Segmentation of 3D Medical MR Images

Master Thesis in Applied Mathemathics

Are Losnegård

Department of Mathematics

University of Bergen

15th September, 2006

Acknowledgements

I want to thank my supervisor Xue-Cheng Tai for his help in the work of my master thesis. He has helped me with ideas, images and to solve my mathematical problem. Many of the programs I have used are based on/are programs I have received from Changhui Yao and Oddvar Christiansen, and Oddvar also worked in parallel to extend algorithm 3 from 2D to 3D. They also helped me with good information on these. I have received the clinical images from Arvid Lundervold, who also performed the segmentation in *Freesurfer* and gave me many tips.

I also want to thank Johan Lie and Erlend Hodneland for their help and tips on mathematical and implementation problems, the masterstudents at MI, UiB, for discussions on my topic, and Jeff Pedersen for correcting my english.

Are Losnegård
Bergen, September 2006

Abstract

In this thesis the problem of segmenting 3D medical images will be treated. Image segmentation is the process of dividing an image into subsections based on intensity values. Two methods will be tested and compared with available software.

Medical imaging is a fast growing field and there are a number of possible applications. As Magnetic Resonance Imaging (MRI) is still developing, the quality is improving, but there are many factors making the segmentation process difficult. When segmenting medical images, one normally wants to identify the different tissues in the image.

We have worked with 3D MR images of the brain. These are large matrices and they need a lot of memory and a fast processor. Three algorithms will be presented. The first method based on algorithm 1 and 2 has a relatively slow convergence. This slowness is the motivation for the second method, where the problem is solved using operator splitting. As we will show, this results in shorter computational time.

We will give an introduction to MR imaging and to some of the important applications in image processing. One of the images we tested is a real MR image. Another one is a synthetic image, designed to be similar to a real one. As we will see, the real MR image is much more difficult to segment, mainly because of more noise, but also because there is no exact answer to the segmentation. This means that if this real MR image was segmented manually by medical doctors or radiologists, we would end up with different results. For the synthetic image, this is an easier process, because the different segments, or phases, are known. In our case, there are 3 interesting phases in the brain, the cerebrospinal fluid, white matter and grey matter. The background is sometimes the 4th phase, otherwise this will be the same phase as the cerebrospinal fluid.

The goal of this thesis has been to improve the algorithms, extend them from 2D to 3D, compare with existing methods, and also to see if they can be used for real applications, like the real MR image. There has been a focus on using few iterations to get a satisfying result.

The main idea behind both methods presented in this thesis is the minimization of a constrained functional of the form

$$F = \frac{1}{2}\int_{\Omega} |u - u_0|^2 \, \mathrm{d}x + \beta R,$$

where u_0 is the initial image, and u changes during iterations. So the first term is an *approximation* term and R is a *regularization* term. We will use a newly developed level-set-method to solve this problem.

Contents

Chapter 1

Introduction

The history of MRI started in 1946, when Felix Bloch and Edward Purcell independently discovered the magnetic resonance phenomenon, a spectroscopic technique used by scientists to obtain microscopic chemical and physical information about molecules. They were awarded the Nobel Prize for this in 1952. Between 1950 and 1970, NMR (nuclear magnetic resonance), was developed and used for chemical and physical molecular analysis. In the late 70's, the name changed from Nuclear magnetic resonance imaging (NMRI) to Magnetic resonance imaging (MRI) because of the bad associations with the word nuclear.

A big medical breakthrough was achieved in 1971, as Raymond Damadian showed that there are differences between tissues and tumors observable with MRI. In 1973, the x-ray based computerized tomography (CT) was also introduced, and hospitals became interested and invested a lot of money in medical imaging hardware. In 1980, imaging of the body was demonstrated.

In the beginning, MRI was a tomographic technique, that is it produced an image of the NMR signal in a thin slice through the human body. Today, it has developed into a volume imaging technique. In this thesis we will look at MRI's of the brain, both synthetic and real volume images.

In this thesis we will manly focus on images obtained with MRI. It is not crucial for the results of this thesis to understand this technique, but a brief discussion will still be given, because it is so widely used in hospitals all over the world.

1.1 Magnetic resonance imaging

As explained in the introduction, Magnetic resonance imaging (MRI) is used extensively to obtain images from inside the body [14]. The image in figure 1.1 is an axial synthetic brain MRI (one slice of the volume image).

The technique is based on nuclear magnetic resonance signals from hydrogen nuclei. These nuclei are abundant in water and fat, which we have a lot of in the body. A property of these nuclei is that they possess nuclear spin. Spin is a fundamental property of nature like electrical charge or mass. It is described by multiples of 1/2 and can be + or -. Protons, neutrons and electrons possess spin, and together they can form particles with spin (or no spin, if it is eliminated). When placed in an external magnetic field, particles with spin can absorb a photon. The frequency, ν, of this photon, depends on the gyromagnetic ratio[1], γ, of the particle and the strength, B_0, of the magnetic field.

[1]The ratio of the magnetic dipole moment to the mechanical angular momentum of a system.

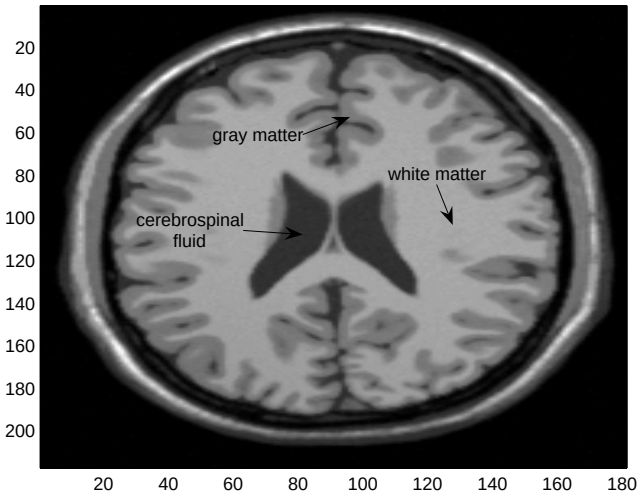

Figure 1.1: Brain MRI

$$\nu = \gamma \, B_0$$

The spin causes the particle to act as a small magnet with a north and south pole. In the external magnetic field, the spin of the particle aligns itself to this magnetic field, like a magnet. When the magnetic poles of the particle are aligned in the opposite direction of the external poles, it is called a low energy state. For the other case, it is called a high energy state. Transitions between these two states happen when the particle absorbs a photon.

A group of spins experiencing the same external magnetic field is called a spin packet. At any instant in time, the magnetic field due to the spin in each spin packet can be represented by a magnetization vector, and when one add these up, we get what is called the net magnetization. The net magnetization vector lies, at equilibrium, along the direction of the external magnetic field B_0.

By exposing the system to energy of a frequency equal to the energy difference between the two spin states, the net magnetization vector can be changed. It can also be changed by changing the external magnetic field, or by adding a different magnetic field to the existing one.

The Bloch equations can be used to describe the magnetization vector under any conditions:

$$\frac{dM_{x'}}{dt} = (\omega_0 - \omega)M_{y'} - \frac{M_{x'}}{T_2}$$

$$\frac{dM_{y'}}{dt} = -(\omega_0 - \omega)M_{x'} + 2\pi\gamma B_1 M_z - \frac{M_{y'}}{T_2}$$

$$\frac{dM_z}{dt} = -2\pi\gamma B_1 M_{y'} - \frac{(M_z - M_{z_0})}{T_1}$$

where $M_{x'}$, $M_{y'}$ and M_z are the x', y' ($'$ indicates rotating coordinate system) and z components of the equilibrium magnetization. T_1 and T_2 are time constants, B_1 is the strength of a magnetic field and ω and ω_0 are frequencies.

One way of getting the positions of the spin is by using a one-dimensional linear magnetic field, also called a field gradient. This gives us a variation in the external magnetic field. In figure 1.2, we imagine that there are three particles in the head that we are interested in. In reality the whole head would contain a signal. Because the frequency

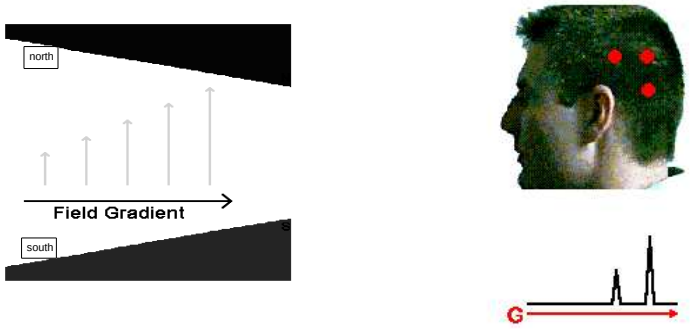

Figure 1.2: A magnetic field gradient is applied to find the position of the spins (marked as red dots). In the left image, the length of the vectors represents the magnitude of the magnetic field. The NMR spectrum under the right image is obtained either by sending a constant frequency into our magnetic field, and varying the magnetic field, or the opposite.

of the photons that the particles can absorb is dependent on the magnetic field, it will be dependent on the position of the spins. The field gradient is represented by G_x.

$$\nu = \gamma(B_0 + xG_x) = \nu_0 + \gamma x G_x \tag{1.2}$$

$$x = \frac{\nu - \nu_0}{\gamma G_x} \tag{1.3}$$

One way of performing MRI is called backprojection imaging. In this technique, the object is placed in a magnetic field. Then a one-dimensional field gradient is applied at several angles, and for each of them, the NMR spectrum is recorded.

Another way of performing a MRI is with the use of a phase encoding gradient. This gradient will be in B_0. The following technique is an example of this method.

1.1.1 Fourier transform tomographic imaging

A sequence of this type of imaging looks like this:

1. Slice selection gradient and slice selection radio frequency (RF) are applied. The slice selection gradient is always applied perpendicular to the slice plane. The RF pulse rotates only those spin packets within the object that satisfies the resonance condition. The location is given by

$$z = \frac{\Delta\nu}{\gamma G_s}$$

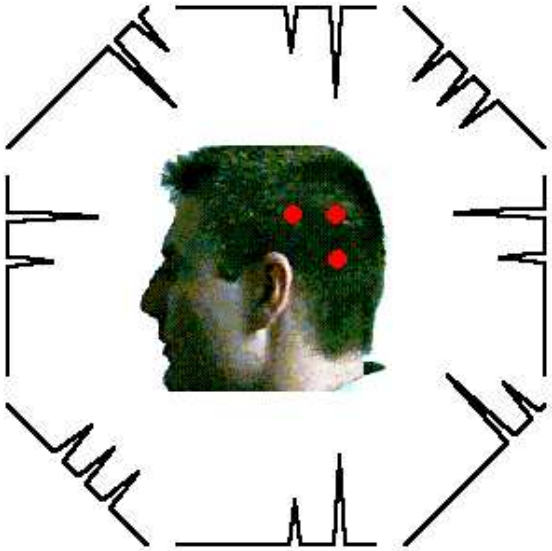

Figure 1.3: Backprojection imaging

Where $\Delta\ \nu = \nu$ - ν_0, G_s is the magnitude of the slice selection gradient, and γ is the gyromagnetic ratio. Now the spin packets rotate with the same frequency.

2. Slice selection gradient and RF pulse are turned off, and a phase encoding gradient is turned on along one of the sides of the image plane. This makes the spin packets in the slice rotate at different Larmor frequencies[2].

3. The phase encoding gradient is turned off. Now the spin packets precess at the same frequency, but they are in different phases. Also a frequency encoding gradient is turned on. This gradient is applied along the remaining edge of the image plane. If this is applied in the Y-direction, the spin packets will start to rotate with a frequency dependent on their position in the Y-direction.

   A simple Fourier transform is capable of determining the phase and frequency of a net magnetization vector somewhere in the slice. For each location in the slice there is a need for a different magnitude of the phase encoding gradient. For each of them a free induction decay (FID) is recorded. Free induction decay is a function of time, representing the current induced in coil of wire located around the x-axis (given that the transverse magnetization rotates about the z-axis). This will be a decaying sine function.

   The next step is to Fourier transform these FIDs to get a picture of the locations of the spins. This is done by first transforming them in one direction to get the frequency domain information, and then in the phase encoding direction to get information about the locations in the phase encoding gradient direction.

This technique, also known as gradient recalled echo, is very often used in volume imaging, as we will look at later in this work.

[2]The rotation frequency about the Z-axis of the net magnetization when placed in the XY-plane. This frequency is equal to the frequency of the photon which would cause a transition between the two energy levels of the spin.

1.1.2 Signal intensity

To be able to distinguish one tissue from another, the signal intensity has to be calculated. For the method presented, we get it from

$$S = \frac{k\rho(1 - e^{\frac{-TR}{T_1}})\sin\theta\ e^{\frac{-TE}{T_2*}}}{1 - \cos\theta\ e^{\frac{-TR}{T_1}}}$$

where S is the amplitude of the signal in the frequency domain spectrum, k is a constant, T_1, T_2 and ρ are specific to a tissue or pathology, θ is the rotation angle of the instrument in use, TR is the repetition time, TE is the echo time and T_2* is dependent on the homogeneity of the magnetic field and the molecular motions.

1.2 Image Processing

With the development of MRI and computers, and the increasing knowledge of diseases like cancer, there is also an increasing interest in image processing. This field has a wide range of applications, not only in medicine. We will now look at a few techniques and applications.

1.2.1 Segmentation

In this thesis we look at tools for segmenting images. What does this mean? If one assumes that an image Ω consists of subsections Ω_i, where in each subsection the image intensity is more or less constant, and we are interested in identifying these different subsections, image segmentation is a helpful tool. In digital images with little or no noise this is a small problem, but the challenge grows if noise is present.

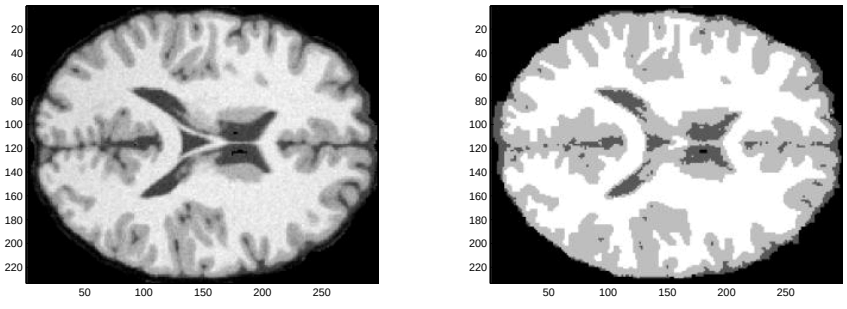

Figure 1.4: The image to the left is a brain with noise, and the one to the right is a segmented brain.

An example is a MR image of the brain, where one wants to identify the cerebrospinal fluid, the white matter and the grey matter for several applications. Several ways of segmenting such images are available, but in this thesis, we concentrate on minimization of functionals. Another application is the recognition of letters in e.g. a car plate.

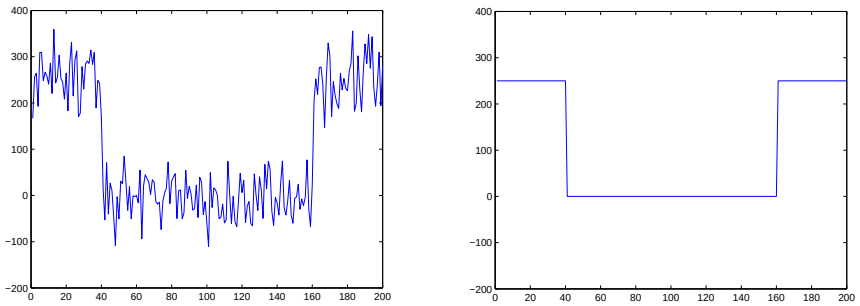

Figure 1.5: Noisy image (left) and image without noise (right).

1.2.2 Denoising

Noise occurs in MRI because of different reasons, the most important are

- inaccuracy in measurements and computer calculation errors
- the different apparatus in a MRI machine affecting each other
- for a normal 2D-image, normally 3-8 recording sequences are made, and the average of those are used. If the person is not lying still, the images will differ much.

The relation between the true image u, the noisy image u_0 and the noise ϵ is defined as

$$u_0 = u + \epsilon. \tag{1.4}$$

When acquiring a MR image, the information is first collected in the computer in a k-space matrix, where the data is in the time domain. Here one can assume that the noise is Gaussian distributed. To get the final image, Fast Fourier transformation takes the data into the space domain. Now the noise is not Gaussian distributed anymore, which means that the noise may vary from one tissue to another.

A normal assumption is that the noise is characterized by fast oscillations with respect to intensity values, and edges are slow oscillations or discontinuities in the image. So for methods dealing with noise, it is important to take this into consideration. In addition to smooth out fast oscillations and preserve edges, regions with smoothly varying intensity should also be preserved.

Different methods have been proposed for denoising. Among others are a modified diffusion equation in [21]. and the heat-equation has also shown promising results. Among the nonlinear problems we find the Total Variation (TV-filter).

1.2.3 Inpainting

Inpainting is the term used for techniques to repair images that have lost an area due to a crack, dust, something that has been removed (e.g. text) or other reasons. How should one fill this area(s)? As when done manually, there are several aspects to consider when doing this [3]:

- the global picture decides how to fill the gap, Ω,
- structure of the area surrounding the gap is continued into the gap,

- the different regions inside the gap, defined by contour lines, are filled with the colour matching those of the boundary, $\partial\Omega$, and

- texture should be added

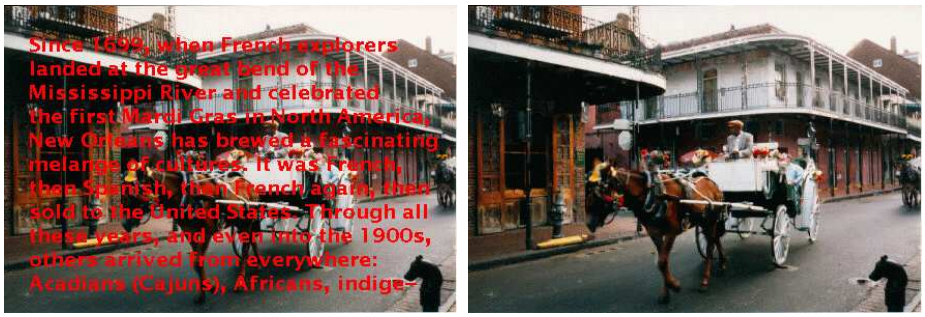

Figure 1.6: Example of application of inpainting.

1.2.4 Registration

Registration is a technique for aligning images in a more comparable way [24]. The reason that this is needed is that two images of a human brain are not equal even if they are acquired from the same person at two sessions with minimal time difference. This is because it is impossible to have the person's head positioned in exactly the same orientation at two different examinations when the head has been repositioned in the head coil. Comparing images of the same subject taken at different times in order to detect any changes is called *intra subject* analysis, and the opposite, *inter subject* (see figure 1.7) analysis is comparing images from different subjects.

The main goal of an image registration is to find a transformation T that will align the images in the expected way. The transformation we seek is actually a transformation of the coordinates of each sampled point to new coordinates, which will give us a new intensity value.

For inter subject studies, one often uses an *atlas* or *template*. This is a standard organ, e.g. a brain, which is constructed from a number of brains and given in a fixed coordinate system to contain information about the physical properties.

The transformation strategies can be divided into *intensity-based* (voxel based) and *model-based* algorithms. In intensity-based algorithms the intensity of the voxels are used to compare the two images we want to match, and in the model-based algorithms the idea is to first extract explicit geometric models representing separate anatomical structures, e.g. curves, surfaces or landmarks. These elements are then matched with their counterpart in the other image.

Examples of registration methods are *Rigid*, *affine* and *elastic* transformations. See [24] for further details.

The output of the registration is a correspondence function T that takes each point $\mathbf{x}$ in one image and finds the correspondent point $T(\mathbf{x})$ in another image. Sometimes the transformation is measured by a displacement vector $\mathbf{u}$ such that $T(\mathbf{x}) = \mathbf{x} + \mathbf{u}$. This displacement vector $\mathbf{u}$ expresses the 3D movement at point $\mathbf{x}$.

Figure 1.7: Example of application of registration. These images are scans of two different human brains. The images shown are the sagittal view of the central slice of each MR volume. They look quite different, also when we compare anatomical structures inside the brain.

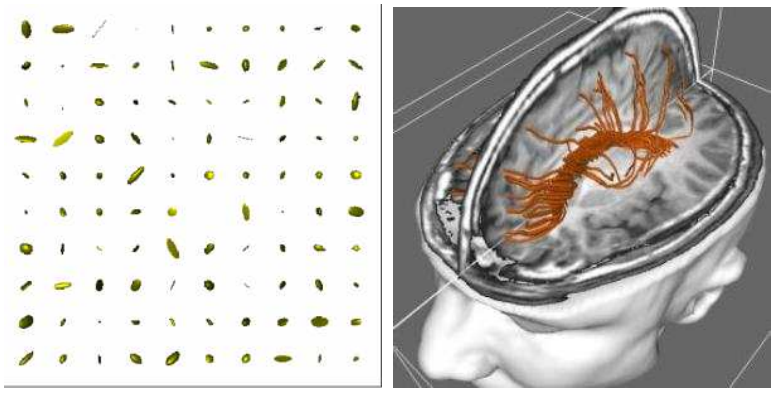

Figure 1.8: The diffusion ellipsoids (left) and an application of diffusion tensor imaging called fiber-tracking (right).

1.2.5 Diffusion tensor imaging

Water molecules diffuse differently in different tissues, and is also greater along fibres than across fibres (anisotrophy) [15] [12]. A MRI scanner can detect this anisotrophy, and from the data aquired one can, for each voxel, construct a 3x3 matrix called the diffusion tensor. This is a symmetric, positive definite matrix whose eigenvalues/eigenvectors describe the magnitude/direction of diffusion. By doing an eigenvalue decomposition, one can visualize the diffusion in each voxel by a diffusion ellipsoid. One application is called fibre-tracking, and here one follows the direction of the eigenvector corresponding to the largest eigenvalue.

Chapter 2

Mathematical tools

In this chapter, we will look at the mathematical tools used in the segmentation algorithms. From the idea of minimizing a constrained functional, we look at some functional analysis and Langrange's multipliers. As we look for saddlepoints of the Lagrangian functional, we will use the method of steepest descent and Newton's method. The latter is explained briefly in this chapter. The operator splitting scheme is explained in general here and more exact in the algorithm chapter. Also, how the Euler-Langrange equations are found from the classical problem of minimizing a functional in calculus of variations is explained. Important is the result that finding the minimum is the same as solving a certain PDE. At last, the idea of level sets is outlined. But first, we explain the Green identities, used in later calculations.

2.1 Green's identities

Let (A_1, A_2, A_3) be the three components of a vector field over some portion of the three-dimensional space in terms of a cartesian coordinate system (x_1, x_2, x_3). The divergence theorem (see [4]) requires that, for continuously differentiable functions $A_i(x_1, x_2, x_3)$, the integral of the divergence of A over any volume V be equal to the outward flux of A over the surface S of that volume, i.e.

$$\int_V \left(\frac{\partial A_1}{\partial x_1} + \frac{\partial A_2}{\partial x_2} + \frac{\partial A_3}{\partial x_3} \right) \mathrm{d}V = \int_S (A_1 n_1 + A_2 n_2 + A_3 n_3) \, \mathrm{d}S, \tag{2.1}$$

where (n_1, n_2, n_3) are the three components of the outward unit normal vector for the surface element ∂S. (We assume S to possess a tangent plane at each point.) For two dimensions, the volume and surface integrals are replaced by area and contour integrals, respectively.

Example 2.1 *Let ϕ be scalar and define A to be the gradient of ϕ, i.e. $A_i = \frac{\partial \phi}{\partial x_i}$, for $i = 1, 2, 3$. Then (2.1) becomes*

$$\int_V \Delta\phi \, \mathrm{d}V = \int_S \frac{\partial \phi}{\partial n} \, \mathrm{d}S.$$

Example 2.2 *Let $A_i = \phi \frac{\partial \psi}{\partial x_i}$, $i = 1, 2, 3$, where ϕ and ψ are scalars. (2.1) then becomes*

$$\int_V \phi \Delta\psi \, \mathrm{d}V + \int_V \left(\frac{\partial \phi}{\partial x_1}\frac{\partial \psi}{\partial x_1} + \frac{\partial \phi}{\partial x_2}\frac{\partial \psi}{\partial x_2} + \frac{\partial \phi}{\partial x_3}\frac{\partial \psi}{\partial x_3} \right) \mathrm{d}V = \int_S \phi \frac{\partial \psi}{\partial n} \, \mathrm{d}S. \tag{2.2}$$

If we rewrite the last equation with ϕ and ψ interchanged and subtract the two results, we get

$$\int_V (\phi\Delta\psi - \psi\Delta\phi)\,\mathrm{d}V = \int_S \left(\phi\frac{\partial\psi}{\partial n} - \psi\frac{\partial\phi}{\partial n}\right)\mathrm{d}S. \tag{2.3}$$

(2.2) and (2.3) are referred to as Green's first and second identity.

Example 2.3 *In (2.2), we set $\phi = u$ and $\nabla\psi = v$. We then get*

$$\int_V u\,div\,v\,\mathrm{d}V + \int_V \nabla u \cdot v\,\mathrm{d}V = \int_S u\,(v\cdot n)\,\mathrm{d}S. \tag{2.4}$$

2.2 Functional analysis

Definition 2.2.1 Linear vector space. *A linear vector space W over a scalar field $\mathscr{F}$ is a nonempty set W with a mapping: $(\mathbf{x_1}, \mathbf{x_2}) \to \mathbf{x_1} \oplus \mathbf{x_2}$ from $W \times W$ into W (addition), and a mapping: $(\alpha, \mathbf{x}) \to \alpha\mathbf{x}$ from $\mathscr{F} \times W$ into W (scalar multiplication). These mappings satisfy the conditions:*

1. *$x \oplus y = y \oplus x$, for all $x, y \in W$ (the commutative property)*
2. *$(x \oplus y) \oplus z = x \oplus (y \oplus z)$, for all $x, y, z \in W$ (the associative property)*
3. *For each $x \in W$, there exists a unique element 0 in W such that $x \oplus 0 = 0 \oplus x = x$ (the existence of the zero element 0)*
4. *For each $x \in W$, there is a unique element $-x \in W$ such that $x \oplus -x = 0$ (the existence of an inverse)*
5. *$\alpha(\beta x) = (\alpha\beta)x$ for all $x \in W$ and all $\alpha, \beta \in \mathscr{F}$*
6. *$(\alpha + \beta)x = \alpha x \oplus \beta x$ for all $x \in W$ and all $\alpha, \beta \in \mathscr{F}$*
7. *$\alpha(x \oplus y) = \alpha x \oplus \alpha y$ for all $x, y \in W$ and all $\alpha \in \mathscr{F}$*
8. *$1x = x$ for all $x \in W$, where 1 is the unit element of the scalar field $\mathscr{F}$.*

Definition 2.2.2 Normed linear space. *A normed linear space is a linear vector space X with a norm $\|\cdot\|_X$ on it, and is denoted by $(X, \|\cdot\|_X)$.*

Definition 2.2.3 Cauchy sequence. *A sequence $\{x_n\}$ of elements in a normed linear space $(X, \|\cdot\|_X)$ is Cauchy if*

$$\|x_n - x_m\|_X \to 0 \quad as\ m, n \to \infty \tag{2.5}$$

All convergent sequences are Cauchy, but all Cauchy sequences need not converge to a point in the space.

Definition 2.2.4 Banach space. *A Banach space is a complete*[1] *normed linear space.*

[1] To have completeness in a space, all Cauchy sequences must have limits in the space.

Definition 2.2.5 Hilbert space. *A Hilbert space is an inner product space which is complete as a normed space under the induced norm.*

Properties of inner products

1. $\langle x, y\rangle = 0$ *for all* $x \in \mathscr{H}$ *implies* $y = 0$
2. $|\langle x, y\rangle| = \|x\|\,\|y\|$ *(Schwarz inequality)*
3. $\|x+y\|^2 + \|x-y\|^2 = 2\|x\|^2 + 2\|y\|^2$ *(parallellogram law)*
4. *If the norm in any space satisfies the parallellogram law, then it is always possible to consider the space as an inner product space, with inner product*

$$\langle x, y\rangle = \frac{1}{4}\left(\|x+y\|^2 - \|x-y\|^2 + \|x+iy\|^2 - i\|x-iy\|^2\right). \tag{2.6}$$

Definition 2.2.6 $\mathbf{L^2}$. *The space of real-valued, square integrable functions on* Ω*, denoted by* $L^2(\Omega)$*, is a Hilbert space under the inner product* $\langle f, g\rangle_{L^2} = \int_\Omega f(x)g(x)\mathrm{d}x$*, and the induced norm* $\|f\|_{L^2} = \sqrt{\int_\Omega f(x)^2\mathrm{d}x}$*.*

Definition 2.2.7 Convex functional. $J : C \to \mathbb{R}$ *is a convex functional if*

$$J(\tau f_1 + (1-\tau)f_2) \le \tau J(f_1) + (1-\tau)J(f_2) \tag{2.7}$$

whenever $f_1, f_2 \in C$ *and* $0 < \tau < 1$*. J is strictly convex provided the inequality (2.7) is strict whenever* $f_1 \neq f_2$*.*

A set C is convex provided $\tau f_1 + (1-\tau)f_2 \in C$ *whenever* $f_1, f_2 \in C$ *and* $0 < \tau < 1$*. C is closed if for any convergent sequence* $\{f_n\} \subset C$*,* $\lim_{n\to\infty} f_n \in C$*.*

2.2.1 Derivatives

To solve the constrained minimization problem, we need to find the derivative of a functional, also called the Gâteaux derivative. In [16] we find the following

Definition 2.2.8 Gâteaux derivative. *Let $D(F)$ be the domain of F. Assume that for all $h \in D(F)$ we have*

$$\lim_{\epsilon\to 0}\frac{F(x_0+\epsilon h) - F(x_0)}{\epsilon} = DF(x_0, h) \quad , x_0 \in D(F), \tag{2.8}$$

where $DF(x_0, h)$ is a linear operator with respect to h. Then $DF(x_0, h)$ is called the Gâteaux differential of $F(x)$ at x_0, and the operator is called Gâteaux differentiable. Denoting $DF(x_0, h) = F'(x_0)h$, we get the Gâteaux derivative $F'(x_0)$.

The definition of Gâteaux derivative is clearly valid for functionals. Suppose $\phi(x)$ is a functional which is Gâteaux differentiable in a Hilbert space and that $D\phi(x, h)$ is bounded at $x = x_0$ as a linear functional in h. Then by Riesz representation theorem[2], it can

[2]Riesz representation theorem: Let $\phi(x)$ be a continuous linear functional given on a Hilbert space H. There is a unique element $f \in H$ such that

$$\phi(x) = (x, f) \quad \text{for every } x \in H.$$

Moreover, $\|\phi\| = \|f\|$.

be represented in the form of an inner product, denoting the representing element by $grad\,\phi(x_0)$, we get

$$D\phi(x_0, h) = (grad\,\phi(x_0), h). \tag{2.9}$$

By this, we have an operator $grad\,\phi(x_0)$ called the gradient of $\phi(x)$ at x_0.

Example 2.4 *$F(\phi) = \int_\Omega \lambda K(\phi)\,\mathrm{d}x$. The Gâteaux differential of F with respect to ϕ is*

$$DF(\phi, \psi) = \lambda \int_\Omega \lim_{\epsilon \to 0} \frac{K(\phi + \epsilon\psi) - K(\phi)}{\epsilon\psi} \psi\,\mathrm{d}x = \int_\Omega \lambda K'(\phi)\psi\,\mathrm{d}x,$$

where $\lambda K'(\phi)$ is the Gâteaux derivative with respect to the L^2 inner product.

Example 2.5 *$F(\phi) = \frac{r}{2}\int_\Omega |K(\phi)|^2 \mathrm{d}x$. The Gâteaux differential of F with respect to ϕ is*

$$\begin{aligned} DF(\phi, \psi) &= \frac{r}{2} \int_\Omega \lim_{\epsilon \to 0} \frac{|K(\phi + \epsilon\psi)|^2 - |K(\phi)|^2}{\epsilon}\,\mathrm{d}x \\ &= \frac{r}{2} \int_\Omega \lim_{\epsilon \to 0} \frac{(|K(\phi + \epsilon\psi)| - |K(\phi)|)\,(|K(\phi + \epsilon\psi)| + |K(\phi)|)}{\epsilon\psi} \psi\,\mathrm{d}x \\ &= \int_\Omega rK'(\phi)K(\phi)\psi\,\mathrm{d}x, \end{aligned}$$

where $rK'(\phi)K(\phi)$ is the Gâteaux derivative.

Example 2.6 *$F(\phi) = \frac{1}{2}\int_\Omega |u - u_0|^2 \mathrm{d}x$, where $u = u(c, \phi)$. The Gâteaux differential of F with respect to ϕ is*

$$\begin{aligned} DF(\phi, \psi) &= \frac{1}{2} \int_\Omega \lim_{\epsilon \to 0} \frac{|u(c, \phi + \epsilon\psi) - u_0|^2 - |u(c, \phi) - u_0|^2}{\epsilon}\mathrm{d}x \\ &= \frac{1}{2} \int_\Omega \lim_{\epsilon \to 0} \frac{(u(c, \phi + \epsilon\psi) - u_0 - u(c, \phi) + u_0)\,(u(c, \phi + \epsilon\psi) - u_0 + u(c, \phi) - u_0)^2 \cdot \psi}{\epsilon\psi} \\ &= \frac{1}{2} \int_\Omega \lim_{\epsilon \to 0} \frac{u(c, \phi + \epsilon\psi) - u(c, \phi)}{\epsilon\psi} \cdot (u(c, \phi + \epsilon\psi) + u(c, \phi) - 2u_0)\,\psi\,\mathrm{d}x \\ &= \int_\Omega \frac{\partial u}{\partial \phi}(u - u_0)\psi\,\mathrm{d}x, \end{aligned}$$

where $\frac{\partial u}{\partial \phi}(u - u_0)$ is the Gâteaux derivative.

In order to make sure that differentiable functionals are continuous we now introduce a stronger concept of derivative: Fréchet derivative [10].

Definition 2.2.9 Fréchet derivative. *Consider $u : X \to Y$, where both X and Y are normed linear spaces. Given $x \in X$, if a linear operator $Du(x, h)$ exists which is continuous, such that*

$$\lim_{\|h\| \to 0} \frac{\|u(x + h) - u(x) - Du(x, h)\|_Y}{\|h\|_X} = 0, \tag{2.10}$$

then u is said to be Fréchet differentiable at x, and $Du(x, h)$ is said to be the Fréchet differential of u at x with increment h. So $Du(x, h) \in \mathscr{L}(X, Y)$[3], and it is easy to see that if the Fréchet differential exists, then so does the Gâteaux differential, and the two are equal. Moreover, if u has a Fréchet differential at x, then u is continuous at x.

[3]The normed linear space $\mathscr{L}(X, Y)$ is the space of bounded linear transformations $T : X \to Y$ with the norm $\|T\| = \sup_{\substack{x \in X \\ x \neq 0}} \left\{ \frac{\|Tx\|_Y}{\|x\|_X} \right\}$

Finally, we include a theorem that is important for our purpose - finding extrema of a functional.

Theorem 2.2.10 *If the functional $f : X \to \mathbb{R}$ has a minimum or a maximum at $x \in X$, and $Df(x,h)$ exists, then $Df(x,h) = 0$.*

2.2.2 Dual representation

Let φ be a convex functional defined on a convex set $C \subset \mathbb{R}^d$. For our purposes, $d = 1, 2$ or 3, $C = \mathbb{R}^d$, and (see [30])

$$\varphi(\mathbf{x}) = \frac{1}{2}\psi(|\mathbf{x}|^2) \tag{2.11}$$

with $|\mathbf{x}|^2 = \mathbf{x}^T\mathbf{x} = \sum_{i=1}^d x_i^2$.

Definition 2.2.11 *The conjugate set C^* is defined by*

$$C^* = \left\{ \mathbf{y} \in \mathbb{R}^d \,|\, \sup_{\mathbf{x}\in C}[\mathbf{x}^T\mathbf{y} - \varphi(\mathbf{x})] < \infty \right\}, \tag{2.12}$$

and the corresponding conjugate functional to φ is

$$\varphi^*(\mathbf{y}) = \sup_{\mathbf{x}\in C}\{\mathbf{x}^T\mathbf{y} - \varphi(\mathbf{x})\}. \tag{2.13}$$

This functional is also known as the Fenchel transform of φ and has the conjugate set C^* as its domain.

One can show that the conjugate set C^* and the conjugate functional φ^* are, respectively, a convex set and a convex functional. The corresponding second conjugates are defined in the obvious manner:

$$\varphi^{**} = (\varphi^*)^* \quad \textit{and} \quad C^{**} = (C^*)^*. \tag{2.14}$$

In a finite dimensional Hilbert space setting, one can show that $\varphi^{**} = \varphi$ and $C^{**} = C$. Consequently, from (2.13) we obtain the dual representation

$$\varphi(\mathbf{x}) = \sup_{\mathbf{y}\in C^*}\{\mathbf{x}^T\mathbf{y} - \varphi^*(\mathbf{y})\}. \tag{2.15}$$

Now, let $\varphi(\mathbf{x}) = |\mathbf{x}|$. By the Cauchy-Schwarz inequality,

$$\mathbf{x}^T\mathbf{y} - |\mathbf{x}| \le (|\mathbf{y}| - 1)|\mathbf{x}|, \tag{2.16}$$

with equality if and only if $\mathbf{y} = c\mathbf{x}$ for some $c \in \mathbb{R}$. If $|\mathbf{y}| > 1$, one can make (2.16) arbitrarily large by taking $\mathbf{y} = c\mathbf{x}$ and letting c increase. If $|\mathbf{y}| \le 1$, then (2.16) is zero or negative, and its maximum value of zero is attained for $\mathbf{x} = 0$. Hence

$$\sup_{\mathbf{x}\in\mathbb{R}^d}\{\mathbf{x}^T\mathbf{y} - |\mathbf{x}|\} = \begin{cases} 0 & \text{if } |\mathbf{y}| \le 1, \\ +\infty & \text{if } |\mathbf{y}| > 1. \end{cases}$$

From definition 2.2.11, we see that the conjugate set must be the unit ball,

$$C^* = \{y \in \mathbb{R}^d \,|\, |\mathbf{y}| \le 1\}, \tag{2.17}$$

and thus the conjugate functional $\varphi^*(\mathbf{y}) = 0$ for each $\mathbf{y} \in C^*$. Putting this into the dual representation (2.15), we get

$$|\mathbf{x}| = \sup_{|\mathbf{y}|\le 1} \mathbf{x}^T\mathbf{y}. \tag{2.18}$$

We will use this result in the next section, finding an expression for the total variation norm.

2.2.3 TV Norm

In [30] we find the following definition of the total variation of a function f defined on the interval [0,1]:

$$TV(f) \stackrel{def}{=} \sup \sum_i |f(x_i) - f(x_{i-1})|, \tag{2.19}$$

where the supremum is taken over all partitions $0 = x_0 < x_1 < \ldots < x_n = 1$ of the interval. If f is piecewise constant with a finite number of jump discontinuities, then TV(f) gives the sum of the magnitudes of the jumps.

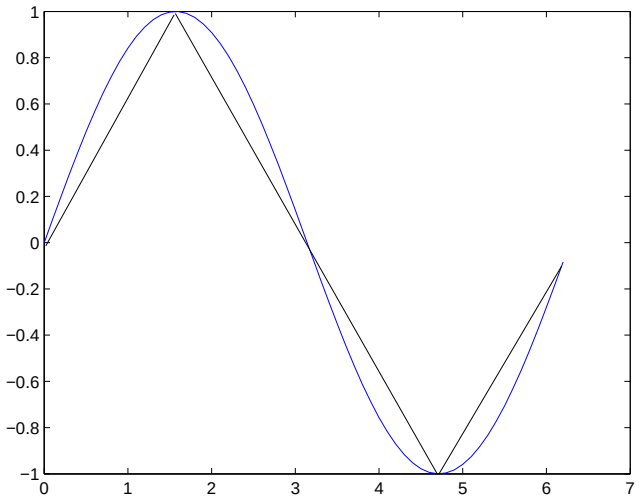

Figure 2.1: Example of two functions with the same TV norm. The blue line represents $f(x) = \sin x$, and the black is a linear and piecewise differentiable function with the same local extrema as f.

In case the function f is smooth, the right-hand side of (2.19) can be multiplied by $\Delta x_i = x_i - x_{i-1}$ and take the limit as the $\Delta x_i \to 0$ to obtain the representation

$$TV(f) = \int_0^1 \left| \frac{df}{dx} \right| \mathrm{d}x, \tag{2.20}$$

and the generalization to two space dimensions is

$$TV(f) = \int_0^1 \int_0^1 |\nabla f| \, \mathrm{d}x \, \mathrm{d}y. \tag{2.21}$$

Definition 2.2.12 *For a function $f \in L^1(\Omega)$ the total variation is defined by*

$$TV(f) = \sup_{v \in V} \int_\Omega f \, div \, v \, \mathrm{d}x, \tag{2.22}$$

where the space of test functions

$$V = \{v \in C_0^1(\Omega; \mathbb{R}^d) \mid |v(x)| \leq 1 \text{ for all } x \in \Omega\}. \tag{2.23}$$

We use the dual representation (2.18) for the Euclidean norm and, like in (2.4), formally apply integration by parts to get

$$\int_\Omega |\nabla f| \, \mathrm{d}x = \int_\Omega \sup_{|v|\leq 1} \nabla f \cdot v \, \mathrm{d}x \tag{2.24}$$

$$= \sup_{|v|\leq 1} \left[\int_{\partial\Omega} f \ (v \cdot n \,) \, \mathrm{d}S - \int_\Omega f \, \mathrm{div}\, v \, \mathrm{d}x \right], \tag{2.25}$$

where $\partial\Omega$ denotes the boundary of Ω and n denotes the outward unit normal to $\partial\Omega$. If v is compactly supported in Ω, then the boundary integral term vanishes. Note that $|v| \leq 1$ if and only if $|-v| \leq 1$, so we can drop the minus sign to obtain (2.22).

2.3 Euler-Lagrange equations

In the traditional setting of minimizing a functional, some equations called the Euler-Lagrange equations can be solved to find this minimum. Here is the background for this.

The following lemma is called the fundamental lemma of calculus of variations [10].

Lemma 2.3.1 *If $v \in C[a,b]$ and $\int_a^b v(t)h(t)\mathrm{d}t = 0$ for all $h \in C^1[a,b]$ with $h(a) = h(b) = 0$, then $v(t) \equiv 0$.*

Proof: Suppose by contradiction that $v(t) > 0$ for some $t_0 : a < t_0 < b$. Then there exists an $\epsilon > 0$, such that $v(t) > 0$ on $[t_0 - \epsilon, t_0 + \epsilon]$. Let us define

$$h(t) = \begin{cases} (t - t_0 + \epsilon)^2 (t_0 - \epsilon - t)^2 & \text{if } t \in [t_0 - \epsilon, t_0 + \epsilon] \\ 0 & \text{else.} \end{cases}$$

Then $h \in C^1[a,b]$ and $h(a) = h(b) = 0$, but

$$\int_a^b v(t)h(t)\mathrm{d}t > 0,$$

which is a contradiction.

Now, let $u : \mathbb{R}^3 \to \mathbb{R}$ have continuous second partial derivatives with respect to all variables, and consider the functional $f : C^1[a,b] \to \mathbb{R}$, defined by

$$f(x) = \int_a^b u\left(x(t), \dot{x}(t), t\right) \mathrm{d}t \tag{2.26}$$

The classical calculus of variation problem is to minimize this functional, where we minimize over the class of curves x belonging to Γ

$$\Gamma = \left\{ \begin{array}{l} x : [a,b] \to \mathbb{R} \text{ such that } x \in C^1[a,b] \\ \text{and } x(a) = x_1,\ x(b) = x_2. \end{array} \right\} \tag{2.27}$$

If we minimize in this class then we must consider variations or pertubations of x of the form $x + h$, where $x + h \in \Gamma$, that is, we require $h \in C^1[a,b]$ and $h(a) = h(b) = 0$.

To find the minimum, we need

$$f(x+h) - f(x) = \int_a^b \Big[u(x(t) + h(t), \dot{x}(t) + \dot{h}(t), t) - u(x(t), \dot{x}(t), t) \Big] \, \mathrm{d}t,$$

which by Taylor-expansion can be expressed as

$$f(x+h) - f(x) = \int_a^b \left[\frac{\partial u}{\partial x}\left(x(t), \dot{x}(t), t\right) h(t) + \frac{\partial u}{\partial \dot{x}}\left(x(t), \dot{x}(t), t\right) \dot{h}(t) \right] \mathrm{d}t + r(h, \dot{h}),$$

where $r(h, \dot{h}) = o\left(\|h\|_{C[a,b]}\right)$, that is

$$\frac{|r(h, \dot{h})|}{\|h\|_{C^1[a,b]}} \to 0 \quad \text{as } \|h\|_{C[a,b]} \to 0.$$

Hence, from (2.10),

$$Df(x,h) = \int_a^b \left[\frac{\partial u}{\partial x}(x(t), \dot{x}(t), t) h(t) + \frac{\partial u}{\partial \dot{x}}(x(t), \dot{x}(t), t) \dot{h}(t) \right] \mathrm{d}t.$$

Integration by parts yields

$$Df(x,h) = \int_a^b \left[\frac{\partial u}{\partial x} - \frac{d}{dt}\left(\frac{\partial u}{\partial \dot{x}}\right) \right] h(t)\, \mathrm{d}t + \frac{\partial u}{\partial \dot{x}} h(t) \Big|_a^b. \tag{2.28}$$

In case $h(a) = h(b) = 0$, the last term vanishes, and from theorem (2.2.10) we have that, if f has an extrema at x, the differential in (2.28) must equal 0. Moreover, from lemma (2.3.1) we have that

$$\frac{\partial u}{\partial x} - \frac{d}{dt}\left(\frac{\partial u}{\partial \dot{x}}\right) = 0. \tag{2.29}$$

(2.29) is called the Euler-Lagrange equation for this problem. In other words, the minimizer of the functional in (2.26) is also the solution of the PDE in (2.29).

This is the classical case. In our case we apply the steepest descent method to our minimization problem, i.e. for a functional $F(c, \phi)$ we want to solve

$$\frac{\partial F}{\partial c} = 0 \quad \text{and} \quad \frac{\partial F}{\partial \phi} = 0. \tag{2.30}$$

As we will see later, the minimization with respect to c can be solved exactly, and for the minimization with respect to ϕ we define

$$\phi_t = -\frac{\partial F}{\partial \phi}, \tag{2.31}$$

and look for a steady-state, i.e. $\phi_t = 0$.

2.4 Newton's method

To find a root of the equation $f(x) = 0$, e.g a zero of a function f differentiable near the root, the Newton's method can be used (see [1]). We will first explain the basic idea before we go on to see how this can be used to solve our minimization problem.

The tangent line to $y = f(x)$ at $x = x_0$ has equation

$$y = f(x_0) + f'(x_0)(x - x_0)$$

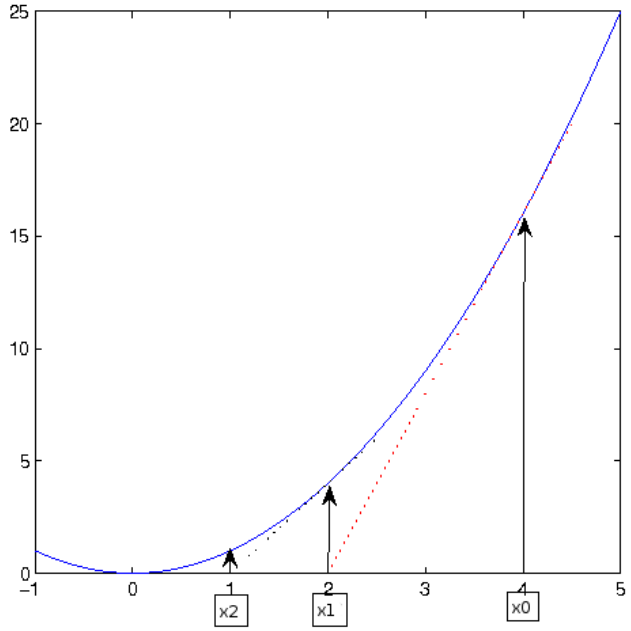

Figure 2.2: Finding the root of $f(x) = x^2$ with Newton's method, with $x_0 = 4$. See how x_1 and x_2 are found by following the tangent line of the previous point until $y = 0$.

From figure 2.2 we see that the point $(x_1, 0)$ lies on this line. This gives us

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

This produces an iteration, where

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

If the limit $\lim_{n\to\infty} x_n = r$ exists, and f/f' is continuous near r, then r must be a root of f, because

$$r = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} x_n - \lim_{n\to\infty} \frac{f(r)}{f'(r)},$$

from which it follows that $f(r) = 0$. Furthermore, defining $g(x) = f'(x)$ and calculating

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)},$$

will then give us an extremum of f if it exists, and $f'(x)/f''(x)$ is continuous near the extremum.

In this thesis we are treating functionals, so consider (see [30]) the quadratic functional $J : \mathbb{R}^n \to \mathbb{R}$, such that

$$J(\mathbf{f}) = c + \langle \mathbf{b}, \mathbf{f} \rangle + \frac{1}{2}\langle A\mathbf{f}, \mathbf{f} \rangle, \tag{2.32}$$

where $c \in \mathbb{R}$, $\mathbf{b}$, $\mathbf{f} \in \mathbb{R}^n$ and A is a symmetric matrix. Note that A is the Hessian of $J(\mathbf{f})$. We can approximate a function $f(x + k)$ using a Taylor series:

$$f(x + k) \approx f(x) + f'(x)k + \frac{1}{2}f''(x)k^2. \tag{2.33}$$

Assuming x is known, we can define the right hand side of (2.33) to be $g(k)$. Similarly, let $\mathbf{s} \in \mathbb{R}^n$ and consider the quadratic approximation to the functional $J(\mathbf{f}_n + \mathbf{s})$,

$$Q_n(\mathbf{s}) = J(\mathbf{f}_n) + \langle \nabla J(\mathbf{f}_n), \mathbf{s} \rangle + \frac{1}{2} \langle \text{Hess } J(\mathbf{f}_n)\mathbf{s}, \mathbf{s} \rangle,$$

where $\mathbf{f}_n$ is an estimate for the minimizer of J and Hess denotes the Hessian matrix. If Hess $J(\mathbf{f}_n)$ is positive definite, then $Q_n(\mathbf{s})$ has a unique minimizer which satisfies

$$\nabla J(\mathbf{f}_n) + \text{ Hess } J(\mathbf{f}_n)\mathbf{s} = 0,$$

which can be verified by calculating the Gâteaux derivative of $Q_n(\mathbf{s})$ with respect to $\mathbf{s}$. If we take $\mathbf{f}_n + \mathbf{s}$ as the new estimate for the minimizer of J, we obtain the Newton iteration

$$\mathbf{f}_{n+1} = \mathbf{f}_n - [\text{Hess } J(\mathbf{f}_n)]^{-1} \nabla J(\mathbf{f}_n), \quad n = 0, 1, \ldots \tag{2.34}$$

Newton's method in this form has some negative aspects, e.g. convergence is guaranteed only for $\mathbf{f}_0$ sufficiently close to a local minimizer $\mathbf{f}_*$, and it might be quite expensive to compute Hessians H = Hess $J(\mathbf{f}_n)$ and solve linear systems $H\mathbf{s} = -\nabla J(\mathbf{f}_n)$ to obtain the Newton steps $\mathbf{s}$.

2.5 Lagrange multipliers

The goal in this thesis is not only to find the minima of some functionals, but also to make sure that a certain constraint is fulfilled. The reason for the constraint will become clear in section 2.6. For the minimization (or finding extrema of) functions as well as for functionals, there is a very helpful tool in Lagrangian multipliers.

So, to find the solution of a constraint minimization problem, we will use Lagrange multipliers [2]. If a scalar field $f(x_1, \ldots, x_n)$ has a relative extremum when it is subject to m constraints, say

$$g_1(x_1, \ldots, x_n) = 0, \quad \ldots, \quad g_m(x_1, \ldots, x_n) = 0, \tag{2.35}$$

where $m < n$, then there exists m scalars $\lambda_1, \ldots, \lambda_m$ such that

$$\nabla f = \lambda_1 \nabla g_1 + \ldots + \lambda_m \nabla g_m \tag{2.36}$$

at each extremum point. The scalars $\lambda_1, \ldots, \lambda_m$ are called *Lagrange's multipliers*.

We will explain this with an easy example. The solution to the problem in figure 2.3

$$\max_{g(x,y)=0} f(x, y) \tag{2.37}$$

is found in the point where the level curves of f and g meet. In this point ∇g and ∇f must be perpendicular to the level curves. If not, one could decompose one of the gradients in a component along the level curve and a component perpendicular to this. The gradient does not have a component along the level curve, because then the function value would not be constant along the level curve. So ∇g and ∇f are parallel. They can, however, point in different directions and have different lengths. Then we must have

$$\nabla f = \lambda_1 \nabla g \tag{2.38}$$

for some scalar λ_1.

In practise, the extremum is found by solving the system of $n + m$ equations obtained by taking the m constraint equations found in (2.35) along with the n scalar equations determined by the vector relation (2.36).

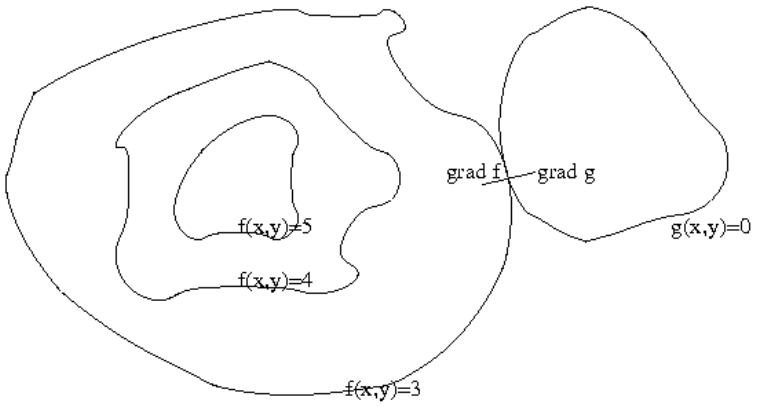

Figure 2.3: For a function f and another function g, some level curves are displayed. To find the maximum of the function f where $g = 0$, we see that the maximum of f on the zero level curve of g must satisfy $\nabla f = \lambda \nabla g$.

2.6 Level set methods

2.6.1 The original idea

Osher and Sethian proposed a level set method in [25]. This is a very helpful tool for separating a domain Ω into subdomains, e.g. according to intensity values in an image. The basic idea, found in [17], is to define a function $\phi(x)$, whose zero level set represents an interface Γ. This implies that

$$\begin{aligned} \phi(x) &> 0, \quad \text{x inside } \Gamma, \\ \phi(x) &= 0, \quad \text{x on } \Gamma, \\ \phi(x) &< 0, \quad \text{x outside } \Gamma. \end{aligned}$$

Often ϕ is required to be the signed distance function to the interface

$$\phi(x) = d(\Gamma, x), \quad \text{x inside } \Gamma, \tag{2.39}$$
$$\phi(x) = 0, \quad \text{x on } \Gamma, \tag{2.40}$$
$$\phi(x) = -d(\Gamma, x), \quad \text{x outside } \Gamma, \tag{2.41}$$

where $d(\Gamma, x)$ is the Euclidean distance between x and Γ. By doing this, we ensure that there is a one-to-one correspondence between the the function ϕ and interface Γ, and the level set function ϕ satisfies the Eikonal equation

$$|\nabla \phi| = 1. \tag{2.42}$$

Finding the solution to (2.42) can be done by solving the following initial value problem to steady-state

$$\phi_t + sgn(\tilde{\phi})(|\nabla \phi| - 1) = 0, \tag{2.43}$$
$$\phi(x, 0) = \tilde{\phi}(x), \tag{2.44}$$

where $\tilde{\phi}$ may not be a distance function. However, when the steady state of (2.43) is reached, ϕ is a distance function having the same zero level curve as $\tilde{\phi}$. This is known as the reinitialization procedure.

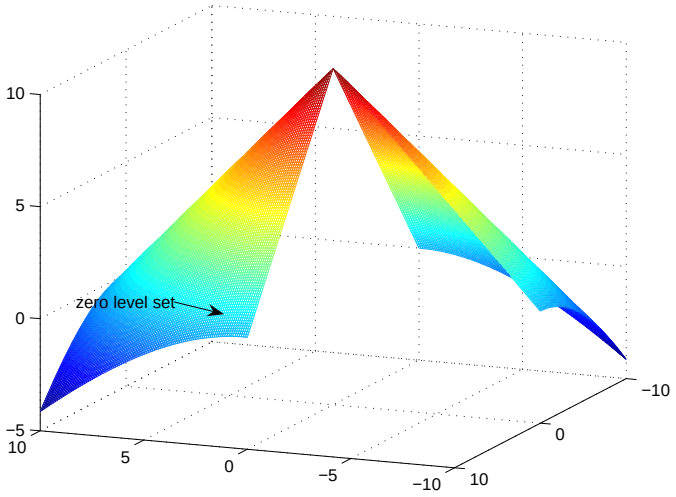

Figure 2.4: The distance function ϕ, where the distance is with respect to Γ, a circle in the xy-plane with center in origo and radius = 10.

The interface Γ is implicitly moved according to the nonlinear PDE

$$\phi_t + v(\phi) \cdot \nabla\phi = 0, \tag{2.45}$$

where $v(\phi)$ is a given velocity field depending on e.g. geometry, position, time and internal or external physics. (2.45) is often modified to just include the velocity normal v_N to the interface Γ

$$\phi_t + v_N(\phi)\nabla\phi = 0. \tag{2.46}$$

For cases with more than two subdomains, like in [18], we can introduce more curves and level set functions. For 4 subdomains, we introduce Γ_1 and Γ_2 associated with the level set functions ϕ_i, i=1,2, in the following way

$$\Omega_1 = \{x \in \Omega,\, \phi_1 > 0,\, \phi_2 > 0\}, \quad \Omega_2 = \{x \in \Omega,\, \phi_1 > 0,\, \phi_2 < 0\}, \tag{2.47}$$
$$\Omega_3 = \{x \in \Omega,\, \phi_1 < 0,\, \phi_2 > 0\}, \quad \Omega_4 = \{x \in \Omega,\, \phi_1 < 0,\, \phi_2 < 0\}. \tag{2.48}$$

This can be extended to N level set functions, giving 2^N regions that are disjunct and whose union is Ω.

2.6.2 Piecewise constant level set methods

We can see that N level set functions give the possibility of 2^N disjunct regions with no overlap. In [18] they wanted to find N regions, $\{\Omega_i\}_{i=1}^N$, which form a partition of Ω. Their idea is to have the possibility for some regions to be empty, so that they can overestimate the total number of phases, and superfluous phases will disappear at convergence. To identify these regions they proposed to find just one function, a piecewise constant function, taking the values

$$\phi = i \text{ in } \Omega_i, \quad i = 1, 2, \ldots, N. \tag{2.49}$$

The discontinuities of ϕ give the curves that separate the regions.

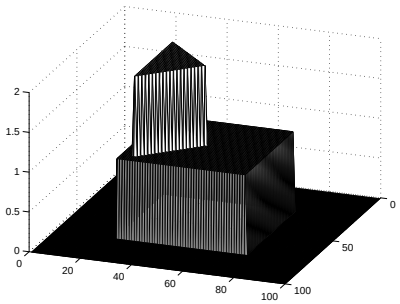

Figure 2.5: Example of an image represented by an piecewise constant function.

In an image, we assume that the intensity values are varying slowly inside each phase, and have large jumps between the phases. In that sense, letting a piecewise constant function represent an image should be reasonable.

To find a piecewise constant function, the values of the constants and the location of the discontinuities have to be found. Mumford and Shah [23] proposed to approximate a function u_0 by a piecewise constant function u:

$$\inf_{u,\Gamma}\left\{F^{MS}(u,\Gamma)=\sum_i\int_{\Omega_i}|c_i-u_0|^2\mathrm{d}x+\nu|\Gamma|\right\}, \tag{2.50}$$

where $\Omega=\cup_i\Omega_i\cup\Gamma$, i.e. they tried to find a decomposition Ω_i of Ω, where $u=c_i$ (constant) inside each connected component Ω_i. The length of the curve Γ is controlled by a positive parameter ν. For a fixed Γ, we see that (2.50) is minimized when $c_i=mean(u_0)$ in Ω_i. One challenge when solving (2.50) is to find a unique representation of the parametrized curve Γ. Chan and Vese [7] reformulated and solved (2.50) using a level set approach

$$\inf_{\mathbf{c},\phi}\left\{F(c_1,c_2,\phi)=\int_\Omega|c_1-u_0|^2H(\phi)\mathrm{d}x+\int_\Omega|c_2-u_0|^2(1-H(\phi))\mathrm{d}x+\nu\int_\Omega|\nabla H(\phi)|\mathrm{d}x\right\}, \tag{2.51}$$

where the Heaviside function $H(\phi)$ equals 1 if $\phi\geq 0$ and equals 0 if $\phi<0$. The length of the interface is measured by the regularization term $\int_\Omega|\nabla H(\phi)|\mathrm{d}x$ and favor smooth curves. By minimizing $F(c_1,c_2,\phi)$ with respect to c_1 and c_2 we obtain: $c_1=mean(u_0)$ if $\phi>0$ and $c_2=mean(u_0)$ if $\phi<0$.

The active contour model without edges in (2.51) can only divide Ω into two distinct regions. A generalization of this 2-phase model to a multiphase level set formulation was presented by Chan and Vese [6] [29] made it possible to solve the problem (2.50) for an arbitrary number of partitions. The energy functional for a 4-phase partition, involving 2 level set functions is then

$$\begin{aligned}F(\mathbf{c},\phi_1,\phi_2)=&\int_\Omega|c_1-u_0|^2H(\phi_1)H(\phi_2)\mathrm{d}x+\int_\Omega|c_2-u_0|^2H(\phi_1)(1-H(\phi_2))\mathrm{d}x\\&+\int_\Omega|c_3-u_0|^2(1-H(\phi_1))H(\phi_2)\mathrm{d}x+\int_\Omega|c_4-u_0|^2(1-H(\phi_1))(1-H(\phi_2))\mathrm{d}x\\&+\nu\int_\Omega|\nabla H(\phi_1)|\mathrm{d}x+\nu\int_\Omega|\nabla H(\phi_2)|\mathrm{d}x.\end{aligned} \tag{2.52}$$

Once the constant values $\{c_i\}_{i=1}^4$ and the level set functions ϕ_1 and ϕ_2 are identified, the

piecewise constant function u can be written as

$$\begin{aligned} u =& c_1 H(\phi_1)H(\phi_2) + c_2 H(\phi_1)(1 - H(\phi_2)) + c3(1 - H(\phi_1))H(\phi_2) \\ &+ c_4(1 - H(\phi_1))(1 - H(\phi_2)). \end{aligned} \tag{2.53}$$

In [5] [13] an alternative form of (2.52) was suggested. They tried to minimize the following energy functional directly with u given in (2.53)

$$F(\mathbf{c}, \phi_1, \phi_2) = \int_\Omega |u - u_0|^2 \mathrm{d}x + \nu \int_\Omega |\nabla \phi_1| \mathrm{d}x + \nu \int_\Omega |\nabla \phi_2| \mathrm{d}x. \tag{2.54}$$

Omitting the regularization term, functional (2.52) equals functional (2.54) except from one scalar. However, the gradients are not equal, producing different numerical schemes.

As mentioned, N level set functions and 2^N constants give the possibility of 2^N disjunct regions with no overlap, whereas the approach in (2.49) just requires one function to identify all the phases.

As we will see later, the basis for finding such a function will be to approximate the image in some sense, i.e. minimization of a functional, and imposing $\phi = i$, $i = 1, ..., N$ as a constraint. Also, some kind of control on the length of the curves (area in 3D) separating the regions will be added. This will make sure that we don't get an oscillating result.

2.7 Operator splitting

To find the ϕ from (2.49), we will use the steepest descent method (except in algorithm 2, where we use Newton's method). During these iterations, rather heavy computations have to be made, especially for the 3 dimensional images, e.g. curvature. In [8] they proposed to use operator splitting to accelerate the computations. We will try to explain the operator splitting scheme in a general setting here, and show how to incorporate this into one of our methods in section 3.2.

Given a function space V and an operator A defined in V and we want to solve the following time dependent equation:

$$\frac{\partial \phi}{\partial t} + A(\phi) = f(t), \quad t \in [0, T], \quad \phi(0) = \hat{\phi} \in V. \tag{2.55}$$

Assuming the operator A and the function f can be split in the following way:

$$A = A_1 + A_2 + \ldots + A_m, \quad f = f_1 + f_2 + \ldots + f_m, \tag{2.56}$$

a splitting scheme can be used to approximate the solution of (2.55). A first example is parallel splitting scheme or additive operator splitting (AOS) scheme [19] [20]. The idea is to choose a time step τ and set $\phi^0 = \hat{\phi}$. At each time level $t_j = j\tau$ we compute $\phi^{j+\frac{i}{2m}}$ in parallell for $i = 1, 2, \ldots, m$ from:

$$\frac{\phi^{j+\frac{i}{2m}} - \phi^j}{m\tau} + A_i(\phi^{j+\frac{i}{2m}}) = f_i(t_j), \quad \text{and then set } \phi^{j+1} = \frac{1}{m} \sum_{i=1}^{m} \phi^{j+\frac{i}{2m}}. \tag{2.57}$$

A second example, also called the multiplicative operator splitting scheme (MOS) can also be used to approximate the solution of (2.55):

$$\frac{\phi^{j+\frac{i}{m}} - \phi^{j+\frac{i-1}{m}}}{\tau} + A_i(\phi^{j+\frac{i}{m}}) = f_i(t_j), \quad i = 1, 2, \ldots, m. \tag{2.58}$$

The AOS and MOS schemes can be combined in different ways. One can split the operator A and the function f in the following way:

$$A = A_1 + A_2 + \ldots + A_k + A_{k+1} + \ldots + A_m,$$

$$f = f_1 + f_2 + \ldots + f_k + f_{k+1} + \ldots + f_m,$$

and use the AOS scheme on the first k terms and then the MOS scheme on the remaining $m - k$ terms. In other words, use the AOS scheme to solve $\phi^{j+\frac{i}{2m}}$ in parallel from:

$$\frac{\phi^{j+\frac{i}{2m}} - \phi^j}{k\tau} + A_i(\phi^{j+\frac{i}{2m}}) = f_i(t_j), \quad i = 1, 2, \ldots, k. \tag{2.59}$$

Set

$$\phi^{j+\frac{k}{m}} = \frac{1}{k}\sum_{i=1}^{k} \phi^{j+\frac{i}{2m}}. \tag{2.60}$$

Then, use the MOS scheme for the remaining terms, i.e. solve $\phi^{j+\frac{i}{m}}$ sequentially from

$$\frac{\phi^{j+\frac{i}{m}} - \phi^{j+\frac{i-1}{m}}}{\tau} + A_i(\phi^{j+\frac{i}{m}}) = f_i(t_j), \quad i = k+1, k+2, \ldots, m. \tag{2.61}$$

This is called a general AOS-MOS scheme.

In our application of this scheme we will split our problem into $d + 1$ terms, where d is the dimension of the image, i.e. 3. We will then use the AOS scheme to solve for $i = 1, ..., d$, and the MOS scheme for the last term. In section 3.2 this will be explained in detail.

Chapter 3

The algorithms

Now we will present three algorithms that are based on techniques from chapter 2. One involves the steepest descent, one involves Newton's method and one involves operator splitting applied to the steepest descent, combined with Newton's method.

In our first approach to segment an image, we will combine the steepest descent and Newton's method. The reason for this is that the steepest descent method converges very slowly, while the Newton's method needs good initial values, but converges fast. The idea is to terminate the steepest descent after a fixed number of iterations, and use the result from this as initial values for Newton's method. That should give satisfying results in as few iterations as possible.

The next approach involves operator splitting. We believe that the results from the first approach can be improved with respect to computational time, especially for 3 dimensional images. The idea is to apply operator splitting, i.e. an AOS-MOS scheme, to the steepest descent method. This results an AOS scheme in the same form as (2.59). For the MOS scheme in (2.61) Newton's method can be used, as we will see later.

The basics are the same for both approaches, i.e. the goal is to find saddlepoints of a certain Langrange functional, and the result is a piecewise constant function.

In the end of this chapter, we will take a look at how *Freesurfer* works. We have used the results from this method to compare with some of our results.

3.1 Steepest descent and Quasi-Newton

This section is based on [27]. We will use the piecewise constant level set function from (2.49) to segment our images. The basic idea is to divide the image Ω into subdomains Ω_i, $i = 1, 2, \ldots, n$, where n is known. To identify these subdomains, we will try to find a piecewise constant level set function ϕ, such that

$$\phi = i \text{ in } \Omega_i, \quad i = 1, 2, \ldots, N.$$

Associated with such a level set function ϕ, the characteristic functions of the subdomains are given as

$$\psi_i = \frac{1}{\alpha_i} \prod_{j=1, j\neq i}^{N} (\phi - j), \quad \alpha_i = \prod_{k=1, k\neq i}^{N} (i - k). \tag{3.1}$$

Given ϕ as in (2.49), we have $\psi_i(x) = 1$ for $x \in \Omega_i$, and $\psi_i(x) = 0$ elsewhere. From these functions we can extract geometrical information from the subdomains and the

interfaces between them. For example,

$$\mathit{Length}(\partial\Omega_i) = \int_\Omega |\nabla\psi_i| \, \mathrm{d}x, \quad \mathit{Area}(\Omega_i) = \int_\Omega \psi_i \, \mathrm{d}x, \tag{3.2}$$

where length and area becomes area and volume, respectively, for 3 dimensional images.

Having defined ψ as in (3.1), we see that the level set function also satisfies the relation $\phi = \sum_i i\psi_i$. We now define the constraints:

$$K(\phi) = (\phi - 1)(\phi - 2)\cdots(\phi - N) = \prod_{\phi=i}^{N}(\phi - i) \tag{3.3}$$

At every point in Ω, the level set function satisfies

$$K(\phi) = 0. \tag{3.4}$$

From the original Mumford-Shah modell presented in [23], where they wanted to find curves γ and constant values c_i to minimize:

$$\sum_i \int_{\Omega_i} |c_i - u_0|^2 \, \mathrm{d}x + \beta|\Gamma|. \tag{3.5}$$

From the initial image u_0, we search for a piecewise constant level set function u that satisfies

$$u = \sum_{i=0}^{N} c_i\psi_i, \tag{3.6}$$

such that $u = c_i$ in Ω_i.

Now we propose the following constrained minimization problem for segmenting an image:

$$\min_{\substack{c,\phi \\ K(\phi)=0}} \Big\{ F(c,\phi) = \frac{1}{2}\int_\Omega |u - u_0|^2 \, \mathrm{d}x + \beta\sum_{i=1}^{N}\int_\Omega |\nabla\psi_i| \, \mathrm{d}x + \nu\int_\Omega |\nabla\phi| \, \mathrm{d}x \Big\}. \tag{3.7}$$

We see that large approximation errors will be penalized by the fidelity term $\frac{1}{2}\int_\Omega |u-u_0|^2$. From (3.2), we see that the last two terms surpress oscillation, whereas the regularization parameters $\beta > 0$, $\nu > 0$ control the effect of the two terms.

In [18], the augmented Lagrangian method was used to solve the constrained minimization problem (3.7), and it defined as

$$L(c,\phi,\lambda) = F(c,\phi) + \int_\Omega \lambda K(\phi) \, \mathrm{d}x + \frac{r}{2}\int_\Omega |K(\phi)|^2 \, \mathrm{d}x, \tag{3.8}$$

where $\lambda \in L^2(\Omega)$ is the Langrange-multiplier and $r > 0$ is a penalty parameter. We normally choose the penalization parameter r large, compared to the other parameters. To minimize (3.7), we have to find the saddle point for L. The saddle point is found by minimizing L with respect to ϕ and c, and maximizing with respect to λ. By minimizing with respect to ϕ and c, we ensure that $F(c,\phi)$ is minimized, and by maximizing with respect to λ, the constraint must be fulfilled at convergence, otherwise the Lagrangian term of (3.8) will not vanish. The result is the the following algorithm:

Algorithm 1 *Choose initial values for ϕ^0, c^0 and λ^0. For $k = 1, 2, \dots$, do:*

1. Find c^k from
$$L(c^k, \phi^{k-1}, \lambda^{k-1}) = \min_c L(c^{k-1}, \phi^{k-1}, \lambda^{k-1}). \tag{3.9}$$
2. Use (3.6) to update $u = \sum_{i=1}^n c_i^k \psi_i(\phi^{k-1})$
3. Find ϕ^k from
$$L(c^k, \phi^k, \lambda^{k-1}) = \min_\phi L(c, \phi^{k-1}, \lambda^{k-1}). \tag{3.10}$$
4. Use (3.6) to update $u = \sum_{i=1}^n c_i^k \psi_i(\phi^k)$
5. Update the Langrange-multiplier by
$$\lambda^k = \lambda^{k-1} + rK(\phi^k) \tag{3.11}$$

The minimizer for (3.10) is solved here by the gradient descent method. We introduce an artificial time variable and look for a steady-state solution to the PDE

$$\phi_t = -\frac{\partial L}{\partial \phi}. \tag{3.12}$$

At steady-state $\phi_t = 0$, which means that $\frac{\partial L}{\partial \phi} = 0$. The time derivative is approximated using a forward Euler scheme

$$\phi_t \approx \frac{\phi^{new} - \phi^{old}}{\Delta t}. \tag{3.13}$$

Combining (3.12) and (3.13) gives us

$$\phi^{new} = \phi^{old} - \Delta t \frac{\partial L}{\partial \phi}(c, \phi^{old}, \lambda^{k-1}). \tag{3.14}$$

The step size Δt is chosen by a trial and error approach and it is fixed during the whole iterative procedure. It is not necessary to solve the minimization problem (3.10) exactly. The gradient iteration (3.14) is terminated when

$$\left\| \frac{\partial L}{\partial \phi}(c, \phi^{new}, \lambda^{k-1}) \right\|_{L^2} \leq \frac{1}{10} \left\| \frac{\partial L}{\partial \phi}(c, \phi^{k-1}, \lambda^{k-1}) \right\|_{L^2}$$

is reached or else after a fixed number of iterations. To compute $\frac{\partial L}{\partial \phi}$, we define $T(\phi) = \int_\Omega |\nabla \phi| \mathrm{d}x = \int_\Omega \sqrt{\phi_x^2 + \phi_y^2}\, \mathrm{d}x\, \mathrm{d}y$, and calculate

$$\begin{aligned} DT(\phi, \psi) &= \int_\Omega \lim_{\epsilon \to 0} \frac{\left[(\phi + \epsilon\psi)_x^2 + (\phi + \epsilon\psi)_y^2\right]^{1/2} - \left[\phi_x^2 + \phi_y^2\right]^{1/2}}{\epsilon} \mathrm{d}x\, \mathrm{d}y \\ &= \int_\Omega \lim_{\epsilon \to 0} \frac{\left[\phi_x^2 + \phi_y^2 + 2\epsilon(\phi_x\psi_x + \phi_y\psi_y) + \epsilon^2(\psi_x^2 + \psi_y^2)\right]^{1/2} - \left[\phi_x^2 + \phi_y^2\right]^{1/2}}{\epsilon} \mathrm{d}x\, \mathrm{d}y, \end{aligned}$$

where we define

$$\left[\phi_x^2 + \phi_y^2 + 2\epsilon(\phi_x\psi_x + \phi_y\psi_y) + \epsilon^2(\psi_x^2 + \psi_y^2)\right]^{1/2} = a,$$

and

$$\left[\phi_x^2+\phi_y^2\right]^{1/2}=b.$$

We then see that $DT(\phi,\psi)$

$$\begin{aligned}
&=\int_\Omega \lim_{\epsilon\to 0}\frac{a-b}{\epsilon}\cdot\frac{a+b}{a+b}\mathrm{d}x\,\mathrm{d}y\\
&=\int_\Omega \lim_{\epsilon\to 0}\frac{a^2-b^2}{\epsilon(a+b)}\mathrm{d}x\,\mathrm{d}y\\
&=\int_\Omega \lim_{\epsilon\to 0}\frac{2\epsilon(\phi_x\psi_x+\phi_y\psi_y)+\epsilon^2(\psi_x^2+\psi_y^2)}{\epsilon(\left[\phi_x^2+\phi_y^2+2\epsilon(\phi_x\psi_x+\phi_y\psi_y)+\epsilon^2(\psi_x^2+\psi_y^2)\right]^{1/2}+\left[\phi_x^2+\phi_y^2\right]^{1/2})}\mathrm{d}x\,\mathrm{d}y.
\end{aligned}$$

Letting $\epsilon\to 0$, we obtain

$$DT(\phi,\psi)=\int_\Omega\frac{\phi_x\psi_x+\phi_y\psi_y}{(\phi_x^2+\psi_y^2)^{1/2}}=\int_\Omega\frac{\nabla\phi\cdot\nabla\psi}{|\nabla\phi|}\mathrm{d}x\,\mathrm{d}y.$$

We define $v=\frac{\nabla\phi}{|\nabla\phi|}$ and from Green's identity (2.4) we see that

$$\int_\Omega v\cdot\nabla\psi\mathrm{d}x=-\int_\Omega\psi\nabla\cdot v\,\mathrm{d}x+\int_{\partial\Omega}\psi(\phi\cdot\mathbf{n})\mathrm{d}x. \tag{3.15}$$

Because of (3.32) and from examples 2.4, 2.5, 2.6 we get that

$$\frac{\partial L}{\partial\phi}=(u-u_0)\frac{\partial u}{\partial\phi}-\beta\sum_{i=1}^N\nabla\cdot\left(\frac{\nabla\psi_i}{|\nabla\psi_i|}\right)\frac{\partial\psi_i}{\partial\phi}-\nu\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right)+\lambda K'(\phi)+rK(\phi)K'(\phi). \tag{3.16}$$

As always, the derivative of L with respect to λ recovers the constraint

$$\frac{\partial L}{\partial\lambda}=K(\phi). \tag{3.17}$$

As u is linear with respect to c_i values, we see that L is quadratic with respect to c_i. Thus the minimization problem (3.9) can be solved exactly. We see that

$$\frac{\partial L}{\partial c_i}=\int_\Omega\frac{\partial L}{\partial u}\frac{\partial u}{\partial c_i}=\int_\Omega(u-u_0)\psi_i\mathrm{d}x\quad\text{for } i=1,2,\ldots,N. \tag{3.18}$$

Therefore, the minimizer of (3.9) satisfies a linear system of equations $Ac^k=b$:

$$\sum_{j=1}^n\int_\Omega(\psi_j\psi_i)c_j^k\mathrm{d}x=\int_\Omega u_0\psi_i\mathrm{d}x,\quad\text{for } i=1,2,\ldots,N. \tag{3.19}$$

Tests have shown that this algorithm alone converges very slowly, so our strategy is to terminate it at a certain number of iterations, and then use a quasi-Newton method to make it converge. Problems arise when trying to solve the segmentation problem with just Newton's method, as described in the next section. The reason for choosing a quasi-Newton method, is that we would have to invert a huge linear algebraic system due to the regularization terms in (3.7). So we define

$$Q(c,\phi,\lambda) = \frac{1}{2}\int_\Omega |u-u_0|^2 \,\mathrm{d}x + \int_\Omega \lambda K(\phi)\,\mathrm{d}x \tag{3.20}$$

Numerical experience also reveals that it is not necessary to use the penalization term with the Newton updatings. Thus, we also define

$$L_0(c,\phi,\lambda) = F(c,\phi) + \int_\Omega \lambda K(\phi)\,\mathrm{d}x \tag{3.21}$$

We see that L_0 is equal to L if we take $r = 0$ in (3.8). Also, the functional L_0 reduces to Q if we take $\beta = \nu = 0$. Thus the Hessian matrix for Q is a good approximation for the Hessian matrix of L_0 using the fact that β and ν are normally very small. So we arrive at the next algorithm:

Algorithm 2 *Choose initial values ϕ^0, c^0 and λ^0. For $k = 1, 2, \ldots$, do:*

1. Find c^k from
$$L(c^k, \phi^{k-1}, \lambda^{k-1}) = \min_c L(c^{k-1}, \phi^{k-1}, \lambda^{k-1}). \tag{3.22}$$
2. Update $u = \sum_{i=1}^n c_i^k \psi_i(\phi^{k-1})$
3. Find ϕ^k, λ^k from
$$\begin{pmatrix} \frac{\partial^2 Q}{\partial\phi^2} & \frac{\partial^2 Q}{\partial\phi\partial\lambda} \\ \frac{\partial^2 Q}{\partial\phi\partial\lambda} & 0 \end{pmatrix} \begin{pmatrix} \phi^k - \phi^{k-1} \\ \lambda^k - \lambda^{k-1} \end{pmatrix} = -\begin{pmatrix} \frac{\partial L_0}{\partial\phi} \\ \frac{\partial L_0}{\partial\lambda} \end{pmatrix} \tag{3.23}$$
4. Update $u = \sum_{j=1}^n c_j \psi_j\left(\phi^k\right)$
5. If converged, end the loop. Otherwise go to step 1.

To solve (3.23), we need

$$\frac{\partial^2 Q}{\partial\phi^2} = \left(\frac{\partial u}{\partial\phi}\right)^2 + (u-u_0)\frac{\partial^2 u}{\partial\phi^2} + \lambda K''(\phi), \qquad \frac{\partial^2 Q}{\partial\phi\partial\lambda} = \frac{\partial^2 Q}{\partial\lambda\partial\phi} = K'(\phi) \tag{3.24}$$

where $\lambda = \lambda^{k-1}$, $\phi = \phi^{k-1}$ and $u = u(c^k, \phi^{k-1})$, and we use that $\frac{\partial L_0}{\partial\lambda} = K(\phi^{k-1})$ and $\frac{\partial L_0}{\partial\phi}$ can be found from (3.16) by setting $r = 0$.

3.1.1 Problems with Newton updating without steepest descent

A natural alternative to the combination of the steepest descent method and Newton's methods is only Newton's method. But when trying to do the segmentation just by searching with Newton's method, one encounters some problems (apart from the known unstability of Newton's method when not sufficiently close to a local extremum). This is easily seen when looking into the algorithm. We have L as before:

$$L(c,\phi,\lambda) = F(c,\phi) + \int_\Omega \lambda K(\phi)\,\mathrm{d}x + \frac{r}{2}\int_\Omega |K(\phi)|^2\,\mathrm{d}x. \tag{3.25}$$

When we want to update with Newton's method, we have to solve the following system:

$$\begin{pmatrix} \frac{\partial^2 L}{\partial\phi^2} & \frac{\partial^2 L}{\partial\phi\partial\lambda} \\ \frac{\partial^2 L}{\partial\phi\partial\lambda} & 0 \end{pmatrix} \begin{pmatrix} \phi^k - \phi^{k-1} \\ \lambda^k - \lambda^{k-1} \end{pmatrix} = -\begin{pmatrix} \frac{\partial L}{\partial\phi} \\ \frac{\partial L}{\partial\lambda} \end{pmatrix} \tag{3.26}$$

Where $\frac{\partial L}{\partial \lambda} = K(\phi^{k-1})$ and $\frac{\partial^2 L}{\partial \phi \partial \lambda} = \frac{\partial^2 L}{\partial \lambda \partial \phi} = K'(\phi^{k-1})$. Solving this, we get

$$\frac{\partial^2 L}{\partial \phi \partial \lambda}\left(\phi^k - \phi^{k-1}\right) = -\frac{\partial L}{\partial \lambda} \tag{3.27}$$

Getting ϕ^0 from *kmeans* in MatLab, we start with an initial image where, in three phases, every voxel takes the value 1,2 or 3. This means $K(\phi) = 0$. $K'(\phi)$ is a function of ϕ. E.g. for a three phase segmentation problem, we will get $K'(\phi) = 3\phi^2 - 12\phi + 11$, which is most likely $\neq 0$. The result of this is that $\phi^k - \phi^{k-1} = 0$. In other words, nothing happens to ϕ.

Initializing the image in another way, we can achieve $K(\phi) \neq 0$, but the updating is still only dependent on $K(\phi)$ and $K'(\phi)$, so changing the regularizing parameter does not affect it.

3.2 Operator splitting scheme and Newton's method

This section is based on [8] and [26]. We propose to solve the following penalization functional:

$$\min_{\mathbf{c},\phi}\Big\{F(\mathbf{c},\phi) = \int_\Omega |u-u_0|^2 \mathrm{d}x + \beta\int_\Omega |\nabla\phi|\mathrm{d}x + \frac{1}{\mu}\int_\Omega W(\phi)\,\mathrm{d}x\Big\}, \tag{3.28}$$

where $W(\phi) = |K(\phi)|^2$ to deal with the constraint $K(\phi) = 0$. The solution must satisfy

$$a)\ \frac{\partial F}{\partial \mathbf{c}} = 0, \quad b)\ \frac{\partial F}{\partial \phi} = 0. \tag{3.29}$$

As in (3.19), the minimization of (3.28) with respect to c can be solved exactly by solving the linear system $Ac^k = b$:

$$\sum_{j=1}^{N}\int_\Omega (\psi_i\psi_j)c_i^k\,\mathrm{d}x = \int_\Omega u_0\psi_i\,\mathrm{d}x, \quad \text{for } i = 1,2,\ldots,N. \tag{3.30}$$

As we will see in the numerical section, we will try to get good initial values for $\mathbf{c}$, and update this only in the end.

Now, for the operator splitting scheme, we need to compute $\frac{\partial F}{\partial \phi}$:

$$\frac{\partial F}{\partial \phi} = -\beta\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + (u(\phi,\mathbf{c}) - u_0)\frac{\partial u}{\partial \phi} + \frac{1}{\mu}W'(\phi). \tag{3.31}$$

The variational formulation also impose the following boundary condition for ϕ on Ω

$$\frac{\nabla\phi}{|\nabla\phi|}\cdot\mathbf{n} = 0 \text{ on } \partial\Omega. \tag{3.32}$$

Using a steepest descent method for the minimization of (3.28) with respect to ϕ we get

$$\phi_t = \beta\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - (u(\phi,\mathbf{c}) - u_0)\frac{\partial u}{\partial \phi} - \frac{1}{\mu}W'(\phi). \tag{3.33}$$

We split the right side of (3.33) into $d+1$ terms:

$$\phi_t = B_1(\phi) + B_2(\phi) + \ldots + B_d(\phi) + C(\phi), \tag{3.34}$$

where

$$B_i(\phi) = \beta D_i \cdot \left(\frac{D_i \phi}{|\nabla \phi|} \right) - \frac{1}{d}(u(\phi, \mathbf{c}) - u_0) \frac{\partial u}{\partial \phi}(\phi, \mathbf{c}),$$

and

$$C(\phi) = -\frac{1}{\mu} W'(\phi).$$

$D_i = \frac{\partial}{\partial x_i}$ and d is the spatial dimension. We apply the AOS-MOS scheme from section 2.7 to (3.34). In each iteration step we use the AOS scheme on the first d terms:

$$\frac{\phi^{n+\frac{i}{2m}} - \phi^n}{d\tau} = \beta D_i \cdot \left(\frac{D_i \phi^{n+\frac{i}{2m}}}{|\nabla \phi^{n+\frac{i}{2m}}|} \right) - (u(\phi^{n+\frac{i}{2m}}, \mathbf{c}) - u_0) \frac{\partial u}{\partial \phi}(\phi^{n+\frac{i}{2m}}, \mathbf{c}) \tag{3.35}$$
$$i = 1, 2, \ldots, d,$$

where $m = d + 1$, and set

$$\phi^{n+\frac{1}{2}} = \frac{1}{d} \sum_{i=1}^{d} \phi^{n+\frac{i}{2m}}. \tag{3.36}$$

We then use $\phi^{n+\frac{1}{2}}$ as initial value and solve

$$\frac{\phi^{n+1} - \phi^{n+\frac{1}{2}}}{\tau} = -\frac{1}{\mu} W'(\phi^{n+1}). \tag{3.37}$$

To solve (3.35), which is nonlinear and implicit, we use the semi-implicit Picard iteration

$$\frac{\phi_i^{new} - \phi^n}{\tau d} = \beta D_i \cdot \left(\frac{D_i \phi_i^{new}}{|\nabla \phi_i^{old}|} \right) - \frac{1}{d}(u(\phi_i^{old}, \mathbf{c}^n) - u_0) \frac{\partial u}{\partial \phi}(\phi_i^{old}, \mathbf{c}^n), \tag{3.38}$$

which can be rewritten as

$$\phi_i^{new} - \tau \beta d D_i \cdot \left(\frac{D_i \phi_i^{new}}{|\nabla \phi_i^{old}|} \right) = \phi^n - \tau (u(\phi_i^{old}, \mathbf{c}^n) - u_0) \frac{\partial u}{\partial \phi}(\phi_i^{old}, \mathbf{c}^n) =: r_i. \tag{3.39}$$

We define the operator

$$A = \nabla \cdot (a(\mathbf{x}) \nabla), \tag{3.40}$$

where $a(\mathbf{x}) = \frac{1}{|\nabla \phi_i^{old}|}$. When operated on ϕ, A can be written as

$$\nabla \cdot (a \nabla \phi) = D_1(a D_1 \phi) + \ldots + D_d(a D_d \phi), \tag{3.41}$$

where $D_i = \frac{\partial}{\partial x_i}$, d is the spatial dimension. Then we can write

$$A_i = D_i \cdot (a(\mathbf{x}) D_i), \quad \text{and} \ \sum_{i=1}^{d} A_i = A. \tag{3.42}$$

Using this we can write (3.39) as

$$(I - \tau \beta d A_i) \phi_i^{new} = r_i, \tag{3.43}$$

where I is the identity matrix. For each i, we choose an initial value as ϕ_i^{old} and get a ϕ_i^{new} which is then taken to be ϕ_i^{old} to get a another ϕ_i^{new}. This procedure is iterated until convergence. After doing this for $i = 1, 2, \ldots, d$, we set

$$\phi^{n+\frac{1}{2}} = \frac{1}{d} \sum_{i=1}^{d} \phi_i^{new}. \tag{3.44}$$

For each i, the matrix $(I - \tau\beta dA_i)$ is a tridiagonal matrix. Thus the systems (3.43) can be solved fast using a tri-diagonal solver (see section 4.2).

To solve (3.37), we use Newton iteration. The value ϕ^{n+1} that satisfies this equation, will also be the solution to $G(\phi) = 0$, where

$$G(\phi) = \phi + \frac{\tau}{\mu}W'(\phi) - \phi^{n+\frac{1}{2}}. \tag{3.45}$$

This problem can easily be solved using the Newton iteration

$$\phi^{new} = \phi^{old} - \frac{G(\phi^{old})}{G'(\phi^{old})}. \tag{3.46}$$

To ensure uniqueness and convergence, we must make some restrictions. Since W' is a polynomial of degree $2N-1$, where N is the number of phases. Thus there are $2N-1$ roots. If we choose τ and μ such that $G' > 0$, we ensure that G is strictly increasing and thus there is only one real root. The rest of the roots will be complex. It is easy to see that

$$G'(\phi) = 1 + \frac{2\tau}{\mu}|K'(\phi)|^2 + \frac{2\tau}{\mu}K(\phi)K''(\phi). \tag{3.47}$$

In [8] they have showed that $G' > 0$ will impose the following constrain on τ and μ:

N	$\tau/\mu <$
2	2
3	0.71
4	0.09
...	...

The bound depends only on the number of phases N. So for a given μ we can calculate the time step τ such that $G' > 0$.

Finally, we arrive at the following algorithm:

Algorithm 3 *For $n = 1, 2, \ldots$ until convergence:*

1. Solve

$$\frac{\phi_i^{n+\frac{1}{2}} - \phi^n}{\tau d} = \beta D_i \cdot \left(\frac{D_i\phi_i^{n+\frac{1}{2}}}{|\nabla\phi_i^n|}\right) - (u(\phi_i^n, \mathbf{c}^n) - u_0)\frac{\partial u}{\partial\phi}(\phi_i^n, \mathbf{c}^n)$$
$$i = 1, 2, \ldots, d, \tag{3.48}$$

2. Set

$$\phi^{n+\frac{1}{2}} = \frac{1}{d}\sum_{i=1}^{d}\phi_i^{n+\frac{1}{2}}. \tag{3.49}$$

3. Use $\phi^{n+\frac{1}{2}}$ as initial value and solve

$$\frac{\phi^{n+1} - \phi^{n+\frac{1}{2}}}{\tau} = -\frac{1}{\mu}W'(\phi^{n+1}). \tag{3.50}$$

3.3 Freesurfer

At last in this chapter we give an introduction to *Freesurfer* [9] [11], as we will present results from this method to compare with our results. This method is very different from the ones presented, it is actually a combination of 7 algorithms, making it exact, but slow (see figure 5.28, 5.29, 5.30).

The components in this method are:

- Talairach Registration. The automated Talairach registration procedure [28] is used to compute the transformation matrix, where the transformation parameters are computed by using gradient descent at multiple scales to maximize the correlation between the individual volume and an average volume composed of a large number of previously aligned brains. In this way we can align our image in a way that is more comparable with other images, or an atlas.

- Intensity Normalization. Due to the noise that arise in the imaging-process (see section 1.2.2), there can be variations in both intensity and contrast across the image. This means that identical tissue types will give rise to varying intensities as a function of their spatial location. This part corrects such image intensity variations, by e.g. Gaussian smoothing.

- Skull-stripping. The next part is to take away the skull from the image (see figure 5.18). This is done by deforming an ellipsoidal template into the inner surface of the skull, with the help of two driving forces. (1) An MRI-based force, designed to drive the template outward from the brain, and (2) a curvature reducing force, enforcing a smoothness constraint on the deformed template.

- White matter labeling. An intensity-based classification, where possible overlap in the intensity distribution of white and grey matter is accounted for by allowing the grey-max-value be higher than white-low-value.

  Then, voxels that lie in a $3 \times 3 \times 3$ neighborhood which contains a significant number (e.g. 20%) of voxels with labels that differ from that of the central voxel are marked for further processing. It is known that the cortical surface is smooth, with finite curvature everywhere, resulting in a locally planar structure where cortical gray matter borders other tissue types such as white matter or CSF. The further segmentation procedure uses this information by detecting the plane-of-least-variance and using intensity information in this plane as a basis for classifying regions.

- Cutting planes. After the image has been segmented, cutting planes are computed which separate the cerebral hemispheres and disconnect subcortical structures from the cortical component.

- Connected components. The next step is to generate a single connected mass representing the white matter structure of each hemisphere. Any interior holes in the components representing white matter are filled, resulting in a single filled volume for each cortical hemisphere. This is accomplished using a connected components procedure.

- Surface tessellation, refinement and deformation. The volume is covered with a triangular tessellation and deformed to produce an accurate and smooth representation of the gray/white interface as well as the pial surface. Tessellation is done by

using two triangles to represent each (square) face separating voxels classified as a white matter of a given hemisphere from differently classified voxels.

Chapter 4

Discretization and implementation

We have used MatLab to implement our algorithms. Some of the more difficult tasks are included in this section.

MatLab has very good tools for working with matrices, and we work with the 3 dimensional images as 3 dimensional matrices, where each voxel is an element in the matrix. However, when computing derivatives and operators on an uniform grid, there might be problems. It should also be noted that many of the calculations involving these matrices are on element-element level, and not regular matrix-operations.

4.1 Curvature

To compute (3.16) we need to calculate the curvature, and for this we use the central difference to approximate the first-derivatives:

$$\phi_x^{i+1/2,j,k} = \phi^{i+1,j,k} - \phi^{i,j,k}, \tag{4.1}$$

and similar for y- and z-derivatives.

For 3-dimensional images the curvature can then be approximated in this way

$$\nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \approx \left(\frac{\phi_x}{\left(\phi_x^2 + \phi_y^2 + \phi_z^2 + \epsilon \right)^{1/2}} \right)_x + \left(\frac{\phi_y}{\left(\ldots \right)^{1/2}} \right)_y + \left(\frac{\phi_z}{\left(\ldots \right)^{1/2}} \right)_z \tag{4.2}$$

The reason for the ϵ is that we want a piecewise constant function. For numerical reasons we set this to be 10^{-5}. As seen in figure 4.1, when using central difference, the coordinates for our first-derivatives don't match for the $x-$ and $y-$direction. We compensate for this, for the first term in (4.2), by calculating the average of the four surrounding $y-$derivatives. For 3 dimensions, we naturally have to do this for the $z-$direction also. Then, we can add $\phi_x^2 + \phi_y^2 + \phi_z^2$.

We now have

$$\frac{\phi_x}{\left(\phi_x^2 + \phi_y^2 + \phi_z^2 + \epsilon \right)^{1/2}}, \tag{4.3}$$

where ϕ_y and ϕ_z are adjusted to ϕ_x. Computing the x-derivative of this with central difference, gives us

$$\left(\frac{\phi_x}{\left(\phi_x^2 + \phi_y^2 + \phi_z^2 + \epsilon \right)^{1/2}} \right)_x \tag{4.4}$$

with points on the nodes on the grid, as we can see from figure 4.1. After doing this for all three terms in 4.2, we can simply add these to get the approximation for the curvature.

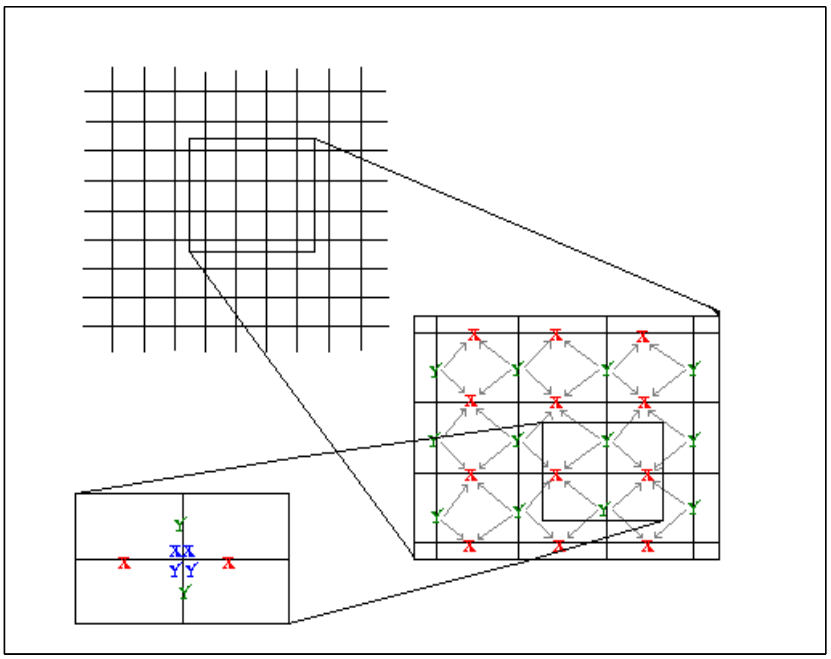

Figure 4.1: Computing the curvature. Every node represents a pixel. Letters in red and green symbolize first derivatives. The arrows show from which derivatives to compute the average, in order to adjust the grids. Blue letters symbolize terms like (4.4). Note that the first-derivatives are not on the grid-nodes, but the terms like (4.4) are.

4.2 Dimensional splitting

The operator A from (3.42)

$$A_i = D_i \cdot (a(\mathbf{x})D_i), \quad \text{where } \sum_{i=1}^{d} A_i = A \quad \text{and } a(\mathbf{x}) = \frac{1}{|\nabla \phi_i^{old}|}$$

is used in the operator splitting method to solve the system (3.43)

$$(I - \tau\beta d A_i)\phi_i^{new} = r_i.$$

Rembember that

$$r_i = \phi^n - \tau(u(\phi_i^{old}, \mathbf{c}^n) - u_0)\frac{\partial u}{\partial \phi}(\phi_i^{old}, \mathbf{c}^n).$$

We are solving this system for each spatial dimension independently. We use a combination of forward difference (D_i^f) an backward difference (D_i^b):

$$D_i^f = \begin{pmatrix} -1 & 1 & 0 & \dots \\ 0 & -1 & 1 & \dots \\ 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \; D_i^b = \begin{pmatrix} 1 & 0 & 0 & \dots \\ -1 & 1 & 0 & \dots \\ 0 & -1 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{4.5}$$

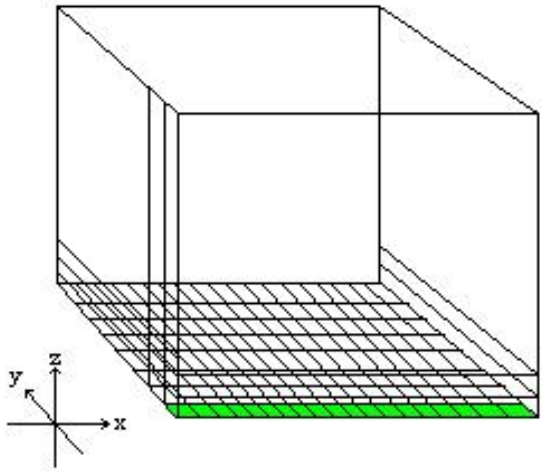

Figure 4.2: Imagine that this is one of the 3 dimensional matrices we are working with, either a or r_i, and every cube is a voxel. The green voxels are representing one row vector which would be used to solve (3.43). Only part of the grid is included to make it easier to see.

Starting in the x-direction ($i = 1$), we approximate a by

$$\frac{1}{\left(\phi_x^2 + \phi_y^2 + \phi_z^2 + \epsilon\right)^{1/2}}, \tag{4.6}$$

where, as in section 4.1, ϕ_y and ϕ_z are adjusted to fit the grid where we have ϕ_x by calculating the average of the 4 surrounding derivatives. For our 3 dimensional image, represented by a 3 dimensional matrix, we now solve (3.43) for all row vectors. Lets index the row vectors with (n, o), n beeing the y-coordinate and o beeing the z-coordinate. Let e be the row vector in question from a. We build the matrix A_i by using D_i^f and D_i^b and imposing zero Neumann boundary conditions to get

$$A_1 = \begin{pmatrix} -2e(1) & 2e(1) & 0 & \dots & 0 \\ e(1) & -e(1)-e(2) & e(2) & \dots & \vdots \\ 0 & e(2) & -e(2)-e(3) & e(3) & 0 \\ \vdots & & & & 0 \\ 0 & \dots & e(end-1) & -e(end-1)-e(end) & e(end) \\ 0 & \dots & 0 & 2e(end) & -2e(end) \end{pmatrix} \tag{4.7}$$

Now we use the row-vector (n, o) in r_1 to create the right hand side of (3.43). As A_1 for each row vector is a tridiagonal matrix, the system can be solved fast using a tridiagonal solver. The vector we get is the row vector (n, o) in ϕ_1^{new}. After doing this for all the row-vectors, we have a complete matrix ϕ_1^{new}.

Then we do this for the y-direction (i=2), where we do this for all column vectors. And finally for the z-direction, where the vectors are vertical.

To conclude, we see that we have to solve the system (3.43) in each spatial dimension, and for each dimension, it has to be solved for all vectors we have in that direction (see figure 4.2).

Chapter 5

Numerical results

In this section the numerical results are presented.

For the initial values, we have used different methods. *kmeans* is explained in the Appendix. In the second method, called *isodata*, we calculate the mean value C^1 of the image, and divide the image according to that value, such that the pixels/voxels with a value smaller then the average belong to the first phase, and the others to the second phase. Then we calculate the mean value C_1^1 and C_2^1 in the two regions, and set the average of these two to be C^2. This process is repeated i times, and we set $c = (C_1^i, C_2^i)$. We then set ϕ^0 to be a piecewise constant function according to the phases we have found during the iterations. As an alternative, we have simply scaled the image, which normally takes the values between 0 and 255, to take the values between 1 and n, where n is the number of phases in the image, i.e.

$$\phi^0(x) = 1 + \frac{u_0(x) - min_{x\in\Omega} u_0}{max_{x\in\Omega} u_0 - min_{x\in\Omega} u_0} \cdot (n-1). \tag{5.1}$$

For algorithm 1 and 2, we have used scaling to find ϕ^0 and *kmeans* to find initial c, and we have skipped the updating of c in (3.9) and (3.22), because this seems to work fine. Initial value λ^0 is set to 0. For algorithm 3 we have used *isodata* for both ϕ^0 and initial c. Also for this algorithm we skip the updating of c (3.30). Concerning the terms ϕ_i^{old} and ϕ_i^{new} in (3.39) we use ϕ^n to calculate r_i and A_i in (3.43) and do only one iteration to get the ϕ_i^{new} used in (3.44).

The montages are made for the brainweb-image in the axial view by showing every 5th slice from slice nr 41 to 156, and in the sagittal view by showing every 6th slice from 21 to 165. For the real image we have in the axial view displayed every 5th slice from 6 to 146, and in the sagittal view, every 4th slice from 11 to 111.

The tests are run on a computer with a 2 Opteron 270 dualcore processor and 8 GB memory.

5.1 Synthetic brain data

In the first part of this chapter, we will look at some test results of an synthetic brain image [22]. As seen in figure 5.1, it is relatively easy to distinguish the three phases. After running a test, one can easily say if the result is satisfying. As this is synthetic data, the correct segmentation is shown in figure 5.12, 5.13 and 5.14 [22]. For the real images, this becomes more difficult.

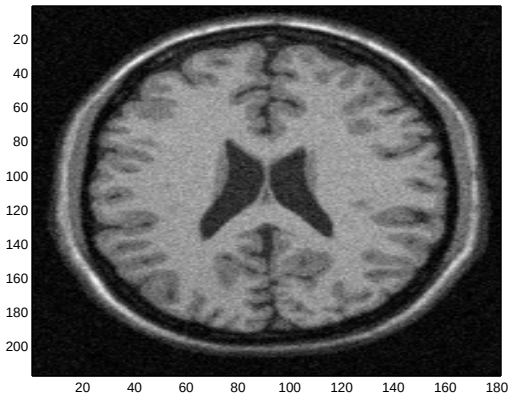

Figure 5.1: Slice 95 of a synthetic brain image from brainweb [22]. The size is $217 \times 181 \times 181$.

When starting to test the image in figure 5.1 (see also figure 5.2, 5.3 and 5.5) with algorithm 1 and 2, we started out with relatively few iterations for algorithm 1, as we wanted the algorithms to be as efficient as possible. The good results, though, first started to show as we approached 500. The results shown in figure 5.6, 5.7 and 5.8 came after 1000 iterations of algorithm 1 followed by 30 iterations of algorithm 2. The parameters are $r = 500$, $\nu = 500$, $\beta = 0$ and $dt = 1e^{-6}$, and we have used 3 phases. This took 97118 sek., or around 27 hours.

Next we have the results obtained with the operator splitting scheme and Newton's method. These are shown in figure 5.9, 5.10 and 5.11. We started with $\mu = 1000$, and for each iteration we set $\mu_{new} = c \cdot \mu_{old}$, where $c = 0.9$. The number of iterations is 100, we set $\beta = 0.03$ and have used 4 phases. The process took 21404 seconds, or around 6 hours.

In figure 5.12, 5.13 and 5.14 the skull is not included, as we have focused on 3 phases in this thesis, i.e. white matter, grey matter and the cerebrospinal fluid.

Finally, in figure 5.15 and 5.16 we try to compare the original image, our methods and the segmented image from brainweb [22], by taking out some of the slices from the montages (see figure-text). In the axial view there are only minor differences, but there are some errors in the results from algorithm 3 in the white matter. In the sagittal view there are some errors in the results from algorithm 1 and 2 with respect to the grey matter, and some errors in the results from algorithm 3 with respect to the white matter.

5.2 Real brain data

In the last part of this chapter, we have tested a real MRI (see figure 5.17). In this section our algorithms will be compared with a test run in *Freesurfer*. This method is explained in the appendix, and one of the features is the so-called skull-stripping, which takes the skull away. This simplifies the segmentation, because we don't have to consider the skull, e.g. by adding more phases to the problem. We have used the volume of the brain that *Freesurfer* returns after performing skull-stripping as our initial image (see figure 5.18, 5.19, 5.20 and 5.21). The results are not as easy to judge as in the case of the synthetic MRI, but we still get an idea of the three phases in the right image in figure 5.18.

For algorithm 1 and 2, as shown in figure 5.22, 5.23 and 5.24, we have used 400 iterations of algorithm 1 followed by 35 iterations of algorithm 2, $\nu = 1000$, $r = 500$,

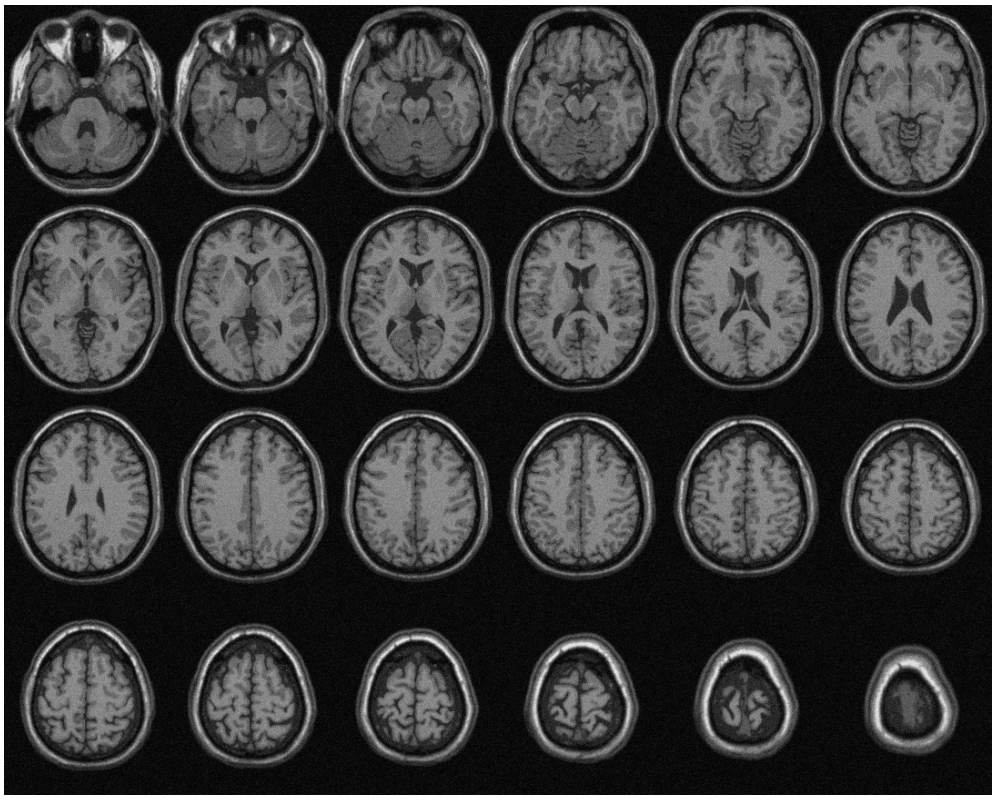

Figure 5.2: Synthetic image from brainweb [22], axial view.

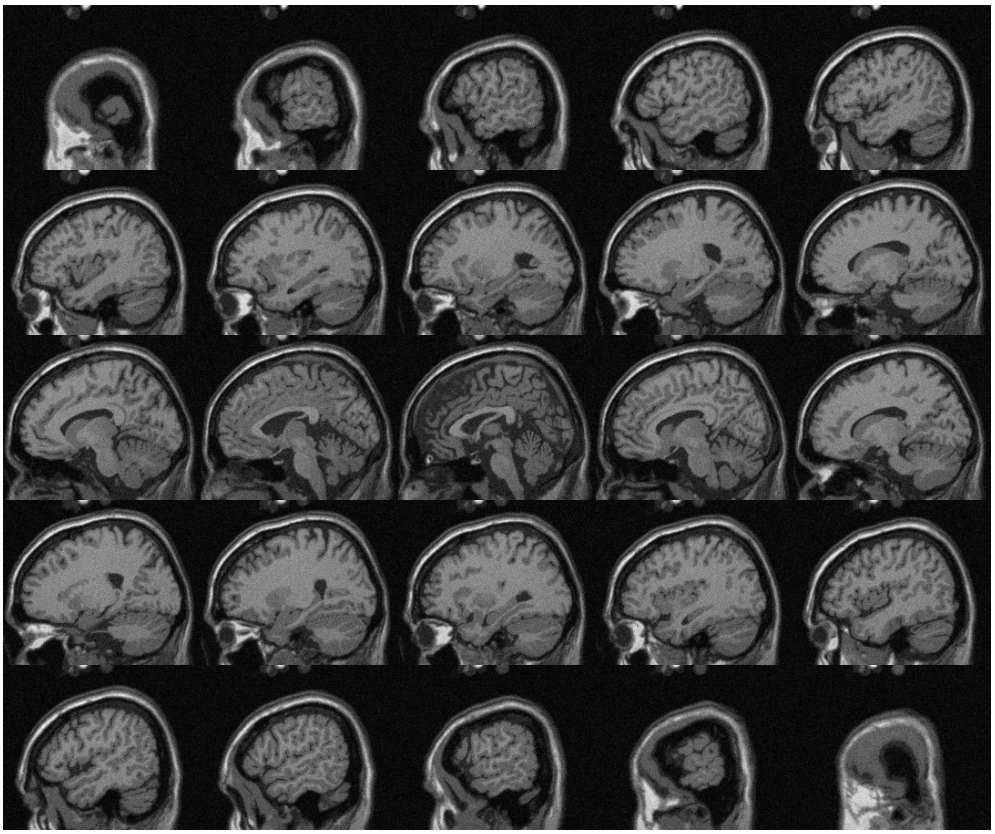

Figure 5.3: Synthetic image from brainweb [22], sagittal view.

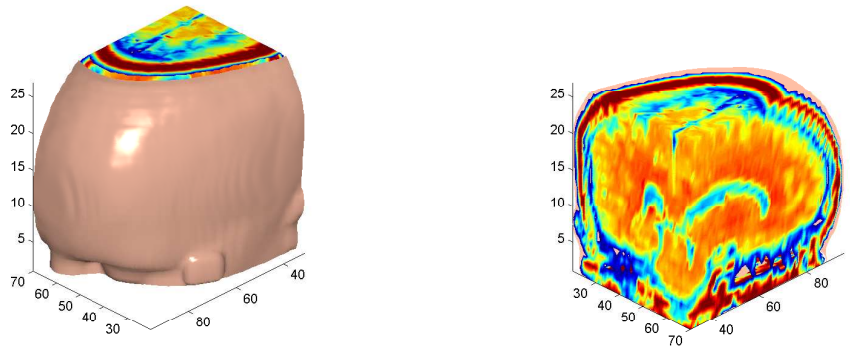

Figure 5.4: The visualization of the three end slices is done by cutting the brain image as shown to the left. After turning this image 180 degrees about the z-axis we get the image to the right.

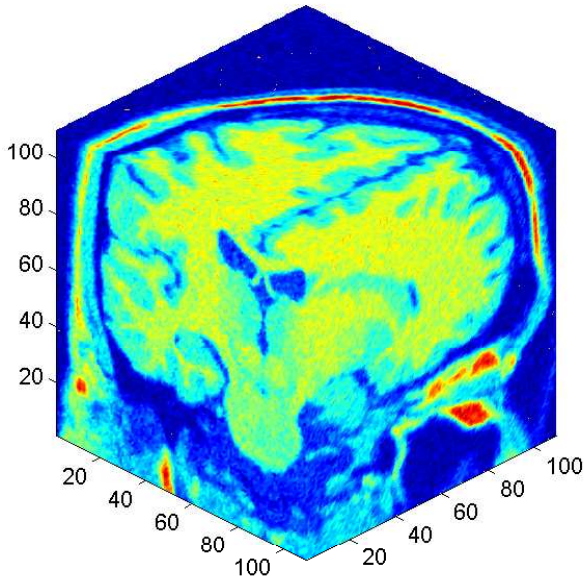

Figure 5.5: Synthetic image from brainweb [22], visualizing the three end slices.

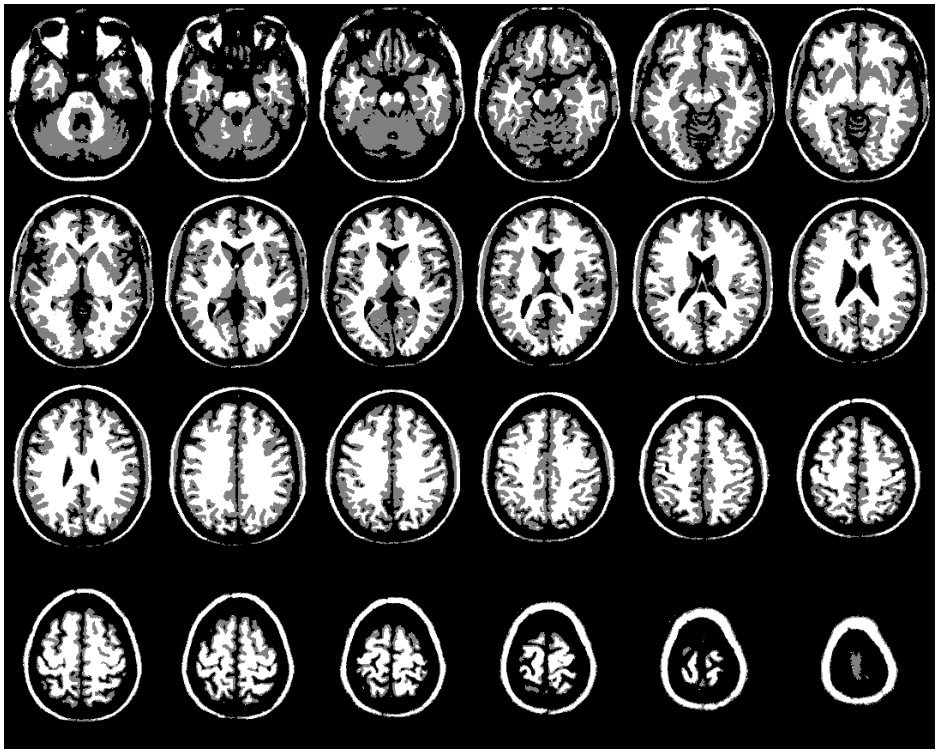

Figure 5.6: Results from algorithm 1 and 2, axial view

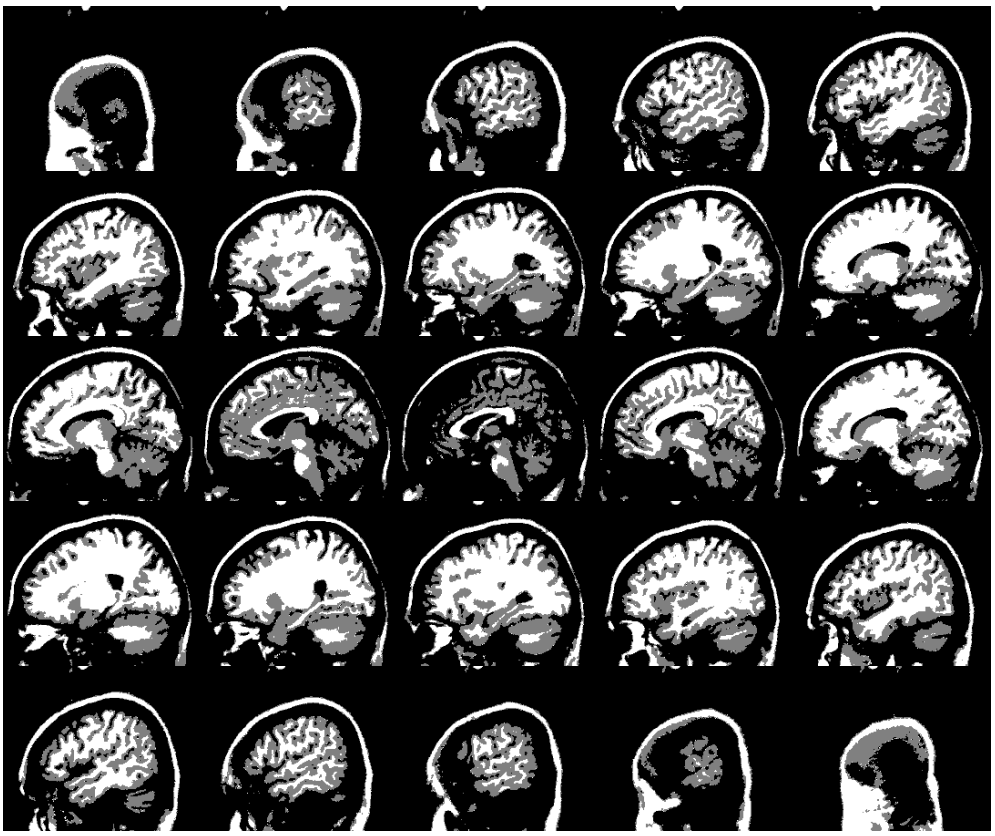

Figure 5.7: Results from algorithm 1 and 2, sagittal view

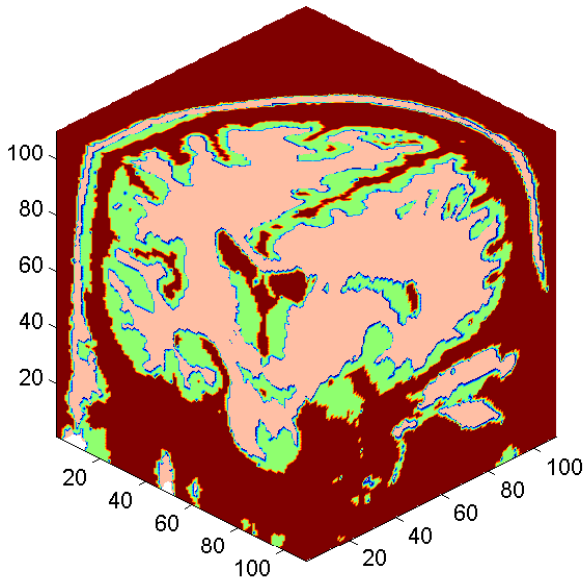

Figure 5.8: Results from algorithm 1 and 2, visualizing the three end slices.

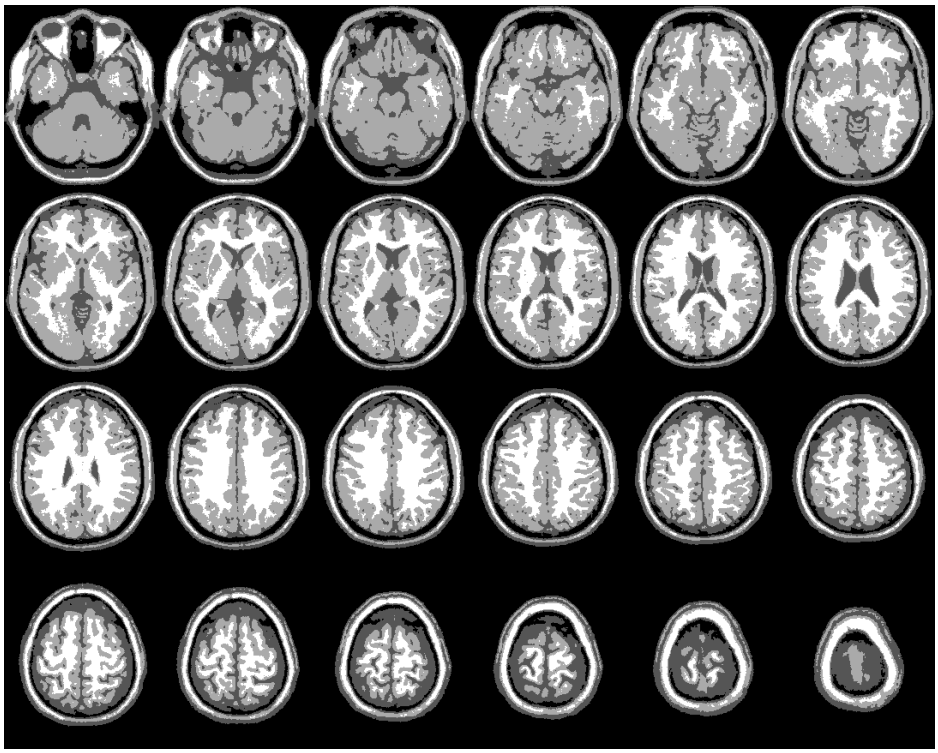

Figure 5.9: Results from algorithm 3, axial view

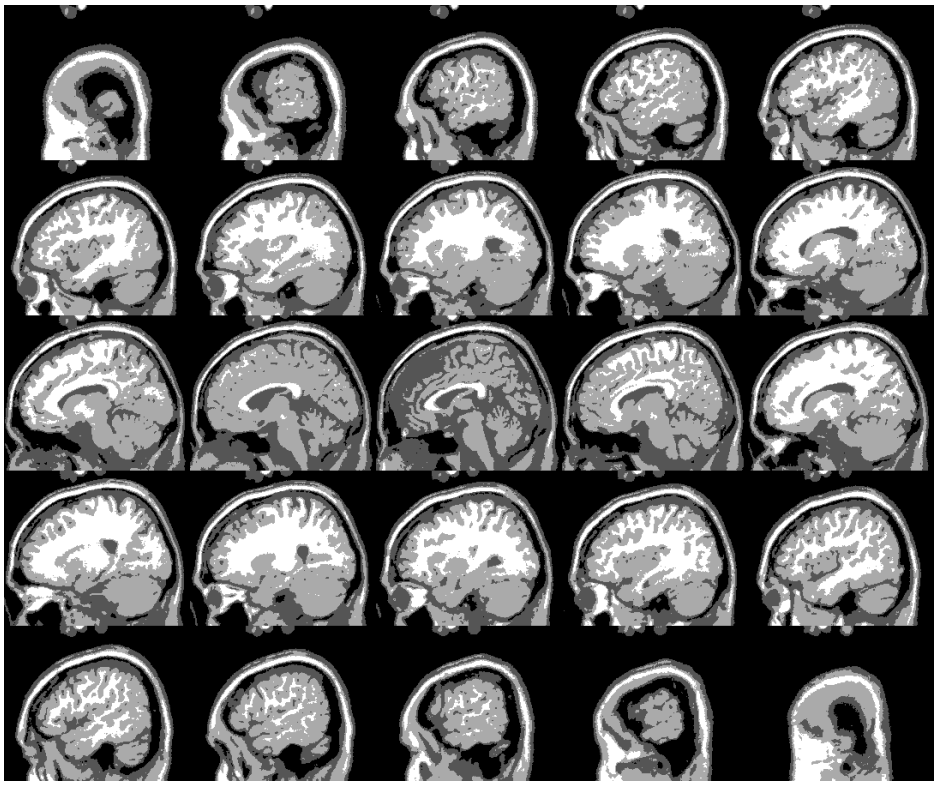

Figure 5.10: Results from algorithm 3, sagittal view

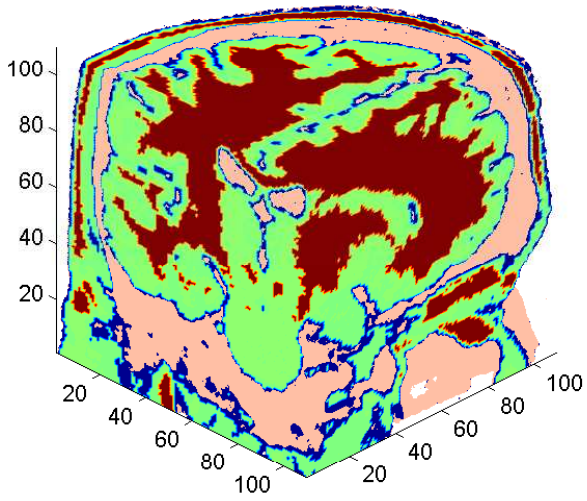

Figure 5.11: Results from algorithm 3, visualizing the three end slices.

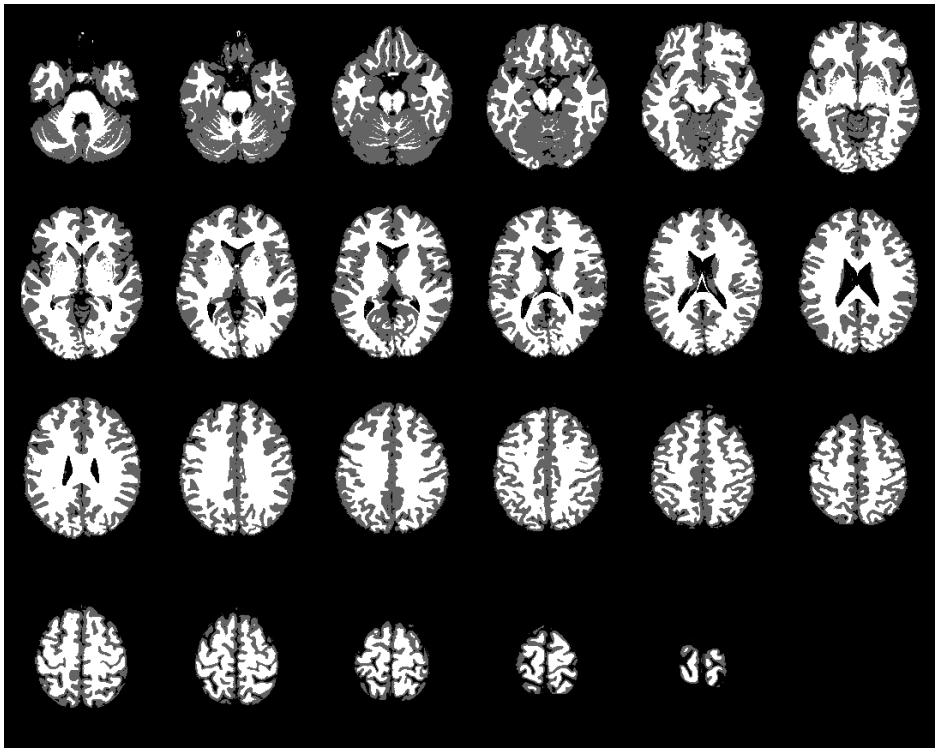

Figure 5.12: Segmented data from brainweb [22], axial view

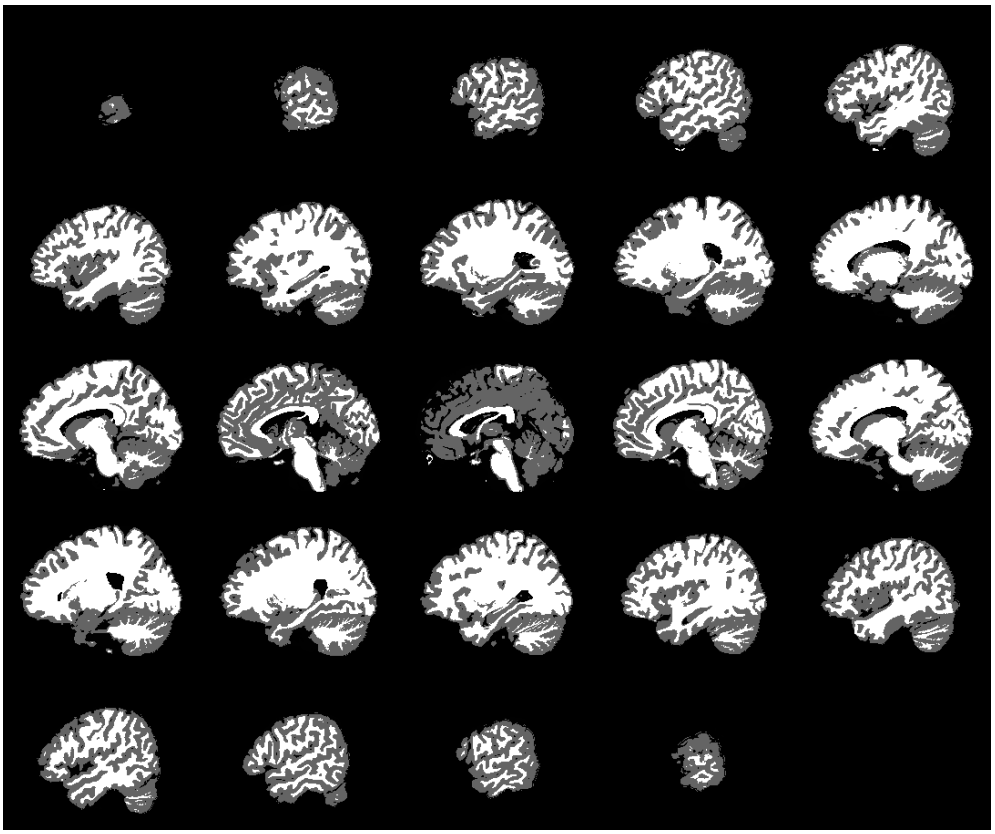

Figure 5.13: Segmented data from brainweb [22], sagittal view

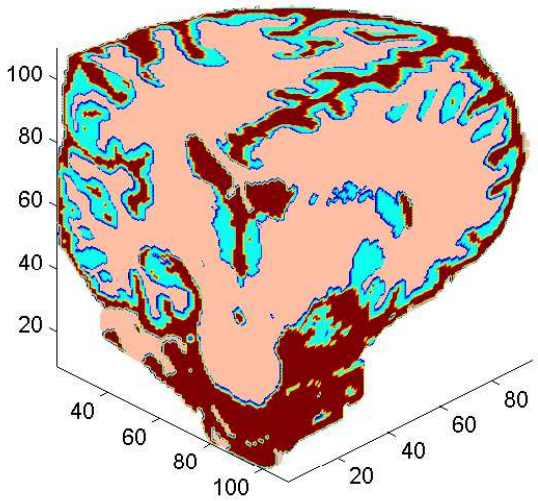

Figure 5.14: Segmented data from brainweb [22], visualizing the three end slices.

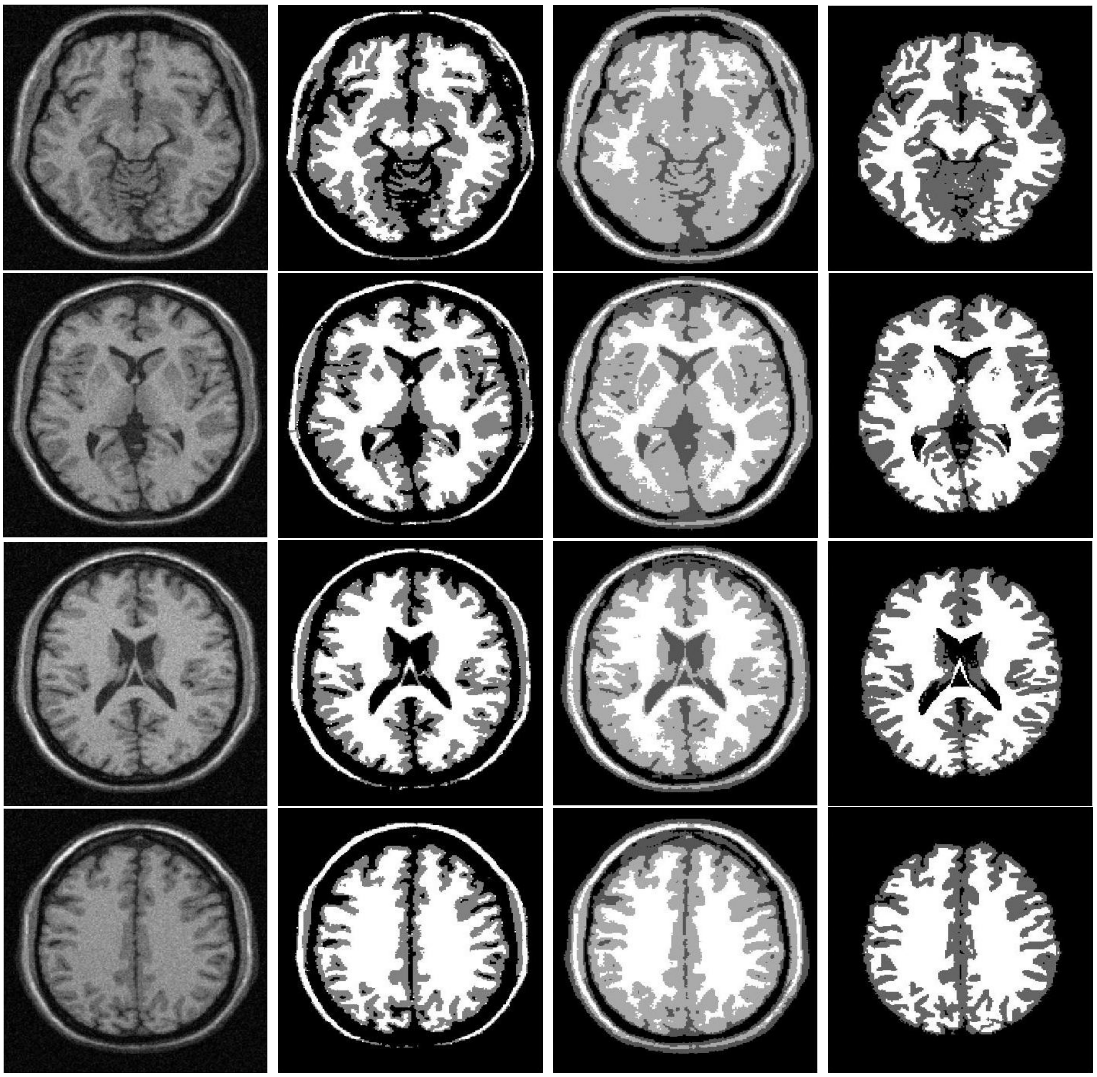

Figure 5.15: Comparing the synthetic image, axial view. From left to right we display: The original image, the result from algorithm 1 and 2, the result from algorithm 3 and the results from brainweb [22]. From top to bottom we show slice nr.: 61, 76, 91 and 106. These can be found in the montages as nr. 5, 8, 11 and 14 (going from left to right and top to bottom)

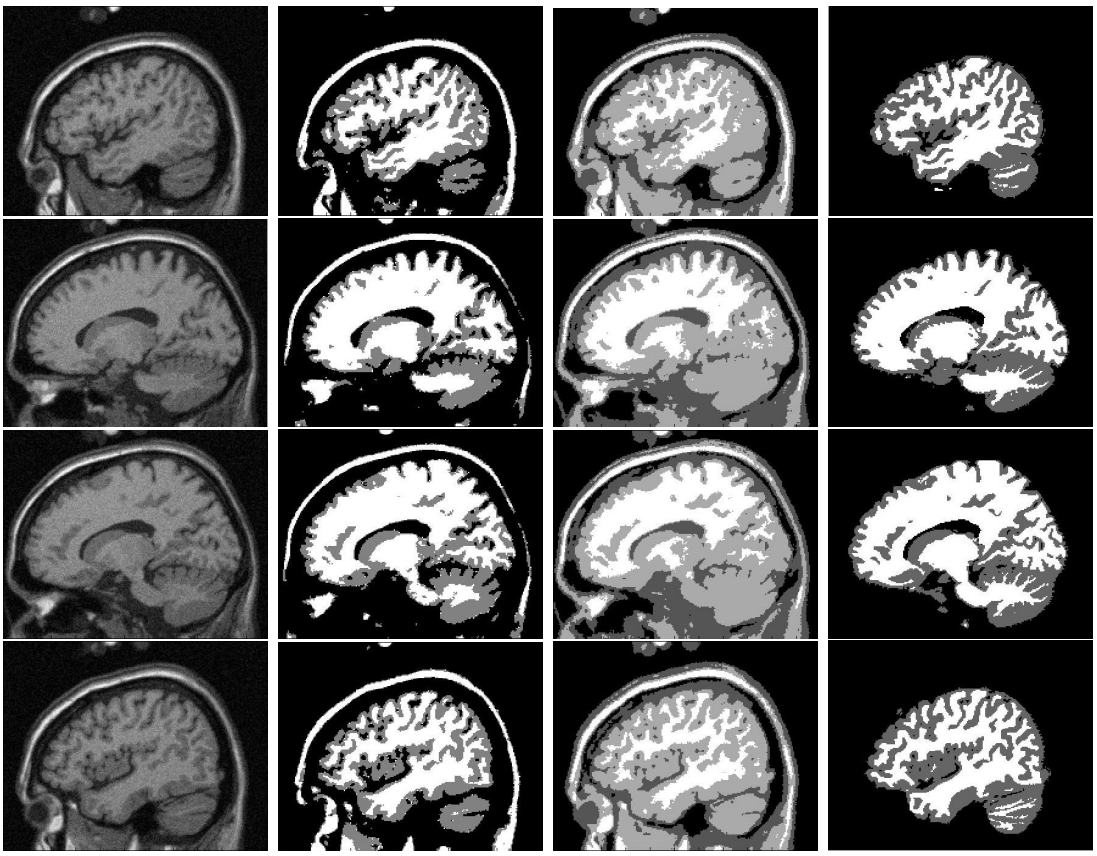

Figure 5.16: Comparing the synthetic image, sagittal view. From left to right we display: The original image, the result from algorithm 1 and 2, the result from algorithm 3 and the results from brainweb [22]. From top to bottom we show slice nr.: 45, 75, 105 and 135. These can be found in the montages as nr. 5, 10, 15 and 20 (going from left to right and top to bottom)

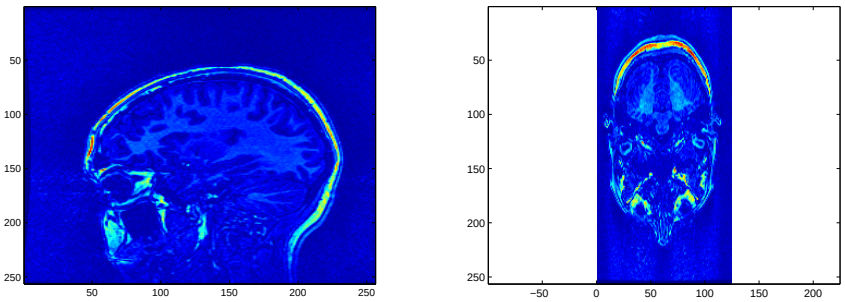

Figure 5.17: Real MRI. Sagittal (left) and axial view (right).

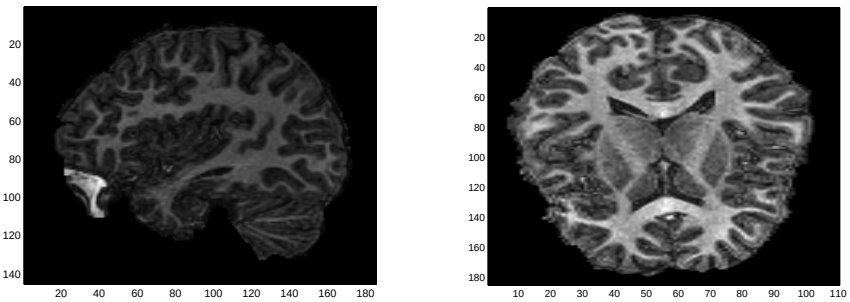

Figure 5.18: Real MRI without the skull. The size is $185 \times 110 \times 145$.

$\beta = 0$, $dt = 5e^{-7}$ and 5 phases. The large amount of phases needed is probably due large variations in intensity values. It took 37322 sec, or around 10 hours.

In figure 5.25, 5.26 and 5.27 the results after algorithm 3 are shown. We have used $\beta = 0.03$, $c = 0.9$ ($\nu = 1000$ in the beginning as before), 80 iterations and 4 phases. This took 7260 sec, or around 2 hours.

Next, in figure 5.28, 5.29 and 5.30 we present the results after performing the segmentation in *Freesurfer*. The process called 'processing stream (-autorecon1 , -autorecon2 , -autorecon3)' of the image in figure 5.17 took around 24 hours on a laptop with 2 GB of memory.

Finally, in figure 5.31 and 5.32 we have tried to compare our results with the original real MR image and the segmentation performed in *Freesurfer*. For the axial comparison, it is clear that there are differences. From what we can recognize as the white matter in the original image (to the left in figure 5.31), we see that all methods finds this with high accuracy. However, for algorithm 1 and 2 (nr. 2 from left), as we had to use 5 phases to do the segmentation, there is parts of another phase between the white and grey matter. Also, the CSF has gone into the same phase as the grey matter. For algorithm 3 it seems that it splitting the grey matter into two phases. This is also seem to be the case for *Freesurfer*, just in smaller amount.

In the sagittal view, it seems that the same errors are occuring, and in addition some parts of the white matter has gone into another phase, e.g. together with the grey matter.

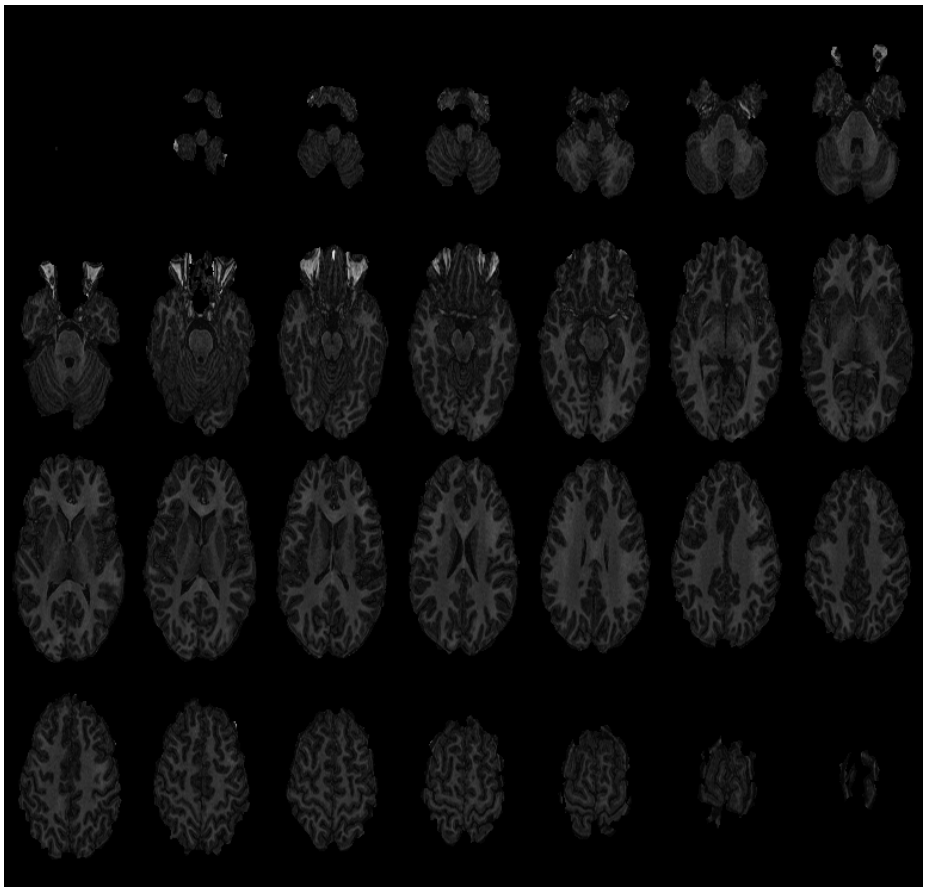

Figure 5.19: Real MRI, axial view.

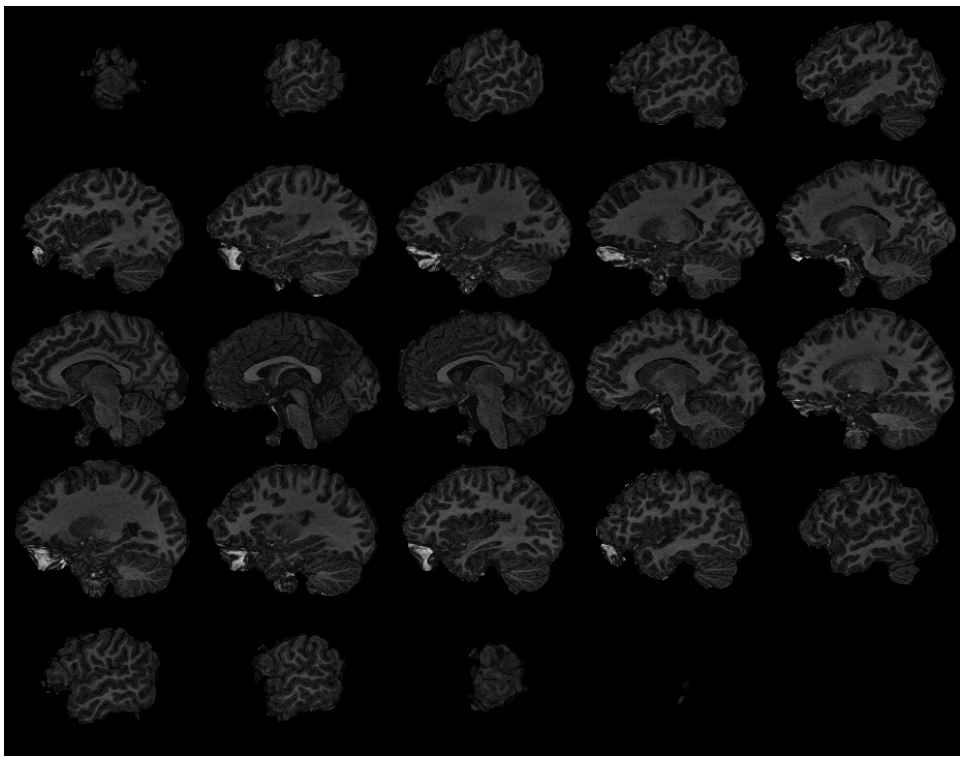

Figure 5.20: Real MRI, sagittal view.

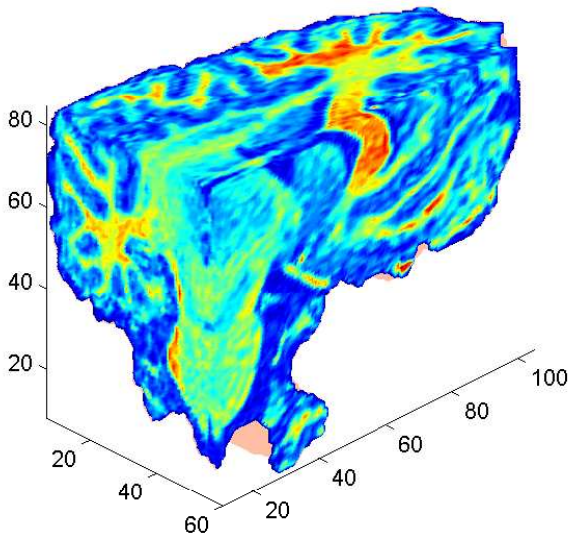

Figure 5.21: Real MRI, visualizing the three end slices.

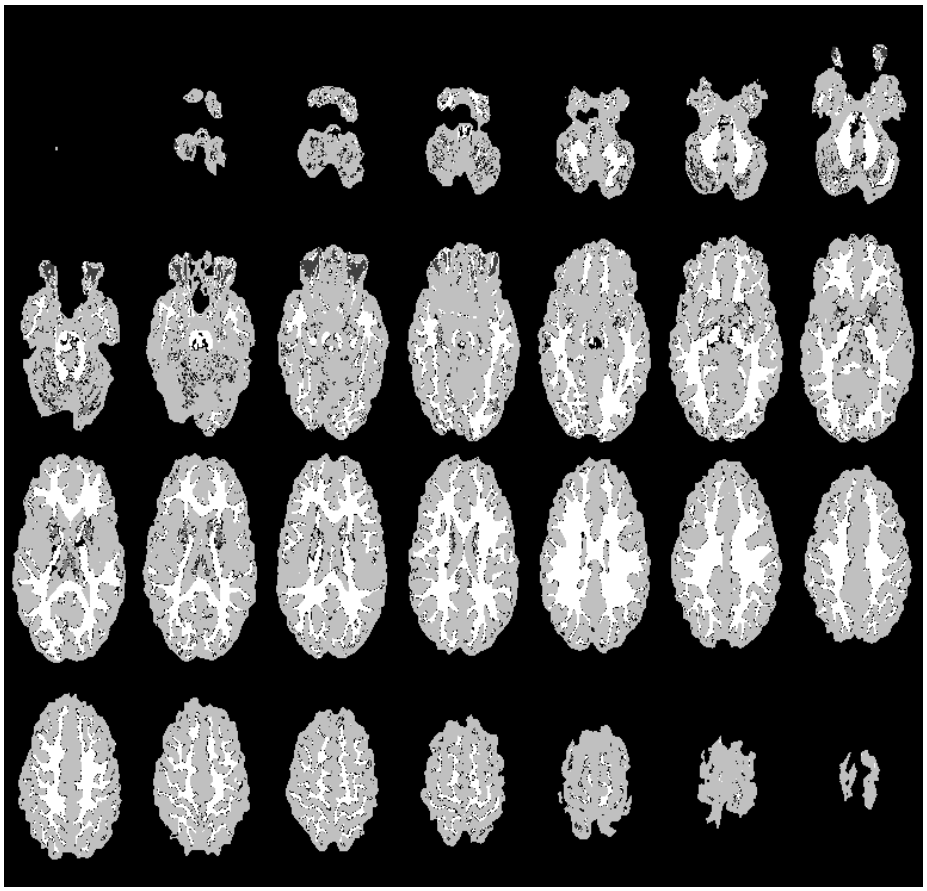

Figure 5.22: Results from algorithm 1 and 2, axial view.

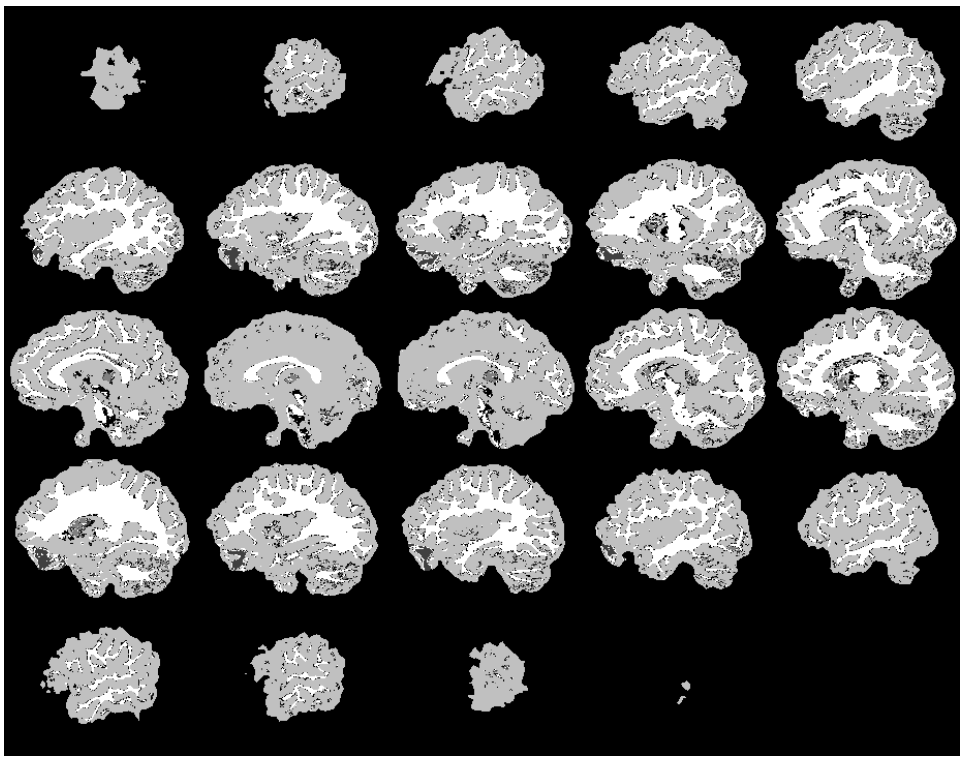

Figure 5.23: Results from algorithm 1 and 2, sagittal view.

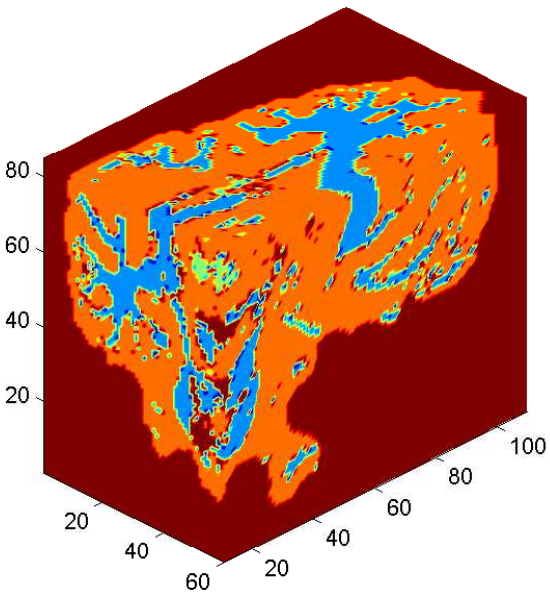

Figure 5.24: Results from algorithm 1 and 2, visualizing the three end slices.

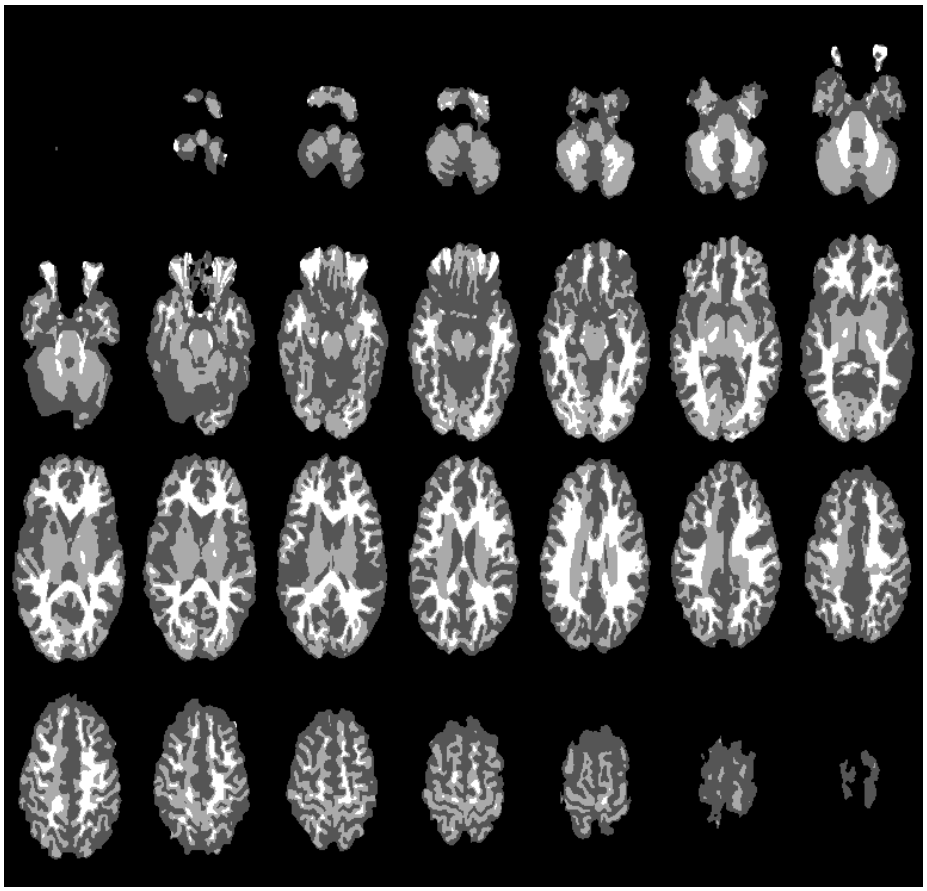

Figure 5.25: Results from algorithm 3, axial view.

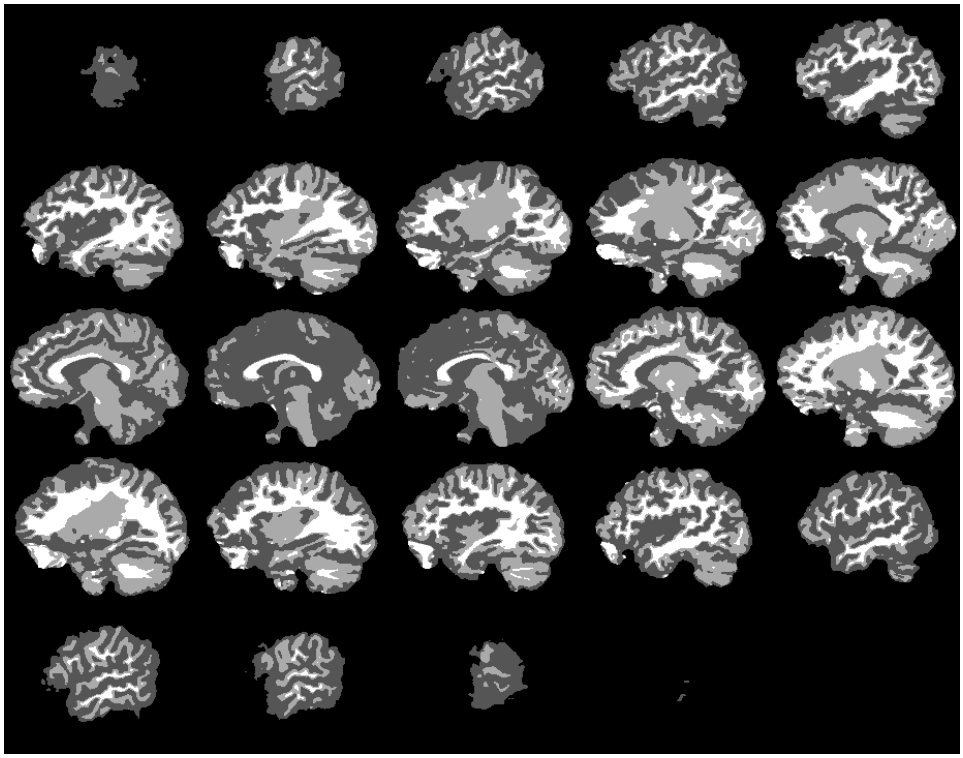

Figure 5.26: Results from algorithm 3, sagittal view.

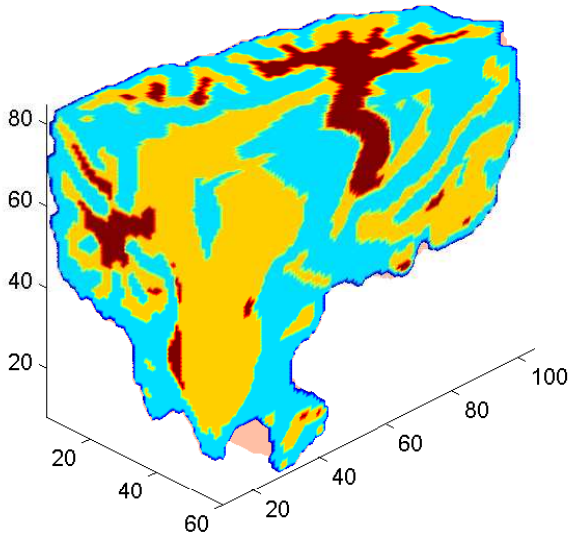

Figure 5.27: Results from algorithm 3, visualizing the three end slices.

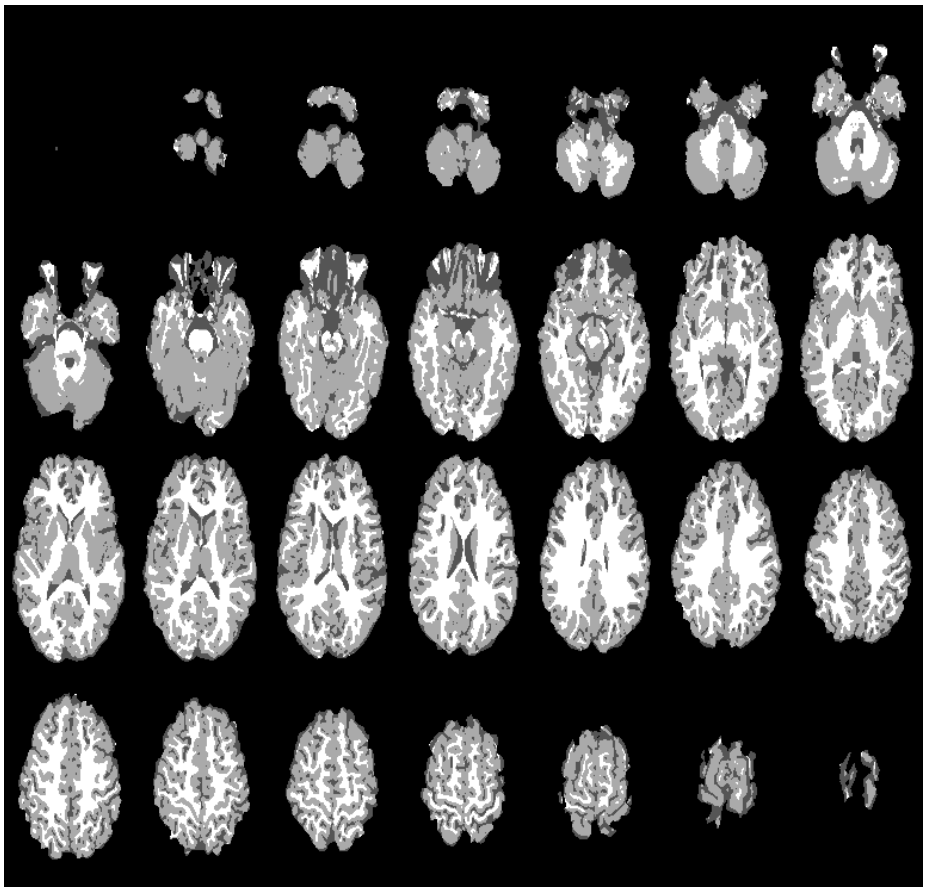

Figure 5.28: Segmentation in *Freesurfer*, axial view.

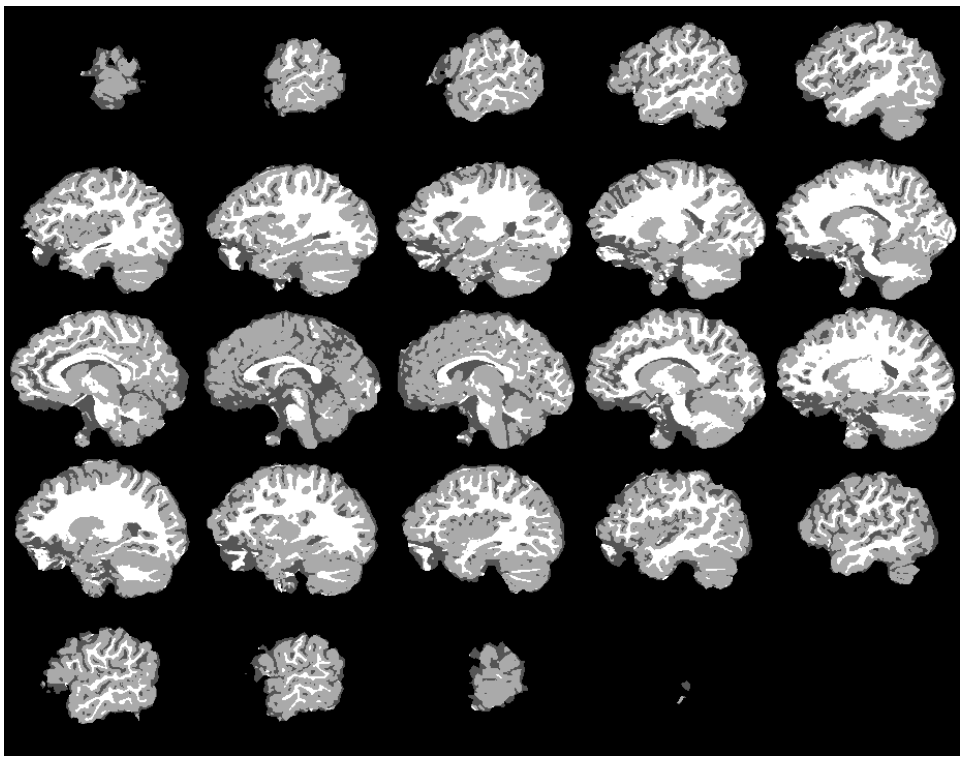

Figure 5.29: Segmentation in *Freesurfer*, sagittal view.

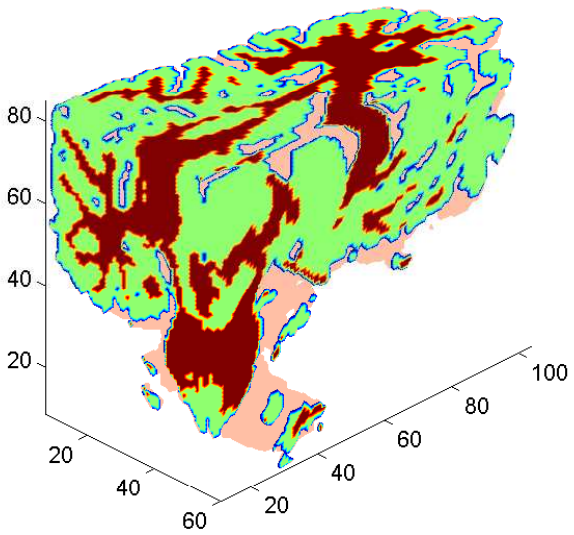

Figure 5.30: Segmentation in *Freesurfer*, visualizing the three end slices.

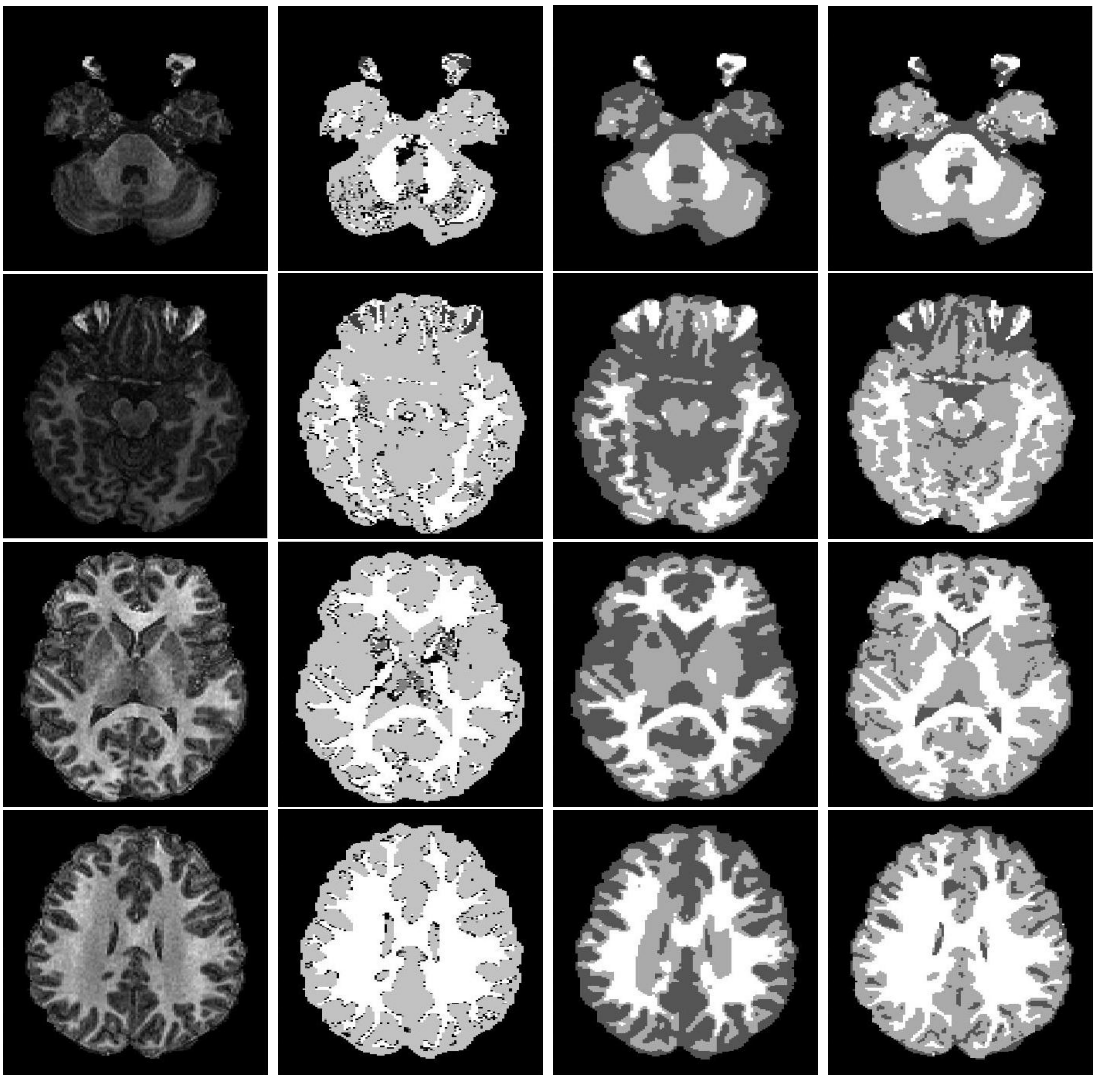

Figure 5.31: Comparing the real MR image, axial view. From left to right we display: The original image, the result from algorithm 1 and 2, the result from algorithm 3 and the results from *Freesurfer* [9]. From top to bottom we show slice nr.: 36, 56, 76 and 96. These can be found in the montages as image nr. 7, 11, 15 and 19 (going from left to right and top to bottom)

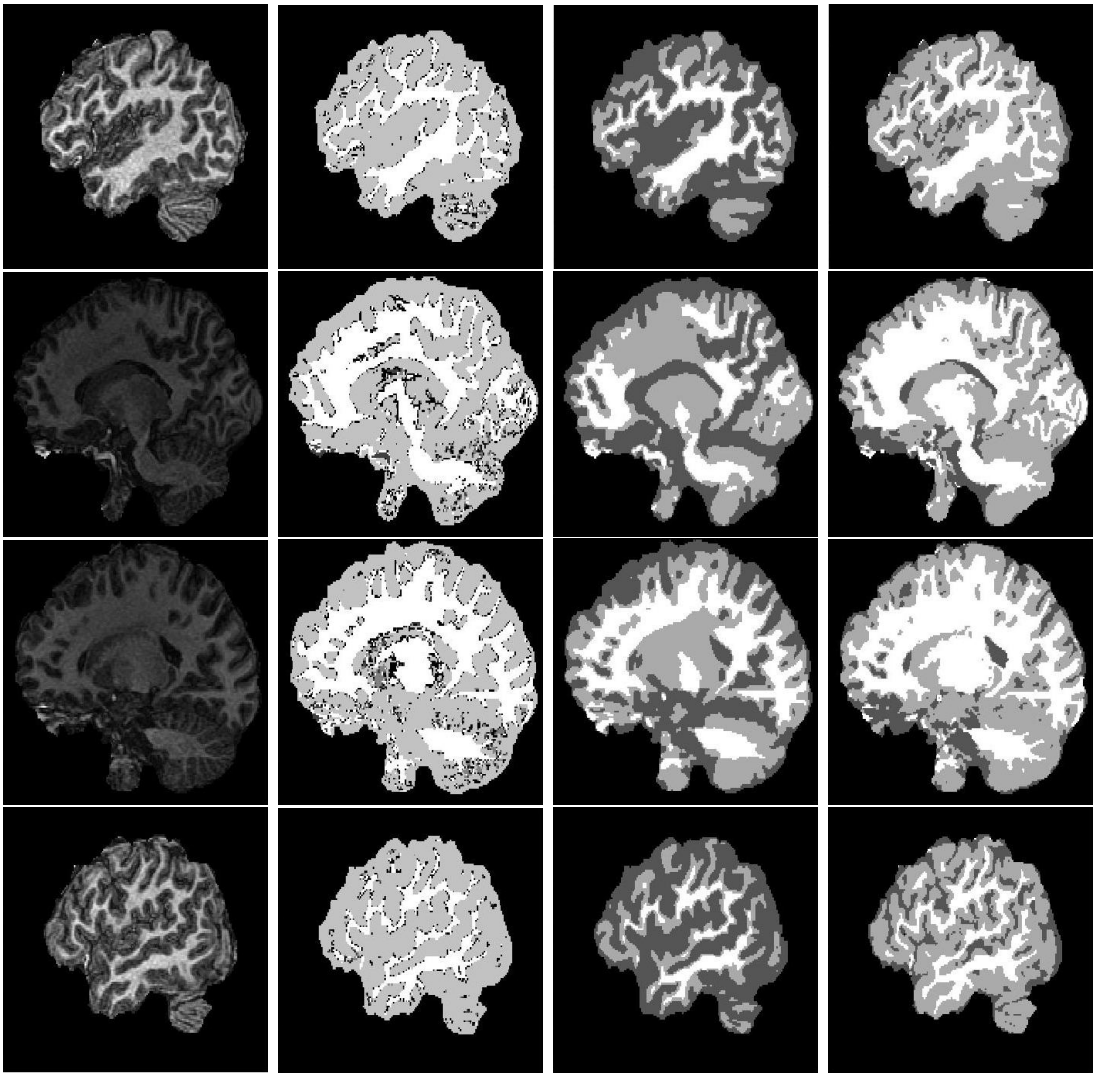

Figure 5.32: Comparing the real MR image, sagittal view. From left to right we display: The original image, the result from algorithm 1 and 2, the result from algorithm 3 and the results from *Freesurfer* [9]. From top to bottom we show slice nr.: 27, 47, 67 and 87. These can be found in the montages as image nr. 5, 10, 15 and 20 (going from left to right and top to bottom)

5.3 Summary and future work

Having in mind that we have presented very complicated images in this thesis, the results from the synthetic image are very good, and for the real MR image they could be improved slightly.

We see that using an operator splitting scheme significantly reduces the computational time. In the case of the synthetic brain image, this does not seem to have a particular negative effect others than those mentioned. This is the same case for the real MR image, where we could perform the same segmentation as for the synthetic image. For algorithm 1 and 2, it seemed that they had greater difficulties with the intensity values in the real MR image, possibly from other types of tissue present, and we had to do the segmentation for 5 phases.

The fact that the test of the real MRI took shorter time than for the synthetic MRI must be seen in connection with the size of the images and the number of iterations we have used. The latter is probably due to e.g. presence of the skull in the synthetic image. Also, the synthetic image used for segmentation contains the skull, which is expected to complicate the process.

An important aspect is the skull-stripping. For the synthetic image this does not seem to have an effect on the results for the brain, but it is complicating the process. For the real MR image it was almost impossible to do segmentation with the skull present, and we used the skull-stripping obtained from *Freesurfer* to do this. It should be able to do this quickly, so that it does not make the total segmentation time much longer.

The results obtained from algorithm 3 could be improved slightly for both images we have presented here. Algorithm 1 and 2 seem to have some problems with real MR images. This could possibly be improved by normalizing the intensity values. The initial values obtained with *kmeans* could vary for one image when running several tests, while the other methods are more stable.

Another possibility for medical images is to incorporate a-priori information, as in *Freesurfer*. This could lead to much better results, but would also require some manual work on the images.

For complicated images, there should also be developed a way to quantify the differences in the results, possibly compared to manual segmentation.

Appendix A

kmeans

The program *kmeans* in MatLab partitions the points in the n-by-p data matrix X into k clusters. The iterative partitioning minimizes the sum, over all clusters, of the within-cluster sums of the point-to-cluster-centroid distances. By default, *kmeans* uses squared Euclidean distances, i.e. $(x_{mi} - \bar{x}_m)^2$, where x_{mi} belong to cluster m, and $\bar{x}_m$ is the mean of the points in that cluster (the centroid). *kmeans* uses a two-phase iterative algorithm to minimize the sum of point-to-centroid distances, summed over all clusters:

- the first phase uses what the literature often describes as "batch" updates, where each iteration consists of reassigning points to their nearest cluster controid, all at once, followed by recalculation of cluster centroids. You can think of this phase as providing a fast but potentially only approximative solution as a starting point for the second phase.
- the second phase uses what the literature often describes as "online" updates, where points are individually reassigning if doing so will reduce the sum of distances, and cluster centroid are recomputed after each assignment. Each iteration during this second phase consists of one pass through all the points.

kmeans converge to a local optimum, in this case, a partition of points in which moving any single point to a different cluster increases the total sum of distances.

Bibliography

[1] R.A. Adams. *Calculus, a complete course*. Addison, Wesley, Longman, 2003.

[2] T.M. Apostol. *Calculus, volume II*. John Wiley and sons, 1969.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. *Image inpainting*. Proceedings of SIGGRAPH 2000, New Orleans, USA, 2000.

[4] G.F. Carrier and C.E. Pearson. *Partial Differential Equations, Theory and Technique*. Academic Press, 1976.

[5] T. Chan and X.-C. Tai. *Level set and total variation regularization for elliptic inverse problems with discontinuous coefficients*. J.Comput.Physics, 193:40-66, 2003.

[6] T. Chan and L.A. Vese. *Image segmentation using level sets and the piecewise constant mumford-shah model*. tech. rep, CAM Report 00-14, UCLA, Math. Depart., 2000.

[7] T. Chan and L.A. Vese. *Active contours without edges*. IEEE Image Proc., 10, pp. 266-277, 2001.

[8] O. Christiansen and X.-C. Tai. Fast implementation of piecewise constant level set methods. In *Image processing based on partial differential equations*, pages 253–272. Springer Verlag, 2006.

[9] CorTechs and A.A. Martinos center for biomedical imaging. *Freesurfer*. www.https://surfer.nmr.mgh.harvard.edu/, 6.9.2006.

[10] R.F. Curtain and A.J. Pritchard. *Functional analysis in modern applied mathematics*. Academic Press, 1977.

[11] A.M. Dale, B. Fischl, and M.I. Sereno. *Cortical Surface-Based Analysis*. NeuroImage 9, 179-194, 1999.

[12] D. L. Bihan et al. *Diffusion tensor imaging: Concepts and applications*. Journal of Magnetic Resonance Imaging, vol. 13, issue 4, 2001.

[13] E. Hodneland. *Segmentation of digital images*. Cand. Scient Thesis, Department of Mathematics, University of Bergen. Available online at www.mi.uib.no/ tai, 2003.

[14] J.P. Hornak. *The Basics of MRI*. www.cis.rit.edu/htbooks/mri/inside.htm, 29.07.2006.

[15] University in Bergen Image processing group, Department of Mathematics. *Diffusion tensor imaging*. www.mi.uib.no/BBG/RESEARCH/dtmri.html, 03.08.2006.

[16] L.P. Lebedev and I.I. Vorovich. *Functional analysis in mechanics.* Springer, 2003.

[17] J. Lie, M. Lysaker, and X.-C. Tai. *A binary level set model and some applications to Mumford-Shah image segmentation.* UCLA CAM 04-31, 2004.

[18] J. Lie, M. Lysaker, and X.-C. Tai. *A variant of the level set method and applications to image segmentation.* IEEE Transactions on Image Processing, 2005.

[19] T. Lu, P. Neittaanmaki, and X.-C. Tai. *A parallel splitting up method and its application to navier-stoke equations.* Applied Mathematics Letters, 4:25-29, 1991.

[20] T. Lu, P. Neittaanmaki, and X.-C. Tai. *A parallel splitting up method for partial differential equations and its application to navier-stokes equations.* RAIRO Math. Model. and Numer. Anal., 26:673-708, 1992.

[21] M. Lysaker. *Støyreduksjon i MR-bilder.* Hovedoppgave i anvendt matematikk, matematisk institutt og gruppa for nevroinformatikk og bildeanalyse, fysiologisk institutt, 2000.

[22] McGill University McConnell Brain Imaging Centre, Montréal Neurological Institute. *Brainweb.* http://www.bic.mni.mcgill.ca/brainweb/, 12.6.2006.

[23] D. Mumford and J. Shah. *Optimal approximation by piecewise smooth functions and associated variational problems.* Comm. Pure Appl. Math, 42, 1989.

[24] L.K. Nielsen. *Elastic Registration of Medical MR Images.* Cand. Scient. Thesis in Computational Science, Department of Mathematics and Neuroinformatics and Image Analysis Group, Department of Physiology, University of Bergen, 2003.

[25] S. Osher and J.A. Sethian. *Fronts propagating with curvature dependent speed: algorithms based on Hamilton-JAcobi formulations.* J. Comput. Phys., vol. 79, no. 1, 1988.

[26] I. Skjælaaen. *Image segmentation using a piecewise constant level set function.* Cand. Scient. Thesis in Applied Mathematics, Universitetet i Bergen, 2005.

[27] X.-C. Tai and C. Yao. *Fast piecewise constant level set methods with newton updating.* UCLA CAM 05-52, 2005.

[28] J. Talairach and P. Tournoux. *Co-planar sterotaxic atlas of the human brain.* Thieme, 1988.

[29] L.A. Vese and T. Chan. *A new multiphase level set framework for image segmentation via the Mumford and Shah model.* International Journal of Computer Vision, 50, pp.271-293, 2002.

[30] C.R. Vogel. *Computational methods for inverse problems.* SIAM, 2002.