Regarding usability, see previous data set (N359 - MRI)).

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	2
Oversegmentation	1	1	2
Undersegmentation	2	3	3
Wrong edges traced	2	2	2
Details missing	2	2	3
Is the result usable?	yes	yes	yes
Manual adjustment required	-	3	3

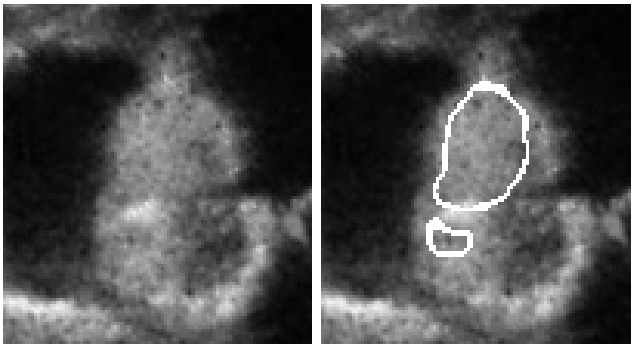
Table 6.8: Expert evaluation of segmentation result from dataset N359 - US (For a description of the evaluation criteria, see section 1.8)



(a)



(b) (c)



(d) (e)

Figure 6.23: Results of running GeodesicLevelSetFilter on dataset N359 - US **(a)** Volume representation of result **(b)** Original slice ($x = 59$) **(c)** Slice ($x = 59$) with result **(d)** Original slice ($x = 83$) **(e)** Slice ($x = 83$) with result

Dataset N360 - MRI

For results from the first MRI (T1) of the dataset N360, see figure 6.24. The master is MRI and the parameters are: seed = (45, 45, 55), initial distance = 10, σ = 1.0, α = -1.4, β = 3.0, propagation scaling = 1.0. Our reference computer used 8,05 seconds to preprocess and 0,75 seconds to run the level set segmentation.

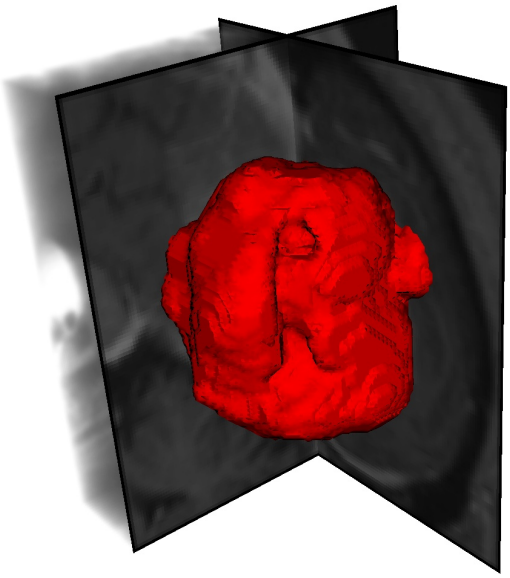
Although the intensity of the tumour is very similar to the surrounding tissue, the geodesic level set has succeeded in tracking the contour of the tumour. If the result is compared with the result from the fast marching level set result (see figure 6.11 (c)), we see that the "leaking problem" is not an issue for this filter. This means that the extra term in the geodesic level set filter, designed to handle diffuse edges and edges with gaps is performing well in this situation. Furthermore, the result is somewhat undersegmented (the hole seen in figure 6.24 (e)), this is due to the intensity deviation from the overall intensity of the tumour. The result also seems to include small details of the tumour well.

This result may very well be used in medical applications, due to the fact that the resulting boundary coincides with the tumour with high precision. Performing some small modifications of the result, that is to e.g. remove the hole seen in figure 6.24 (e) will yield an almost perfect result.

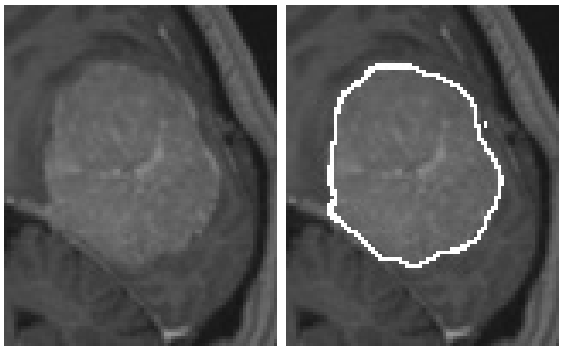
Comments from the expert panel may be seen in table 6.9. Expert 2 stated that "some normal tissue was included."

Criterion	Expert 1	Expert 2	Expert 3
Misplacement	1	1	1
Oversegmentation	1	2	1
Undersegmentation	2	2	2
Wrong edges traced	1	2	2
Details missing	1	3	2
Is the result usable?	yes	yes	yes
Manual adjustment required	-	3	1

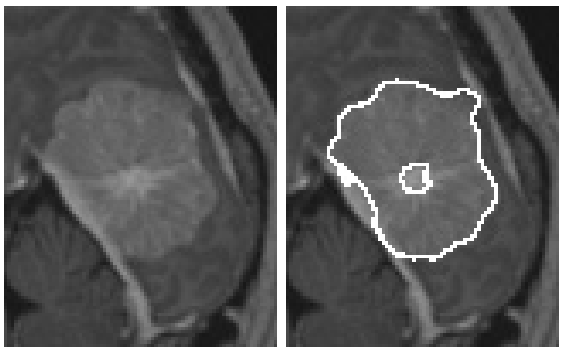
Table 6.9: Expert evaluation of segmentation result from dataset N360 - MRI (For a description of the evaluation criteria, see section 1.8)



(a)



(b) (c)



(d) (e)

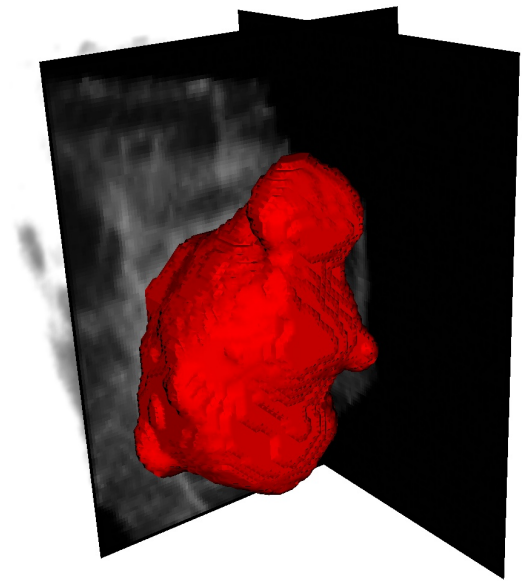
Figure 6.24: Results of running GeodesicLevelSetFilter on dataset N360 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 37$) **(c)** Slice ($x = 37$) with result **(d)** Original slice ($x = 29$) **(e)** Slice ($x = 29$) with result

Dataset N360 - US

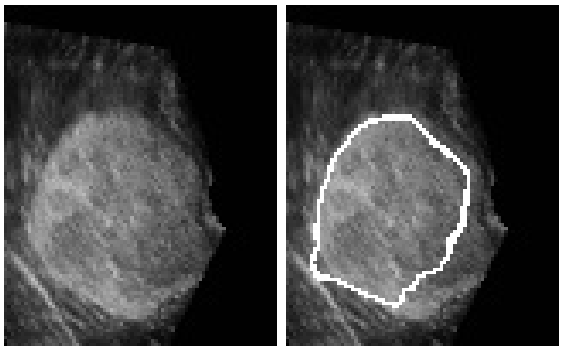
For results from the first US from the dataset N360, see figure 6.25. The master is MRI and the parameters are: seed = (45, 33, 55), initial distance = 5, σ = 2.0, α = -1.5, β = 2.0, propagation scaling = 1.0. Our reference computer used 7,92 seconds to preprocess and 3,38 seconds to run the level set segmentation.

The geodesic level set segmentation has failed to pick up large parts of the tumour contour and this is probably most due to the lack of edge information. The left side of the tumour, with a relatively strong edge as seen in figure 6.25 (c) and (e), has resulted in a fairly good edge detection. On the other hand, where the edge is weak it has resulted in both a very undersegmented result (slice c) and some oversegmented parts in the upper part of the tumour (see figure 6.25 (e)).

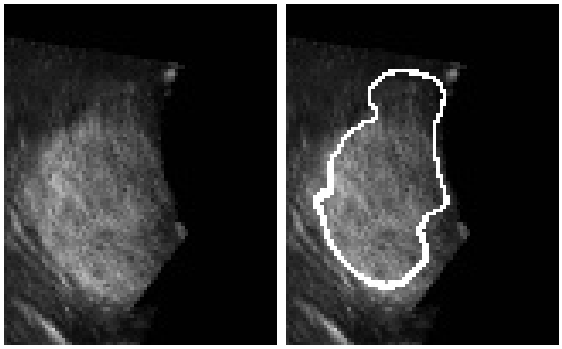
The result does not give a very good indication of the tumour scope, and is therefore not very useful in a medical context. Manual modifications would probably be very labour-intensive.



(a)



(b) (c)



(d) (e)

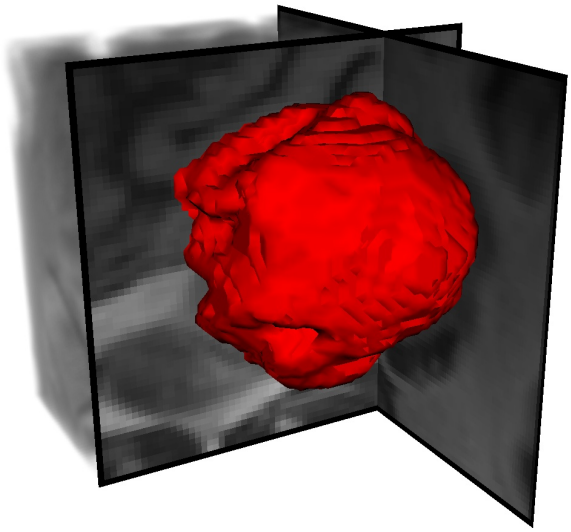
Figure 6.25: Results of running GeodesicLevelSetFilter on dataset N360 - US **(a)** Volume representation of result **(b)** Original slice ($x = 35$) **(c)** Slice ($x = 35$) with result **(d)** Original slice ($x = 51$) **(e)** Slice ($x = 51$) with result

Dataset N378 - MRI

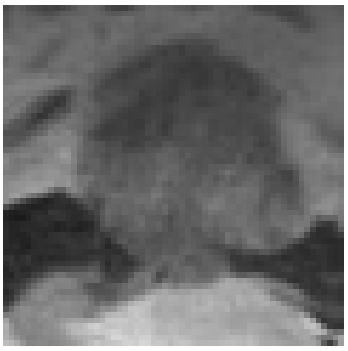
For results from the first MRI (T1) from the dataset N378, see figure 6.26. The master is MRI and the parameters are: seed = (30, 30, 30), initial distance = 5, σ = 1.5, α = -1.0, β = 2.5, propagation scaling = 100.0. Our reference computer used 3,41 seconds to preprocess and 9,53 seconds to run the level set segmentation.

On the overall, the geodesic level set filter has detected the boundary of the tumour well. The result is undersegmented in some parts, as can be seen in the lower parts of the tumour in figure 6.26 (c) as well as figure 6.26 (e). Where edge information is strong, as the transition between bright and dark tissue as seen in figure 6.26 (c), the result is very good. As seen in figure 6.26 (e), the boundary is oversegmented due to the similarity in intensity between the tumour and the surrounding tissue.

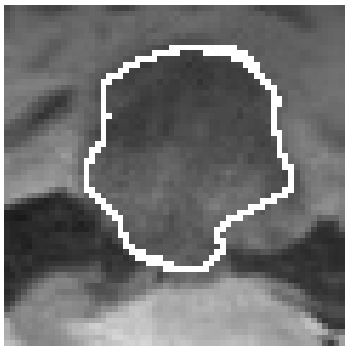
The result is useful as a guide to the overall shape and size of the tumour, but is not very useful in applications where high precision is required.



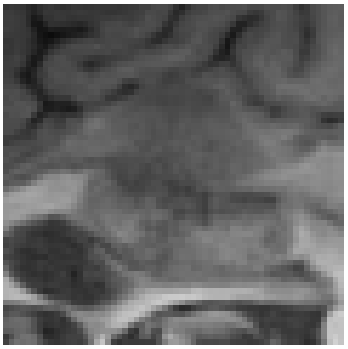
(a)



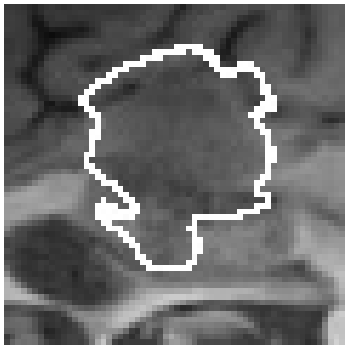
(b)



(c)



(d)



(e)

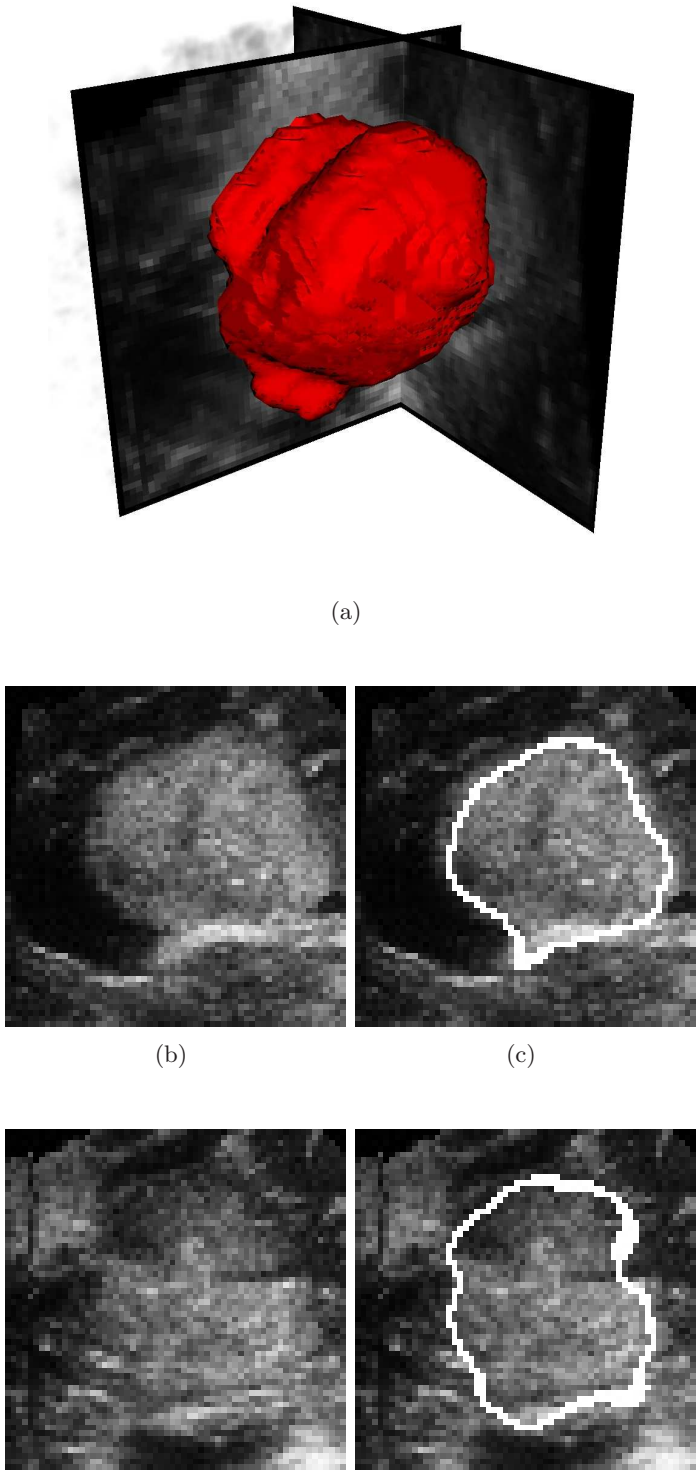
Figure 6.26: Results of running GeodesicLevelSetFilter on dataset N378 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 45$) **(c)** Slice ($x = 45$) with result **(d)** Original slice ($x = 34$) **(e)** Slice ($x = 34$) with result

Dataset N378 - US

For results from the first US from the dataset N378, see figure 6.27. The master is MRI and the parameters are: seed = (30, 30, 30), initial distance = 5, σ = 2.0, α = -2.0, β = 2.5, propagation scaling = 100.0. Our reference computer used 3,42 seconds to preprocess and 12,31 seconds to run the level set segmentation.

Although the original ultrasound data is corrupted by noise and the uniformity of the tumour intensity is relatively low, the geodesic level set filter has traced the contour very well on most parts of the tumour. Due to the initial smoothing the result is slightly undersegmented, this can be best seen in figure 6.27 (c). In some parts of the tumour that has been very corrupted by noise, the level set segmentation has also been able to trace the contour fairly well. Furthermore, if the result in figure 6.27 (c) is compared with the result of the fast marching level set (figure 6.14 (c)), we can see that the problem with the small areas detected outside the tumour has been removed with the use of geodesic level set.

The result gives a fairly good impression of the overall structure and scope of the tumour. Therefore it is definitively useful in medical applications, although not where high precision is required.



(a)

(b) (c)

(d) (e)

Figure 6.27: Results of running GeodesicLevelSetFilter on dataset N378 - US **(a)** Volume representation of result **(b)** Original slice ($x = 38$) **(c)** Slice ($x = 38$) with result **(d)** Original slice ($x = 32$) **(e)** Slice ($x = 32$) with result

6.4 Canny edge level set segmentation

Canny edge detection is a variant of edge detection that has proven to be very robust to white noise. The idea is to make the speed term as well as the advection term of the level set equation to depend on Canny edges in the input image. This variant of the level set method also requires the user to present an initial model, while the two previous methods we have described grow from seed points.

The Canny edge detector has three optimality criteria:

- *detection criterion* — no important edge in an image should be missed and the detector should not produce any false results
- *localization criterion* — meaning that the distance between the actual and detected position of an edge should be minimal
- *one response criterion* — there should be only one response per edge

6.4.1 Theoretical background

The algorithm of the Canny edge detector is given below:

1. Smooth the image with a Gaussian (with scale σ)
2. Estimate local edges
3. Locate the strongest part of these edges
4. Compute the edge magnitudes
5. Threshold the edges to eliminate false responses
6. Repeat steps 1-5 with increasing values of σ

Smoothing the image (step 1) is done by convolving the image with a Gaussian distribution. Local edges in direction $\mathbf{n}$ in an image can be found by convolving the image with G_n, the first derivative of a Gaussian in the normal direction $\mathbf{n}$:

$$G_n = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G. \tag{6.10}$$

The local edges for each pixel in the image (step 2) are then estimated using the following formulae:

$$\mathbf{n} = \frac{\nabla (G * f)}{|\nabla (G * f)|}. \tag{6.11}$$

At this point, an edge is located at the local maximum of the image f convolved with G_n:

$$\frac{\partial}{\partial \mathbf{n}} G_n * f = 0. \tag{6.12}$$

Substituting G_n from (6.10) into (6.12) results in:

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0, \tag{6.13}$$

which gives the strongest part of an edge (step 3), that is the local maxima of the edge in the direction perpendicular to the edge itself. This technique, named *non-maximal suppression*, works by checking the edge information at each pixel. If the edge strength at one or both of the two pixels' neighbours (in the direction perpendicular to the edge) is stronger than the first one, the pixel is marked for deletion. See figure 6.28. When the entire image is checked, the edge information at marked pixels is set to 0.

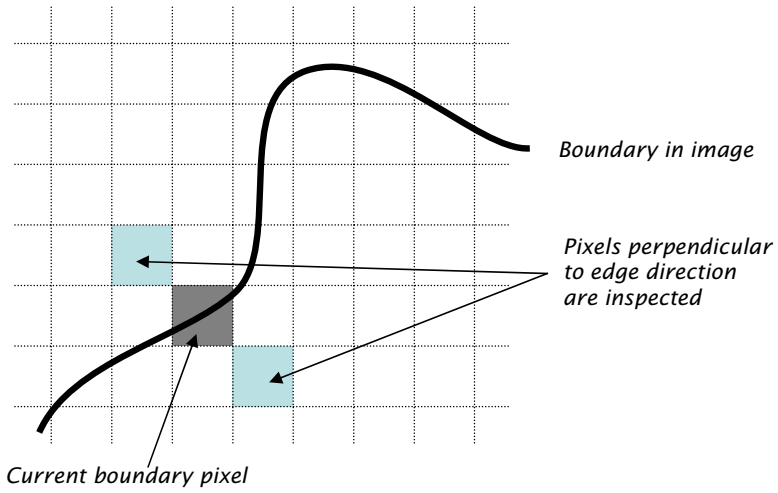


Figure 6.28: Non-maximal suppression [28].

Next, the magnitudes of the edges (step 4) are computed using the following equation:

$$|G_n * f| = |\nabla(G * f)|, \tag{6.14}$$

and finally, each edge is passed through a simple threshold filter. This reduces the sensitivity and makes the algorithm more robust by eliminating false responses (step 5). These steps are then repeated at increasing scale levels (with different σ) [28].

The Canny edges returned by this algorithm are then used to guide the level set equation. The idea is to force the expanding level set surface to move towards Canny edges in the image. This is done by making both the advection term as well as the propagation term dependent to these edges. First, the advection term is determined by minimizing the squared distance transform from the Canny edges:

$$min \int D^2 \Rightarrow D \nabla D, \tag{6.15}$$

where D is a distance transform, containing information about the distance to an object/objects of interest in an image. In our case the objects of interest would typically

be Canny edges. Finally, the propagation term is simply set to be D. This will make the surface of the level set expand towards Canny edges in the input image [16].

6.4.2 Implementation

An overview of the different steps in our implementation of the canny edge level set filter can be seen in figure 6.29. The input image is first passed through the itk::GradientAnisotropicDiffusionImageFilter, that smoothes the image while preserving edge information. An initial model is used as a feature image. The model may either be a binary image with some surface that already has been segmented, or it may be a sphere with size and position given by the user. Internally in the Canny edge filter the squared distance map (6.15) is calculated using the itk::DanielssonDistanceMapImageFilter. The output from the itk::CannySegmentationLevelSetImageFilter is then passed through a simple thresholding filter, the itk::BinaryThresholdImageFilter. The resulting image is a binary image, representing the segmented surface.

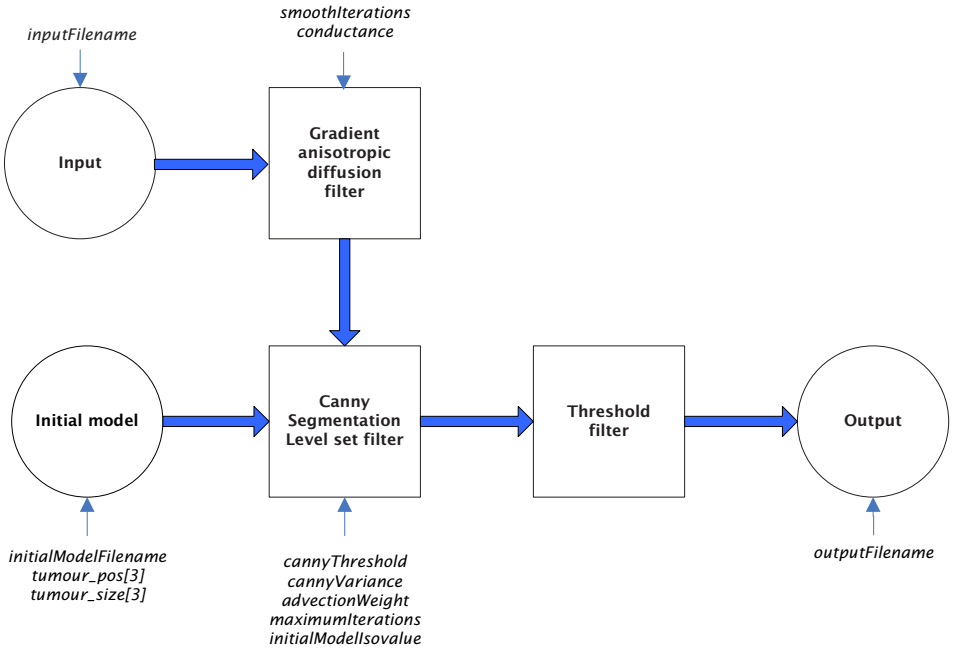


Figure 6.29: Flowchart of the CannyEdgeLevelSet filter.

An overview of the different methods in the CannyEdgeLevelSetFilter class may be seen in figure 6.30. The most important parameters and a description of these may be seen below:

- Conductance: Controls the strength of the smoothing of the input image
- SmoothIterations: Controls the number of smoothing iterations the input image is run through
- CannyThreshold: Only Canny edges above this value is used

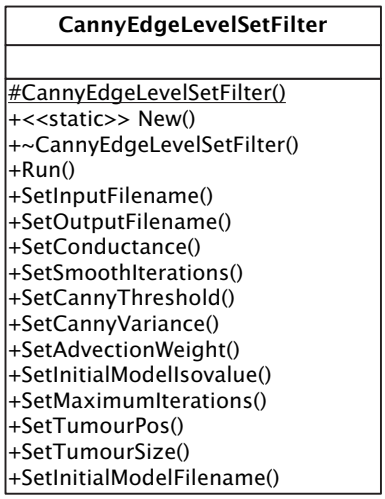


Figure 6.30: The CannyEdgeLevelSetFilter class.

- CannyVariance: Determines how much the input to the Canny Edge filter should be smoothed
- AdvectionWeight: Sets the weight of the advection term in the level set equation
- InitialModelIsovalue: The feature image, that is the initial model, is thresholded with this value
- MaximumIterations: The maximum number of iterations this filter will run. The filter will either stop after this maximum or when it has reached a default error rate
- TumourPos and TumourSize: If no initial model is given by its filename, the user defines the position and size of a sphere that is used as an alternative, initial model
- InitialModelFilename: The feature input to the Canny edge filter is specified with this parameter. If no filename is given the filter will assume that the user has specified the size and position of the tumour instead
- InputFilename and OutputFilename: These parameters set the filenames to read from and write to

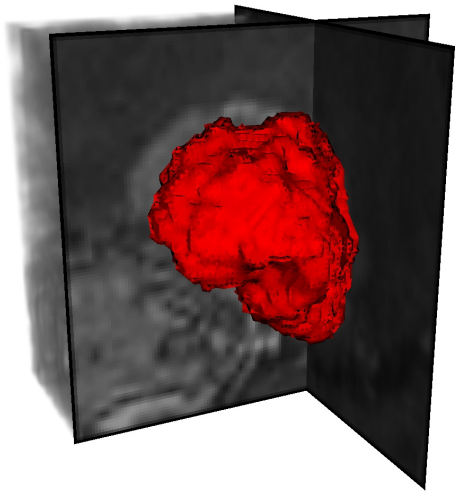
6.4.3 Results

The best results from the use of the fast marching and geodesic level set filters was used as input for the canny edge level set filter. The goal was not to improve the overall segmentation result, but to see if it was possible to improve the detection of small details of the tumour.

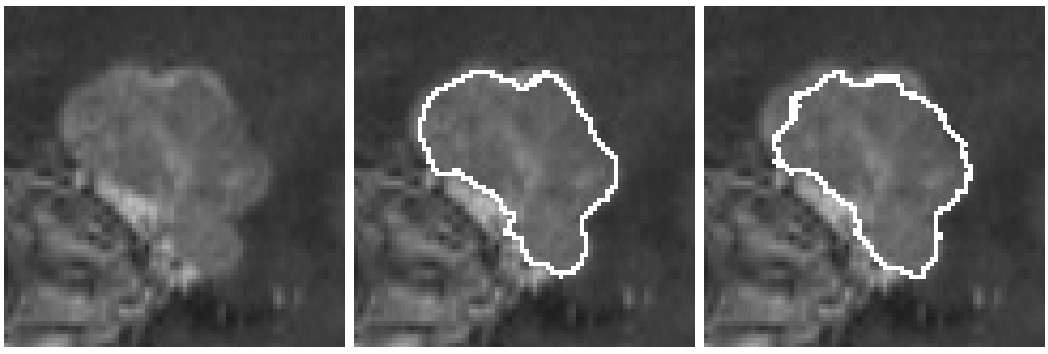
Dataset N241 - MRI

For results from the first MRI from dataset N241, see figure 6.31. The master is MRI and the parameters are: conductance = 1.0, smoothing iterations = 5, canny threshold = 15, canny variance = 1, advection weight = 10, isovalue = 0.5, max iterations = 100. Our reference computer used 5,72 seconds to preprocess and 7,89 seconds to run the level set segmentation.

The initial model (figure 6.31 (c)) is from running the geodesic level set. As seen in figure 6.31 (d), the canny edge level set filter has not managed to improve the result at all, the result is in fact worse than the original. The reason for the bad edge detection is probably due to the lack of strong edge information in the original MRI data (figure 6.31 (b)).



(a)



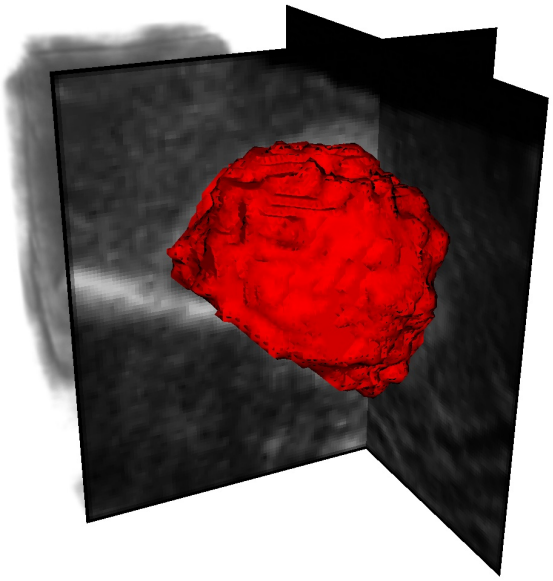
(b) (c) (d)

Figure 6.31: Results of running CannyEdgeLevelSetFilter on dataset N241 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 42$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

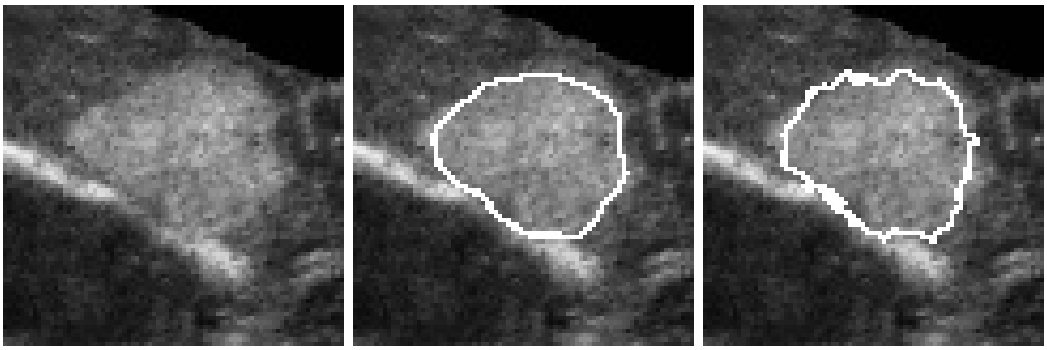
Dataset N241 - US

For results from the first US of the dataset 241, see figure 6.32. The master is MRI and the parameters are: conductance = 3.0, smoothing iterations = 5, canny threshold = 7, canny variance = 1, advection weight = 10, isovalue = 0.5, max iterations = 100. Our reference computer used 5,83 seconds to preprocess and 7,41 seconds to run the level set segmentation.

The initial model is from the geodesic level set filter, see figure 6.19. We see from figure 6.32 (d) that the canny edge level set filter tries to adjust to the local edges, but the yielded result is not better than the original. This is probably due to many local edges in the image.



(a)



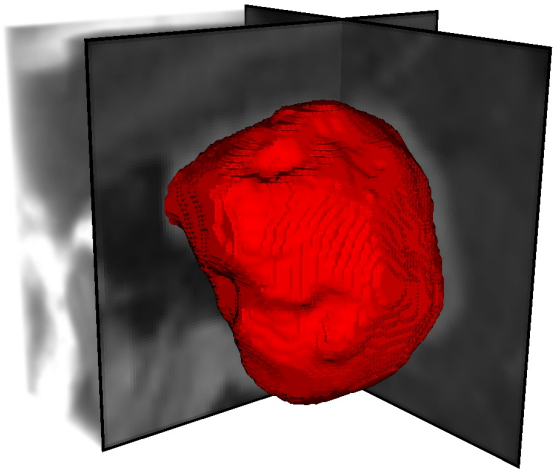
(b) (c) (d)

Figure 6.32: Results of running CannyEdgeLevelSetFilter on dataset N241 - US **(a)** Volume representation of result **(b)** Original slice ($x = 35$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

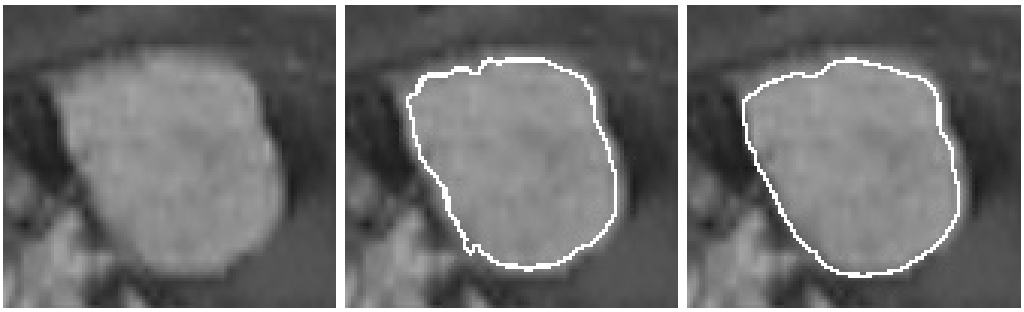
Dataset N351 - MRI

For results from the first MRI (T1) of the dataset N351, see figure 6.33. The master is US and the parameters are: conductance = 1.0, smoothing iterations = 5, canny threshold = 40, canny variance = 5, advection weight = 10, isovalue = 0.5, max iterations = 10. Our reference computer used 12,78 seconds to preprocess and 8,42 seconds to run the level set segmentation.

The initial model is from the geodesic level set filter seen in figure 6.20. Here, the preprocessing by the canny edge level set filter has produced result that is slightly more rounded off, but the result has not improved in any significant degree.



(a)



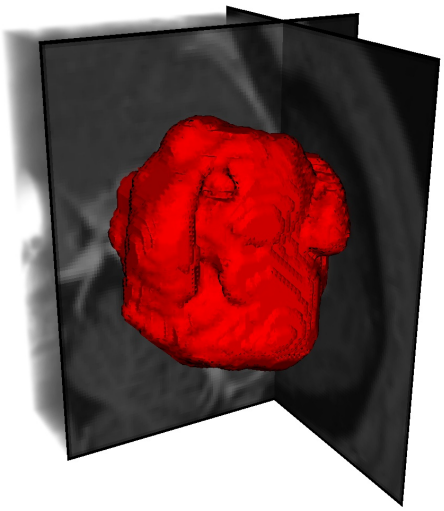
(b) (c) (d)

Figure 6.33: Results of running CannyEdgeLevelSetFilter on dataset N351 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 52$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

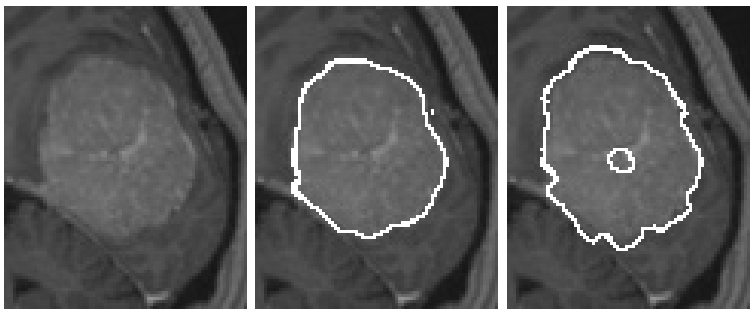
Dataset N360 - MRI

For results from the first MRI (T1) from dataset N360, see figure 6.34. The master is MRI and the parameters used are: conductance = 1.0, smoothing iterations = 5, canny threshold = 7, canny variance = 5, advection weight = 10, isovalue = 0.5, max iterations = 100. Our reference computer used 7,05 seconds to preprocess and 11,64 seconds to run the level set segmentation.

The initial model is the output of the geodesic level set filter seen in figure 6.24. In this case, it seems like the canny edge has partly succeeded in producing a more accurate contour. For example, as seen in the lower left part of the tumour, the canny edge filter has traced the two sharp corners better than the geodesic level set filter. On the overall the result has become worse, for instance it introduce a hole in the center of the tumour in figure 6.34 (d).



(a)



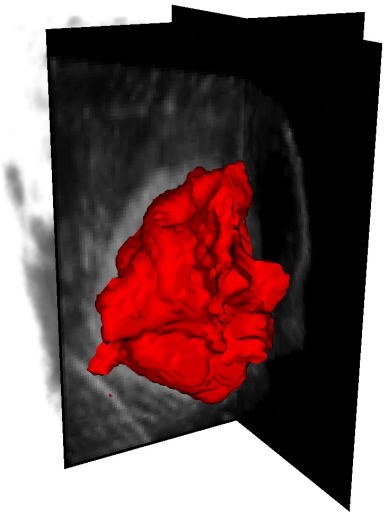
(b) (c) (d)

Figure 6.34: Results of running CannyEdgeLevelSetFilter on dataset N360 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 37$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

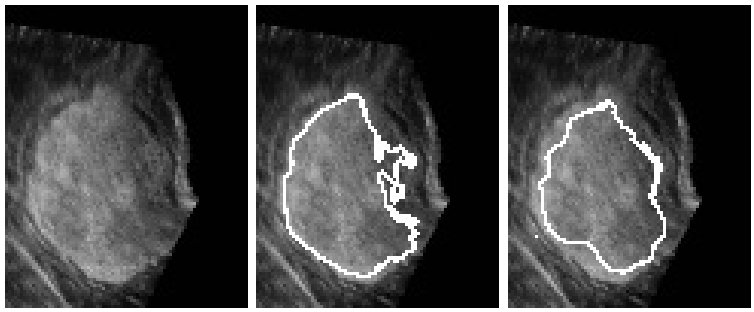
Dataset N360 - US

For results from the first US of the dataset N360, see figure 6.35. The master is MRI and the parameters are: conductance = 2.0, smoothing iterations = 5, canny threshold = 7, canny variance = 5, advection weight = 10, isovalue = 0.5, max iterations = 100. Our reference computer used 6,98 seconds to preprocess and 9,88 seconds to run the level set segmentation.

The initial model was taken from the output of the fast marching level set filter, see figure 6.12. After preprocessing with the canny edge filter it seems like the filter has managed to improve some parts, like the smoothing out of the rough edge on the left side of the tumour (see figure 6.35 (c)). On the other hand the canny edge filter has made other parts of the result worse as seen in the lower left part of the tumour in figure 6.35 (d), where the correctly traced contour from figure 6.35 (c) is moved away from the edge.



(a)



(b) (c) (d)

Figure 6.35: Results of running CannyEdgeLevelSetFilter on dataset N360 - US **(a)** Volume representation of result **(b)** Original slice ($x = 39$) **(c)** Slice with initial model (fast marching level set) **(d)** Slice with result

6.5 Laplacian level set segmentation

This variant of level set segmentation uses edges detected by Laplacian filtering of the input image. The idea is to attract the expanding level set surface to these edges. As with the canny edge level set filter, this filter also requires an initial model to be provided by the user. This may be done either as giving an already segmented image as input, or specify the size and position of an initial sphere.

6.5.1 Theoretical background

Laplacian edge detection tries to locate zero-crossings of the second derivative of edges in an image. This is accomplished by convolving the input image with the Laplacian operator. For 3D images, the operator is defined as:

$$\nabla^2 g(x,y,z) = \frac{\partial^2 g(x,y,z)}{\partial x^2} + \frac{\partial^2 g(x,y,z)}{\partial y^2} + + \frac{\partial^2 g(x,y,z)}{\partial z^2} \tag{6.16}$$

The operator has the same properties in all directions and is therefore invariant to rotation in the image. Many discrete approximations of the Laplacian exists, the following matrix is an example for 2 dimensions:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{6.17}$$

6.5.2 Implementation

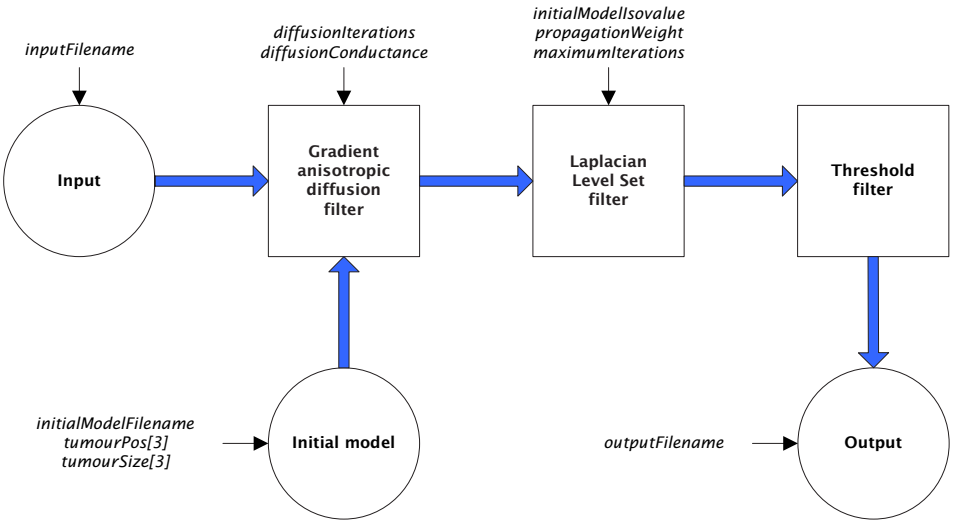


Figure 6.36: Flowchart of the different steps in the LaplacianLevelSetFilter.

Figure 6.36 shows an overview of the different steps in the LaplacianLevelSetFilter. First, the input image is passed through the itk::GradientAnisotropicDiffusionImageFilter,

that smoothes the image and preserves edge information. The initial model that is required by the level set filter, may either be provided as a file by the user, or the user may specify the size and position. If this is done, our filter will construct a sphere and pass it the Level set filter. Next, internally in the itk::LaplacianSegmentationLevelSetImageFilter, edge information is extracted from the input using the itk::LaplacianImageFilter. The output is passed through a thresholding filter, the itk::BinaryThresholdImageFilter that outputs the final surface.

For a class diagram of the LaplacianLevelSetFilter, see figure 6.37. The different parameters along with a brief description of these may be found below:

- DiffusionIterations: The number of iterations the diffusion filter should process the input image
- DiffusionConductance: The strength of the diffusion filtering
- PropagationWeight: Controls how much the initial surface is allowed to expand during the segmentation process
- InitialModelIsovalue: Specifies the thresholded value used on the initial model
- MaximumIterations: How many iterations the segmentation process is allowed to run
- TumourPos and TumourSize: If specified, use these dimensions and position to construct a sphere and use this as an initial model
- InitialModelFilename: If specified, use this file as an initial model
- InputFileName and OutputFileName: Specifies the filenames to read and write from, respectively

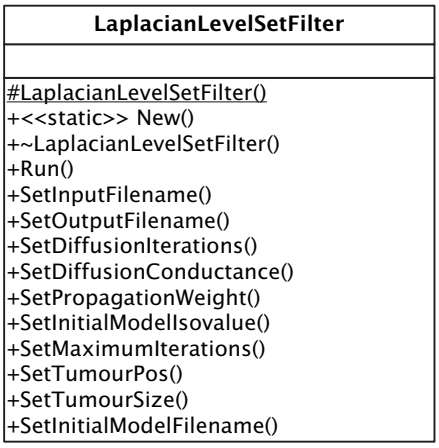


Figure 6.37: The LaplacianLevelSetFilter class.

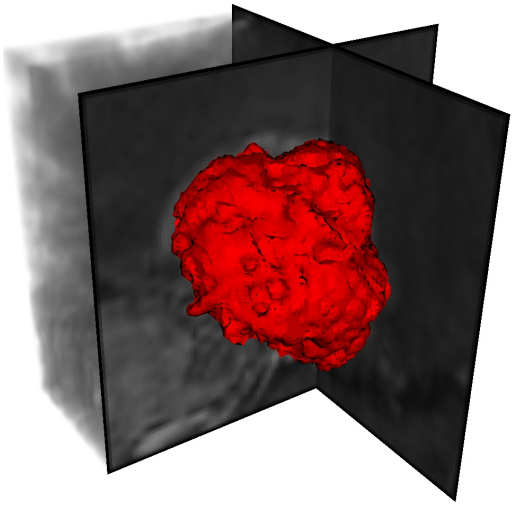
6.5.3 Results

As with the canny edge level set filter, the best results from the fast marching and geodesic level set filters are used as input to the laplacian level set filter. The goal is to improve the result, particularly with focus on detecting small details in the tumour contour.

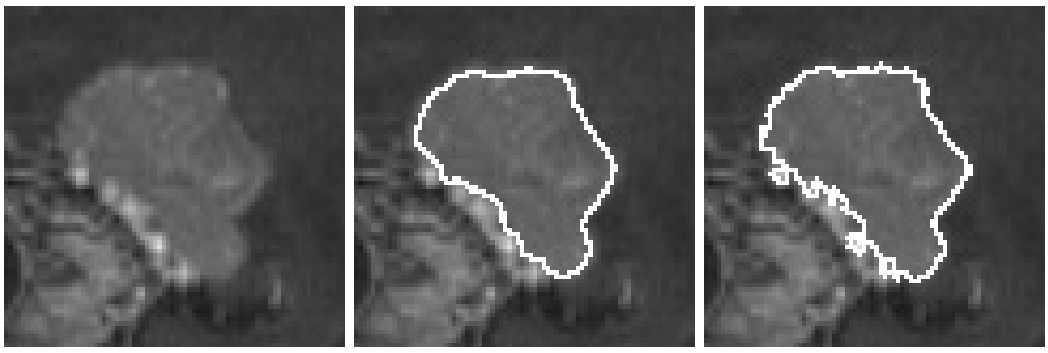
Dataset N241 - MRI

For results from the first MRI (T1) from the dataset N241, see figure 6.38. The master is MRI and the parameters are: diffusion iterations = 5, diffusion conductance = 2, propagation weight = 100, isovalue = 0.5, max iterations = 50. Our reference computer used 5,69 seconds to preprocess and 2,73 seconds to run the level set segmentation.

The initial model used here is taken from the geodesic level set filter, see figure 6.18. The laplacian filter has not managed to improve the segmentation result significantly. It has however managed to pick up a lot of local edges in the lower left part of the tumour, as seen in figure 6.38 (d). This is probably due to the fact that the lacplacian filter has a tendency to get stuck in local edge information. On the other hand, it seems like the result has improved slightly on the right part of the tumour.



(a)



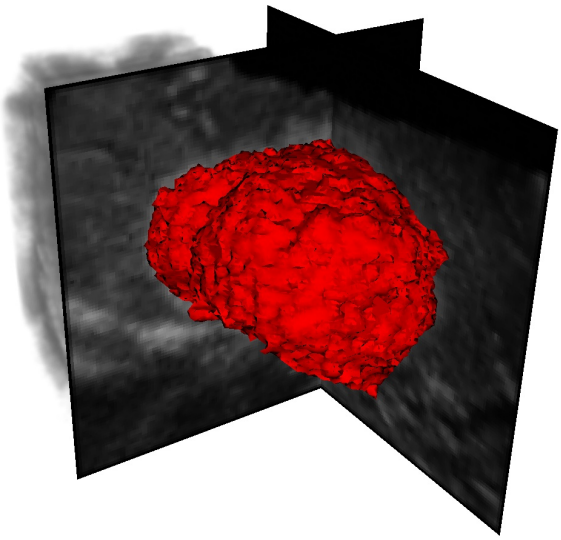
(b) (c) (d)

Figure 6.38: Results of running LaplacianLevelSetFilter on dataset N241 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 45$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

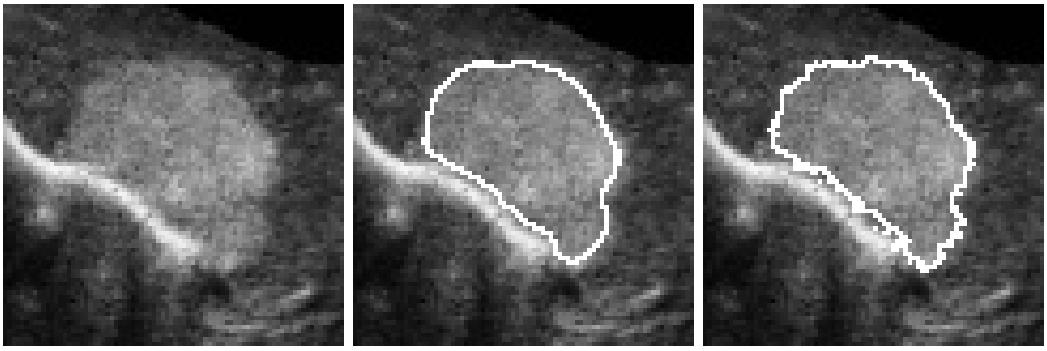
Dataset N241 - US

For results from the first US from the dataset N241, see figure 6.39. The master is MRI and the parameters are: diffusion iterations = 5, diffusion conductance = 3, propagation weight = 100, isovalue = 0.5, max iterations = 50. Our reference computer used 5,67 seconds to preprocess and 3,08 seconds to run the level set segmentation.

The initial model used here is from the geodesic levels set filter (figure 6.19). Because the edge information from the original ultrasound data is heavily corrupted by noise, it seems like the laplace filter get stuck in local edges. The result has not been improved.



(a)



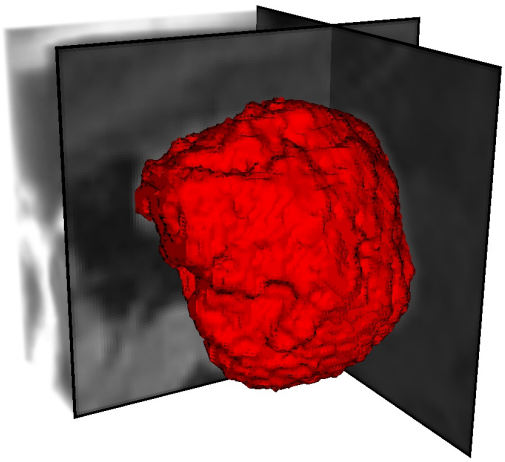
(b) (c) (d)

Figure 6.39: Results of running LaplacianLevelSetFilter on dataset N241 - US **(a)** Volume representation of result **(b)** Original slice ($x = 48$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

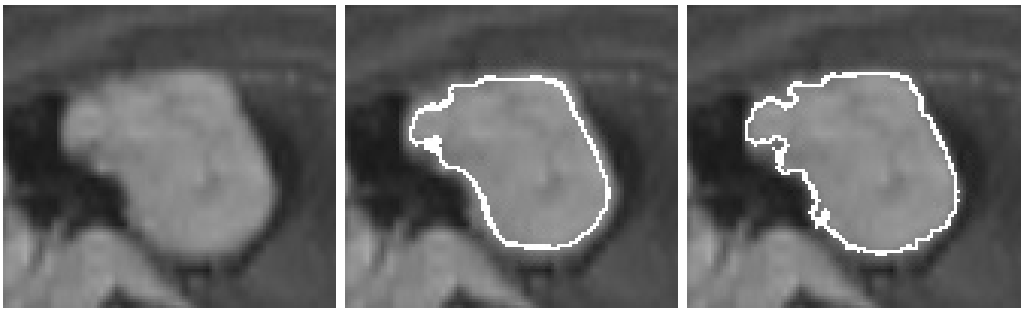
Dataset N351 - MRI

For results from the first MRI (T1) from the dataset N351, see figure 6.40. The master is US and the parameters are: diffusion iterations = 5, diffusion conductance = 2, propagation weight = 100, isovalue = 0.5, max iterations = 50. Our reference computer used 12,74 seconds to preprocess and 6,91 seconds to run the level set segmentation.

The initial model used here is from the geodesic levels set filter (figure 6.20). The laplacian filter has managed to improve the original result slightly. As can be seen in figure 6.40 (d), the small details in the upper part of the tumour has been picked up by the filter. The reason is probably that the edge information is very strong and not too influenced by noise as in many other cases.



(a)



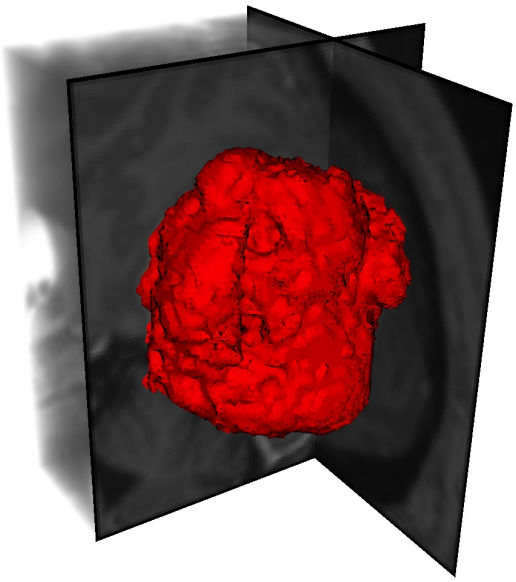
(b) (c) (d)

Figure 6.40: Results of running LaplacianLevelSetFilter on dataset N351 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 42$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

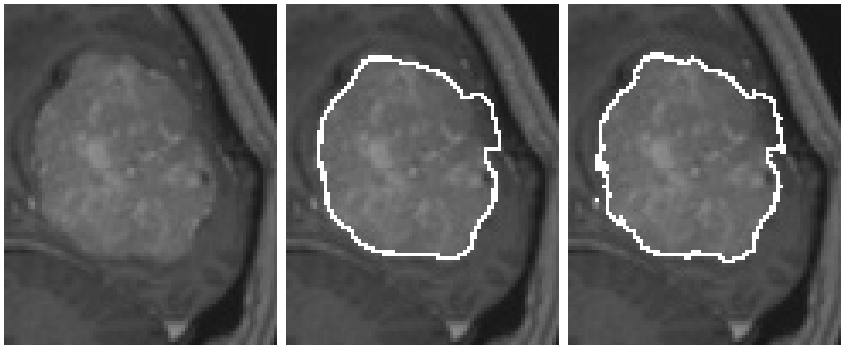
Dataset N360 - MRI

For results from the first MRI (T1) from the dataset N360, see figure 6.41. The master is MRI and the parameters are: diffusion iterations = 5, diffusion conductance = 2, propagation weight = 100, isovalue = 0.5, max iterations = 50. Our reference computer used 7,02 seconds to preprocess and 3,75 seconds to run the level set segmentation.

The initial model used here is from the geodesic levels set filter (figure 6.24). The laplacian filter has managed to improve the tracing of small parts of the contour such as the one seen on the upper part as well as the one in the lower part of the tumour in figure 6.41 (b). The filter has not managed to improve the false detected part in the upper, right corner of the tumour in figure 6.41 (b).



(a)



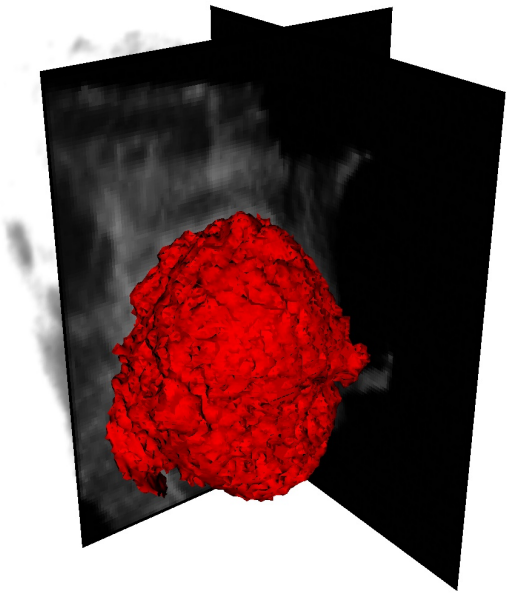
(b) (c) (d)

Figure 6.41: Results of running LaplacianLevelSetFilter on dataset N360 - MRI **(a)** Volume representation of result **(b)** Original slice ($x = 46$) **(c)** Slice with initial model (geodesic level set) **(d)** Slice with result

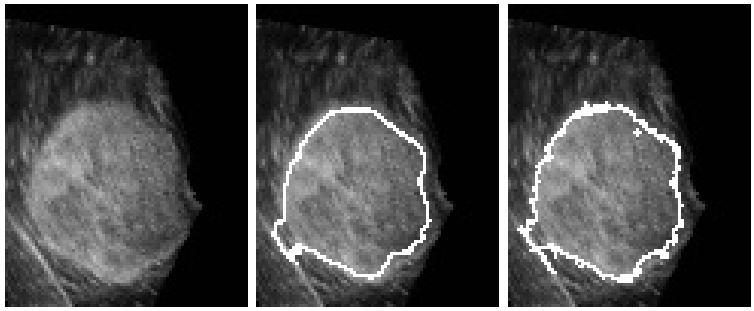
Dataset N360 - US

For results from the first US of the dataset N360, see figure 6.42. The master is MRI and the parameters are: diffusion iterations = 5, diffusion conductance = 3, propagation weight = 100, isovalue = 0.5, max iterations = 50. Our reference computer used 6,89 seconds to preprocess and 3,91 seconds to run the level set segmentation.

The initial model used here is from the geodesic levels set filter (figure 6.12). As the case was with the laplace result seen in figure 6.39, the filter has got stuck in small edges in the ultrasound image and does not produce a better result.



(a)



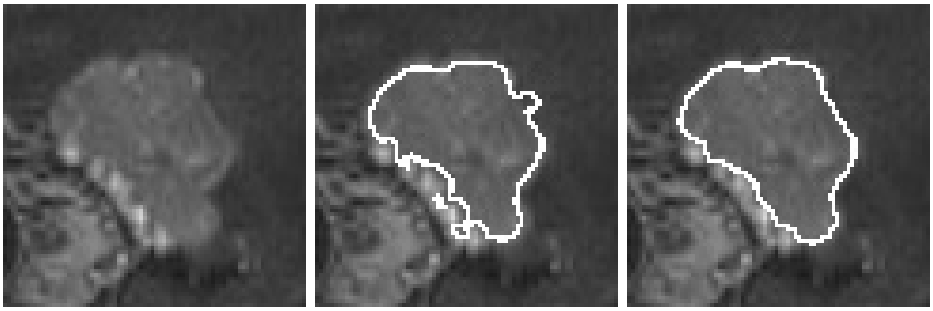
(b) (c) (d)

Figure 6.42: Results of running LaplacianLevelSetFilter on dataset N360 - US **(a)** Volume representation of result **(b)** Original slice ($x = 34$) **(c)** Slice with initial model (fast marching level set) **(d)** Slice with result

6.6 Comparison of results

In this section we will provide a short comparison of results from the FastMarchingLevelSetFilter and the GeodesicLevelSetFilter. This section is included in order to better understand the difference of the two segmentation filters.

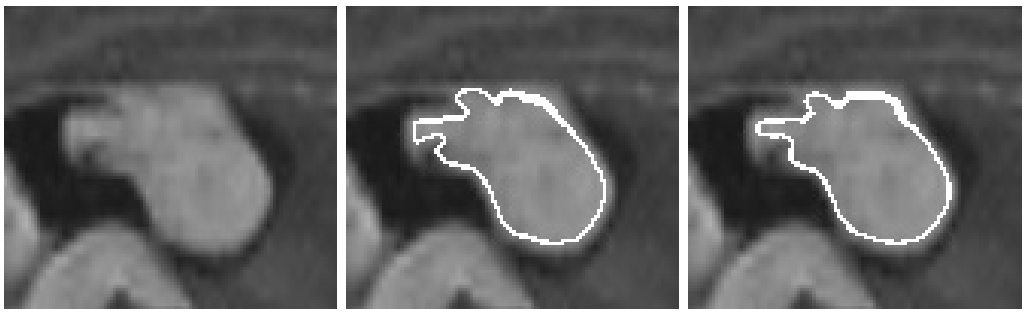
In dataset N241 - MRI, see figure 6.43(a), we see that the intensity distribution of the tumour lie close to the intensity distribution of the surrounding tissue. In such cases the FastMarchingLevelSetFilter, see figure 6.43(b), tends to "leak", producing false responses. The GeodesicLevelSetFilter on the other hand, deals with this problem very well and would be the filter to prefer in such cases, see figure 6.43(c).



(a) (b) (c)

Figure 6.43: Comparison of results from dataset N241 - MRI **(a)** Original slice ($x = 46$) **(b)** Result from the FastMarchingLevelSetFilter **(c)** Result from the GeodesicLevelSetFilter.

In dataset N351 - MRI, see figure 6.44(a), we see that the intensity distribution of the tumour is relatively uniform. In such cases the FastMarchingLevelSetFilter, see figure 6.44(b), as well as the GeodesicLevelSetFilter produce good results, see figure 6.44(c).

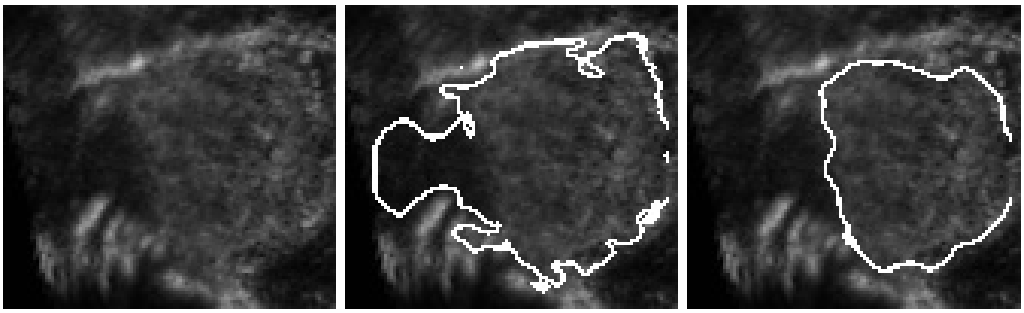


(a) (b) (c)

Figure 6.44: Comparison of results from dataset N351 - MRI **(a)** Original slice ($x = 37$) **(b)** Result from the FastMarchingLevelSetFilter **(c)** Result from the GeodesicLevelSetFilter.

In dataset N351 - US, see figure 6.45(a), we see that the tumour has a non-uniform intensity distribution. In such cases the FastMarchingLevelSetFilter (see figure 6.45(b))

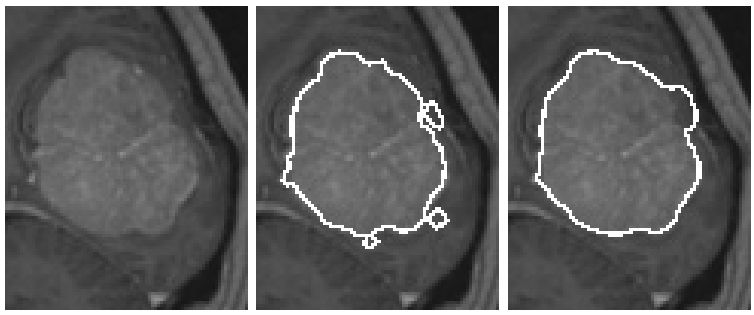
faces great problems due to the missing edge information. This is better handled by the GeodesicLevelSetFilter (see figure 6.45(c)), that produce much better results in such cases.



(a) (b) (c)

Figure 6.45: Comparison of results from dataset N351 - US **(a)** Original slice ($x = 53$) **(b)** Result from the FastMarchingLevelSetFilter **(c)** Result from the GeodesicLevelSetFilter.

In dataset N360 - MR, see figure 6.46(a), the tumour has a relatively diffuse contour. Again we see that the FastMarchingLevelSetFilter (see figure 6.46(b)) produces small, false responses due to the lack of edge information. And again, the GeodesicLevelSetFilter (see figure 6.46(c)), manages to extract a better contour of the tumour.



(a) (b) (c)

Figure 6.46: Comparison of results from dataset N360 - US **(a)** Original slice ($x = 42$) **(b)** Result from the FastMarchingLevelSetFilter **(c)** Result from the GeodesicLevelSetFilter.

6.7 Discussion

The image segmentation results presented in this chapter have proven level set methods to be very useful in the context of medical images. In most cases where the image data is fairly good such as the case is with most MRI data, the level set methods have yielded good results. Furthermore, even with image data heavily corrupted by noise, such as the

case often is with ultrasound data, level set methods have managed to produce relatively good results.

The fast marching level set filter (section 6.2) produced good results from most of the MRI data, and from some of the ultrasound data. Although the filter often managed to trace the overall contour of the tumour fairly well, it often had problems when there was little or weak edges. In the geodesic level set filter (section 6.3) this "leaking" problem has to a great extent been solved with the introduction of an extra term. This term was designed to particularly take care of situations where edge information is missing in the image, and results from the use of this filter also showed an improvement in such situations.

The best results from the fast marching as well as the geodesic level set filter was then used as input to the canny edge (section 6.4) and laplacian (section 6.5) level set filter. The goal was to see if the results could be improved, with focus on the tracing of small details. From the results, we see that little or no improvement was achieved, in fact in most cases the results turned out worse than the original ones.

The number of free parameters is relatively the same in all level methods investigated. Adjusting the parameters was mostly done by trying and failing to see which values yielded the best results. Although we tried to find robust parameter values that could be used in several situations, it turned out this was hard to achieve due to high variance in the image characteristics.

The different level set filters were run on a Intel Pentium 4 – 3.0 GHz computer with 768 MB of RAM. The operation system was Windows XP Professional with Service Pack 2. As seen from the description of the results, a large part of the computation time was related to preprocessing. The first filter investigated, the fast marching level set filter, had a computation time of typical 0,5 - 2,0 seconds. This is due to the fast marching implementation that doesn't compute values iteratively, but only runs once. The computation time needed for the geodesic level set filter varies more, with a maximum at 41,55 seconds. This is due to a more complex handling of image information to compensate for missing or weak edge information. Furthermore, the computation time needed by the canny edge level set filter lies around 10 seconds. The number of iterations in adjustable by the user and hence the computation time will depend linearly on this value. Finally, the laplacian level set filter used 3,0 - 7,0 seconds to refine the segmentation result. As with the case is with the canny edge filter, the computation time here is adjustable by the user.

Chapter 7

Visualization

In this chapter we will give a brief overview of the visualization software we developed as part of this thesis. First we will describe the basic visualizer, followed by a description of a more sophisticated visualizer we have partly implemented.

Early in our work with this thesis we soon discovered the need for some software capable of visualization segmentation results. Although we found many commercial as well as free software packages, including the CustusX software developed by SINTEF, we was not find able to find any software that suited all of our needs. Either the software were too complex, or it did not have the all the functionality we needed. Based on this experience, we decided to develop a simple visualization tool on our own.

7.1 Basic visualizer

This section describes the first, basic version of the Visualizer tool. There are three main requirements to the visualization software: First, it must be capable of 3D volume visualization of MRI as well as ultrasound data. Next, it must be able to slice through the volume in both x-, y- and z- direction. Finally, it must be able to display the result from the segmentation overlayed on the volume data. In addition, all these functions should have a simple user interface.

7.1.1 Implementation

The basic visualizer is implemented in C++ using the open-source visualization toolkit VTK, see section 2.3.2. The reason for choosing VTK was that it seemed to cover all the needs we had, and it is also a very popular, stable and well-documented toolkit for use with scientific and medical visualization.

7.1.2 User interface

A screenshot of the visualizer may be seen in figure 7.1. The user may interact with the visualizer with a combination of using the mouse and the keyboard. The mouse controls the rotation, zooming and placement of the camera that the user sees through. By pressing down the left mouse button and moving the mouse, the user may rotate the camera in any direction. Pressing the middle mouse button down and moving the mouse causes the camera to pan horizontally or vertically. Finally, by pressing the right

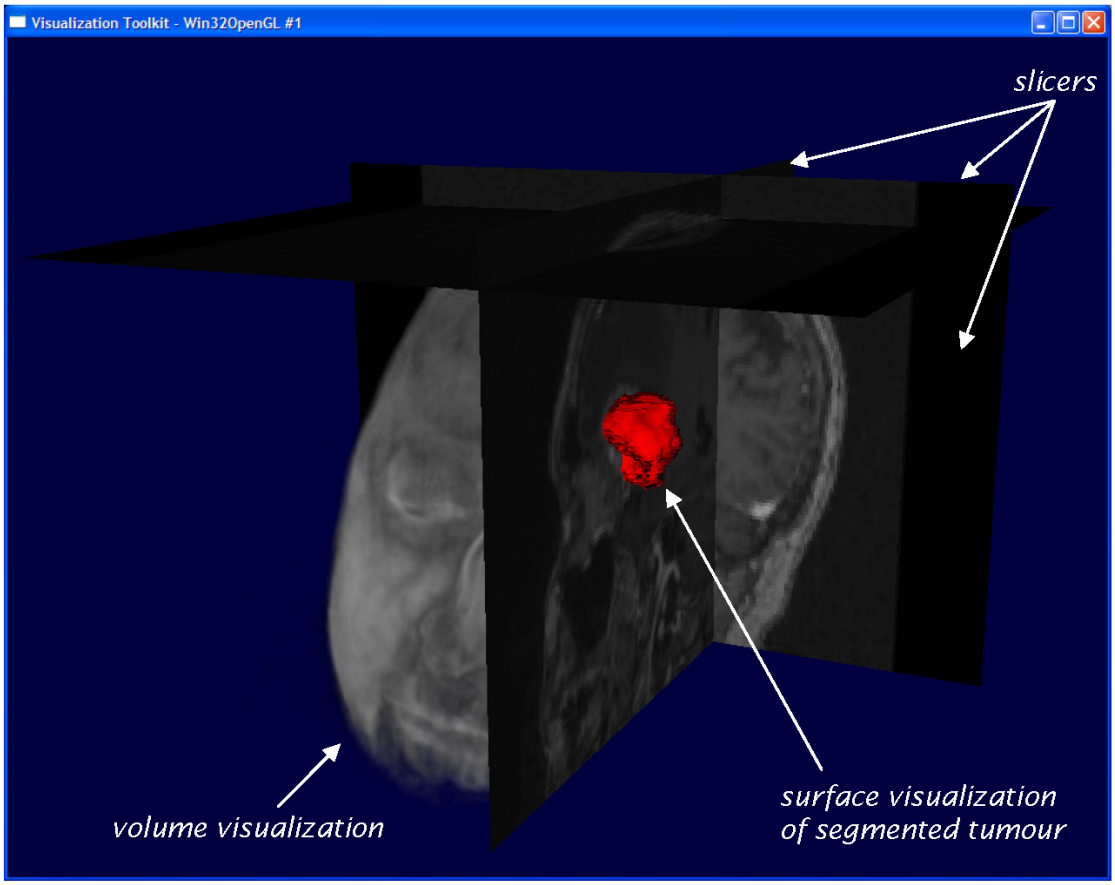


Figure 7.1: Screenshot from the basic visualizer application.

mouse button down the user may zoom in and out. The user may move the slicers through a combination of mouse and keyboard. Pressing either one of the keys "1", "2", or "3" in combination with moving the mouse will cause the slicer in the x-, y- or z- direction to move, respectively. Pressing "4" and moving the mouse will alter the opacity of the volume visualization, while "5" will alter the opacity of the visualization of the segmented result, that is the red surface in figure 7.1. Finally, it is also possible to overlay a 2D version of the segmented result on the 2D slicers. The opacity of this overlay may be altered with a combination of the "6" key and moving the mouse. A screenshot showing this feature may be seen in figure 7.2.

For convenience when documenting segmentation results, the user may choose to output a screenshot using the "x" key. Pressing the "shift" key in combination with one of the keys "1", "2", "3", outputs the content of the slicer in the x-, y- or z- direction, respectively.

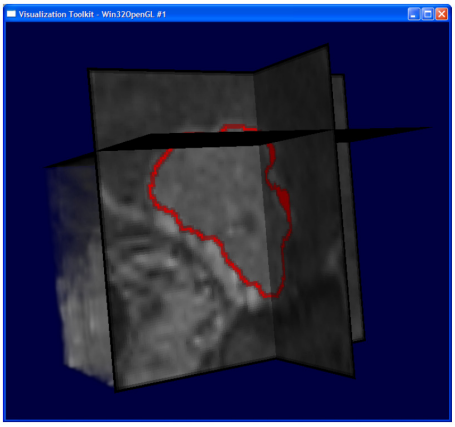


Figure 7.2: Screenshot with segmentation result overlayed on slicers.

7.2 Advanced visualizer

This section describes a more advanced visualizer tool. This version was never fully implemented, but we will give a brief description of the functionality the advanced visualizer should have.

In addition to the functionality of the basic visualizer, the advanced visualizer should offer:

- the possibility to access and execute the segmentation filters directly from the GUI of the Advanced Visualizer.
- the possibility to adjust all the parameter values needed to execute the particularly filter directly from the GUI of the Advanced Visualizer.
- the possibility to interactively set initial seed point(s) or an initial contour for the segmentation algorithms if this is required by the filter.
- the possibility to generate a preview of the result at a lower resolution than the original data set.
- the possibility to directly manipulate the evolving surfaces by dragging them towards the tumour boundaries during filter execution.

We have implemented a prototype of the Advanced Visualizer. As seen in figure 7.3, this Visualizer has access to filters from menus, and the parameters of each filter can be adjusted in a GUI.

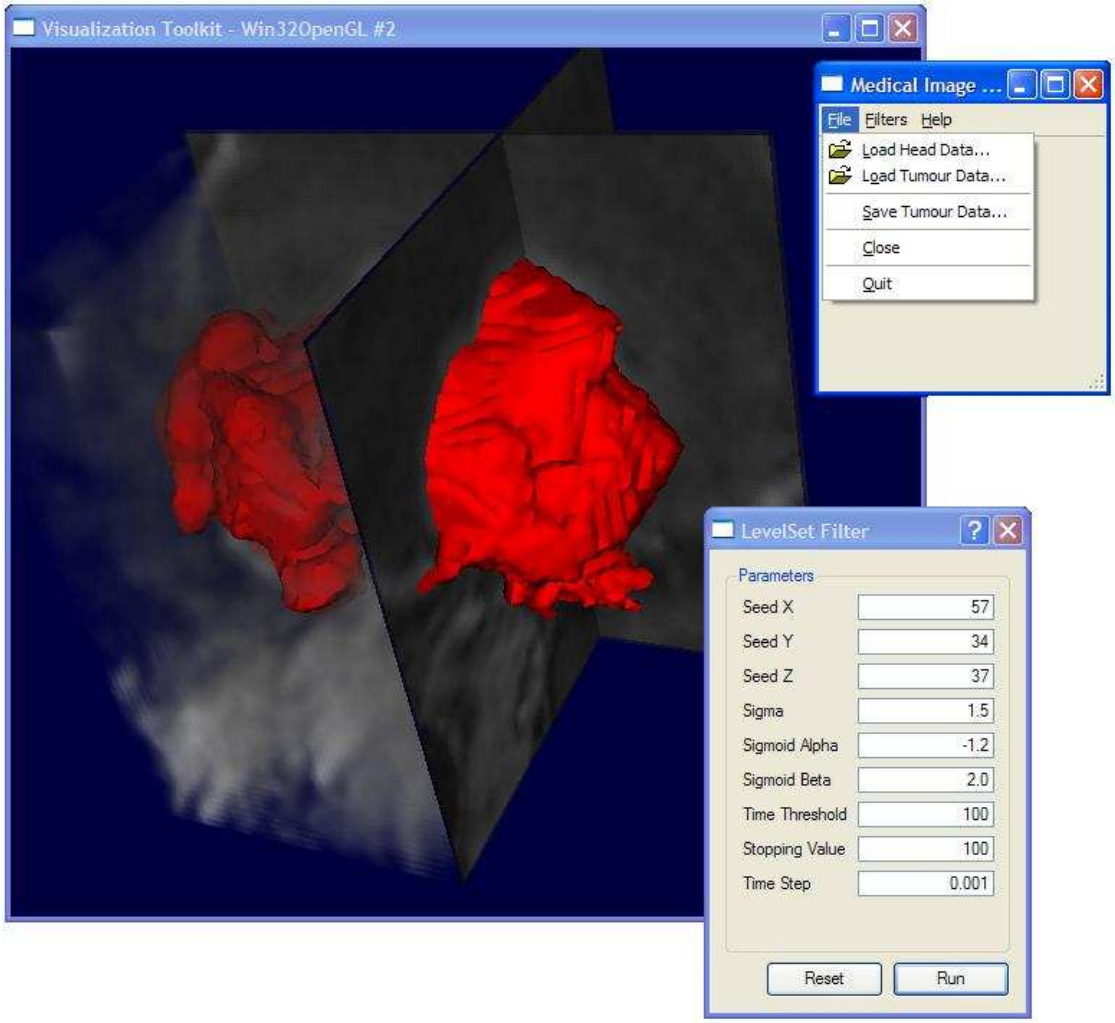


Figure 7.3: Screenshot from the Advanced Visualizer application.

Chapter 8

Discussion and conclusions

We have investigated different methods for use in medical image segmentation, more specific on how to segment tumours from MRI and ultrasound data of the human brain. In this chapter we will summarize and discuss the most important aspects of the material we have provided.

8.1 Discussion

During our work with this thesis, we have not been provided with manually segmented images to compare our results with. Manual segmentation is a very labour-intensive process and must also be done by experts to be able to produce satisfactory results. We was therefore not able to evaluate our results by a numerical comparison to a "perfect" segmentation. Some of our results have been evaluated by experts, but because we had limited time with them, only a few of our results have been evaluated this way. The rest of the evaluation has been done by us. Because we do not have any medical background, the results from our evaluation is less reliable. In some of the cases it is rather easy to see the exact position of the tumour boundary, but in other cases, particularly with the ultrasound images, the border of the tumour is often difficult to detect and can easily be misplaced. We have nevertheless tried to give a thorough judgement of the results.

Because there was not a reliable snake implementation available in ITK, we have implemented a somewhat simplified version of the GVF snake. The snake produced very promising results, in some cases the tumour boundary was found with very high precision, but in other cases the snake produced weaker results. It was often very sensible to initialization.

We have implemented and testes four variants of the level set method. Two of the methods gave very good results when used on the MR images and in some cases when used on the ultrasound images. The second one of these methods gave slightly better results than the first one in most cases, but is much slower. The other two variants of the level set method were supposed to take as input an already segmented tumour and try to improve this result. These filters did not give any good results.

Compared to the region growing method, which is a much simpler segmentation approach, the active contour algorithms perform very good. The region growing approach is only able to produce satisfactory results when the tumour is uniform and clearly separated from the surrounding tissue and the background.

The performance of the active contour filters we have tested depends strongly on the quality of the underlying image and the result from the preprocessing. In cases where the tumour is relatively uniform and is clearly separated from the surrounding tissue, active contours are able to find the details of the boundary with very high precision. In cases where the image is heavily corrupted by noise, the active contours are not able to find the boundary that precisely. This is because the image is smoothed during preprocessing. Even though we use smoothing filters that is suppose to preserve edge information in the image, the edge information is smoothed due to the high amount of noise.

In cases where the tumour lies close to other brain structures with similar intensity distribution, the active contour methods get more problems than when the tumour is clearly separated from the surrounding tissue. This is caused by weak tumour edges in the transition between the tumour and the neighbouring brain structure. The active contours now have to rely more on the shape of the surface, and the tumour rarely has a completely circular shape (like most of the filters assume).

In some cases the active contour filters, particularly the snake, find the wrong tumour edges because of inaccurate initialization. If the initial surface lies close to other edges in the image, the surface has a tendency to converge towards this edge. This problem can however be solved by allowing the user to guide the evolution of the surface during filter execution.

Some of the same problems may be experienced if the tumour has a non-uniform intensity distribution. There may occur a lot of edges inside the tumour causing undersegmentation. The problem can be solved by applying more smoothing of the image, but this often comes to the expense of that the tumour edges are smoothed as well, resulting in an oversegmentation of the result.

We presented a selection of our results to a group of experts. The group consisted of two surgeons and one researcher. On the overall, they were very satisfied with our results. In particular they were positive surprised over the quality of the results we have achieved on the ultrasound images. They also stated that a higher resolution on the datasets may lead to even better results. For a segmentation result to be used for navigation during surgery, it has to be very close to a perfect segmentation. The group of experts pointed out that even if the boundary does not completely coincide with the contour of the tumour, the result may very well be useful in other applications, for instance as a help during surgery planning.

All of the active contour filters require the user to specify several parameters. This can be a very time consuming task, and in order to be done effectively, it require the user to understand how the parameter values influence the result. This may make the filters difficult to use for surgeons.

The computation time needed for each of the level set filters was recorded. It turned out that a large part of the computation time was related to preprocessing. This is because a preprocessing filter works on the entire image, while in most cases an active contour calculates values close to the evolving boundary. Due to this, a great part of the computation time can be reduced by running the active contour filter on a cropped version of the original dataset. The computation time also varies according to whether the active contour calculates the result iteratively or not.

8.2 Conclusions

The simplified snake we have implemented has given promising results. The snake approach gives fairly good results in some cases, but has to be further developed and tested. The fast marching level set and the geodesic level set approaches have generally provided the best segmentation results. The former is faster to execute but the latter produces slightly better results. We therefore recommend to use one of these two approaches for brain tumour segmentation.

The results achieved in this report show that active contours are well suited for use on segmentation of neuro tumours. Their ability to take both high- as well as low-level information from an input image make them far more robust compared with more simple segmentation methods such as region growing.

Chapter 9

Future work

Through our work we have tested a set of segmentation methods on a selection of MR and US images of brain tumours. We have found a couple of methods producing good results in most of the test cases. The test cases consists of MR and US images of five different tumour types, most of them are low grade tumours.

The snake algorithm has only been tested to a small extent and only with a simplified three dimensional model. A more advanced model of the snake surface should be tested. There exists other snake formulations that might perform better than the one we have tested. This needs to be tested to a greater extent.

The algorithms we have tested have experienced some problems related to neighbouring edges positioned close to the tumour edge. This has often caused the algorithms to either under- or oversegment the tumour. To eliminate this problem a solution may be to increase the user interaction. Level set algorithms initialised by seedpoints may perform better if the number of seed points is increased. Methods that start evolving from an initial contour, may benefit from a more sophisticated way to declare the initial contour. It should also be tested if user guidance during snake evolution will lead to better results.

The applications we have developed are stand-alone applications. They can easily be integrated into existing software, like eg. CustusX. To be able to make them a useful tool for surgeons and other medical personnel the applications should offer a user interface that is intuitive and easy to use. They should also be integrated into an application offering other functionality, like eg. registration.

To further improve the usability of the application, it would be beneficial to develop a set of parameter values that work well on images with particular characteristics. This would reduce the time and effort needed to adjust parameter values.

If the application is supposed to be used during surgical procedures, the computation time of the segmentation algorithms should be reduced. This can make it possible to give real-time feedback to the surgeon.

Bibliography

[1] Center for Morphometric Analysis, Massachusetts General Hospital. http://www.cma.mgh.harvard.edu/seg/typical_data.html.

[2] General Electric Excite 3 MRI Scanner. http://www.gehealthcare.com/usen/mr/s_excite3/index.html.

[3] Imaginis Corporation. http://imaginis.com/obstetrics/.

[4] Siemens ACUSON Sequoia Echo 256 ultrasound scanner. http://www.medical.siemens.com/.

[5] American Brain Tumor Association (ABTA), editor. *A Primer of Brain Tumors*. ABTA, 2004.

[6] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours, 1993.

[7] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic Active Contours, 1995.

[8] R.F. Chang, W.J. Wu, D.R. Chen, and W.K. Moon. 3-D Snake for US in Margin Evaluation for Malignant Breast Tumour Excision Using Mammotome, 2003.

[9] G. E. Diaz. Ultrasound. http://www.drgdiaz.com/first.shtml.

[10] E. Helseth et al. Intrakranielle svulster hos voksne. *Tidsskriftet Norsk Lægeforening*, 2003.

[11] J. G. Bjåle et. al. *Menneskekroppen*. Gyldendal akademisk, 2000.

[12] R. C. Gonzales and R. E. Woods. *Digital Image Processing, 2nd Edition*. Prentice Hall, 2001.

[13] T. A. Gould. How MRI works. http://electronics.howstuffworks.com/mri.htm.

[14] S. Holm. Medisinsk ultralydavbildning. http://www.ifi.uio.no/~sverre/papers/99_-FFV.pdf.

[15] J. P. Hornak. The basics of MRI. http://www.cis.rit.edu/htbooks/mri/.

[16] L. Ibáñez, W. Schroeder, L. Ng, and J. Cates. The ITK software guide, 2003. http://www.itk.org/ItkSoftwareGuide.pdf.

[17] National Cancer Institute. What you need to know about brain tumors. http://www.cancer.gov/cancertopics/wyntk/brain.

[18] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models, 1987.

[19] F. Lindseth. Ultrasound guided surgery: Multimodal Visualization and Navigation Accuracy, 2002.

[20] S. Lobregt and M. A. Viergever. A Discrete Dynamic Contour Model, 1995.

[21] Tim McInerney and Demetri Terzopoulos. A Dynamic Finite Element Surface Model for Segmentation and Tracking in Mulitdimensional Medical Images with Application to Cardiac 4D Image Analysis, 1994.

[22] I. Middleton and R. I. Damper. Segmentation of Magnetic Resonance Images using a combination of Neural Networks and Active Contour Models, 2003.

[23] G.M. Navestad and S.D. Gjedrem. Medical Image Segmentation, 2004.

[24] C. J. F. Noorden, L. C. Meade-Tollin, and F. T. Bosman. Metastasis. 1998.

[25] S. Osher and J.A. Sethian. Fronts propagating with Curvature-Dependent Speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79, 1988.

[26] P. Perona and J. Malik. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[27] J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1999.

[28] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision, 2nd Edition.* PWS Publishing, 1999.

[29] J.K. Udupa, V.R. LeBlanc, H. Schmidt, C. Imielinska, P.K. Saha, G.J Grevera, Y. Zhuge, L.M Currie, P. Moholt, and Y. Jin. A Methodology for Evaluating Image Segmentation Algorithms, 2002.

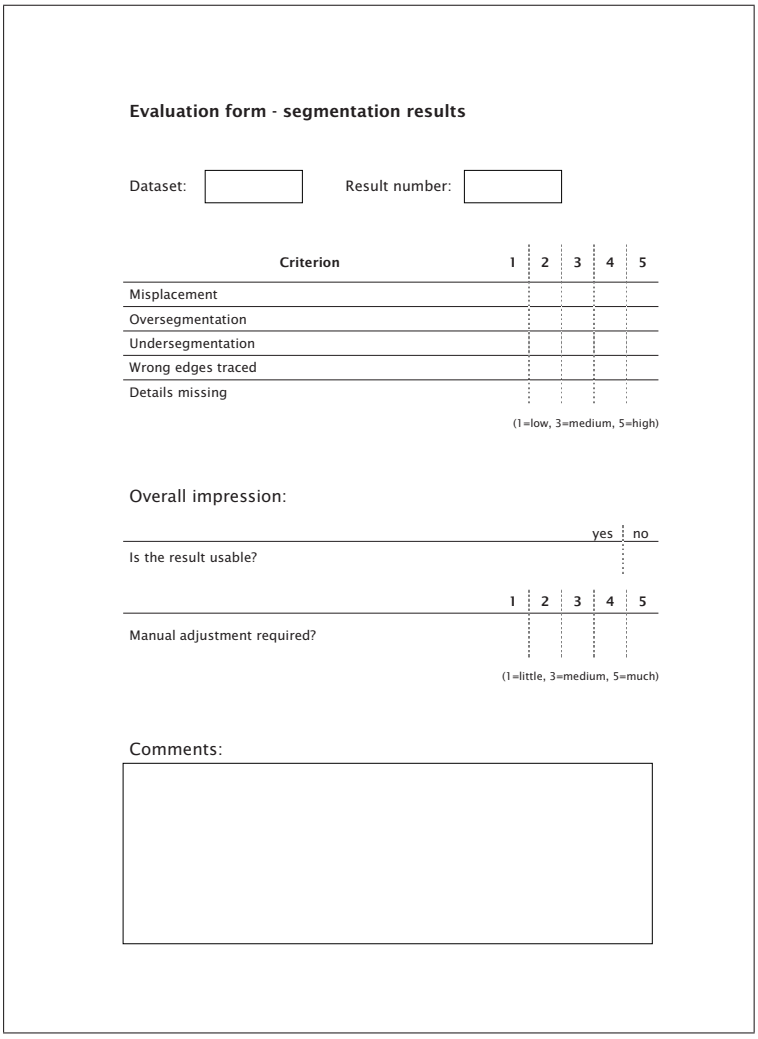
[30] J. Wang and X. Li. A system for segmenting ultrasound images, 1998.

[31] C. Xu and J. L. Prince. Gradient Vector Flow, Deformable Models, 2000.

[32] P.J. Yim and D.J. Foran. Volumetry of Hepativ Metastases in Computed Tomography using the Watershed and Active Contour Algorithms, 2003.

Appendix A

Evaluation form



Evaluation form - segmentation results

Dataset: Result number:

Criterion	1	2	3	4	5
Misplacement					
Oversegmentation					
Undersegmentation					
Wrong edges traced					
Details missing					

(1=low, 3=medium, 5=high)

Overall impression:

	yes	no
Is the result usable?		

	1	2	3	4	5
Manual adjustment required?					

(1=little, 3=medium, 5=much)

Comments:

Figure A.1: The form used for evaluation of segmentation results.

Appendix B

Source code and datasets

The source code and the test datasets can be found on the enclosed CD. This appendix provides a short description of the CD content.

B.1 Segmentation filters

Development of the segmentation filters requires ITK version 2.0. The segmentation filters are located in the following folders:

- Segmentation/RegionGrowing
- Segmentation/GvfSnake
- Segmentation/FastMarchingLevelSet
- Segmentation/GeodesicLevelSet
- Segmentation/CannyEdgeLevelSet
- Segmentation/LaplacianLevelSet

Each filter is organized as follows:

- Segmentation/$filtername$ — source code
- Segmentation/$filtername$/bin — executables
- Segmentation/$filtername$/Project — MS Visual Studio project files

The documentation of the segmentation filters is located in the following folder:

- Segmentation/doc

B.2 Basic visualizer

Running the basic visualizer requires VTK version 4.4. Development of the basic visualizer requires VTK version 4.4 and ITK version 2.0. The basic visualizer application can be found in the following folders:

- BasicVisualizer/src — source code
- BasicVisualizer/doc — documentation
- BasicVisualizer/bin — executables
- BasicVisualizer/project — MS Visual Studio project files

B.3 Advanced visualizer

Running the advanced visualizer requires VTK version 4.4. Development of the advanced visualizer requires QT version 3.3.3, VTK version 4.4 and ITK version 2.0. The basic visualizer application can be found in the following folders:

- AdvancedVisualizer/src — source code
- AdvancedVisualizer/doc — documentation
- AdvancedVisualizer/bin — executables
- AdvancedVisualizer/project — MS Visual Studio project files

B.4 Datasets

Each of the five datasets N241, N351, N359, N360 and N378 is organized in the following folders:

- Datasets/Nxxx/Master MRI/Fullsize — full-size MR and US data with MRI as master
- Datasets/Nxxx/Master MRI/Subset — subset of the MR and US data with MRI as master
- Datasets/Nxxx/Master US/Fullsize — full-size MR and US data with US as master
- Datasets/Nxxx/Master US/Subset — subset of the MR and US data with US as master