Computer Aided Analysis of MR Brain Images

Stefan Wolff

Kgs. LYNGBY 2001
EKSAMENSPROJEKT
NR. 08/2001

IMM

Trykt af IMM, DTU

Preface

This thesis has been prepared at the numerical analysis section of Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark (DTU) in partial fulfillment of the requirement for the degree of Master of Science in Engineering.

During this work I have had the privilege of cooperating with a group of truly delightful people. First of all I would like to thank my advisor, Per Skafte Hansen, for his encouragement, flexibility, and commitment. I would also like to thank Lars G. Hanson and Torben Lund (Danish Research Center for Magnetic Resonance) for always being ready to share their knowledge of the science of magnetic resonance and state-of-the-art MRI techniques, as well as for their contagious enthusiasm. I am also indebted to Margrethe Herning and Anne-Mette Leffers (DRCMR) for suggestions for improvements and for lifting a corner of the veil obscuring the esoteric field of Yedi radiology. Next, I would like to thank the Oticon Foundation for supporting this work. I would also like to thank the faculty, staff and students of the numerical analysis section of IMM for providing a stimulating and relaxed environment.

Finally, I would like to thank my friends and the members of my family for each providing someone to miss.

Kgs. Lyngby, Denmark, August 31st, 2001
Stefan Wolff

Summary

Cortical dysplasia, the malformation of the cerebral cortex (the layer of gray matter forming the surface of the brain), can take on a number of forms, such as:

- A thickening of the cortex
- A diffuse organization, with white and gray matter apparently mixing
- Brain geometry deformation
- Heterotopia, the misplacement of gray matter inside white

The object of the work reported here is to provide a clinical expert (a radiologist) with an algorithmic tool that will assist in the detection of the first two of these. The algorithm analyzes 3D data sets, obtained as luminance level values representing the output from MR scans of the human brain. The data is organized as a voxel image, with a geometry representing a straightforward discretization of that of the brain under study. The only information available as to the category of a given voxel (gray matter, white matter, cerebro-spinal fluid, vascular tissue or background) is that of the luminance level. An automated segregation will suppress precisely the information sought by the present algorithm, while (time-consuming) human intervention would be tantamount to a solution of the problem addressed. The algorithms are developed with this limitation observed throughout.

Keywords

Magnetic Resonance, Cortical Dysplasia, Gradient Correlation Algorithm, Voxel Images, Visualization.

Dansk resumé

Cortical dysplasi, fejludvikling af hjernebarken (et lag af grå substans der udgør hjernens overflade) kan give sig udtryk på adskillige måder, såsom:

- Fortykkelse af hjernebarken
- Diffus overgang fra grå til hvid substans
- Deformation af hjernen
- Heterotopi: grå substans fejlplaceret inde i hvid substans

Hensigten med nærværende arbejde er at udruste en klinisk ekspert (en radiolog) med et algoritmisk værktøj der kan hjælpe med at detektere forekomster af de første to af de ovennævnte fænomener. Algoritmen analyserer 3D-datasæt bestående af luminansværdier fremkommet af MR-skanninger af en menneskehjerne. Data er organiseret som et voxel-billede, hvis geometri hidrører fra en direkte diskretisering af den undersøgte hjerne. Den eneste til rådighed stående information, der kan give et fingerpeg om vævstypen (grå substans, hvid substans, cerebro-spinal væske, karvæv eller baggrund) er luminansværdien. En automatisk inddeling vil undertrykke netop den information, der søges af den her præsenterede algoritme, og (tidskrævende) menneskelig indblanden ville svare til en løsning på problemet. Algoritmerne er udviklet med denne begrænsning.

Contents

List of Figures

Chapter 1

Introduction

The advent of the possibility of visualizing the interior of the human body without surgical intervention has had a significant impact on the world of medical science ever since the discovery of x-rays. One fairly recent imaging modality, known as magnetic resonance imaging (MRI), offers detailed images of the human anatomy with high soft tissue contrast. Owing to the multitude of imaging parameters involved, and the fact that, unlike many other modalities, MRI is not confined to transverse cross-sections, it constitutes a versatile way of performing non-invasive *in vivo* evaluation of tissue anatomy, flow, metabolism, etc.

As MR examinations become routine in many hospitals, the problem arises of combing through the vast amount of generated data. Conventional visualization media such as film sheets and computer screens are inherently two-dimensional, rendering them incapable of displaying true three-dimensional data sets. Attempts to overcome this limitation, e.g. by imposing transparency, are likely to fail in the general case because of the reduced dimensionality involved. Clinical examiners (radiologists) have to make do with two-dimensional data, in the form of planar or curvilinear slices.

The purpose of the work presented here is to provide a clinical expert (a radiologist) with a way of overcoming the reduced dimensionality inherent in conventional media by the development of a tool to assist in searching through large 3D magnetic resonance (MR) images. This tool has been specifically applied to the visualization and detection of development disorders, known as focal cortical dysplasia, in the layer of gray matter forming the surface of the brain (the cerebral cortex). Disorders of this kind are associated with epilepsy. Although subtle dysplastic lesions may not have any direct effect on the intellect, the effects of medication and frequent epileptic seizures may lead to cognitive impairment. Owing to drug resistance or medically refractory epilepsy, surgery may provide the only solution. Subtle lesions often show good prognoses for surgical intervention, but may be hard to detect owing to their subtlety as well as the level of noise in the MR image. Even for an expert radiologist, subtle lesions may take days to detect, wherefore any reduction in the time spent searching would be beneficial.

This thesis describes an algorithm originating from 3D image analysis for automatic detection of key symptoms of focal dysplasia. The algorithm is embed-

ded in a real-time graphics system to facilitate the reading and interpretation of results. A description of key features of this graphics system as well as a short introduction to relevant subjects are also included in the thesis.

The reader is assumed to be familiar with basic computer graphics and image analysis. Prior knowledge of magnetic resonance and MRI is not a prerequisite.

A related paper was submitted to the 10th international conference on Discrete Geometry for Computer Imagery in Bordeaux, France.

This work was supported in part by a grant from the Oticon Foundation.

Chapter 2

Introductory Theory

This chapter attempts to give the reader sufficient insight into image processing, magnetic resonance imaging (MRI), and the human brain so as to make accessible the subsequent chapters of this thesis. It is by no means intended as a comprehensive treatment of the subjects.

The next chapter similarly covers the prior work on which the method presented here is based.

The contributions comprising the kernel of this thesis are presented from chapter 4 and onwards.

2.1 Magnetic Resonance Imaging Basics

Ensembles of atomic nuclei possessing an odd number of neutrons or protons exhibit behaviour comparable to that of a compass needle, that is to say these ensembles, when placed in a magnetic field, tend to align in the direction of this field. This alignment does not happen instantly, but takes some time depending on the surroundings of each nuclei as well as the strength of the applied magnetic field. This gradual alignment is called *relaxation*. During this relaxation, the ensembles precess in the plane perpendicular to the applied field at a frequency called the *Larmor frequency*, which is defined by

$$f = \frac{\gamma}{2\pi} B \,, \tag{2.1}$$

where f is the resonance frequency, B is the field strength, and γ, called the *gyromagnetic ratio*, is a number depending on the type of nuclei. This transversal component of the magnetization is detectable, since the magnetic change may induce a current in an appropriately positioned reveicer coil. The temporal signal represented by this current is called the free induction decay (FID).

The relaxation is approximately exponential, but longitudinal and transversal relaxation need not happen at the same rate. This means the compass needle analogy breaks down, since the length of the needle, corresponding to magnetization strength, may change during relaxation. The longitudinal relaxation time constant is denoted by T_1 and its transversal counterpart is denoted by T_2. Different tissue types have (slightly) different time constants – in actuality,

these time constant differences are often what is being depicted. MR images emphasizing differences in T_1 are known as T_1-*weighted* images, whereas those differentiating tissue types by their T_2 time constants are known as T_2-*weighted* images.

When the ensembles have been subject to the above static magnetic field (known as the main field) for some time, a state of equilibrium is reached and the ensembles no longer exhibit any measurable precession. However, by applying a transverse magnetic field pulsating at the Larmor frequency, it is possible to force the ensembles out of this equilibrium. This process is known as *magnetic resonance*, and the applied field is known as the excitation, or radiofrequency (RF) field, as the Larmor frequencies are usually within the RF range. By applying a third kind of magnetic field, a field that varies with position, it is possible to encode positional information into the received signal (FID), thus enabling the transformation from FID to MR image. This third kind of magnetic field is known as a *gradient* field.

However, applying such an inhomogeneous magnetic field leads to phase dispersion, or destructive (neutralizing) interference, which causes the FID to deteriorate. One way of reducing this problem is by using *gradient echoes*. Assume that all ensembles ("compass needles") are in-phase at time $t = 0$. Further assume that a gradient field is applied at time $t = 0$ in addition to a main field. At the time $t = \tau$, the gradient field has caused some phase dispersion. Assume that the gradient field is somehow reversed at time $t = \tau$. This means that the ensembles that lost phase when compared to ensembles unaffected by the gradient field now gain phase, and vice versa. At time $t = 2\tau$, the ensembles have rephased, causing a so-called *gradient echo*. By sampling the FID close to this echo, the signal-weakening dephasing caused by the gradient fields will be reduced.

For a thorough treatment of the subject of magnetic resonance imaging, refer to [Nish94]. For a treatment in Danish, including a brief description of the MR phenomenon based on quantum theory, see [Wolf01].

2.1.1 Scanner output

The usual MRI sequence used for identifying cortical dysplasia is called MP-RAGE (see [Bran92]), which stands for *Magnetization Prepared Rapid Gradient Echo*. This is the gradient echo technique described above, preceded by an initial $180°$ excitation performed to further enhance the T_1 dependency of the image. This is an MR sequence producing a three-dimensional T_1-weighted image with (typically) near-isotropic voxels. In these images, the main components (see section 2.3) of the brain are[Edel96]:

White matter:	Bright
Gray matter:	Dark
Cerebrospinal fluid:	Very dark

This imaging sequence gives good brain tissue contrast while at the same time offering bearable scan times ($\approx$ 15 minutes).

2.1.2 Multi-Planar Reformatting

Multi-Planar Reformatting (MPR) is an imaging method that allows the radiologist to slice through the 3D image and view the resulting image in three orthogonal planes. This helps overcome the lack of a third dimension on computer screens and may improve the interpretation of the data, i.e. by providing three orthogonal sectional views each centered on suspicious tissue. The data is usually a volume of near-isotropic microliter voxels (approximately $1 \times 1 \times 1$ mm^3), often from an MP-RAGE scan. Figure 2.1 shows a set of corresponding orthogonal images.

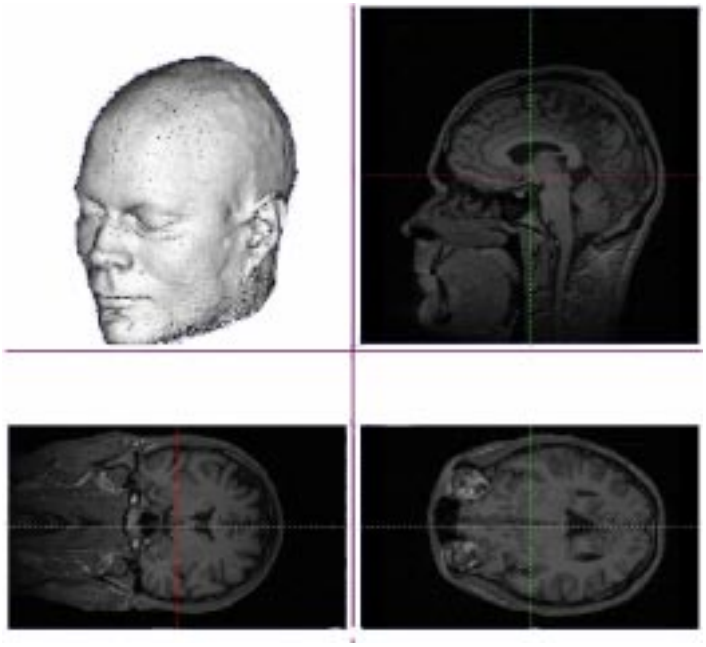

Figure 2.1: *Multi-planar reformatting and a three-dimensional representation of a human head. Upper right: Sagittal. Lower left: Coronal. Lower right: Axial.*

2.2 On Rendering, Image Processing, and Image Analysis

The following sections briefly describe some of the basic concepts of rendering, image processing and analysis, as well as the conventions used in the following chapters.

2.2.1 Texture Mapping

As will become clear, rapid display of two-dimensional images is an important feature of the prototype in which the developed method is to be embedded (see chapter 4). To make this prototype portable, a graphics library called OpenGL, which is available across many platforms, is used. The recommended way of

Figure 2.2: *This image shows what seems to be a large lump of gray matter – see also figure 2.3.*

interactively displaying a picture in a quick and portable way using OpenGL on consumer hardware, using the picture as a texture map, is to perform a 1:1 mapping onto an appropriate quadrilateral. This texture mapping approach gives the added advantage of being able to perform affine warps of the image effortlessly, as will be shown in section 6.2.

2.2.2 Coordinate Systems

To allow for arbitrary orientation of a three-dimensional object (a three-dimensional MR brain image) in a graphics display, two left-handed three-dimensional coordinate systems are defined: a world, or object coordinate system and a camera, or view coordinate system. The viewing transformation is given by a 4×4-matrix, transforming a point in object space into the corresponding point in view space by right multiplication. The points are represented using homogeneous coordinates, so the viewing transformation of a point is given by

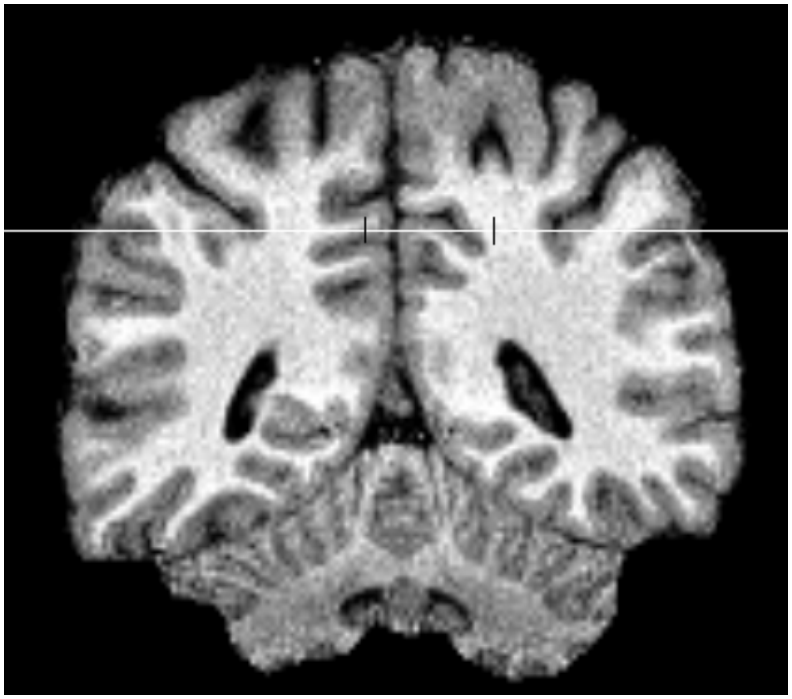

Figure 2.3: *This image, indicating the image plane of the image in figure 2.2 as a horizontal white line, shows that the apparent cortical thickening is an artifact caused by the local cortical geometry being near-parallel to the image plane.*

$$
\begin{array}{ccc}
v_{view} & = & M_{view} \cdot v_{object} \\
 & \Updownarrow & \\
\begin{bmatrix} x_v \\ y_v \\ z_v \\ w_v \end{bmatrix} & = & \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}
\end{array}
\tag{2.2}
$$

The viewing transformation, M_{view}, includes a projection (here, a parallel projection is employed) such that the image plane coordinates of a point $p_v = [x_v, y_v, z_v, 1]^T$ will be determined as $p_i = [x_v, y_v]^T$.

The positive directions of the x, y, and z camera axes correspond to the directions right, up, and forward (into the screen), respectively, forming a left-handed coordinate system. The same goes for the object space axes in the initial configuration, i.e. when M_{view} is equal to the identity matrix.

The first three columns of the viewing matrix of equation (2.2) are equal to the world space coordinates of the camera space axes (x, y, and z, respectively). The fourth column contains information on the translation of the origin of camera space with respect to the origin of world space.

2.2.3 On Discrete Regular Grids

The data obtained from an MR scanner sequence suitable for MPR is normally a set of near-isotropic voxels on a regular grid. A number of metrics lend themselves to distance determination in such a grid. In this thesis, however, the employed metric is the Euclidean distance between voxel centers. Other metrics and their impacts on the developed method will be discussed in section 7.3. The work presented here will assume isotropic voxels – the case of non-isotropic voxels will be discussed in section 8.5.1.

2.2.4 Gradient Fields

To an intensity field, such as a three-dimensional volume of intensities output by an MR scanner, can be associated a so-called *gradient field*[1], which is a vector field displaying the rate of change of the intensity field, each vector showing the direction of maximum change.

To represent the digital approximation of the gradient, the symbol ∇ (*nabla*), conventionally used to denote the gradient of a continuous field, is used.

As a one-dimensional example, determining the gradient along a line of pixels with no attention paid to scaling, one could use the following expression to approximate the gradient of the n'th pixel ∇_n, using the pixel intensities i_n:

$$\nabla_n = i_{n+1} - i_{n-1} \tag{2.3}$$

Note, that the n'th element of the vector of intensities $[i_1, i_2, i_3, \ldots]$ convolved with the vector $[1, 0, -1]$ leads to the same result.

In two dimensions, the gradient is a vector of two elements. The first element, representing the horizontal component of the gradient, may be derived in the exact same way as the one-dimensional case, i.e. by convolving each horizontal scanline. The vertical component can be derived in a similar fashion, namely by convolving each vertical scanline by $[1, 0, -1]^T$. Since this is only an approximation of the gradient, other filter masks may provide a better estimate. An alternative set of filter masks for 2D images are:

$$M_{horiz} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_{vert} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.4}$$

This set of masks estimates the gradient components using a larger neighbourhood but puts emphasis on nearer pixels. Note, that the masks have been rotated by 180 degrees when compared to the vectors above. This is due to the reflection between the filter mask and the point spread function: if $h(k, l)$ is the point spread function, the filter mask elements (called filter weights) are usually given as $h(-k, -l)$.

The three-dimensional filter masks used for gradient approximation in this thesis are appropriately reoriented versions of the mask shown in figure 2.4.

For further reading, refer to [Nibl86].

[1] Not to be confused with the gradient fields appearing in magnetic resonance.

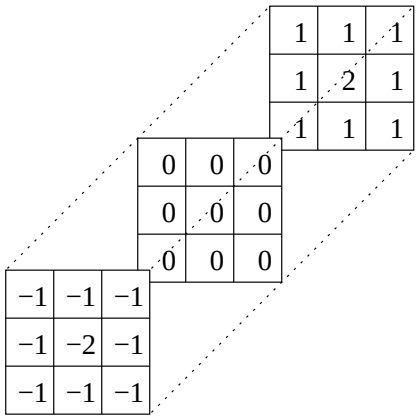

Figure 2.4: *The filter mask used for three-dimensional gradient approximation.*

2.2.5 Resampling

The program offers a choice between two ways of determining the voxel intensity at a given position: Nearest-neighbour filtering or trilinear interpolation filtering. The following example illustrates the problem and a few possible solutions.

Example 2.1: One-dimensional resampling. Assume that the values of a function $f(x)$ are sampled at integer values of x. What would be a reasonable value for, say, $f(1.4)$? Using the nearest-neighbour resampling filter, the answer would be $\tilde{f}_{nn}(1.4) = f(1)$, the value of the nearest neighbour. Using a linear resampling filter, however, the answer would be a linearly weighted average of the nearest values, i.e. a point corresponding to $x = 1.4$ on the straight line between $(1, f(1))$ and $(2, f(2))$:

$$\tilde{f}_{lin}(1.4) = (2 - 1.4)f(1) + (1.4 - 1)f(2) \tag{2.5}$$

Refer to figure 2.5.

The nearest-neighbour resampling filter is cheaper in terms of computing time, whereas the linear interpolation filter is considerably less “blocky”, since the output of the filtered function $\tilde{f}_{lin}(x)$ is continuous.

Both the nearest-neighbour resampling filter and the linear interpolation resampling filter may be extended to higher dimensions. The latter is known as bilinear interpolation in two dimensions (section 6.1) and trilinear interpolation (section 5.1.1) in three dimensions.

Non-linear resampling is another possibility, with sinc interpolation, as well as quadratic and cubic interpolation resampling being the most popular choices. A more thorough introduction to resampling may be found in [Nibl86].

2.3 On the Human Brain

The human brain consists roughly of two kinds of tissue arranged in three layers. In the depth close to the midline, a core of gray matter called the basal

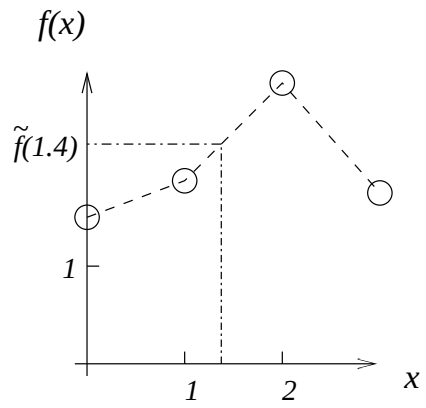

Figure 2.5: *Linear interpolation between integer sample points of the function f. Refer to example 2.1.*

ganglia is found. The basal stuctures are surrounded by a relatively thick layer of white matter, and this is covered by an approximately 3 mm thick cortex. During the development of the cerebral cortex, which mainly takes place from the seventh week after conception to birth, a number of things may go wrong. One of the possible malformations is called cortical dysplasia, to which some of the associated MR imaging findings can be seen in table 2.1.

1. Cortical thickening.
2. Blurred gray / white matter junction, i.e. blurred transition between gray matter and white matter.
3. Brain deformation.
4. Heterotopia (gray matter misplaced inside white matter).

Table 2.1: *Typical MR imaging findings in dysplastic brains.*

2.3.1 Cortical Dysplasia

Cortical dysplasia associated with seizure disorders are often divided into three categories[Edel96]:

- Focal lesions, in which the amount of abnormal tissue is relatively small and localized.
- Unilateral disorders, roughly affecting one entire cerebral hemisphere.
- Generalized, or bilateral, disorders, affecting the entire brain.

The extent of the abnormal tissue in the unilateral and generalized cases make them susceptible to easy detection by visual inspection of MR images, whereas the focal lesions may be subtle, and thus more difficult to detect. The central point in this thesis is the development of a set of tools to aid a trained neuroradiologist in locating subtle focal dysplastic lesions of Taylor

type[Tayl71]. This type of disorder is characterized by localized cortical thickening and poor gray / white matter differentiation, item 1 and 2 of table 2.1, and is believed to be among the most epileptogenic lesions associated with epilepsy.

The localized cortical thickening makes the cortex of the affected region look like a carpet with a rock underneath. The size of the rock obviously has a great impact on the detectability of the disorder. Mild cortical disorganization may be undetectable by MRI, since the affected region may be smaller than the resolution of the MR image. Section 2.4 describes the currently used techniques for detecting subtle cortical dysplasias of Taylor.

2.4 Detection of Dysplastic Lesions

Cortical development disorders of any of the above categories may be detected using conventional MPR techniques. However, in recent years, work has been done to aid in the detection of the subtler focal lesions. Various forms of curvilinear reformatting (CR), such as the one described in chapter 7 of this thesis, as well as methods such as voxel-based morphometry, described in the section 3.4, have been proposed.

2.5 Consequences

The possibility of detection of a focal cortical dysplasia in a patient with epilepsy may take the patient from the category of life-long medication and even drug resistance to the category of possible surgical candidates. If other neurophysiological investigations (e.g. electro-encephalography, EEG) points to exactly the same region and to no other regions, the prognosis for surgical intervention is very good.

2.6 Important Names and Concepts

Axial	Slice orientation. Bottom view. See figure 2.1.
Basal ganglia	A structure of gray matter forming the core of the brain.
Coronal	Slice orientation. Rear view. See figure 2.1.
Cerebral cortex	The exterior substance of the brain. Consists of gray matter.
Cerebro-spinal fluid, CSF	A serous fluid secreted by the membranes covering the brain and spinal cord. The brain "floats" in CSF.
Dysplasia	Abnormal development (of organs or cells) or an abnormal structure resulting from such growth.
Electro-encephalogram, EEG	A graphical record of electrical activity of the brain
Frontal lobe	The part of the cerebral cortex in either hemisphere of the brain lying directly behind the forehead.
Gray matter	Grayish nervous tissue forming the cerebral cortex and the basal ganglia.
Gyrus, pl. *gyri*	The convoluted ridges between the sulci.
Sagittal	Slice orientation. Side view. See figure 2.1.
Sulcus, pl. *sulci*	The fissures and grooves in the cerebral cortex.
Temporal lobe	The part of the cerebral cortex in either hemisphere of the brain lying inside the temples of the head.
White matter	Whitish nervous tissue forming a rather thick layer on the basal ganglia.

Chapter 3

Prior Work

3.1 Curvilinear Reformatting

As a means of overcoming the lack of a third dimension inherent in computer screens and regular MR images, it is possible to perform a virtual incision in the MR image that allows for inspection along a given cutting surface. These surfaces are most often planar (like those of MPR, see section 2.1.2), but may also be curvilinear, although typically limited in shape to an extrusion of a manually drawn planar curve. The extraction of curvilinear surfaces is called curvilinear reformatting.

Bastos et al. [Bast99] presented a new way of extracting curvilinear slices. This method allows the radiologist to examine the three-dimensional brain image using a series of thin slices curved along the hemispheric convexities of the brain, much like the layers of an onion. As opposed to the planar slices offered by conventional MPR, this may significantly reduce the impression of cortical thickening caused by obliquity of the plane of section in relation to the gyri. Figure 3.1 shows how curvilinear reformatting obtains a parallel incidence of the slice in relation to the gyral cortex. This reduces the artifactual cortical thickness shown in figures 2.2 and 2.3.

The method of Bastos et al. requires a neuroradiologist to manually delineate the brain surface. Subsequently, the software generates a set of 3D polygonal models including the corresponding texture maps. Following this, the actual examination takes place, during which the radiologist can display and manipulate the previously generated curvilinear[1] data, as well as planar slices generated on-the-fly[2].

3.2 Shear-Warp Rendering

The title of this section is shorthand for volumetric rendering using a shear-warp factorization of the viewing transformation.

This factorization can be written

[1] Since the generated data consists of polygonal models, it is in actuality not curvilinear.

[2] This information was acquired through personal correspondence with Mr. Roch Comeau.

Figure 3.1: *Coronal diagram showing cortex (gray) and the plane of a curved slice (black bold). Notice how the slice intersects the cortex at near-right angles.*

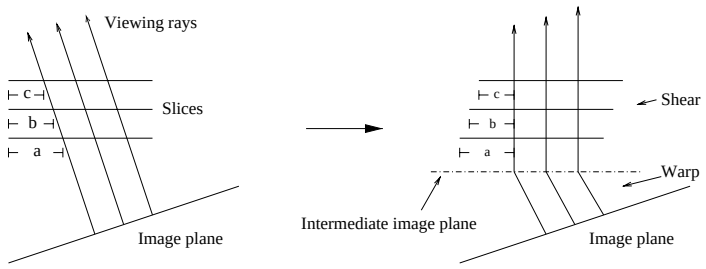

Figure 3.2: *Ray casting using the shear-warp factorization.*

$$M_{view} = M_{warp} \cdot M_{shear} \tag{3.1}$$

The idea behind this technique is to divide the rendering process in a manner similar to the factorization:

1. Perform a 3D shear parallel to the data slices such that the viewing rays become parallel to each other and perpendicular to the slices (see figure 3.2).
2. Project the sheared slices to form an intermediate but distorted image.
3. Perform a 2D warp to form an undistorted image.

The advantage of this factorization is that rows of voxels in the volume are aligned with rows of pixels in the intermediate image. This allows the volume and the intermediate image to be traversed in synchrony, which in itself leads to rapid rendering, but also enables the exploitation of spatial coherence

in both the volume and the intermediate image, which leads to even greater rendering speed.

3.3 Other Render Methods

Current alternatives to volume rendering using the shear-warp transformation in terms of image quality and render speed are manifold. However, a few typical directions of research may be deduced:

- Using parallel computer systems or specialized graphics hardware ([Came92], [Fuch90]).
- Exploiting special features (such as multitexturing, 3D textures, register combiners) of emerging consumer graphics hardware (see e.g. [Enge01]).
- Accelerating ray casting techniques, e.g. using templates ([Lee97]).

The current generation of consumer graphics hardware have accelerated support for volumetric textures, but the texture size is limited, often to $64 \times 64 \times 64$ elements (corresponding to 1 Mb of graphics memory), whereas 2D textures are allowed to use 16 Mb. This indicates that the limitation of volumetric textures lies not in memory consumption, but is a hardware or driver issue. If the incentive is there for the graphics hardware manufacturers, the support for large ($256 \times 256 \times 256$ elements or more) volumetric textures is only a hardware generation away. At the current rate of progress, rendering of high-resolution volumetric images at smooth interactive rates on consumer hardware is only a year or so away.

3.4 Voxel-Based Morphometry

One way of performing a voxel-wise comparison of the cortical structure between two groups of subjects is called voxel-based morphometry. The idea is to compare the cortical structure of a patient with that of a statistical model of a normal cortex, obtained by correlating the information from a database of healthy cortices.
At the 2001 meeting of the Organization for Human Brain Mapping a team from the University of Freiburg presented a method making use of voxel-based morphometry to locate focal cortical dysplasia[Hupp01]. The method is based on gray matter probability images, generated by low-pass filtering the outcome of a cortical gray matter segmentation of a three-dimensional MR image. A normal data base was created from gray matter probability images of 30 healthy volunteers. The examination of a patient's MR image would then consist of the following steps:

1. Normalize the MR image into a standard position and orientation (into standard stereotactic space).
2. Segment cortical gray matter. In other words, discard everything but the gray matter of the cortex.
3. Smooth the gray matter segments. The result of this smoothing is proportional to an approximate cortical gray matter probability image.

4. Calculate, voxel by voxel, the difference between this probability image and the mean probability image of the normal data base. The local maxima of this difference image indicates where the patient's cortex differs the most from that of the normal data base, thus indicating where dysplasias are to be expected.

The considerable variability of geometry within the range of healthy brains detracts from the overall virtue of the method. Furthermore, with the segmentation and low-pass filtering involved, the method runs the risk of eradicating signs of subtle dysplastic lesions. However, according to the abstract, the method "seems to provide a valuable additional tool for the detection of focal cortical dysplasias", and does show promising results.

For more information on voxel-based morphometry, see [Ashb00].

Chapter 4

Purpose

The ideal cortical dysplastic lesion detection tool would take a three-dimensional MR image, process it, and find out whether or not a dysplastic lesion was present in the cortex, and, in case a lesion was found, pass on information on the location of the lesion directly to the neurosurgeon. For several reasons this is not a feasible solution: subtle focal lesions may be invisible on the relatively coarse MR images[Edel96], and thus undetectable regardless of the method used; ethical considerations require a human to verify alleged lesions before any operative procedures be commenced, and to examine the brain in case the employed tool failed to detect a lesion.

The purpose of this work has been to develop a method of automatically detecting focal dysplastic lesions, and also to create a working prototype. In keeping with the above requirement while also allowing for general verification of the developed method, such a prototype needs to allow for verification by visual inspection. Therefore, a substantial amount of this thesis deals with aspects of graphics and visualization, as already apparent in chapter 3.

This chapter aims at giving the reader an overview of the subjects that need to be scrutinized in order to implement a working prototype. In the rest of the thesis, this prototype will be referred to as *the program* or *the prototype*.

4.1 Detection

Large dysplastic lesions are typically easily detected using conventional MPR methods. Subtle lesions, however, may be very difficult to detect – the proverbial needle in a haystack would be an appropriate analogy. Therefore the focus should be on developing a method suitable for finding subtle dysplastic lesions. To avoid the need for re-scanning patients, the detection method should operate on conventional three-dimensional MPR data, such as the result of scanning using the MP-RAGE sequence. Furthermore, the techniques used on the data must either be simple (and therefore quick) or require no user input, in order to avoid wasting radiologist resources. A method that requires no user input is equivalent to a fully automatic detection system, which would be infeasibly difficult to implement owing to the subtlety of the dysplastic lesions sought. The proposed method should aid the radiologist – it is by no means intended to replace the efficient image analysis tool that is the human eyes and

brain.

The following sections describe the assumptions and constraints which constitute the frame in which the proposed method has been developed.

4.1.1 Assumptions

Throughout this thesis the following assumptions have been made:

- It is assumed that each voxel corresponds to a fixed-size cubic (and thus isotropic) volume in world space. The case of non-isotropic voxels is described in section 8.5.1.
- Tissue time constants T_1 and T_2 remain constant over all of the scan volume. Ignoring magnetic field inhomogeneities, this means that all voxels corresponding to e.g. white matter have the same intensity independent of position. The effects of magnetic field inhomogeneities are discussed in section 10.2.
- It is assumed that radiologists hold the correct answer, and are thus able to determine whether or not a particular cluster of voxels signify abnormal tissue. This thesis will not consider verification of imaging findings by post-operative histology, nor will it discuss subjects such as: what type of MR images offers the most advantages in terms of detection (be it manual or automatic) of cortical dysplasia ?

4.1.2 Constraints

In order to complete the work within the time allotted, a number of constraints have been posited:

1. Any voxel-wise classification of brain tissue must be performed by a trained neuroradiologist. However, automatic brain / non-brain tissue classification is permissible.
2. Constraint number 1 leads to the fact that any decision making based on global knowledge of brain geometry, besides location and basic shape must also be left to a radiologist.
3. Out of the typical imaging findings (see table 2.1) the proposed method only attempts to detect *cortical thickening* and *blurred gray matter / white matter junction*. The remaining types of findings are either easily detectable by a radiologist or require some degree of knowledge of the geometry of the brain, which renders them outside the scope of this work.
4. Owing to the growth and development of the brain during the first few years of life, MR images of infant brains may look very different from those of adults. The developed method focuses on adult brains, but may in fact turn out to work on infants' brains as well.

Ad 1: The reasoning behind this constraint may not be obvious, but since the gray matter / white matter junction of dysplastic lesions may be indistinct, automatic classification is likely to err exactly at these positions of maximum importance.

4.2 Visualization

To verify that the prototype functions as supposed and in keeping with the aforementioned ethical considerations, some degree of data visualization is necessary. The visualization method attempts to make use of the experience the radiologist might have gained using conventional planar reformatting, and also to exploit recent advances in visual detection of dysplasia.

This has lead to a design featuring both an image of a planar slice and an image of a three-dimensional model of the brain; a design which necessitated the application of two different rendering methods. The two-dimensional image displays planar slices while the three-dimensional image displays curvilinear slices as well as indicating the orientation of the planar slice shown in the two-dimensional image.

This necessity of implementing two different ways of rendering is part of the reason why the visualization takes up a large part of the rest of the thesis.

The diagram in figure 4.1 shows the general design of the prototype.

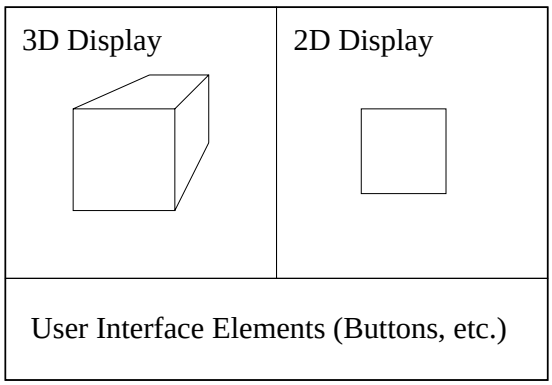

Figure 4.1: *Prototype graphical user interface layout showing a cube*

4.3 An Overview of this Thesis

The following chapters of this thesis describes the development of the detection method and the practical implementation of prototype. Following these chapters are two chapters demonstrating and discussing the results of the work.

- **Two-dimensional display** describes the planar slice extraction and resampling as well as the actual slice display.
- **Three-dimensional rendering** describes the mechanisms of shear-warp rendering as well as additions for direct rendering and correct intersection with other objects.
- **Curvilinear reformatting** describes the automatic extraction of curvilinear slices.
- **Dysplastic lesion detection aid** describes the crux of this thesis: Automatic detection of cortical dysplasia.

- **Prototype implementation** contains a description of the implementation of the prototype in more detail.
- **Results** contains examples of the capacity and limitations of the developed method.
- **Summary** contains a concluding examination of the presented work, acting as a recapitulation of the thesis.

Chapter 5

Two-Dimensional Display

This chapter describes the way the two-dimensional display works. Most of the content of this chapter is standard, but it is included to provide a full understanding of how the program works.

The part of the program performing two-dimensional display of planar slices can be said to work in two steps:

1. Render the appropriate slice into an intermediate image, applying resampling as chosen by the user.
2. Use this intermediate image as a texture map, and map it onto an appropriate quadrilateral.

5.1 Rendering the intermediate image

Assuming that the intermediate image is w pixels wide and h pixels high, the location of the slice plane is determined by the point $\mathbf{c}$ and two perpendicular vectors (of equal magnitude) $\mathbf{a}$ and $\mathbf{b}$:

for $v = 1$ **to** h
  for $u = 1$ **to** w
    $\mathbf{p} = \mathbf{c} + \left(u - \frac{w}{2}\right)\mathbf{b} + \left(v - \frac{h}{2}\right)\mathbf{a}$
    if $\mathbf{p}$ inside object **then**
      $\mathrm{Image}(u, v) = \mathit{Determine\_Pixel\_Intensity}(\mathbf{p})$

From the pseudocode above we see that the point $\mathbf{c}$ gets mapped to image coordinates $(u, v) = \left(\frac{w}{2}, \frac{h}{2}\right)$, right in the middle of the image. We also see, that $\mathbf{a}$ is vertical in the image plane, whereas $\mathbf{b}$ is horizontal. If $\mathbf{a}$ and $\mathbf{b}$ are not perpendicular, the intermediate image will be spatially distorted when transformed from object space to image space, which, in general, is undesirable. The same goes for the situation where $\mathbf{a}$ and $\mathbf{b}$ are not of equal magnitude[1]. Simultaneously changing the magnitude of $\mathbf{a}$ and $\mathbf{b}$ corresponds to a change of scale. Thus, a "zoom" effect can be achieved this way.

The center point $\mathbf{c}$ and the vectors $\mathbf{a}$ and $\mathbf{b}$ are in fact all derived from a viewing transformation matrix. In homogeneous coordinates, the center is set

[1] However, this effect could be used to compensate for non-square pixels.

equal to the fourth column of the viewing matrix. Since this column is equal to the coordinate translation vector, this makes sense (refer to section 2.2.2). The vector $\mathbf{a}$, corresponding to the vertical in the intermediate image, is equal to the second column, which corresponds to the y-axis, which is defined to be vertical in the configuration position. Using the same arguments, we set $\mathbf{b}$ equal to the first column of the viewing matrix, corresponding to the x-axis.

The function $Determine\_Pixel\_Intensity(\mathbf{p})$ is supposed to determine the pixel intensity corresponding to the intensity of the scanned object at the position indicated by the vector $\mathbf{p}$. This task may be divided into two parts:

1. Determine the voxel intensity at the position indicated by $\mathbf{p}$.
2. Determine the pixel intensity corresponding to this voxel intensity.

These will be described in the following two sections.

5.1.1 Intensity Determination

To approximate the intensity at a given position $\mathbf{p}$, the program offers a choice of two methods: Nearest-neighbour resampling or trilinear interpolation (refer to section 2.2.5). The former offers speed, the latter smoother images, particularly in magnification. Figure 5.1 shows the difference between these resampling methods. Note, that the underlying resolution is the same; the rightmost image does not contain more information than its neighbour. The rightmost image basically trades its “blockiness” for “blurryness”.

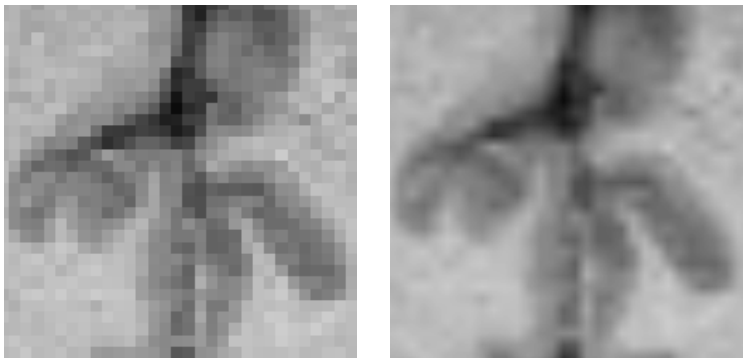

Figure 5.1: *Closeup of a slice resampled using nearest-neighbour filtering (left) and a slice resampled using trilinear interpolation (right).*

5.1.2 Intensity Mapping

In order to maximize visual contrast between tissue types, the program allows the user to adjust the parameters of an affine mapping of voxel intensities to pixel intensities. The map is a simple windowed affine transformation: Assuming a minimum image pixel intensity (black) of zero and a maximum image pixel intensity (white) of one, the mapping is described by two user-selectable

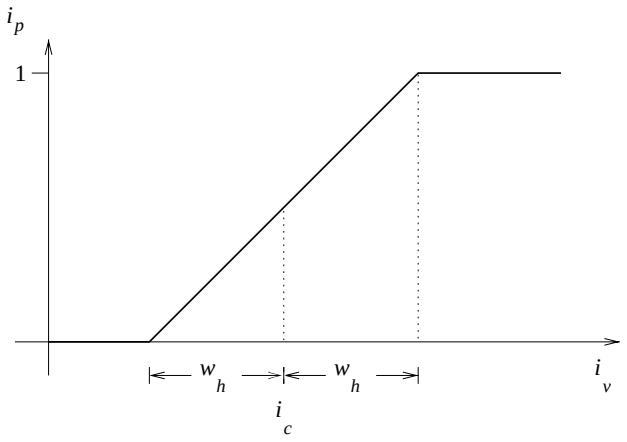

Figure 5.2: *The windowed affine intensity mapping with user-defined parameters w_h and i_c.*

variables. The intensity corresponding to the center of the window is denoted by i_c and half the window width is denoted by w_h, see figure 5.2.

$$i_p = \begin{cases} 0 & i_v \leq i_c - w_h \\ \frac{i_v - i_c}{2\,w_h} + \frac{1}{2} & i_c - w_h < i_v < i_c + w_h \\ 1 & i_v \geq i_c + w_h \end{cases} \tag{5.1}$$

5.1.3 A Slight Improvement

Since, in general, the viewing ray corresponding to every single pixel of the intermediate image does not intersect the scan volume, computational cost may be reduced somewhat by ignoring those pixels whose viewing rays do not intersect the scan volume. Processing each pixel to see if its viewing ray intersects the scan volume is obviously not a feasible solution, since this check is tantamount to the normal ray-casting step performed. The determination of some of these "superfluous" pixels may be performed by taking into consideration the fact that the scan volume is a rectangular parallelepiped, i.e. a convex volume. This has the as a consequence that if the vector variable $\mathbf{p}$ of the above algorithm should happen to leave the volume at any point while traversing a scanline of the intermediate image, it will never return to the volume while on this particular scanline. In other words, once we leave the volume while traversing a scanline, we can skip the rest of that scanline. Refer to figure 5.3.

On the other hand, this necessitates an initial clearing of the intermediate image, since the skipped pixels are not assigned a value. In practice, however, this constitutes an overall improvement of the speed of the extraction.

The following pseudocode uses the boolean variable $\mathrm{inside\_object}$ to indicate whether any of the points corresponding to the current scanline has been inside the object.

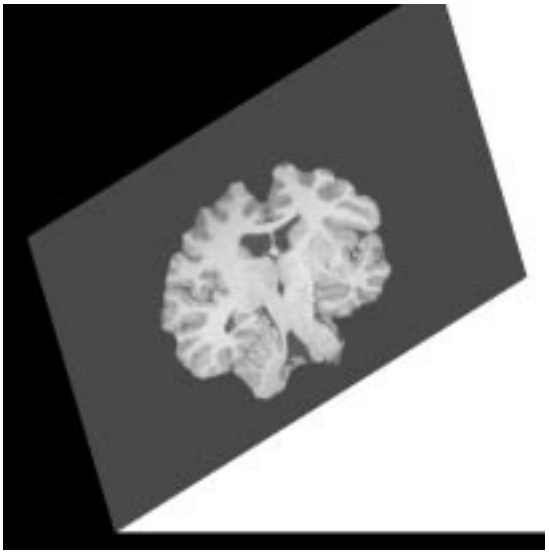

Figure 5.3: *Pixel skipping. This extracted intermediate image shows processed pixels outside the scan volume (black), and inside the scan volume (shades of gray) as well as skipped pixels (white).*

```
inside_object = false
for v = 1 to h
    for u = 1 to w
        p = c + (u − w/2) b + (v − h/2) a
        if (p is inside the object) then
            inside_object = true
            Image(u, v) = Determine_Pixel_Intensity(p)
        else
            if (entered_volume = true) then
                inside_object = false
                Skip to next scanline
```

5.2 Applying the texture map

After extraction of the slice, the intermediate image needs to be registered as a texture to the graphics hardware. Following this, the texture map needs only be mapped to a quadrilateral. Ideally, this should be a 1:1 mapping, but since the user is allowed to resize the window (thus resizing the quadrilateral onto which the intermediate image is to be mapped), this is not always the case. This non-ideal mapping might lead to visual artifacts such as distinct pixelization, but since OpenGL is able to filter the texture appropriately (thus reducing artifacts to a minimum), this does not pose a problem.

5.3 Possible improvements

It would be faster, and possible without significant loss of image quality, to let the magnitude of $\mathbf{a}$ and $\mathbf{b}$ remain at unity (thus always performing slice extraction on a 1:1 scale which means zooming will not reduce the number of skipped pixels) and then performing any zooming by enlarging the quadrilateral onto which the two-dimensional intermediate image is mapped. The advantage of this method would be that is transfers the work of zooming from the CPU and onto the graphics hardware, thus freeing the CPU for other work. Obviously, for this to be an actual advantage, texture mapping hardware is required. Using the appropriate texture filtering technique (nearest-neighbour or linear interpolation, depending on the slice resampling filter) the image quality loss would be negligible.

Another improvement would be to let the program use nearest-neighbour resampling while the user is interacting with the object and then render the graphics using trilinear interpolation filtering when the interaction stops. This approach will achieve good rendering speed during interaction and good image quality of static images.

Chapter 6

Three-Dimensional Rendering

One of the problems of rendering two-dimensional images of three-dimensional objects is the sheer amount of data processing needed. One way of reducing the render time is commonly known as shear-warp rendering, briefly described in section 3.2. However, shear-warp rendering does just that: rendering. It leaves no information for the graphics library on how to handle the interaction of the shear-warp object and other graphics objects. Let us assume that we want to indicate with a polygon a certain planar slice of the three-dimensional object in question. Shear-warp rendering, by itself, generates no information on how to handle the intersection of the polygon and the shear-warp object.

This chapter describes the implementation of three-dimensional rendering using a shear-warp factorization of the viewing transformation, along with the developed modifications for interactive display. Furthermore, a flexible way of combining the aforementioned voxel-based rendering with conventional polygonal rendering is presented.

6.1 Shear-Warp Rendering

In order to achieve good performance on modern computers, it is important to keep the data used in the fastest memory type possible. Since it is rarely possible to keep all data in CPU registers alone, as much as possible should be kept in the fast data cache memory. This means focusing on small coherent portions of memory instead of making spurious access of data at locations spread all over the memory at random. To achieve good cache coherency, an ideal volume rendering method must perform in-order traversals of both volume and (screen) image data. Klein and Kübler [Klei85] proposed a method in which the view transformation matrix is factorized into a shear matrix and a warp matrix. Using this factorization, rendering may be performed in four steps:

1. Choose the appropriate slicing direction. It is assumed that the volume data may be sliced in three directions, each perpendicular to one of the world space axes. In other words the voxels within each slice all have

equal x-, y-, or z-coordinates. Choose the slicing direction such that the corresponding set of slices is most perpendicular to the viewing direction. The axis corresponding to this slicing direction is said to be the *principal axis*.

2. Shear (translate) each slice such that a viewing ray perpendicular to the set of slices in the sheared object space will intersect each slice in the same manner as the oblique viewing ray would intersect the slices in untransformed object space (see figure 3.2). This means, that in the sheared object space, all viewing rays are parallel to the principal axis. To perform a perspective transform, slices must be scaled as well as sheared.
3. Project the slices into a distorted 2D intermediate image in sheared object space. Since the shear coefficients are not confined to integers, proper resampling is necessary to retain image quality.
4. Warp the intermediate image into image space, producing the correct final image.

The process of transforming the intermediate image into image space requires a general warp operation. However, since this is a 2D operation, the cost is limited (about 10% of the total cost in typical cases, according to [Lacr95]).

Note, that the resampling used during the projection takes place in-slice. This potential problem of using a 2D rather than a 3D reconstruction filter to resample the volume data is discussed in section 6.4. The prototype allows resampling using the 2D nearest-neighbour resampling filter or a bilinear filter, the latter being slightly slower owing to the fact that two voxel scanlines must be traversed simultaneously.

Assuming the volume consists of fully transparent and fully opaque voxels only, the projection could be done using an approach akin to the painters algorithm: Projecting one sheared slice at a time in back-to-front order, new opaque voxels overwriting earlier (and thus more distant) voxels. Obviously, this would lead to a lot of overdraw in the typical case. It turns out to be advantageous to project the slices front-to-back, ignoring any intermediate image pixel once an opaque voxel has been projected onto it.

The factorization of the viewing transformation leads to a general warp and a simple shear operation. The advantage of this is the possibility of traversing each intermediate image scanline and each object volume scanline in synchrony, leading to maximum cache coherency, which in turn means maximum cache efficiency.

Note, that while it is possible to do each of the above steps “by hand” (on the CPU), it is likewise possible to make use of the graphics hardware through OpenGL in the final 2D warping stage of the process, thus relieving the CPU of this task. See section 6.2.

Since there are six principal viewing directions, two for each axis, it is possible to achieve in-order object traversal by keeping one copy of the data volume corresponding to each of these directions. However, relaxing the whole-object in-order traversal constraint to in-order traversal of each slice, it is possible to use only half the amount of memory at a minute cost. This way, all slices are traversed in-order, but the order in which the slices are traversed may change to accommodate the front-to-back projection order (e.g. when the viewer is “behind” the object). Therefore, three copies of the volume are needed.

6.1.1 Spatial Coherency

Apart from the in-order traversal of both image space and object space, what sets the shear-warp rendering method apart from other volumetric rendering methods is its ability to exploit spatial coherency in both image space and object space. Images of human brains contain one coherent object, namely the brain itself. This object space coherency leads to image space coherency.

One way of exploiting object space coherency would be to store voxel scanlines as runs of voxels, e.g. “10 transparent voxels”, then “13 opaque voxels”. This is called *run-length encoding*[Fole97]. If object space coherency data is stored along with the object data in this way, the run-time transparency checking of each individual voxel is superfluous, since transparent voxels may be ignored once and for all. Since typically 80% of the voxels of an MPR image are transparent[1], this constitutes significant savings in terms of run-time data processing as well as storage space, since three run-length encoded copies are usually smaller than a single unprocessed volume.

However, preprocessing data by run-length encoding imposes severe limitations on the extent to which data may be changed on the fly (see section 6.3.6), since e.g. inserting an opaque pixel in a transparent run (thereby breaking up the run) will necessitate a re-encoding of the volume.

Image space coherency may be exploited in a similar way. Recalling that, once a pixel has been set (i.e. once it has turned opaque) in the intermediate image, it will not be changed again, it is possible to achieve a considerable speed increase by keeping track of runs of opaque pixels in the intermediate image. Assume that each opaque pixel is associated with a value that contains the length of the rest of the current run. This way, it is possible to skip whole runs of opaque pixels as well as the corresponding object voxel runs, since image space and object space are traversed in synchrony. However, when the first transparent pixel after an opaque run turns opaque, the length values of the whole run must be updated (since the length of the run just increased). In [Lacr95] it is shown, that *path compression*[Corm97] is a fast way of approximating this update. Path compression is also the technique used in the prototype described in this thesis.

6.1.2 The Shear-Warp Factorization

Assuming the z-axis is the principal axis (see section 6.1), the factorization may be written[Lacr95]:

$$M_{view} = M_{warp} \cdot M_{shear} \tag{6.1}$$

If the z-axis is not the principal axis, the viewing transformation matrix is pre-multiplied by a permutation matrix, see [Lacr95].

The shear matrix corresponding to a parallel projection will look like this:

[1] From personal experience, about 80% of the voxels are transparent. However, this figure also corresponds well to the figures mentioned in [Lacr95].

$$M_{shear} = \begin{bmatrix} 1 & 0 & s_x & 0 \\ 0 & 1 & s_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

where s_x and s_y denote the shear coefficients. Since this is always a regular matrix, its inverse is well defined, and the warp matrix may be derived as:

$$M_{warp} = M_{view} \cdot M_{shear}^{-1} \tag{6.3}$$

Once the principal axis has been determined, the shear coefficients are chosen such that all the viewing rays are parallel to the principal axis (refer to figure 3.2). Subsequently, the slices are sheared using the shear matrix (6.2), projected and composited into the intermediate image. Finally, the intermediate image is subjected to an affine warp by the warp matrix (6.3). The following sections describe how to take advantage of available hardware acceleration when warping and how to retain proper depth information, enabling appropriate intersection of subsequently added polygonal graphics.

For more information on rendering using a shear-warp factorization of the viewing transformation, the reader is referred to [Lacr95].

6.2 Modifications for Interactive Display

Since the warp described in the previous section is an affine transformation, it is possible to perform the warp using the texture mapping feature of the graphics library. As opposed to section 5.2, where a 1:1 mapping was desirable, mapping the intermediate image as a texture onto a *warped* quadrilateral produces the desired warpage. This is just a question of transforming the vertices of the quadrilateral using the warp matrix of equation (6.3). A polygon thus pretending to be a three-dimensional object is called an *impostor*.

6.3 Combining Voxels and Polygons

The problem of combining voxel-based graphics and polygonal graphics is that of depth sorting. OpenGL uses the z-buffer (or depth buffer) technique, but it knows nothing about the depth information inherent in the rendered volume. It just "sees" a warped quadrilateral parallel to the image plane.

To solve this problem, proper depth information is generated and stored in the z-buffer, thereby overwriting any earlier information. Therefore, the shear-warp object must be the first object being drawn in each frame.

6.3.1 Generating Depth Information

By modifying the inner loop of the shear-warp renderer to make a second intermediate image which for each image pixel contains depth information of the corresponding voxel, it is possible to gain the depth information needed for appropriate interaction with other objects.

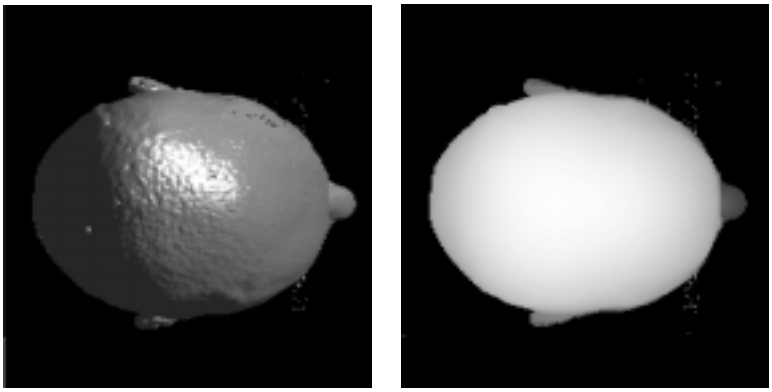

Figure 6.1: *A human head seen from the top (left) and the corresponding depth image (right).*

This intermediate depth image is mapped as a texture onto a warped quadrilateral in exactly the same way as in section 6.2. This maps the depth values as intensity values in the colour buffer (image buffer, as opposed to depth buffer). Reading back this colour information pixel by pixel and storing it in the z-buffer then has the desired effect. The z-buffer contains the appropriate depth information of the voxel volume.

Subsequently, the intermediate image is mapped onto a warped quadrilateral and rendered to the colour buffer. In order to avoid z-buffer corruption by the imposing quadrilateral, z-buffer comparison and updating must be disabled.

Once the z-buffering is reenabled, the graphics system is ready to display polygons intersecting the voxel volume properly, without further ado.

The following form shows the steps needed to draw each frame (refer to figure 6.2):

1. Determine primary viewing axis and factorize the viewing transformation.
2. Render the intermediate images (color and depth information) to texture memory.
3. Clear the colour buffer.
4. Map the intermediate depth image onto a quadrilateral in the colour buffer. The vertices of the quadrilateral have been transformed by the warp matrix of equation 6.3.
5. Transfer the contents of the colour buffer into the depth buffer. This step is assumed to affect the whole depth buffer.
6. Clear the colour buffer (but not the z-buffer).
7. Disable z-buffer update and comparison.
8. Map the intermediate depth image onto a warped quadrilateral in the colour buffer. This quadrilateral has the same coordinates as the one in step 4, i.e. the warps are identical.
9. Enable z-buffer update and comparison.
10. Draw polygons.
11. Show the finished image (e.g. by swapping the colour buffers).

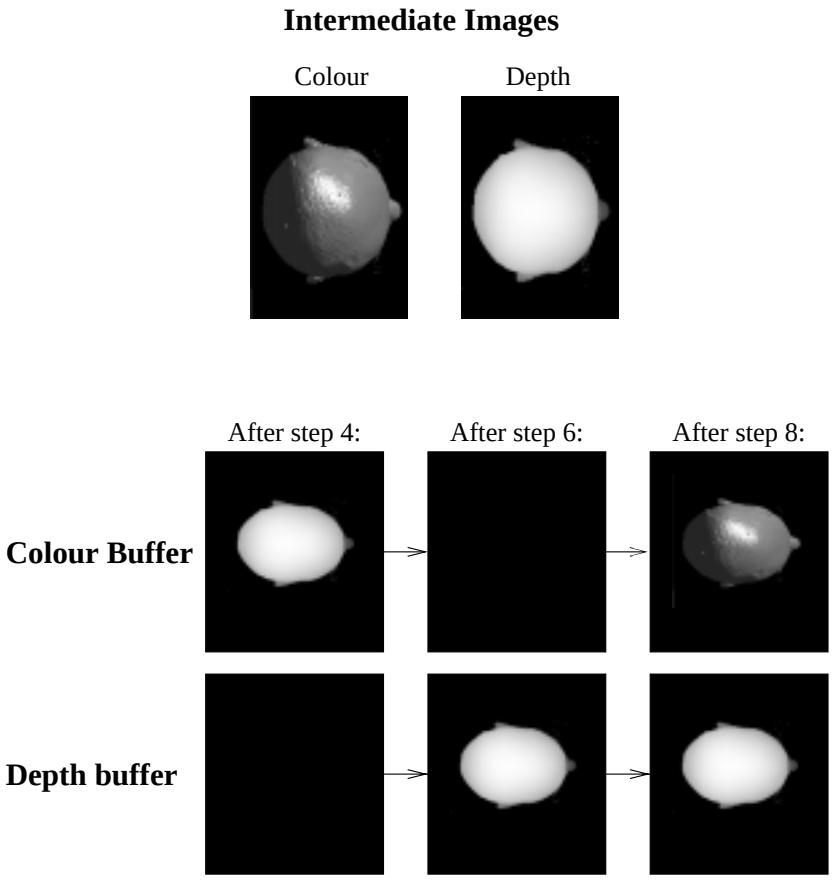

Figure 6.2: *Generation of depth information. The intermediate images are mapped to warped quadrilaterals. Depth information is generated via the colour buffer to insure correct warpage.*

Please note, that the depth image formed in step 4 is never shown directly, but transferred to the depth buffer. To see how such a depth image would look, refer to figure 6.1. Ideally, the intermediate image containing the depth image should be rendered directly to the depth buffer, but to the best of the author's knowledge, OpenGL does not allow this.

The z-buffer must be disabled when performing step 8. Otherwise, the z-buffer contents would be overwritten with depth information corresponding to the imposing quadrilateral. If the impostor was behind the virtual object represented by the z-buffer content, it would not be drawn at all.

Once the z-buffer contains the correct data corresponding to the shear-warp object, any number of polygons may be added. Of significant use in this application would be a polygon indicating the position and orientation of the current slice, see figure 6.3.

6.3.2 Limited buffer resolution

Since typical colour buffers use 8 bits precision per channel (red, green, blue, and opacity) the naïve implementation of the above would have a (maximum) z-buffer precision of 8 bits, and thus 256 equidistant depth values. Since these 256 different depth positions are of the same order as magnitude as typical scan volume resolutions, this limited precision does not pose a problem. In this section, however, a method to make the most of the limited resolution will be presented.

Every element of the intermediate depth image is initialized to 255, which we define as the depth value corresponding to the far plane of the viewing frustum. In order to get the most out of the limited precision, we map the depth value of the nearest voxel to 0 and that of the most distant voxel to 254. Any depth value between these extremes is subject to an affine mapping. In the following, this mapping will be referred to as *depth coding*, the result of which is referred to as *depth codes*.

In practice, the depth values of the nearest and the most distant voxels, z_{min} and z_{max}, respectively, are found by examining the eight corners of the scan volume transformed into camera space.

The following affine transformation is then used to determine the depth code z_c of any depth value z between z_{min} and z_{max}:

$$z_c = \text{round}\left(254\,\frac{z - z_{min}}{z_{max} - z_{min}}\right) \tag{6.4}$$

Before the buffer transfer step above (step 5) is performed, is it necessary to translate the information of the colour buffer into proper depth buffer content. To do this, we need the z-axis distance of the near and far clipping planes, z_{near} and z_{far}, respectively. In the default mode of OpenGL, a z-distance of z_{far} corresponds to a depth buffer value of 1, whereas a z-distance of z_{near} corresponds to a depth buffer value of 0. Values between z_{near} and z_{far} are subject to an affine mapping.

Ignoring rounding errors, the depth of a pixel having depth code z_c is equal to:

$$z = \begin{cases} z_{far} & z_c = 255 \\ \frac{z_c}{254}(z_{max} - z_{min}) + z_{min} & z_c < 255 \end{cases} \tag{6.5}$$

Assume that the z-buffer assigns a value of zero to points whose z-component is equal to z_{near} and one to points whose z-component is equal to z_{far}. Assuming any voxel in question is inside the viewing frustum, the appropriate z-buffer value z_z for a pixel corresponding to a voxel having a depth value of z is

$$z_z = \frac{z - z_{near}}{z_{far} - z_{near}} \tag{6.6}$$

Thus, the appropriate z-buffer value z_z for a pixel having depth code z_c is:

$$z_z = \begin{cases} 1 & z_c = 255 \\ \frac{\left(\frac{z_c}{254}(z_{max} - z_{min}) + z_{min}\right) - z_{near}}{z_{far} - z_{near}} & z_c < 255 \end{cases} \tag{6.7}$$

Remark: The current implementation does not work exactly the way described above. It uses a slightly different mapping to optimize for speed: The nearest and most distant points, z_{min} and z_{max}, are mapped to 1 and 255, respectively. The far clipping plane z_{far} is mapped to zero. Using this mapping, clearing the intermediate depth image to zeroes (which can be done swiftly) leads to the desired initialization of the depth buffer. Furthermore, this mapping has the added benefit of enabling contour indication on the slice indication – see figure 6.3 and section 6.3.3.

To recapitulate the modified version of the above steps 2 through 5: Alongside the normal intermediate image, the modified shear-warp renderer creates an intermediate depth image, consisting of values translated into depth codes using equation (6.4). This depth image is mapped onto a warped quadrilateral in a colour buffer. This colour buffer is read into system memory, and the codes (pixel luminance) are converted back into approximate depth values using equation (6.7). The colour buffer is cleared, and the depth values are then stored in the z-buffer.

6.3.3 Contour Drawing

Figure 6.3 shows the frontal part of a brain peeking through a slice plane. Note how the contour of the part of the brain behind the slice plane is visible. This contour is actually an artifact caused by filtering the depth image when performing step 4 in section 6.3.1. Figure 6.4 shows how the contours come into existence. On figure 6.4.4, a slice indicator is placed. Since contour point c_1 is in front of the slice indicator, it will "punch a hole" in it. The contour point c_2, however, will not be visible, since the depth value of the slice indicator is less than that of the contour point. This shows that this method of contour drawing is less than perfect – but, after all, this contouring phenomenon is just an artifact, included as a curiosity, although it work quite well in practice. Note, that this contour drawing technique is based on the coding scheme described in the remark in section 6.3.2.

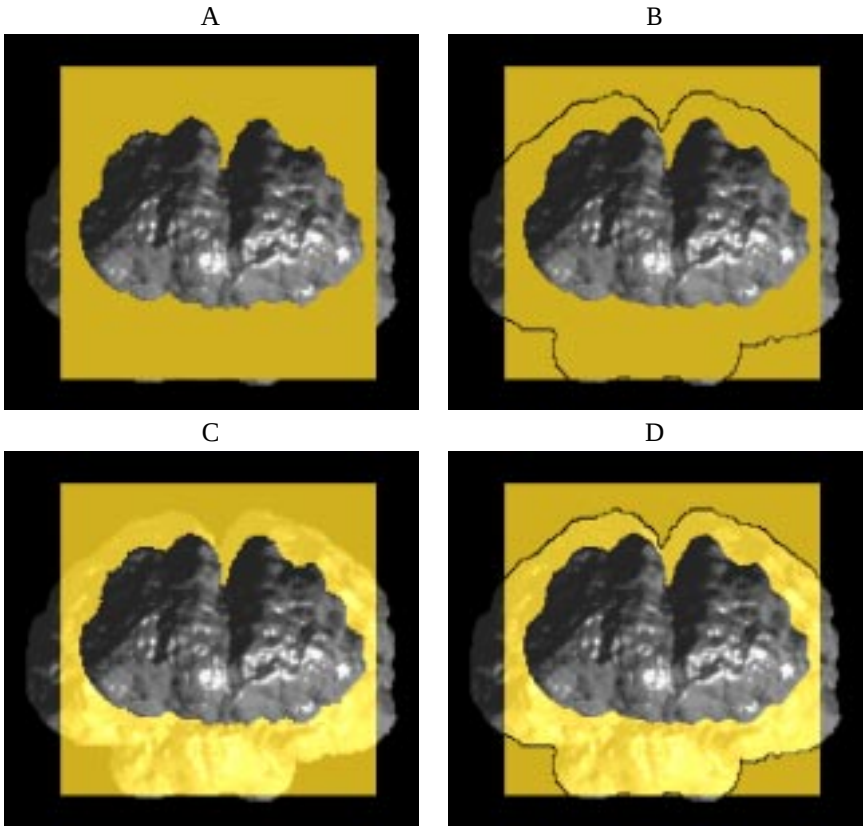

Figure 6.3: *Brain with indication of coronal slice. A: Opaque slice indication. B: Slice indication with contour. C: Transparent slice indication. D: Transparent slice indicator with contour.*

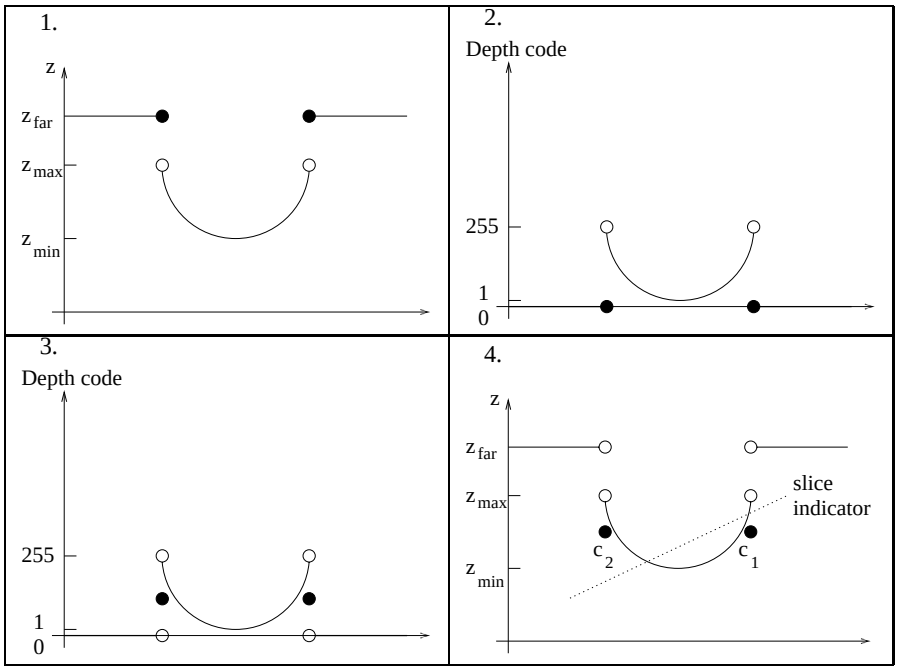

Figure 6.4: *1. A z-buffer scanline of a hemispheric object. 2. Scanline z-values are converted to codes using equation (6.4). 3. Using the codes as luminance, step 4 is performed. Due to the applied texture filtering, the codes are "smoothed". 4. The codes are converted back into depth values using equation (6.7), leaving the contour points c_1 and c_2 "sticking out". Note, that the conversion equations used here have been modified to comply with the coding scheme described in the remark on page 42.*

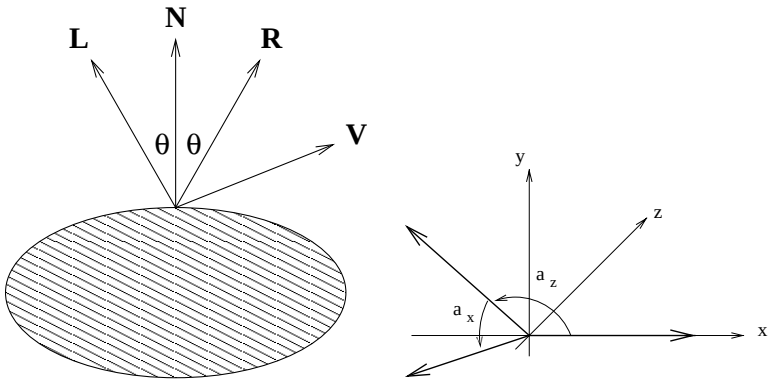

Figure 6.5: *Left: The vectors used in the illumination calculations. Right: Decomposing any unit vector into two angles.*

6.3.4 Phong Illumintation

The 3D display serves a dual purpose: To display the curvilinear slices and to assist in the navigation by showing the position of the current planar slice shown in the 2D display. To facilitate the latter purpose, the user may choose between two 3D shading methods: Each voxel is either rendered using the luminance from the MR image or according to the Phong illumination model[Fole97]. When rendering using the latter method, all the opaque voxels are assumed to be of equal luminance, and the intensity associated with an illuminated voxel is determined by:

$$I = I_a + I_d\,(\mathbf{N}\cdot\mathbf{L}) + I_s\,(\mathbf{R}\cdot\mathbf{V})^n \tag{6.8}$$

where I_a, I_d, and I_s are the ambient, diffuse, and specular intensity coefficients, respectively. The unit vector pointing from the point of incident to the light source is denoted by $\mathbf{L}$; the unit vector pointing from the point of incidence to the viewer is denoted by $\mathbf{V}$ (cf. figure 6.5). The surface normal is denoted by $\mathbf{N}$, and the reflection vector $\mathbf{R}$, defined as $\mathbf{L}$ mirrored about $\mathbf{N}$, may be determined by

$$\mathbf{R} = 2\mathbf{N}\,(\mathbf{N}\cdot\mathbf{L}) - \mathbf{L} \tag{6.9}$$

Assuming that the distances from the 3D object to the light source and to the viewer are infinite, $\mathbf{L}$ and $\mathbf{V}$ can be assumed to be constant for every object voxel. If the specular-reflection exponent n and the three intensity coefficients are assumed to be constant, the intensity equation (6.8) is reduced to a function of the normal vector of the point of incidence.

Lookup Table Indices

To achieve fast Phong illuminations, a lookup table is used. The following shows how a normal vector is decomposed and quantized to fit in a 16-bit lookup table index.

A normal vector is a unit vector in three-dimensional space. Any such vector may be constructed like so (cf. figure 6.5):

1. Start with the vector $\mathbf{v} = [1, 0, 0]^T$.
2. Rotate v about the z-axis at an angle $0 \leq a_z \leq 180°$.
3. Then, rotate v about the x-axis at an angle $0 \leq a_x < 360°$.

Next, the angles a_x and a_z are quantized such that

$$\begin{aligned} a_x &\in \{0,\ 1,\ 2,\ \ldots,\ 180\} \\ a_z &\in \{0,\ 1,\ 2,\ \ldots,\ 359\} \end{aligned} \tag{6.10}$$

At this point, a_x and a_z can easily be "wrapped" into a single integer t_a as follows:

$$t_a = 360a_x + a_z \tag{6.11}$$

and unwrapped as follows:

$$a_x = \left\lfloor \frac{t_a}{360} \right\rfloor \tag{6.12}$$

$$a_z = t_a \mod 360 \tag{6.13}$$

The integer wrapping t_a fits into a 16-bit unsigned integer, since, for any a_x and a_z satisfying equation (6.10), the following holds:

$$0 \leq t_a \leq 65159 < 2^{16} \tag{6.14}$$

In practice

In the prototype, a preprocessing step (see section 9.2.2) calculates the normal vector corresponding to each voxel by normalizing the gradient approximation described in section 2.2.4. This vector is then decomposed into the angles a_x and a_z which are subsequently quantized and wrapped into a 16-bit integer stored along with the intensity value of the corresponding voxel.

When rendering, t_a is directly used as an index into a table of intensities, which must be recalculated with each frame[2] using equation (6.8) for every combination of the quantized angles a_x and a_z given by the expression (6.10).

[2]More precisely: Table recalculation is necessary whenever object orientation changes relative to light or viewer.

6.3.5 Possible Improvements

To cater for the limited depth buffer resolution, another option would be to use several channels (red, green, blue, and opacity) for saving depth information. This way it would be possible to achieve up to 32 bits of depth buffer precision, which is more than most graphics hardware and libraries offer. Using this approach, it is essential to use a nearest-neighbour filter when mapping the depth texture (step 4 above), since any kind of smoothing might corrupt the depth information, depending on how the depth information is laid out in the 4×8 bits.

6.3.6 Alternatives

An alternative to this hassle of generating proper depth buffer data would be to voxelize the polygons and let them be drawn together with the original shear-warp object. However, as seen in section 6.1.1, this would require run-length encoding the volume at every change of the orientation of any polygon relative to the original voxel object (e.g. the brain). One way around this would be to forgo the advantages of exploiting object space coherency and store the object data in an unprocessed format. However, this still leaves the chore of re-voxelising the polygons at each change of orientation or position, and thus re-traversing the volume. The method described in the previous sections allows quick graphical updates when changing polygon orientation, since the "artificially" generated z-buffer data does not change. In other words (assuming the generated z-buffer data is stored in a safe place): If any of the polygons change (but the volumetric object does not), step 1 and 4 of the recipe of section 6.3.1 may be skipped, along with the shear-warp rendering step itself.

6.4 Disadvantages of Shear-Warp Rendering

The disadvantage of shear-warp rendering is one concerning image quality. Since inter-slice filtering is not performed, "holes" may occur in the intermediate image (and thus in the final image). With the understanding that the compositing occurs in front-to-back order, this is illustrated in figure 6.6. Viewing ray $\mathbf{a}$ is nearer to the transparent left pixel than the opaque right pixel in slice 0. Thus, passing slice 0 will not turn the pixel corresponding to this ray opaque. When passing slice 1, however, ray $\mathbf{a}$ is nearest to the opaque left pixel, and pixel $\mathbf{a}$ is set accordingly. Since both of the rays $\mathbf{b}$ and $\mathbf{c}$ are closer to the transparent pixels than to the opaque ones in both slices, the corresponding pixels remain transparent, thus creating an undesired hole in the image. Note, that this artifact is independent of the resampling filter used, since resampling filters only act on non-transparent voxels. In a full-fledged shear-warp renderer featuring semi-transparent voxels (such as the one in [Lacr95]), the problem would be insignificant, since direct transitions from fully transparent to fully opaque voxels (such as the ones in figure 6.6) are rare in typical MR data. Note, that, in practice, the problem only occurs when rendering "thin" objects, such as voxelized surfaces, and not when rendering actually volumetric data. See also section 7.2.

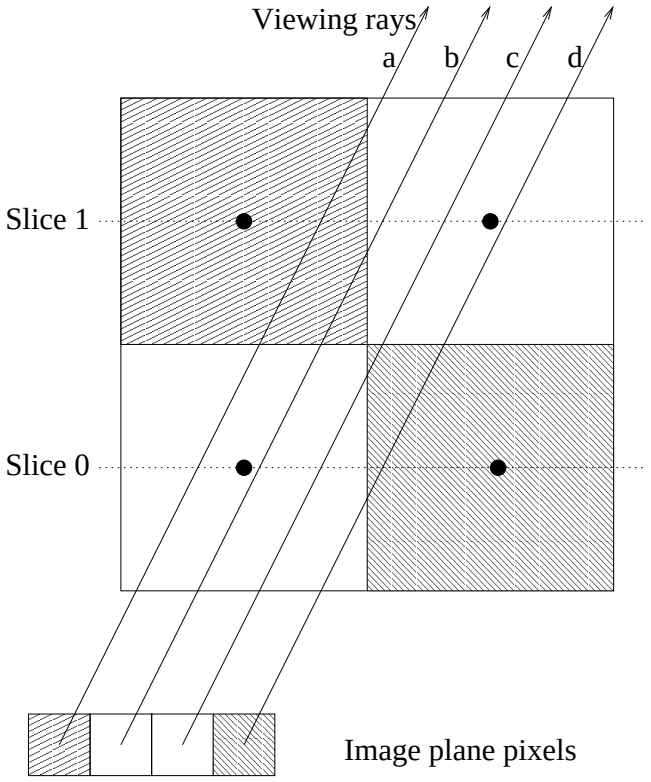

Figure 6.6: *The disadvantage of shear-warp rendering: Holes may form when rendering "thin" objects. Refer to section 6.4.*

Chapter 7

Curvilinear Reformatting

In order to automatically extract curvilinear slices as proposed by Bastos et al., it is necessary to determine the shape and location of the brain. This is equivalent to segmenting brain from non-brain tissue, which is exactly what the appropriately named *Brain Extraction Tool* (BET) from the Oxford Centre for Functional Magnetic Resonance Imaging of the Brain does[Smit00]. BET is a part of the free[1] FMRIB Software Library (FSL).

7.1 Brain Extraction Tool

BET reads a file containing a three-dimensional MR image, delineates the brain surface and writes an MR image containing the brain only. However, in addition to this brain-only image, BET is also able to output a brain mask, i.e. an image of the same dimensions as the original image, containing ones where BET deems there be brain tissue, and zeros elsewhere.

To delineate the brain surface, BET starts by determining the parameters of a "best-fit" sphere. This sphere is tesselated into a mesh of triangles, which is subsequently deformed to obtain a better fit. The deformation, which is done iteratively until an optimal solution is found, takes two things into account: Fit quality and mesh smoothness. When a solution is found, an approximative measure of self-intersection is applied. If this measure indicates that the mesh most likely is self-intersecting, the whole process is repeated with stricter smoothness constraints.

7.2 Surface Extraction

Using the brain mask delivered by BET, it is possible to extract the outermost curvilinear slice[2] by selecting every voxel satisfying the following two conditions:

1. The voxel must be a brain tissue voxel according to the brain mask.

[1] Distributed under the GNU public license.

[2] Note, this slice is not a unique set of voxels, but one depending on the employed notion of connectivity.

2. At least one of the neighbours must be a non-brain voxel.

One way of extracting the next slice (the next layer of the analogous onion) would be to mark the previously extracted voxels as being non-brain voxels and subsequently repeat the selection above.

However, to get, say, the eighth layer, one would have to perform eight iterations of selecting and re-marking. This is evidently inefficient, should one want to access slices in a random order. See also section 7.3.

Another way would be to calculate for each brain tissue voxel the Euclidean distance to the nearest non-brain tissue voxel, and considering the results as a function value or *measure* assigned to the voxels. The set of voxels having a Euclidean distance measure in a given interval would correspond to a slice of a certain thickness. Once the distance calculation is done (needs only be done once), a single selection is all that is needed to extract any slice of any thickness.

Using this method, one could extract the eighth slice mentioned above by selecting those voxels whose distance measure was greater than seven and less than or equal to eight. Note, that the set of voxels of this slice would in general not be equal to the set extracted above, since connectivity has not been considered, i.e. the meaning of "neighbour" has not been rigorously defined.

This method is the one employed in the prototype. Figure 7.1 shows four slices extracted using this technique.

In order to avoid the problems associated with shear-warp rendering of thin surfaces (see section 6.4), the program does not allow the display of single layers (i.e. sets consisting of voxels of distance measure $d_{limit} - 1 < d_{voxel} \leq d_{limit} + 1$). The selection phase only includes those voxels whose distance measure is greater than a user-selected threshold. Thus, if the user set a distance threshold value of zero, the selection phase includes all brain voxels.

The algorithm used to perform the Euclidean distance transformation is due to Saito and Toriwaki[Sait93]. This algorithm calculates the exact distance transform (as opposed to methods such as those employing the chamfer algorithm[Cars00]) of a 3D voxel volume in a matter of seconds on the development system.

Since the Euclidean distance (in voxels) between two voxel centers may be determined as

$$d(v_{ijk}, v_{stu}) = \sqrt{(s-i)^2 + (t-j)^2 + (u-k)^2} \tag{7.1}$$

the squared distance is always an integer. In the prototype, the squared distance is the employed distance measure, since it saves the computation of many square roots and since integer comparisons are preferable to floating point comparisons.

7.3 Other Metrics

In some cases, it may be advantageous to make use of other measures of distance. Should one want to extract surfaces in the first way described in section 7.2, but without having to repeat the extraction to get subsequent layers,

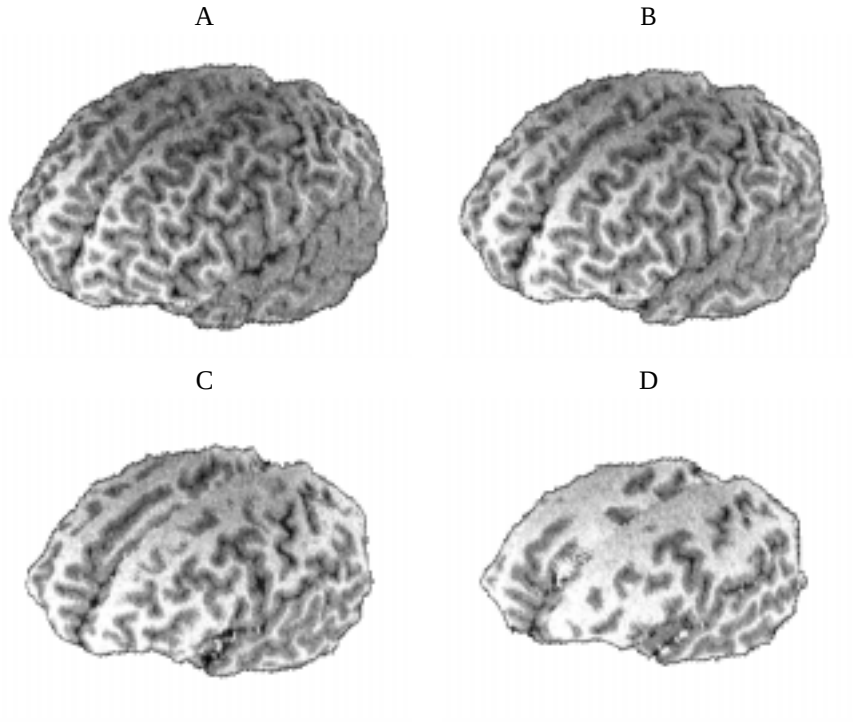

Figure 7.1: *Curvilinear reformatting at varying distances from the surface. A: 2 mm. B: 4 mm. C: 8 mm. D: 12 mm.*

one solution would be to perform a distance transform using the appropriate notion of connectivity, i.e. calculating the distance from each object voxel to the nearest (using the appropriate metric) non-object voxel. Assuming two-dimensional 4-connectivity[3], this would mean calculating the distance transformation using the "city-block" metric. This distance transformation can be performed swiftly using e.g. the chamfer algorithm[Cars00]. Defining a border voxel to be an object voxel having a non-object voxel in its neighbourhood, selecting all voxels whose distance is d gives the exact same set of voxels as performing the following:

iterate $(d-1)$ times
    Mark all border voxels.
    Turn marked voxels into (unmarked) non-object voxels
Select all border voxels.

This duality, which can be extended to other simple dual connectivity / metric pairs[4], may be of some use due to the simplicity and the closeness to the layers-of-an-onion analogy, as well as the swiftness with which a distance transforms may be performed.

[3]With 4-connectivity, the neighbours of a voxel at position (x, y) are $(x \pm 1, y)$ and $(x, y \pm 1)$.

[4]Other pairs include the chess-board metric and 8-connectivity and equivalents in higher dimensions.

Chapter 8

Dysplastic Lesion Detection Aid

In order to be able to detect dysplastic lesions of the cortex, it is necessary to know what characterizes this abnormal tissue as opposed to normal tissue. Since MRI is the imaging method of choice, it would be particularly beneficial to know what MR imaging findings to expect when confronted with a dysplastic lesion.

As described in section 4.1.2, the aim is to detect cortical thickening and / or blurred gray matter / white matter junction. In order to detect cortical thickening, a measure of cortical thickness is needed, which calls for the ability to tell cortical tissue from non-cortical tissue. However, according to the constraints of section 4.1.2, any gray matter / white matter segmentation must be done by a trained expert. Here, a user-selected threshold is applied on the voxel intensities, which is not assumed to perform an accurate segregation of tissue types on a voxel-wise basis, but is only intended to discard the set of voxels unlikely to represent gray matter. This obviously calls for conservative thresholds.

8.1 Intensity Thresholding

Figure 8.1 shows part of a typical intensity histogram[1] corresponding to an MPR image showing part of a human's head. The histogram has has three humps. The hump corresponding to the lowest voxel intensity derives mainly from the background[2], the middle hump mainly from gray matter and the rightmost hump is due to white matter.

Judging from the histogram, the set consisting of brain voxels whose intensity is between 30 and 90 is very likely to contain a large proportion (if not all) of the gray matter voxels. Obviously, these thresholds are conservative, but provide good initial threshold values.

Figures 8.3A and 8.3B show a planar slice before and after intensity thresholding.

[1] A conventional bar graph was unsuitable for the number of intensities, so a regular connected-line graph was used.

[2] The low-intensity voxels also derive from e.g. bone and CSF

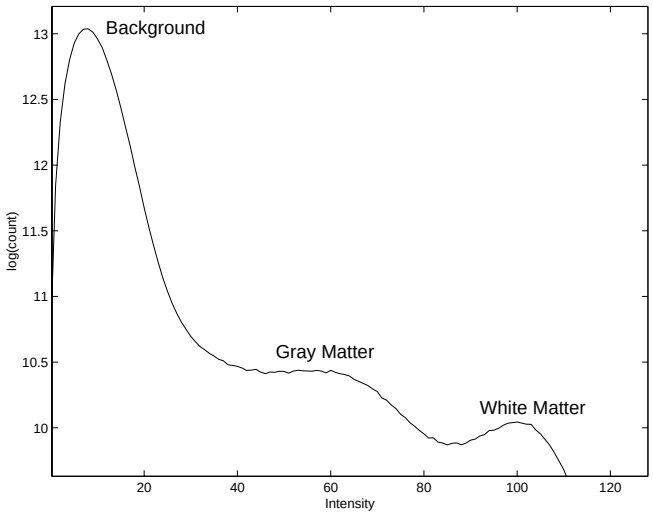

Figure 8.1: *Part of a typical intensity histogram.*

8.2 Gradient Correlation

One simple way of detecting edges in conventional image analysis is to determine large gradient magnitudes, since these usually signify "hard" edges. At this point, the search for dysplastic lesions differs from conventional image analysis, since hard edges (in the right places) do not signify abnormal tissue. A "soft" and blurred gray matter / white matter junction, however, does. In order to detect these blurry edges, a measure of "blurredness" is needed.

The *gradient correlation* is just that. For a voxel v with a specified set of neighbours N, the gradients ∇ of which are all well-defined, we define the gradient correlation as:

$$c_v = \begin{cases} 1 & \nabla_n = 0 \; \forall n \in N \\ \frac{\left|\sum_{n \in N} \nabla_n\right|}{\sum_{n \in N} |\nabla_n|} & \text{else} \end{cases} \tag{8.1}$$

Here $|x|$ denotes some norm of x. The gradient correlation is a measure of the extent to which gradient vectors are pointed in the same direction, in other words, a measure of the correlation of the gradients, hence the name.

In order to examine the properties of expression (8.1), the following general properties of norms will be needed[Hans99]:

$$|x| = 0 \text{ if and only if } x = 0 \tag{8.2}$$
$$|x| \geq 0 \text{ for any } x \tag{8.3}$$
$$|x + y| \leq |x| + |y| \tag{8.4}$$
$$|ax| = a|x| \text{ for any } a \in \mathbb{R}_+ \cup \{0\} \tag{8.5}$$

Using these properties, it can be shown that $0 \leq c_v \leq 1$.

The following examples illustrate the meaning of the gradient correlation c_v.

Example 8.1: A linear field. Consider a field of voxels whose intensity vary linearly with one coordinate. Ignoring the field edges, the voxel gradient ∇v is constant and non-zero. Assuming each neighbourhood to consist of k voxels, the gradient correlation of each voxel is equal to:

$$c_v = \frac{\left|\sum_{n \in N} \nabla_n\right|}{\sum_{n \in N} |\nabla_n|} = \frac{|k \nabla_v|}{k |\nabla_v|} = 1 \ , \tag{8.6}$$

implying field that is not at all blurred.

Example 8.2: A noisy field. Consider a field of voxels whose intensity are given by a white noise function. Furthermore, assume that all gradients are non-zero, yet any sum of gradients over a neighbourhood is equal to zero, since the gradients themselves are so "noisy". This leads to a field of voxels whose gradient correlation is equal to

$$c_v = \frac{\left|\sum_{n \in N} \nabla_n\right|}{\sum_{n \in N} |\nabla_n|} = \frac{|0|}{\sum_{n \in N} |\nabla_n|} = 0 \ , \tag{8.7}$$

implying a totally blurred field.

Example 8.3: An isointense field. Assuming a field of equal intensity, the gradient belonging to every voxel is equal to zero. By definition, this means the gradient correlation is equal to one, reflecting the fact that an isointense field is not at all blurred.

In practice, the voxel neighbourhood definition used is the $3 \times 3 \times 3$-neighbourhood centered on the voxel in question, including the voxel itself. The norm used is the three-dimensional Euclidean norm, given by $|v| = \sqrt{v_x^2 + v_y^2 + v_z^2}$.

Since hard edges are of no interest here, it would be natural to dispose of them by discarding any voxel whose gradient is above a certain threshold, thus performing an "edge disregarding" as opposed to a conventional edge detection. In practice, however, the gradient correlation of voxels constituting a hard edge is close to one, since the gradient magnitude of these voxels are several orders of magnitude larger than the "noise" that would cause blur. This means that any hard edges are removed by applying a maximum correlation threshold (i.e.. discarding voxels whose gradient correlation exceeds some threshold).

Figures 8.3B and 8.3C show the same slice before and after thresholding on gradient correlation.

8.3 Cluster Sizes

Using the methods described in the previous sections, the algorithm is capable of detecting gray matter at various degrees of blur. The voxels satisfying

the selected parameters of both of these double thresholds are called *t-voxels* (threshold voxels). In order to detect cortical thickening, a measure of size is needed for these clusters of t-voxels. One way of doing this would be for each t-voxel, using some definition of connectivity, to count the number t-voxels which could be reached from this voxel without leaving the cluster. This method, however, would be computationally expensive, and might assign a high score to a large, but flat, cluster. The desired clusters have the shape of a rock under a carpet – this method will not give good indication as to whether or not there is a rock under the carpet. Alternatively, one could imagine measuring thickness perpendicular to the cortical surface. However, given the complex geometry of the surface, this would be a daunting task. In practice, a much simpler method is used.

By using the three-dimensional Euclidean distance transformation on what has already been detected (the t-voxels), a measure of cluster size is obtained. That is, for each voxel which is within the intensity interval selected and whose gradient correlation is within the correlation interval selected, measure the Euclidean distance to the nearest voxel which does not satisfy these conditions. The local maxima of such a distance field provides information on the sizes of the corresponding clusters.

The following example demonstrates how this measure of size works. Additionally, a shortcoming of this simple method is shown.

Example 8.4: Limitations of the measure of size. The two-dimensional example of figure 8.2 shows two clusters, differing only by one voxel. The leftmost cluster would have a size of $\sqrt{8}$, since this is the maximal distance value. The cluster to the right, however, would only have a size of $\sqrt{2}$ – half the size of its neighbour. This shows the shortcoming of this measure of size, and emphasizes the need for conservative thresholds, to reduce the frequency of these holes.

		1	1	1	1	
	1	$\sqrt{2}$	2	2	1	
	1	2	$\sqrt{8}$	2	1	
	1	$\sqrt{2}$	$\sqrt{5}$	2	1	
		1	$\sqrt{2}$	$\sqrt{2}$	1	
			1	1		
			1			

		1	1	1	1	
	1	$\sqrt{2}$	1	$\sqrt{2}$	1	
	1	1		1	1	
	1	$\sqrt{2}$	1	$\sqrt{2}$	1	
		1	$\sqrt{2}$	$\sqrt{2}$	1	
			1	1		
			1			

Figure 8.2: *Two-dimensional examples of the measure of cluster size.*

In practice, the selection of clusters is done by displaying only the t-voxels that satisfy the selected distance requirements. These voxels will be called *d-*

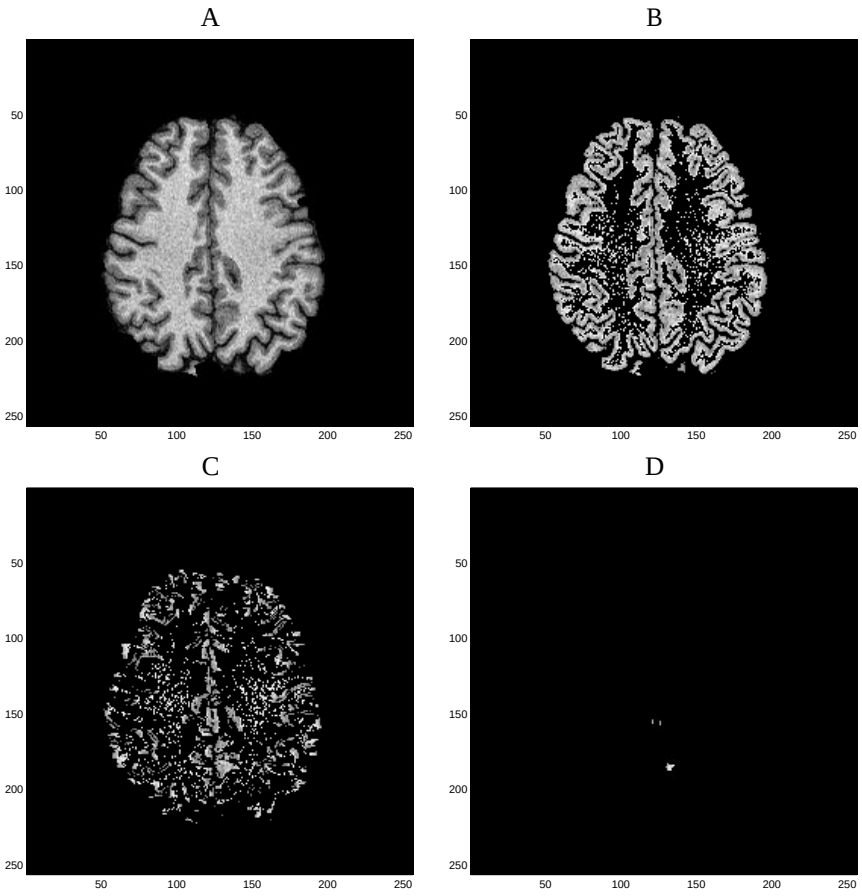

Figure 8.3: *Detection of dysplastic lesions, step by step. A: Original image. B: After intensity thresholding. C: After correlation thresholding. D: After cluster size thresholding.*

voxels in the following. Assuming an infinite upper size limit and a lower size limit of 1.5, only seven voxels of the leftmost cluster of figure 8.2 will be d-voxels, and none of the voxels of the rightmost cluster. This thresholding strategy saves the determination of local maxima of the distances at the cost of the display of the periphery of the cluster. Alternatives to this strategy will be considered in section 8.4.

Figures 8.3C and 8.3D show the same slice before and after thresholding on cluster size.

Note, that it may seem strange to have a *minimum* cluster size, when the smallest lesions are the hardest ones to spot. The need for such a minimum size limitation is seen on figure 8.3. The two double thresholds leave a large number of isolated t-voxels which need to be removed, lest, owing to their sheer number, they nullify the idea of requiring separate consideration. As an alternative to a minimum cluster size, these isolated t-voxels could be removed by employing an anti-speckle method (see [Nibl86]).

8.4 Alternatives

Displaying the periphery of the t-voxel clusters would be an improvement, since the shape and the actual size of the cluster of t-voxels would be more accurately reproduced this way. This could be done by displaying any t-voxel located in a cluster containing a d-voxel. Using conditional dilation [Serr88] this would be possible, but computationally expensive. A relatively cheap approximation would be the following:

for each d-voxel v
    display any voxel closer than d_v to v

Here, d_v denotes the distance from v to the nearest voxel that is not a t-voxel nor a d-voxel. Note, that a requirement that any voxels displayed be t-voxels is not necessary. This might not give the correct shape of the cluster of t-voxels, but it does provide the general outline of the rock under the rug.

One could also consider improving the differential molecules used to estimate the gradients and the neighbourhood used when measuring gradient correlation. Other filter kernels may be able to improve detection in some way, e.g. by reducing directional bias. The filter kernels chosen here are by no means claimed to be optimal; One should, however, be wary not to choose too large a kernel. Since the sought-after features are typically subtle, they are easily eradicated by excessive smoothing.

8.5 Discussion

The lack of a third dimension inherent in computer screens and radiographic film imposes a severe handicap on a radiologist attempting to detect and locate abnormal tissue in the brain. The radiologist is limited to a set of two-dimensional views of the data. Even if the object in question is derived from a three-dimensional data set, it is still projected onto a two-dimensional medium. The advantage of using a computer program to aid in the detection of dysplastic lesions lies in the fact that the algorithm is not restricted to two-dimensional views of the data.

Note how the method described in this chapter differs from conventional image analysis. Usually, one would attempt to find the sharpest edges in an image (e.g. road markings) or areas containing no edges (i.e. no obstructions); Here, the desired features, the blurred, soft edges, are somewhere in between. A defined, sharp transition from gray to white matter is usually a sign of healthy tissue, whereas a blurred transition usually indicates abnormal tissue, such as a dysplastic lesion.

Although the three double thresholds (intensity, gradient correlation, and cluster size) are coupled, it is possible to shut out any one of them by suitable parameter choices. For example, by setting the lower and upper gradient correlation thresholds to zero and one, respectively, the gradient correlation becomes devoid of any importance.

8.5.1 Non-Isotropic Voxels

This thesis has so far assumed voxels to be isotropic, i.e. having the same extent in all coordinate directions. Although the graphics subsystem does not have this limitation, the detection aid and the curvilinear slice extraction discussed in this and the previous chapters would suffer from severe directional bias if confronted with (highly) anisotropic voxels.

In the case of the detection aid, the problem would be severe directional bias caused by the differential molecules employed to estimate image gradients. Correcting biased gradients is a question of multiplying each component of the gradient vectors by appropriate factors, i.e. factors inversely proportional to the voxel size in the corresponding coordinate direction. Once the gradients are corrected, the problem of rectifying the calculation of the gradient correlation remains. One way of doing this would be to associate weights (depending on the distance to the center voxel v) to the gradients and gradient magnitudes of equation (8.1), e.g. like a Gaussian filter kernel (see [Nibl86]), thus transforming the unweighted sums in the numerator and denominator into weighted sums.

The cluster size thresholding and the curvilinear slice extraction both employ the Euclidean distance transform, which must also be able to cope with the anisotropy. Fortunately, the distance transformation is also convertible to the anisotropic case. To quote [Sait93]:

> [The proposed algorithms] *are also applicable with slight modification to a digitized picture sampled with the different sampling interval in each coordinate axis.*

According to said article, the distance transform can be adapted to typical 3D MR images, where voxels are larger in the coordinate direction perpendicular to the slice, without any difficulty.

Although the method developed in this thesis is aimed at (near-) isotropic data sets, it can be modified to the general case without much trouble by following the above guidelines.

8.5.2 Blur Versus Noise

Since the noise level in 3D MR images is rather high, it may be difficult to differentiate the noise arising from structural changes in the cortex (such as “bizarre cells”) from the noise also present in MR images of normal subjects. This latter noise type arises from patient motion (gross patient motion, respiratory motion and motion caused by pulsatile blood flow), scanner hardware, and the employed MR acquisition technique. In a particularly noisy image, the gray matter / white matter junction may seem blurred all over. Whereas the human vision system may be able to detect differences in the noise types, this may not be the case for the automated system described in this chapter. When using the developed method and encountering a highly noisy MR image, it may be necessary to disregard the blurring altogether and focus on detecting other symptoms. At the time of writing, the author has had no opportunity to test the method on a severely noisy MR image.

Chapter 9

Prototype Implementation

The following sections describe implementation specific details of the developed prototype.

Note, that the prototype makes heavy use of indication by colour coding. Since this is does not reproduce well in documents printed in black and white, the detection images in this and the following chapter have subsequently had indicators (arrows and frames) added using an image manipulation program.

9.1 General

The prototype implements the detection technique of chapter 8 along with the automatic curvilinear reformatting method of chapter 7 and the visualization mechanisms of chapters 5 and 6. This constitutes a tool that enables a radiologist to perform an automatic scan for symptoms dysplastic cortical tissue. Any tissue deemed abnormal is highlighted by a colour marking. The program allows verification of these findings by visual inspection, and also for conventional manual search using both planar and curvilinear reformatting.

Although MatLab[1] was used in the development phase, the program is written in approximately 5000 lines of C-code. It was developed on a Linux system, with graphics support through OpenGL[2] (http://www.opengl.org) and the OpenGL Utility Toolkit (GLUT). Since these libraries exist on most prevalent platforms, the program is portable to other systems. The programming interface to the rendering routines is fairly clean and uses a naming convention similar to that of OpenGL.

9.2 User Interaction

Figure 9.1 shows the graphical display of the prototype. By keypresses or simple click-and-drag mouse operations, the user can manipulate both the 2D and the 3D images. The brain and slice images may be independently zoomed and rotated about their respective midpoints, and the slice may be translated.

[1] MatLab is a registered trademark of the MathWorks, Inc.
[2] OpenGL is a registered trademark of Silicon Graphics, Inc.

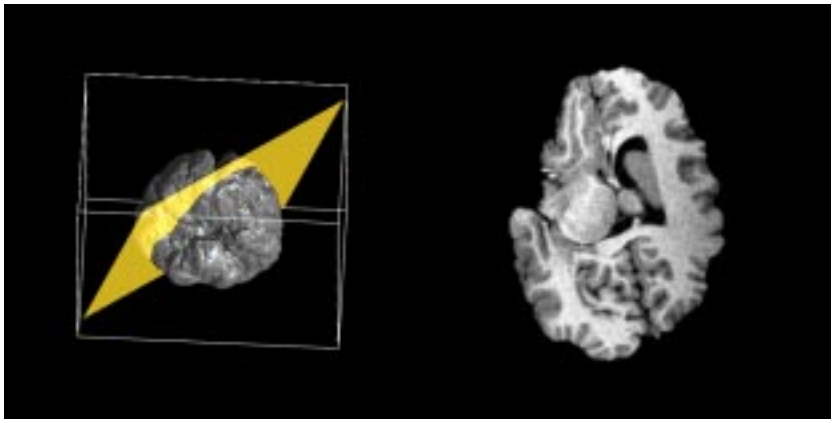

Figure 9.1: *Screenshot of the graphics display. On the left is the 3D display with a slice indicator. On the right is the 2D slice.*

The display of curvilinear slices mainly affects the 3D display. However, a curve indicating the intersection between the curvilinear slice and planar slice is shown on the 2D display (see figure 9.2). This curve is updated whenever the user selects a different curvilinear slice. Curvilinear reformatting may be turned on or off.

The display of detections, shown in figure 9.2, can be turned on or off. When on, the detections are shown on both the 2D and 3D displays, including curvilinear slices. After adjusting detection parameters (the double threshold values on intensity, correlation, and cluster size), the user may start a re-detection.

To enable user control of the image contrast, the user may adjust the parameters of the intensity mapping described in section 5.1.2. The adjustment is performed by dragging the mouse while holding down the middle mouse button, which is the de facto standard of contrast control in e.g. MPR software.

The resampling method employed is selected at a keypress.

Sections 9.3 and 9.4 describes two extra features that may ease the detection of cortical disorders.

9.2.1 Selective Updating

By only recalculating and redrawing when necessary, it is possible to save a considerable amount of data processing, which in turn leads to a higher rate of graphical updating. By saving intermediate images as well as depth information, much work can be saved, since e.g. rotating the planar slice does not require retraversal of the three-dimensional MR image, even though the slice indicator on the 3D display would need to be updated.

The following table describes the necessary updates corresponding to a given user action.

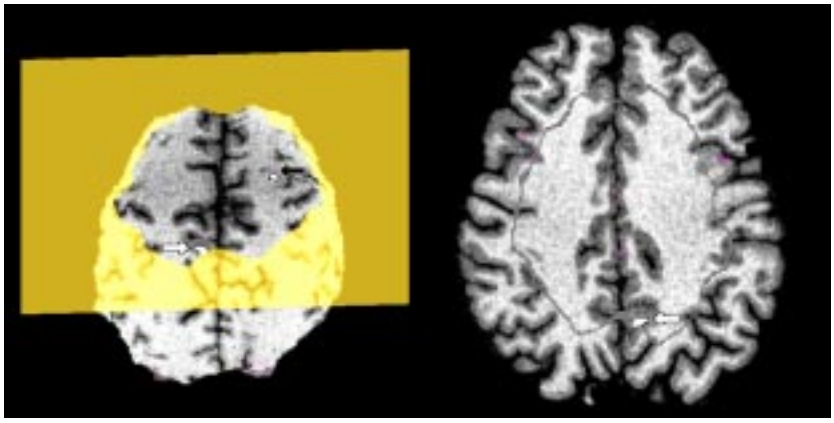

Figure 9.2: *Screenshot with detected dysplasia (cf. figure 10.1C). The 3D display shows the curvilinear slice 18 mm from the surface. The 2D display shows the intersection of the curvilinear slice and the planar slice as a gray curve. The detected dysplasia is indicated with white arrows. Note the false detection (indicated with a black arrow) at the top of the brain (see section 10.2). A magnified version of this figure may be found on pages 84-85 in the appendix.*

Action	Retraverse 3D	Redraw 3D	Redraw 2D
Re-detect	X	X	X
Rotate 2D		X	X
Rotate 3D	X	X	
Zoom 2D		X	X
Zoom 3D		X	
Translate 2D		X	X
Resampling 2D			X
Resampling 3D	X	X	
Intensity map.	X	X	X
CR on / off	X	X	X
DLDA on / off	X	X	X

Notes to this table:

- Redrawing the 3D display is often necessary when working with the 2D display because the slice indicator needs to be updated.
- *CR* and *DLDA* are abbreviations of *curvilinear reformatting* and *dysplastic lesion detection aid*, respectively.
- *Intensity map.* refers to the contrast enhancing intensity mapping described in section 5.1.2.
- *Resampling* refers to a change in the resampling method used in a particular display.

9.2.2 Preprocessing

The preprocessing of the data is done using a separate program. This program creates the necessary run-length encoded volumes. To achieve in-order object traversal regardless of view angle, three encoded volumes are needed (see

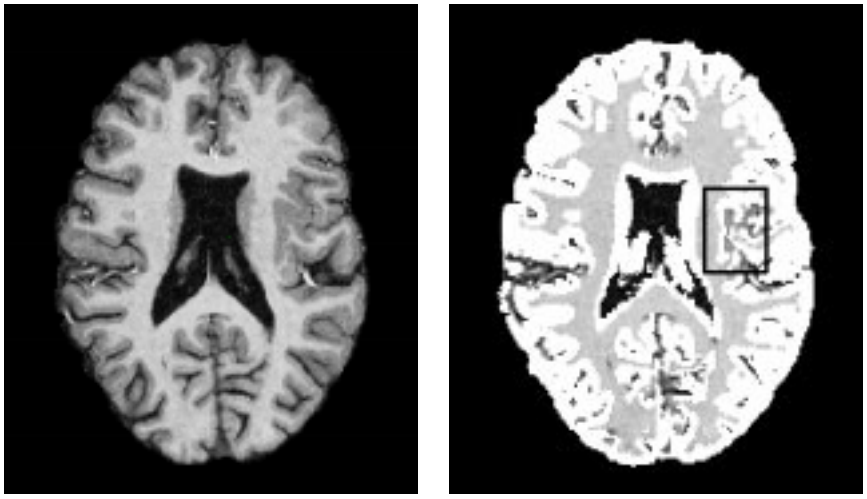

Figure 9.3: *Emphasizing the cortical structure facilitates detection of abnormal cortical asymmetries (black frame).*

section 6.1.1). The program accepts a user-selected threshold and classifies all voxels as being transparent (if their intensity is below the threshold) or opaque (if their intensity is above the threshold). The gradients of all opaque voxels are then calculated, encoded (see section 6.3.4), and stored along with the intensity of the corresponding voxel in each of the three run-length encoded volumes. The gradient correlation corresponding to each voxel is also calculated and stored in a separate file.

9.3 Histogram Display

To aid in the determination of feasible parameters for the intensity thresholds, the program features the display of intensity histograms (see figure 8.1). The user can zoom in and out on any parts of the histogram by clicking the mouse buttons. This histogram, which is drawn as a connected-line graph, may help the user find feasible parameters by showing the approximate ranges of main voxel types (background, gray matter, and white matter).

9.4 Cortical Detection

In order to emphasize the cortical structure, the prototype contains a simple method of detecting gray matter. The method uses the same parameters as the dysplastic lesion detection method, but in a different way:

1. Select all voxels whose intensity is within the range of the user-selected intensity thresholds.
2. Remove all voxels whose gradient correlation is *below* the *upper* correlation threshold. This ensures that only gray matter matter near an edge

(background / gray matter edge or gray matter / white matter edge) are included. The voxels that pass steps 1 and 2 are called b-voxels.

3. Discard the b-voxels, whose $3 \times 3 \times 3$-neighbourhood contains less than b_{min} b-voxels, where b_{min} is a user-selected threshold. This last step removes isolated voxels and small clusters of voxels which may distract the radiologist.

Figure 9.3 shows how putting visual emphasis on cortical structure facilitates the visual detection of cortical asymmetry.

An alternative would have been to use a more advanced and accurate segmentation tool, such as FMRIB's Automated Segmentation Tool (see section 10.2). FAST takes about 45 minutes segmenting a standard MR image on the fastest settings, and even that is considered fast in the field of automated tissue segmentation. Since the prototype is supposed to be an interactive tool, advanced segmentation techniques are thus out of the question.

It should be noted, that the cortical detection algorithm described in this section does not perform a voxel-wise tissue segmentation. It only approximates the cortical structure so as to be able to emphasize it for facilitating detection of asymmetry of the cortical structure.

9.5 Data

The preprocessing program and the main part of the prototype are both capable of reading data in the *simple file* and *Analyze* formats, provided the intensity values be represented using 16-bit signed integers using little-endian byte ordering or 32-bit ieee floating point numbers. Internally, intensities are represented using 16-bit signed integers.

The loaded data consists of:

- Three run-length encoded volume files (containing intensity and gradient information), each corresponding to a principal slicing direction.
- One (unencoded) volume file containing voxel intensities, used for planar slicing and display.
- One (unencoded) volume file containing gradient correlation.
- One (unencoded) brain mask (generated by Brain Extraction Tool).

The gradient correlation file is optional, as is the brain mask. However, without the gradient correlation, automatic symptom detection is disabled, and without the brain mask, curvilinear reformatting is disabled.

Each of the above four items represent approximately 18 megabytes of data for a $256 \times 256 \times 140$ voxel volume, totalling 72 megabytes of data loaded when optional features are enabled.

9.6 Performance and Requirements

The main program allocates 18 to 20 bytes per voxel loaded, plus an additional 7 Mb for textures and intermediate storage. This means that, using a normal

$256 \times 256 \times 140$ voxel volume as data source, the program takes up approximately 180 megabytes of memory.

On the development computer, an AMD 1.3 GHz CPU equipped system with 512 megabytes of RAM, the rendering speed was about 3 frames per second when simultaneously updating both displays. This is barely enough for hassle-free interaction.

The development system is equipped with a graphics card capable of accelerating OpenGL graphics in hardware. Although OpenGL is only used to draw three texture mapped quadrilaterals (2d, 3d, and depth images) as well as for one colour buffer read operation and one depth buffer write operation, hardware acceleration has significant impact on frame rates.

In the current implementation, the OpenGL driver takes by far the largest portion of the frame time:

60% of the frame time is used by the OpenGL driver performing the colour buffer read and the depth buffer write.

25% of the frame time is used by the OpenGL driver uploading the appropriate textures to graphics memory.

This leaves only 15% of the frame time for everything else, such as traversing the volume to form the intermediate images, converting depth information, etc. The OpenGL buffer reads / writes are notoriously slow, in part because a graphics pipeline flush and stall are needed. Although most modern graphics hardware supports it, there is not yet a render-to-texture mechanism in OpenGL[3]. By optimizing these issues alone (e.g. by performing the warped mapping in software, thus bypassing the driver), frame rates in the twenties are estimated to be achievable on the development system.

Updating the array of detected voxels after a change of parameters takes about 10 seconds on the development system. A slight improvement may expected to arise from optimizing the distance transform implementation for speed.

At a slight cost in speed, memory consumption may also be approximately halved by packing data in a more efficient manner. As an example: Each voxel in the brain mask contains one bit of information (brain / non-brain), yet its internal format is 16-bit signed integer.

[3]Sadly, even the OpenGL version 1.3 specifications released 14th of August 2001 contain no render-to-texture feature.

Chapter 10

Results

This chapter describes the results so far obtained. The purpose of the tests, which were performed by the author of this thesis, who is not a radiologist, was to verify the diagnosis made by a radiologist. At the time of writing, the MR images of four patients, each with a diagnosis of focal cortical dysplasia, have been subject to verification by the developed prototype. Owing to this limited amount of practical experience, an optimal range of parameter values cannot be said to have arisen. Each of the four cases are shown in the following section.

10.1 Detection of Dysplastic Lesions

Figure 10.1 shows four axial slices, each from different patients with focal dysplasia. Although visually different, they were detected using parameter sets differing only in intensity threshold values (a change necessitated by the different intensity histograms of the four volumes). The remaining parameters were:

Gradient correlation, lower limit	0.3
Gradient correlation, upper limit	0.8
Cluster size, lower limit	2 mm
Cluster size, upper limit	∞

The upper limit on cluster size being at infinity is equivalent to having no upper limit.

The detection of the dysplastic lesions in B, C, and D were fairly robust to varying parameter sets. A slight varation to the parameters of A leads to multiple false detections (refer to figure 10.2). In the leftmost image, the lower threshold of the gradient correlation has been lowered to 0.2 (all other parameters comply with the above table). A number of false detections appear in areas of very soft edges, e.g. in the basal ganglia. The findings in the basal ganglia may easily be disregarded by the experienced user, but the remaining detections require separate consideration. Conversely, when raising the upper limit on correlation (right image), false detections appear in areas around harder edges. Magnified versions of figure 10.1A and 10.2 may be found on

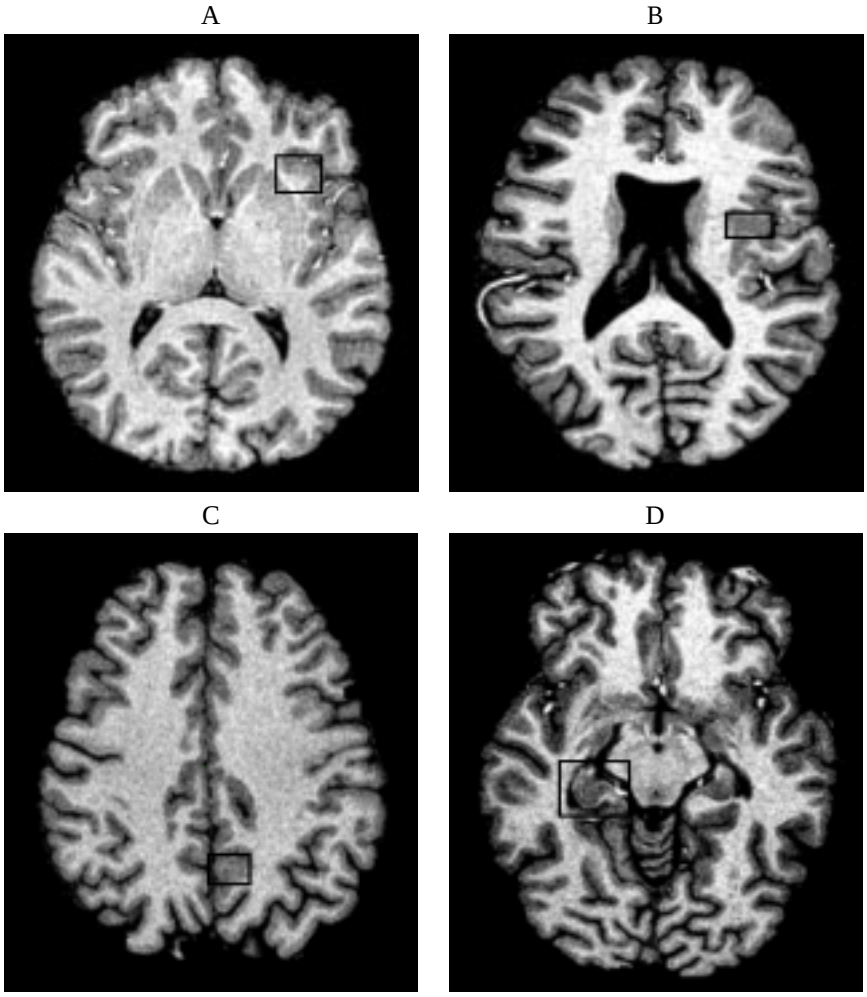

Figure 10.1: *Detected focal dysplasias verified by a radiologist. Dysplastic lesions are framed.*

pages 86–88 in the appendix.

Unfortunately, the prototype has as yet been used only to verify previously made diagnoses – the acid test of actually making a diagnosis awaits the near future.

10.2 Effects of Intensity Nonuniformity

Owing to inhomogeneous receiver coil sensitivity, inhomogeneous main field and RF field, and inhomogeneous magnetic susceptibility (“magnetizability”) of the scanned object, spatial intensity variation unrelated to the anatomic structure may arise. This variation, or bias, typically leads to a clearly visible decrease in intensity level and tissue contrast at the top and bottom of sagittal

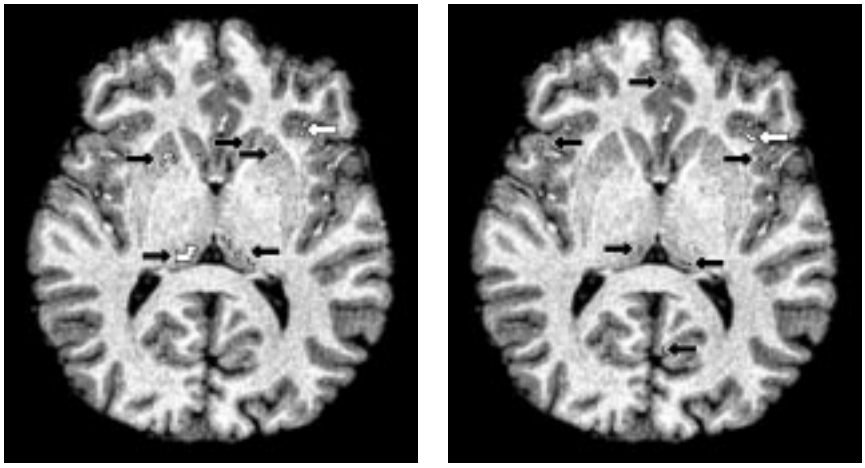

Figure 10.2: *Axial slices demonstrating the effects of parameter variation in patient A. Left: Lower correlation threshold set to* 0.2*. Right: Upper correlation threshold set to* 0.9*. The dysplasia is indicated with white arrows, false detections with black arrows. Magnified versions of these images can be found on pages 87-88 of the appendix.*

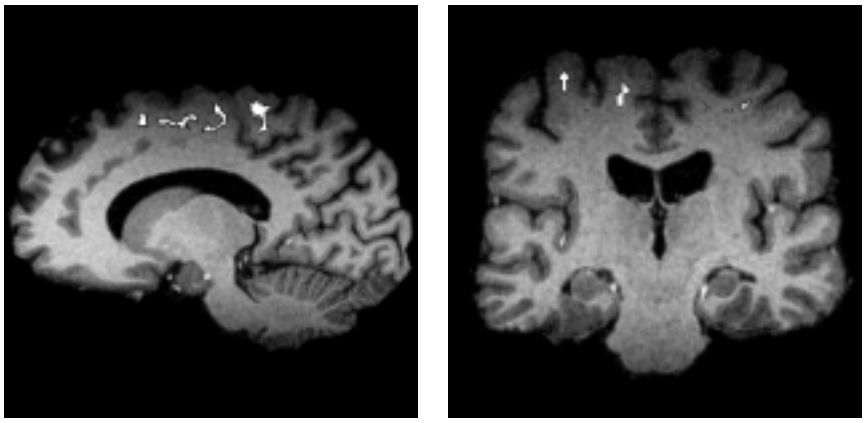

Figure 10.3: *Sagittal (left) and coronal slices showing false detections caused by intensity nonuniformity. Magnified versions of these can be found on page 89 of the appendix.*

and coronal views. Whereas the decrease in intensity level may not have great significance on the visual interpretation of brain images, the results of computational procedures may suffer a great deal. This type of nonuniformity may be seen as a "shadow" at the top of the brain depicted in figure 7.1.

For the following two reasons, the method described in this thesis can be expected to have a significant amount of false detections when used on such nonuniform images:

- The general decrease in intensity makes white matter voxels have luminance values in the gray matter area, so that everything seems to be gray matter.
- The decrease in tissue contrast gives rise to a decrease in the signal-to-noise ratio, making every remaining edge seem fuzzy.

This expectation proves to hold in practice: figure 10.3 shows a considerable amount of false detections due to image bias. Note, that the false detection on figure 9.2 is also caused by this kind of intensity nonuniformity.

The FMRIB software library (from where Brain Extraction Tool originates) contains a tissue segmentation program capable of performing for bias correction. This program, called FAST (FMRIB's Automated Segmentation Tool, see [Zhan00]), has been used to correct a biased image. Unfortunately, the number of false detections did not decrease significantly using this corrected image, most probably because the tissue contrast in the parts of the original image affected the most by this bias had already inreparably deteriorated.

A comprehensive evaluation of nonuniformity correction methods can be found in [Arno01].

Chapter 11

Summary

Manual detection of the development disorder of the cerebral cortex, known as focal cortical dysplasia (FCD), may take several days of combing through magnetic resonance (MR) data, even for a clinical expert. An algorithm to assist in the detection by 3D image analysis has been presented. This algorithm detects two key symptoms of FCD, namely thickening of the cortex and poor differentiation of gray matter versus white matter. To facilitate manual verification of detections, the algorithm has been embedded in a graphics system combining conventional planar as well as curvilinear slice view to facilitate interpretation of data. A strong point in the algorithm is the fact that it is three-dimensional in nature, as opposed to the inspection by a human user, who is restricted to examining a few two-dimensional surfaces at a time. It should be noted, however, that the method presented here constitutes a tool for assisting a clinical examiner, and not for automatically making diagnoses.

The results so far are few, but encouraging.

11.1 Main Contributions

In summary, the main contributions of this thesis are as follows:

- A method for fast automatic detecting of key symptoms of focal cortical dysplasia.
- Automation of the type of curvilinear reformatting introduced in [Bast99].
- Extension of the shear-warp rendering method to achieve the appropriate behaviour when interacting with subsequently added graphical objects.
- Demonstration of the use of texture mapping hardware to perform the affine 2D warp required by shear-warp rendering.

11.2 Possible Improvements

The method reports a fair amount of noise as well as some strong false findings. The former is characterized by the erroneous detection of small clusters

(typically consisting of a single voxel) and most may be disregarded when taking their size into consideration. The latter are caused by the following:

- Restricting the program to consider local geometry information only; to have no knowledge of the global tissue anatomy. False detections in the basal ganglia (also consisting of gray matter), for example, belong to this class
- Intensity nonuniformity. This is easily identified as a general decrease in intensity and contrast

If these types of false findings are disregarded, the remaining detections are usually few, and may be accepted or rejected manually. Whether or not experience will teach radiologists to unconsciously disregard the false findings of these types, only time will tell.

The program has been assessed by a radiologist, and general navigation and positioning of slices was found to cause inconvenience. A few simple navigational tools were suggested to solve the problem, see section 11.4.

11.3 Benefits

The method is able to detect visually different dysplastic lesions using the same set of parameters. Using other parameter sets, it should be possible to facilitate the detection of other abnormalities such as tumours and subtle cerebral hemorrhages. The algorithm uses simple (and thus fast) image analysis techniques and may easily be extended to detect other features.

By being an inherently three-dimensional algorithm, it is capable of helping a radiologist “see through” the flat images on a computer screen, thereby reducing the effect of the reduced dimensionality of conventional MR analysis tools.

The graphics system, having a 2D as well as a 3D image with indication of contours and trouble spots is reported as being “very appealing”.

11.4 Future Work

Obviously, the method still needs much testing in order to pin down feasible parameter ranges, as well as to provide the radiologist with the experience needed to swiftly discern different classes of findings. Further issues of future work include:

- Tools for orientation. A simple indication of the object coordinate system was suggested, as well as a means of correlating the images of the 2D and 3D displays.
- Reducing the number of false detections.
- Optimizing other parameters, such as the filter kernels and differential molecules.
- Extending the algorithm to provide valid results for data volumes consisting of non-isotropic voxels.

- Optimizing the graphics system for speed. A dramatic increase is assumed possible.
- To let graphics hardware perform the 2D zooming as described in section 5.3.
- To sacrifice imaging quality for speed when interacting with the object, and then to render at maximum quality once the interaction stops (also described in section 5.3).
- To introduce alternative measures of cluster size to improve the detection of cortical thickening regardless of shape and natural variations in cortical thickness.
- Displaying the periphery of detected clusters to emphasize cluster sizes and shapes.
- By finding local maxima in the distance-to-nearest-non-t-voxel volume, the approximate size and location of dysplasias can be determined. This way, the program would be able to lead the radiologist directly to the strongest detections.

11.5 Conclusion

The object of this work reported here was to develop a tool to assist a radiologist in the localization of subtle focal cortical dysplastic lesions. An algorithm capable of detecting key symptoms of such disorders was developed along with a graphics system to support verification of findings by visual inspection. Much experimental work is needed to determine whether or not the method is valuable, yet the results so far are encouraging.

The development continues.

Appendix A

Selected Images

This following pages contain blow-ups of the more detailed images of this thesis. The images have been enhanced using an image manipulation program to compensate for the lack of colour in this reproduction.

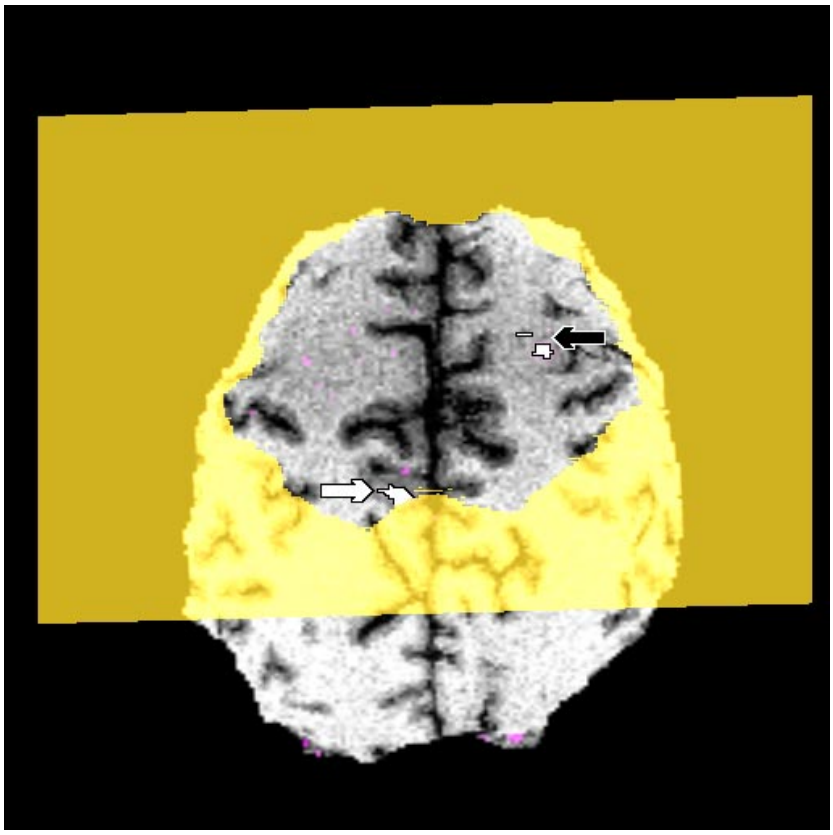

Figure A.1: *Magnification of part of figure 9.2. Screenshot with detected dysplasia (cf. figure 10.1C), showing the curvilinear slice 18 mm from the surface. The detected dysplasia is indicated with a white arrow. Note the false detection (indicated with a black arrow) at the top of the brain (see section 10.2).*

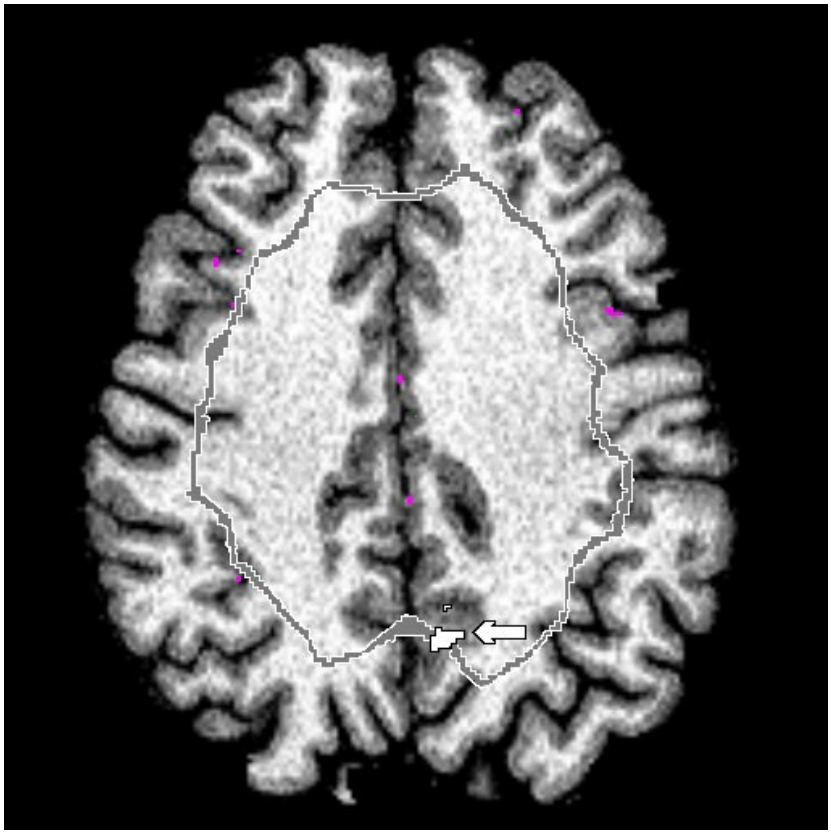

Figure A.2: *Magnification of part of figure 9.2. Screenshot with detected dysplasia (cf. figure 10.1C), showing the intersection of the curvilinear slice and the planar slice as a gray curve. The detected dysplasia is indicated with a white arrow.*

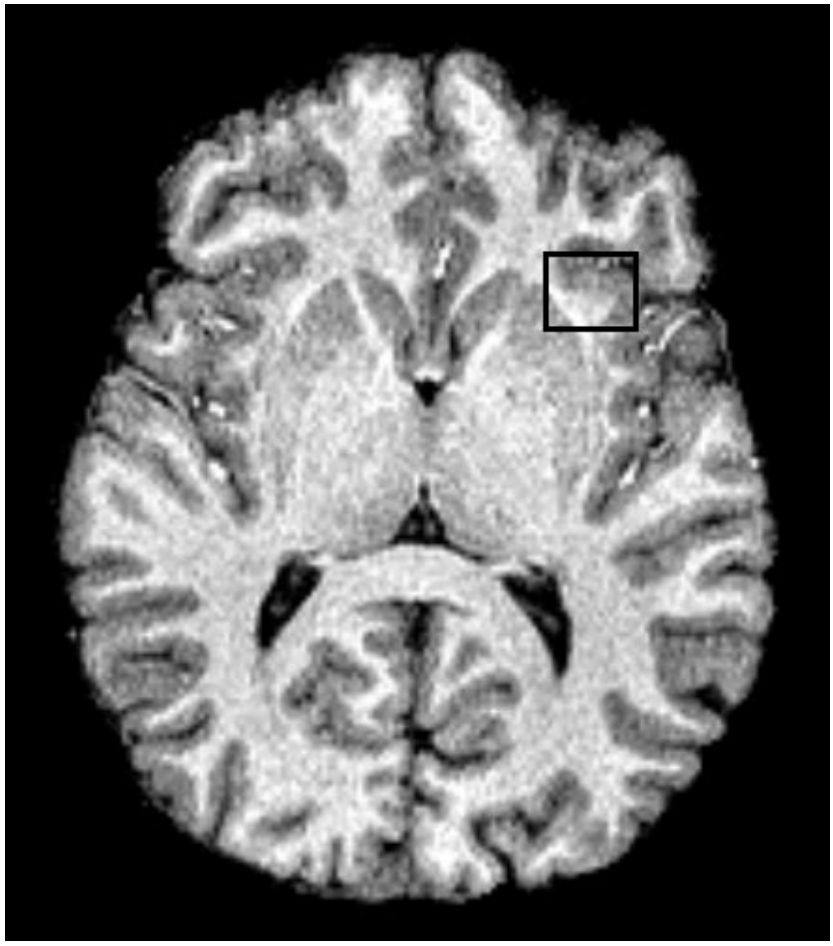

Figure A.3: *Blowup of figure 10.1A. Detected focal dysplasia verified by a radiologist. The dysplastic lesion is framed.*

Figure A.4: *Magnification of figure 10.2A. Axial slice demonstrating the effects of parameter variation in patient A. Lower correlation threshold set to* 0.2*. The dysplasia is indicated with white arrows, false detections with black arrows.*

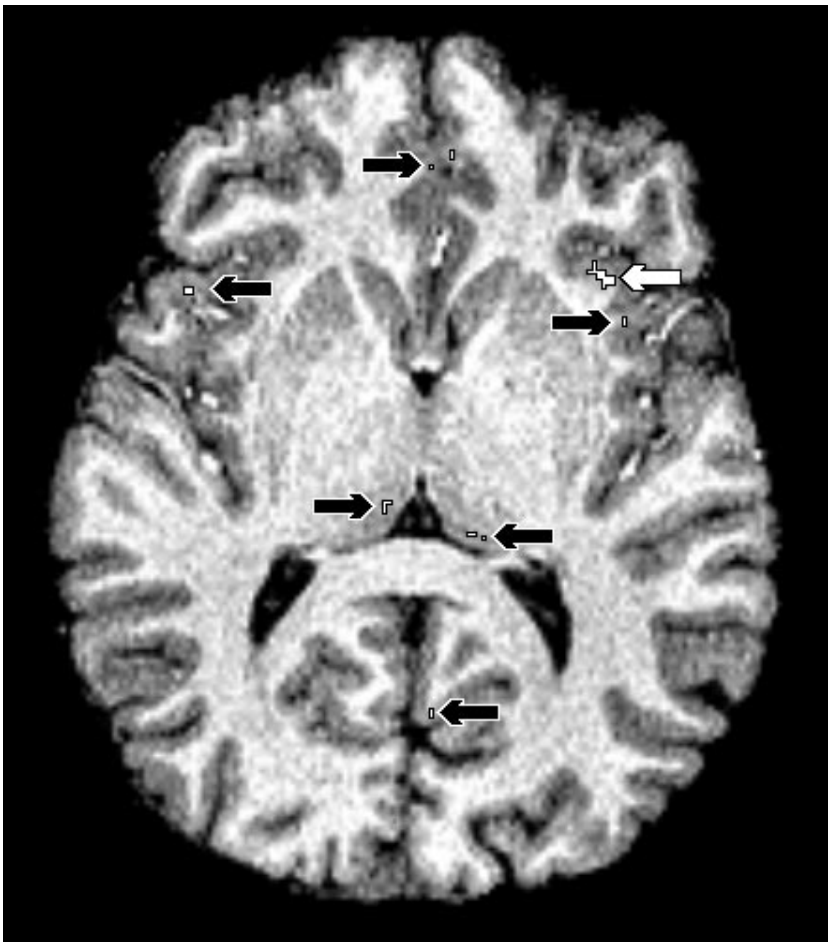

Figure A.5: *Magnification of figure 10.2B. Axial slice demonstrating the effects of parameter variation in patient A. Upper correlation threshold set to* 0.9*. The dysplasia is indicated with white arrows, false detections with black arrows.*

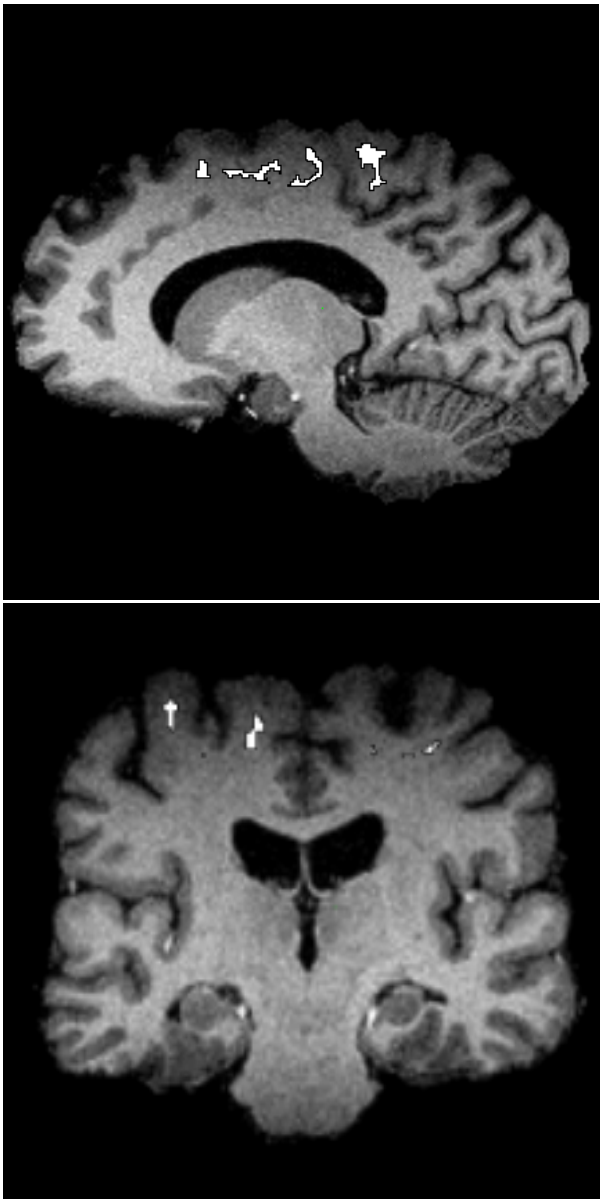

Figure A.6: *Magnification of figure 10.3. Sagittal (top) and coronal slices showing false detections caused by intensity nonuniformity.*

Appendix B

Resources on the Internet

The following addresses are valid at the time of writing (August 2001), but owing to the dynamic nature of the internet, this may change in time.

- The FMRIB Software Library (Brain Extraction Tool):
  http://www.fmrib.ox.ac.uk/fsl
- A technical report on the Brain Extraction Tool (on-line version of [Smit00]):
  http://www.fmrib.ox.ac.uk/analysis/research/bet/bet/
- Abstract on detection of Focal Cortical Dysplasia by Voxel-Based Morphometry (on-line version of [Hupp01]):
  http://www.academicpress.com/www/journal/hbm2001/9991.html
- Article on template-based ray casting (on-line version of [Lee97]):
  http://cglab.snu.ac.kr/~chlee/template.ps
- Philip Lacroute's Ph.D thesis on shear-warp rendering (on-line version of [Lacr95]):
  http://www-graphics.stanford.edu/papers/lacroute_thesis/
- Article on volumetric rendering using hardware accelerated pixel shading (on-line version of [Enge01]):
  http://wwwvis.informatik.uni-stuttgart.de/~engel/pre-integrated/

Bibliography

[Arno01] James B. Arnold, Jeih-San Liow, Kurt A. Schaper, Joshua J. Stern, John G. Sled, David W. Shattuck, Andrew J. Worth, Mark S. Cohen, Richard M. Leahy, John C. Mazziota, and David A. Rottenberg: *Qualitative and Quantitative Evaluation of Six Algorithms for Correcting Intensity Nonuniformity Effects*. NeuroImage 13, pp. 931-943, 2001.

[Ashb00] John Ashburner and Karl J. Friston: *Voxel-Based Morphometry – The Methods*. NeuroImage 11, pp. 805-821, 2000.

[Bast99] Alexandre C. Bastos, Roch M. Comeau, Frederick Andermann, Denis Melanson, Fernando Cendes, Francois Dubeau, Suzanne Fontaine, Donatella Tampieri, and Andre Olivier: *Diagnosis of Subtle Focal Dysplastic Lesions: Curvilinear Reformatting from Three-Dimensional Magnetic Resonance Imaging*. Annals of Neurology, Vol 46, No 1, July 1999.

[Bran92] Michael Brant-Zawadzki, Gary D. Gillan, and Wolfgang R. Nitz: *MPRAGE: A Three-Dimensional T_1-Weighted Gradient Echo Sequence – Initial Experience in the Brain*. Radiology 182(3), pp. 769-775, 1992.

[Came92] G. G. Cameron and P. E. Undrill: *Rendering Volumetric Image Data on a SIMD-Architecture Computer*. Proceedings of the Third Eurographics Workshop on Rendering, pp. 135-145, 1992.

[Cars00] Jens Michael Carstensen (Editor): *Digital Image Processing*. IMM, Technical University of Denmark, 2000.

[Corm97] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest: *Introduction to Algorithms*. ISBN: 0-262-53091-0. MIT Press, 1997.

[Edel96] Robert R. Edelman, Michael B. Zlatkin, John R. Hesselink, et al.: *Clinical Magnetic Resonance Imaging*, second edition. W.B. Saunders Company, 1996.

[Enge01] Klaus Engel, Martin Kraus, and Thomas Ertl: *High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading*. Eurographics / SIGGRAPH Workshop on Graphics Hardware '01 (to appear), 2001.

[Fole97] James D. Foley, Andries van Dam, Stephen K. Feiner, John F. Hughes: *Computer Graphics. Principle and Practice*, second edition. ISBN 0-201-84840-6. Addison-Wesley, 1997.

[Fuch90] Henry Fuchs: *Systems for Display of Three-Dimensional Medical Image Data*, in: 3D Imaging in Medicine, ISBN: 3-540-52663-3. ISBN: 0-387-52663-3. Springer Verlag, 1990.

[Hans99] Vagn Lundsgaard Hansen: *Fundamental Concepts in Modern Analysis*. ISBN 981-02-3894-0. World Scientific, 1999.

[Hupp01] Hans-Jürgen Huppertz, Jan Kassubek, Freimut D Jüngling, Andreas

Schulze-Bonhage: *Detection of Focal Cortical Dysplasia by Voxel-Based Morphometry*. Abstract from the 7th Annual Meeting of the Organization for Human Brain Mapping 2001, NeuroImage, Academic Press, 2001.

[Klei85] F. Klein and O. Kübler: *A Prebuffer Algorithm for Instant Display of Volume Data*. Proceedings of SPIE (Architectures and Algorithms for Digital Image Processing), Vol 596, pp. 54-58.

[Lacr95] Phillipe G. Lacroute: *Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation*. PhD thesis, Stanford University, September 1995.

[Lee97] Cheol-Hi Lee, Yun-Mo Koo, and Yeong Gil Shin: *Template-Based Rendering of Run-Length Encoded Volumes*. Proceedings of Pacific Graphics '97, pp. 139-147, 1997.

[Nibl86] Wayne Niblack: *An Introduction to Digital Image Processing*, ISBN 0-13-480674-3. Prentice-Hall ltd, 1986.

[Nish94] Dwight G. Nishimura: *Introduction to Magnetic Resonance Imaging*. Course notes, Stanford University, 1994.

[Sait93] Toyofumi Saito and Jun-ichiro Toriwaki: *Fast Algorithms for N-Dimensional Euclidean Distance Transformation*. Proceedings of the 8th Scandinavian Conference on Image Analysis, pp.747-754, 1993.

[Serr88] Jean Serra: *Image Analysis and Mathematical Morphology vol 2: Theoretical Advances*. ISBN 0-12-637241-1. Academic Press ltd, 1988.

[Smit00] Stephen M. Smith: *BET: Brain Extraction Tool*. FMRIB technical report TR00SMS2, Oxford Centre for Functional Magnetic Resonance Imaging of the Brain, 2000. See also appendix B.

[Tayl71] D.C. Taylor, M.A. Falconer, C.J. Bruton, and J.A.N. Corsellis: *Focal Dysplasia of the Cerebral Cortex in Epilepsy*. J. Neurol. Neurosurg. Psychiat., 34, 369-387, 1971.

[Zhan00] Yongyue Zhang, Stephen Smith, and Michael Brady: *Hidden Markov Random Field Model and Segmentation of Brain MR Images*. FMRIB Technical Report TR00YZ1, 2000.

[Wolf01] Stefan Wolff: *Simulation af MR-skanning* (in Danish). Project report, IMM, Technical University of Denmark, 2001.