Pedro Miguel Tenreiro Cardoso

Licenciado em Engenharia Informática

Modeling and visualization of medical anesthesiology acts

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientador : Adriano Lopes, CITI, FCT/UNL

Júri:

Presidente:	João Alexandre Carvalho Pinheiro Leite
Arguente:	Samuel de Sousa Silva
Vogal:	Adriano Martins Lopes

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro, 2013

Modeling and visualization of medical anesthesiology acts

Aos meus pais

Acknowledgements

First and foremost I would like to thank my dissertation supervisor, Dr. Adriano Lopes, for all the support he offered me during the development of this dissertation. I am also grateful to Dr. Samuel Silva for the changes he suggested that I make in this dissertation. The MITK community was also very helpful while trying to solve some of the problems related to the MITK toolkit. Lastly, I would like to thank my sister, Maria J. Cardoso, for helping me to identify some of the anatomical structures in the lumbar region.

Resumo

A visualização médica evoluíu das simples imagens 2D num quadro de luz para imagens computorizadas em 3D. Esta tendência possibilitou aos médicos encontrarem melhores formas de planear cirurgias e de diagonosticar pacientes. Embora exista uma grande variedade de software de imagiologia em 3D, este é ainda insipiente nas respostas que proporciona a actos de anestesiologia. De facto, tem sido pouco o trabalho relacionado com anestesiologia. Consequentemente, médicos e estudantes de medicina têm pouco apoio no estudo da anestesia no corpo humano. Com este trabalho, esperamos contribuir para a existência de melhores ferramentas de visualização médica na area da anestesiologia. Médicos e em particular estudantes de medicina devem poder estudar os actos de anestesiologia de uma forma mais eficaz. Eles devem poder identificar as melhores localizações para administrar a anestesia, estudar sobre o tempo que demora a anestesia a surtir efeito no paciente, etc. Neste trabalho apresentamos um protótipo de visualização médica com três funcionalidades principais: pré-processamento de imagem, segmentação e renderização. O pré-processamento é usado sobretudo para remover ruido das images, obtidas através de scanners de imagem. Na fase de segmentação é possível identificar estruturas anatómicas relevantes. Como prova de conceito, a nossa atenção esteve centrada na região lombar do corpo humano, com dados obtidos através de scanners de ressonância magnética. O resultado da segmentação implica a criação de um modelo 3D. Relativamente à renderização, os modelos 3D são visualizados usando o algoritmo marching cubes. O software desenvolvido também suporta dados dependentes do tempo. Assim, poderemos representar o movimento da anestesia no corpo humano. Infelizmente, não nos foi possível obter tal tipo de dados. No entanto, utilizámos dados de pulmões humanos para validar esta funcionalidade.

Palavras-chave: anestesiologia, visualização médica, segmentação, dados dependentes do tempo.

Abstract

In recent years, medical visualization has evolved from simple 2D images on a light board to 3D computarized images. This move enabled doctors to find better ways of planning surgery and to diagnose patients. Although there is a great variety of 3D medical imaging software, it falls short when dealing with anesthesiology acts. Very little anaesthesia related work has been done. As a consequence, doctors and medical students have had little support to study the subject of anesthesia in the human body. We all are aware of how costly can be setting medical experiments, covering not just medical aspects but ethical and financial ones as well. With this work we hope to contribute for having better medical visualization tools in the area of anesthesiology. Doctors and in particular medical students should study anesthesiology acts more efficiently. They should be able to identify better locations to administrate the anesthesia, to study how long does it take for the anesthesia to affect patients, to relate the effect on patients with quantity of anaesthesia provided, etc. In this work, we present a medical visualization prototype with three main functionalities: image pre-processing, segmentation and rendering. The image pre-processing is mainly used to remove noise from images, which were obtained via imaging scanners. In the segmentation stage it is possible to identify relevant anatomical structures using proper segmentation algorithms. As a proof of concept, we focus our attention in the lumbosacral region of the human body, with data acquired via MRI scanners. The segmentation we provide relies mostly in two algorithms: region growing and level sets. The outcome of the segmentation implies the creation of a 3D model of the anatomical structure under analysis. As for the rendering, the 3D models are visualized using the marching cubes algorithm. The software we have developed also supports time-dependent data. Hence, we could represent the anesthesia flowing in the human body. Unfortunately, we were not able to obtain such type of data for testing. But we have used human lung data to validate this functionality.

Keywords: anesthesiology, medical visualization, segmentation, time-dependent data,

Contents

1

Introduction

Visualization as a discipline to present information is not a recent phenomenon. Before the so-called *information society*, visualisation techniques were used in maps and scientific drawings for over a thousand years.

Figure 1.1: The *Ptolemy world map* as an example of data representation.

In this chapter we provide a generic overview about computer visualisation, in particular in relation to medical visualization. Then, we set out the boundaries for the problem we are studying and finally we point out the organisation of the remaining chapters in this dissertation.

1.1 Visualization Overview

Today we are exposed to a sheer amount of information. As a consequence, the tasks of filtering and understanding information have become a difficult ones. In order to

investigate a huge amount of data, we can use computer generated graphs or images. Visualization itself has already proved to be a powerful tool for data investigation in the past so the know-how accumulated for many centuries has been transferred to computer visualization.

The emphasis on computer visualization started in the early 1990s with the special issue of *Computer Graphics on Visualization in Scientific Computing*. Since then, multiple conferences solely focusing on visualization have been organised. Two of then are the *Eurographics Workshop on Visualization* and the *IEEE Conference on Visualization*.

Despite initial differences that were set up according to the information to be represented, that is, whether it was scientific numerical data or more generic non-numerical data, we are witnessing a more common approach to data visualization. Hence, the fields of *Scientific Visualization* and *Information Visualization* are getting more and more interrelated.

In order to illustrate the importance of visualization nowadays, we list below some important applications:

Medical visualization. Anatomic data is acquired through scanning device machines, which is then presented using a volume visualization technique, for instance isosurface extraction. This kind of applications can be used to help physicians to diagnose cancer for example.

Flow visualization. The objective is to make flow patterns visible. Most fluids (air, water, etc.) are transparent, thus their flow patterns are invisible to us without some special methods to make them visible. The data can be obtained by flow simulation or by measuring experiments. The design of new aircrafts can be validated by using simulations with flow visualization. In this way, there is no longer a need to construct expensive prototypes in order to test the project design.

Geographic information systems. For a long time maps have been used to visualize geographic data. Techniques like height fields or isolines are commonly used in computer visualization to show topographic information like mountains.

Information visualization. One of the focus is to use computer-based tools to explore large amount of data. Big databases and multi-modal data increasingly require appropriate visualization techniques. For example, business data visualization is widely used to represent economic data.

Microscopic data visualization. Current microscopes are capable of generating large numbers of images. Data visualization techniques can also be applied to these images to quickly identify which ones are important. Microscopic data visualization is used in the visualization of molecules or atomic structures studied in biology, chemistry and physics.

Large-scale data visualization. It is particularly used when studying natural phenomena. For example, earth sciences turbulence calculations can produce large amount of data, leading to huge demands on power computation. Astrophysics is another example which deals with data measured at a huge scale so no direct data investigation is possible. Some techniques used in large-scale data visualization have to deal with data reduction, sometimes trading data resolution for some level of user interactivity.

The first application we have mentioned above – Medical visualisation – is an important part of the visualisation field. It is also the one where our work fits in. In the following we will extend this subject.

After the x-ray technology was invented in the 19th century, it was then clear that it would be a valuable diagnostic tool. Nevertheless, the x-ray technology was still limited as physicians had to work with a minimal set of images.

Once computed tomography (CT) and Magnetic Resonance (MR) were introduced, we were able to work with larger set of images from a given patient and to have a more detailed view of his body. The amount of images was still limited, but the traditional light boards started to become small to contain all the images, thus making it harder to analyze the patient body. With the rise of computational power, computers helped to solve the problem. By the end of the 1990s, computers were just an alternative to the light board.

When the first multi-slice CT scanners were introduced, there was a new task for computers: to create a 3D representation of the anatomical structure pictured in the images provided by these scanners. It was a perfect solution, as the huge amount of images obtained were too hard to analyze one by one, and consequentely it was hard to get an overall description of the patient's body. Fortunately a 3D representation would help to solve these problems.

With the rise of medical visualization, also new ways to visualize volumes in medicine have emerged. For example:

Diagnostics. With 3D medical visualization, new ways to diagnose patients have been found. This is the case of Virtual Endoscopy, which avoids the painful process the patient would have to face by using a real endoscope. An example of a virtual endoscopy is shown in Figure 1.2.

Communication. 3D medical visualization now helps the communication between the radiologists, surgeons and patients. It is a fact that 3D renderings are much easier to understand by people.

Planning. 3D visualization is able to aid procedures of treatment, surgery, etc. An example of this is shown in Figure 1.3.

Training. Students and doctors can be trained with the help of a virtual training environment.

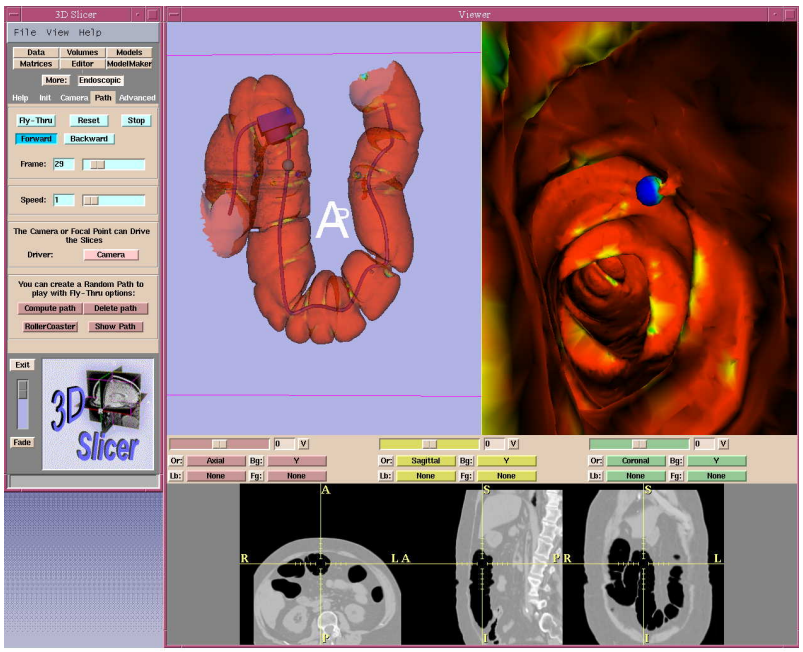

Figure 1.2: Virtual Endoscopy of the colon (virtual colonscopy) [MITa].

1.2 Problem Description

Some anesthesias (e.g. epidural anesthesia) can be hard to administer by doctors, particularly when they are professional juniors. Teaching a doctor or student the behavior of the anesthesia along with the location in which it must be administered can be an hard process.

Most of the available 3D medical applications don't allow the representation of time-varying data volumes. Because of this, there isn't much support when it comes to anesthesiology acts. Furthermore, most of the research in the area of medical visualization is focused in the fields of diagnostics and surgical planning.

The main purpose of this dissertation is to provide a tool that allows students or young doctors to study anesthesiology acts. This tool should create a 3D representation of an anatomical structure along with the anesthesia substance. Also, it should incorporate an appropriate annotation system to facilitate the studying process.

By nature, we are dealing with time-dependent data. That is, as time goes by we should be able to see where the anesthesia substance is flowing inside the body. As a consequence, it would be interesting to provide a 3D model of the all process varying in

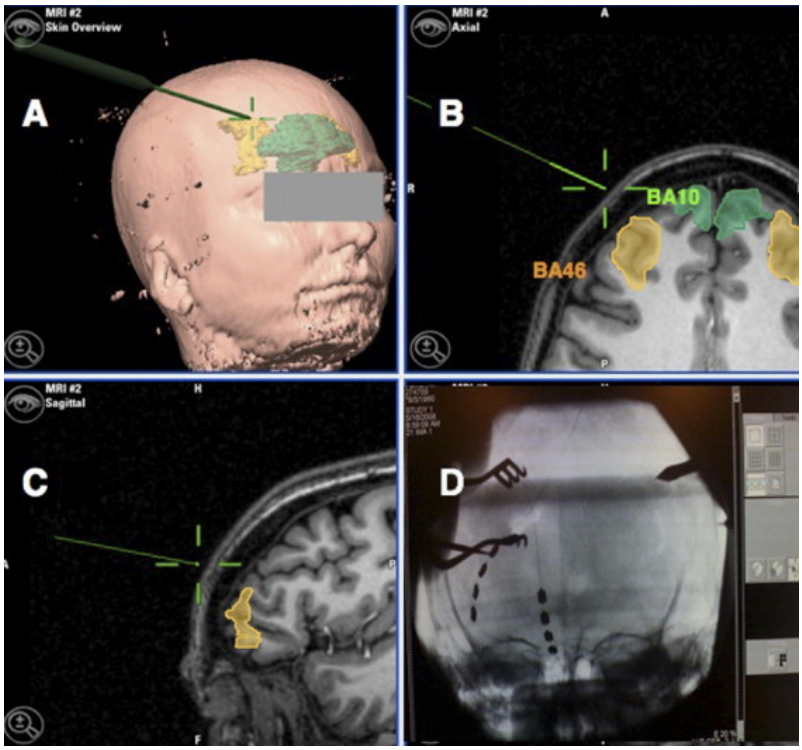

Figure 1.3: Surgical plan of a neurosurgery [vdB].

time.

In order to answer positively to the requirements we have set out, we first must identify our main challenges and to establish how we can overcome them.

Before physicians started using 3D imaging software, they used to analyze the patients with the 2D images from CT or RM scanners. As the computational power has increased, 3D imaging started being used more often and it became an active area of research in computer science. Today we have multiple ways of representing the human body in 3D from data acquired using medical imaging machines. Although a lot of research has been done, to the best of our knowledge there is no software allowing doctors or medical students to study the behaviour of anesthesia substances in the human body.

As it was mentioned, there are multiple methods to represent the human body in 3D. These methods have their own advantages and disadvantages, and so one of the problems we are facing revolves around choosing the best methods to create our tool. We envisage four major areas to look at:

- User interface and basic visualization.
- Image segmentation.
- Rendering of data volume.
- Rendering of time-varying data volume.

User interface and basic visualization. This part of the problem consists of designing a proper user interface with the potential of satisfying all user requirements. For example, we have to decide what menus and what types of visualization will be available.

Image segmentation. As previously stated, 3D medical imaging software usually goes through a phase of segmentation, so our application will not be different. There are several segmentation methods available, some of them are more suited to our problem than others. For instance, for our problem we are interested in those that work better with soft tissues. There are other aspects we must also take into account when choosing the segmentation method. For example, how accurate is the method, how sensitive is the method to noise, etc. The level of interactivity in the segmentation phase is also a major concern. The physician must be able to validate the segmentation and, in case the segmentation is faulty, he must be able to make proper adjustments in the used segmentation method used.

It is crucial that this phase of the problem is well addressed because the next phases depend a lot on this one.

Rendering of data volume. This phase depends on the image segmentation phase. At this point, the components identified in the segmentation phase are used to create the corresponding 3D model. To do this, there are multiple approaches (e.g. volume rendering and surface-based rendering). We have to decide which approach better fits our objectives. The chosen approach must be of low computational complexity, because the physician must be able to examine a given anatomical structure from different perspectives and if using a complex rendering method might slow down this process. Although at first this might not seem very important nowadays because of the growth in CPU/GPU power, we must also take into account that, in the next phase, we need to represent a time-varying data volume. The representation of multiple members identified in the segmentation phase must be well distinguishable. It is a fact that it is necessary to differentiate the anesthesia from the anatomical structure at hand.

Rendering of time-varying data volume. Because we want to be able to analyze the behaviour of the anesthesia, it is necessary that such behaviour varies as time passes. That is what happens in reality. The main issue here might be the size of the data set used, even though we do not expect to have very large data sets during the testing phase. However, that doesn't mean it will not be the case in the future.

Data type. Regarding the type of data in which we focus our attention, we intend to use mainly MR images. As for CT images, we might not able to obtain the amount of images we would need so we will not give CT too much attention. Moreover, no other type of scanning images are considered at all.

1.3 Dissertation Organization

The remaining chapters in this dissertation are organized in the following way:

We start Chapter 2 by analysing work that might be related to ours. The main themes are segmentation, rendering and time-dependent visualization. In the segmentation section we present basic segmentation algorithms and algorithms widely used in medical visualization. In the rendering section we present the marching cubes algorithm and volume rendering techniques. Regarding time-dependent visualization, we present some of the main problems and suggested techniques to solve these.

In Chapter 3 – Lumbar region segmentation – we discuss the techniques used during the segmentation of the lumbar region and the reasoning behind it. The whole segmentation process goes through 3 main stages: *pre-processing*, which addresses noise reduction, *algorithm initialization*, which demonstrates some of the necessary intermediate steps we have to go through to run a segmentation algorithm successfully and finally the *segmentation*, which deals with choosing and applying segmentation algorithms. The anatomical structures segmented are the abdominal aorta, back muscles, spinal canal and the vertebral body.

Chapter 4 – Time-Dependent rendering – presents and compares two approaches used to segment and render time-dependent datasets. The first approach tries to process each of the time slices one at a time while the other has no defined order to process the time slices (parts of each time slice can be processed instead of a whole slice) and simply presents the final results at the end.

Chapter 5 – Prototype – presents the architecture used to build the prototype and demonstrates how the interface works given that the user wants to segment a certain anatomical structure and interact with, or modify the segmentation.

Finally, Chapter 6 – Conclusions – presents the conclusions taken from the work carried out and indicates areas for further research.

2

Related Work

In this chapter we present some related work covering aspects of data segmentation, rendering, time-varying visualization and development toolkits.

First, we start by introducing the framework of the data visualization process. Then we lay out some aspects of data acquisition, followed by a discussion about some techniques for image pre-processing. Then we present segmentation algorithms, whether being the general ones or the ones commonly used in medical imaging. Next, we discuss 3D visualization techniques, both in terms of volume and surface-based visualization. We then make reference to the difficult issue of time-varying visualization. Finally, we present a set of toolkits that can be used to develop a tool for medical visualization.

2.1 Data Visualization Process

The data visualization process can be described according to the pipeline presented in Figure 2.1.

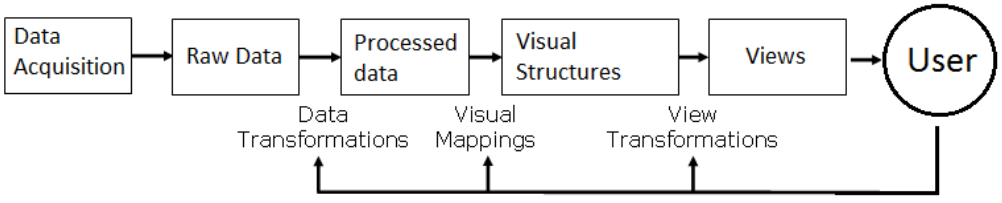

Figure 2.1: The Data Visualization pipeline[cp].

Data Acquisition represents the process to obtain *Raw Data*. In medical applications,

the method can be a CT or MR scanning. In other kinds of applications (e.g. flow visualization), the raw data can be obtained through a simulation.

To make the *Raw Data* useful, *Data Transformations* need to be applied in order to create a set of *Processed Data*. Examples of *Data Transformations* are statistical computations like averaging values. Operations like data stratification or ordering *Raw Data* by some attribute can also be made at this stage of the visualization process. If there are problems like errors in data, they must be detected and addressed right at this stage in order to avoid presenting erroneous information to users.

Then, a mapping function is applied to the *Processed Data* in order to create a *Visual Structure*. At this stage, it is important to consider the goal of the visualization so a proper mapping function will be used. For example, statistical representation (e.g. histograms) is a classic technique to build *Visual Mappings*.

Finally, *View Transformations* can then be applied interactively in order to modify representations. An example of a transformation made at this stage is *Zooming*.

2.2 Data Acquisition

Medical data acquisition, as the first stage in the visualization pipeline (see Figure 2.1), is particularly concerned with the technologies used to acquire data sets of the anatomical structure of a given patient. In the following, we present some of the most important technologies considering our problem.

2.2.1 Computed Tomography

Computed Tomography (CT) was the first imaging technique to generate slice images and therefore it became a very important technology in 3D visualization.

Figure 2.2: A CT Scanner [Hos].

A CT scanner can be compared to digital x-ray device, where a x-ray tube is used to generate radiation. The amount of radiation generated by the x-ray tube is tracked by a reference detector. The x-rays passes through the patient's body and are attenuated by the different types of tissue. In the end, another detector gathers the radiation that passed

through the patient's body. The value obtained by the last detector is then compared to the radiation tracked by the reference detector in order to check how much did the radiation weakened.

2.2.2 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is also an important imaging technique in 3D visualization. Similarly to CT, this method can also generate slice images. An MRI scanner contains a powerful magnet which is used to align the magnetization of some atomic nuclei in the body. Radio frequency fields are then used to alter the alignment of this magnetization. This causes the nuclei to produce a rotating magnetic field detectable by the scanner. This information is used to construct and image of the scanned area of the body.

2.2.3 Positron Emission Tomography

Positron Emission Tomography (PET) is an imaging technique that produces a 3D data volume of functional processes in the body (a radioactive tracer is used to show chemical activity of anatomical structures). To produce a 3D data volume, the system detects gamma rays emitted indirectly by a positron-emitting radionuclide, which is introduced into the body. The 3D volumes are then constructed by using computerized reconstruction procedures, like the ones used in CT and MRI.

2.2.4 Discussion

As expected, the data acquisition technologies we have mentioned above have their own advantages and disadvantages.

Computed Tomography is suited to detect bone injuries, chest imaging and cancer detection. This technique is widely used in emergency room patients. When it comes to details of bony structures, CT provides a good detail. Although with less detail, it is also able to depict soft tissues and blood vessels.

Magnetic Resonance Imaging is suited for soft tissue evaluation such as ligament and tendon injury, spinal cord injury and brain tumours. However, it is worth to point out that bony structures are not as detailed as soft tissues.

Positron Emission Tomography is used to scan biological processes within the body. It is widely used to detect cancer due to its ability to distinguish living and dead tissues or benign and malignant disorders. This method is sometimes combined with CT so that physicians can more accurately diagnose cancer, as well as heart disease and certain brain disorders.

2.3 Image Pre-processing

Before applying a data segmentation algorithm, sometimes it is necessary to denoise the image as the segmentation algorithm to be used may be too sensitive to noise. If there is no proper care then the resulting segmentation will have poor quality. In this section we present two smoothing methods widely used: edge preserving smoothing and median filter.

2.3.1 Edge Preserving Smoothing

The main problem of smoothing/denoising is that it tends to blur away sharp boundaries in the image that could help to distinguish anatomical structures. Perona and Malik [PM90] introduced a method called *anisotropic diffusion*. With this technique it is possible to reduce image noise while preserving or even enhancing edges in the image. This was possible by introducing a variable diffusion coefficient (which was constant before). This way it is possible to limit the amount of smoothing near edges or boundaries. The *anisotropic diffusion* equation if given in equation 2.1, where $I(x, y, t)$ is the image intensity, div and Δ are the divergence and gradient operators, and $c(x, y, t)$ is the diffusion coefficient.

$$\frac{\partial}{\partial t} I(x, y, t) = div(c(x, y, t) \nabla I(x, y, t)) \tag{2.1}$$

The diffusion coefficient must encourage the forward diffusion inside smooth regions and backward diffusion at high gradient locations. Perona and Malik [PM90] proposed multiple mathematical functions for the diffusion coefficient, being equation 2.2 one of them. The parameter *k*, the *conductance parameter*, controls the sensitivity of the process to edge contrast.

$$c(x, y, t) = \exp\left(- \left(\frac{|\nabla I(x, y, t)|^2}{k} \right) \right) \tag{2.2}$$

This particular equation takes advantage of the high contrast edges rather than the low contrast ones.

2.3.2 Median Filter

The median filter can also be used as a robust approach for noise reduction. The filter is particularly efficient against *salt-and-pepper noise* 2.3. The median filter computes the value of each output pixel as the statistical median of the neighborhood of values around the corresponding input pixel. A radius for each dimension must be provided to create a neighborhood.

An example of this process in depicted in Figure 2.4, where the pixel in the middle (value of 25) is the one that will be modified. In 3D, the process is analogous.

Figure 2.3: Example of salt-and-pepper noise [Blo].

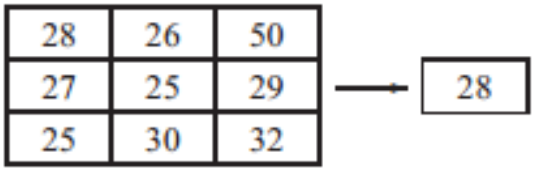

Figure 2.4: Median filter example [HJJtISC13].

This method tends to not damaging the boundaries and edges too much and it is of very little complexity. It is also known that, by applying the algorithm multiple times with the same radius, we can obtain results fairly similar to the ones that we obtain using the *anisotropic diffusion* filter mentioned above.

2.4 Generic Segmentation Algorithms

Image segmentation is the process of partitioning an image into multiple segments. This is a crucial task in medical visualization. The goal is to change the representation of the image in order to ease the identification of the objects of interest in the image and thus, to make the overall image analysis much easier. An image segmentation technique usually assigns labels to the image pixels and the pixels with same label share certain visual characteristics.

When applied to a stack of images, which is a typical scenario in medical imaging, the resulting contours after image segmentation can then be used to create 3D representations.

In the following we present some of the basic segmentation algorithms. As a first objective, we have to understand how basic segmentation can be done.

2.4.1 Thresholding

Thresholding is probably the simplest method of image segmentation [SS04]. It can be divided into two categories: *global thresholding* and *local thresholding*.

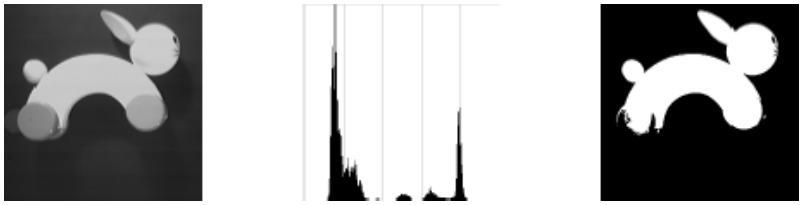

Figure 2.5: Thresholding of an image with a bi-modal intensity distribution [oE].

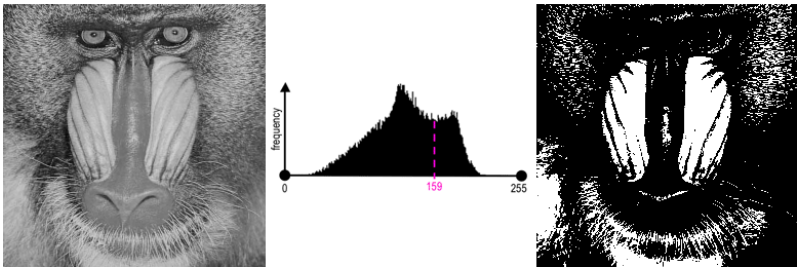

Figure 2.6: Thresholding of an image with a multi-modal intensity distribution [oUSoc].

In *global thresholding*, there are two groups of pixels: The group 1 corresponds to the set of pixels that have an intensity value greater or equal to the threshold value t; the rest of pixels (the ones with an intensity value lesser than t) belong to group 2. The complexity of this algorithm is very low, but because of this, it brings us some shortcomings. *Global thresholding* works fine with images with a bi-modal distribution (see Figure 2.5) but, as we can see in Figure 2.6, the algorithm will perform poorly if the image has a multi-modal distribution. To obtain better results with images with a multi-modal distribution, entropy based threshold algorithms were suggested in [PSY97]. These algorithms try to maximize the entropy in the thresholded image. That is, in images with bigger contrast between pixels, we can obtain better segmentation results using threshold techniques. An alternative to entropy based algorithms are the histogram shape based threshold algorithms [CL98], which try to force multi-modal intense distributions into a bi-modal distribution.

Instead of *global thresholding*, we can use *local thresholding*. This local method divides an image into sub-images and selects an independent threshold value for each sub-image. This technique is very sensitive to noise and some of the segmented regions might not correspond to the objects in the original image.

Recently, in the field of medical visualization, threshold methods have also been developed for CT images. K. J. Batenburg and J. Sijbers [BS09] suggested a projection distance minimization distance, which uses tomographic projection data to determine optimal thresholds. The thresholds are computed by minimizing the distance between the forward projection of the segmented image and the measured projection data.

2.4.2 Region-based Segmentation

Region-based image segmentation rely on grouping similar neighbouring pixels. There are two main methods: *split and merge* [CP80] and *region growing* [AB94].

2.4.2.1 Split and Merge

The *split and merge* method divides an image into multiple regions and then merges the ones that are similar according to some criteria (see Figure 2.7). To be more precise, the whole image is first considered as a single region. If a certain homogeneity criterion is not met, then the region is split into four quadrants. This process is repeated recursively within each region. When each of the regions contain only homogeneous pixels, the algorithm starts comparing all regions with their neighbor regions and merge the ones that are similar. It is an algorithm computationally fast.

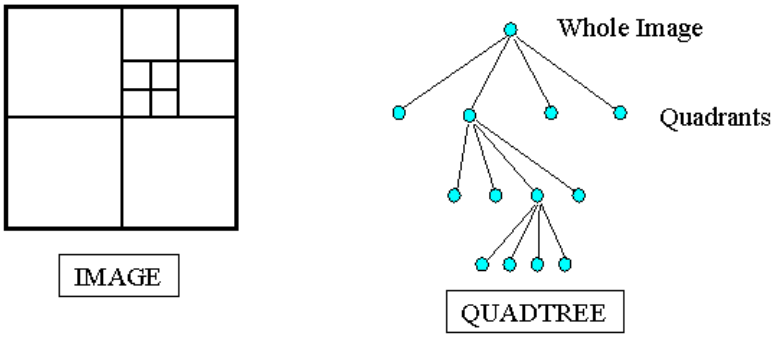

Figure 2.7: Split process of the Split and Merge algorithm [oUDoB].

2.4.2.2 Region Growing

The *region growing* method works in the opposite way relative to the *split and merge* method. It begins by selecting n seed pixels. The algorithm will then try to merge each of these pixels with similar neighbors (pixels with similar intensities) so creating a region of multiple pixels. An illustration of this process is shown in Figures 2.8 and 2.9.

R. M. Haralick and L. G. Shapiro have proposed a variant of this technique [SH01]. Basically the mean and variance of a region are used now to decide if a pixel belongs or not to that region.

The seed pixel can be selected manually or automatically. One algorithm that has been proposed to do so is the *Converging Square* algorithm [OS83]. However, images with too much noise might cause the algorithm to produce undesired segmentation.

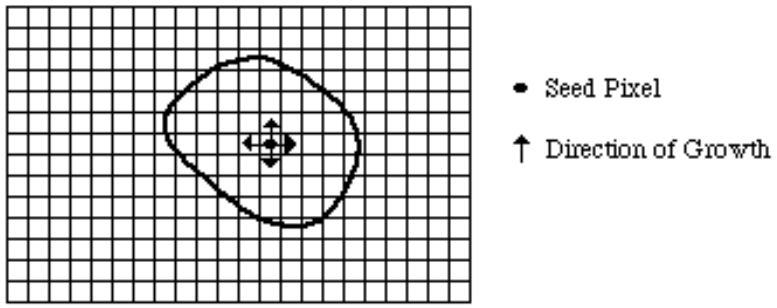

Figure 2.8: Region Growing algorithm: Start of a growing region [oCSI].

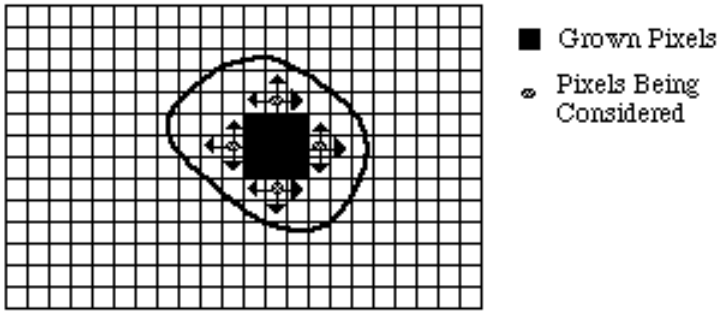

Figure 2.9: Region Growing algorithm: Growing process after a few iterations [oCSI].

2.4.3 Edge-based Segmentation

Edge-based segmentation algorithms use edge detectors to create segments in a given image. This helps to distinguish the multiple structures in an image. Examples of such algorithms, presented below, are the gradient magnitude filter and the canny edge detector.

2.4.3.1 Gradient Magnitude

The magnitude of the image gradient is extensively used in image analysis, mainly to help the determination of object contours and the separation of homogeneous regions. A gradient magnitude operator detects the amplitude edges at which pixels change their gray-level suddenly. For an image volume, the magnitude of the gradient vector assumes a local maximum at an amplitude edge. The magnitude is zero if the volume is constant. Equation 2.3 can be used to calculate the magnitude for a volume $f(\mathbf{x})$.

$$|\nabla f| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2 + (\frac{\partial f}{\partial z})^2} \tag{2.3}$$

The algorithm can assume a discrete differentiation approach based on the Prewitt operator [HJJtISC13] (other operators can be used). This operator computes an approximation of the gradient of the image intensity function. By applying the operator at each

point in the image, the result is either the corresponding gradient vector or the norm of this vector.

In the case of 2D, the operator uses two 3x3 matrices (masks), which are applied to the original image to calculate the approximations of the derivatives. One of the matrices is used for horizontal changes and the other one for vertical. Let A be the source image, G_x and G_y the images that contain the horizontal and vertical derivative approximations:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \text{ and } G_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & +1 \end{bmatrix} \tag{2.4}$$

Both images, (G_x and G_y), can be combined using equation 2.5.

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.5}$$

2.4.3.2 Canny Edge detector

An example of detector is the *Canny Edge* detector [Can86], which uses a double thresholding technique. After going through a smoothing step, the algorithm finds the edge strength by taking the gradient of the image. The Prewitt operator (see Equation 2.4) can be used to approximate the absolute gradient magnitude (edge strength). The edge direction must also be calculated. To do this we can simply use Equation 2.6, where θ is the angle (which corresponds to the edge direction) and $Arctan()$ is the arctangent. Finally, a threshold t_1 is used to detect edges by comparing the pixels in the image with the threshold. Any pixel that has a greater value than t_1 is presumed to be an edge pixel. Another threshold t_2 is defined as a criterion to connect possible edges to the already defined edge pixel. Any pixels that are connected to the edge pixels defined by t_1 and have a greater value than t_2 are also selected as edge pixels. Figure 2.10 shows a result of this detector.

$$\theta = Arctan(G_y/G_x) \tag{2.6}$$

Edge-based segmentation algorithms tend to find edges which are irrelevant to the object, and the contours of the object sometimes are disjointed. Edge-based segmentation algorithms usually have low computational complexity.

2.4.4 Graph-based Segmentation

Graph-based segmentation consists in the formation of a weighted graph, where each vertex corresponds to a region in the image. The edges are weighted with respect to some measure. Graph-based algorithms often try to minimize cost functions such as a *cut* function. For example, given an image **I**, a graph **G** is constructed where the connection

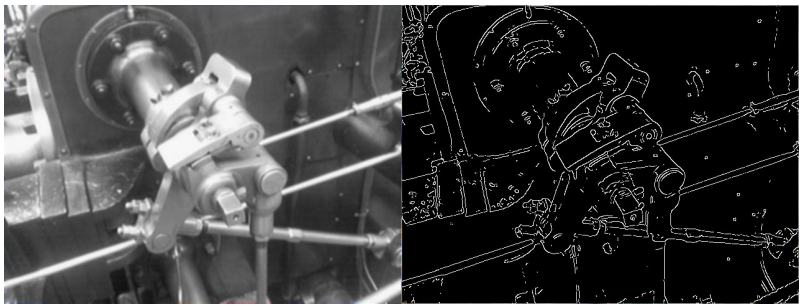

Figure 2.10: Example of result of the Canny Edge detector [vddcAUp].

i.e., the edge in the graph between two similar pixels (which will be represented by two vertexes) will have a high weight value and the connection between two very distinct pixels will have low weight, i.e., **G** = (**V**,**E**) where **V** is the set of vertexes and E the set of edges and **A** and **B** two disjoint sets of **V**, we have the cut function:

$$cut(A, B) = \sum_{a \in A,\, b \in B} w(a, b) \tag{2.7}$$

Because similar pixels are connected with high weight edges and the distinct ones are connected by low weight edges, by minimizing the *cut* function, the partitions **A** and **B** are created and will represent two segments in the image.

Graph-based segmentation algorithms usually find global optimal solutions. As expected, these algorithms are computationally expensive.

2.4.5 Discussion

The segmentation methods we have presented in this section are all very similar when it comes to complexity, with the exception of graph-based segmentation, a complex one. While region-based and graph-based methods can cause over segmentation, edge-based can create disjoint edges and thresholding depends on the intensity distribution.

Despite all that, these algorithms can be tailored and combined together to achieve better results in medical image segmentation. But if generic implementations are used, then the boundaries obtained will not necessarily correspond to the real boundaries we aim to find out.

2.5 Common Segmentation Algorithms for Medical Imaging

Medical images might pose specific needs and challenges when comparing to other type of images. Above all, we need to understand which type of algorithms are more helpful for our work and which ones are less helpful.

On that basis, and despite some generic segmentation algorithms have already been presented, we focus now on some segmentation algorithms commonly used in medical imaging. Namely, we draw attention to the *active contour model*, the *active shape mode* and

the *atlas-based segmentation*. We expect that other methods with similar characteristics will deliver similar efficiency and quality.

2.5.1 Active Contour Model

Active contour model, or snakes, proposed by Kass *et al.* [WT88], consists of a deformable geometric spline which is influenced by image forces and other constraint forces. These forces pull the spline towards object contours. A snake can be defined parametrically as

$$\mathbf{v} = \mathbf{v}(s) = (x(s), y(s)), s \in [0, 1]. \tag{2.8}$$

The matching between the snake and the image is made by means of energy minimization. The total energy of the snake can be written as

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))ds = \int_0^1 (E_{internal}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s)))ds \tag{2.9}$$

where $E_{internal}$ represents the internal energy, E_{image} represents the image forces and E_{con} represents the external constraint forces. The internal energy can be written as

$$E_{int} = (\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2)/2. \tag{2.10}$$

The first order term represents the stretching force of the snake and the second order term represents its bending force. Large values of $\alpha(s)$ will increase the internal energy of the snake the more it stretches, if the value of $\alpha(s)$ is small, the energy function will be insensitive to the amount of stretch. Simillarly, large values of $\beta(s)$ will increase the internal energy of the snake the curves it has, if the value of $\beta(s)$ is small, the energy function will be insensitive to curves in the snake.

The image energy can be written as

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{2.11}$$

Adjusting the weights will determine which features will be considered by the snake. E_{line} is the intensity of the image, which can represented by $E_{line} = I(x, y)$. Adjusting the sign of w_{line} will make the line attracted to dark lines of light lines. Edges in the image can be found by following an energy function which makes the snake attract towards contours with large image gradients. E_{edge} can be represented by $E_{edge} = -|\nabla I(x, y)|^2$. The constraint energy E_{con} can be used to interactively guide the snake.

Jia Liang *et al.* [LDW08] used a more sophisticated implementation of the snakes model, called Gradient Vector Flow (GVF), in the work of segmentation of the left ventricle. GVF allows the snake to start at a position far from the object and GVF can handle

broken edges and subjective contours. While the original snake model cannot converge well to concave features, GVF also solves this problem.

The values of α and β of the internal energy are chosen subjectively. Gao *et al.* [JGK98] proposed a solution to define these values in the extraction of the liver boundary. After a preprocessing phase, which include steps like removing the noise of the original images, the liver boundary is refined by snake algorithm. The value of α is proportional to the distance of adjacent control points, and β is inversely proportional to the curvature value.

2.5.2 Level set methods

Level sets were firstly introduced by Osher and Sethian [OS88] and have been widely used in medical image segmentation. This is mainly due to their ability to track the boundaries of the biological structures of interest. It is worth to point out that *active contours* can be implemented using *level sets*.

The *level sets* method creates a contour $\Gamma(t)$, which evolves towards the boundaries of the desired object. It is defined as the zero levo set of and embedding function $\phi(x, y, t) : R^3 \rightarrow R$ as given in equation 2.12.

$$\Gamma(t) = \{(x, y) | \phi(x, y, t) = 0\} \tag{2.12}$$

Using the contour defined in equation 2.12, we can define two domains: $\Omega^+ = \{(x, y) | \phi(x, y, t) > 0\}$, which is the interior of the contour, and $\Omega^- = \{(x, y) | \phi(x, y, t) < 0\}$, the exterior.

The level set function $\phi(x, y, t)$ can be defined using the euclidean distance function. Equation 2.13 shows a possible definition. The distance $d(x, y, t)$ is defined as the length of the shortest path from point (x, y) to the desired contour.

$$\phi(x, y, t) = \begin{cases} -d(x, y) & : (x, y) \in \Omega^- \\ 0 & : (x, y) \in \Gamma(t) \\ d(x, y) & : (x, y) \in \Omega^+ \end{cases} \tag{2.13}$$

With the level set function defined, we can now introduce the outer unit normal vector $\overrightarrow{N}$ (equation 2.14)

$$\overrightarrow{N} = -\frac{\delta\phi}{|\delta\phi|} \tag{2.14}$$

The contour will evolve in time due to several artificial forces, which makes the contour move in the normal direction.

2.5.2.1 Fast Marching

The *fast marching* is an example of a level set method. It is a numerical method for solving boundary value problems of the Eikonal equation [Tsi95] (see equation 2.15), which is commonly used in problems of wave propagation. A boundary value problem usually

describes the evolution of a closed curve as a function of time T with speed $F(x)$ in the normal direction at a point x on the curve. Concerning the time function in the equation, the user can specify at which time the algorithm stops (this is used to save computational time). The user must also provide at least one seed point with an initial value.

$$F(x)|\nabla T(x)| = 1 \tag{2.15}$$

The speed function values are usually obtained from a speed image. The main purpose of this image is to use it as an indicator of where the level sets function should grow faster and where it should grow slower. Suppose we have an image which intensities are between the values 0.0 and 1.0. Ideally the speed value should be 1.0 in homogeneous regions of anatomical structures and the values should decay rapidly to 0.0 around the edges of structures. There are several methods to produce a speed image. For instance, we can use the sigmoid filter. The sigmoid filter gives us more control over the resulting speed image because of its parameters α and β.

The sigmoid filter is based on the sigmoid intensity transformation (see equation 2.16). This filter is commonly used as an intensity transform: It maps a specific range of intensity values into a new intensity. Because the sigmoid is a non linear function, it is possible to focus attention on a particular set of values, while progressively attenuating the values outside that range. The parameter α in equation 2.16 defines the width of the input intensity range and β defines the intensity around which the range is centered (see Figure 2.11).

$$I' = (Max - Min) * \frac{1}{1 + \exp(-(\frac{I-\beta}{\alpha}))} + Min \tag{2.16}$$

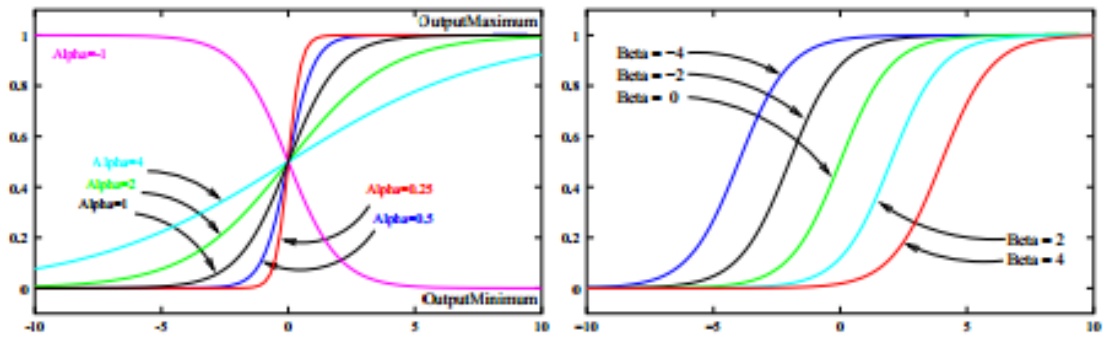

Figure 2.11: Influence of the alpha and beta parameters in the sigmoid function [HJJtISC13].

Since we want to create a speed image, the parameters Min and Max are set to 0.0 and 1.0 respectively. Alpha (α) and Beta(β) can be set by the user.

2.5.3 Active Shape Model

Organs and other biological structures in medical images have a tendency towards some average shape. This tendency can be very useful when proposing algorithms.

Cootes *el al.* [TFCJ94] proposed to use *Active Shape Models* in order to locate structures in medical images. This method requires a training phase in order to take advantage of the average shape of a biological structure.

Once the training phase is finished, the algorithm applies *Principal Component Analysis* (PCA) to identify the relevant shapes. After this step, it is possible to create an arbitrary shape by the linear combination of these identified shapes. The outcome can then be deformed by changing the coefficients used in the linear combination. A similar method was used by Gleason *et al.* [SSGM02] to detect poly cystic kidney disease in CT images. The main disadvantage of this algorithm is to require of a lot of training samples.

2.5.4 Atlas-based Segmentation

In some applications, a clinical expert can manually label several images and so segmenting unseen images is a matter of extrapolating from these manually labeled training images. Methods like this are referred to as atlas-based segmentation methods.

The atlas construction can be based on a single object. The training images are used separately to construct the atlas, but this poses several problems. Since the selection of a sample is arbitrary, it means that the selected object may not be a typical one. Furthermore, the fact that only one sample is used, means that no information about the variability of the object is available so it is impossible to determine if the resulting shape is acceptable. To solve these problems, *probabilistic atlas* has been proposed [PBM03]. This method uses a spatial distribution of probabilities to determine if a pixel belongs or not to an organ. To obtain the probability distribution, multiple samples are used. Usually it is required the use of image registration in order to align the atlas images (the samples) to a new image. Similarly to the active shape model (which can be considered as a probabilistic atlas), the disadvantage of probabilistic atlas is that it requires a large set of training samples in order to work properly.

2.5.5 Discussion

The main disadvantage of atlas-based and active shape model methods is that they both require a large set of training samples. Active contour model is a good choice if we are dealing with soft tissues [SSGM02]. Furthermore, it also provides some interactivity if energy constraint parameters are to be changed. Atlas-based segmentation and active shape model are usually less sensitive to noise in comparison to active contour model or other methods that don't require training. Because of this, a preprocessing step may be necessary before applying an active contour method. Considering our problem, active contour models and region based algorithms are the most appropriate types of segmentation since we don't have enough data to use as training data. Region based algorithms

and active contour models also provide a greater level of interactivity since they don't require as much time to execute as atlas-based segmentation or other methods that require training.

2.6 Volume Rendering

Volume rendering is a rendering technique aiming to display a 3D sampled data set entirely, therefore laying emphasis on the overall as opposed to the detail. Usually, the data set is a group of 2D slice images acquired by a CT or MRI scanner. These slices are taken following a regular pattern (e.g. one slice every milimeter) thus allowing us to reconstruct the complete volume.

In the following we present the most important techniques used in volume rendering.

2.6.1 Ray Casting

Ray Casting was introduced in 1988 by Marc Levoy [Lev88] and, as we can see in Figure 2.12, it generates high-quality images. In medical imaging, we can use ray casting in the field of virtual endoscopy [HO00] for example.

In this technique, a ray is generated for each desired image pixel. Originating from the camera position, multiple rays are traced through the data volume. Along the part of the ray that lies within the volume, equidistant samples are selected. If the volume is not aligned with the ray, sampling points will be located between voxels and it will be necessary to interpolate the values of the samples from its surrounding voxels. After all sampling points have been shaded, then the final color for the pixel currently being processed is computed. To do so, the sampling points are combined along the ray. In general, this composition starts with the samples farthest from the viewer and ends with the one nearest to him, in a blending fashion of layers. Masked parts of the volume do not affect the resulting pixel.

Ray casting used to be demanding technique, in particularly priori to recent advances in graphic cards technology. Several techniques have been proposed in the past to speed up the technique [RAW94, RGY98].

It is worth to point out that ray casting nowadays should be parallelized, that is, instead of casting rays in a sequential fashion, multiple rays should be casted at the same time, in parallel. Thanks to this, ray casting is now considered as an interactive technique.

2.6.2 Shear Warp

The *shear warp* approach was popularized by Philippe Lacroute and Marc Levoy in 1994 [LL94]. In this technique, the voxels in the computer's memory are rearranged with the purpose of allowing an optimized ray traversal (shearing of the volume data set). It is a fact that the viewing position is independent of the way data was initially stored in the

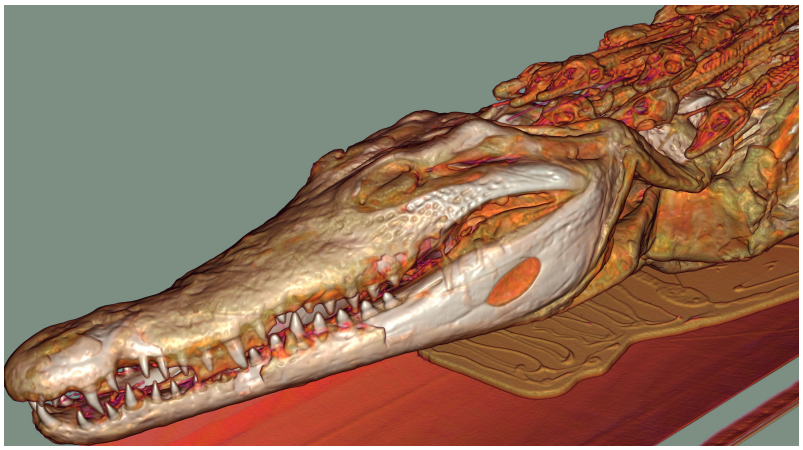

Figure 2.12: Volume ray casting applied to CT data of a crocodile. The image was rendered by Fovia's High Definition Volume Rendering engine [Wikb].

computer's memory. Because the pixels have to be rearranged, the image is distorted and a final correction step is necessary (warping the image).

This technique is relatively fast in software at the cost of less accurate sampling and worse image quality when compared to ray casting. Nevertheless, the reasonable trade-off between speed and quality made it an interesting choice for implementation in special hardware architectures.

2.6.3 Splatting

The *splatting* technique was proposed by Lee Westover in 1991 [WW91]. This technique is meant to trade quality for speed. With splatting, volume rendering displays are composed by accumulating splatted data voxels on the viewing surface in back to front order. The degree of influence of each voxel is defined by a reconstruction filter.

The all process resembles the result of throwing snow balls on a wall.

2.6.4 Maximum Intensity Projection

Unlike the accumulation process of other volume rendering techniques, the maximum intensity projection method only picks out and projects the voxels with maximum intensity that fall in the way of parallel rays traced, from the viewpoint to the plane projection.

This technique is a widely used technique in medical visualization. For example, it is the rendering technique of choice when it comes to the investigation of vascular structures [CWQ+06]. However, we must draw attention to the quality of images we can obtain. They are not the most accurate ones if compared to the ones generated by other methods. Being the area of medical visualization of particular concerned as far as quality imaging is concerned, we wonder if this *status quo* should be changed for the sake of the well-being of patients.

2.7 Surface Rendering

While volume rendering is concerned with the display of the data sets in their entirety, surface-based rendering techniques focus on depicting surfaces within the volume. Hence, it is a useful technique to emphasize material transitions within the volume, as depicted in Figure 2.13.

In the following we present the common surface-based rendering techniques. In particular, iso-surface extraction and segmentation masks.

As a historical note, we should make reference to the technique called *contour approaches*. They were widely before the introduction of iso-surfacing techniques but not anymore. Basically, in a semi-automatic or manual way, the 2D contours (points of same value but in a 2D slice) of the structures of interest were identified on single slices. Then geometric algorithms were used to connect the contours of adjacent slices in order to form a closed surface [KSSe00].

Figure 2.13: An example of a surface-rendered human skull [Ren].

2.7.1 Iso-surface extraction

Iso-surface extraction aims to represent all points within a volume with the same property or value. The Marching Cubes (MC) technique is the best established iso-surface extraction approach. It was introduced by W.E. Lorensen and H.E. Cline in 1987 [LC87] and since then it has been broadly used in various domains.

With the rectangular volume data defined as a set of voxels, the algorithm starts by examining each of these voxels, one by one. Each voxel has eight neighbor locations, thus forming an imaginary cube. Then the algorithm determines the polygon or polygons needed to represent part of the isosurface that passes through the cube. The polygons are then used to produce the desired surface.

There is a total of 256 possible polygon configurations (see Figure 2.14). To determine the polygon(s), the 8 scalar values (neighbors) are compared to a pre-defined iso-value: If the scalar value is greater, then the appropriate bit is set to one; if it is lower, the bit is set to zero. The final value after all 8 scalars were checked leads to a particular case of a lookup table. The vertex of the generated polygons is placed in the appropriate position

along the cube's edge by linearly interpolating the two scalar values that are connected by that edge. This time consuming process motivated researchers to come up with several acceleration strategies [IK94, LA98]. Moreover, there were some issues related to the accuracy of the initial solution, which led to further research work [LB03].

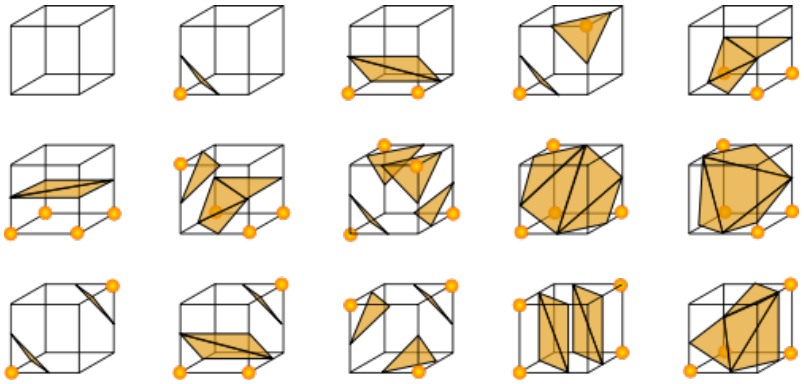

Figure 2.14: The original 15 MC configurations, considering both rotational and reflective symmetry. These configurations led to further research to address issues of accuracy so they have been extended [Wika].

2.7.2 Segmentation Masks

This technique simply consists of displaying regions of the data volume which have been identified by segmentation techniques. Therefore it is just a matter of representing graphically the points already identified. Unlike iso-surface extraction, there is no need to process the all volume data. But in this surface-rendering method, special smoothing techniques have to be employed because segmentation masks usually originate staircase artifacts.

2.8 Time-Varying Visualization

Most of the applications in 3D visualization use a single data set that corresponds to a single instant of time of its domain of interest. Such data set is of dimension $width \times height \times depth$.

However, when referring to time-varying visualization, as it happens in medical visualisation, we have to deal with an extra dimension: The time. As a consequence, the data set dimension is now $width \times height \times depth \times time$.

The main main challenge in this situation is to update the data structures, from one time frame to the next one but in real-time, so users can visualize data both in space and time. Most of the research in time-varying data visualization deals with massive data volume, which is very common in engineering problems

Philip Sutton and Charles D. Hansen proposed the use of Temporal Branch-on-Need Trees (T-BON [SH99b]. In the preprocessing step, a branch-on-need octree for each time step in the data, the tree infrastructure is stored to disk. Before any isovalue queries, the tree infrastructure is read from disk and recreated in main memory. After this, queries are accepted in the form *(timestep, isoval)*. The values of the nodes corresponding to *timestep* are first read. If the values span *isoval* (i.e. greater than the *isoval*), the children of these nodes are read. If a lead node spans *isoval*, the disk block containing the data for the cells in that leaf is added to a list. Once the necessary portions of the tree have been brought into memory, all blocks in the list are read from disk. This was proposed in 1999, when the memory capacity and computer power was much lower then it is today, which means that this algorithm can be tailored to read less data from the disk.

Han-Wai Shen *et al.* proposed a volume rendering technique which uses a time-space partitioning (TSP) tree [SH99a]. A TSP tree is similar to a regular octree. The difference is that the TSP tree node contains both spatial and temporal information about the underlying data in the subvolume, while a regular octree only contains the spatial information.

Zhe Fang *et al.* [FMHC07] suggested that the goal of medical diagnosis is not to observe dynamic in real time as much as to differentiate healthy from diseased tissue. Hence it is not so important to provide interactive selection of arbitrary time-slices, but rather the comprehensive evaluation of the time-behaviour of each tissue sample and its classification into different tissue types and tissue behaviours.

Zhe Fang *et al.* [FMHC07] also proposed a way of rendering time-varying medical images by analyzing the time activity curve (TAC) of the voxels within a given volume. The TAC of each voxel is compared to a template TAC, which can be obtained by interactively probing the underlying data by specifying an arbitrary voxel via its spatial location, or it can also be generated from the result of some previous analysis, such as finding the average of a region of interest. The comparison between the template TAC and the other TACs results in a scalar value (i.e. each voxel associated with one scalar distance). The volume can then be rendered using classical (scalar) volume rendering algorithms.

Ma *et al.* have provided a survey of different time-varying data visualization techniques used in large-scale data visualization [ML05].

S. Silva *et al.* [SSM13] proposed a real-time offline segmentation technique for the vocal tract. The segmentation starts with the definition of a region of interest (ROI) over just one of the image frames. This is possible because there is spatial coherence among images, which means the ROI can be replicated. A ROI is defined for each of the anatomical structures we want to identify. Since we are dealing with real-time MRI, the segmentation technique used must be computationally fast. The authors applied region growing (which is a fairly simple algorithm) to each of the ROIs to obtain the desired quick segmentation. The ROIs previously defined also help the region growing algorithm to not spread beyond the desired regions, which means the algorithm will be faster than a standard region growing (without the ROIs).

Finally, it is worth mentioning that most of the research in type-varying visualization is more common in areas other than medicine. For instance, in chemistry, mechanical engineering and other areas with various types of simulations. In medical visualization, the data sets tend to be smaller when it comes to time, because patients usually can not be exposed to PET or CT scanners for a long period of time.

2.9 Development Toolkits for Medical Visualization

In the following we present some of the most popular toolkits used to develop medical visualization applications. This section ends with a brief discussion and comparison between such toolkits.

2.9.1 ITK – Insight Segmentation and Registration Toolkit

The Insight Segmentation and Registration Toolkit [ITK] is an open-source toolkit, developed in C++ by several institutions and funded by the National Library of Medicine. It was developed mainly for image processing, segmentation and registration. ITK provides a large number of segmentation algorithms such as thresholding, edge detection, region growing, active contour and watershed. Image visualization, manipulation or analysis features are not provided by ITK and so other libraries have to be used to cover these aspects. Also, ITK does not support 3D image rendering. But because of this, ITK provides interfaces to the Visualization Toolkit (VTK) [VTK], which can handle 3D image rendering.

2.9.2 MITK – Medical Imaging Interaction Toolkit

The Medical Imaging Interaction Toolkit (MITK) [MITb], developed at he German Cancer Research Center, combines ITK and VTK, thus easing the effort required to create applications for medical image analysis. Notice that, as it was previously stated in Section 2.9.1, ITK requires additional visualization and interaction features. The MITK community provides good documentation, as well as multiple tutorials and installation instructions to combine with Qt[Qt], a convenient tool to build graphical interfaces with MITK. Since MITK is derived from ITK and VTK, most of the ITK's and VTK's documentation can be used as well.

2.9.3 IGSTK – Image-Guided Surgery Toolkit

The Image-Guided Surgery Toolkit (IGSTK) [IGS], developed by the Insight Software Consortium, also combines ITK and VTK. As for the user interface, it uses the Fast Light Toolkit (FLTK) [FLT]. IGSTK provides a set of modules which allow the representation of objects such as needles (see Figure 2.15). The IGSTK community also provides good documentation.

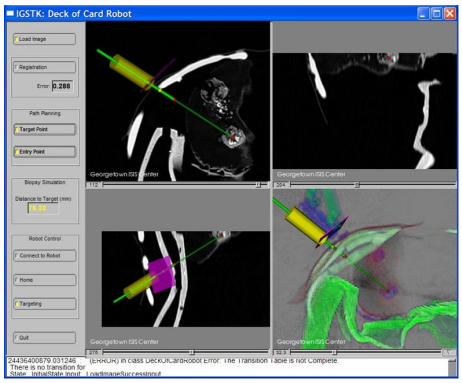

Figure 2.15: User interface of an application developed using IGSTK for robot-assisted needle placement in biopsies [AEC07].

2.9.4 User interface Development

As we can conclude from the sections above, it is necessary to use an additional library to build the user interface. The libraries wxWidgets [wxW], Fast Light Toolkit (TLTK) [FLT] and KWWidgets[KWW] are examples of open-source libraries which can be successfully used to build user interface for medical imaging applications. Even though not being open-source, Qt [Qt] provides a free version which can be used alongside ITK and VTK.

2.9.5 Application Builders

Multiple application builders and/or Problem Solving Environments (PSEs) have also been developed in the past. Such tools allow users to interactively design solutions to a certain class of problems, by providing a graphical interface where it is possible to build a processing pipeline. This is achieved by adding different modules and establishing a network of connections among them.

In this section we present some of the application builders that can support medical image analysis, processing and visualization.

DeVIDE. The Delft Visualization and Image processing Development Environments (DeVIDE) [DeV] provides several visualization and image processing features by integrating several libraries, such as VTK [VTK] and ITK [ITK]. It provides a higher-level function to ease their usage.

It is also possible to add new features to DeVIDE, by including new modules or adding code to existing ones in order to modify their behavior.

MeVisLab. MeVisLab[MeV] was developed by Mevis Medical Solutions AG and Fraunhofer MEVIS, Bremen, Germany, and integrates the libraries VTK and ITK. MevisLab allows the usage of these two libraries without the need of knowing a programming language. It is also possible to create networks where multiple modules are combined to achieve the desirable result.

We notice that MeVisLab provides a helpful documentation tutorial.

SCIRun. SCIRun [SCI] is a problem solving environment developed at the Scientific Computing and Imaging Institute, University of Utah. Similarly to the application builders already mentioned, a user selects software modules and connected them in a visual programing environment in order to create a high level workflow. The modules include several ITK functionalities for image segmentation and registration, as well as it provides MATLAB integration.

It is worth mentioning that some of the applications developed at University of Utah have evolved from SciRun's features, namely, Seg3D [Seg] and ImageVis3D [Ima].

2.9.6 Discussion

ITK is a great tool for image processing and segmentation, even though it lacks visualization features. This can be fixed by also using VTK, MITK and IGSTK. We therefore can put together image processing methods, visualization and interface design, all in a single toolkit. With such framework, the developer's effort is minimized. IGSTK seems to be a good choice when it comes to image-guided surgery.

Regarding the application builders, overall they are very similar to each other and have great potential. The main difference between them is the availability of documentation. Both SCIRun and DeVIDE lack developer documentation. On the other hand, MeVisLab provides both user and developer documentation. DeVIDE also lacks the ability of building user interface for the software modules.

3

Lumbar Segmentation

The lumbar region of the human body is one of the places where epidural anaesthesia is performed (see Figure 3.1). There are several structures in this region that we believe are relevant during epidural anaesthesia, namely, the spinal canal, the vertebral body, the intervertebral disc, the abdominal aorta and the inferior vena cava.

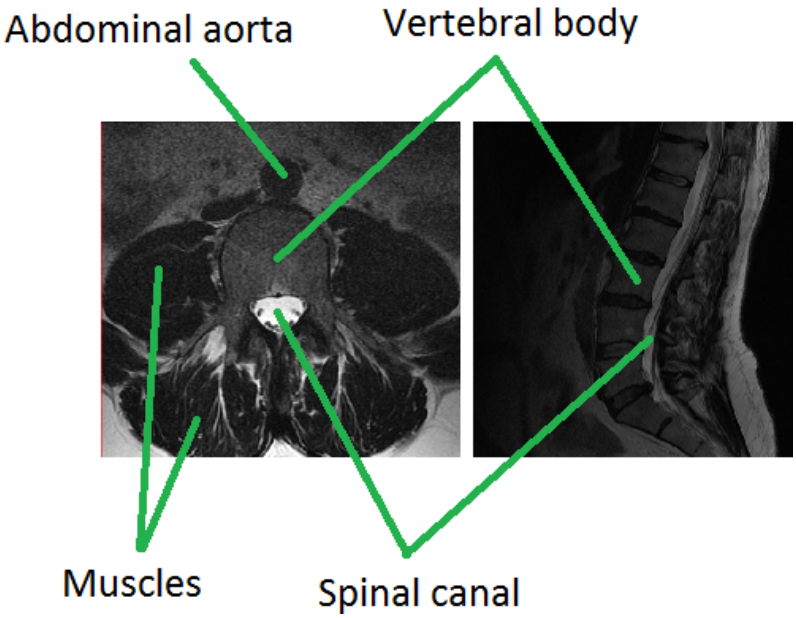

Figure 3.1: Lumbar region of a human body.

In this chapter we discuss the multiple stages of the pipeline we have set up to segment images of the lumbar region. First, we introduce the pipeline, the data sets and the toolkits we have used. Then we discuss the various stages and associated results, ranging

from image pre-processing to the segmentation itself. We evaluate results for the case of spinal canal, abdominal aorta, vertebral body and back muscles. This chapters ends with major conclusions we can draw from the tests carried out.

3.1 Segmentation Pipeline

The segmentation pipeline we set up for the purpose of image segmentation is depicted in Figure 3.2. Hereafter we just consider MRI images.

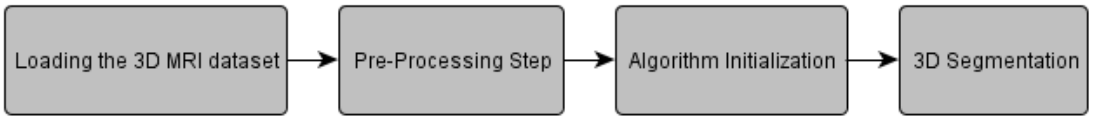

Figure 3.2: Segmentation pipeline for a 3D MRI dataset.

After loading the MRI dataset to be analysed, the process starts with pre-processing of the images, having in mind the reduction of noise and the enhancement of the features of interest.

The next step is the related to initilization of algorithms. Actually, some segmentation algorithms require the use of additional filters to work properly.

The following step is the proper segmentation. At this point, we have used two types of algorithms, namely, the region growing and the level sets algorithm.

We should point out that there is usually an additional step at the end, where the obtained segmentation can be edited. However, as an interactive one, we have decided not to address it in this chapter.

Table 3.1 presents information about the datasets of the lumbar region that have been used in these experiments. They were taken from the lumbosacral region. The back muscles in these images are represented with low intensity values (notice that muscles can also be represented with high intensity values).

Name	Size	Inclination
MRIX LUMBAR	512 x 512 x 26	Axial
R66y (901)	321 x 334 x 20	Axial
R66y (801)	321 x 334 x 20	Axial
MRIX LUMBAR	512 x 512 x 12	Sagittal
LOMBIX	660 x 644 x 25	Sagittal
LOMBIX	770 x 770 x 12	Sagittal
A10y	435 x 432 x 11	Sagittal

Table 3.1: 3D MRI data sets of the lumbar region to be segmented. These images are all in the DICOM format.[2]

[2]The datasets R66y and A10y were obtained from the portuguese hospital "Santa Maria". MRIX LUMBAR

As far as toolkits to help carrying out the experiments, and following the discussion presented in Section 2.9, we have decided to use the Medical Imaging Interaction Toolkit (MITK) [MITb]. As mentioned, MITK is a free open-source software system for development of interactive medical image processing software. We will be also using the Insight Toolkit (ITK) [ITK] and the Visualization Toolkit (VTK) [VTK], since MITK combines both ITK and VTK as an application framework.

3.2 Image Pre-processing

As previously stated, the image pre-processing stage is used mainly to reduce noise and enhance image features. To denoise the image, we have used two main techniques: Median filter and *anisotropic diffusion* filter (as edge preserving smoothing). Both techniques are described in Section 2.3. The images presented in this section are from the dataset *MRIX LUMBAR*[Osi] Axial.

3.2.1 Comparison

We have tested both methods with different parameters and values. The *signal-to-noise* ratio was used to compare both methods, although it doesn't take into account edge preserving characteristics. The equation 3.1 sets the computation of the *sinal-to-noise* ratio, where μ is the mean of the intensities in the image, and σ the standard deviation.

$$SNR = \frac{\mu}{\sigma} \tag{3.1}$$

Table 3.2 shows the results obtained for the median filter and for the *anisotropic diffusion* filter. The *signal-to-noise* ratios obtained with the median filter are lower than those obtained with *anisotropic diffusion* filter, as illustrated in Figure 3.3. The median filter was used with the radius value of 1 to avoid damaging structure boundaries too much. The *anisotropic diffusion* filter was used with conductance parameter of 1.0.

In this particular image, the fact that *anisotropic diffusion* filter doesn't change the mean of the image, helps the *signal-to-ratio* ratio to grow faster than the one obtained with the median filter.

Note that if we kept increasing the number of iterations, the *signal-to-noise* ratio obtained with the median filter would eventually surpass the *signal-to-noise* ratio obtained with the *anisotropic diffusion*. This is because median filter is a linear smoothing filter and doesn't take into account the boundaries of the structures in the figure, even though we can obtain good edge preserving results in some cases.

Figure 3.4 show the images obtained for each of the filters using 35 iterations. The boundaries in the image obtained by the median filter were damaged and some have disappeared. Hence, we conclude that the median filter works better when there is less

and LOMBIX were obtained from OsiriX [Osi].

	Median filter		*Anisotropic diffusion* filter	
Iterations	μ	SNR	μ	SNR
5	162.978	1.144	168.318	1.155
10	161.636	1.149	168.318	1.188
15	160.791	1.153	168.318	1.209
20	160.153	1.156	168.318	1.225
25	159.628	1.158	168.318	1.238
30	159.189	1.160	168.318	1.250
35	158.820	1.162	168.318	1.260

Table 3.2: Mean and SNR results for both median and *anisotropic diffusion* filters.

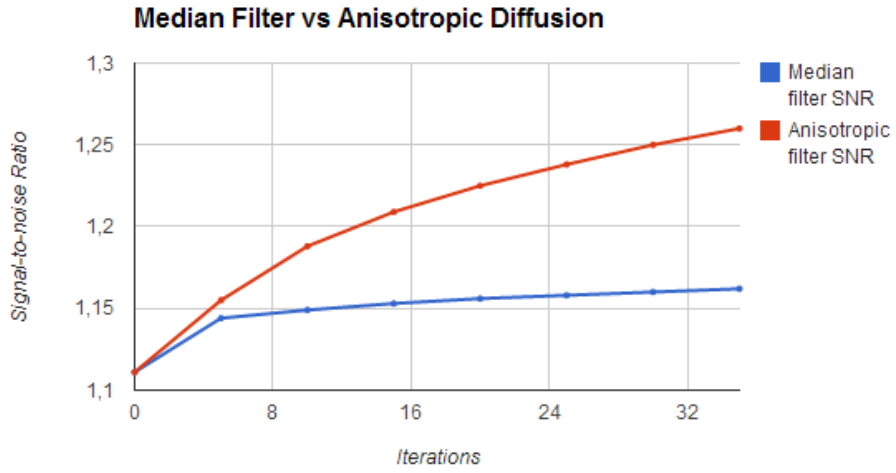

Figure 3.3: Median filter $vs.$ *anisotropic diffusion* filter.

noise in the image. If so, less iterations are needed to smooth the image, which means that there is a greater chance to preserve the boundaries of the structures.

Anisotropic diffusion usually takes longer to execute than the median filter. It may be an issue if there are time constraints at the segmentation stage.

On the basis on the results gathered, we are in favour of using *anisotropic diffusion* in our experiments.

3.3 Initialization

Some segmentation algorithms, namely level sets, require additional initialization other than applying smoothing methods. The level sets method usually requires a speed image to be provided at the beginning. The intensity values in a speed image are interpreted as the values of a speed function used by the level sets (see Section 2.5.2.1).

The speed image must provide high intensity values in homogeneous areas and low intensity values at the edges of the anatomical structures. This way the level sets method

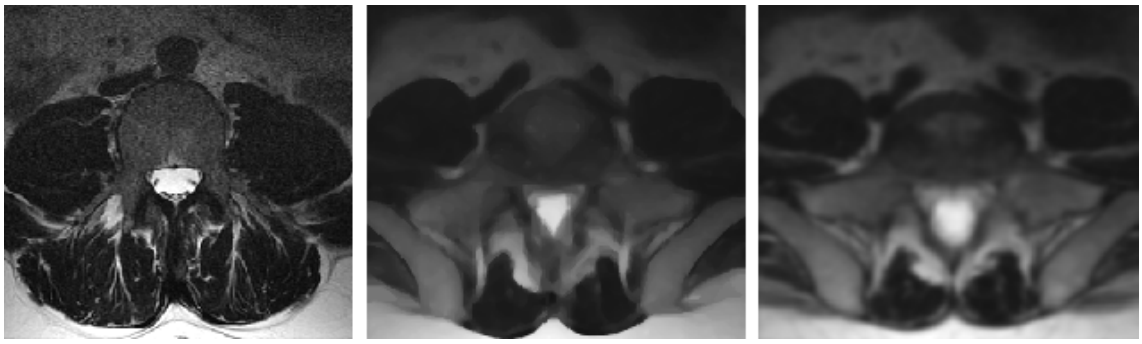

Figure 3.4: Original image (left). Use of median filter with 35 iterations and radius 1 (center), and *anisotropic diffusion* filter with 35 iterations and 1.0 as conductance parameter (right).

can easily stop when the values start dropping to low intensity values.

3.3.1 Edge Detection Filters

The task ahead of obtaing the speed image itself is to apply edge detection filters. We have used two filters for that purpose: the *gradient magnitude* filter and the *canny edge detector*.

3.3.1.1 Gradient Magnitude Filter

The gradient magnitude filter can be used as an edge detecting filter. Unless we use the gradient magnitude filter with smoothing, no additional parameters are required. Since we already went through a smoothing process, we have decided to use the gradient magnitude filter with no additional smoothing.

Figure 3.5 shows the resulting image after applying the gradient magnitude filter to the smoothed image of the lumbar region. But if no smoothing is applied, then the edges become mixed with the noise, as depicted in Figure 3.6.

3.3.1.2 Canny Edge Detector Filter

Another algorithm we have used to detect the edges was the canny edge detector. This method requires both a low threshold value and a high threshold value. The values must be less than 1 and negatives values will not work, which means that $low < high < 1$.

As depicted in Figure 3.7, we have tested multiple threshold values. If both values are too close to 0, more unwanted edges are shown in the resulting image. On the other hand, if the values become closer to 1, less edges are shown in the final image. This outcome is true for most of the images. The best threshold values for this particular image under analysis are 0 and 0.1, respectively for low and high threshold values.

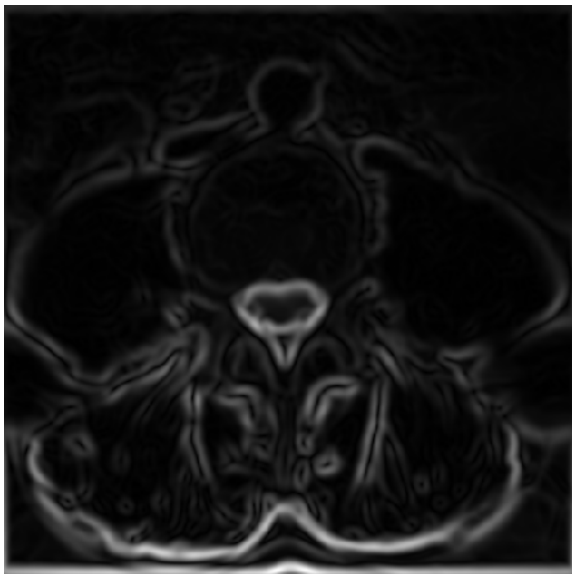

Figure 3.5: Gradient magnitude filter applied after smoothing the image.

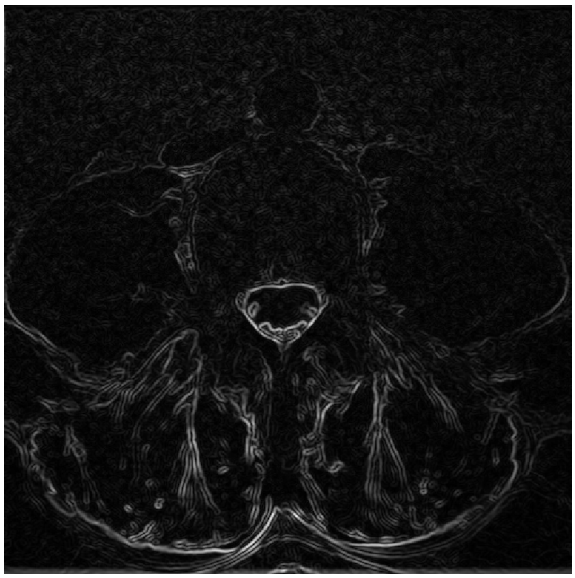

Figure 3.6: Gradient magnitude applied but without smoothing the image first.

3.3.1.3 Comparison

Regarding the execution time, it is a non-issue for both methods. Indeed, they both execute very fast and so execution time is negligible.

The canny edge detector filter fails to completely connect some of the edges, while the gradient magnitude does this very well.

Another advantage of the gradient magnitude is that it represents edges in different intensities. This is good because the unwanted edges are usually represented with a lower intensity, sometimes very similar to the one used in the homogeneous areas. Furthermore, the gradient magnitude filter doesn't require the setting of any parameters,

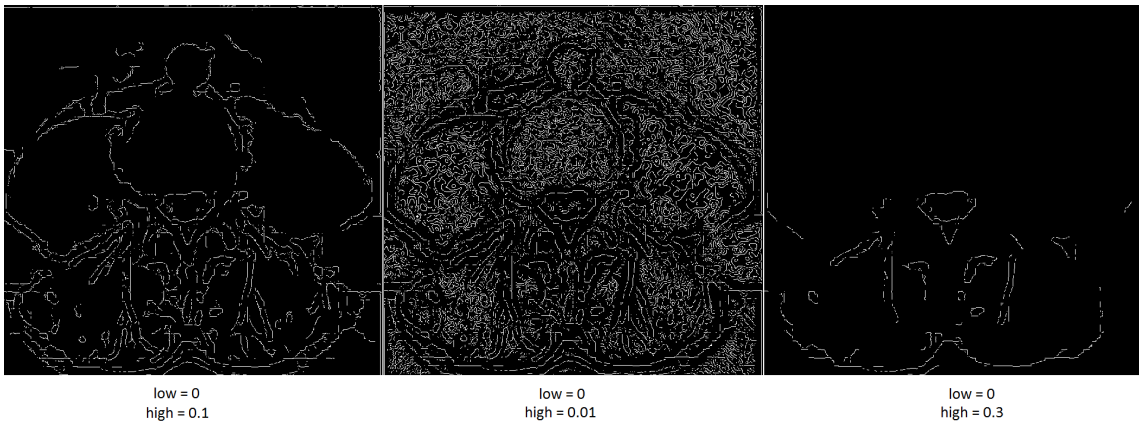

Figure 3.7: Canny edge detector results.

which simplifies the whole process.

3.3.2 Speed Image

As mentioned, a speed image is an image which values are interpreted as values of a speed function. Its main purpose is to be used as an indicator of where the level sets function should grow faster and where it should grow slower. Suppose we have an image which intensities are between the values 0.0 and 1.0. Ideally, the speed value should be 1.0 in homogeneous regions of anatomical structures and the values should decay rapidly to 0.0 around the edges of structures.

There are several methods to produce a speed image. For instance, we can use a sigmoid filter and, as an alternative, a reciprocal filter. We have used both for that purpose.

3.3.2.1 Sigmoid Filter

The sigmoid filter maps a specific range of intensity values into a new intensity. Because the sigmoid is a non linear function, it is possible to focus on a particular set of values, while progressively attenuating the values outside that range.

We can approach the problem of creating the speed image in two different ways: we can focus attention on the values of edges or alternatively on the values of homogeneous areas. The parameter β is used for this purpose. To simplify the choice of the value, we have rescaled the intensities in the edge image to values between 0.0 and 1.0. The edge intensity values are closer to 1.0 and the intensity values in the homogeneous areas are closer to 0.0. Since we want the homogeneous areas to become brighter and the edges to become darker, but we have currently the opposite, we must provide a negative value for α in order to invert the intensities.

Figure 3.8, shows multiple images obtained with different β values and a negative α value. Figure 3.9 shows the difference between the use of a negative α and a positive one.

The last decision to be taken is the choice of the absolute value of the parameter α, that is, $|\alpha|$. The greater the $|\alpha|$, the greater variety of intensity values we have. If we're

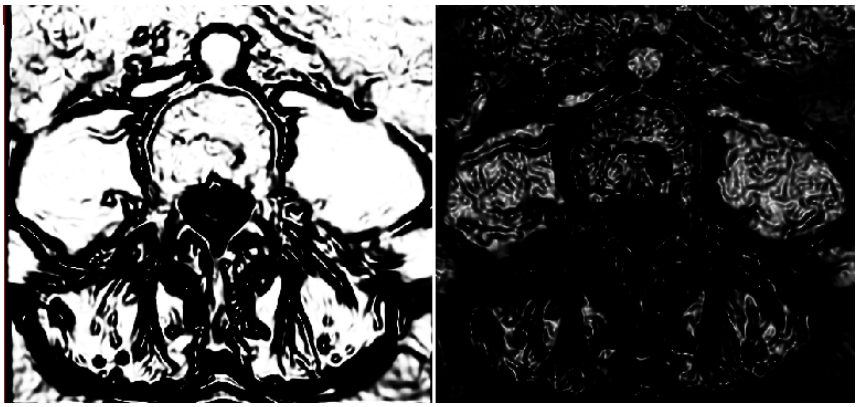

Figure 3.8: Left image was obtained with the parameters $\alpha = -1.0$ and $\beta = 10.0$. Right image was obtained with the parameters $\alpha = -1.0$ and $\beta = 0.0$.

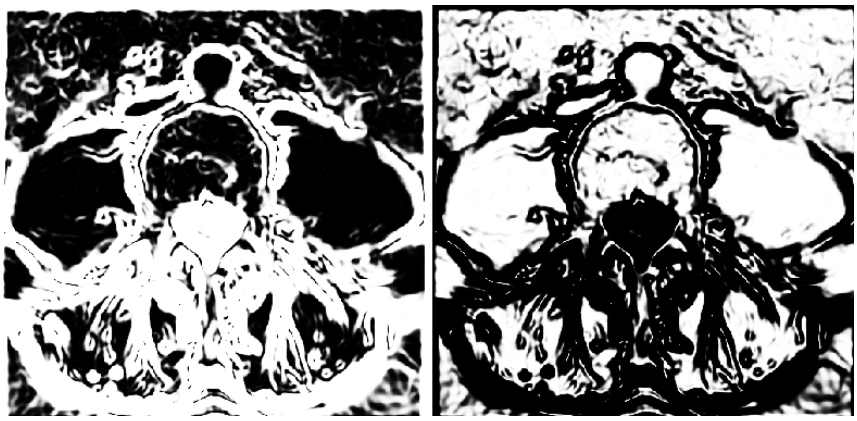

Figure 3.9: Left image was obtained with a positive α value and the righ image with a negative one.

aiming for a speed image with just the values 0.0 and 1.0, then $|\alpha|$ should be closer to 0.0.

Although a binary speed function (with two possible values only, 0 and 1) may be desirable, it is not effective in some cases. For example, some of the edges may not be well defined, and these are usually the cause for leakages during the segmentation stage. These edges are also represented with lower intensity values than the ones that are well defined. If we provide a small $|\alpha|$ (close to zero), the edges that are not well defined will completely disappear. On the other hand, if we provide a bigger $|\alpha|$, we can represent these edges with a lower intensity value (despite not being the desirable zero) and it slows down the speed function.

3.3.2.2 Reciprocal Filter

Since a speed image has low intensity values for the edges and high intensity values for the homogeneous areas but we want the opposite, we can apply a filter that changes

intensities I for their reciprocal values. This filter follows Equation 3.2.

$$I' = \frac{1}{1+I} \tag{3.2}$$

Figure 3.10 shows the resulting image after applying the reciprocal filter. The intensities are lower (darker) than the ones obtained by using the sigmoid filter. This means that the speed function will be slower if we use the image obtained by using the reciprocal filter.

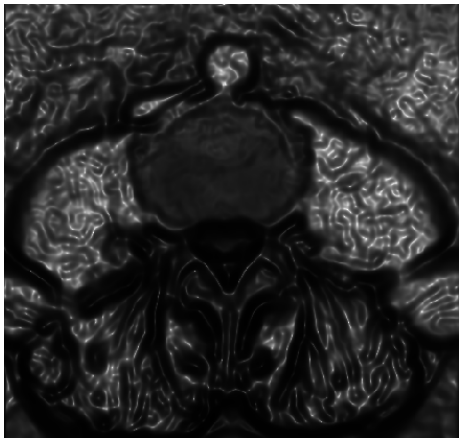

Figure 3.10: Resulting image after applying the reciprocal filter to the gradient magnitude image.

3.3.2.3 Comparison

We have applied both methods 5 times with different sized datasets and, even though the sigmoid filter always took longer to finish, their execution times are fairly similar considering that the segmentation and smoothing times are much superior (see Table 3.3). This result was expected because they both work by computing an equation for each pixel of the image. By changing the α and β parameters in the sigmoid filter, we have a greater control over the resulting speed image. We can even create similar images to those obtained using the reciprocal filter. On the other hand, the reciprocal filter doesn't need any user input to obtain a satisfactory speed image. Since none of the methods has a clear advantage over the other, both will be used at the segmentation stage.

Reciprocal filter (ms)	Sigmoid filter (ms)	Dataset size
125	203	512 x 512 x 26
46	79	320 x 320 x 36
375	687	512 x 512 x 76
31	63	166 x 195 x 25
1875	3313	512 x 512 x 355

Table 3.3: Reciprocal and sigmoid filters average execution times in ms.[3]

3.4 Segmentation

As it was mentioned at the beginning of this chapter, there are several structures of interest in the lumbar region. In this section we describe how each of these structures can be segmented, given pre-processed images obtained by applying one of the methods described in Section 2.3.1. It is important to mention that additional filters were applied to the images for a proper initialization of the segmentation.

As far as segmentation algorithms are concerned, the Insight Segmentation and Registration Toolkit (ITK) offers three main types of algorithms: Region growing, watersheds and level sets.

3.4.1 Spinal Canal

The spinal canal is one of the easiest structures to segment, mainly because its range of intensities is very different from the rest of the image. It is fair to mention that the data set we have used had a problem with its intensities. As can be seen in Figure 3.11, as we move upwards the darker the image becomes. This means that some undesirable regions will be in the threshold result, so this has to be taken into account when choosing the segmentation algorithm.

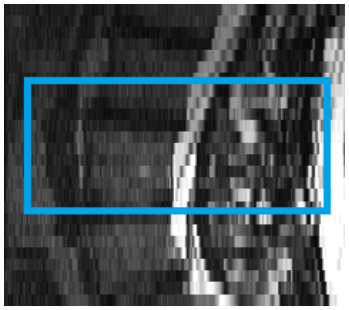

Figure 3.11: MRI with intensities going darker/brighter as you move upward/downward.

We have decided to start the experiments using a region growing algorithm. This is due to the fact that this algorithm is one of the simplest approach provided by ITK. There are multiple variations for this algorithm, namely, the *connected threshold*, the *neighbor connected* and the *confidence connected* [HJJtISC13].

The *connected threshold* is the basic one. A seed point is provided alongside a lower threshold value and an upper threshold value. The algorithm will include in the segmentation any pixel (or voxel) that is connected to the seed point and it has intensity between the lower and upper threshold values (see Equation 3.3).

$$I(X) \in [lower, upper] \tag{3.3}$$

[3]Times obtained using a computer with a 4 Core processor at 2.50GHz and 8192 MB of memory.

Figure 3.12 shows the results obtained with this variant of the region growing algorithm. The problem about the intensities in the image we have mentioned before makes the use of this algorithm very difficult. A wider range between the threshold values must be provided in order to include the whole spinal canal. Unfortunately this means that undesired parts of the image will be included in the segmentation. Notice that the inclusion criteria in this method is trivial.

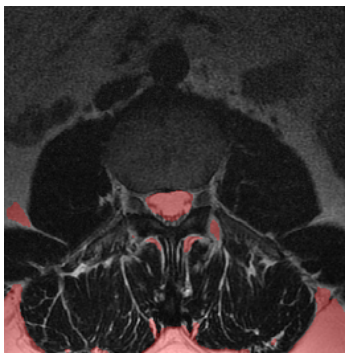

Figure 3.12: Connected threshold region growing results.

On top of the seed point and the lower and upper threshold values, the *neighbor connected* method requires the input of a radius for the pixel (or voxel) neighborhood. The algorithm will include the pixel in the segmentation result if and only if all the neighbors of that pixel have a value inside the threshold range. This means that the segmented structure will be smaller, depending on the radius provided. That is, if the radius is too large the resulting structure becomes smaller or may even be nonexistent.

Figure 3.13 shows results obtained with this variant of the region growing algorithm. The resulting structure is very similar to the one in the original image.

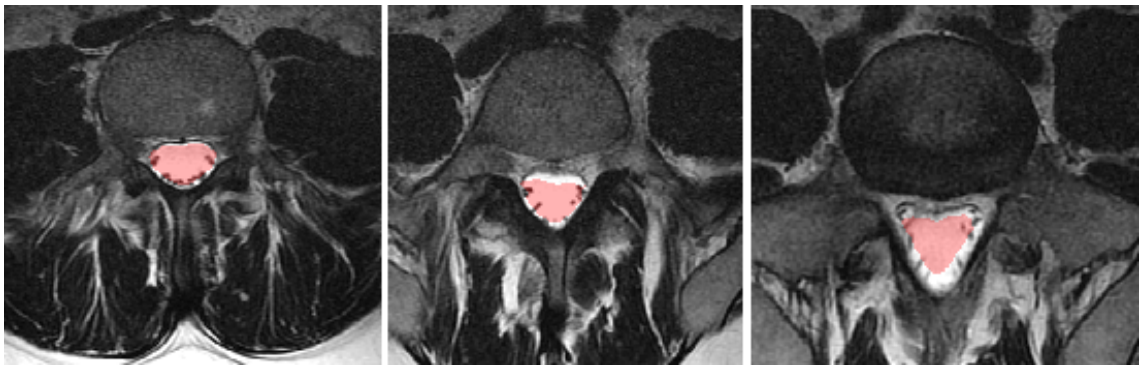

Figure 3.13: Neighbor connected region growing results.

We have decided to try one more region growing variant. The *confidence connected* doesn't need the user to provide two threshold values. Instead, the user must provide a factor, a number of iterations and a radius.

The method starts by computing the mean and standard deviation of intensity values for all voxels inside the region set by the radius and the seed point. The user-provided

factor is used to multiply the standard deviation and to define a range around the mean. Neighbor pixels whose intensity fall inside the range are accepted and included in the region. Before any iteration, the algorithm will recalculate the mean and the standard deviation for all the pixels that were included in previous iterations.

Equation 3.4 shows the range used as the inclusion criterion, where I is the image, X is the position of the neighbor pixel to be included in the region, μ is the mean, f is the factor defined by the user and σ is the standard deviation.

$$I(X) \in [\mu - f\sigma, \mu + f\sigma] \tag{3.4}$$

Figure 3.14 shows the results obtained with this variant of the region growing algorithm.

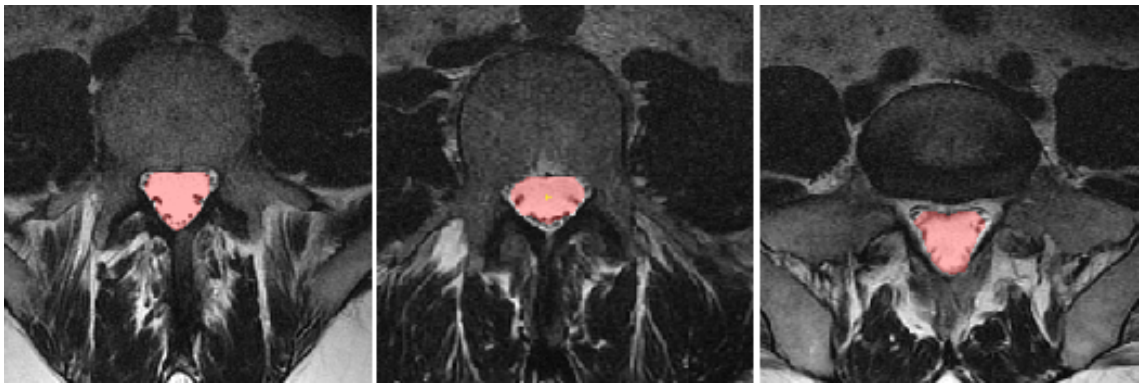

Figure 3.14: Confidence connected region growing results.

We can now draw some conclusions. When comparing the *confidence connected* algorithm with the *neighborhood connected*, we realise that both are very similar and any of the two can be used to segment the spinal canal. The only downside to the *confidence connected* is that it only accepts one seed point.

3.4.2 Abdominal Aorta

The segmentation process of the abdominal aorta we present is similar to the one suggested by Bjørn Sollie citebjorn2002. Figure 3.15 shows the underlying pipeline.

As mentioned before, we have to smooth the image prior to creating the appropriate speed image. Some tests concerning the filters to create speed images were made in Section 3.3.2. The key point is that the wanted speed image must provide sufficient data about the boundaries of the abdominal aorta.

Given that we started by using the sigmoid filter and in some data slices under analysis the boundaries of the abdominal aorta are not well defined, we must provide a $|\alpha|$ value sufficiently large so that the speed function can slow down in some areas. If this accomplished, we have a greater control over the evolution of the level sets. The β value can taken from the tests made in Section 3.3.2.

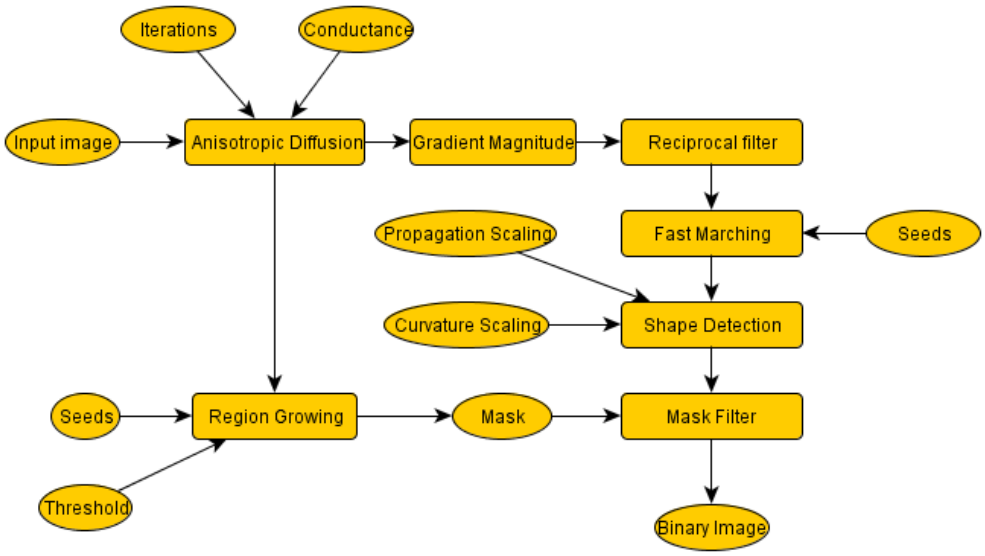

Figure 3.15: Abdominal aorta segmentation pipeline.

Having generated the speed image we then applied the fast marching method. This method requires the user to provide seed points and two parameters – the *initial distance* and the *stopping time*. The *initial distance* is the value at which the initial contour should be relative to the defined seed points. Since the contour will propagate continuously over time, it is desirable to stop the process once a certain processing time has been reached. In this case, we save computation time.

Figure 3.16 shows the results obtained after applying the fast marching method, using the parameters and related values shown in Table 3.4. The speed image we have used was generated by the sigmoid filter. We should point out that a threshold was applied because the images obtained were too dark (the colors were inverted).

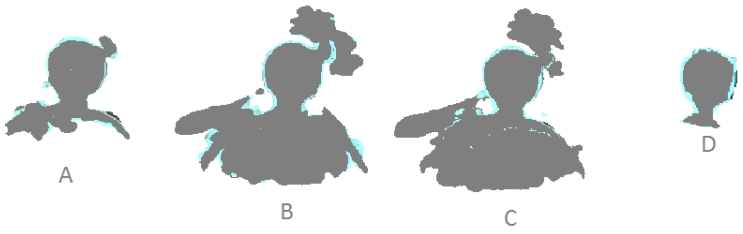

Figure 3.16: Fast marching method applied to the sigmoid filter speed image. Table 3.4 lists the values of parameters used.

In respect to Figure 3.16, images A, B and C have a lot of leakage. Image D, although not with as much leakage as the others, has another problem. By using a lower stopping value we have limited the growth of the level . This means that the level didn't spread to some of the slices, thus the segmentation was incomplete. To solve this, we can simply add more seed points.

Image	Initial Distance	Stopping Value	Sigmoid time (ms)	Reciprocal time (ms)
A	0	100	1031	1015
B	50	100	2688	2734
C	50	50	984	1047
D	0	50	500	546

Table 3.4: Parameters used to generate the images in Figure 3.16 and Figure 3.17. Also, the computation time using the sigmoid and the reciprocal filters, for speed images.

Figure 3.17 shows the results obtained when using the speed image generated by the reciprocal filter. The parameters used in each of the images are listed in Table3.4.

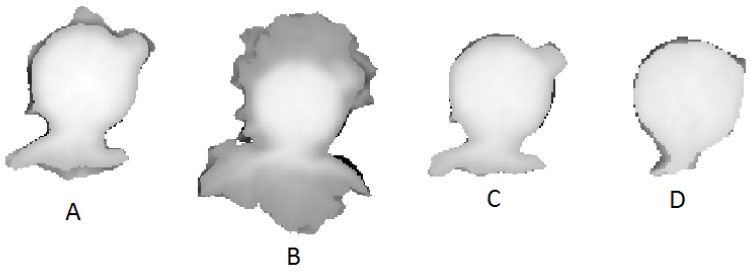

Figure 3.17: Fast marching method applied to the reciprocal filter speed image. Table 3.4 shows the parameters used.

There is less leakage in the second set of images of Figure 3.17. It may be because the reciprocal speed image is "slower" than the sigmoid speed image. Nevertheless, there is still some leakage in the results obtained and because of this, we have decided to apply a shape detection algorithm.

The algorithm we have used for shape detection is based on the one presented by Malladi *et al.* [RMV95]. This choice is due to the fact that it is the simplest shape detection algorithm that ITK provides. Moreover, it gives us great control with the adjustment of its parameters.

The algorithm requires the user to provide two parameters, the *propagation scaling* and the *curvature scaling*. The propagation parameter controls the expansion or propagation of the front (see Figure 3.18). The curvature scaling controls the curvature smoothing. The larger the curvature scaling the smoother the resulting segmentation. However, this parameter shouldn't be too large, as it will misrepresent the shape boundaries. These two parameters compete with each other so the user just provides their relative weighting.

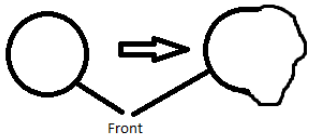

Figure 3.18: Propagation of the front.

The level set evolution will stop if the convergence criteria or the maximum number of iterations is reached. The root mean squared (RMS) is used to define the convergence criteria: The evolution is said to have converged if the RMS change between two iterations is below a certain value.

Concerning the image input of the shape detection algorithm, it requires an initial level set (which is provided by the fast marching method) and an edge potential image. The speed image used in the fast marching method can also be used as the edge potential image for this method.

Figure 3.19 shows the results, after applying the shape detection algorithm using the parameters and related values listed in Table 3.5.

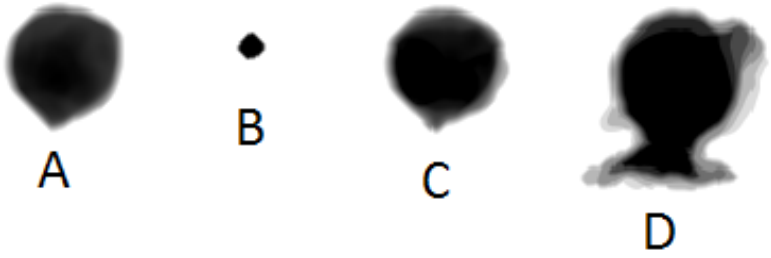

Figure 3.19: Results after applying the shape detection algorithm using the parameters listed in Table 3.5.

Image	Iterations	Initial Distance	Propagation Scaling	Curvature Scaling	Time
A	800/800	5	5	1	52515
B	40/800	5	5	1.2	12375
C	800/800	5	100	0.5	49891
D	1/1	100	1.0	1.0	18016

Table 3.5: Parameters and values used to generate the images in Figure 3.19, using a shape detection algorithm.

Regarding Figure 3.19 and Table 3.5, the parameters' values used in the images A, B and C highly depend on the seed position. By changing the curvature scaling by only 0.2, we obtain a very different image (images A and B). Because of these two factors, we are in favor of two parameters like the ones used in image D. Even though there are some leakages in image D, these can be mitigated using a mask generated by a threshold filter or a region growing algorithm.

Figure 3.20 shows a mask generated with the help of a region growing algorithm. The mask is represented with color red. Figure 3.21 shows the situations before and after applying the mask.

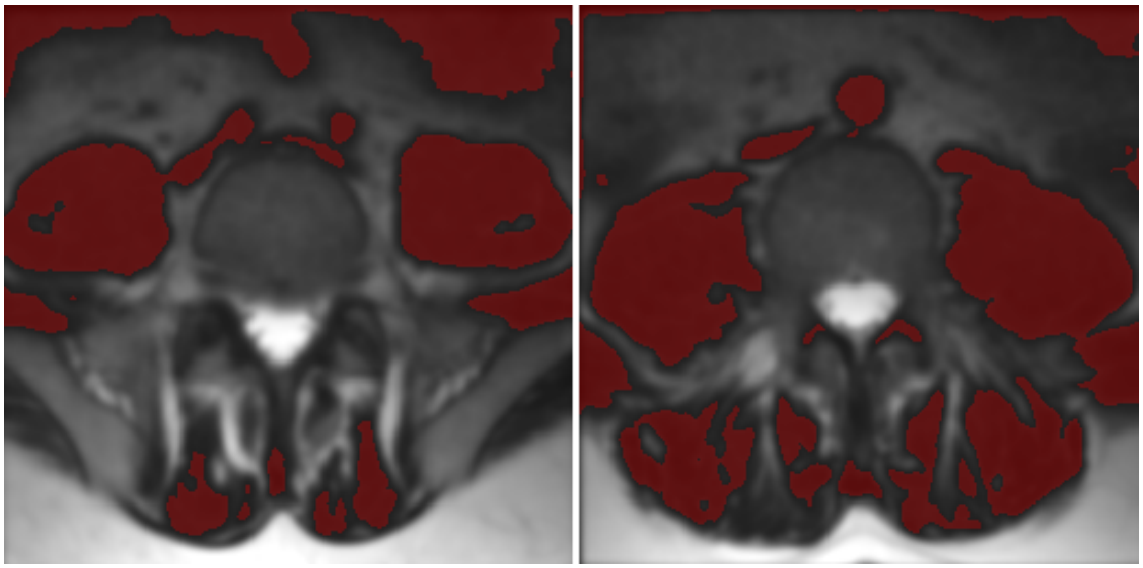

Figure 3.20: Mask obtained using the neighborhood connected region growing algorithm.

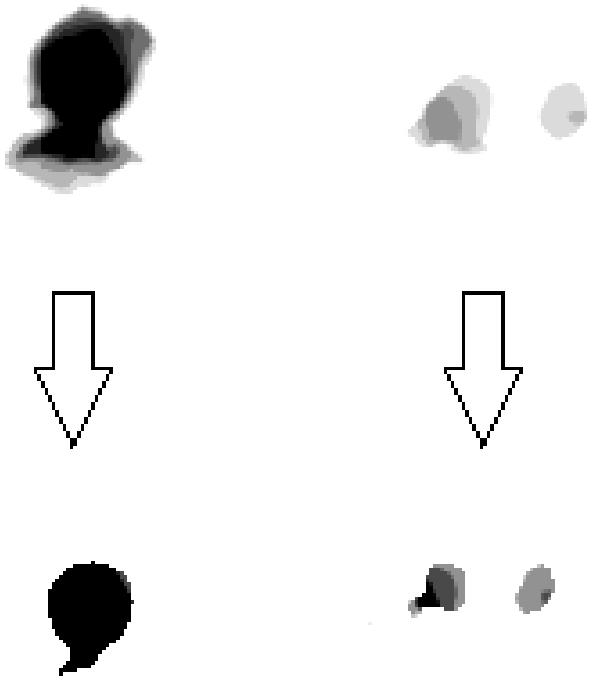

Figure 3.21: Before and after applying the mask.

3.4.3 Vertebral Body

As a first attempt to segment the vertebral, we have used several region growing algorithms. Figure 3.22 shows the results (the red color represents the achieved segmentation).

Considering Figure 3.22, the left image is the resulting segmentation from a simple *connected threshold* region growing algorithm. There is a lot of leakages using this method and that's the main reason we've decided to try the *neighbor connected* and *confidence connected* region growing algorithms. The *neighbor connected* resulting segmentation is displayed in the middle image. With this method, we were able to reduce leakage (compared to previously), but the segmentation still spreads to many unwanted regions. The right image shows the resulting segmentation with the *confidence connected* region growing. Overall, there is no clear improvement relative to the previous method and some of the areas of interest have shrinked.

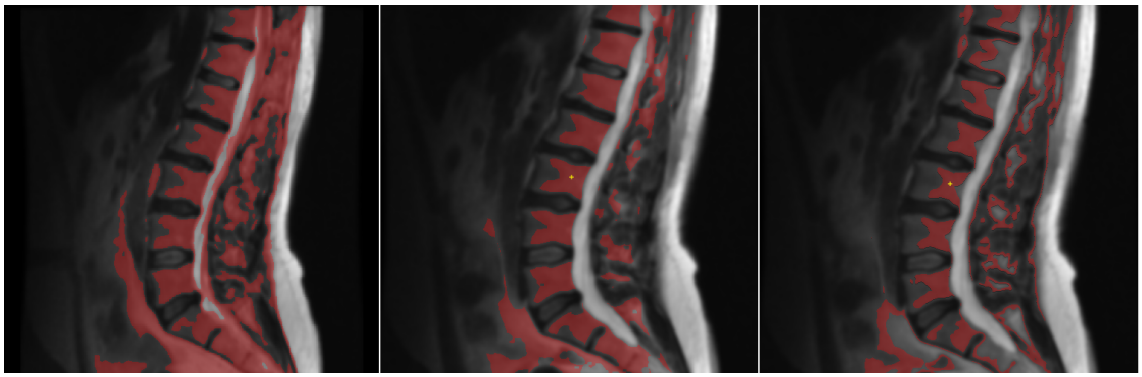

Figure 3.22: Three region growing algorithms applied to a RMI dataset of the lumbar region. From left to right: connected threshold, neighbor connected and confidence connected.

Having observed the results in the left and middle images, we have decided to apply a region of interest (ROI) mask (that has to be defined by the user). The user only needs to define the ROI in one of the slices, because the vertebral body stays in the same position in all slices. It means that the ROI can be propagated through all slices. This works well in the sagittal and coronal perspectives, but if dealing with the axial perspective, we would have to take into account the normal curvature of the human vertebral column. The results obtained using the ROI mask are displayed in Figure 3.23.

Since the structure obtained is not very accurate, we have decided to take a different approach, using level sets. The initial steps are similar to the ones used at the time of segmenting the abdominal aorta, described in Section 3.4.2.

Figure 3.24 shows the pipeline used for the segmentation of the vertebral body. Figure 3.25 shows the intermediate results, after applying some of the filters established in the pipeline.

The first level of the fast marching method coincides with the area occupied by the vertebral body. Since the first level is represented by a darker color, we can simply apply a threshold filter and avoiding to waste computational time applying a shape detection algorithm. Figure 3.26 shows the results after applying the threshold filter to the fast marching image.

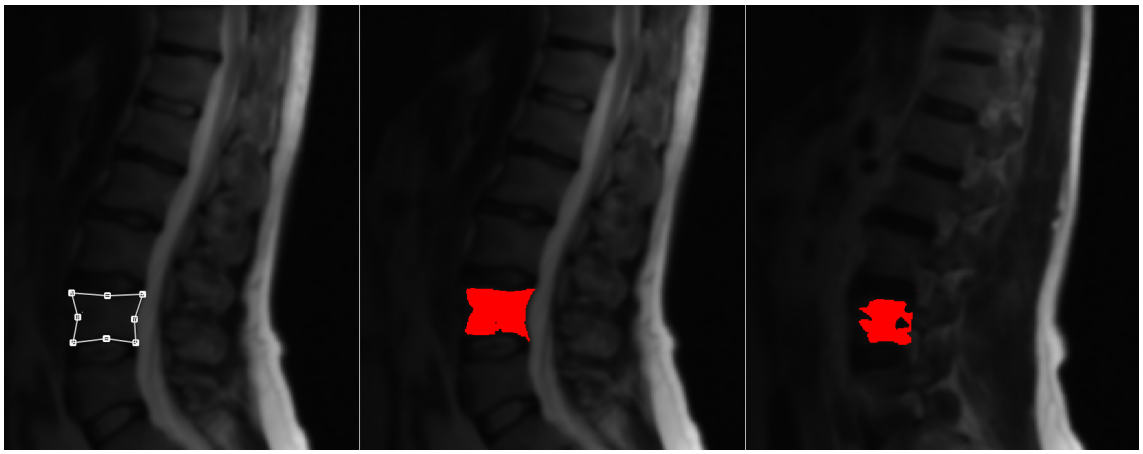

Figure 3.23: Results after applying a region of interest mask.

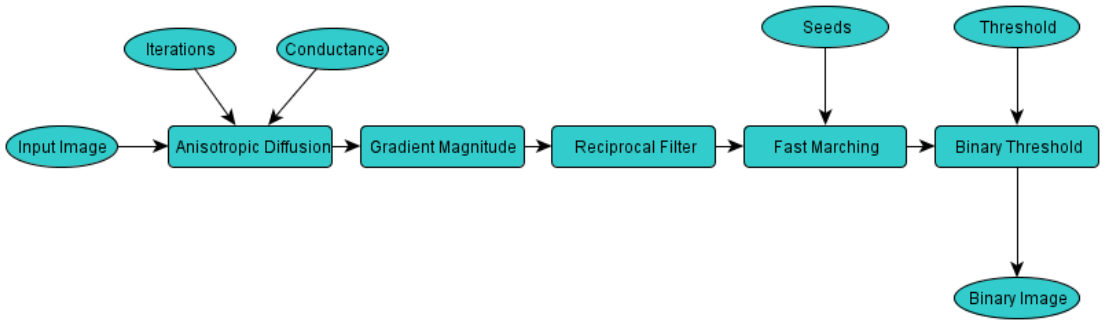

Figure 3.24: Segmentation pipeline used to segment the vertebral body.

We compared the execution time of both methods (shown in Table 3.6), region growing and level sets. Unfortunately, all the datasets taken from a sagittal perspective had the same dimension. The smoothing stage was not considered, since both methods use a smoothing algorithm in the pre-processing stage. As expected, the region growing method is faster but, as we have seen before, it provides lower quality results. The parameters' values used in the fast marching method were $initialdistance = 100$ and $stoppingvalue = 100$.

Region Growing (ms)	Fast Marching (ms)	Dataset size
1703	5125	512x512x12
1422	5172	512x512x12
1519	5033	512x512x12

Table 3.6: Parameters and values used to generate images in Figures 3.16 and 3.17. Also, the computation time using the sigmoid and the reciprocal filters for speed images.

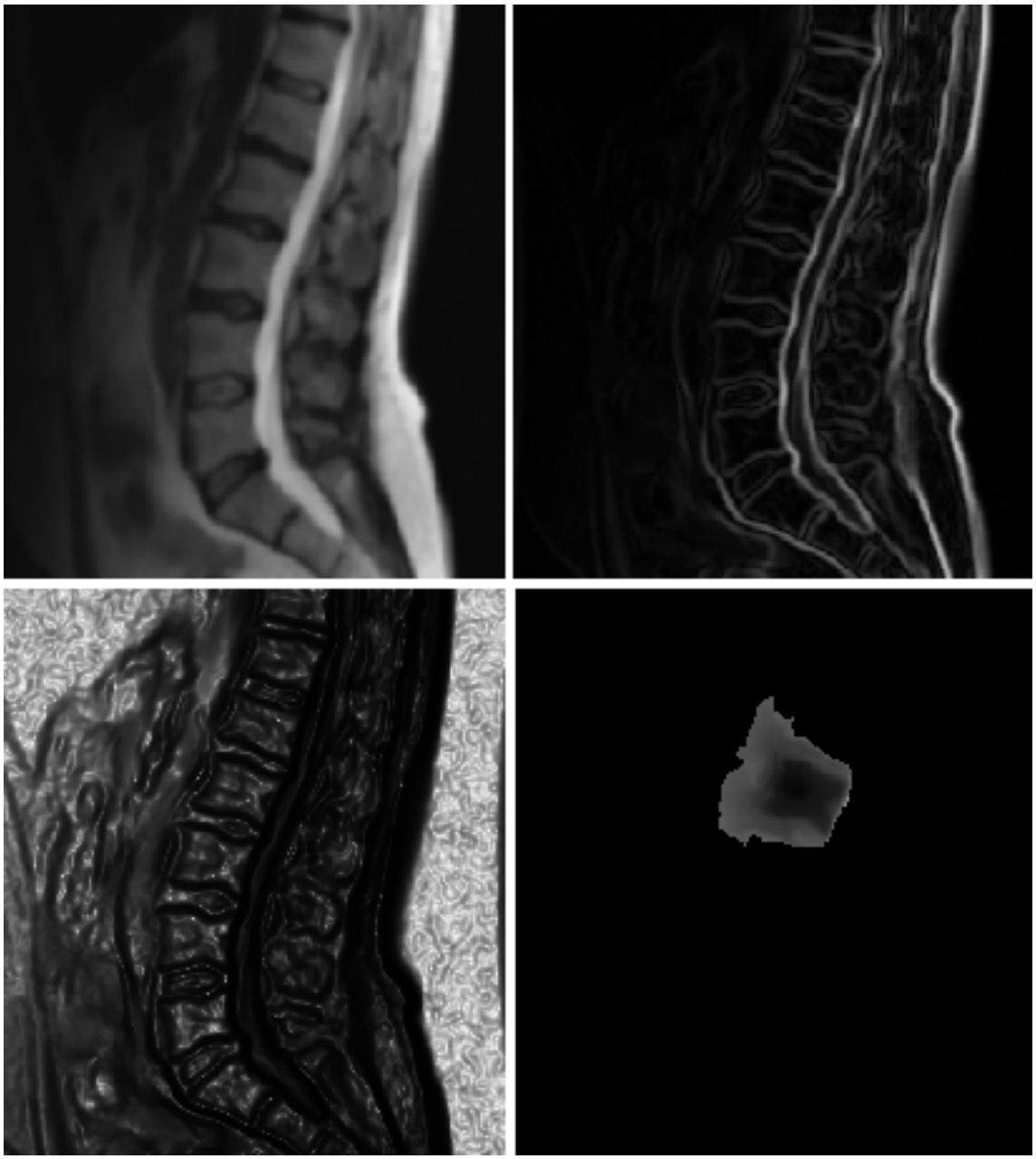

Figure 3.25: Top-left image shows the results after applying the gradient magnitude filter. The top-right image shows the results after applying the reciprocal filter and the bottom left image shows the fast marching results.

3.4.4 Back Muscles

Muscles in MRI images can be represented with high intensity or low intensity values. In the datasets we have used, the muscles are represented with low intensity values.

The first segmentation algorithm we have tested was the region growing. Several parameters's values were tested but the obtained results were similar. Even more, in some cases the segmentation didn't spread sufficiently so that the whole anatomical structure could not included in the final result. Figure 3.27 shows some of the resulting images.

Figure 3.26: Thresholded fast marching results.

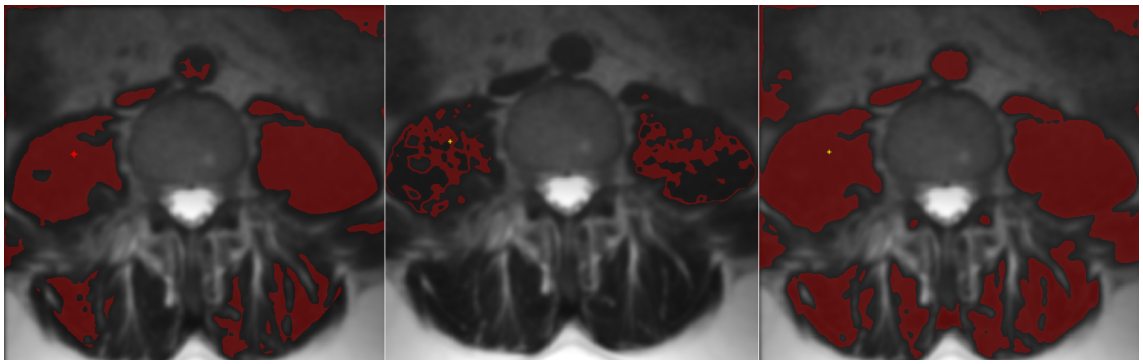

Figure 3.27: Region growing applied to the interior back muscles.

Having achieved poor results with region growing methods, we have decided to try a level sets method. Unfortunately, the results were not better. Figure 3.28 shows that there is least a leakage problem (i.e. the segmentation spreads too much).

A similar approach to the one used in Section 3.4.2 – when we were segmenting the abdominal aorta and a mask generated by threshold or region growing to reduce leakage was applied – would not work. This is because the leakages in the region growing method coincide with the leakages in the level sets method.

Having obtained poor results with the methods above, we have decided to take a different approach. Since the problem were the leakages, we can define a ROI around the area that the muscle will appear throughout the dataset. Of course, we have to take into account that the muscle will be in a different position in a different slice. This is because we are dealing with the axial perspective of the lumbar region.

Figure 3.29 shows a possible ROI. Having defined the ROI, we have two options:

Figure 3.28: Fast marching method applied to the interior back muscles.

Either using the region growing results or the level sets results. If we decide to use the region growing results, the user may have to define several threshold values for different slices in the dataset. This is related to the quality of the dataset being used. On the other hand, a level sets edge based algorithm will not have problems addressing this issue. But we have decided to test both methods.

Figure 3.30 shows the results after applying the fast marching method masked with a ROI and Figure 3.31 shows the results after applying a region growing method (with multiple threshold values) masked with a ROI. A time comparison between region growing and fast marching was already made in Section 3.4.3.

The region growing algorithms have a tendency to spread all over the image instead of a "localized" spread. Because of this, the spread created by the fast marching method is easier to control. Furthermore, the region growing algorithms require the user to input more data and adequate threshold values may be hard to find. Still, we consider both approaches as useful segmentation strategies.

3.5 Conclusions

Having tested various algorithms and variants to segment structures of the lumbar region, we are now in a position to draw some conclusions about segmentation.

First, regarding the pre-processing phase, the median filter is ideal when dealing with images that have little noise but still need some smoothing. This is because images with

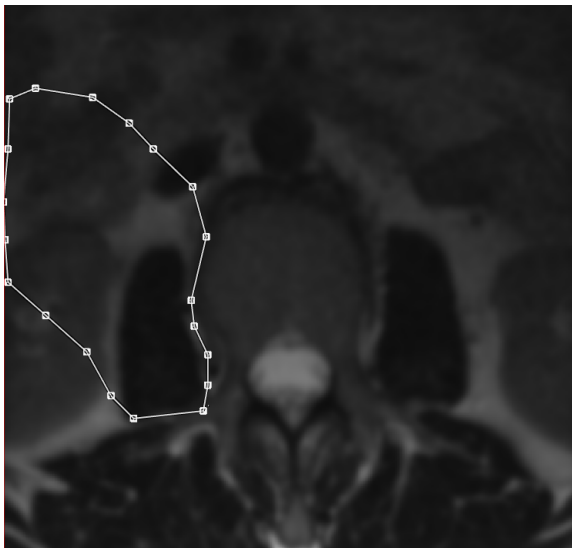

Figure 3.29: A possible region of interest to use in the segmentation of interior back muscles.

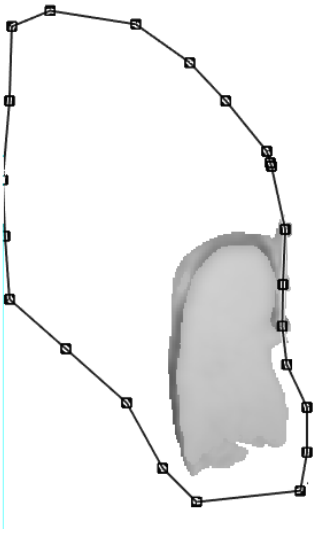

Figure 3.30: ROI mask applied with the fast marching method.

less noise require less iterations. If the median filter is applied too many times, the boundaries in the images start to get damaged, which will harm the segmentation phase. On the other hand, anisotropic diffusion is ideal when there is too much noise. Otherwise, the median filter is preferable due to its simplicity.

In Section 3.4.1, neighborhood connected region growing and confidence connected were presented. Both methods can deliver good results, but the confidence connected algorithm may be less intuitive to use, which makes it more difficult to find the right parameters' values. The radius in both methods can not be too large otherwise it slow down the process too much. This is particularly relevant when dealing with 3D images.

Figure 3.31: ROI mask applied with a region growing algorithm.

Another important aspect is that images should be smoothed before applying the gradient magnitude filter. If the image is not smoothed, the speed image filter will return bad quality results, which then harms the segmentation process.

We have used two types of segmentation algorithms: Region growing and level sets [4]. We consider that it is preferable to use region growing (if the results are satisfactory) instead of level sets. Furthermore, level sets require additional intermediate steps to initialize the algorithm, using gradient magnitude filter, sigmoid filter, etc. The two methods can be combined together in order to obtain better results. It is worth mentioning, that after the segmentation step, the user can make his own adjustments to further improve the segmentation results. Chapter 5 address some of these possible adjustments.

Figure 3.32 shows the rendering results of all the structures mentioned in this chapter. No smoothing algorithm was applied.

[4] Although ITK also provides segmentation based on watersheds, we have decide not to use it because it takes too long to show results when applied to 3D datasets.

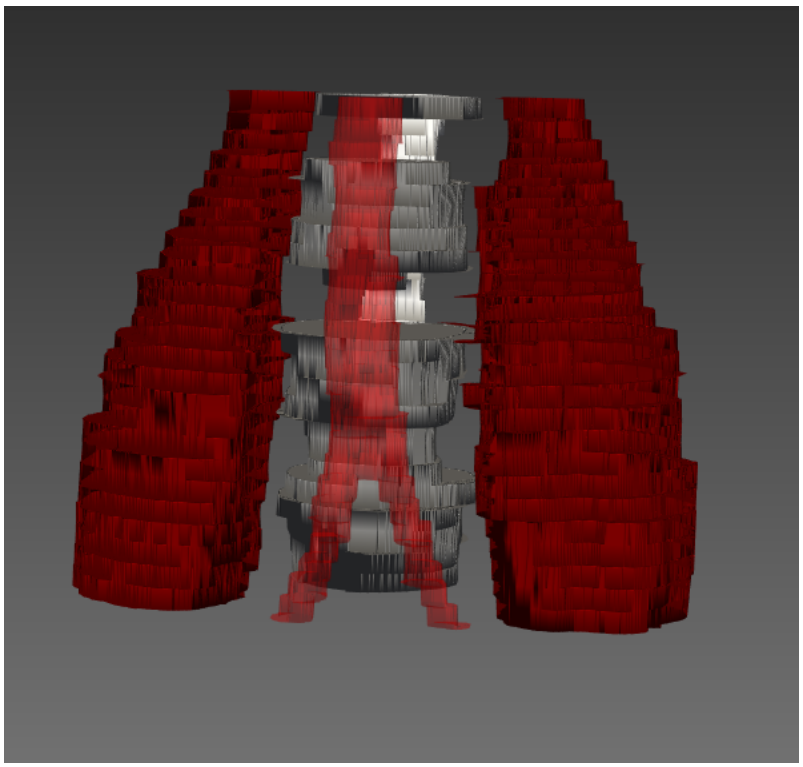

Figure 3.32: Rendering of all the structures mentioned in this chapter.

4

Time-Dependent Rendering

One of the objectives we set out in this dissertation was the representation of anaesthesia movements inside the body. Therefore the datasets have to be time-dependent, that is, the data under analysis varies will vary in time.

Unfortunately, we were not able to obtain datasets related to anaesthesia movements. In relation to the datasets used in Chapter 3, it was missing the time dimension. We have to recognise that it is difficult to establish a case study where patients are exposed to image scanners for a long period of time. That was precisely what happened when we asked some professionals in a national hospital to help us with this issue. Not only there were some medical and ethical concerns to deal with but financial ones as well.

Giving this scenario, we looked at other regions of the human body as the processes are somehow similar. Hence, we have used a 4D dataset of a human lung, depicting its functioning in time [ea13]. Notice that anaesthesia movements are essentially influenced by human organs, which means that representing organ movements is a fairly analogous process to representing anaesthesia movements.

In this chapter, after setting the proper framework, we present two approaches to represent time-dependent datasets: With multiple 3D datasets and with one 4D dataset. The number of 3D datasets used was 10 with the dimensions $50 \times 224 \times 224$ (thus $10 \times 50 \times 224 \times 224$) and were taken from the sagittal inclination. In the end we compare both approaches and discuss the results obtained.

4.1 Framework

Regardless of being time-dependent or not, the rendering of data has to go through a pre-processing step, a segmentation step and a rendering step. The main difference now with

time-dependent data is that we no longer have just one resulting model – the segmented parts – but multiple models as time evolves. On that basis, for similar quality, we will need to spend, on average, more time to accomplish all these steps.

In respect to user interactivity, we have two situations: Models are changed by user or by the application. If it is the user's responsibility, they may be more focused on details. On the other hand, if the application is changing the models for the user (movie or animation analogy), the user may not notice enough model details. Of course, it also depends on the frame rate achieved or set.

To summarise, we have three main objectives in time-dependent rendering:

- To maintain interactivity to some degree.
- To achieve a reasonable quality in the resulting models.
- To deal with memory usage efficiently.

4.2 4D Dataset

In this approach we start by joining multiple 3D datasets, representing the same organ but at different times, into one 4D dataset to simplify the programming process.

When joining the 3D datasets, we must make sure they are correctly ordered (relative to their timeline). After obtaining the 4D dataset, we go through the multiple stages of segmentation mentioned in Chapter 3. The main problem here is the size of the 4D dataset. The pre-processing and segmentation stages will take more time to finish.

Knowing that this approach will be slow, and by slow we mean the user has to wait longer to see any visual representation of results, we may infer that time is not an issue if the user decides to use this method. If that is the case, then slower segmentation algorithms can be used if needed to obtain good quality rendering.

Figure 4.1 shows the pipeline used in this approach.

The rendering encompass multiple 3D models as we operate segmentation on multiple 3D datasets. In this particular approach we are not able to render the models separately. That is, since we have only one data set, MITK will render the whole dataset. Even though we will not be able to see all the 3D models at the same time, they are stored in memory. Creating a 3D model using the marching cubes algorithm is very efficient and so the time spent building the 3D models is negligible when compared to the time spent on segmentation.

To change between models, we simply have to select a different time slice of the dataset. If we want to create an animation with various models, there will be some limitations related to the frame rate. As all 3D models are stored in memory, the frame rate will depend mostly on loading the models from memory and displaying them.

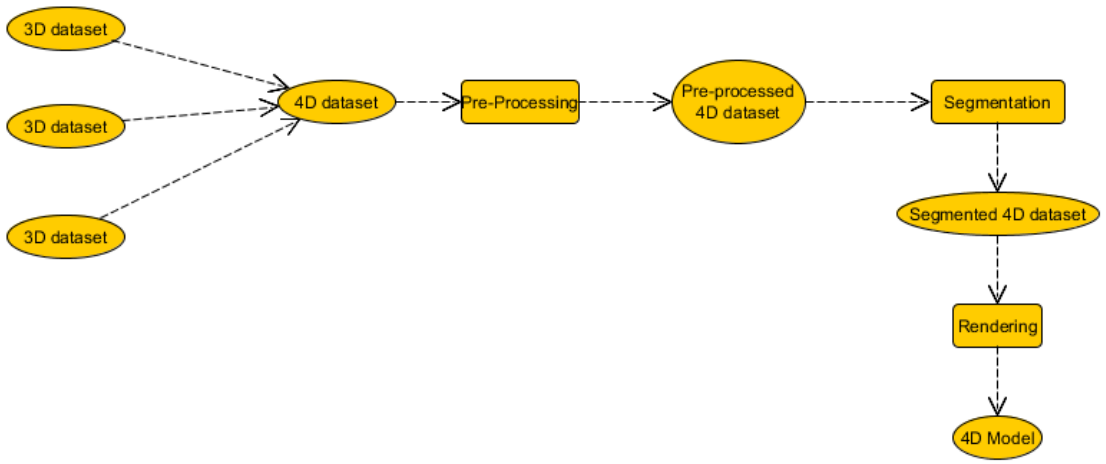

Figure 4.1: Pipeline for time-dependent rendering using multiple 3D datasets.

4.3 Multiple 3D Datasets

This approach relies on processing each of the 3D Datasets separately.

By doing this, we can process multiple slices independently. Furthermore, there is the possibility of frame skipping if the user wishes to do so, which means that some time slices will not be segmented nor renderer at all.

In this context, we have decided to use a similar strategy as the one proposed by S. Silva *et al.* [SSM13]. Figure 4.3, shows the pipeline we have set.

We start by defining a Region Of Interest (ROI) around the lung (see Figure 4.2). This ROI is used to set the boundaries for the region growing algorithm to a certain area. Also, we provide a seed point for the region growing algorithm.

Given that we are dealing with multiple 3D datasets, the defined ROI and seed point must be the same in the remaining datasets. Also, when defining the seed point, the user must take care not to select a seed point too close to the boundaries of the anatomical structure (in this case the lung) because it might shrink (for example, when exhaling the lungs shrink).

Regarding the pre-processing stage, and taking into account conclusions drawn in Chapter 3, we have decided to use the median filter for smoothing as it is faster than *anisotropic diffusion*.

Table 4.1 shows the time spent to execute the median filter with 5 iterations, radius of 1 (in all three dimensions, X, Y and Z) and with different number of threads. Unfortunately, the maximum number of threads allowed by ITK equals the number of processor cores available in our computer.

Since one of the objectives of this approach is allowing offline real-time segmentation, (even though not needed, since the tool being developed is going to be used as a learning tool), we have used region growing because it is one of the fastest segmentation algorithms. Table 4.2 shows the average time spent to execute the algorithm for each 3D

Figure 4.2: Region of interest around the lungs.

dataset (or time slice).

Threads	Time (ms)
1	31107
2	17749
3	12118
4	9687

Table 4.1: Execution time for applying the median filter to a dataset of size $50 \times 224 \times 224$, and variable number of threads.

As mentioned before, the time spent building the 3D models is negligible when compared to the segmentation time. Because of this, we did not include a table with such data.

This approach also has the advantage of being able to create multiple 3D models one at a time. It means we can avoid storing all 3D models simultaneously but only the ones currently used. Moreover, since the marching cubes algorithm is very fast, we can discard a 3D model after using. If we need a 3D model already discarded, we can simply use the marching cubes algorithm again without too much delay.

Figure 4.4 shows some of the 3D models obtained.

Slice number	Region growing (ms)
1	812
2	650
3	799
4	927
5	853
6	817
7	822
8	719
9	633
10	895

Table 4.2: Execution time for applying the region growing algorithm to different (time) slices.

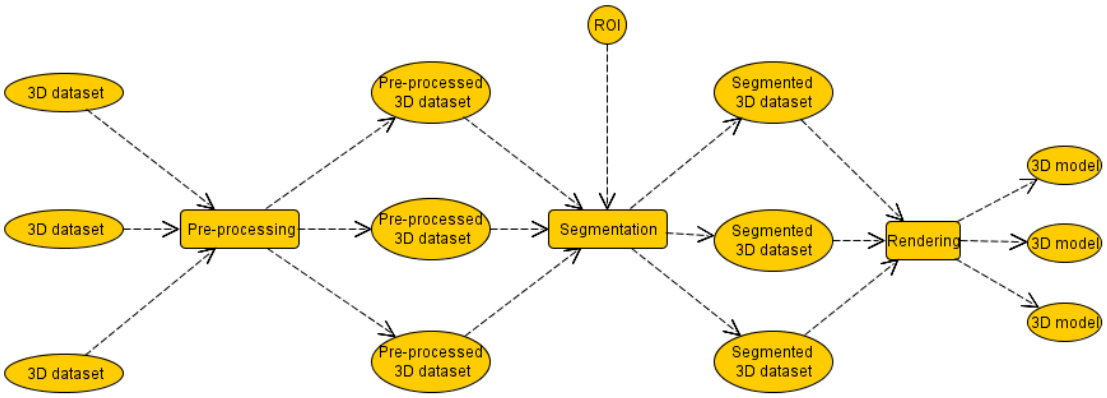

Figure 4.3: Pipeline for time-dependent rendering using 4D dataset.

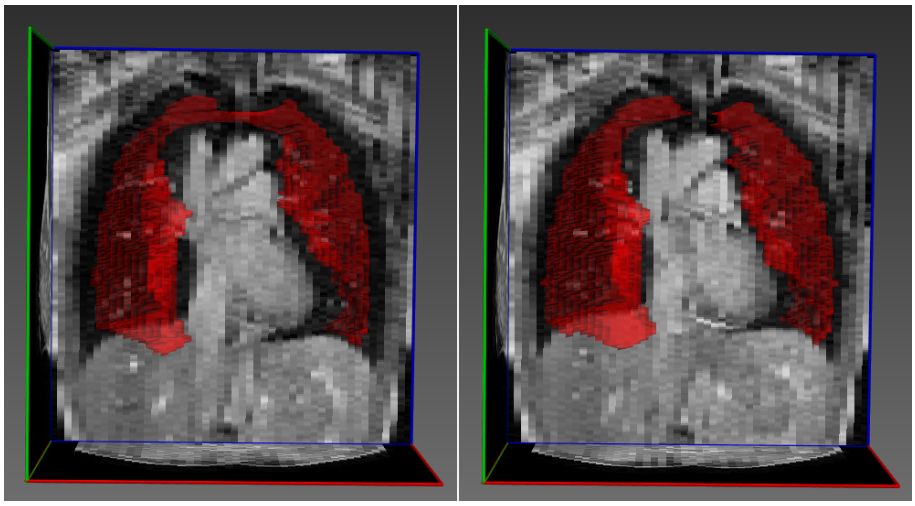

Figure 4.4: Rendering results for two different time slices.

4.4 Conclusions

Overall, when comparing both approaches presented above, we prefer the Multiple 3D Datasets approach, mainly because it gives more processing options.

The tests we made point out that a considerable amount of time is spent during the pre-processing and segmentation stages. As expected, the amount of time, besides depending on the size of the dataset under consideration, also depends on the number of threads used (in our case, limited by the computer processor).

Our tests have shown that the pre-processing step takes beyond 3 seconds. However, it is worth mentioning that the number of iterations used in the median filter depends on the quality of the dataset being used. Moreover, the execution time heavily depends on the radius parameter being set.

As a final note in relation to changing between 3D models, one can apply an interpolation technique to provide smoother transitions and to avoid processing of some time slices.

5

Prototype

The research described in the last two chapters relied on the prototype we have implemented. As mentioned at the time, we have used MITK, putting together ITK, VTK and Qt. Again, these toolkits are widely used in medical visualization applications, have good documentation and there is a variety of helpful algorithms available.

In this chapter, we start with a general overview of the prototype. Then we discuss its underlying software architecture and finally we take a close look at its graphical user interface.

5.1 Overview

Apart from the chosen toolkits mentioned above – mostly open-source and suitable for medical visualization – the prototype was designed under the following assumptions:

1. To be able to process 3D and 4D datasets, namely DICOM images, as this format is very common in medical visualization.

2. To provide user navigation in the displayed image along any data axis – axial, sagittal, coronal and time – commonly used in medical visualisation tools.

3. To provide various smoothing and segmentation algorithms, according to the techniques discussed in Chapter 3.

4. To save segmentation results, to which names and colors can be assigned for better organization.

5. To create a 3D model after segmentation of an anatomical structure.

6. To render segmentations and to update parametrization values of chosen algorithms on-the-fly. Users should grasp immediately the outcome of changes they have made. They should not proceed to next operations without assurance of previous results. As an example, they have to figure out if additional image smoothing is required or not.

7. To allow the creation of annotations in the images. The prototype can be used as a learning tool so the user must be able to associate annotations with the image and its anatomical structures. Moreover, if it is a time-dependent dataset, time-framed annotations should be allowed as well.

8. To change working 3D model from multiple 3D models available.

9. To allow the visualization of animated sequences of 3D models.

5.2 Software Architecture

The prototype's software architecture is based on the *Model-View-Controller* (MVC) architecture. The component *Model* represents the data, which in our case are MRI images stored in disk. The *View* component is associated with the graphical user interface. Finally, the *Controller* processes the requested operations. For example, opening a file, applying a filter over an image, etc.

Figure 5.1 shows the component diagram of the software architecture. The implementation of these components rely on the toolkits already mentioned, as follows:

- ITK provides the implementation of a wide range of image processing algorithms, some of which mentioned in Chapter 3.

- VTK is mainly used for the marching cubes algorithms, alongside other 3D operations such as decimation (a technique to reduce the geometric complexity of a surface).

- MITK is used for loading images from disk. Usually these images are in a different format from the one used in ITK. MITK handles conversion between different formats, as well as it provides graphical widgets for data visualization.

- Qt framework provides the basic elements usually found in graphical user interfaces: Windows, buttons, labels, lists etc.

To build this MVC based architecture we have followed the object-oriented programming paradigm. Not only it was suitable for our purposes but we were familiar and comfortable with. Three main packages were created, with the names *Model*, *View* and *Controller*. They were structured in the following way:

Model

This package contains the classes that represent data. The main types of data we have are *project data*, *datasets*, *segmentation data* and *3D models*.

Controller

This package contains the classes used to apply segmentation, smoothing and rendering algorithms. Also, it includes data loaders.

View

This package contains the graphical user interface classes. These are responsible for allowing the user to issue commands to the application, as well as the visualization of results after executing such commands.

Figure 5.2 shows a class diagram with the most important classes.

5.3 Graphical User Interface

Graphical user interface plays an important role in the acceptance of the prototype. We now introduce the main interface and how it works. It is also an opportunity to see how segmentation experiments described in Chapter 3 can be worked out by users.

The main window is the root window of the prototype. This window was designed so that the user easily understands how to start using the prototype. Figure 5.3 shows this window, with numbers superimposed in the image to identified its major widget areas.

The list below describes each of these widgets, from 1 to 4:

1. In this widget the user can visualize datasets. Each of the square areas in the widget offers a different viewing perspective: axial, sagittal, coronal and 3D. The user can interact with each square area to navigate through the dataset or to visualize the 3D image from a different viewing point.

2. This widget allows the user to navigate through the dataset, in axial, sagittal and coronal perspective, but in a more precise way. If it is a 4D dataset, then the navigation can be operated through time slices.

3. In this widget the user can select a specific segmentation to interact with. As various segmentations can be created, the purpose here is to allow the user to modify a particular one if he wants to achieve some modification.

4. In the top menu the user can access most of the functionalities of the prototype. For example, opening a project, to run smoothing and segmentation algorithms, to render a surface, etc.

In order to have a better understanding about how the interface works, we present a use case.

The goal in this use case is to create a 3D model of the spinal canal. First, the user selects the option $File > OpenDataset > LoadDICOM$ (Figure 5.4). After that, a folder selection window will pop up and then the folder where the DICOM files are is selected.

The dataset is then displayed in the visualization widget (identified by 1 in Figure 5.3). Now, the user can open the segmentation window by selecting $Segmentation > RegionGrowing$. It will opens the window shown in Figure 5.5. In this new window, the user can select a smoothing algorithm if he wants to smooth the image. In this particular case, the *Median Filter* was selected. Then, the widgets to input the parameters' values of the selected algorithm become enabled. To apply the algorithm, the user pushes the button *Run* and, by pushing the button *Accept*, it proceeds to the next step.

Since the user has chosen *Region Growing*, a window with a panel containing multiple region growing methods is shown (Figure 5.6). The selected option was *Neighbour Connected*. Once again, the widgets related to the parameters of the selected algorithm become enabled. To set the seed points for the region growing the user simply press *Shift* key and click in the targeted region for the seed point. To apply the selected region growing method, the user pushes the button *Run* and results are sent to to the main window by pushing the button *Accept*.

Instead of directly choosing the region growing method, the user can select $Segmentation > Pre-defined > SpinalCanal$. This option provides some pre-defined parameters' values for this particular segmentation. Hence, the user does not need to care about region growing.

Another interesting use case is the creation of a 3D model. Considering the example above, the user can select the segmentation already obtained and then select $Rendering > RenderSurface$. Figure 5.7 shows the results.

Once the segmentation is selected, the user is able to modify the obtained results from the region growing method. The prototype provides some tools to do so. Figure 5.8 shows the results of using the *Add Contour* tool, accessible via the menu path $Edit > Tools > AddContour$. The green line in the image is the contour to be added.

Finally, if there are multiple 3D models of the same anatomical structure, the user can see an "animated" visualization of the models. This is achieved by selecting $Rendering > Play$. This option only works with 4D datasets and the 3D models must correspond to each time slice.

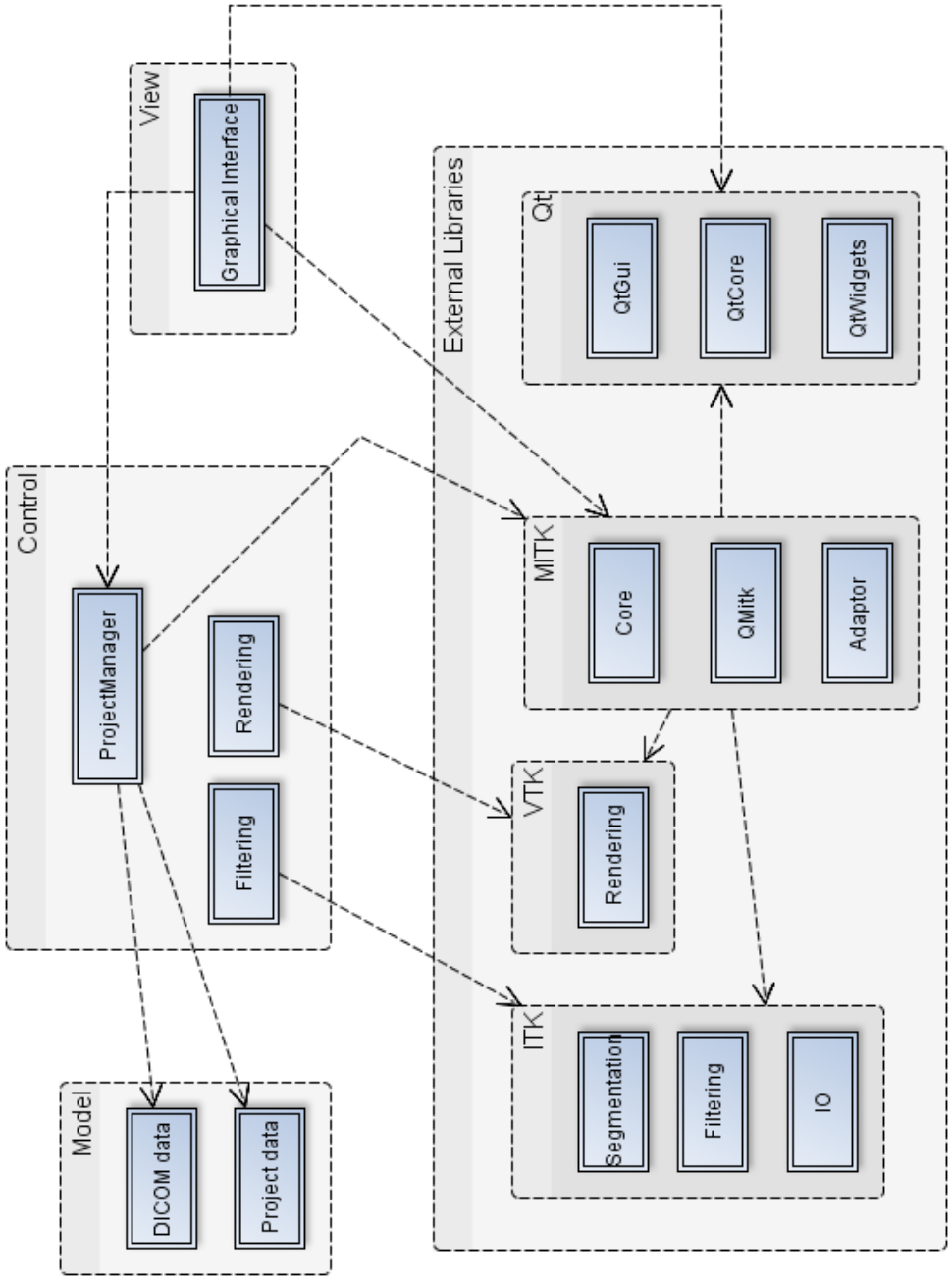

Figure 5.1: Component diagram of the prototype's software architecture.

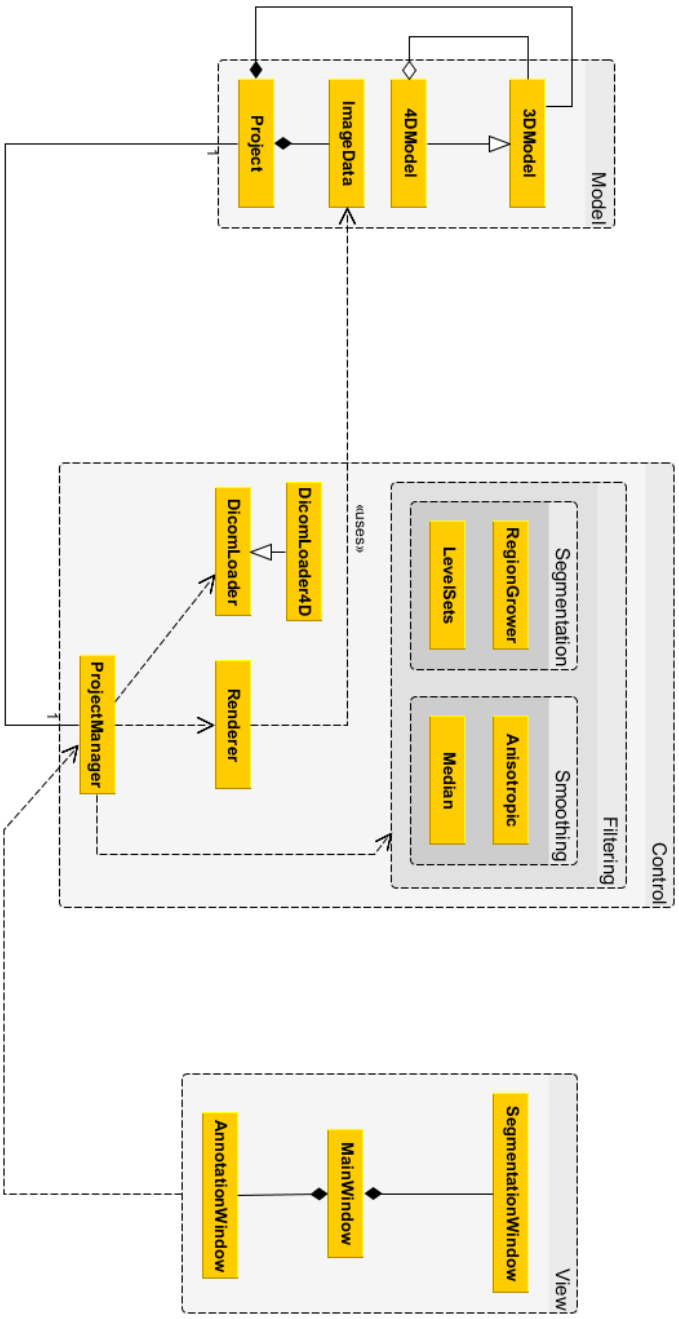

Figure 5.2: Classes diagram associated with the implementation of the prototype.

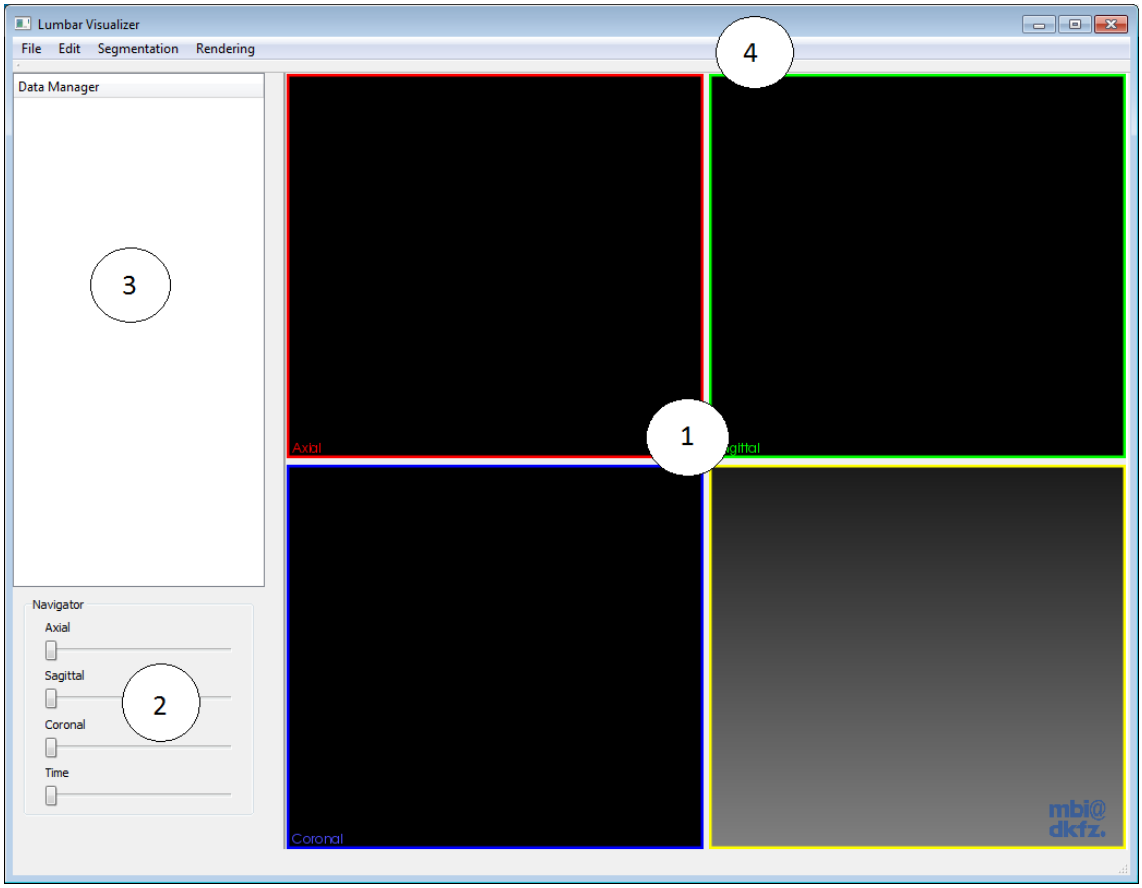

Figure 5.3: The main window of the prototype.

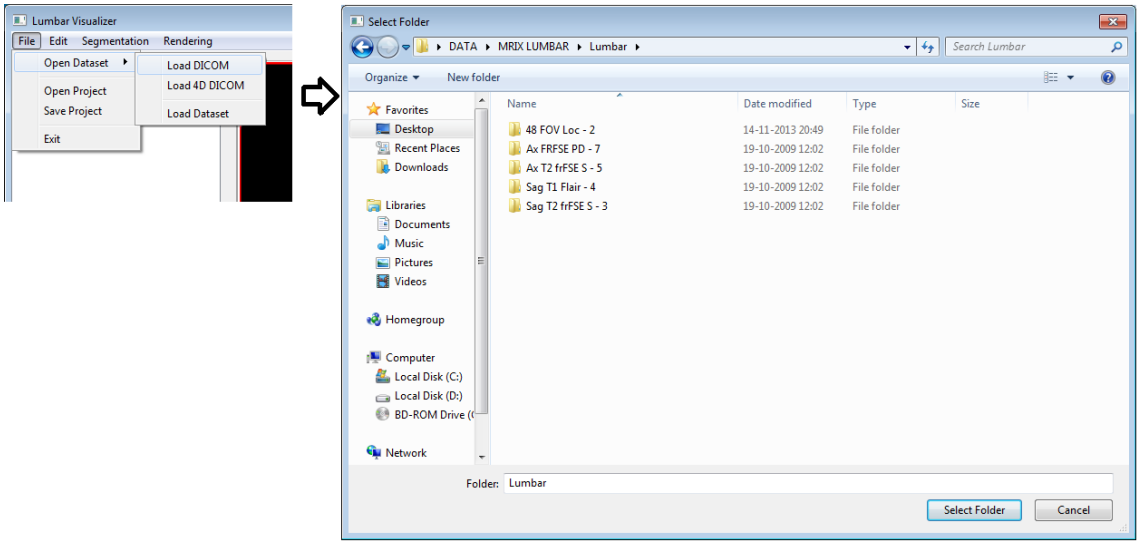

Figure 5.4: Loading a DICOM dataset.

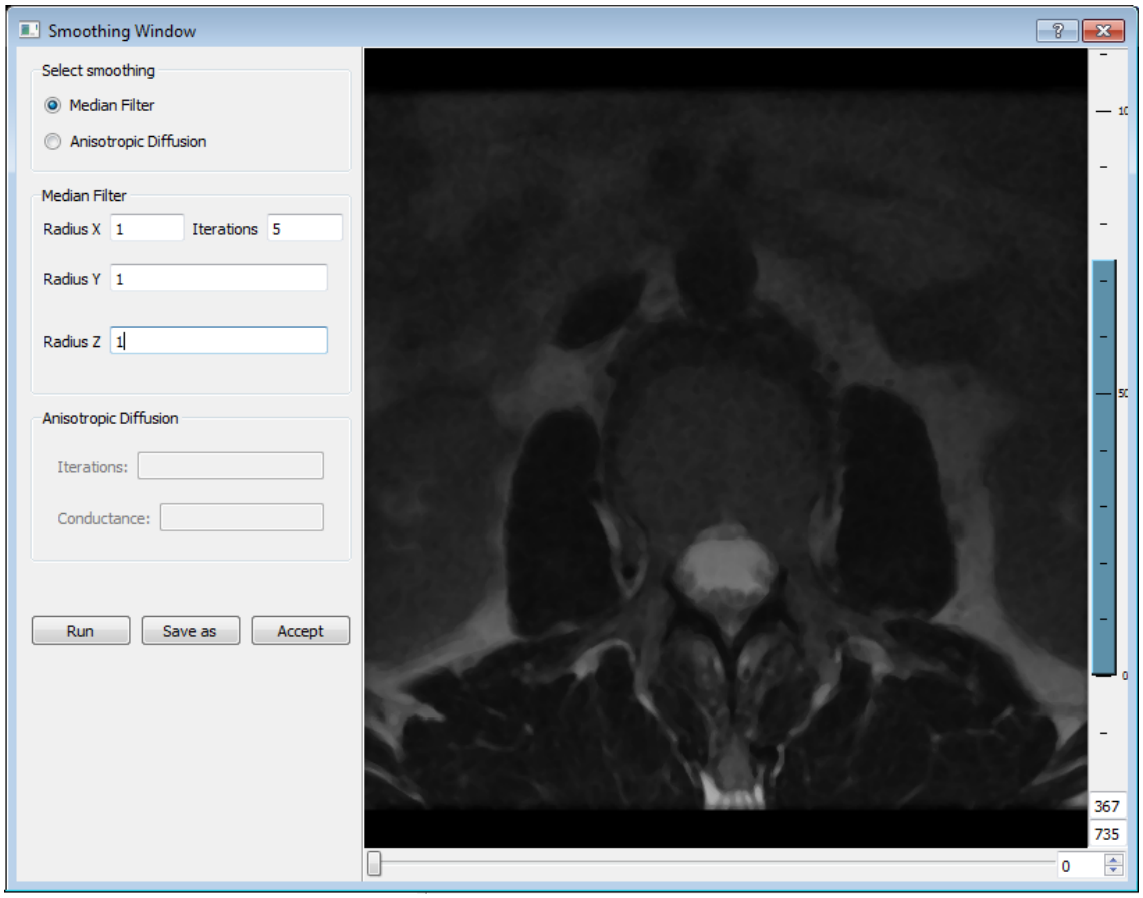

Figure 5.5: Smoothing window.

Figure 5.6: Region growing window.

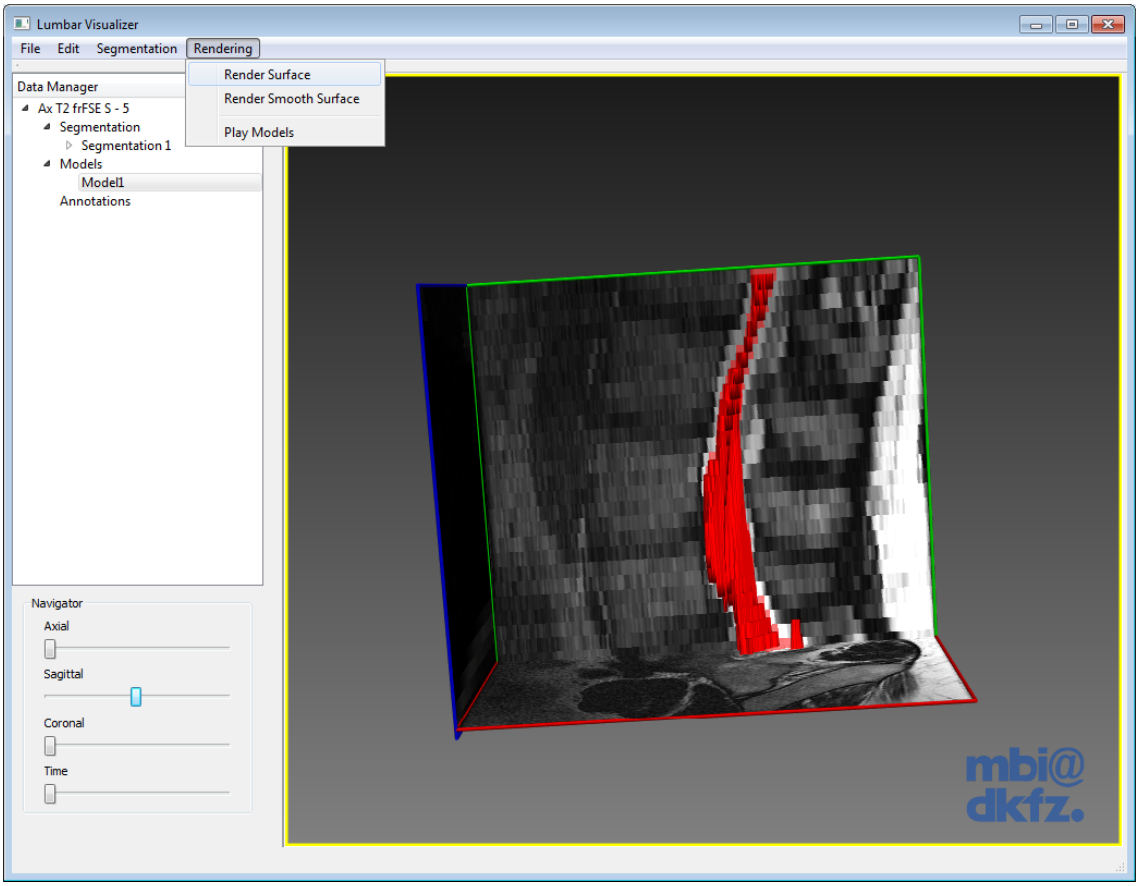

Figure 5.7: Rendering a 3D model.

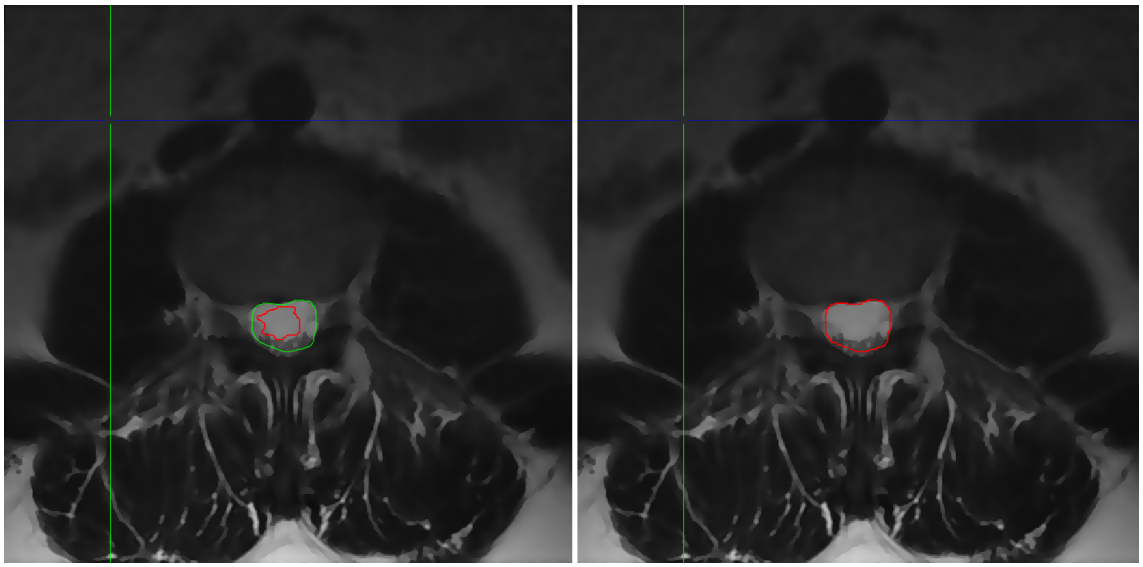

Figure 5.8: Adding contour to the segmentation results.

6

Conclusions

In this final Chapter we summarize and discuss the most important research contributions of our work. We conclude with some suggestions for further work.

6.1 Summary

In this dissertation, a standalone prototype has been developed with the purpose of allowing users to successfully segment the multiple anatomical structures in the lumbosacral region of a human body. Additionally, there was the goal of providing young medical practitioners a learning tool to analyse and evaluate anesthesia acts performed in that region.

The focus was just on using MRI imaging techniques. As these techniques provide lower resolutions when compared to CT imaging, the implemented algorithms have to cope with problems associated with lack of resolution. Hence, some tests were made to find suitable methods to remove noise in MRI scans. In that respect, the median filter was compared to the anisotropic diffusion filter. Both methods were good at removing noise but each one has shown its own advantages and disadvantages. Two notes are worth to be singled out: (a) By applying the median filter too many times the boundaries of the anatomical structures may be damaged; and (b) Anisotropic diffusion filter is much slower that the median filter.

As far as 3D segmentation of the multiple anatomical structures in the lumbosacral region is concerned, two methods were used: Region growing and level sets methods.

The level sets method requires initialization so a gradient magnitude filter alongside a reciprocal/sigmoid filter were used.

Both segmentation algorithms were tested using multiple 3D images. Some of them

were taken in the axial position, others in the sagittal position.

Regarding time-dependent rendering, the ability to render an anatomical structure in real-time highly depends on the pre-processing and segmentation steps. Because of this, these steps must be executed as fast as possible. Having that situation in mind, we have to recognise that sometimes it is necessary to use simpler methods for segmentation and smoothing (e.g. median filter and region growing). Moreover, the algorithm to be used may also need additional user input to further reduce the time spent during these steps. As additional input we are particularly referring to user-defined regions of interest. Somehow we have to consider the all process as a semi-automatic one.

In respect to analysing the anatomical structures, we highlight the fact that the spinal canal segmentation and the vertebral body segmentation have shown very good results. The interior back muscles were successfully segmented but if providing a region of interest area and using a level sets method. In the case of the abdominal aorta, there were some leakages. These leakages were mitigated by applying a mask filter as result of a region growing method.

6.2 Further Work

Despite some comprehensive study and analysis about segmentation of the anatomical structures has been done, further work is required to improve the algorithms. In particular, we draw attention to the following aspects:

- Some of the datasets had an issue with their intensities (the same anatomical structure would become darker). Therefore the problem of quality of datasets should be addressed. As it stands now, it poses some problems when region based algorithms are used.

- New edge enhancement techniques can be very useful. For instance, at the moment there are some issues when it comes to distinguish the inferior vena cava from the abdominal aorta.

- Volume interpolation techniques should be incorporated in the prototype. Then smoother transitions when visualizing multiple models can be obtained.

- Execution time can be an issue under certain circumstances. The more quality of results we pursue the more time the algorithms take to perform their tasks. So the use of parallel programming is worth to be considered for that matter. Not only personal computers have nowadays potentialities to be explored in that area but also some proprieties of datasets and the algorithms themselves can be adequate for parallel programming.

Bibliography

[AB94] Rolf Adams and Leanne Bischof. Seeded Region Growing. *IEEE Transaction on Pattern analysts an Machine Intelligence*, 16(6):641–647, 1994.

[AEC07] K. Gary L. Ibanez D. Gobbi F. Lindseth Z. Yoniv S. Aylward J. Jomier A. Enquobahrie, P. Cheng and K. Cleary. The Image-Guided Surgery Toolkit IGSTK: An Open Source C++ Sofware Toolkit. *Digital Imaging*, 2007.

[Blo] My Gerg Life Blog. Salt-And-Pepper noise. http://www.gergltd.com/. accessed on December 2013.

[BS09] K J. Batenburg and J. Sijbers. Optimal Threshold Selection for Tomogram Segmentation by Projection Distance Minimization. *IEEE Transactions on Medical imaging*, 28(5):676–686, 2009.

[Can86] J. Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–98, June 1986.

[CL98] Jinhai Cai and Zhi-Qiang Liu. A new thresholding algorithm based on all-pole model. *Pattern Recognition, International Conference*, pages 34–36, 1998.

[cp] Information Visualization community platform. Visualization Pipeline. http://www.infovis-wiki.net/. accessed on December 2013.

[CP80] P. Chen and T. Pavlidis. Image segmentation as an estimation problem. *Computer Graphics and Image Processing*, 12:153–172, 1980.

[CWQ+06] Ming-yuen Chan, Yingcai Wu, Huamin Qu, Albert C S Chung, and Wilbur C K Wong. MIP-Guided Vascular Image Visualization with Multi-Dimensional Transfer Function. pages 372–384, 2006.

[DeV] DeVIDE. The Delft Visualisation and Image processing Development Environment. http://graphics.tudelft.nl/Projects/DeVIDE. accessed on February 2013.

[ea13] D. Boye et al. Population based modeling of respiratory lung motion and prediction from partial information. *Proc. SPIE 8669, Medical Imaging 2013: Image Processing*, 8, 2013.

[FLT] FLTK. Fast Light Toolkit. http://www.fltk.org/. accessed on February 2013.

[FMHC07] Zhe Fang, Torsten Möller, Ghassan Hamarneh, and Anna Celler. Visualization and exploration of time-varying medical image data sets. *Proceedings of Graphics Interface 2007 on - GI '07*, page 281, 2007.

[HJJtISC13] L. Ibáñez H. J. Johnson, M. McCormick and the Insight Software Consortium. The ITK Software Guide, Fourth Edition, 2013.

[HO00] R. Hietala and J. Oikarinen. A visibility determination algorithm for interactive virtual endoscopy. *Proceedings of the Conference on Visualization (Vis) 2000*, 2000.

[Hos] Lady Willigdon Hospital. CT Scan. http://www.manalihospital.com/. accessed on December 2013.

[IGS] IGSTK. The Image-Guided Surgery Toolkit. http://www.igstk.org/. accessed on February 2013.

[IK94] T. Itoh and K. Koyamada. Isosurface generation by using extrema graphs. *Proceedings of the Conference on Visualization*, pages 77–84, 1994.

[Ima] ImageVis3D. A volume rendering program developed by the NIH/NIGMS. http://www.sci.utah.edu/cibc/software/41-imagevis3d.html. accessed on February 2013.

[ITK] ITK. Insight segmenttion and registration toolkit. http://www.itk.org/. accessed on February 2013.

[JGK98] A. Kosaka J. Gao and A. Kak. A deformable model for human organ extraction. *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, 3:323–327, 1998.

[KSSe00] Reinhard Klein, Andreas Schilling, and Wolfgang Straß er. Reconstruction and Simplification of Surfaces from Contours. *Graphical Models*, 62(6):429–443, November 2000.

[KWW] KWWidgets. A free, cross-platform and open-license GUI Toolkit. http://www.kwwidgets.org/. accessed on February 2013.

[LA98] Jianping Li and Pan Agathoklis. An efficiency enhanced isosurface generation algorithm for volume visualization. *The Visual Computer*, 13:391–400, 1998.

[LB03] A. Lopes and K. Brodlie. Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing. *IEEE Transactions on visualization and computer graphics*, 9(1), January-March 2003.

[LC87] William E Lorensen and Harvey E Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. 21(4):163–169, 1987.

[LDW08] Jia Liang, Gangyi Ding, and Yuwei Wu. Segmentation of the Left Ventricle from Cardiac MR Images Based on Radial GVF Snake. *2008 International Conference on BioMedical Engineering and Informatics*, pages 238–242, May 2008.

[Lev88] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[LL94] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, pages 451–458, 1994.

[MeV] MeVisLab. Medical Image Processing and Visualization. http://www.mevislab.de. accessed on February 2013.

[MITa] MIT/CSAIL. MIT Computer science and artificial intelligence laboratory. http://www.csail.mit.edu/. accessed on December 2013.

[MITb] MITK. The Medical Imaging Interaction Toolkit. http://www.mitk.org/. accessed on February 2013.

[ML05] Kwan Liu Ma and Eric B. Lum. Techniques for visualizing timevarying volume data. *The Visualization Handbook*, pages 511–531, 2005.

[oCSI] Cardiff School of Computer Science and Informatics. Region Growing. http://www.cs.cf.ac.uk/Dave/Vision_lecture/node35.html. accessed on December 2013.

[oE] IOWA State University/College of Engineering. Segmentation. http://www.engineering.iastate.edu/. accessed on December 2013.

[OS83] L. O'Gorman and A. C. Sanderson. The converging squares algorithm: An efficient multidimensional peak picking method. *International Conference on Acoustics, Speech, and Signal Processing*, 8:112–115, May 1983.

[OS88] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations". *J. Comput. Phys.*, 79:12—-49, 1988.

[Osi] OsiriX. OsiriX Imaging Software. http://www.osirix-viewer.com/. accessed on December 2013.

[oUDoB] The University of Utah Department of Bioengineering. Region Based Segmentation. http://www.bioen.utah.edu/wiki/index.php?title=Region-Based_Segmentation. accessed on December 2013.

[oUSoc] The University of Utah School of computing. Image Processing. http://www.cs.utah.edu/~croberts/courses/cs7966/project3/. accessed on December 2013.

[PBM03] Hyunjin Park, Peyton H Bland, and Charles R Meyer. Construction of an abdominal probabilistic atlas and its application in segmentation. *IEEE transactions on medical imaging*, 22(4):483–92, April 2003.

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 12:629–639, 1990.

[PSY97] Carrye Wilkins Prasanna Sahho and Jerry Yeager. Threshold selection using renyi's entropy. *Pattern Recognition*, 30:71–84, 1997.

[Qt] Qt. A cross-platform application and UI framework. http://qt.digia.com/. accessed on February 2013.

[RAW94] L. Hong A. Kaufman H. Pfister C. Silva L. Sobierajski R. Avila, T. He and S. Wang. VolVis: A diversified volume visualization system. *Proceedings of the Conference on Visualization*, pages 31–39, 1994.

[Ren] Illustrative Volume Rendering. Point-based surface rendering. http://bmia.bmt.tue.nl/Research/MVIAV/IVR/ivrsb/index.php?Page=Stippling. accessed on December 2013.

[RGY98] Ch. Henn R. Grzeszczuk and R. Yagel. Advanced Geometric Techniques for Ray Casting Volumes. *SIGGRAPH 98 course 4*, 1998.

[RMV95] J. A. Sethian R. Malladi and B. C. Vermuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. on Pattern Analysis andMachine Intelligence*, 17(2):158—-174, 1995.

[SCI] SCIRun. A problem solving environment. http://www.sci.utah.edu/cibc-software/scirun.html. accessed on February 2013.

[Seg] Seg3D. A free volume segmentation and processing tool. http://www.sci.utah.edu/cibc-software/seg3d.html. accessed on February 2013.

[SH99a] Philip Sutton and Charles D. Hansen. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree. *IEEE Visualization '99*, pages 371–378, 1999.

[SH99b] Philip Sutton and Charles D. Hansen. Isosurface extraction in timevarying fields using a temporal branch-on-need tree (T-BON). *Proceedings of IEEE Visualization '99*, pages 147–153, 1999.

[SH01] L. G. Shapiro and R. M. Haralick. *Computer Vision*. 2001.

[SS04] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):220, 2004.

[SSGM02] M. A. Abidi O. Karakashian S. S. Gleason, H. Sari-Sarraf and F. Morandi. A new deformable model for analysis of X-Ray CT images in preclinical studies of mice for polycystic kidney disease. *IEEE Transactions on Medical Imaging*, 21(10):355–366, 2002.

[SSM13] C. Oliveira S. Silva, A. Teixeira and P. Martins. Towards a Systematic and Quantitative Analysis of Vocal Tract Data. *Proc. InterSpeech 2013*, pages 1307–1311, August 2013.

[TFCJ94] C. J. Taylor T. F. Cootes, A. Hill and J.Haslam. The use of active shape models for locating structures in medical images. *Image and Vision computing*, 12(6):355–366, 1994.

[Tsi95] J. N. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[vdB] Walter van den Broek. New Kind of Brain Stimulation for Treatment Resistant Depression. http://www.shockmd.com/. accessed on December 2013.

[vddcAUp] Cat vs. dog drawings categorization. A University project. Canny edge detector. https://jeena.net/catdog. accessed on December 2013.

[VTK] VTK. The Visualization Toolkit. http://www.vtk.org/. accessed on February 2013.

[Wika] Wikipedia. Marching Cubes. http://en.wikipedia.org/wiki/Marching_cubes. accessed on December 2013.

[Wikb] Wikipedia. Volume Rendering. http://en.wikipedia.org/wiki/Volume_rendering. accessed on December 2013.

[WT88] M. Kass. A. Witkin and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, pages 321–331, 1988.

[WW91] Lee Alan Westover and Lee Alan Westover. SPLATTING : A Parallel , Feed-Forward Volume Rendering Algorithm SPLATTING : A Parallel , Feed-Forward Volume Rendering Algorithm by. 1991.

[wxW] wxWidgets. Cross-Platform GUI Library. http://www.wxwidgets.org/. accessed on February 2013.