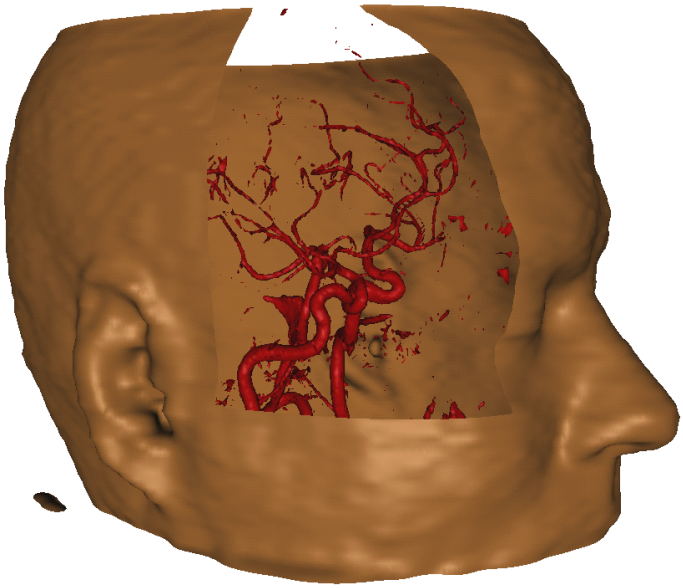
Aarhus Universitet
Matematisk Institut
Datalogisk Afdeling
Ny Munkegade
8000 Aarhus C

May 27th 1998

Towards Automatic Segmentation, Visualization and Animation of Cerebral Arteries.

Master Thesis in Computer Science.



Christian Vandsø Andersen *890634*
Jesper Rene Frandsen *880531*

Advisor: Jørgen Lindskov Knudsen

Contents

1 Introduction

The purpose of this master thesis is to explore the area of medical imaging while combining 2D and 3D graphics. Neither of us had any prior experience in medical imaging, and found this an exciting area to explore. An important motivation for this work is the fact that the research is useful in the medical world as well as in the graphics area. We will aim to use both traditional image processing methods along with novel combinations of existing methods. In November 1997 we contacted *Aarhus Kommunehospital*, a hospital known for its research in medical imaging concerning a possible cooperation. In January 1998 we had reached a sound problem statement as a basis for our thesis. The problem is also the title of the thesis: *Towards Automatic Segmentation, Visualization and Animation of Cerebral Arteries.* This had, to the best of our knowledge, not been accomplished yet. This thesis is the result of six months of research in the field of medical imaging.

1.1 Problem statement

Neurosurgery is immensely complicated, and surgeons are faced with a number of complications which makes their job difficult:

- The brain is completely surrounded by bone, which makes the approach to the surgical target more difficult than for surgeries on other parts of the body.
- Critical structures like blood vessels limit the choice of possible approaches to reach the target.
- Tissue to be manipulated or removed is not always visually distinguishable from healthy tissue.

In a number of brain pathologies such as brain tumors, arterio-venous malformations (AVM) and aneurisms, surgery is necessary to cure the patient. Performing surgery, it is essential not to damage normal tissue, as this may ultimately cause patient disability or even death. A number of factors are therefore taken into account when planning brain surgery:

- The exact location and extent of the brain pathology must be known. For example, leaving an active tumor residue in the patient will cause early relapse and poor patient outcome. This information is, in most cases, obtained from 'structural' Magnetic Resonance (MR).

- The exact location of vessels must be known. Vessels in relation to the diseases mentioned above ruptures easily. Bleeding is extremely serious, as it is difficult to stop in the small space available during surgery. Also, blood left after surgery causes vasospasms and thereby ischemia in the days following surgery. Therefore, severe symptoms may develop after surgery in which bleeding has occurred. In the cases of AVM's and aneurisms, the disease is located in the vessels themselves, and the necessity to know vascular morphology is therefore important in planning surgery. The vascular structure is often visualized by means of MR Angiographies (MRA).

- Lastly, the location of normal brain tissue in relation to the pathology is important in planning surgery. If brain areas responsible for the patients speech or movement are damaged, severe disability will follow. Therefore, important brain structures must be localized, either from structural MR images or from functional images. The latter are made by activating brain regions externally (e.g. asking the patient to move the hand) and locating brain changes in activity by MR or PET. This results in coregistered images with high signal intensity at the location of the tissue one wants to spare.

The ability to visualize cerebral vessels along with either images of anatomy, pathology or coregistered images of brain function are therefore of great importance to plan the route of access in brain surgery in such a manner that complications are minimized. The visualization should be based on images with high information content and allow the surgeon to visualize structures in three dimensions in order to give the user full information of the structures that are encountered in a given access route.

Successful implementation of such an approach will serve to minimize the number of complications due to surgery. Furthermore, full pre surgical visualization will minimize the invasiveness of surgery, as smaller access routes can be used.

This project aims to provide the surgeon with a pre surgical planning tool which is able to visualize the blood vessels by automatically producing a 3D model of the cerebral arteries and their position relative to the head from a set of MR brain scan images. The model can be rotated, moved and otherwise animated. The segmentation and visualization process should be, if not real-time, then reasonably fast.

The motivation for the automatic process is that in a clinical environment, user interaction should be kept at a minimum. The surgeon should not have to think about computer science while performing brain surgery, and the assistants should not need to know much about computers to operate the equipment.

It is vital that the segmentation is correct. Since the result is used as a roadmap for the surgeon, a missing vessel can have crucial consequences for the patient.

On the other hand, a large number of artifacts could complicate the operation unnecessary.

1.2 Overview

From the moment the data is loaded to the final 3D output, it will undergo different types of treatment:

1. Preprocessing. The noise in the image needs to be reduced before commencing segmentation.

2. Segmentation. The aim of the segmentation part is to isolate the blood vessels from the rest of the data. To improve the situational awareness we have furthermore segmented the head shape to show interposed on the vessels.

   The segmentation methods used are

   (a) Thresholding

   (b) 3D Hysteresis and connectivity

   (c) Mathematical morphology

   Several methods has been considered. Besides the ones we use, a short presentation of the considered methods will be given in the segmentation section.

   Mathematical morphology is a valuable tool in the segmentation process. To give the reader the best introduction to this area chapter 5 is devoted to mathematical morphology and its uses in image processing.

3. Visualization. We have implemented two methods for solid volume rendering and isosurface generation: The traditional marching cubes algorithm and the more recent marching tetrahedrons. We give a presentation of the pros and cons of generating isosurfaces using these approaches.

Our research yielded a working application which proves that it is indeed possible to make a fast and automatic segmentation and visualization of the blood vessels. In chapter 9 we give a presentation of the program A.S.A.P (**A**utomatic **S**egmentation and **A**nimation **P**rogram).

After the conclusion and discussion about further work, a short introduction to basic MRI physics and explanation of the various terms are given.

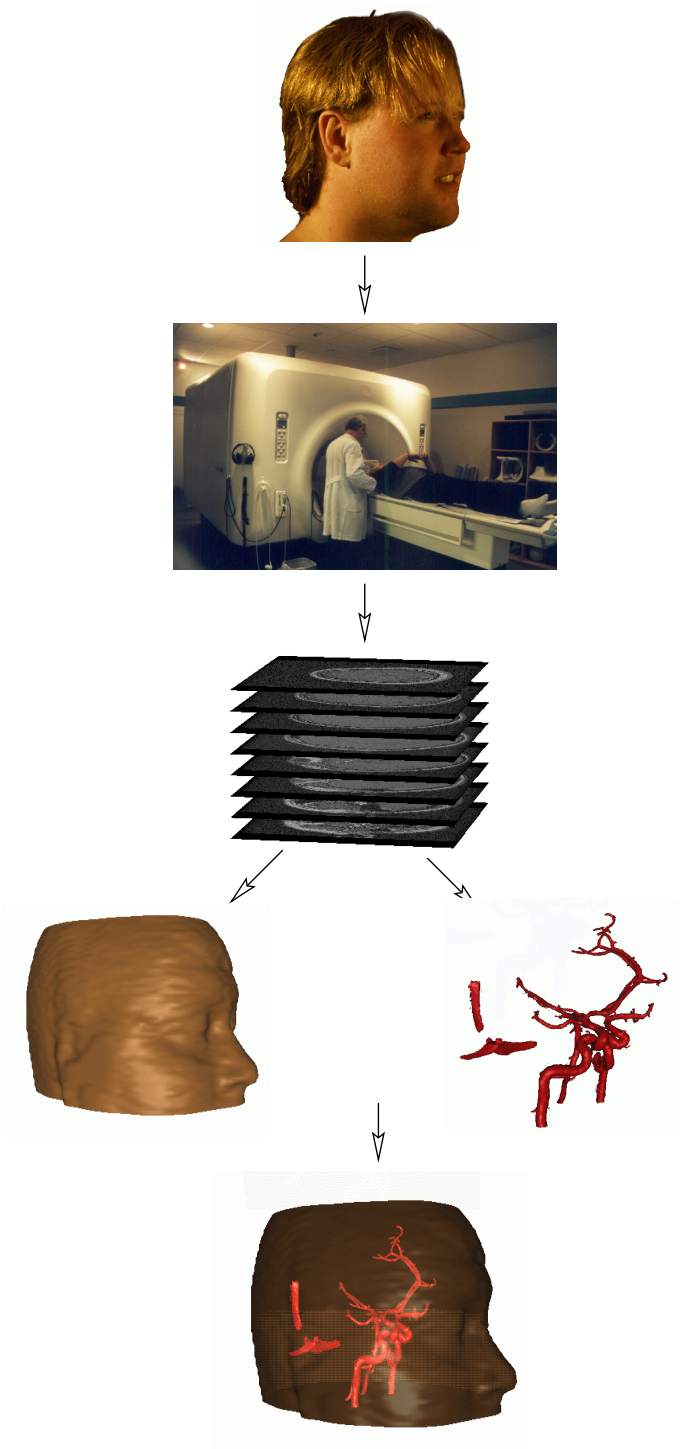


Figure 1: The process flow; from scanner to final result.

1.3 Results

We have come very close to the goal of automatic segmentation and visualization. The segmentation and visualization of both the vessels and the head is fully automatic and very fast. The head shape, although not as important as the vessels, is very accurately segmented.

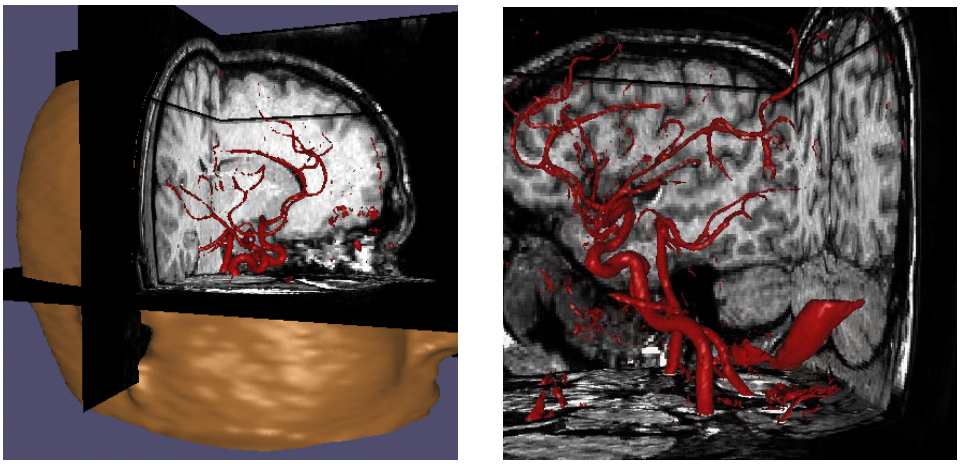


Figure 2: Left. A corner of the skull is removed to show the vessels. Right: A vessel tree and the anatomy of the brain.

These examples were made on an SGI Octane and took less than 2 minutes from the time the data was loaded to the 3D model was ready. This includes both the segmentation and the surface generation. To test the usability on a low-end system, we also ran the test on a standard PC running Linux 2.0.32 equipped with a 120 Mhz Pentium CPU and 128 Mb RAM. The data was a 20 Mb scan with 256×256 resolution and 156 slices, and the table below shows that even a low-end system is able to run our program remarkably well.

Machine	Memory (Mb)	Time (Sec)
SGI Octane	256	108
SGI O2	128	122
Intel Pentium 120 Mhz	128	315

Speed comparison on different CPU's

In figure 2 (left) a corner of the skull is removed to show the inside of the head. This resembles actual surgery where the entrance hole sometimes is made

bigger than necessary to check the surroundings. In figure 2 (right) the head shape is removed and only the vessels and the anatomy is shown.

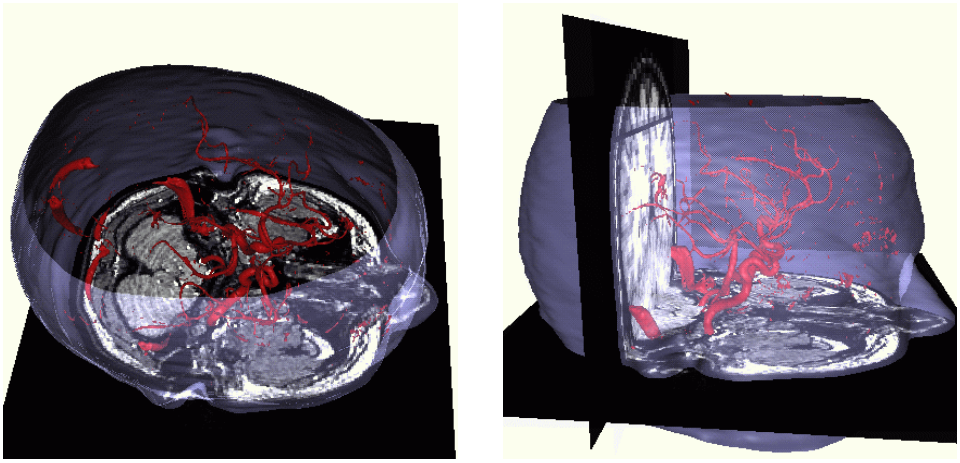


Figure 3: Left: Vessel tree seen from above. Right: Transparent head with vessels showing.

The visualizations shown here can be rotated in three dimensions and the planes showing the anatomy can be positioned freely along their respective axis. The head shape can be viewed as either transparent or opaque or a combination of both.

1.4 Acknowledgments

We would like to thank our advisor Jørgen Lindskov Knudsen for valuable support and encouragement. Leif Østergaard M.D. from Aarhus Kommunehospital has been our contact to the medical world and has patiently answered all our questions about the brain anatomy as well as providing us with the research data. Ryan Sangild, Skejby hospital, has kindly made several MR scans of his head available to us. Chief physician Finn Taagehøj from Skejby hospital operated the MR scanner while we had our heads scanned and explained the basic usage of the scanner. Flemming Andersen and Anders Rodell, both PhD students of computer science, helped us along in the early stages with sound advises and ideas. Steffen Ringaard, PhD in physics, provided us with the necessary knowledge about the error sources of the MR scanner. Finally we would like to thank Søren Dalsgaard, PhD student in history, for sound ethical discussion and proof reading of the thesis.

2 Data acquisition

Depending on the nature of a specific problem, the surgeon uses different methods of acquiring information about the physiology of the brain. Even today, these non-processed images are often all the surgeon depend on when planning the surgery.

2.1 X-ray

Historically we start with the X-ray, which was discovered in 1895 by the German physicist Wilhelm Conrad Roentgen. X-ray was for a long time the only way of "looking" into the head without actually opening it.

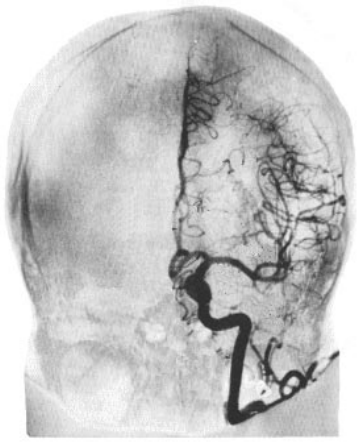


Figure 4: An X-ray contrast image of the blood vessels.

By injecting a contrast agent into the blood stream it is possible to image the blood vessels. This is not without risk for the patient, and the use has been discouraged in recent years.

Apart for the risk, this technique has the limitation of not allowing unambiguous localization of points in three dimensional space since it provides only a projection of the vessels along one dimension. It is not possible to determine the positions of structures perpendicular to the imaging plane.

2.2 Computerized Tomography (CT)

A recent X-ray device offers clear views of all parts of the anatomy. Called the body scanner, or computerized tomography (CT) scanner, it rotates 180 degrees around a patients body, sending out a pencil-thin X-ray beam at many different points. Crystals positioned at the opposite points of the beam pick up and record

the absorption rates of the varying thicknesses of tissue and bone. Figure 5 shows an example of a CT scan of the brain.

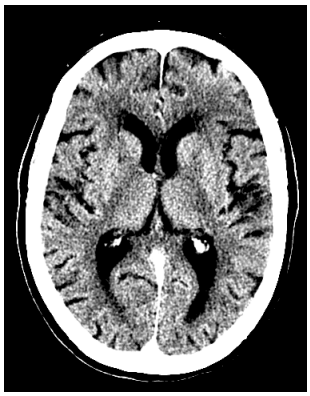


Figure 5: CT scan of the brain.

The CT scanner is particularly useful for scanning bone structures, but has its limitations when scanning living tissue. Today this is, along with MR, one of the most commonly used scanning methods for pre surgical planning.

2.3 Positron Emission Tomography (PET)

In this type of scan the patient is injected with a positron emitting agent. The positrons collide with electrons in the body and produce photons which are tracked by the Positron Emission Tomography (PET) scanner.

PET is especially useful for diagnosing brain tumors and the effect of strokes on the brain, as well as various mental illnesses. PET scanners are also used in brain research and the mapping of brain functions, as areas of high intensity reflect brain activity.

2.4 Single-Photon Electron Computed Tomography (SPECT)

Single-Photon Emission Computed Tomography (SPECT) is a technology used in nuclear medicine in which the patient is injected with a chemical tracer that emits gamma rays. The radioactivity is collected by a gamma camera.

The gamma camera consists of two massive cameras opposite each other which rotate 180 degrees around the center axis. When the gamma camera rotates around the body, it stops at certain intervals to collect data. Each of the cameras collects a matrix of values which correspond to the number of gamma counts detected in that direction and angle.

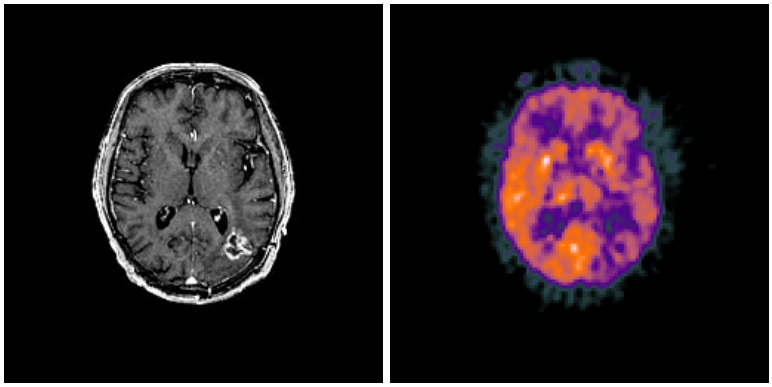


Figure 6: An MR scan of the brain and the corresponding PET scan.

SPECT is often used on cancer patients because the method is precise to the extent of letting doctors detect abnormalities in the early stages of disease development.

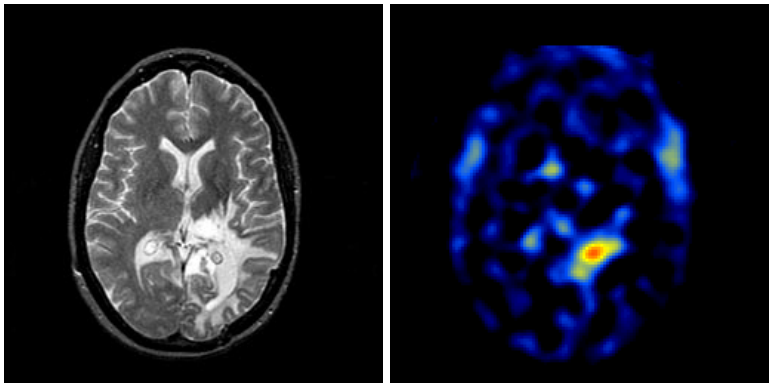


Figure 7: An MR scan of the brain and the corresponding SPECT scan.

2.5 Magnetic Resonance Imaging (MRI)

While CT scanning is excellent for bone structure imaging and PET and SPECT are good for information of the brain functionality, Magnetic Resonance Imaging (MRI) is superior when it comes to detailed anatomical information and when focusing on blood vessels.

The spatial resolution is far better with MR, and the technology allows scans focusing on blood without injecting a contrast agent. For an overview of the

physics behind MRI, please refer to appendix A. Because it does not make use of ionizing agents, MRI is risk free except for patients with cardiac pacemakers, patients with inner ear transplants, and patients with aneurysm clips in their brains. They all have metal in the body that will be affected by the strong magnetic field.

In current medical practice, MRI is preferred for diagnosing most diseases of the brain and central nervous system.

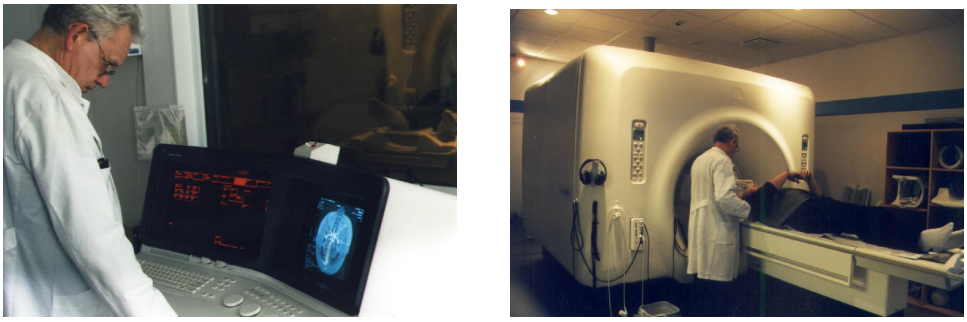


Figure 8: Left: Chief physician Finn Taagehøj, Skejby hospital, at the control desk. Right: The MR scanner we got all our data from - model Signa 1.5 Tesla GE.

The subject is placed in the scanner (figure 8), and a preview scan is made. From that preview, the area to be scanned is determined. Figure 9 shows the preview prior to the scanning. Notice the grid that overlays the brain. The rows in this grid determine the *slabs*. The MR-scanner is only capable of scanning approximately 5-6 cm of the head at a time. Therefore when scanning an entire head, it must be done in turns. These sections are called slabs. After the scan, the slabs are combined to create the final image.

The result of the MR scan is a number of *slices* (figure 10). Each slice is a grayscale image in 2D, although it actually has a thickness. The typical resolution is 512×512 (or 256×256) per slice with 4096 levels of gray (12 bit colordepth). The number of slices varies with the object of interest. For a complete brain scan approximately 130 slices are scanned. A resolution of this size gives a spatial resolution of $0.5mm \times 0.5mm$ per slice and a distance of $1mm$ between the slices.

After the scan is completed, it is digitally stored in the scanner. The neurologist can then connect to the scanner and download the raw data to his workstation. The data is typically assembled into a single file. Several file formats exists, but a very common format, which is also the format we use, is the Minc format from McGill University. It stores the data in sequence with optional compression and offers a header where subject and technical information can be stored.

It is common to use a Maximum Intensity Projection (MIP) of the slices to

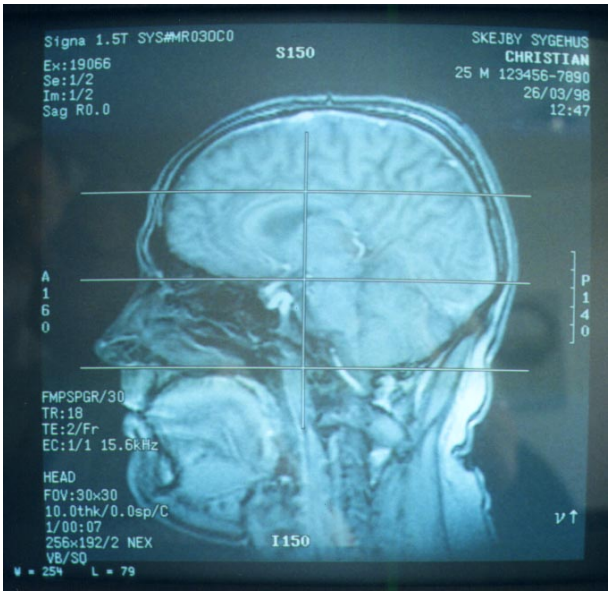


Figure 9: A photo of the preview scan.

set a diagnose. The maximum intensity encountered along a projection ray is depicted on an image plane (see figure 11). Albeit a simple method, the information presented is quite good, and today this is often the only image the surgeon uses when planning an operation.

However, the MIP has its limitations. Like X-ray, there is no depth information in the picture and no stationary tissue to use as reference points. Small vessels and low intensity vessels are often lost in the process because fat or other high intensity tissue overlap the vessels.

The type of images we focus on in this thesis, are those in which the cerebral arteries are shown. This is accomplished by using a scantype that detects flows. Since arteries and veins flow with different velocity, the scanner can distinguish between them and focus only on the arteries. One vein in the neck will, however, often show because the sensitivity to flow is high at that certain points. The name of this scanning type is “flow weighted T1”. Another type called T2 can be weighted for blood and will show all the vessels (both arteries and veins). To depict the anatomy, T1 scans are used. Appendix A describes how the scanner distinguishes between the different tissue types.

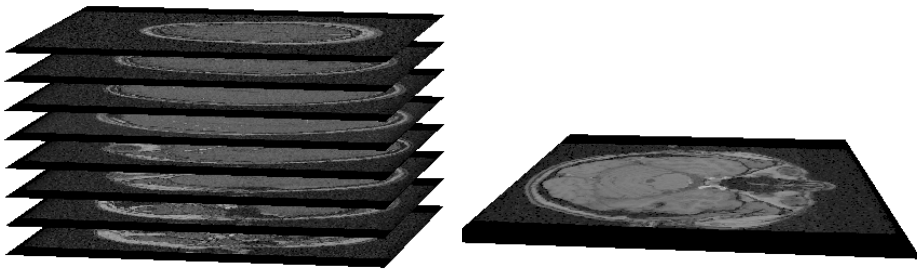


Figure 10: Left: The MR scans the head in slices. Only selected slices are shown here for clarity. Right: A typical slice.

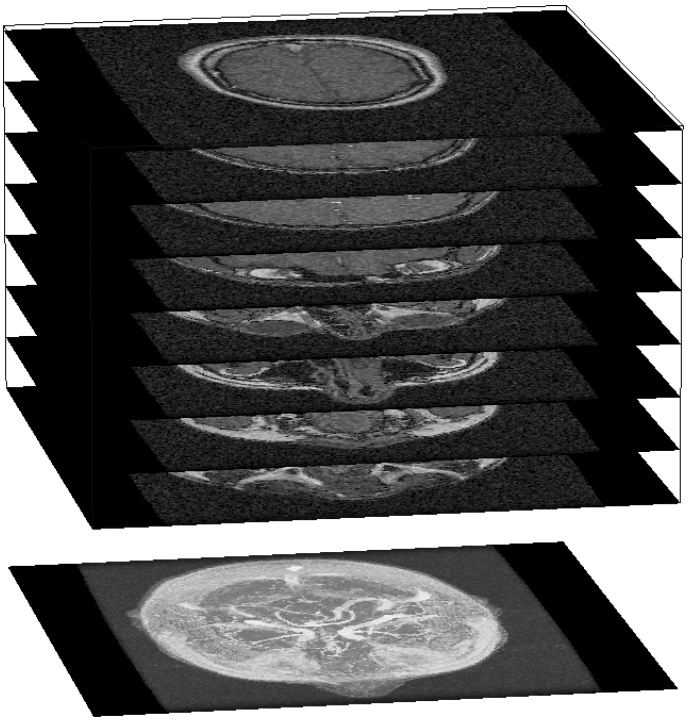


Figure 11: A Maximum Intensity Projection.

3 Preprocessing

The preprocessing has three objectives.

1. Making the images more suitable for segmentation
2. Making the data more suitable for visualization
3. Enhancing the visual appearance

The preprocessing is done on a single slice at a time. One of the purposes of the preprocessing tools is to ease the segmentation process by enhancing the regions of interest and reduce data of no interest. The preprocessing involves standard operations such as Gaussian blur and median filtering.

3.1 Image processing

The preprocessing has been divided into two sections. The techniques described in this section operates directly on the image based solely on the spatial position of the pixels and their intensities. The next section deals with analysis of the image intensity distribution and operates on the histogram.

3.1.1 Kernels

Some of the functions mentioned in this section can be approximated with discrete values called kernels. A short introduction to kernels is given here, but please refer to [19] and [15] for a thorough introduction to kernels.

A kernel is basically an $n \times n$ matrix. It is applied to each pixel in the image as shown in equation 1. Let dst be the resulting pixel, $kernel$ the filter to be applied and $image$ a function that returns the pixel value at (i, j) + an offset.

$$dst = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} kernel[i, j] * image(i, j)}{\sum_{i=1}^{n} \sum_{j=1}^{n} kernel[i, j]} \tag{1}$$

From equation 1 we see that the destination pixel is a function of all its neighbours (up to $n/2$ pixels away). Ideally, n should be infinite, but most often a kernel of limited size is chosen, depending on the function.

Although we use kernels as a method, the values are not predetermined, but calculated on run time. This enables us to use kernels of arbitrary size.

3.1.2 Gaussian smoothing

The Gaussian smoothing, also known as Gaussian blur, is widely used in the world of discrete images. It takes an area of the image usually consisting of different pixels and blurs the area, thus simulating a continuous image. The idea is to weight the surrounding pixels corresponding to a Gaussian distribution. Distant pixels adds less to the result than pixels close to the center of the operator. The Gaussian formula is given in equation 2.

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

The only parameter of the Gaussian filter is the standard deviation σ. It is proportional to the size of the neighbourhood on which the filter operates. That is, the size of the matrix.

A 3D plot, as shown in figure 12, shows clearly how the weight is distributed throughout the kernel.

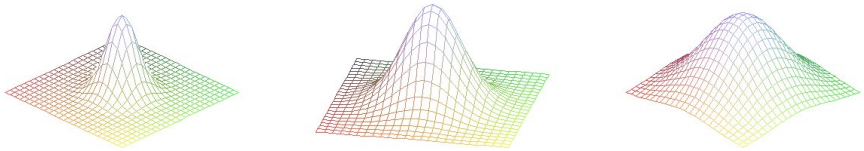


Figure 12: Gauss curves for $\sigma = 2$, $\sigma = 3$, $\sigma = 5$.

3.1.3 Edge enhancing

In order to compute an edge enhancing function, a mathematical representation of an edge must be found. Assume the image is generated by an image function f. The first derivative of the image function should have a maximum at the position corresponding to the edge in the image. But instead of searching the derived function for extremes, it is much easier (and faster) to search the second derivative for zeros (see figure 13).

The gradient operator known as ∇ is the second partial derivative of the function. The Laplacian operator ∇^2 reflects edge magnitudes without regard to orientation and is shown in equation 3.

$$\nabla^2(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} \tag{3}$$

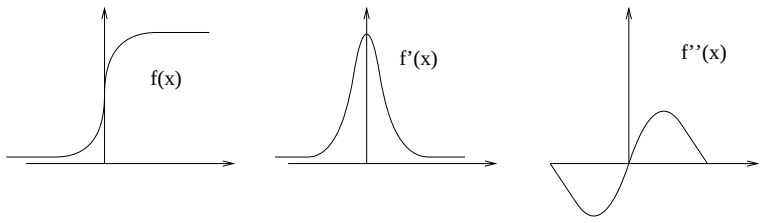


Figure 13: First and second derivative of an edge.

The Laplacian can be approximated in the discrete world by a convolution sum, i.e. it can be represented by a kernel. However, the approximation is highly sensitive to noise. A common way of solving this problem is to apply a Gaussian smoothing first.

Enhancing edges is thus often a two step procedure.

1. Application of Gauss smoothing.
2. Application of the Laplacian operator.

This does not mean that two filters are applied, because a single mask can be created to do both operations simultaneously. Equation 4 shows the second derivative of the Gauss equation. This function can be approximated and used in a single pass, as an edge enhancing kernel. To detect the edges and not just enhancing them, the result of applying the Laplace of Gaussian can be searched for zeros (also known as Zero Crossing).

$$\nabla^2 G(x,y) = \frac{1}{\sigma^2}\left(\frac{x^2+y^2}{\sigma^2} - 1\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4}$$

Sonka et. al [21] call this method of finding edges very robust and they give a reference to a neurophysician who claims that this is how the human eye detects edges as well. Figure 14 shows the difference between a standard edge detector compared to Gauss/Laplace.

3.2 Median filtering

The previously mentioned image operations all suffer from the same weakness. The new pixel value is the result of an arithmetic computation, thereby introducing a possible error source in terms of precision. The median filter is defined by a logical ranking of element values. This means that median filtering operates on images of arbitrary pixel precision.

Figure 14: Original image. Conventional edge. Gauss/Laplace edge.

The median of an array A with n elements is defined as

$$Med(A[n]) = A_{sorted}[\frac{n+1}{2}] \tag{5}$$

We have implemented two types of filtering based on the median. The first operates on n x n matrixes constructed of n parallel scanlines of length n. The second forms two arrays as shown in figure 15. The process is

1. Create two arrays A_1 and A_2
2. Calculate $M_{A_1} \leftarrow median(A_1)$ and $M_{A_2} \leftarrow median(A_2)$
3. Calculate $result \leftarrow median(M_{A_1}, M_{A_2}, CurrentPixel)$

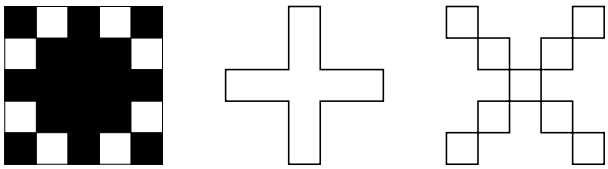


Figure 15: a) Considered pixels. b) Construction of A_1. c) Construction of A_2.

Notice that the center pixel is included three times. This makes the filter more stable. That is, chances are the pixel is not affected, unless it differs a fair amount from its neighbours. This method is preferred if the image contains fine lines, as they are preserved this way.

Median filtering is excellent to remove isolated pixels, also known as shot noise, while preserving the overall image quality. Figure 16 shows an application of the standard median filter.

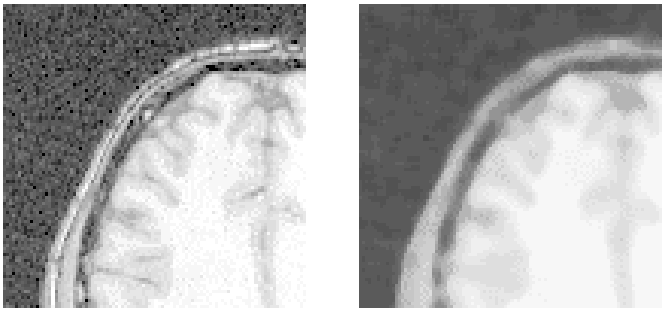


Figure 16: (Left) Image before median filtering with shot noise. (Right) Image after median filtering.

3.3 Histogram techniques

Automatic contrast enhancement of images is widely done by manipulating the histogram. Two different approaches are *Histogram stretching* and *Histogram equalization*. We will use the histogram techniques later when we need to show both the original MR scan and the rendered blood vessels.

Definition of a *histogram*: Let $[0, k]$ be the input grey scale and define the histogram as $H(p)$, where $p \in [0, k]$.

$$H(p) = \frac{\#pixels\ with\ intensity\ p}{xsize \times ysize} \tag{6}$$

Equation 7 is called the cumulative histogram for the image. We use the cumulative histogram extensively to determine the amount of pixels below a certain threshold. This is further discussed in the segmentation section.

$$L(p) = \sum_{i=0}^{p} H(i) \tag{7}$$

3.3.1 Histogram stretching

The histogram stretching algorithm *stretches* the dynamic range $[t_1, t_2]$ to the entire range (figure 17). Where to set the boundaries depends on the position of the features in the histogram. Since all information below the lower bound and above the upper bound is removed to give space to the stretching, this method should be used with care. The algorithm works in three steps:

1. Let $t_1 = max(p)|L(p) < lowerbound.$
   Let $t_2 = min(p)|L(p) \geq upperbound.$

2. Map all pixels in the image to the scale $[t_1, t_2]$, by setting all pixels with intensity below t_1 to t_1 and pixels with intensity above t_2 to t_2.

3. Map all pixels (now in the range $[t_1, t_2]$) to use the entire dynamic range, where p is the original intensity and $p_{stretched}$ is the new:

$$p_{stretched} = \frac{k(p - t_1)}{t_2 - t_1} \tag{8}$$

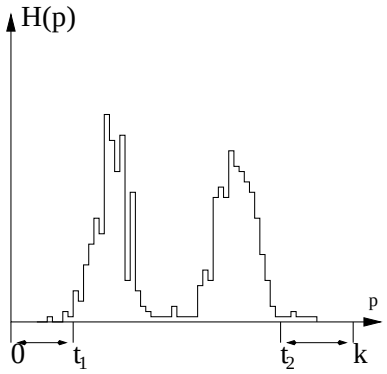


Figure 17: Setting the boundaries.

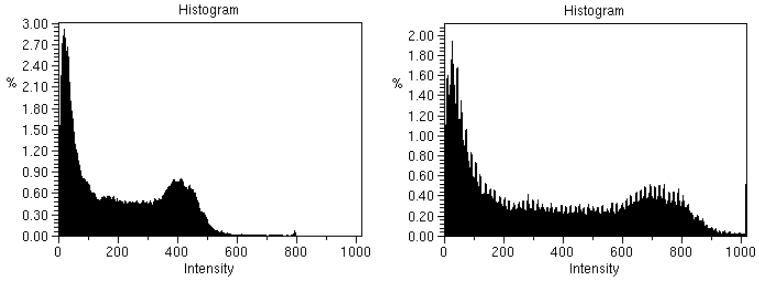


Figure 18: Left: The histogram before stretching. Right: The histogram after stretching

3.3.2 Histogram equalization

Histogram equalization is based on the assumption that the ideal histogram is flat with equally distributed brightness levels over the entire brightness scale (see figure 19).

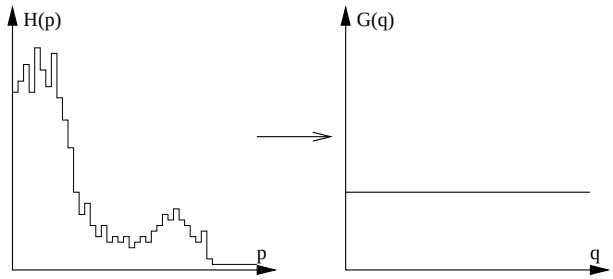


Figure 19: Histogram equalization.

As before, let the $H(p)$ be the input histogram and $[0, k]$ the input gray scale.

The equalization is done by a monotonic transformation $\mathcal{T}$ that returns a uniform histogram $G(q)$ where $q = [0, k]$.

Because $\mathcal{T}$ is monotonic, the number of pixels is the same before and after the transformation (equation 9).

$$\sum_{i=0}^{k} G(i) = \sum_{i=0}^{k} H(i) = 100\% \tag{9}$$

Since the equalized histogram consists of k equal values, the equalized histogram $G(q)$ corresponds to the probability function f with constant value.

$$f = \frac{1}{k} \tag{10}$$

When dealing with discrete digital images, the equalized histogram will not be a monotonic function, but rather an approximation.

$$\sum_{0}^{q} \frac{1}{k} = \frac{q}{k} = \sum_{i=0}^{p} H(i) \tag{11}$$

We have already seen the last sum in equation 11 as the cumulative histogram $L(p)$ in equation 7. The discrete approximation of the transformation can be written as equation 12, which is simply a constant k (the maximum intensity) multiplied with the result of a table lookup, since $L(p)$ can be computed by examining all pixels once.

$$\mathcal{T}(p) = k * L(p) \tag{12}$$

Figure 20 illustrates an application of histogram equalization. The histogram is shown in figure 21.

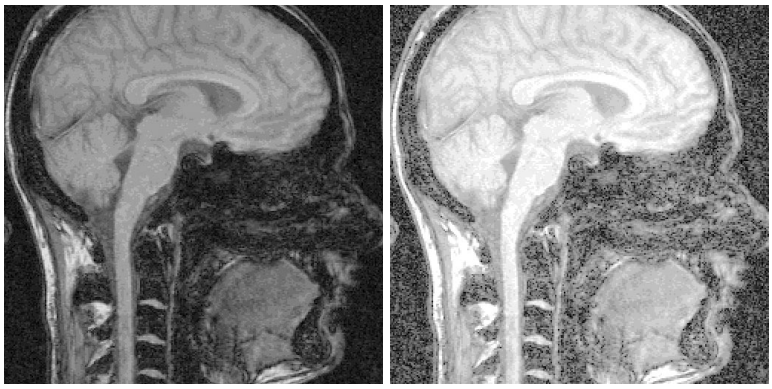


Figure 20: Left is original image. The right is image after equalization.

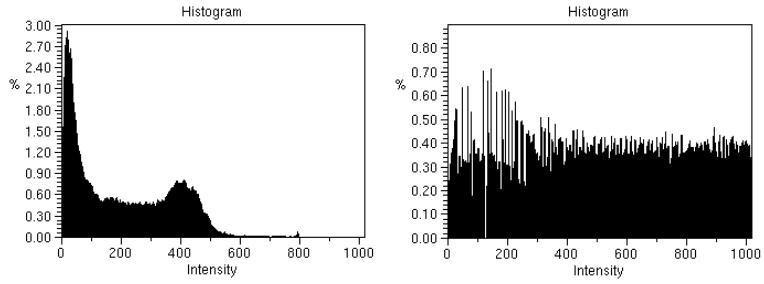


Figure 21: Left is the histogram before equalization. To the right the histogram after equalization.

4 Segmentation

Segmentation is a fundamental problem in image processing, especially in the field of medical imaging. Most of the further analysis rely on the efficiency of the segmentation procedure. The presence of noise, the variability of the background and the low and varying contrast of vessels make the segmentation quite difficult.

Image segmentation make use of many techniques, among which are: thresholding, edge based segmentation, region based segmentation, mathematical morphology, neural networks, or a combination of these techniques.

Depending on the object of interest, we use different segmentation methods. The head shape is isolated using mathematical morphology, while we use an extended version of hysteresis combined with mathematical morphology to isolate the blood vessels.

We will start by giving a review of considered segmentation methods, their strengths and limitations and will motivate our choice of segmentation methods for the blood vessels. Due of the versatility of mathematical morphology, we have dedicated the next chapter in its entirety for covering this technique.

4.1 Threshold based segmentation

Thresholding is the simplest form of segmentation when done on a grey level image. Given a constant threshold value, all pixels with intensity below this threshold will be treated as background pixels. The idea of thresholding is that it is possible to differentiate between background pixels and object-of-interest pixels simply by selecting the proper threshold intensity.

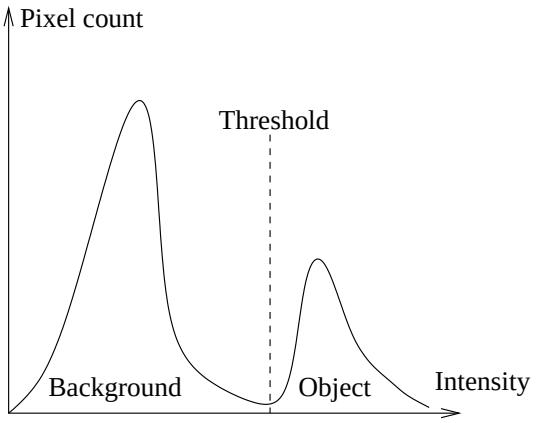


Figure 22: Simple thresholding.

4.1.1 Simple thresholding

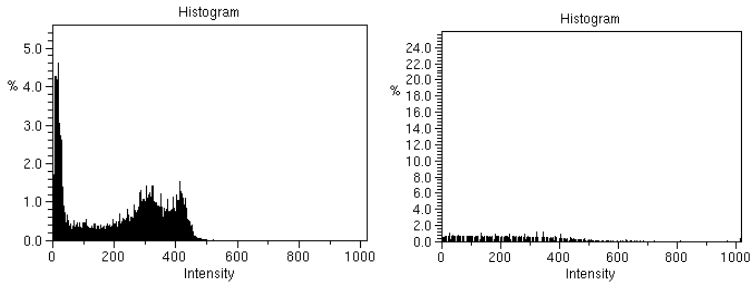


Figure 23: Two histograms of two different slices of an MR image.

The two histograms in figure 23 illustrates the difficulty in finding the proper threshold. The first histogram is almost bimodal (pixels of objects form one peak, while pixels of the background forms the second peak near zero). In the second histogram it is not easy to determine a proper threshold, since it is not bimodal. Approximately 25 % of the pixels have an intensity of zero, the rest are almost uniformly distributed over the full intensity scale.

The best way of finding the proper threshold is done by histogram analysis. Although our images are multimodal (depicting several distinct objects), we treat them as bimodal for each feature we are interested in. By doing so, we can settle with a single threshold value. Usually bimodal threshold detection algorithms find the two highest local maxima, then determine the threshold as the local minimum between these. As figure 24 illustrates this is not an optimal way of determining the threshold, since the histogram is not necessarily bimodal even if the histogram is a sum of two distributions of intensities (background and object).

4.1.2 Optimal threshold

An alternative method of finding the proper threshold is based on approximation of the histogram as the sum of two (or more) probabilities with normal distribution. The threshold is the minimum probability between maxima of normal distributions, which gives the minimum segmentation error. This method requires the brightness to be $normal$ distributed. If this assumption can not be made, a simpler algorithm is found in Sonka et al [21] which works well even if the histogram is not bimodal.

Algorithm for optimal threshold selection:

1. Let $C(p)$ be an alternative histogram for the image with input scale [0,k]:

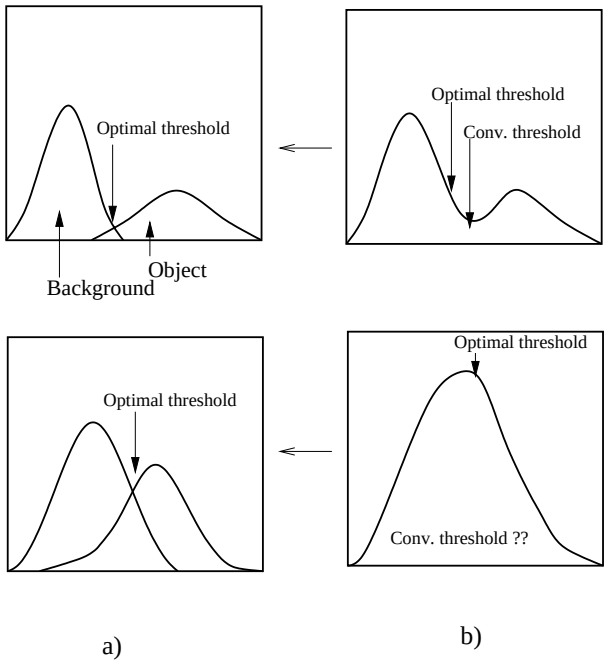


Figure 24: a) Histogram where foreground and background has separate curves. b) Real histogram with only one curve, which is the sum of the object and background.

$C(p)$ = #pixels in image with intensity p, which can be calculated by scanning all pixels once.

2. We have no knowledge of the location of the object. Therefore we approximate the initial threshold to mean of all pixels: $T^0 = mean(all\_pixels)$.

3. At step t, calculate mean background intensity $\sigma^t_{background}$ and mean object intensity σ^t_{object}. The threshold T^t that separates object from background is calculated at step $t-1$.

$$\sigma^t_{background} = \frac{\sum_{i=0}^{i<T^t} iC(i)}{\sum_{i=0}^{i<T^t} C(i)} \tag{13}$$

$$\sigma^t_{object} = \frac{\sum_{i=T^t}^{i=k} iC(i)}{\sum_{i=T^t}^{i=k} C(i)} \tag{14}$$

4. Set

$$T^{(t+1)} = \frac{\sigma^t_{background} + \sigma^t_{object}}{2} \tag{15}$$

5. If $T^{(t+1)} = T^{(t)}$ the algorithm terminates; otherwise return to 3.

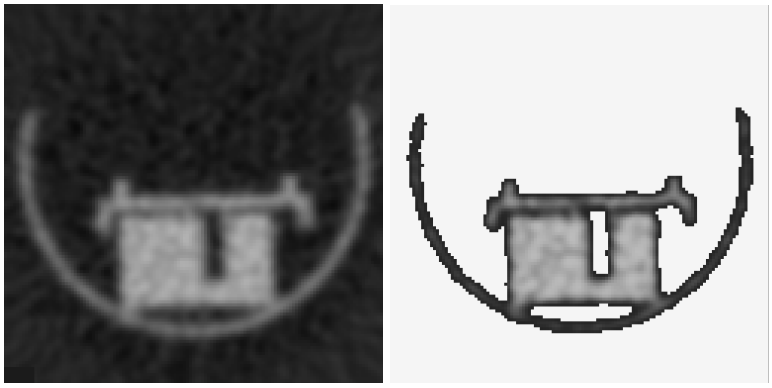


Figure 25: Left: Original picture before optimal threshold. Right: Automatically segmented object using optimal threshold.

As figure 25 shows, the optimal threshold algorithm successfully detects the image in the otherwise noisy picture. The object is a piece of plastic scanned by an MR scanner.

4.1.3 Dynamic threshold based segmentation

Until now we have used a single threshold for segmenting the data. This requires the illumination to be uniform for the complete MR dataset. This is however not the case, as the illumination can change from slice to slice. What is the best threshold for one slice, may be a less optimal threshold for the next slice.

It is therefore necessary to use an individual threshold for each slice. As the MR scans show the blood vessels with high intensity, it can be assumed that the region of interest (ROI) is the top 1% of the intensity range.

In figure 26 the maximal intensity for all slices is showed by the solid curved line. The dynamic threshold for each slice is showed with a dashed line. The single threshold T for all slices is the solid straight line. The figure clearly shows that if the illumination is non-uniform through the slices, selecting a single threshold for all slices may prove problematic. Many of the middle slices simply do not contribute with the selected threshold T.

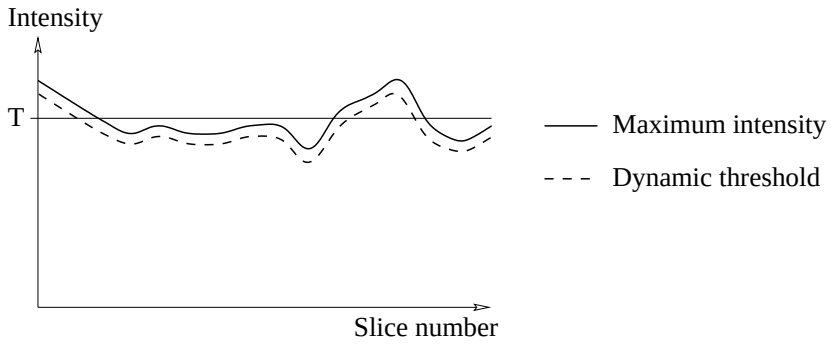


Figure 26: Comparing dynamic threshold and traditional threshold.

4.2 Seeding and region growing

Simple thresholding makes no use of the geometrical connectivity information in the image. Seeding and region growing is a technique where a set of initial pixels is created either manually selected by the user or calculated automatically. These seed pixels will usually share some property, like belonging to the same kind of tissue. The region growing technique will then add connected pixels to the set of segmented pixels, see figure 27. To be connected, two pixels must share at least one corner and have an intensity above the specified threshold. The algorithm recursively add pixels to the set of segmented pixels until no more pixels can be added to the set.

4.3 Hysteresis

In this technique a different threshold is used to grow a region than to select the seed pixels. Otherwise this technique works like the *Seeding and region growing* technique. The growing threshold will typically be lower than the seed threshold, see figure 27. This technique produces segmented pixel set with more clearly defined borders. It is important that the growing threshold is selected properly for the segmented pixel set to be meaningful.

4.3.1 Extending hysteresis to volumetric data set

Normal hysteresis works on 2D images (slices). We would like the method to utilize the 3D volume information of the entire dataset. Therefore we extend the algorithm to work on the entire set of slices. As the growing goes across slices, we will use volumetric pixels or *voxels*.

Basically the idea is the same as before. A set of initial voxels grows to some set of segmented voxels.

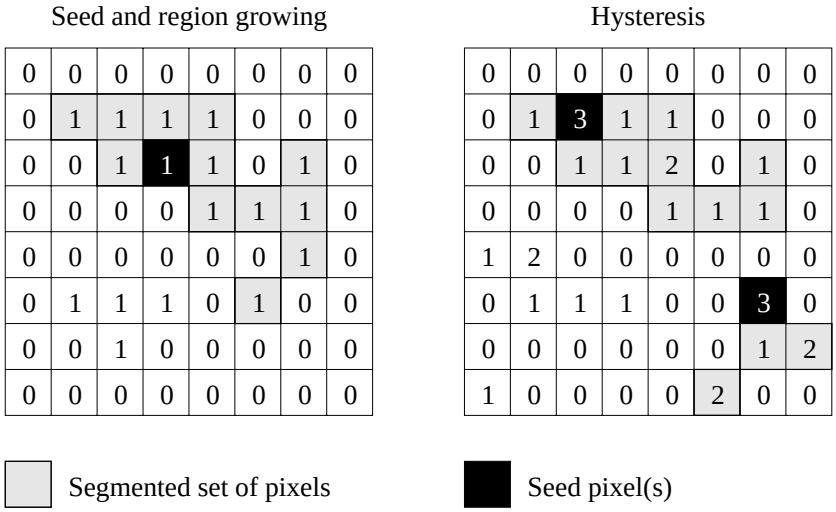


Figure 27: Illustration of Seed and region growing and hysteresis technique.

Voxels can be R-connected, where $R \in [0, 1, 2, 4]$. Two voxels are 0-connected if they share no corners, and they are 1-connected if they share 1 corner etc. Figure 28 shows the different kinds of connectivity.

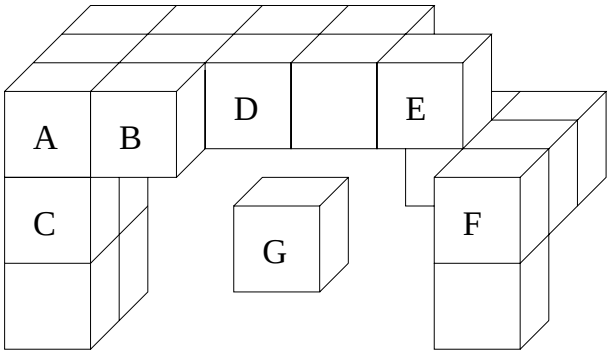


Figure 28: Illustration of R-connectivity of voxels. Voxel A and B (and A and C) are 4-connected, voxel B and D are 2-connected, voxel E and F are 1-connected and voxel G is 0-connected to all others.

The most conservative growing measure is to require 4-connectivity. 1-connectivity and 2-connectivity are more sensible to noise.

Practice has shown that this technique is excellent for finding the larger blood vessels of the brain. The selection of the growing threshold is crucial. If the threshold is too low, tissues with high intensity will be segmented also. We will

later discuss how to handle this situation.

4.4 Edge-based segmentation

Threshold based segmentation uses global information to segment the image, namely the threshold value. Edge-based segmentation methods works solely on local image information as they look for explicit or implicit boundaries between regions by using edge detecting operators.

After applying an edge detecting operator, such as the Laplacian as defined in equation 3, further processing steps must combine the partial edges found by the edge detection operator into contours that follow the regions of the image.

Algorithms which can handle "holes" in the edges are *Edge relaxation*, *Border tracing*, *Edge following by graph searching* and *Hough transformation* to mention some. These techniques (and edge-based segmentations) are covered in-depth by Pratt in [15] and Sonka et al in [21].

Because edge-based segmentation only uses local image information, this technique is highly sensitive to noise and can produce spurious, missing or discontinuous edges. Edge detection operators will actually amplify the noise already present in the image. Smoothing the image before applying the edge detecting operator helps, but this also hides or blurs fine structures such as small blood vessels.

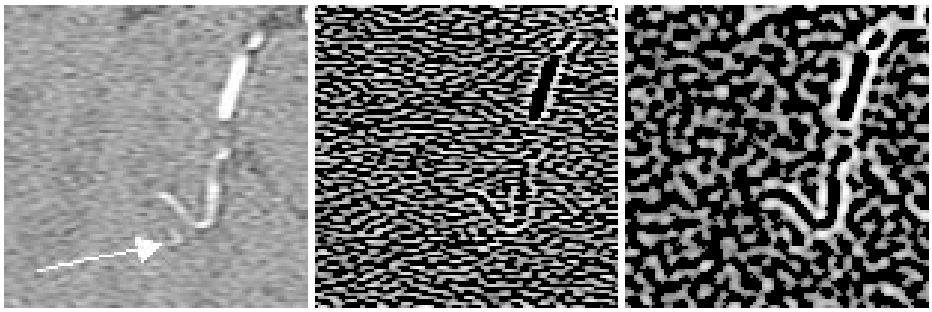


Figure 29: Example of edge detection.

Figure 29 shows the application of the Laplacian operator on a small region of the brain. The leftmost image is the original, the center image is applied with the Laplacian operator, and the rightmost image is smoothed before applying the Laplacian operator. This example shows how the fine structure indicated by the inserted arrow is lost in the edge detecting.

4.4.1 Border tracing

The idea behind border tracing is that it is possible to segment a region by following the border around the region. Given a starting point P_0 in the region border, the border tracing algorithm traverse the border of the region by considering the intensity of the pixels in a 3x3 neighbourhood (figure 30).

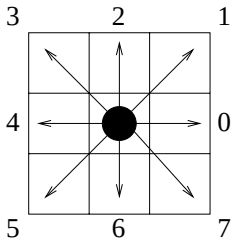


Figure 30: The 3x3 neighbourhood in border tracing.

The neighbourhood of the current pixel is traversed in a counter-clockwise direction. The current direction of the tracing will determine which one of the 8 possible search directions is used as the initial search direction.

Figure 31 shows the two applications of border tracing. The figure to the left shows an active border tracing.

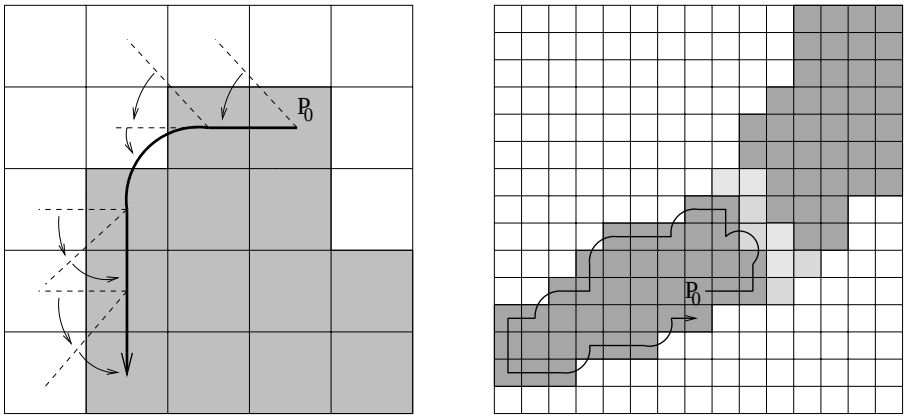
The right figure shows how the border tracing fails to segment a complete region if holes appear in region. Extensions to border tracing, such as *extended border tracing*, *heuristic graph search* and *edge following by dynamic programming* tries to overcome this sensitivity to noise and holes, and are covered in [21].

Border tracing may prove to be useful when segmenting properly defined regions such as the brain or the gray matter. For small vessels with weakly defined borders, border tracing is not very useful.

4.4.2 Hough transformation

The Hough transform is a method for isolating objects of a given shape in an image. Since the shape has to be given in some parametric form, the most common use for the Hough transform is regular curves. The generalized Hough transform can find objects of arbitrary shape, whereas the fuzzy Hough transform allows detection of objects whose exact shapes are unknown.

The basic idea behind the classic Hough transform is that each input pixel contributes to a global solution. The number of solution classes need not to be known beforehand. Consider for example the problem of detecting a straight line in an otherwise noisy image. The formula for a straight line in parametric notion is:



Segmentation by border tracing A hole in the region border

Figure 31: Border tracing.

$$r = xcos(v) + ysin(v) \tag{16}$$

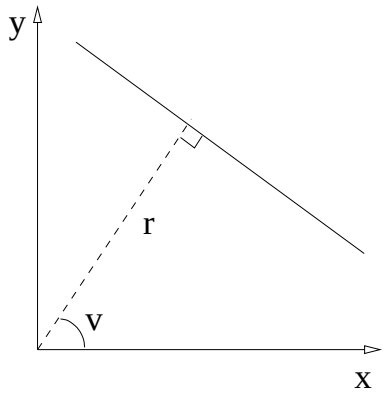


Figure 32: Parametric description of a line.

The unknown factors in equation 16 is r and v, while x and y are known from the image. The same line can be represented in (r, v) space. Plotting the possible (r, v) values defined by each (x, y) points in the image to curves in the polar parameter space, gives the Hough transformation. The more lines that have the same (r, v) parameters, the more points will be plotted in the parameter space.

To implement the Hough transform, the parameter space is quantized into a finite value grid. The cells are called the accumulator. Each time a value is inserted

in a cell, it is added to the value already there. Detecting lines is now a question of calculating local maxima in the accumulator grid.

Segmentation of the brain is somewhat more complicated than detecting lines. However, the Hough transform can be generalized to any shape. Unfortunately, this turns out to be both computationally and mathematically exacting.

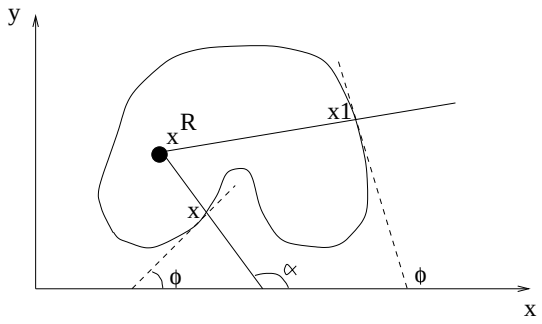


Figure 33: Setting reference points for a brain shape.

In the general Hough transformation, a shape is partially parameterized using a reference point X^R. A line is constructed from this point towards the border and the edge direction ϕ is found at this point. A reference table (or R-table) is constructed, consisting of intersection parameters and border directions. The accumulator holds the potential reference points $A(x_1, x_2)$. For each input pixel, the edge direction is found along with all potential reference points and the accumulator is increased accordingly.

The generalized Hough transform is very robust. However, we have chosen to use other segmentation methods. Using the general Hough transform to segment the brain would be extremely complicated. Different R-tables has to be constructed depending on which slice is considered and, moreover, the calculation is computationally very complex.

4.5 Volume matching

Matching is yet another approach to segmentation. The idea is to fit a well defined general model of the object to the patient data. The matching can be done in both two and three dimensions. The two dimensional method matches edges, contours or two dimensional regions of a general model with the corresponding patient data. The three dimensional method matches surfaces, curves and three dimensional regions with the patient data.

The matching approach must be able to handle rotational differences and scaling between the general model and the patient data, which makes the matching

approach very computationally expensive.

A drawback of the volume matching technique is the requirement that the object to be segmented should not differ much from the general model. Although volume matching works when segmenting the brain, this technique falls short when trying to segment the blood vessels. The structure of the brain is quite similar in everybody as long as healthy brains are considered, but this does not always hold for the blood vessels. As mentioned in the introduction, the visualization of blood vessels is especially interesting in case of arterio-venous malformations (AVM). These malformations can occur if a tumor in the brain has grown so big that it has pushed a number of blood vessels aside (to new locations). For this reason we can not use volume matching segmentation for our project, as the brain scans we consider may contain tumors or AVM.

A survey of volume matching of the brain is given by Rodell et al in [18].

4.6 Active contour approach to segmentation

The *snake* (or *active contour model*) was introduced in 1987 by Terzopoulos et al in [13]. An overview is given by Sonka et al in [21]. The snake is defined as an energy minimizing spline controlled by internal energy functions and influenced by external forces that pull it towards image features. Parametrically defined, the snake is

$$\mathbf{v}(s) = (x(s), y(s)) \tag{17}$$

where $x(s)$, $y(s)$ are x, y co-ordinates along the contour and $s \in [0, 1]$. The snake will, through a series of iterations, search for an energy minimum and terminate when a minimum is found. The energy function to be minimized is a weighted combination of several forces:

$$E_{snake} = \int_0^1 (E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s)))ds \tag{18}$$

where E_{snake} is the total snake energy. The internal energy is specified by:

$$E_{int}(\mathbf{v}(s)) = \alpha(s) \mid \frac{d\mathbf{v}}{ds} \mid^2 + \beta(s) \mid \frac{d^2\mathbf{v}}{ds^2} \mid^2 \tag{19}$$

The term $\alpha(s) \mid \frac{d\mathbf{v}}{ds} \mid^2$ specifies the elasticity of the snake and is also denoted E_{cont} (for continuation). The term $\beta(s) \mid \frac{d^2\mathbf{v}}{ds^2} \mid^2$ specifies the curvature or stiffness

of the snake and is denoted E_{curv}. The internal energy favours active contours which are continuous and smooth.

E_{image} is the image energy and depends solely on the image intensity values along the path of the snake. This energy is a weighted combination of functions which attracts the snake to lines, edges and terminations. Typically the term is approximated to the gradient of the image:

$$E_{image}(\mathbf{v}(s)) \approx - \mid \nabla f(x,y) \mid^2 \tag{20}$$

such that the energy will be low when the change in image intensity is large.

E_{con} comes from external constraints from the user, automated methods or some other mechanism. Usually E_{con} is based on some a priori knowledge of the region to be segmented.

Snakes have been successful in performing tasks such as edge detection, corner detection and motion tracking. Solling et al [20] use snakes to segment kidneys and temporal lobe of the brain in MR images with excellent results.

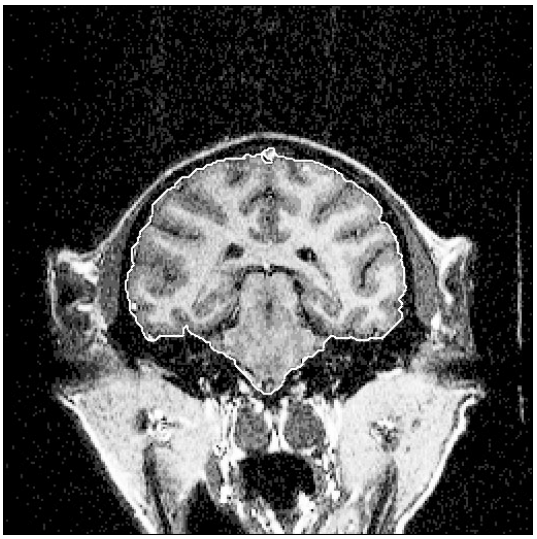


Figure 34: The brain of a monkey segmented by a snake.

"Balloons" which inflate the snake, are introduced as an additional pressure force on the contour of the snake. The balloon force helps the snake in overcoming local minima. This inflationary force also reduces the dependence on the initial contour, reducing user interaction.

Snakes have several advantages over standard feature extraction techniques: they can be controlled interactively and they are more resistant to image noise.

The main problems with snakes are dependency on the initial contour and lack of convergence to the global energy minimum. Another potential problem with snakes occur when the snake tries to minimize the energy over the entire path of the contour and overlooks small features in the image.

We will not use snakes for segmenting the vessels of the brain since the size of the smallest vessels pose a problem to snakes. Also, the manual pinpointing of starting points contradicts our goal, which is to make the process fully automatic. Still, snakes is a very interesting technique and it may prove to be a valuable extension to other segmentation techniques in identifying blood vessels.

4.7 Using neural networks for segmentation

A completely different approach to segmenting images is the use of neural networks. Neural networks are very useful for recognizing patterns and is also used in image processing. An introduction to neural networks is given by John Hertz et al in [10].

Ideally a neural network should be able to imitate biological vision and thereby be able to recognize patterns in images in the way the human eye does it, even when noise and tissue inhomogeneity are present in the image. On the other hand they suffer from sensitivity to geometric scale, orientation and intensity distribution.

Toulson et al uses in [23] neural networks for segmenting MR images and Wang et al in [22] uses neural networks for segmenting noisy images.

4.8 Chosen segmentation techniques

We have chosen the hysteresis approach to image segmentation in which neighbouring pixels are examined and added to the set of segmented pixels if the intensity exceeds a certain threshold. The process works recursively on all pixels in the set. Extending the hysteresis algorithm to volumetric data sets enables us to segment the blood vessels by connectivity. Connectivity is defined as two pixels are connected if there is a way from one to the other within a given intensity range.

Hysteresis offers several advantages over conventional segmentation techni - ques. Unlike edge detection methods, the borders of the set of segmented voxels found by hysteresis are sharply defined (since we only add voxels to the exterior of our set) and connected. The algorithm is also very stable with respect to noise. The set of segmented voxels will never contain too much of the background, as long as the thresholds are defined correctly. The main problem with hysteresis is the fact that if the growing threshold is too low this leads to explosive growth.

Besides hysteresis we have also chosen mathematical morphology for segmentation. This technique will be covered in-depth in the next chapter.

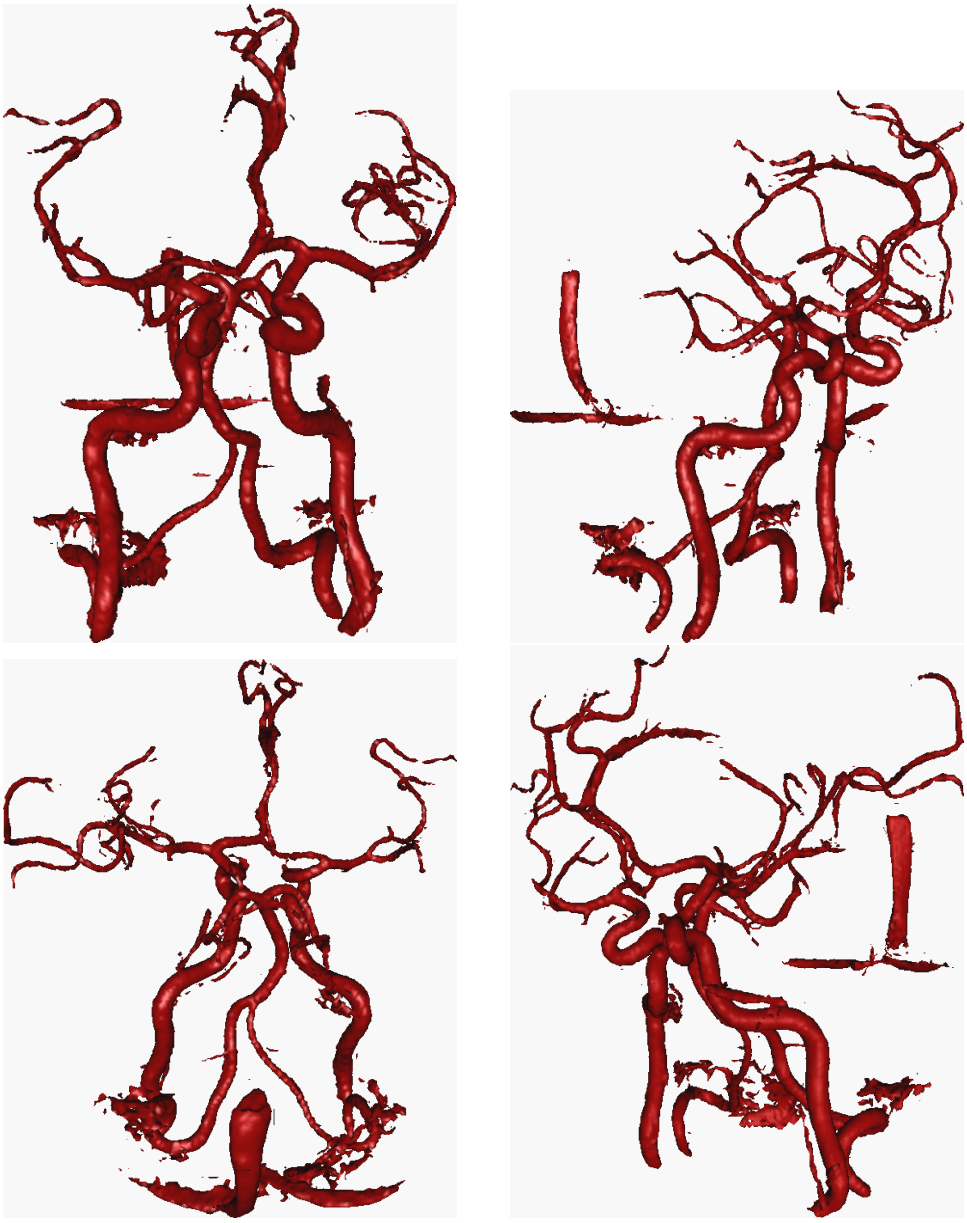


Figure 35: Four different views of segmented vessels.

5 Mathematical morphology

Mathematical morphology is the theory for analyzing spatial structures. The name morphology implies that is is useful for analyzing the shape and form of objects, and it is called mathematical because it is based on set theory and topology.

Conventionally, an image is described as an array of coordinates and intensity values. Alternatively, an image can be represented as a *set of coordinates*, where a set is either a collection of image pixels or background pixels. In the following, we will assume the image to be binary, although most of the morphology theory can be easily generalized to n-bits pictures. Binary images are created from grayscale images by thresholding.

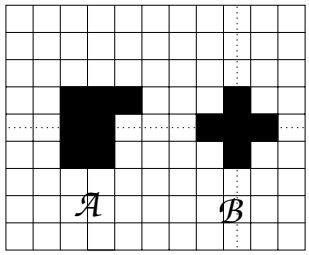


Figure 36: Two images consisting of two sets A and B.

Definition:
Let A and B be two sets.

$$\mathcal{A} \oplus \mathcal{B} = \bigcup_{b \in \mathcal{B}} (\mathcal{A} + b) \tag{21}$$

$$\mathcal{A} \ominus \mathcal{B} = \bigcap_{b \in \mathcal{B}} (\mathcal{A} + b) \tag{22}$$

These operators are also known as *Minowski* operators, and it is from these that we define the fundamental mathematical morphology operations.

5.1 Dilation and Erosion

A *structuring element* (or SE) consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. Normal cartesian coordinates are used, so a convenient way of representing the element is as a small image

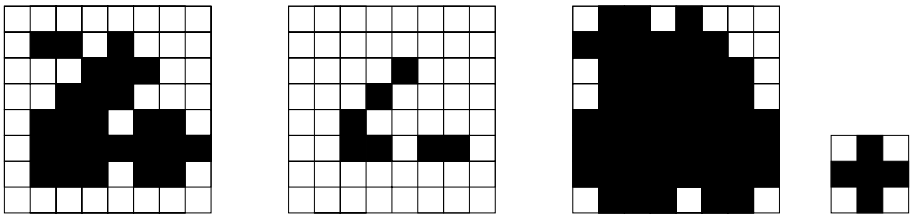


Figure 37: a) Original image. b) Eroded image. c) Dilated image. d) Structuring element.

on a rectangular grid, much like the kernels described in the section concerning preprocessing.

The dilation of a binary image by a structuring element is calculated by considering each background pixel of the input image. Place the structuring element on top of each background pixel so that the origin of the structuring element coincides with the input pixel position. If at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value. If all the corresponding pixels in the image are background however, the input pixel is left at the background value. The result is, that the image is *dilated* or widened. Examples of dilation and erosion are given in figure 37.

Using the operators defined in equation 21 and 22, dilation can be described as

$$D(\mathcal{A}, \mathcal{B}) = \mathcal{A} \oplus \mathcal{B} \tag{23}$$

Erosion is defined in a similar way. To compute the erosion of a binary image by a structuring element, again consider each of the foreground pixels in the input image. For each foreground pixel (input pixel) place the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel coordinates. If, for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background however, the input pixel is set to background value. The result is, that the image is *eroded* or shrunk.

$$E(\mathcal{A}, \mathcal{B}) = \mathcal{A} \ominus \mathcal{B}^c \tag{24}$$

The transpose of $\mathcal{B}$ is given as $\mathcal{B}^c = \{-b : b \in B\}$

Notice that the dilation of an object is equivalent to the erosion of the background and vice versa.

The structuring element can have any shape. If the goal is to, say, dilate an image in one direction, the SE should be made as a line in that direction.

5.2 Opening and Closing

Opening and *closing* are two important operators from the field of mathematical morphology. They are derivatives of the fundamental operators *erosion* and *dilation*. As with other morphological operators, the exact operation is determined by a structuring element. The effect of an operator could be to preserve foreground regions that have a similar shape to the structuring element, or that can completely contain the structuring element, while eliminating all other regions of foreground pixels.

An opening is defined as an erosion followed by a dilation using the same structuring element for both operations.

$$O(\mathcal{A}, \mathcal{B}) = D(E(\mathcal{A}, \mathcal{B}), \mathcal{B}) \tag{25}$$

Similar, closing is defined as the erosion of a dilation.

$$C(\mathcal{A}, \mathcal{B}) = E(D(\mathcal{A}, \mathcal{B}^c), \mathcal{B}^c) \tag{26}$$

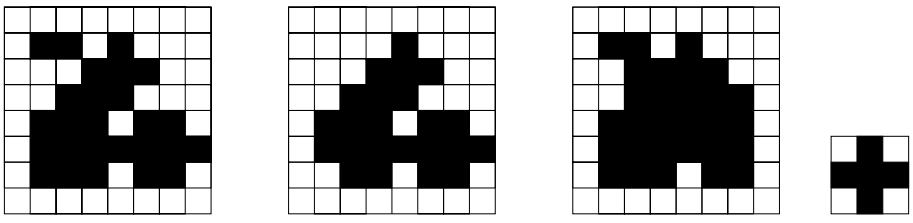


Figure 38: a) Original image. b) Opened image. c) Closed image. d) Structuring element.

To visualize how an opening is performed, imagine taking a structuring element sliding it around inside each foreground region. All pixels that are covered by the SE while it is entirely inside the foreground region will be preserved. All

the foreground pixels that cannot be reached by the SE without it crossing the border of the region will be eroded away. This also means that after the opening has been carried out, the resulting borders of the foreground regions will be such that the SE fits inside them. Thus, if another opening is applied, it will have no effect on these borders. The shape is stable. An operator with this property is known as an *idempotent* operator. Both opening and closing are idempotent operators.

To summarize the intuition: The *threshold* is the fundamental tool to identify regions of homogeneous intensity. *Dilation* is used to force spatial connection and to counteract erosion after object selection. *Erosion* forces spatial isolation by breaking false connections. *Closing* fills out holes and indentations without changing the gross shape of the object. *Opening* can remove thin junctions and is therefore an excellent tool for the removal of false connections between objects.

5.3 Thinning and Thickening

So far, the structuring elements has been binary. An entry in the element is either set, or we ignore the element. Expanding the definition of a structuring element to include both foreground and background (as well as "ignore's") we can introduce new functions. A structuring element $\mathcal{B}$ is now composed of two disjoint subsets. $\mathcal{B}= (B_1, B_2)$. A new operator called *the hit or miss operator* (HoM) is introduced.

$$HoM(\mathcal{A}, \mathcal{B}) = \{x : B_1 \subset \mathcal{A} \wedge B_2 \subset \mathcal{A}^c\} \tag{27}$$

The operator works the same way as the previous. It is translated to every pixel in the input image. If the foreground and background pixels in the structuring element match foreground and background pixels in the image, then the pixel underneath the origin of the structuring element is set to the foreground colour. If it does not match, then that pixel is set to the background colour.

This operator can be used to define thinning and thickening.

$$Thin(\mathcal{A}, \mathcal{B}) = \mathcal{A} \cap -HoM(\mathcal{A}, \mathcal{B}) \tag{28}$$

$$Thick(\mathcal{A}, \mathcal{B}) = \mathcal{A} \cup HoM(\mathcal{A}, \mathcal{B}) \tag{29}$$

An example of thinning is given in figure 39. To thin an image equally, the SE should be applied four times, each time after it has been rotated 90 degrees. Some times, it is desirable to thin (or thicken) only certain shapes of an image, and this is where these operators come in. To do a dumb thinning, an erosion will do just as well, removing the outer pixels on the entire image.

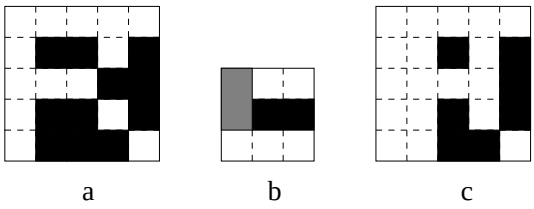


Figure 39: Example of thinning. a) Original image. b) Structuring element (black=foreground, grey=background). c) Thinned image.

5.4 3D Mathematical morphology

Expanding the theory to three dimensions is trivial due to the set representation and the generic Minowski operators. When processing volumetric image data, the extra dimension results in a more precise segmentation because the connectivity is taken into account.

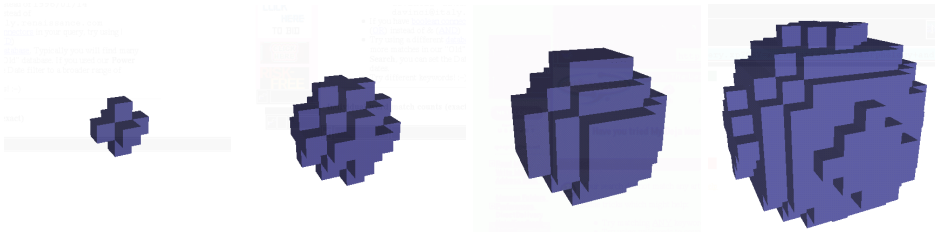


Figure 40: Four different circular 3D structuring elements of diameter 3,5,7 and 11 pixels.

We use both cubic and spheric structuring elements. The spheric shape is best seen in the large elements, see figure 40. Since more data is taken into account, there is a significant difference between the result. Figure 41 shows two masks segmented with a 2D and a 3D structuring element, both with a diameter of 7 pixels. The 3D segmentation produces much smoother curves than the corresponding in 2D.

The computational load increases when moving from 2D to 3D morphological operations, and the result is worth it.

5.5 Applications of morphology

Once the basic functionality of morphology is understood, it opens a variety of potential usages. Quite often this method is used to eliminate background noise, or performing actual segmentation.

Figure 41: Left is original image. Middle is a mask made with a 2D SE, right is made by a 3D SE. The operator used in both cases is closing.

We present both traditional but also novel usage of mathematical morphology.

5.5.1 Masking

Shape extraction by means of mathematical morphology is a known issue. Removing everything but the shape of the head is desirable.

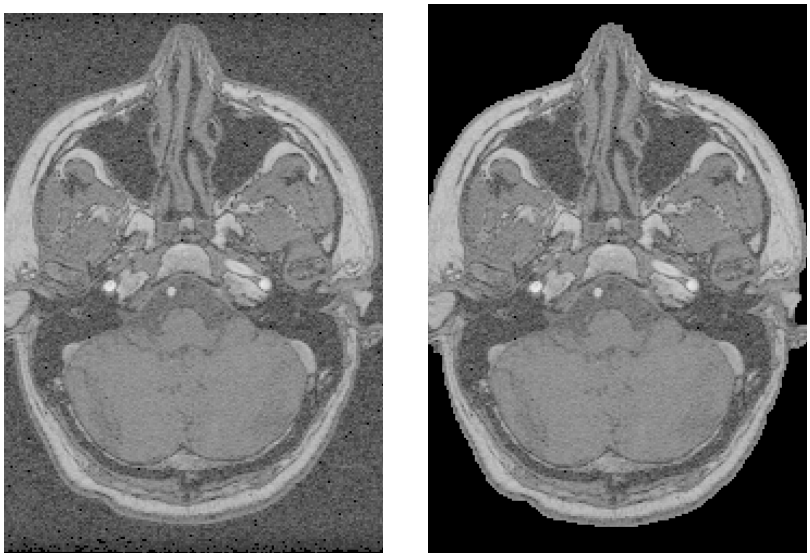


Figure 42: Sample slice (left) and extracted head (right).

The head shape in figure 42 is made from the left image using a closing operator and a circular SE with a 15 pixel diameter. Notice that although some of the tissue inside the head has same intensity as the background, it is not removed.

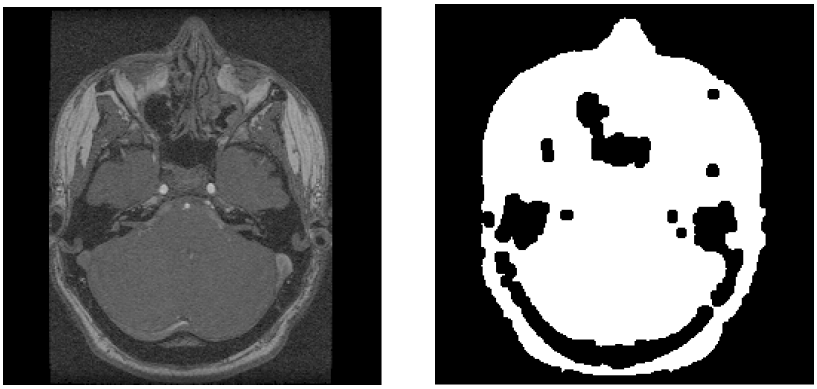


Figure 43: Left is before closing. Right is after. Notice the flaws (holes).

The reason why the shape is so perfectly extracted lies in the properties of the closing operator. The intuition is to check each background pixel. If the SE can cover that pixel without touching the foreground, the pixel remains background. Otherwise, it is set to foreground. The size of the SE determines how close it can get to the border between foreground and background. This border is determined by a threshold.

Since background and foreground is determined by an intensity value, some regions inside the brain (which is considered foreground) is categorized as background. This yields some holes in the mask, because sometimes the SE can fit inside these areas. We have implemented a function called “close mask” which, as the name implies, closes the current mask (see figure 43 - 44.) Slice-wise filling in 2D is especially useful in the case of holes that have an unwanted connection to the background

Figure 44: The mask is closed using the “close mask” function.

An area is labelled as a hole if

1. It has background color
2. It is surrounded by foreground color

The algorithm searches the image for potential holes. Briefly, it scans through each line in the image, remembering if it has seen a background pixel. The second time it meets a background pixel, it is either on the other side of the head, or inside a hole. To check if the latter is the case, the algorithm performs a variation of the flood fill algorithm. Instead of visiting each neighbour, it visits every fourth, checking if the background pixel is surrounded by foreground pixels. If this is the case, the hole is filled using a flood fill algorithm.

Since most of the potential holes are the background pixels immediately to the right of the image, the first thing we do is to check if the right border can be reached from the pixel in question. If so, the rest of the checks are omitted, since we know it can not possibly be a hole.

6 Methodology

We have presented all the tools we will use for preprocessing and segmentation of the data. This section will explain how we put it all together.

We will do a step-by-step presentation, but the actual program (A.S.A.P.) will be able to perform all the actions automatically.

First, the data is preprocessed to correct for non-uniform illumination from slice to slice. Next, the vessels are isolated and segmented. Lastly, the head is segmented for improved visualization.

6.1 Adjusting slice intensity

The illumination of the scanning may be non-uniform from one slice to the next. Furthermore the last slices of a slab overlap with the first slices of the next slab and unfortunately this affects the intensity of these slices. This creates a large intensity gap when leaving a slab. In figure 45 the slab borders are seen as lines in the maximum intensity projection (clearly the head is composed of 3 slabs). The 3D hysteresis segmentation (connectivity) will be severely effected by this intensity gap and it must be accounted for before commencing segmentation.

Figure 45: A maximum intensity projection showing the slabs.

Examining the intensity average for each slice, shown on figure 46, the intensity average rises significantly from slice 50 to slice 60 and again from slice 100 to slice 110.

The problem is solved by calculating a new intensity average for each slice. This new intensity depends on the intensity averages of a number of slices prior to and after the current slice. The equation for calculating the new intensity average is shown in equation 30.

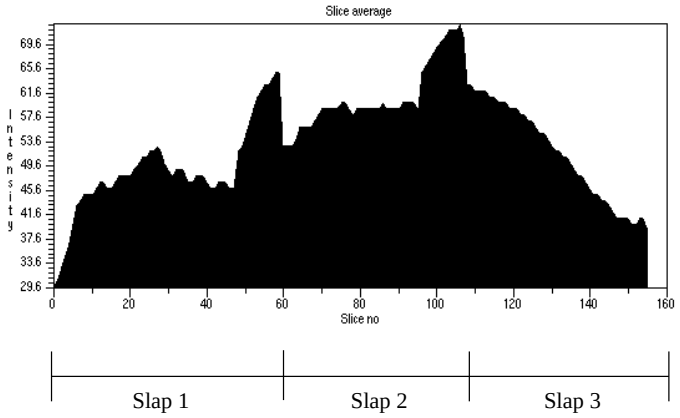


Figure 46: Intensity average of all slices. Note the peaks.

$$A'_j = \frac{1}{2k+1} \sum_{i=j-k}^{i=j+k} A_i \tag{30}$$

A_i is the intensity average of slice i, and A'_j is the new calculated intensity average for slice j. k is the number of slices prior to and after the current slice which contributes to the new intensity average, and k depends on which slice is being adjusted. Slices near the start and end of the scanning should only examine few other slices when calculating the new intensity average. Slices in the middle of the scanning should examine a larger range of the neighbouring slices. k is therefore a function of i and one representation of such a function is showed in figure 47. N is the total number of slices in the data, and $k \in [0..\frac{N}{10}]$.

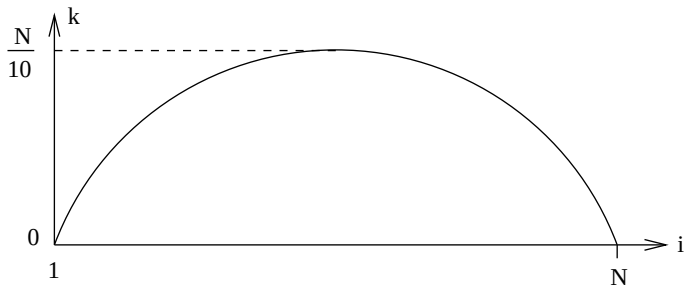


Figure 47: k as a function of i.

When the new intensity average for all slices is known, the intensity of each

slice is adjusted to match the new average. We adjust the intensity of the slices by using standard gamma curve technique.

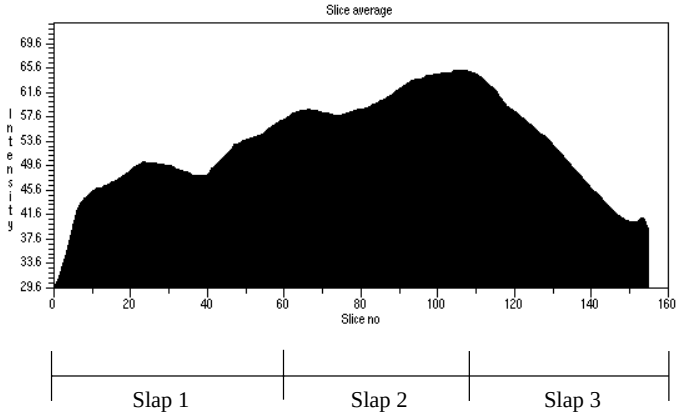


Figure 48: Intensity average of all slices after adjustment.

Figure 48 shows the slice intensity average after adjustment. The steep discontinuity of the intensity is gone. A maximum intensity projection of the image (figure 49) shows that the illumination through the slices is almost uniform.

Figure 49: After averaging. The slab borders are nearly gone.

6.2 Segmenting blood vessels

The non-uniform illumination through the slices has been accounted for and we proceed to segment the blood vessels by 3D hysteresis and connectivity.

Unfortunately other kinds of tissue has high intensity. These include fat and the muscles behind the eyes and on the sides of the head (see figure 50). The intensity of these tissues overlap the intensity of the blood vessels. Since the blood vessels may be connected to, or passing near by, such tissues, they will be segmented as well if the growing threshold is set too low.

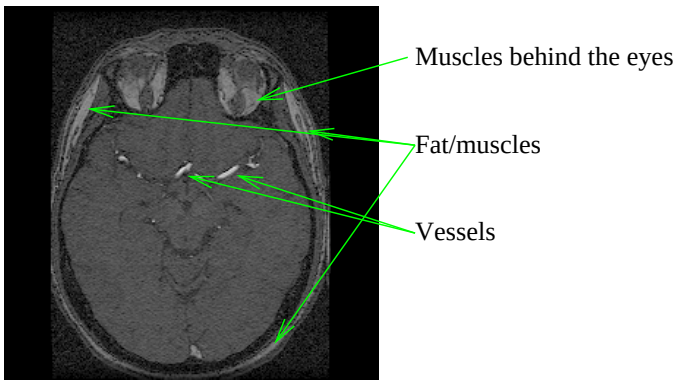


Figure 50: Fat, muscles and vessels have overlapping intensities.

To handle this, we first isolate and remove the large blood vessels that has intensity values larger than fat and muscles. The segmented large blood vessels are placed in a separate buffer. Next, high intensity tissue like fat and muscles is removed, and finally the large vessels are inserted back into the data.

6.2.1 Isolating large blood vessels

The large blood vessels are segmented using 3D hysteresis, which works with two thresholds, one threshold to find the seed voxels, and the second threshold is used as a growing measure.

The seed voxels are automatically calculated by histogram analysis. We will use some of the brightest voxels in the data as seed voxels. As the data is weighted for blood flow, the brightest voxels are found in the large blood vessels. First, a median filter is applied on a copy of the data to remove shot noise, which otherwise could have made us pick a bright voxel in the middle of i.e. fat tissue (certainly not in one of the large blood vessels).

After median filtering the data, the histogram is inspected and the 200 brightest voxels are selected as seed voxels. The growing threshold is also determined by histogram analysis and is set to the intensity value which corresponds to the brightest 2 % of the voxels.

This is a very conservative threshold, and could be lowered in order to include more voxels to segmentation, but this will increase the risk of including unwanted

high intensity tissue in the segmentation.

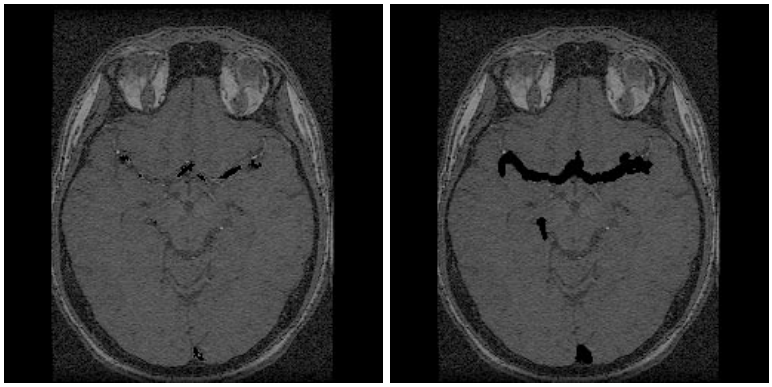


Figure 51: More voxels are segmented after a dilation. Left: Segmented voxel before dilation. Right: Segmented voxels after dilation.

Figure 51 (left) show a slice after an application of the 3D hysteresis. The larger vessels have been removed by the segmentation. Since we are using a conservative growing threshold, some of the voxels in the vicinity of the vessels have a high intensity, indicating that they are part of the vessels. These voxels should be included in the segmentation, but we do not allow a lower growing threshold because of the risk of oversegmentation.

The solution is to use dilation from the theory of mathematical morphology on the final set of segmented voxels. Dilating the set of segmented voxels with a 3D spherical structuring element of $5 \times 5 \times 5$ voxels will "inflate" the set to include the voxels in the near vicinity. The segmentation on a single slice after applying dilation is shown on figure 51 (right).

Figure 52 shows how the entire set of segmented voxels is affected by a dilation. The set is shown from three different angles for improved visual understanding.

By now we have finished segmenting the large blood vessels. We continue by removing unwanted tissue like fat and muscles as they display the same high intensity as the smaller blood vessels.

6.2.2 Removing high intensity tissue like fat

As shown on figure 50, the muscles behind the eye and the fat/muscles on the side of the head share intensity values with the small blood vessels. Before we are able to segment the vessels by thresholding we must remove these types of tissues.

We will remove the unwanted tissue by using 3D hysteresis once again along with mathematical morphology. This time we cannot use histogram analysis to

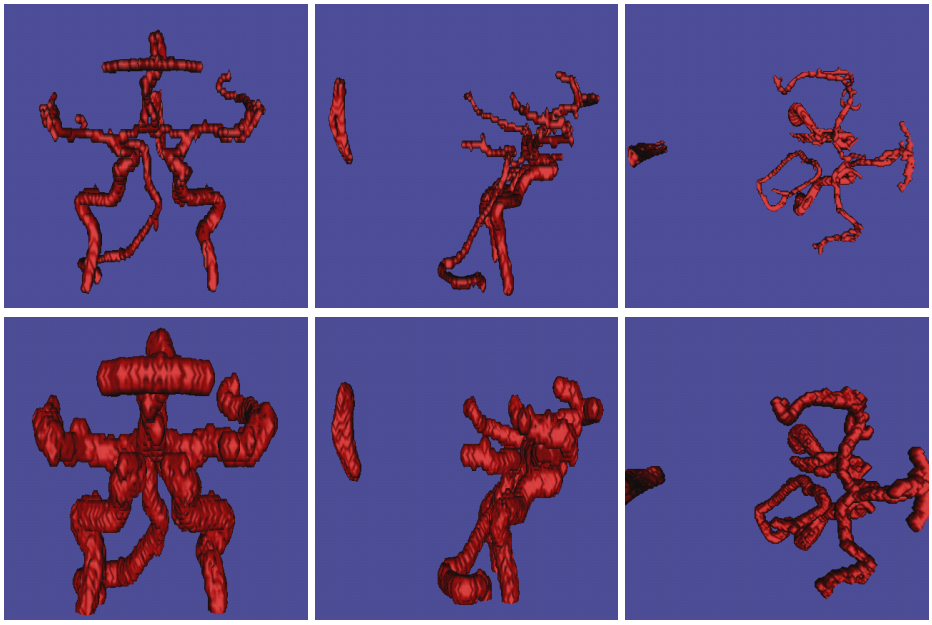


Figure 52: The segmentation of the large vessels from different angles. Top three images are before dilation, bottom three images are after dilation.

determine the set of seed voxels because of the overlapping intensity values.

Instead we will use the idea of letting a circular structuring element search every slice for large areas of high intensity. The structuring element will be created with such a large diameter that it will only be able to fit within the muscles behind the eyes, and in the muscles/fat on the side of the head, see figure 53 (left). For a voxel to be included in the set of seed voxels, we put the constraint on the voxel, that all voxels in the vicinity that are covered by the structuring element must have an intensity value above a certain threshold.

As we have already removed the larger blood vessels the remaining vessels are far smaller than the structuring element. This ensures that the set of seed voxels only will consist of voxels in the large areas of muscle tissue behind the eyes and possible of voxels in the large areas of fat or muscles on the side of the head. The size of the structuring element depends on the size of the dataset and is thus automatically calculated.

By now the set of seed voxels has been found by the structuring element, and before we use 3D hysteresis to remove the unwanted tissue, the growing threshold for the hysteresis is determined by histogram analysis.

Figure 53 (middle) shows the result after removing fat and muscles from the data. The larger blood vessels, which were segmented in the previous section, are

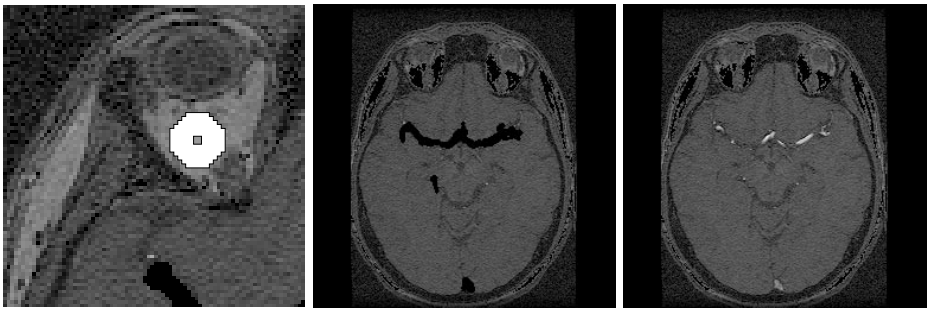


Figure 53: Left: The structuring element has found a new voxel for the set of seed voxels. Middle: The fat and muscles have been removed. Right: The vessels are inserted in the image.

reinserted into the data, and the result is shown in 53 (right).

6.2.3 Segmenting blood vessels by thresholding

The fat and the muscles behind the eyes and on the side of the head has been removed, and we want to conclude the segmentation of the vessels.

One approach would be to use the hysteresis technique once again, and the result would indeed be acceptable. But as a result of the partial volume effect, the segmentation would not be able to include the small vessels with sizes approaching the voxelsize.

Instead we use simple thresholding for the vessels, and this leads to surprisingly good results. Since fat and muscles has been removed only vessels has high intensity and by setting the threshold for the segmentation to the proper low value a remarkable number of vessels are segmented.

This threshold is automatically determined by histogram analysis, and will be used as the isovalue when generating the isosurface for the vessels.

6.3 Segmenting the head

To enhance the visual understanding of the position of the vessels, the head contour is segmented and visualized at the same time as the vessels. The high resolution data used for segmenting the blood vessels is often of a higher resolution than needed for the headshape. Thus, a smaller resampled copy of the data is being made, typically to a resolution of $64 \times 64 \times 30$, although this resolution is fully user definable.

As when segmenting the vessels, the first thing to be done is to adjust non-uniform illumination through the slices on the resampled data.

Second, in order to remove shot noise median filtering is applied on the resampled copy. We use either 3x3 or 5x5 median filtering, depending of size of the resampled data.

For segmenting the head shape, we use closing from theory of mathematical morphology as described in chapter 5. A $5 \times 5 \times 5$ 3D spherical structuring element is used for the segmentation. The threshold used is calculated by the optimal threshold algorithm. Possible holes in the head shape are closed as described in chapter 5.

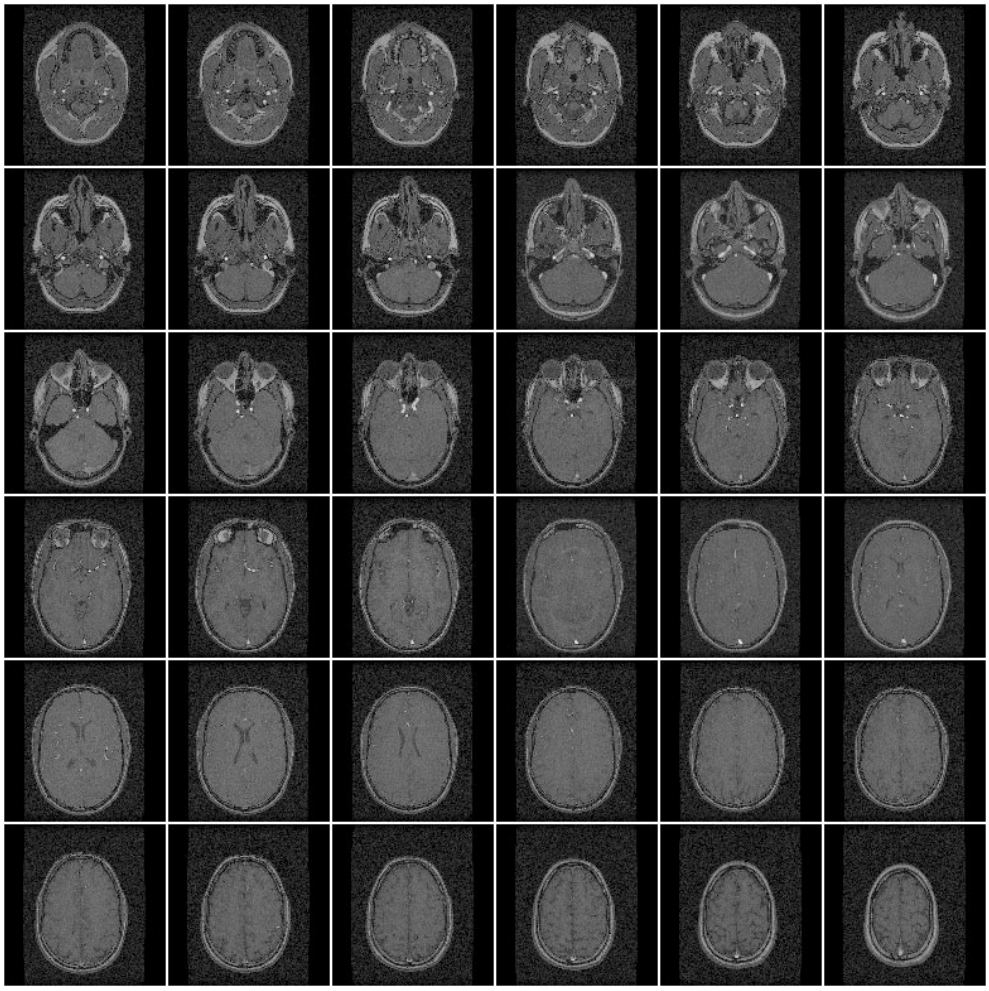


Figure 54: 36 representative slices of one of our datasets.

To remove isolated pixels, median filtering is applied once again.

Figure 54 show 36 representative slices from the resampled data and figure 55 shows the same slices after the headshape have been segmented.

Lastly, the sharp borders of the head shape are smoothed by applying a Gaussian smoothing kernel to the slices. This will result in a smoother surface and thus a better shading for the visualization.

Most of the examples throughout the thesis where the head can be seen are created from images with 64×64 spatial resolution. The most detailed head is made with a resolution of 256×256.

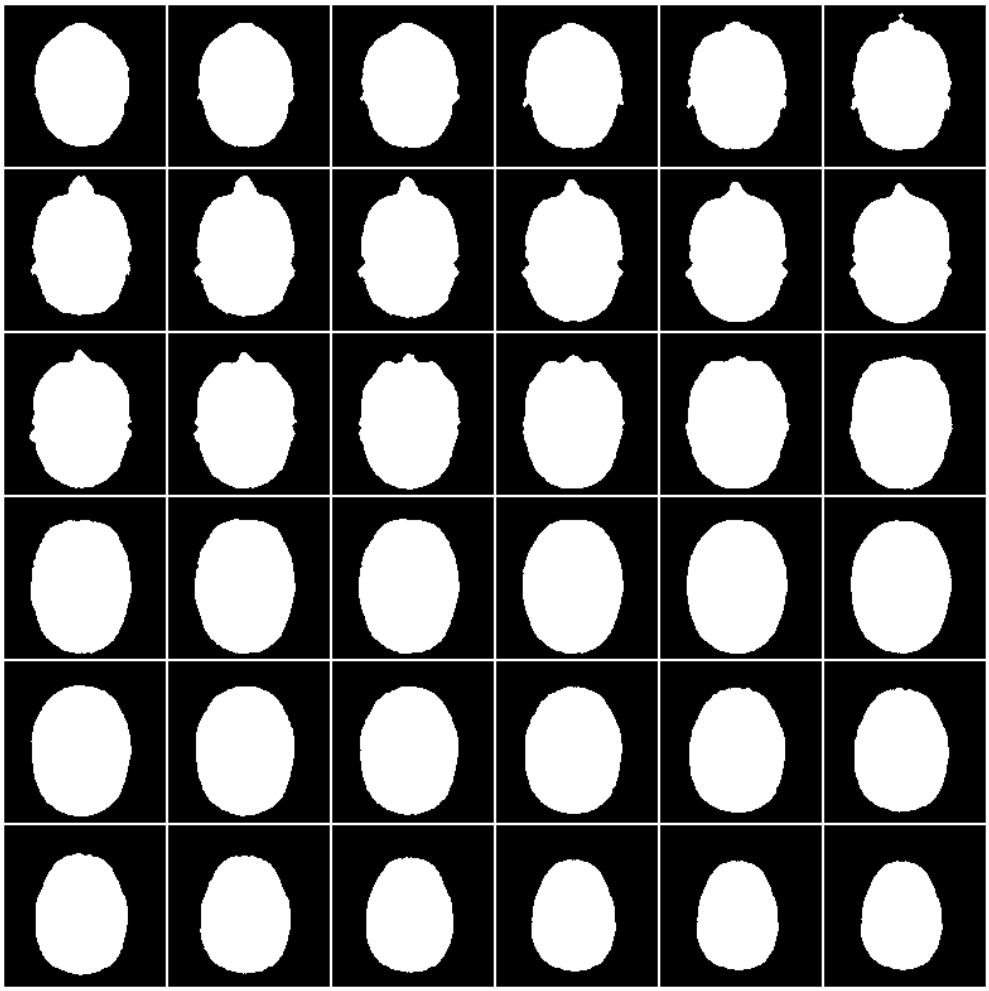


Figure 55: 36 representative slices of one of our datasets. The headshape has been automatically segmented.

7 Visualization

By now, the head structure and the vessels have been identified, but still only in 2D. For each slice, we have categorized the objects. The next step is to gather all the information into a 3D isosurface. We present two methods for isosurface generation from volumetric data, namely *Marching cubes* and *Marching tetrahedrons*. Other algorithms, such as raycasting [4], exist. Raycasting is also called Direct Volume Rendering because it renders the surface directly without generating an intermediate geometrical representation. Instead, the volume is treated as semi transparent, and rays of light is sent through the volume. These rays might be reflected, occluded or scattered creating specularity and shadows. This yields very detailed pictures, because everything in the volume can contribute to the result. Raycasting, however, proves to be too slow for interactive rendering, since every view has to be calculated. A survey of volume rendering techniques can be found in [25] and [24]. [1] and [8] compare different implicit surface polygonizers.

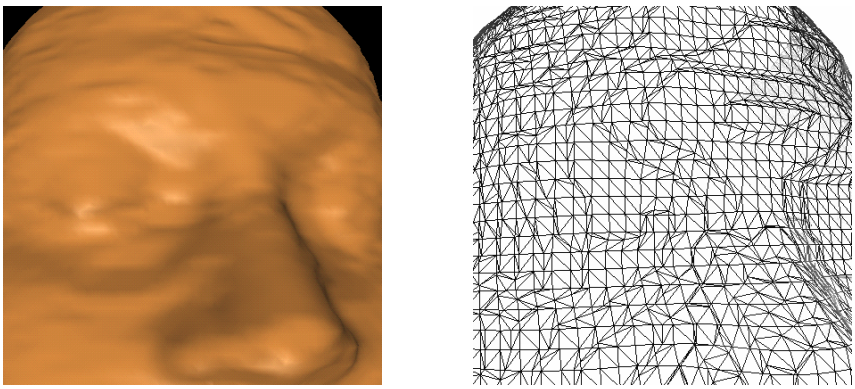


Figure 56: A rendering of facial structure and the corresponding triangles.

7.1 Marching cubes

The marching cubes algorithm was independently reported by Wyvill and McPheeters in 1986 [27] and Lorenson & Cline [7] in 1987. Since the latter is designed specifically for three-dimensional medical data, we will use this approach. An example of an isosurface generated by marching cubes is given in figure 56.

7.1.1 Algorithm overview

The basic idea is to define a voxelcube by the pixel values at the eight corners of the cube. If one or more corner pixels of a cube has values less than the specified

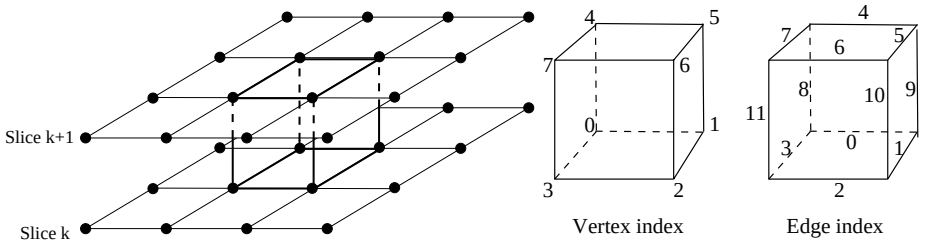


Figure 57: Left: The marching cube. The vertices are pixels. Right: The indexing convention for the algorithm.

isovalue, and one or more pixels have values greater than this value, the voxel must contribute one or more triangles to the isosurface. By determining which edges of the cubes are intersected by the isosurface, triangular patches which divide the cube into regions within the isosurface and regions outside can be created. By connecting the triangles from all cubes on the surface boundary, we get a surface representation.

If we define each corner as either being below or above the isovalue, this yields $2^8 = 256$ possible ways a surface can intersect a cube (eight vertices with each two states). By rotational and complementary symmetry, this number can be reduced to 15, which is showed in figure 58.

Case 1 is trivial: all points are either inside or outside the cube and this case does not contribute to the isosurface. For all other cases we need to determine where the isosurface cross and use the edge intersection points to create the triangles patches for the isosurface.

Each of the 14 non-trivial cases contributes with one to four triangles to the isosurface. The actual vertices for the triangles are computed by linear interpolations along the edges. Lorensen et al [7] state that higher degrees of interpolation show little or no improvement over linear interpolation.

Instead of finding the vertices for the triangles by interpolation, the midpoint of the edges can be used as proposed by Montani et al [3]. This will be computationally cheaper, but reduce the surface quality. We use interpolation, as this give us better shading calculations and thus a smoother surface.

As we know how to generate triangles for a single cube, we let the cube *march* through the entire voxelspace (hence the name *marching cubes*).

The marching cubes algorithm is very fast since it can be implemented to work almost entirely by using lookup tables. As mentioned, if a corner is above the threshold, it is assigned a value of one, otherwise zero. This forms an 8 bit vector. Using a predefined table of edge intersections, looking up that number returns a 12 bit vector of edge intersection information. If bit n is set in this vector, it means

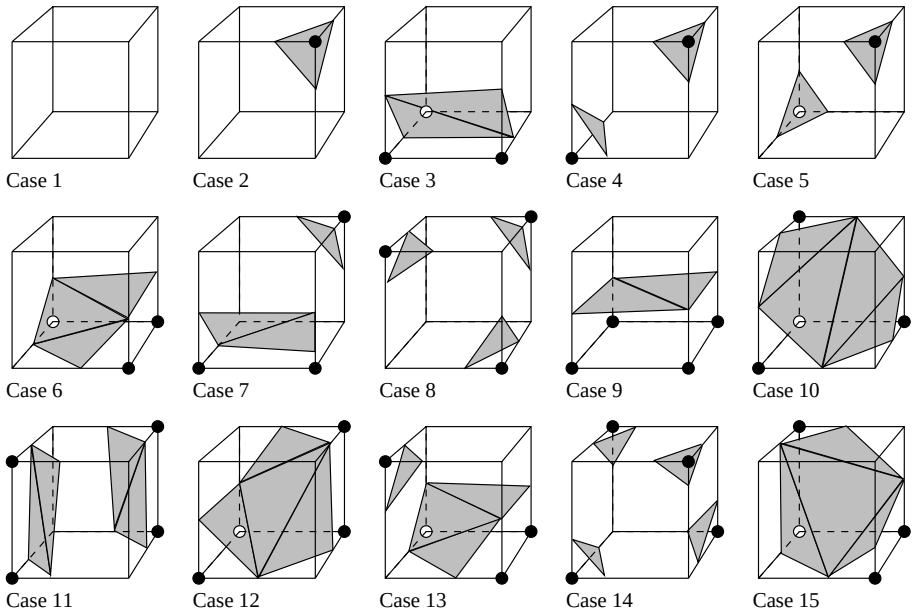


Figure 58: The 15 different cases of the algorithm.

that edge n is intersected by the isosurface (see figure 57 right).

For each cube vertex, the gradient is calculated. Let g be the gradient and f be the intensity, the gradient is defined as

$$g(x, y, z) = \nabla f(x, y, z) \tag{31}$$

The gradient vectors at the cubes vertices are estimated using the intensity differences along the three coordinate axes:

$$G_x(x, y, z) = \frac{f(x + 1, y, z) - f(x - 1, y, z)}{\Delta x} \tag{32}$$

$$G_y(x, y, z) = \frac{f(x, y + 1, z) - f(x, y - 1, z)}{\Delta y} \tag{33}$$

$$G_z(x, y, z) = \frac{f(x, y, z + 1) - f(x, y, z - 1)}{\Delta z} \tag{34}$$

Normalizing the gradient by dividing with the length results in a set of normal vectors used to produce a shading effect. To find the normal vector of the point of intersection, we use linear interpolation between the normal vectors of the cube vertices.

7.1.2 Shading the triangle mesh

The triangle mesh that forms the isosurface consists of triangles of a certain size. The size of these depends on the size of the sampling grid. For a smoother surface with the same mesh resolution, a shading is applied. We offer two types of shading. The computationally cheap flat shading and the more demanding Gouraud shading. Figure 59 shows the difference between these two types.

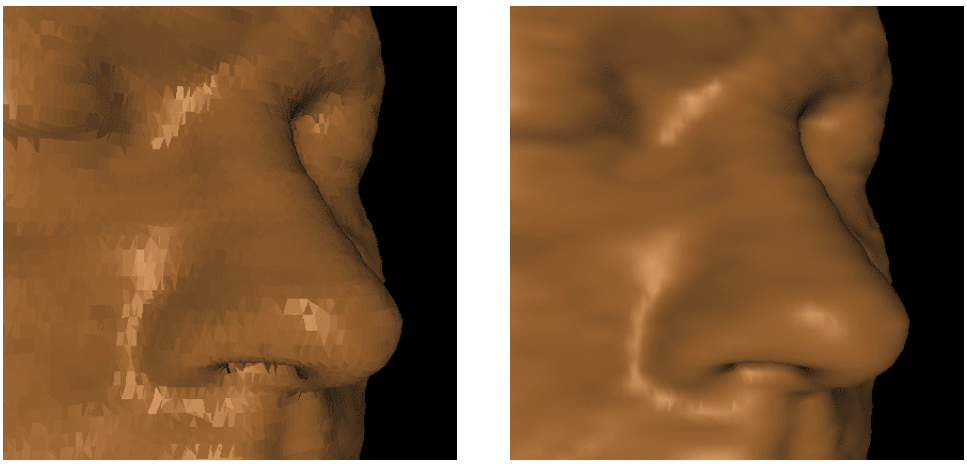


Figure 59: Left is flat shaded. Right is Gouraud shaded.

The reason why flat shading is fast to calculate is that the entire polygon is assigned the same color. Gouraud shading will assign each pixel a color that depends on the location of the pixel on the polygon.

Gouraud shading is a scanline algorithm. It runs through the polygon and calculates the color at the endpoints of the scanline by linear interpolation (see figure 60). The color ca is interpolated from $c1$ and $c2$, and likewise from cb. The interpolation is given in equation 35 and 36 where $0 \leq d1 \leq 1, 0 \leq d2 \leq 1$.

$$ca = d1 * c1 + (1 - d1) * c2 \tag{35}$$

$$cb = d2 * c3 + (1 - d2) * c2 \tag{36}$$

Now, ca and cb are interpolated to give the values along the scanline. The result is that curved surfaces approximated by planar polygons look curved.

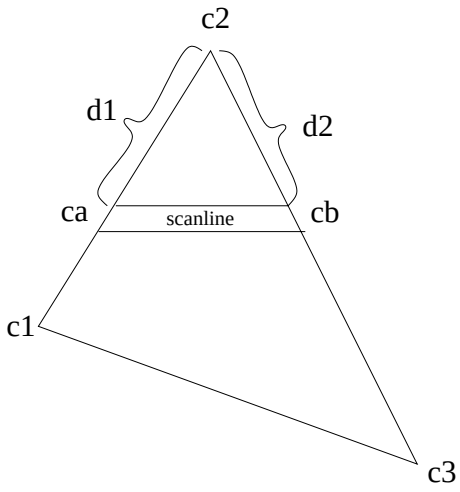


Figure 60: Scanline interpolation to achieve shading.

7.1.3 Disadvantages of the marching cubes algorithm

The marching cubes algorithm tends to produce a large number of triangles, and the time to render all these triangles and the memory cost to store them is considerable. If the surface becomes too large, some kind of reduction should be made, provided that the overall quality and fidelity is preserved. Our model is not suitable for decimation and we have not implemented any decimation methods, but we will sketch a few.

Besides the number of triangles generated, the marching cubes algorithm has another unfortunate property. Ambiguity of the surface configurations in cubes, where holes in the surface can be generated (reported in 1988 by Dürst [6]). We will later motivate why we do not handle ambiguity of the isosurface.

7.1.4 Reducing number of triangles

One way to reduce the number of triangles is to merge several triangles into a single polygon, if these triangles share some edge and are in the same plane. A simple method to test if two triangles are in the same plane is to calculate the cross product $\vec{c}$ of the normal vectors of the adjacent triangles.

$$\vec{c} = \vec{n_1} \times \vec{n_2} \tag{37}$$

If $\vec{c} = 0$ the two normal vectors are parallel, and if they share an edge, they must be in the same plane and thus can be replaced by a single polygon.

If we relax the demand $\vec{c} = 0$ to $\vec{c} \approx 0$, we allow adjacent triangles which are almost in the same plane to be replaced by a single polygon. This results in further reduction of triangles, but will also result in loss of image quality, since it will introduces cracks in the isosurface, where the polygons meet. These cracks will have to be removed by adding more triangles to the isosurface.

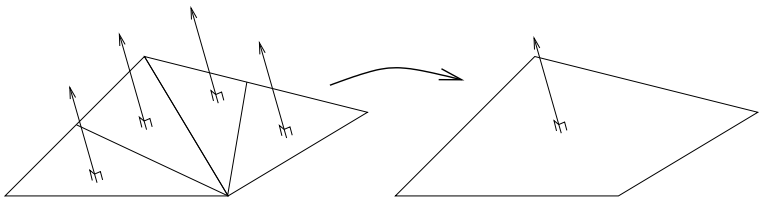


Figure 61: Adjacent triangles are merged into a single polygon.

Shekhar et al [16] presents an octree based decimation of marching cubes surfaces using an adaptive approach. It is a collection of enhancements that all in all yields up to 70% decimation of the number of triangles. This method only works on connected surfaces with large homogeneous areas.

In [11] Park et al give an algorithm for reducing the triangles by classifying the configurations of the marching cubes into types. Surface patches traversing neighbouring cubes of the same type can be merged in to patches, which can be approximated with fewer and larger triangles.

Renben Shu et al [17] presents another adaptive approach to the marching cubes algorithm, called *adaptive marching cubes*. It reduces the number of triangles by adapting the size of the triangles to the shape of the surface.

7.1.5 Ambiguity of the surface

Dürst [6] reports an ambiguity of the surface configurations of the marching cube which may cause false holes in the generated isosurface (figure 62). The ambiguity is introduced when the 256 different ways an isosurface can intersect the cubes are reduced to 15 by rotational and complementary symmetry. It is the complementary of some of the configurations that introduces the ambiguity (case 4 and 11 from figure 58).

Figure 63 clearly shows how false holes are introduced into the isosurface.

We have not made an effort to remove this ambiguity, but modified marching cubes algorithms which do, can be found in [9], [11] and [14].

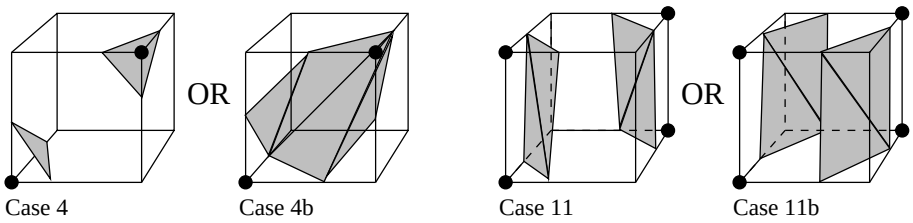


Figure 62: The ambiguity of the marching cubes.

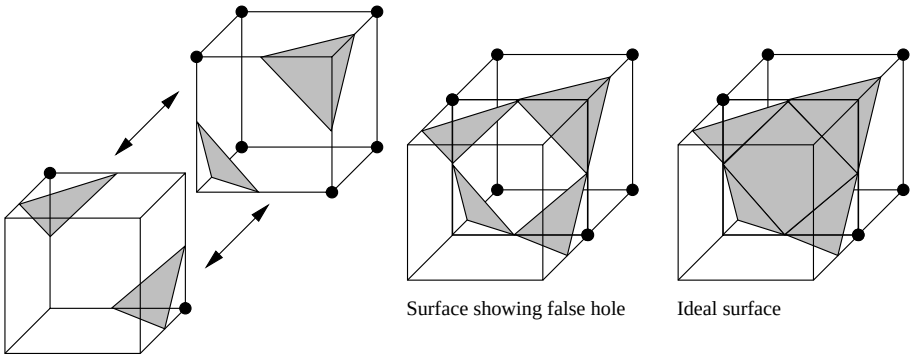


Figure 63: False hole.

7.2 Marching Tetrahedrons

The *marching tetrahedrons* works like the marching cubes algorithm, but instead of a single cube, the cube is divided into 6 tetrahedrons (see figure 64). This method was proposed Carnerro et al in [2], their motivation was to avoid the ambiguity in marching cubes.

No	Vertices
1	(4,6,7,0)
2	(7,6,3,0)
3	(3,6,2,0)
4	(2,6,1,0)
5	(6,5,1,0)
6	(4,0,5,6)

The creation of the tetrahedrons

Dividing the cube into tetrahedrons increases the number of possible surface intersections thus increasing the resolution of the isosurface and the number of

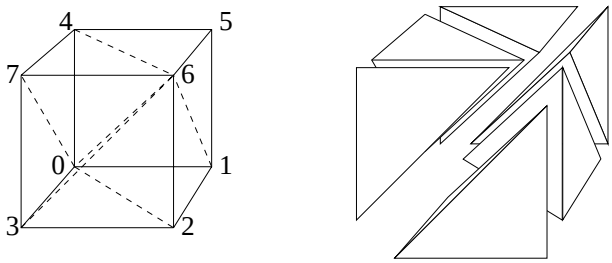


Figure 64: Dividing a cube into 6 tetrahedrons.

triangles. By connecting the vertices shown in the table above, six tetrahedrons are created.

The surface generation equal to the surface generation of the marching cubes. Each vertex of the tetrahedron can either be above of below the isovalue. The different types of intersections is shown in figure 65.

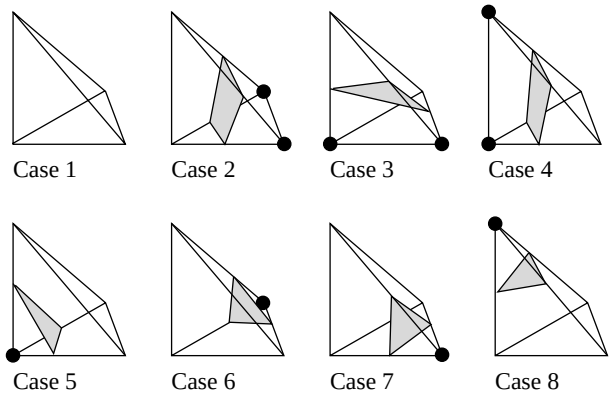


Figure 65: The 8 possible surface intersections.

Case 1 is trivial, all vertices are either below or above the isovalue, and there is no contribution to the isosurface. The remaining 7 cases contribute each with 1 or 2 triangles to the isosurface.

Figure 66 shows a comparison of the marching cubes (MC) and the marching tetrahedrons (MT) algorithm. The input image is a 64x64x30 MR scan. While both methods finished the surface generation in less than a second, the MT produced more than three times as many triangles as MC. The following table compares the results of two algorithms.

	Marching cubes	Marching tetrahedrons
Generation time	0.3 seconds	0.7 seconds
Triangles	15213	48364

Comparison of the two generated isosurfaces.

The result does not justify the large number of triangles. After applying shading to the surfaces, the rendered heads are almost alike.

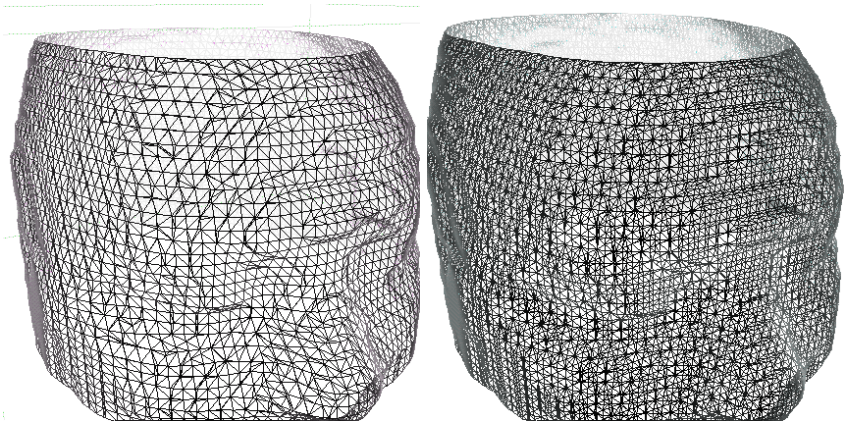


Figure 66: Left: Head generated by marching cubes. Right: Head by tetrahedrons.

7.3 Experiences with the Marching Cubes algorithm

We have mentioned two methods of generating isosurfaces in the previous sections and mentioned the drawbacks of the marching cubes algorithm. Still we have chosen this algorithm, mainly because it is very fast and it generates a very smooth surface.

The algorithm theoretically suffers from an ambiguity when it comes to interpreting the cube configuration, but we have not seen this in our renderings. Hall [8] states that this problem seldom arises, thus we have not implemented the unambiguous version.

The large number of triangles produced by the marching cubes algorithm does slow down the interaction of the rendering model a bit, and some reduction should ideally be made. Unfortunately the algorithms for reducing the triangles introduce another problem: cracks in the isosurface. When combining several smaller triangles not exactly in the same plane into larger polygons, cracks appear between the polygons. The cracks are usually closed with more triangles. The total number of triangles are reduced but the rendered model loses fidelity.

Furthermore these triangle reduction schemes work best on an isosurface with little curvature such as the forehead of the skull. Our primary visualization objects are the small vessels and therefore the isosurface we visualize has indeed a high degree of curvature and cannot easily be decimated without loss of quality and fidelity.

As fidelity is a high priority issue when constructing a pre surgical planning tool, we cannot allow the reduction of triangles to introduce cracks. We have done a study on how many triangles can be reduced if we allow adjacent triangles to merge as in figure 61. The overall decimation would be no more than 4-5 %. A typical visualization of the vessels consists of approximately 150.000 triangles for the vessels and 15.000 triangles for the head. The conclusion was a reduction of at most 1-2% of the triangles for the vessels and about 30% of the triangles for the head. Since the vessels are responsible for about 90% of the total number of triangles used, there is no point in trying to reduce the triangles for this kind of visualization.

7.4 Multimodal Viewing

As additional support for the surgeon, the anatomy of the brain is added to the visualization. The reason is that there are other types of critical tissue surrounding the vessels, and they show best on an anatomy scan (see Appendix A). Critical tissue is for example tumors of the skull base or deep seated brain tumors. Planning surgery for these lesions requires viewing from different angles and estimates of the three dimensional extent of the lesion.

The requirements were:

- The anatomy should come from a structural T1 scan (if available) rather than the blood weighted T1.
- The anatomy and the blood vessels must coincide perfectly.
- The anatomy must be placed on planes in all three directions.
- The planes must be freely and independently movable along their axis.
- The anatomy must follow the model if the window is resized

It is common to make an anatomy scan when making an MRA. The two scans are made with different modality, therefore they have to be made separately. Since they are made in sequence while the patient is restrained, they are already coregistered - i.e. they coincide in space if they have the same origo. A small misalignment can still occur depending on the restraining method.

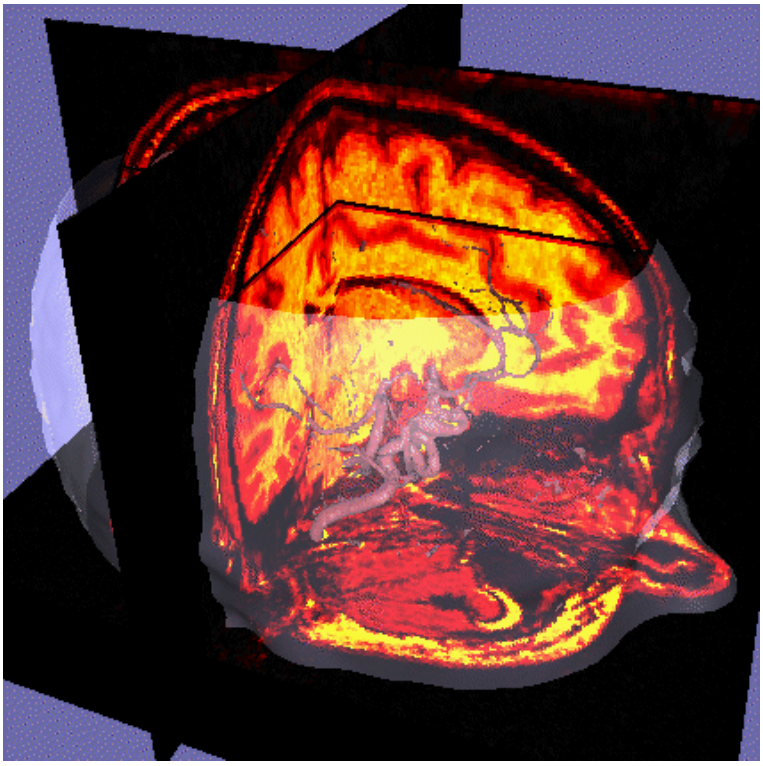


Figure 67: Brain anatomy loaded and visualized.

We solve the problem by adding three texture mapped planes to the model. The user can load the anatomy in any resolution regardless of the model, because the texture is scaled using Fants algorithm as described in [26]. Fants algorithm handles both enlarging and reduction of images. It reads the input pixels one by one and integrates the value into an accumulator. An input pixel is either entirely consumed by the output pixel, and perhaps completing it, or the input pixel completes the output pixel but is larger than the output pixel. In this case, a value is approximated using linear interpolation from the input neighbours. Both the accumulation and the interpolation yields an antialiasing effect making this rescaling algorithm a good choice.

If an anatomy scan is not available, the MRA can be used instead. Sometimes, this is desirable. Since the vessels are segmented from the MRA, these are visible on the texture, and the quality of the segmentation can be judged very directly. Missing vessels and artifacts will show very clearly. In figure 68 a bad segmentation can be seen. The segmentation procedure missed two vessels.

It is important that we do not claim perfect visualization, and that the surgeon has the possibility to check the result. The view looks very authentic and the

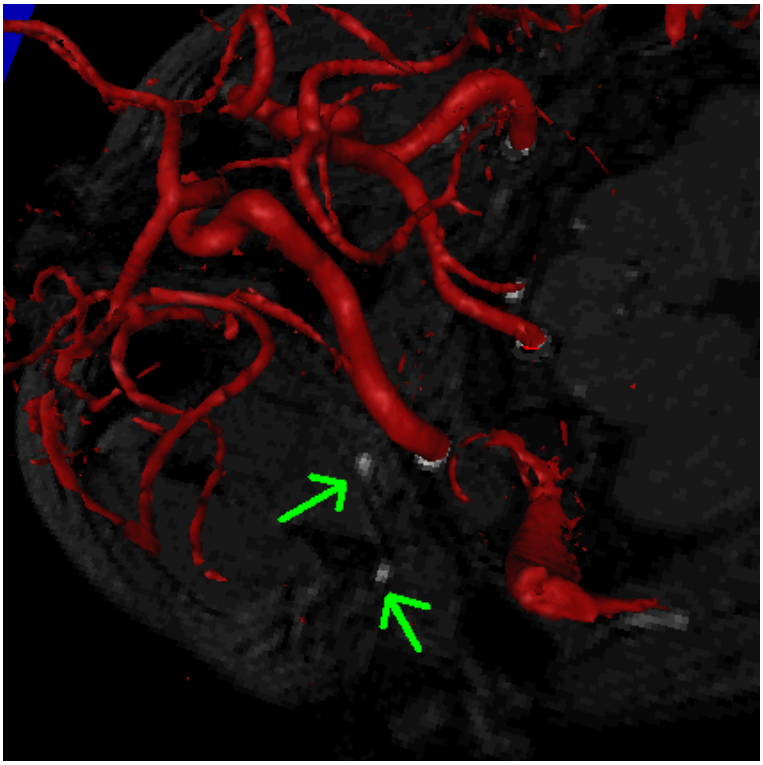


Figure 68: Arrows shows lost vessels.

user might be too easily convinced that this is a 100 % correct visualization. But human lives is at stake here, and as the case study will show, our method does not always produce perfect results, especially if the MR scanner has not been optimal calibrated. The application as it is now is still a powerful tool, but the surgeon must be aware of its limitations.

8 External error sources

Artifacts are aspects of an image that are not related to the depicted object. It is important to recognize the existence of these problems and their influence on the final result. A few of the problems described here require some knowledge of the physics involved. The basics of MR physics is presented in Appendix A.

8.1 Partial volume effect

The signal to noise ratio (SNR) is reversely proportional to the spatial resolution of the MR scan. High resolution means a very noisy image. This is a tradeoff the radiologist has to consider before each scan.

Assume a vessel with a diameter just the size of the spatial resolution. If the sampling grid coincides with the vessel position, a high intensity pixel is produced, but if the vessel runs between the cells in the grid, the produced intensity is distributed to more than one pixel (four pixels in the worst case). Figure 69 shows an example of this effect. Although the vessel has the same width, the sampled vessel is fragmented.

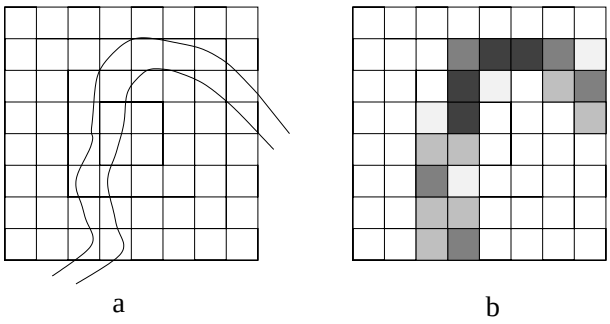
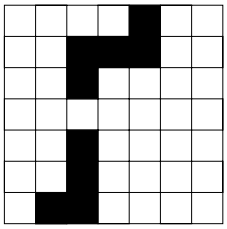


Figure 69: Partial volume effect. a) the real position of the vessel. b) Sampled vessel.

The partial volume effect cannot be bypassed. We will experience discontinuous vessels when the size goes below the resolution. A possible restoration of the vessels can be accomplished by:

- Snakes. By initiating a snake in each vessel part and give continuity high priority - i.e. low energy the longer the snake is - a possible spline interpolation between the parts can be constructed.
- Mathematical morphology. By defining several different structuring elements to catch the possible permutations the vessels could be restored. Fi-

gure 70 shows a broken vessel that would be restored by the given structuring element.



a

0	1	-1
0	0	0
-1	1	0

b

Figure 70: a. Disconnected vessel. b) Structuring element. 0 means background, 1 is foreground and -1 is “ignore”.

8.2 Chemical shift

The chemical shift artifact arises where fat and other tissues form borders. In the frequency direction, the MR scanner uses the frequency of the signal to indicate spatial position. Since water in organs and muscle resonate at a different frequency than fat, the MRI scanner mistakes the frequency difference as a spatial (positional) difference. As a result, fat containing structures are shifted in the frequency direction from their true positions. This causes a black border at one fat-water interface, and a bright border at the opposite border. Fat suppression, a special feature of the scanner, can reduce this problem significantly. The scans we acquired have all been made with maximum fat suppression.

8.3 Aliasing

Aliasing, also known as wrap around, is very common and most of our scans suffer from this artifact. It arises when the field of view is smaller than the body part being imaged. Note, that this is different from the definition of aliasing known from image processing. The part of the body that lies beyond the edge of the field of view is projected on to the other side of the image. It is very easy to correct this image if the overlap only happens in the background area, but since the artifact often only occurs outside the region of interest, we have not used much time on aliasing. Figure 71 shows a very large wrap around effect that cannot easily be nullified.

Figure 71: Severe case of aliasing.

8.4 Phase encoded motion artifacts

Phase-encoded motion artifacts appear as bright noise or repeating densities oriented in the phase direction, occurring as the results of motion during acquisition of a sequence. These artifacts may be seen from arterial pulsations, swallowing, breathing, and physical movement of a patient. Motion artifacts extends across the entire field of view. Phase-encoded artifacts can be reduced by various techniques depending on their cause and location. Arterial pulsation artifacts can be reduced by saturating pulses prior to entry of the vessel into the slices. Spatial pre-saturation can also reduce some swallowing and breathing artifacts. Figure 72 shows a motion artifact. The position and width of the line is random.

This artifact is almost impossible to avoid when making angiography since a scan takes almost an hour to produce. It is difficult to remain still for an hour, and different restraining methods are used. A common method is a frame that is mounted on the patients head. Also, screws inserted in the skull are used. When we had our brains scanned for this project, a piece of adhesive tape was used with success (see figure 73).

8.5 Ghosting

Ghosting occurs when a slice is seen as a faint echo somewhere out of sequence. Figure 72 (right) shows the reflection of the top part of the skull in the upper part of the brain. Flowing blood also causes ghosting effects, because the velocity of the blood is different during systole and diastole.

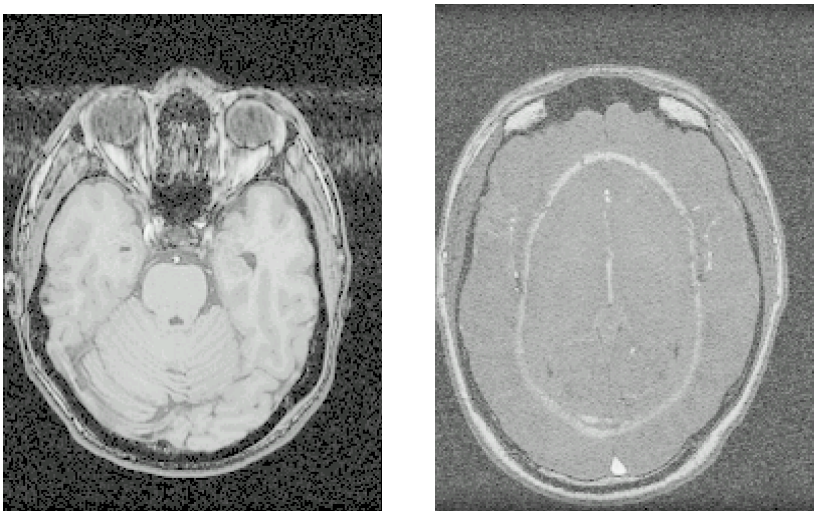


Figure 72: Left: Motion artifact visible at the top of the image. Right: Example of ghosting.

Figure 73: The patients head is restrained to avoid movement.

8.6 Non-uniform inter-slice illumination

As the MR scanner processes 5-6 cm of the object at a time in slabs, the illumination changes from slice to slice in a slab, and the intensity difference between two slabs is significantly as figure 74 (a blood-weighted T1 scan) shows.

Going up through the slices, the intensity increases at the end of a slab, and drops when entering the next slice.

We have already explained how we handle this non-uniform illumination from slice to slice. The illumination on a single slice is practically uniform. When talking to Steffen Ringaard, who is currently taking his PhD in MR-physics, we were told that the coils in MR scanners today have become so good, that bias

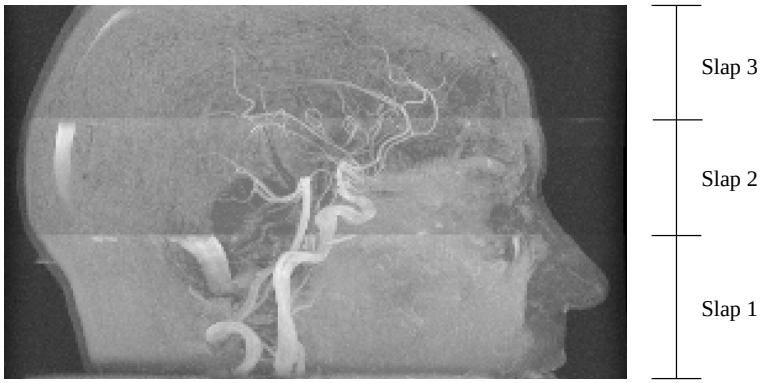


Figure 74: M.I.P. showing non-uniform illumination.

errors on single slices rarely occurs anymore. This holds as long as dealing with head scans, as we are. When scanning other body parts such as the lungs or the abdomen, another type of coils (surface coils) are used, which produces non-uniformly illuminated slices.

Figure 75 shows the scan from figure 74, in which high intensity tissue, such as fat and muscles, are removed. This figure shows another artifact with blood-weighted T1 scans. As this type of scan focus on the flow of blood, only vessels with a certain flow (the arteries) should be displayed. But as the the figure shows, the sensibility for the veins increases through a slab, and veins occur. Figure 75 shows these veins.

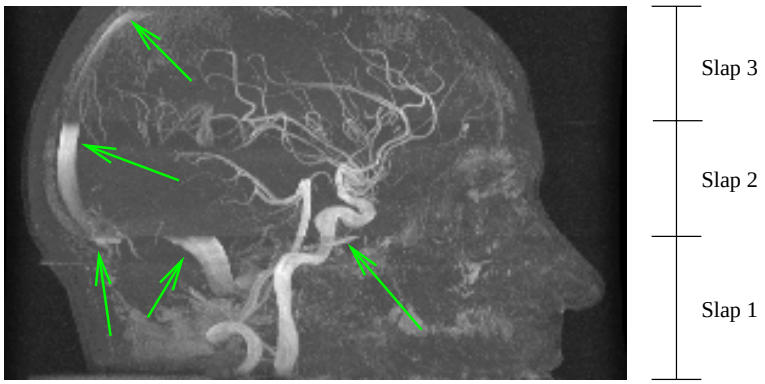


Figure 75: M.I.P. showing increasing sensibility for veins.

This is not an exhaustive list of possible artifacts. For a more thorough discussion please refer to [5]. As mentioned in the introduction to this chapter, we

do not deal with all the external error sources. This means, that there might be times when the automatic segmentation will produce artifacts or miss vessels due to these errors. Therefore, it is important that the radiologist or the surgeon takes a look at the data to determine if precautions should be made.

9 Presentation of A.S.A.P.

We have implemented the theory described in the previous chapters into a working application, the A.S.A.P (**A**utomatic **S**egmentation and **A**nimation **P**rogram).

The source code is platform independent. All that is needed is the X-windows environment and an ANSI-C compiler. For the 3D part, hardware accelerated OpenGL is recommended, but not required.

Although the program accepts command line parameters, every function can be accessed through a graphical user interface.

The end user is intended to be a surgeon or assisting nurse and should not have access to all the features described below, but for experimental and research purposes a lot of the image processing tools are available through the interface. The program should still be easily accessible to a novice user, since most of the advanced usage is hidden in the menus. However, in the finished program, the interface should be further simplified.

9.1 Implementation decisions

Currently, a number of graphics packages that manipulates images exists. For some of the more tedious work, we were tempted to utilize these libraries. It turned out that if we wanted speed, we could not rely on a generic image processing library. We had to implement the functions if we wanted the desired speed. Not only could we implement the fastest possible algorithms, but we could also direct the result towards medical images.

For the rendering, we decided to use Silicon Graphics OpenGL standard. With that in mind we could, at an early stage, try to make the data structure for the 3D models to something easily rendered using OpenGL.

The MR images came in a format called MINC (Medical Image NetCDF) which in turn is based on NetCDF from University Corporation for Atmospheric Research. The MINC format is developed by McConnell Brain Imaging Centre, Montreal Neurological Institute, McGill University. This file format is more compact than raw slice data, and is widely used at various hospitals, our contact hospital (Aarhus Kommunehospital) included. McGill University offers a library for loading the MINC files.

The program is wrapped in a graphical user interface. We have implemented the system on Unix/X-Windows using OpenGL as rendering, and also on Linux/XFree using Mesa (An OpenGL port). The system is designed to compile and run without any modifications on any machine with an ANSI-C compiler and the X development environment.

The user interface is standard X Window System created using the XForms library.

The following libraries were used during our implementation. The libraries were used only to access the data files and to display the graphics.

- Mesa GL by Brian Paul, Avid Technologies (Linux only).
- MINC by McConnel Brain Imaging Centre
- NetCDF by UCAR
- Xforms Library by T.C. Zhao and Mark Overmars.

9.2 The main screen

Figure 76 shows the main screen and a few open windows. The file selector allows the user to browse directories for Minc files. Loading a Minc file is either fully automatic, with all necessary information such as size and color scale taken from the Minc header, or the user can specify these numbers manually. For most purposes, it is not necessary to load the MR scan in full resolution.

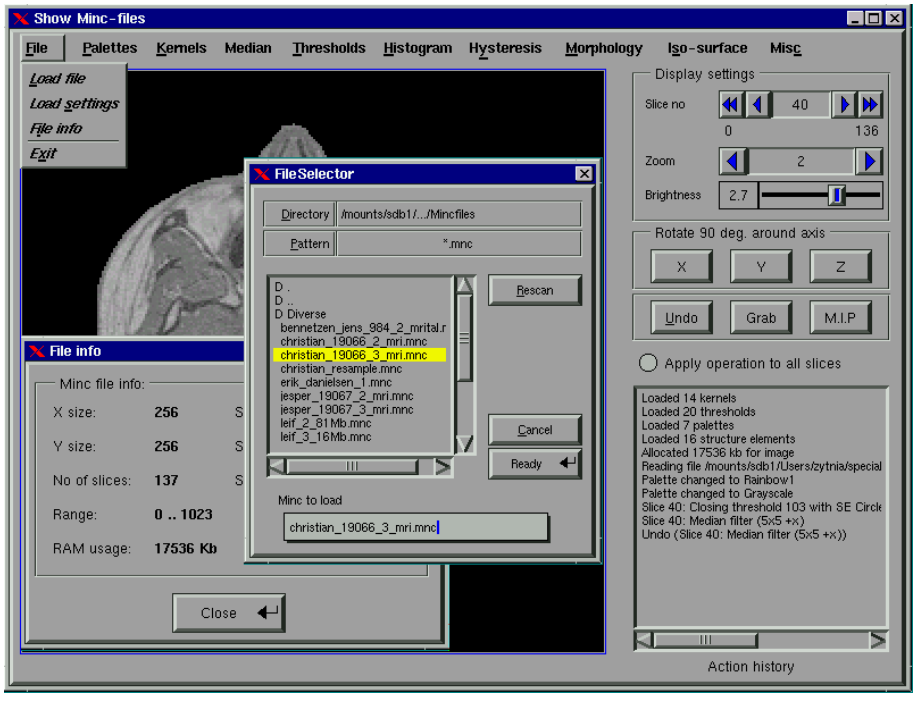


Figure 76: The main screen and a few open windows.

The palette-menu gives the user access to a variety of pseudo color palettes for easy viewing. The colors are not used in the actual picture processing.

In the automatic process, we apply various kernels as described in section 3.1.2, but we have made it possible for the user to apply different operators to the image. As figure 77 shows, the user can select either a known function and adjust the parameters, or edit a kernel up to 9x9 size. The calculation, of course, is not limited to 9x9.

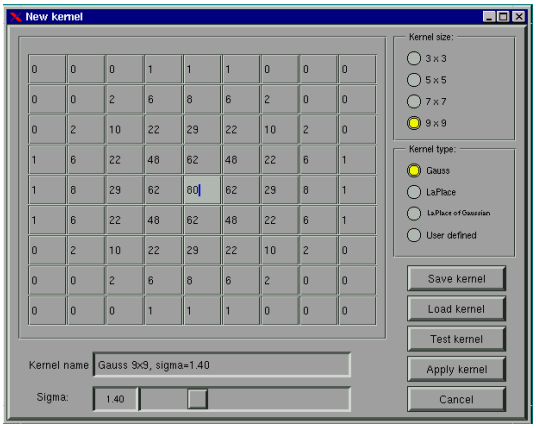


Figure 77: Creating a new kernel. Selecting a known kernel type will automatically fill in the matrix.

The undo facility works for all actions made on a single slice. The undo buffer is implemented as a cyclic FIFO (First In First Out) stack with a capacity of 100 slices. If the button *apply to all slices* is checked, no undo is available. A typical MR scan weighted for blood is over 80 Mb in size. It would be too memory demanding to store an entire scan in an undo buffer. Instead, a warning dialog box is presented each time the user applies an action to all slices.

All actions are showed in the *action window* (see figure 78). This window gives information about what actions have been taken, and other bits of information. This is especially useful when researching, because when the best image is found, all the actions are recorded for easy recreation.

When defining a new threshold, the optimal threshold and average intensity is shown as a guide (figure 78). Both sliders can be adjusted, such that either a pixel intensity can be chosen, or the amount of pixels in percent can be used. As with the kernels, a threshold can be saved to a file. When the program starts, it reads the contents of the threshold directory (along with the kernel and palette directory) and places the user defined values in a menu.

The morphology menu allows the user to apply different types of morphological operators. Both the fundamental operators and their derivatives are available. Figure 79 shows the choices available for closing. A.S.A.P is able to handle 2D

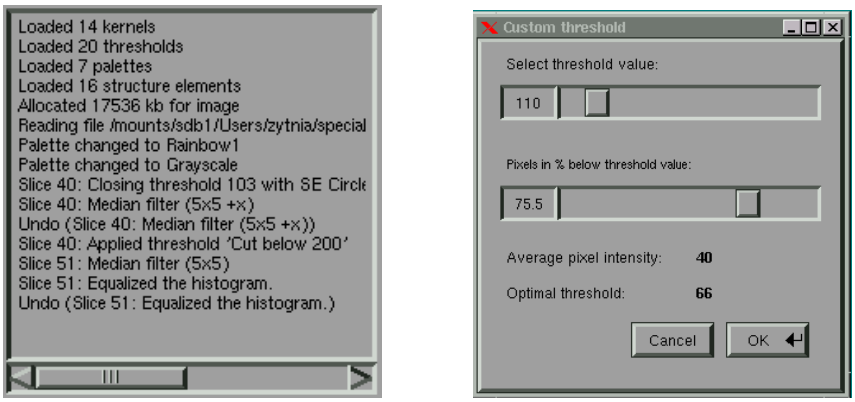


Figure 78: Left: Action history window. Right: The custom threshold window.

structuring elements of size 40 $\times$ 40, and 3D structuring elements of size 11 $\times$ 11 $\times$ 11.

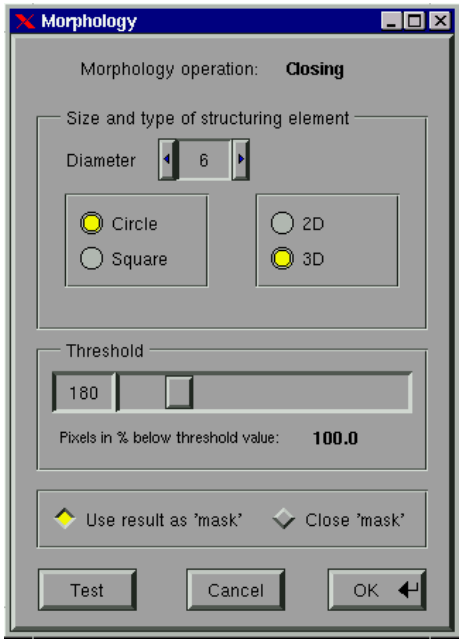


Figure 79: When using morphology, the user can choose the size and shape of the structuring element, as well as 2D/3D.

The surface generation is done from the isosurface menu. General options, such as which algorithm to use (marching cubes or marching tetrahedrons), are

also to be found in this menu. The user has the option to save the generated 3D model to disk for later viewing in a standalone program.

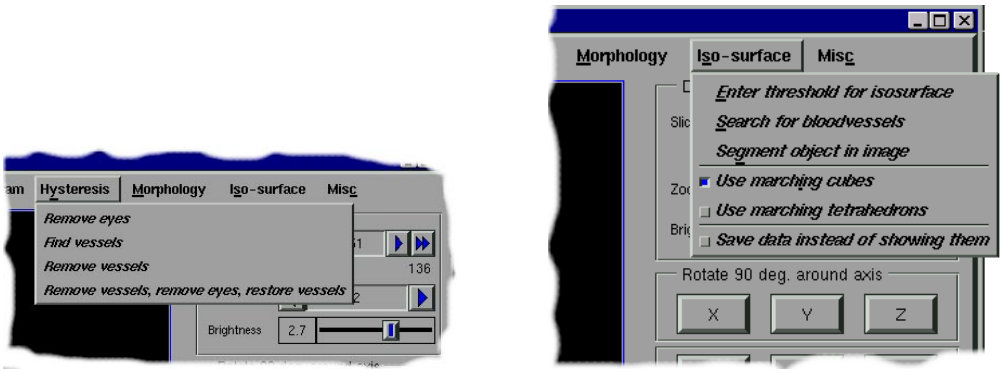


Figure 80: Left: Hysteresis menu. Right: isosurface menu

Some of the entries in these menus are not meant for the end user, since they are steps in the automatic process, but while researching it is desirable to follow the individual procedures and their result.

The remainder of the interface offers the possibility to rotate the voxel space, grab the current slice and save it to a file, and generate a Maximum Intensity Projection.

9.3 The visualization window

Once the model has been created, a new window is opened in which the model can be animated.

The visualization program is a stand alone program that communicates with the main application through shared memory. Due to the large amount of data, we wanted to avoid copying the models. Instead, we created a chained list of memory blocks of limited (1 Mb) size. This list is given a unique ID, which the visualization program can use to reach the data with. This program runs as a stand alone program as well, taking a filename as parameter. The file should already be made from within the main program.

Via sliders, the user can rotate the figure and zoom in and out. The same functionality is available using the mouse on the picture. Moving the mouse with the left mouse button pressed will rotate the image. The middle button enables zooming while the right button moves the viewpoint. The sliders follow the movement automatically and updates to reflect the user defined view. If the window is resized, the model is resized accordingly. The position of the texture mapped cut planes is adjustable via slides as well.

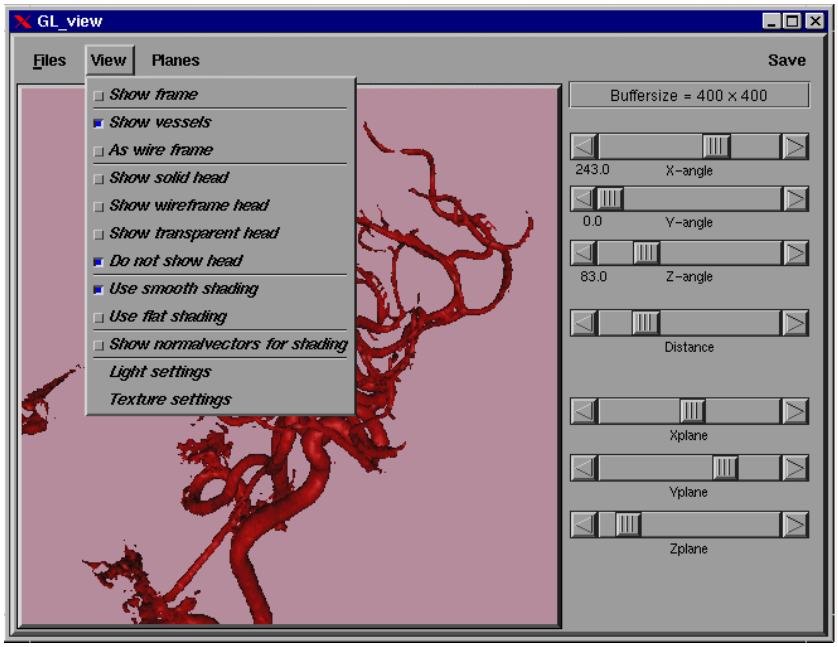


Figure 81: The 3D model view.

The “view” menu offers several different viewtypes. The shading types and the display of normal vectors is for research only. The “files” menu offers the possibility to load a different modality and display it along with the geometric model.

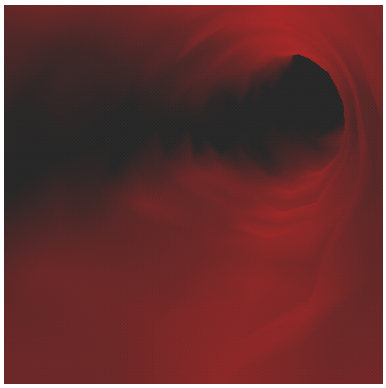


Figure 82: It is possible to take a tour *inside* the vessels.

The model can be saved either as a single graphics file or a complete 360 degrees animation can be generated by the "save" option. As a special feature, a

movie can be saved where the transparency fades from opaque to transparent.

If the vessel tree is large, there is the possibility to hide the vessels or just display a wireframe while rotating to find the optimal viewpoint. This is useful if the machine does not have hardware support for OpenGL.

The head can be shown in four different modes - a solid view of the head, a wireframe model, a transparent head and no head at all. The different views are shown in figure 83.

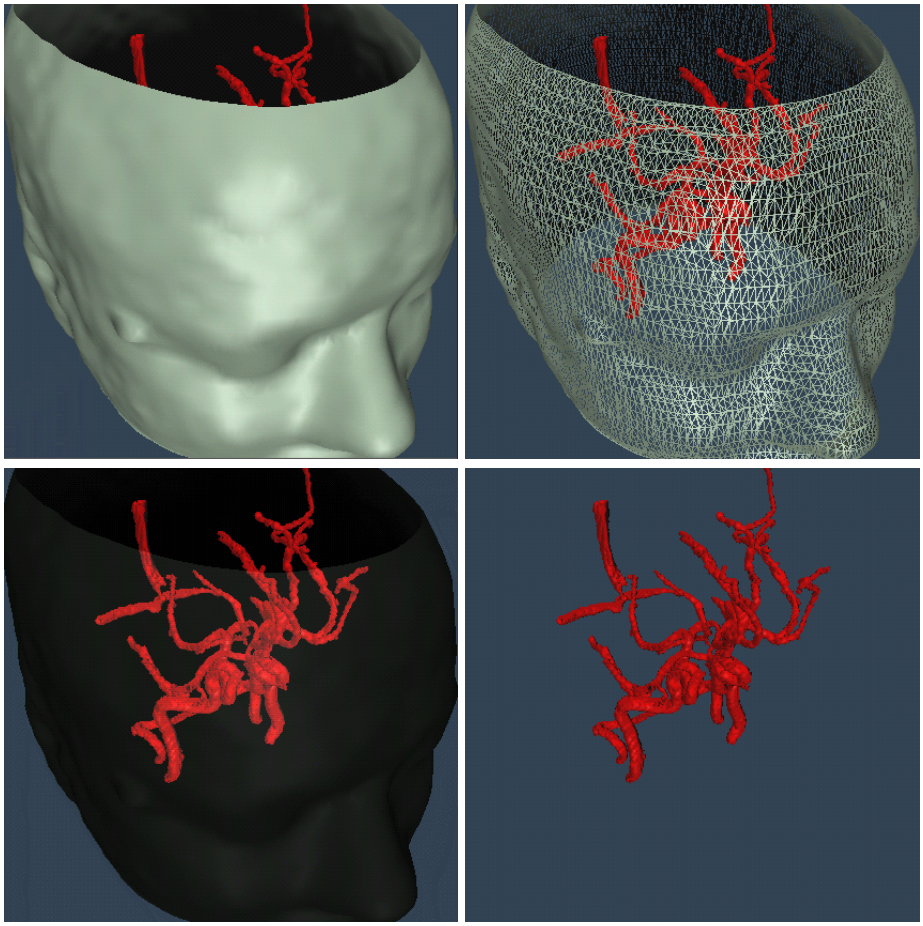


Figure 83: The four different viewtypes. Solid head, wireframe, transparent and invisible.

The wireframe makes the animation faster, and was a valuable tool when developing the isosurface algorithms, because the actual triangles that form the head are seen.

Via the “view” menu a new window can be opened in which the user can specify the rendering parameters such as material, lightning, transparency, color,

specularity etc (see figure 84). The default values should satisfy the average user, but if the images are to be used in a different media such as in print or on the Web, it is preferable to be able to adjust the model accordingly to produce the best appearance of the pictures.

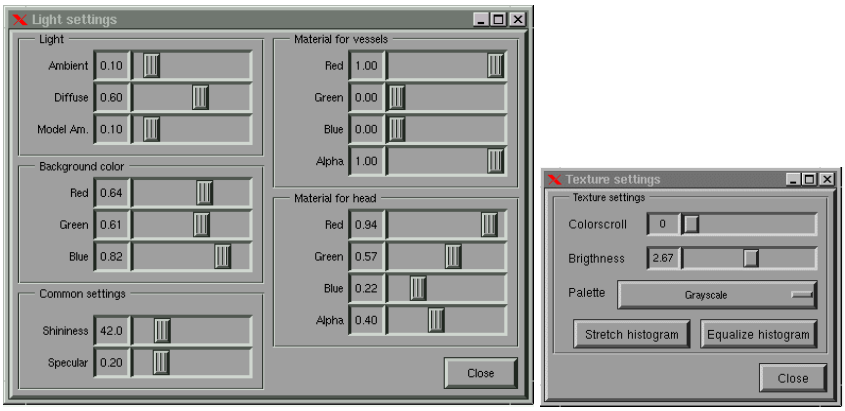


Figure 84: The settings window for lightning and texture properties.

There is no “apply” button in any of the model windows, because all changes are reflected real time on the model.

10 Case study

We present the result of four actual applications of the automatic segmentation procedure. The scans differ in both quality and content. Ideally, we would have several more scans to compare, but we have not been able to acquire more than these four scans.

All screenshots has been made from the same angle under the same conditions. For each of the four subjects, a MIP before and after the segmentation is shown. Notice the amount of high intensity pixels removed.

The two first scans suffer from a number of artifacts. The scanner just had the gradient coils replaced and the scans were made before they were calibrated. We have included them to show a worst case scenario, but also because these are the only complete scans we have available. The rest of the test data shows different parts of the head and is not directly comparable to these.

All the vessel trees shown has been rotated to the same position as figure 85.

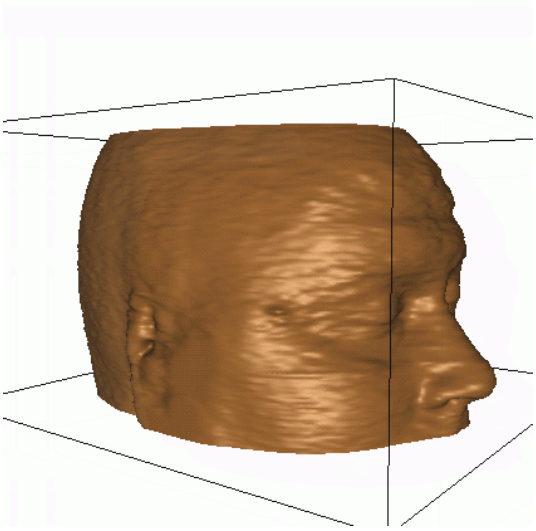


Figure 85: The vessel trees conforms to this position of the head.

10.1 Case 1

Although this is one of the malformed scans, the result is quite good. The inhomogeneous slabs are adjusted perfectly and most of the noise is removed. Two of the slabs has veins showing in the back of the head. The noisy area behind the eyes has been removed and the remaining noise is fat from the cheek and neck. The small vessels, albeit disconnected, are still visible. Connectivity is only used on the large vessels. At the last step of the segmentation, simple threshold is used because at that point, everything above a certain threshold is vessels. This enables us to catch the small sub-voxel vessels, as figure 87 shows.

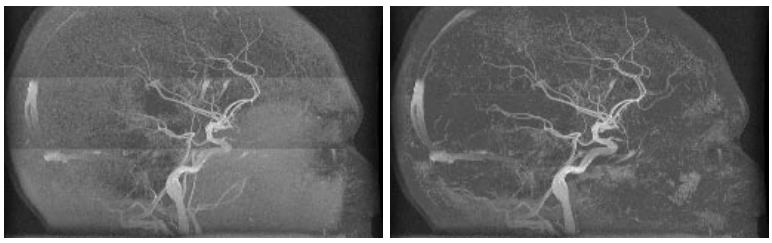


Figure 86: Left is before noise removal. Right is after.

Figure 87: Fully automatic segmented vessel tree of person 1.

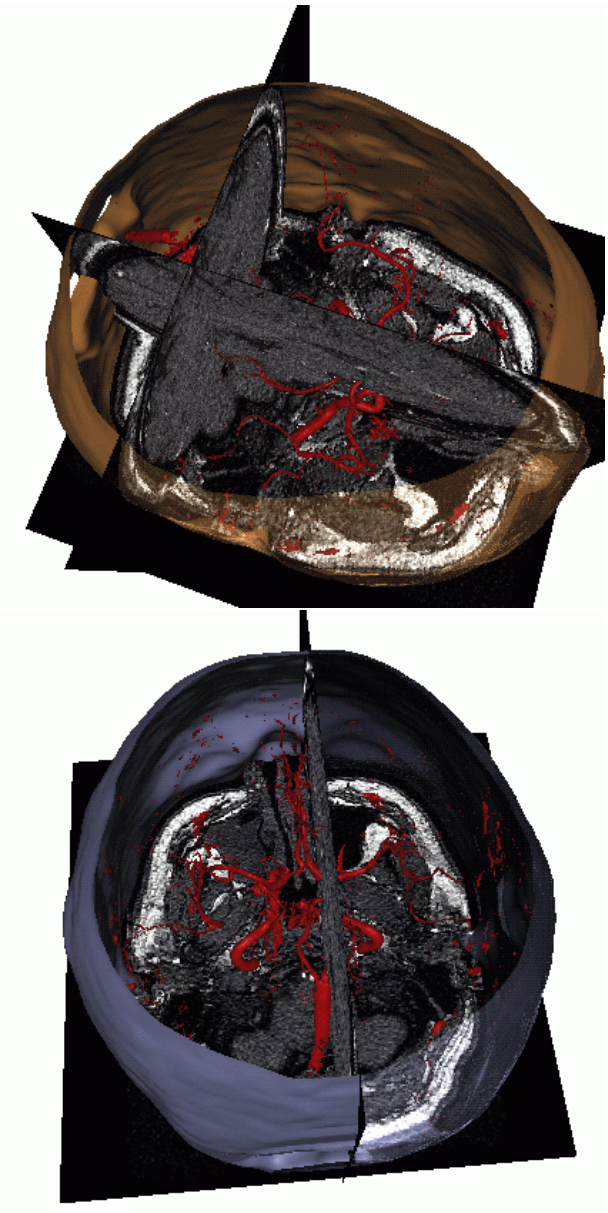


Figure 88: Two different views of the result from the top. The vessels shows clearly on the B/W texture.

10.2 Case 2

This is the worst of the scans. Not only is it made with an uncalibrated coil, the subject moved during the scan causing motion artifacts. The scan is not made with the same parameters as the others, as the weighting of flow is not as high. The vessels are showing, but not nearly as bright as the other cases. The contrast in the two first slabs is much to low, and it shows on the MIPs (figure 89). As figure 90 shows, the amount of noise in the vessel tree is high. Even in the presence of this much high intensity noise, our automatic segmentation produces fair results. However, the automatic segmentation is, among other things, based on the cumulative histogram and the assumption that the remaining pixels above a certain threshold is vessels. Because the noise moves the scale, this segmentation misses a number of small vessels.

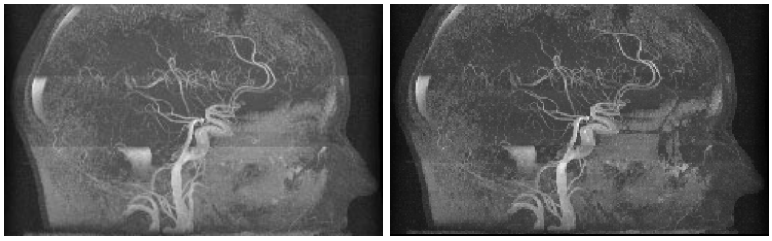


Figure 89: Left is before noise removal. Right is after.

Figure 90: Fully automatic segmented vessel tree of person 2.

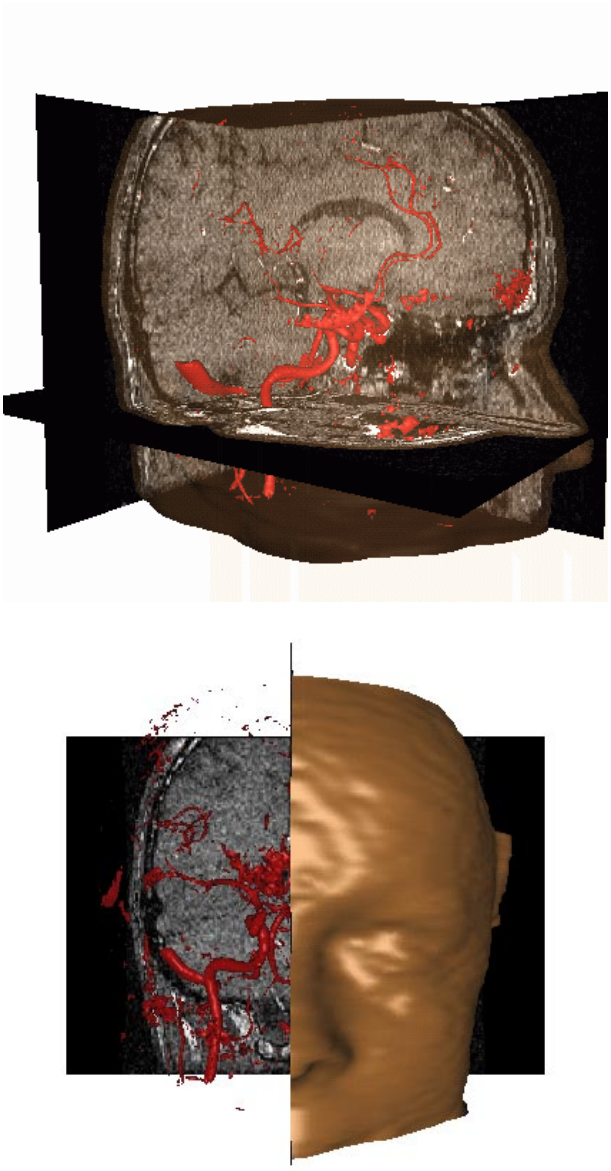


Figure 91: Although an imperfect scan, the result can be made to look decent.

10.3 Case 3

This is a good segmentation. The artifacts are the veins in the neck and the profile of the teeth as seen in figure 93. Otherwise, the vesseltree is almost complete with both big and small vessels. The scan was made with maximum fat suppression and with 156 slices instead of 137. The high resolution enables us to picture a complete vesseltree and also a very detailed head as shown in figure 94. This shows that the calculation of the structuring element used to segment the head is good. The fine structures of the ear and the nostrils are clearly shown.

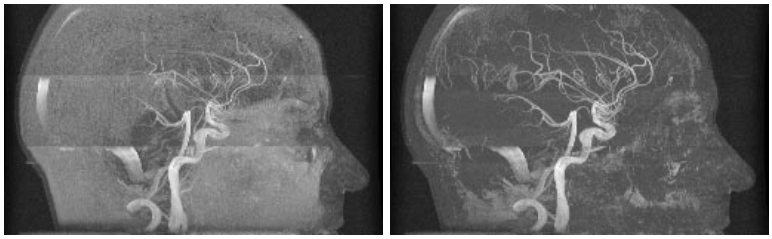


Figure 92: Left is before noise removal. Right is after.

Figure 93: Fully automatic segmented vessel tree of person 3.

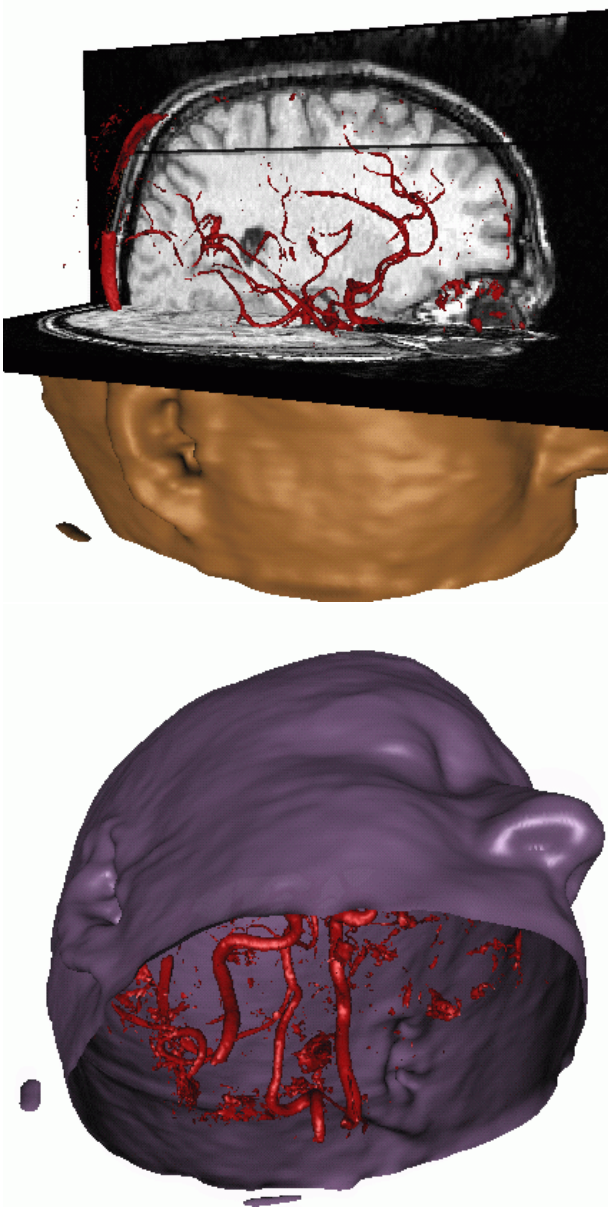


Figure 94: The head shape of this scan is so fine that even the structures of the ear can be seen.

10.4 Case 4

This scan was made in a lower resolution than the others, because the subjects head could not fit into three slabs. The intensity gap between the slices has been equalized perfectly. As the others, the veins in the neck are showing. Figure 96 shows a little more noisy vesseltree than the previous. This is due to fat in the neck area and on the side of the head. However, as the MIP on figure 95 shows, quite a large amount of unwanted high intensity pixels has been removed and the result is still satisfying.

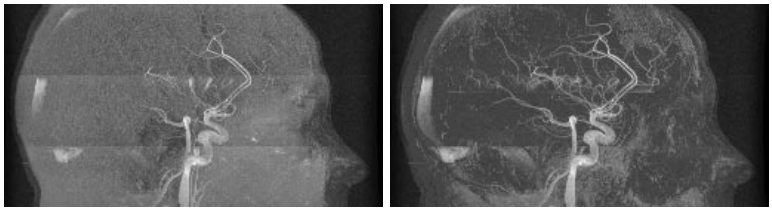


Figure 95: Left is before noise removal. Right is after.

Figure 96: Fully automatic segmented vessel tree of person 4.

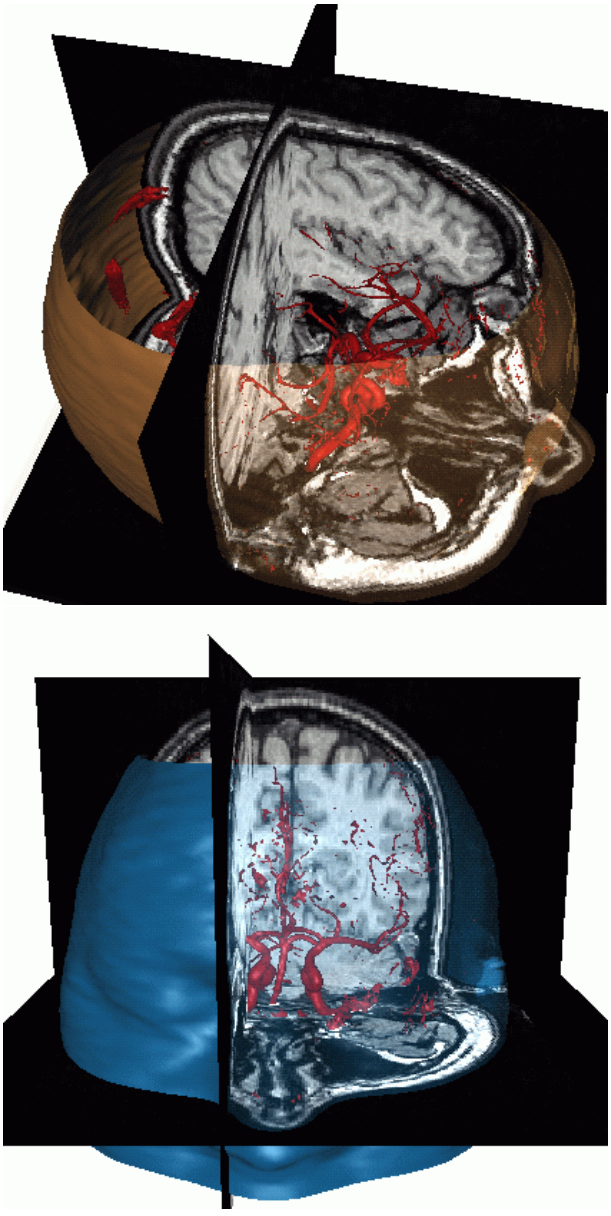


Figure 97: Using different colors sometimes enhances the contrast and visibility.

11 Conclusion and further work

We have presented a technique for automatic segmentation and visualization of the cerebral arteries. Some work remains to be done, but already the implementation has proven to be innovative and useful. The idea of making the process automatic holds, and should definitely be further explored. We have not dealt with any kind of post processing of the result. This might enable us to remove the remaining noise and connecting the broken vessels.

In the "further work" section we will sketch a few of the things remaining. The list would grow as the work continues, and is only meant as a list of starting points. Although we have several references throughout the thesis, none of these try to accomplish the same as we do. This makes it difficult to compare our project to other related works. The segmentation is based on existing methods, whereas the specific usage and the combination is a novelty. The type of visualization we present which can be animated is a rather new area, simply because the computers until recently were too slow to manipulate this amount of data.

Unfortunately we had only a limited number of brain scan available for our project. Although the brain scans available provided us with the necessary test data, the project would definitely benefit from a more comprehensive analysis on several sets of data. We are confident that the available brain scans was representative for healthy human brains. Unfortunately we were not able to test our project on brains with different kinds of tumors. It would have been very interesting to examine the segmentation and visualization results, as brain tumors may push the blood vessels in the vicinity of the tumor to new locations.

The empirical test on new data should be followed by a clinical test by surgeons in a realistic environment. Before the program is taken into use, precise documentation of inherited inadequacies should be made, and the users should go through a special course to acquaint themselves with the image generation procedure of the program. It is easy to be blinded by the authoritive look of the visualization, and such courses should teach the user to use it not as a direct representation of the inside of the head, but rather as a comfortable guide and valuable tool.

After a showcase where we presented our work, we had M.D Leif Østergaard evaluate our work. The signed evaluation is shown on the next two pages. We were pleased to read that our implementation (A.S.A.P) is ranked among the finest existing pre surgical planning programs.

The theory and work presented in this thesis produces sound results, and should definitely be pursued.

ÅRHUS
UNIVERSITY HOSPITALS

Department of Neuroradiology and
Positron Emission Tomography Center
Nørrebrogade 44
DK-8000 Århus C
Denmark

Leif Østergaard, M.D., B.Sc., M.Sc.
Research Fellow
Fax +45 89 49 30 20
Phone +45 89 49 33 55

Århus, May 20, 1998

Evaluation of A.S.A.P.
(**A**utomatic **S**egmentation and **A**nimation **P**rogram).

Based on a demonstration of the program on May 10, 1998, this evaluation comments on its medical and research applications, based on a physicians needs and other, available soft-ware products.

1. Utility of ASAP for Segmentation of Magnetic Resonance Angiographies (MRA).

Over the past years, imaging vessels by Magnetic Resonance Imaging (MRI) has improved significantly, allowing diagnosis by a fast, non-invasive technique. Due to the underlying physics, the most commonly used MRA technique (Time of Flight angiography – TOF) shows high signal intensities both in vessels and in subcutaneous fat tissue. On maximum intensity projection (MIP) images, important features are therefore sometimes missed. On some recent scanners, costly post-processing tools allow the physician to manually segment away subcutaneous tissue. Needless to say, this is a very time-consuming and thereby expensive process, requiring a trained persons work for substantial periods of time.

A.S.A.P is designed to handle segmentation automatically, thereby freeing valuable time for the physician. The segmentation process seen during the program demonstration consisted of few, logical steps, and was very quick compared to even the more advanced, interactive tools. The quality of the segmented image was of high quality, at least comparable to that produced by a trained user of a MR segmentation console. The design of the algorithms is such that one could expect this to produce reliable, segmented images without loss of important, anatomical information.

A.S.A.P. offers the further advantage that the 3D, segmented image can be viewed on the screen: In contrast, most commercially available products produce MIP images from a few, preset angles, with the subsequent loss of valuable, 3D-information.

2. Utility of ASAP for Surgical Planning.

During the past decade, several new imaging techniques have emerged, allowing detailed, 3-dimensional imaging of brain anatomy (Computerized Tomography –CT and Magnetic Resonance Imaging, MRI), as well as brain function (Functional MRI – fMRI and Positron Emission Tomography – PET). This allows detailed localization of pathologies (e.g. tumors) as well as structures that are crucial not to damage during surgery (e.g. major vessels, centers for understanding and generating speech). To utilize the full potential of these new imaging modalities,

tools must be developed that allow the surgeon to simultaneously display these modalities. Only thereby can a minimally invasive surgical approach be chosen, thereby minimizing risks of bleeding or damaging important structures. This involves for example viewing anatomical T_1 images together with images of brain vessels (MRA).

A.S.A.P allows the user to easily choose the location and nature of the anatomical information in two planes (for the demonstration, this was a T_1 weighted image) relative to the functional information (for the demonstration this was a MRA). By allowing the user to easily choose viewing angle, location of anatomical data and the transparency of overlying tissue, a very impressive, 3D visualization has been achieved. By allowing the user to use other functional data, e.g. co-registered PET data showing the location of vital structures, the program seem of great value to surgeons.

3. Visualization of images.

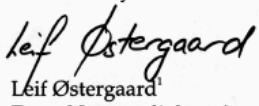
The program utilizes a widely used data-format (minc), allowing a moderately trained researcher to read raw images from any imaging modality and co-register them to 3D MR images (like those used for the presentation).

A.S.A.P. has an impressive number of built-in image processing tools and color schemes, allowing one to visualize data acquired with different modalities. The program will thereby greatly facilitate presentation of 3D data, collected across many modalities.

4. Conclusion.

When compared to existing tools for segmentation, animation and display of medical images, the program is well thought out, seemingly robust, and contains a remarkable number of features

Aarhus, May 20, 1998.



Leif Østergaard[1]
Dept. Neuroradiology/
PET-Centrer,
Århus University Hospital.

[1] *Leif Østergaard earned his M.Sc. in Astrophysics from the Institute of Physics and Astronomy at Aarhus University in 1992. He earned his M.D. from the Faculty of Health Sciences at Aarhus University in 1994. He is now finishing a Ph.D. thesis on PET- CT- and MR-scanning techniques of the brain. L.Ø. has worked extensively with MR-image analysis and visualization tools at Massachusetts General Hospitals NMR-Center. These were mostly programs developed by graduate students from Massachusetts Institute of Technology (M.I.T). The comparisons with commercial MRA segmentation tools is based on work with the General Electric Advantage Windows Console at the Department of Neuroradiology, Århus University Hospital.*

11.1 Further work

Both the theory and the application are open for further work. Some of the ideas in this chapter are our own, some originated with the doctors. What we would like, as computer scientists, is to further elaborate the theory behind the process and maybe even improve some of the algorithms. Also, we have some ideas to further automate the process.

11.1.1 Macro language

To complete the idea of a fully automatic system, we need to get rid of the mouse clicks and menu selections the current system requires. We have been working on a macro language which should be able to record a series of actions and save them to a named file for later playback. Since the macro program would be generated from within the program, parsing the program wouldn't need much error checking if the generation is correct.

In order to keep the macro simple, not all keystrokes and menu selections will be recorded. Changing palette for instance is of cosmetic value only and should not be recorded.

The framework should remain as it is now, i.e. we should not have to introduce a second version of the procedures just for the macros. This would mean that the procedures should know as much about the settings as possible. For example the "apply all" option could be implemented in two ways:

```
if (apply_all == TRUE) {
  for (i=0;i<=slices;i++) {
     <<call some procedure>>
  }
}
```

or, all procedures could take an argument telling the procedure whether it should run once, or on all slices.

```
some_procedure(arg1,arg2,..., apply_all=TRUE)
```

and then let the procedure itself handle it all. The latter is preferred to simplify the macro language. Instead of an actual language with loop structures etc, a simple listing of procedure calls with their arguments would suffice. The macro facility could be accomplished by the use of Tcl. Tcl is a textual programming

language with a simple syntax. It is also a library package that can be embedded in application programs. The Tcl library consists of a parser for the Tcl language, routines to implement the Tcl built-in commands, and procedures that allow each application to extend Tcl with additional commands specific to that application. The application program generates Tcl commands and passes them to the Tcl parser for execution. Commands may be generated by reading characters from an input source, or by associating command strings with elements of the applications user interface, such as menu entries, buttons, or keystrokes. When the Tcl library receives commands it parses them into component fields and executes built-in commands directly. For commands implemented by the application, Tcl calls back to the application to execute the commands.

The application could benefit from Tcl support because a lot of the functionality could be easily accessed by a user without having to read through the sometimes complicated C source. The addition of Tcl support would also omit the use for a proprietary macro language, since the macro functions could be implemented as Tcl routines as well.

11.1.2 Virtual Reality

The model, albeit in 3D, is displayed on a regular monitor with a 2D screen. For a true depth illusion additional hardware is required. The most affordable of these is the Shutter Glasses. The purpose of these glasses is to simulate the way the eyes work. First, the left eye is allowed to see, while the right eye is shut (hence the name shutter glasses). The viewpoint is then moved according to the distance between the eyes, and the right eye is allowed to watch. Since this happens 120 times a second, no flickering can be seen.

With a 6D (x, y, z, yaw, pitch, roll) input device and a pair of shutter glasses as the only extra hardware needed, the doctor would be able to move around inside the model in real space.

Recently, a new area in Virtual Reality (VR) has begun with the introduction of the Virtual Workbench. The Virtual Workbench displays stereoscopic images on a table top - or workbench - surface that is viewed by a group of users around the table. Using stereoscopic shutter glasses, they observe a 3D image rising above the tabletop. By tracking the movements of the user by means of electromagnetic 3D trackers, the Workbench allows an intuitive interaction with the model. The other members of the group can see the same thing, but only the user equipped with the VR glove can make interactions.

Computer assisted surgery represents a rapidly emerging area of both medical and computer science research, combining a number of disciplines. A virtual workbench would enable the user to visualize and rehearse clinical, surgical procedures. In this context, surgery simulation systems can ideally provide a safe and

realistic method for training clinicians in a variety of surgical tasks. This is the motivation for exploring this area.

Our basic work is unchanged regardless of the visualization method, be it on a monitor or on a high end virtual workbench.

The use of Virtual Reality started as an improved graphical user interface for the end user, but since the equipment is fairly expensive, other usages have been explored. In [12], Serra et al presents a method to connect vessels that have been separated during the segmentation process. Using a virtual workbench, the user can drag lines in 3D space through the vessels. If a vessel is disconnected, but has a line through it, their software interpolates between the two components, creating one vessel. Their results are good, and they have implemented a visualization akin to ours, using the Virtual Workbench. Only drawback is the amount of time it takes to produce the vessel tree, but since their system is not meant as a real time system but rather as a place where students can rehearse, the segmentation should only be done once, which justifies the tedious process. It could be interesting to combine our segmentation theories with this type of visualization. This would mean that the surgeon could take a test drive through the vital areas before commencing the actual operation. Combined with force feedback gloves and a high resolution workbench, it would be almost like a real surgery - only with an “undo” button.

11.1.3 Connecting fragments of small vessels

Because of the partial volume effect, the blood vessels will be fragmented when their size approaches the size of the voxels. It would be interesting to use active contours like snakes to connect these fragments. Figure 98 show how this would look. The vessels are taken from one of our datasets, and the spline is drawn manually.

It would probably be a tedious work to do this manually and a real challenge to it automatically.

In [12] Serra et al. introduces a VR environment for an interactive vessel tracing.

11.1.4 Enhancement of the blood vessels

We have presented methods to reduce noise in the data before the actual segmentation of blood vessels. Since the boundaries of the vessels are usually smooth and fuzzy on a noisy background, edge-based segmentation fails. An interesting approach would be to enhance the blood vessels before the actual segmentation.

Applying multiple convolution kernels, each designed to detect vessels of a given width and direction would be one approach.

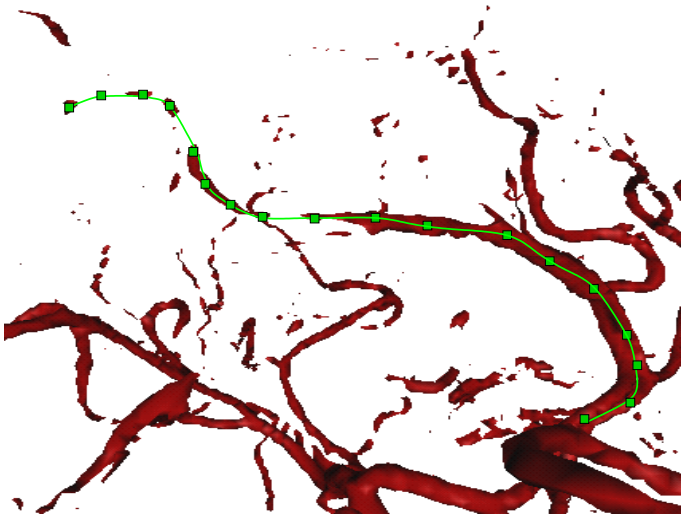


Figure 98: Fragmented vessels could be assembled with active contours.

Another approach could be multiple morphological structuring elements, also designed to detect vessels for a number of directions.

We have recently read about a third technique, namely applying a three dimensional nonlinear anisotropic diffusion filter (NADF). This technique is derived from a standard Gaussian kernel, but reduces the noise in the image without blurring the frontiers between different regions. The NADF was introduced by Perona and Malik in 1990, and is defined as a diffusion process that encourages intra-region smoothing while inhibiting inter-region smoothing. The technique is complicated, but promising.

11.1.5 Segmenting and visualizing the brain

Currently we are not able to segment and visualize the brain as a geometric model at the same time as visualizing vessels, as we have focused solely on the vessels. Automated brain segmentation and visualizing would enhance the usability of our system. The brain segmentation should be able to label the different regions of the brain, such that they can be visualized in i.e. different colors. On the wish list is the ability to work on yet another modality, the CT scan. Given a coregistered CT scanning of the head, we would be able to visualize the skull as well. The segmentation and labeling of the different parts of the brain is definitely possible with todays tools, but is beyond the scope of this thesis.

11.1.6 Framework

A number of the tools developed during this process were made as general as possible. That enabled us to change different parameters during the programming without having to edit all the procedures. To exploit these tools even more, a general image processing library could be created, giving other users easy access to the procedures. Since the procedures are almost ready to be compiled into a library, all that is needed is documentation on the functionality and description of the framework.

A Basic physics of Magnetic Resonance Imaging

Magnetic Resonance Imaging is basically the study of hydrogen atoms. When a positively charged proton spins around its axis, it creates a magnetic field.

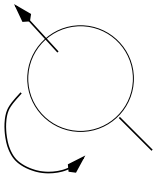


Figure 99: Protons spins around the axis.

If the protons are outside the magnetic field, the orientation is random, and thus the resulting magnetization is zero.

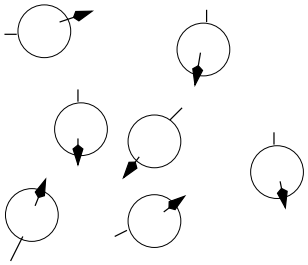


Figure 100: Random spins resulting in a non magnetic field.

When they are under the influence of a magnetic field, however, they align themselves according to the current. There will be an excess of up-spins. This is due to the fact that up-spins are at a lower energy level than down-spins. This property will be used later on.

Actually, they are not aligned perfectly parallel to the field. They precess (see figure 101) with a certain frequency.

The frequency is given by the Larmor equation 38.

$$w_0 = g \times B_0 \tag{38}$$

Where w_0 is the frequency in Hz, g is a constant called the gyromagnetic ratio and B_0 is the external magnetic field in Tesla. The scanner that provided us with MR images has a magnetic field strength of 1.5 Tesla (this is approximately 30,000 times the strength of the earth's natural magnetic field).

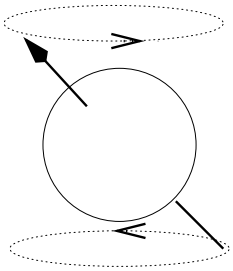


Figure 101: Precession.

This is the corner stone in MR. A radio frequency pulse (RF) is emitted with the purpose to disturb the alignment of the protons. This can only be done with the correct frequency, which is calculated using equation 38.

After emission of an RF pulse, some of the spins gain energy and go to a higher level of energy, i.e. they go from being up-spins to down spins. The result is that the spins all point in the same direction and are in phase. At this point, the RF pulse is switched off, and the protons return to their previous state of energy - relaxation. The time to relax is called T1.

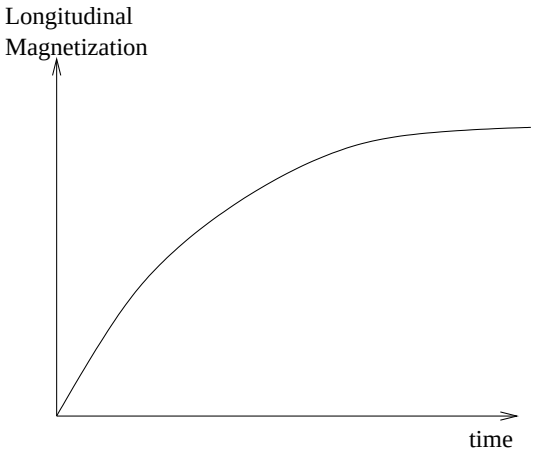


Figure 102: The longitudinal relaxation time T1.

The T1 is tissue specific. This allows us to distinguish between the different types of tissue in the brain.

When the RF pulse is switched off, the spins also lose the phase. The transversal magnetization created by the phase spinning is losing its magnitude - it relaxes. The relaxation time for this is called T2.

The T2 curve is also tissue specific. Depending on the feature of interest, the

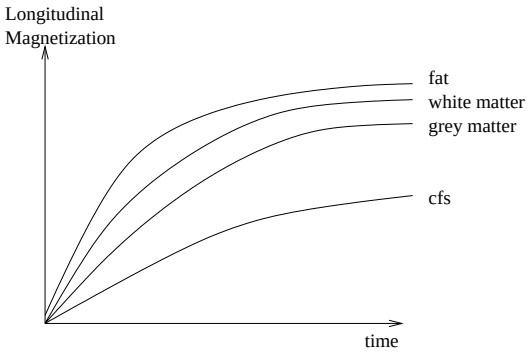


Figure 103: Different T1 types.

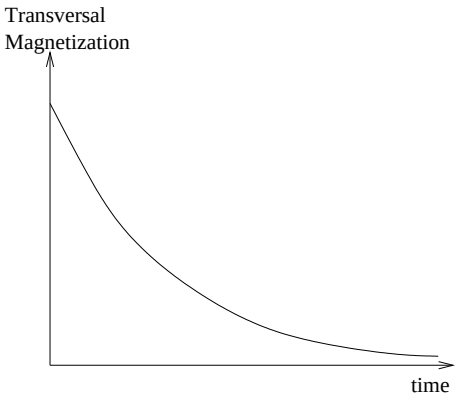


Figure 104: The transversal relaxation time T2.

MR scan is either T1-weighted or T2-weighted. T2-weighted images makes fluids bright. A T1 image would make the fluid darker than the solid tissue. However, a T1 can be made to detect flow, and this will make the flowing blood bright even on a T1. A T2 would show both veines and arteries, while the flow weighted T1 only shows arteries.

To create the slices that we use, a second magnetic field is introduced. It is a large coil that goes all around the body. It moves one step for each slice. The size of the step and the frequency range determines the thickness of the slice. This is the z-axis. To get the spatial information - the x and y axis, a new gradient field is introduced. This field decreases from left to right (and another one from top to bottom). This means that depending on the position, the protons emits different frequencies. We end up with a picture completely described in the frequency domain. It is now only a matter of doing an inverse Fourier transformation to get a spatial image.

B Glossary of terms

1. **Aneurism**: Aneurism is a weak or thin spot on a blood vessel wall. The weak spots that cause aneurisms are usually present at birth. Aneurisms develop over a number of years and usually do not cause detectable problems until they break.
2. **aMRI**: **a**natomical **MRI**, as distinguished from functional MRI or fMRI.
3. **AVM**: **A**rterio-**V**enous **M**alformations.
4. **CT**: **C**omputerized **T**omography. From the Greek word tomos, meaning "slice" or "section" and graphia, meaning "recording".
5. **Coregistered**: Two different scans of an object are coregistered if they are aligned in space.
6. **EM**: **E**xpectation-**M**aximization. A technique used to bias correct images.
7. **Ischemia**: Blood shortage.
8. **MESA**: Software OpenGL renderer.
9. **Modality**: The term modality is used of one kind of scan, that may be MR, CT, PET or SPECT. T1-weighted MR and T2-weighted MR are also said to be of different modality. When combining multiple modalities, the result is described as multi-modal.
10. **MIP**: **M**aximum **I**ntensity **P**rojection.
11. **MINC**: **M**edical **I**mage **N**etCdf. A multimodal, multidimensional image format specifically developed for brain mapping applications. Image volumes are referenced by stereotaxic coordinates, in mm, rather than by voxel coordinates.
12. **MR**: **M**agnetic **R**esonance.
13. **MRA**: **M**agnetic **R**esonance **A**ngiography. Flow weighted T1.
14. **MRI**: **M**agnetic **R**esonance **I**maging.
15. **NADF**: **N**onlinear **A**nisotropic **D**iffusion **F**ilter.
16. **PET**: **P**ositron **E**mission **T**omography. When positron emitting glucose is injected into a vein in a patient and a PET scan is taken, the resulting image of the brain reveals a topographic display of the amount of glucose utilised by different parts of the brain.

17. **RF**: **R**adio **F**requency (signals).
18. **ROI**: **R**egion **o**f **I**nterest. A set of 2D pixels, upon which some analytic operation is performed, e.g. average intensity.
19. **SE**: **S**tructuring **E**lement. Structuring elements are used in mathematical morphology.
20. **Slabs**: An MR scan can consist of multiple slabs. A slab typically covers 5-6 cm of the object of interest.
21. **SNR**: **S**ignal **N**oise **R**atio.
22. **SPECT**: **S**ingle-**P**hoton **E**mission **C**omputed **T**omography. This technology is used in nuclear medicine where the patient is injected with a radiopharmeceutical (a harmless tracer chemical of some sort) which will emit gamma rays. The radioactivity is collected by an instrument called a gamma camera.
23. **structural MR**: T1 weighted MR.
24. **Vasuspasms**: Contraction of blood vessels.
25. **VOI**: **V**olume **O**f **I**nterest. A set of 3D voxels upon which some analytic operation is performed, e.g. average intensity.
26. **Voxel**: **Vo**lumetric pi**xel**.
27. **Xforms**: X Windows development kit.

References

[1] Jules Bloomenthal. An implicit surface polygonizer. In Paul S. Heckert, editor, *Graphics Gems IV*, pages 324–349. AP Professional, 1994.

[2] Bernardo Piquet Carnerro. Tetra-cubes. Technical report, Inst. de Matematica Pura e Aplicada, 1996.

[3] Scateni Contani and Scopigno. Discretized marching cubes. Unpublished.

[4] Hao-Ren De and Ruei-Chuan Chang. Sample buffer: A progressive refinement ray-casting algorithm for volume rendering. *Comput. & Graphics*, 17(3):277–283, 1993.

[5] J.A. den Boer and M. T. Vlaardingerbroek. *Magnetic Resonance Imaging*. Springer-Verlag, 1996.

[6] M.J Dürst. Letters: Additional reference to "marching cubes". *Comput Graph*, 22:72–73, 1988.

[7] Lorensen W. E. and Cline H.E. Marching cubes: A high resulution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–9, 1987.

[8] Mark Hall. Defining surfaces from sampled data. In Andrew S. Glassner, editor, *Graphics Gems I*. AP Professional, 1990.

[9] Steve Hill. Surface models and the resolution of n-dimensional cell ambiguity. In Alan W. Paeth, editor, *Graphics Gems V*, pages 98–106. AP Professional, 1995.

[10] Anders Krogh John Hertz and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Westley Publishing Company, 1991.

[11] Kyu Ho Park Kwang-Man Oh. A type-merging algorithm for extracting an isosurface from volumetric data. *The Visual Computer*, 12(8):406–419, 1996.

[12] Chua Beng Choon Louis Serra, Ng Hern and Timothy Poston. Interactive vessel tracing in volume data. Technical report, Centre for Information enhanced Medicine, Singapore, 1997.

[13] A. Witkin M. Kass and D. Terzopoulos. Snakes: Active contour models. *Proceedings, First Internation Conference on Computer Vision, London, England*, pages 259–268, 1987.

[14] Scateni Montani and Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer*, 10, 1994. to appear.

[15] William K. Pratt. *Digital Image Processing*. John Wiley & Sons, Inc., 1991.

[16] Shekhar R. Fayyad E. Yagel R. and Cornhill J.F. Octree-based decimation of marching cubes surface. Technical report, Biomedical Engineering Center, 1996.

[17] Mohan S. Kanhanhalli Renben Shu, Chen Zhou. Adaptive marching cubes. *The Visual Computer*, 11:202–217, 1995.

[18] Anders Bertil Rodell and Flemming Andersen. Automatisk registrering og sammenlægning af mr, ct og pet skanningsbilleder af hjernen ved symmetribetragtninger af en kraftfeltmodel. Master's thesis, Aarhus Universitet, Datalogisk Afdeling, 1995.

[19] John C. Russ. *The Image Processing Handbook*. CRC press inc, 1992.

[20] T. Solling and K. Vestergaard. Segmentering i magnetisk resonans billeder ved brug af den aktiv-kontur baserede segment algoritme. Master's thesis, Aarhus Universitet, Datalogisk Afdeling, 1995.

[21] V. Boyle R. Sonka M. Hlavac. *Image Processing, Analysis and Machine Vision*. Chapman And Hall, 1993.

[22] X. Zhuang T. Wang and X. Xing. Robust segmentaion of noisy images using a neural network model. *Image and Vision Computing*, 10:233–240, 1992.

[23] D.L. Toulson and J. F. Boyce. Segmentation of mr images using neural nets. *Image and Vision Computing*, 10:324–328, 1992.

[24] Elvins TT. A survey of algorithms for volume visualization. *Computer Graphics (SIGGRAPH 92 Proceedings*, 26(3):194–201, 1992.

[25] A. Watt and Watt M. *Advanced Animation and Rendering Techniques,*. Addison-Westley, 1992.

[26] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press Monograph, 1990.

[27] C. Wyvill, B. McPheeters and Wyvill G. Animating soft objects. *The Visual Computer*, 2(2):227–34, 1986.

List of Figures