This work should be cited as:
Kartaszyński, R. H. (2010). Four Dimensional Analysis of Perfusion in Brain. Ph. D. Thesis. Polish-Japanese Institute of Information Technology.

POLISH-JAPANESE INSTITUTE OF INFORMATION TECHNOLOGY

Department of Computer Science

Four Dimensional Analysis of Perfusion in Brain

Rafał Henryk KARTASZYŃSKI, MSc

P H D T H E S I S

Thesis Advisor:
Prof. Paweł MIKOŁAJCZAK, Ph.D.

Warsaw, 2010

Acknowledgments

To educate yourself for the feeling of gratitude means to take nothing for granted, but to always seek out and value the kind that will stand behind the action. Nothing that is done for you is a matter of course. Everything originates in a will for the good, which is directed at you. Train yourself never to put off the word or action for the expression of gratitude.

- Albert Schweitzer

Inspired by those words, I want to express my gratitude to my supervisor and, naturally to the radiologists from the Department of Radiology MSWiA Hospital in Lublin, Poland, for helping and guiding me through the process of writing this thesis. Thank you very much to Professor Paweł Mikołajczak for giving me insight into the field of image processing. He has always had time to discuss medical image processing problems and has been a major motivator for this thesis. Thank you Professor for giving me invaluable information and for arousing my interests in digital image processing. A big thank you to Doctor Michał Siczek, MD for the great help in the medical domain of my work. Without him the project described in this thesis would have never been initiated. It was he who proposed starting clinical research on perfusion analysis on MR images. Thank you, Doctor, for admitting me into this project and for all the interesting and motivating discussions we have had.

∼Thank you to my wife, parents and grandparents for always supporting and helping me ∼

Contents

Chapter 1

Introduction

Make no little plans:
they have no magic to stir men's blood...
make big plans, aim high in hope and
work.

- Daniel H. Burnham

Contents

1.1 Thesis and research target

Cerebrovascular diseases are often a cause of death (right after heart diseases and cancer). Very often there are ischemic changes, caused by blockage of arteries as a result of embolism or clot. Early diagnosis gives a great chance to reduce the consequences of stroke. Therefore, it is important to early detect acute cerebral ischemia. Nowadays, new techniques of cerebral blood flow visualisation are being tested in Poland. The most important of them are the perfusion studies obtained using computed tomography imaging and diffusion and perfusion magnetic resonance imaging. According to the unanimous opinion of experts, the application of these methods provides the basis for believing that it would be possible to detect ischemic changes in their earliest stages, when no irreversible changes have taken place in brain tissue yet.

Little use of perfusion CT and MRI in Poland is primarily attributable to the fact that most diagnostic imaging labs do not have suitable software or methodology. In

the labs, which have software for perfusion analysis the quality of such software is unsatisfactory.

To locate the brain areas with reduced blood flow (resulting from the narrowing of arteries leading blood to the brain or other diseases - such as Alzheimer's) tests are performed using perfusion MRI (or CT). At the time of their execution contrast medium is given, which "reveals" blood vessels and tissues reached by the blood with contrast. The images must be acquired with sufficient frequency to illustrate the dynamics of saturation of tissues with blood containing the contrast agent. To achieve this, it is necessary to reduce the resolution and take smaller number of slices in several series at equal intervals.

Sets of images, obtained in this way, are then analysed by a physician in order to identify areas where the density of tissue with contrast is changing rapidly or there is a delay in saturation, while compared to the healthy tissue. This is a four-dimensional analysis, as the data about blood flow in a volume of a brain (three dimensions) is measured in time (fourth dimension).

In perfusion studies the following parameters of cerebral circulation are assessed:

- volume of blood in the brain
- blood flow in the brain
- the average time of transition

In diagnostic and therapeutic procedures with patients with ischemic stroke, the following methods should be used:

- CT (necessary in order to exclude hemorrhage)
- CT perfusion studies
- MR perfusion studies with diffusion
- Time to reach the amplitude

We see clearly that in stroke treatment centres appropriate software to support analysis of CT and MRI studies should be accessible. Nowadays, a lot research power is directed to extend our knowledge and technical possibilities in perfusion diagnosis and cancer detection [1, 2, 3].

1.1.1 Objectives

The study presented here has been developed in cooperation with the Hospital of Ministry of Internal Affairs and Administration in Lublin, Poland. Thanks to such close collaboration, with medical specialists, we gained a deep knowledge of the subject and specificity of physicians‘ daily work. We also used information from literature: [4]. During our research, we have focused on developing tools that would be useful for them. Together we have formulated the following problem, which this research was to solve:

Theorem 1.1.1. *develop a new effective methodology for automatic (or semi-automatic) detection of the acute cerebral ischemia in MR perfusion study of the brain.*

Attention was also paid to accurate visualisation of the ischemic structures in time. An algorithm provides a visuals of:

- The map of regional cerebral blood volume
- The map of regional cerebral blood flow
- The map of the average transition time
- The map of the time to reach the amplitude

One of major problems of Polish health care is the obsolete diagnostic equipment and limited funds to replace it. This is especially the case in health units away from the capital. Often when a new MR unit is bought, there are no funds left to buy a fully functional DICOM workstation. While dealing with the main research goal, we wanted also to show that

Theorem 1.1.2. *it is possible to develop a non-expensive commercial-quality MR perfusion analysis application with very limited resources.*

In this case, the research team was limited to only one person, the author of this work. Even when resources were so limited, this thesis shows that it is possible to develop a high quality application that can compete with commercial solutions. Of course, most important for us was the scientific aspect of this research. We were therefore using highly-developed programming languages (C# on MS Visual Studio) supported by open source packages (for example, to open and manage DICOM files) to make the implementation as little time consuming as possible.

While researching into the medical objectives of this thesis, we have introduced another subject we wanted to show:

Theorem 1.1.3. *it is possible to accurately and automatically, compare two completely different brains in the way, that takes into consideration shape and anatomical structure of the human head.*

This theorem my be considered as the main outcome of our research, from the point of view of computer science. Our main goal is more of the application of our "computer" tool to help solve a real-life problem, which is also something new from the health care point of view.

It is important to mention an assumption we have made, which is a direct consequence of the methodology of performing perfusion examination. The Statistical Perfusion Brain Model or symmetry tool are very sensitive, as hemodynamic parameters are sensitive to technique of study acquisition and equipment used. Therefore it would be best to use studies done in one hospital, with one methodology. It will be very difficult to get studies acquired with the same technique from two hospitals, as they use different equipment and methodology, making unification almost impossible. So the best solution is to restrict oneself to one hospital and its approach. The software would be used in one hospital, anyway, and program could be "trained" using studies from that institution before use. Therefore, while developing some algorithms, we have assumed that, methods will not have to be universal and will be automatically tuned to cope with different conditions in different hospitals.

To reach the thesis‘ main goal, we have proposed two new solutions. First, using the symmetry information to find ischemic changes. The idea is that ischemic changes (for example a stroke) most often strike only one side of the brain. So if we compare perfusion parameters on both sides of the brain, we should be able to pinpoint the location of any differences outside a given margin. This idea was developed and implemented as a semi-automatic tool that aids doctors in finding abnormalities in perfusion parameter behavior.

The second proposition is to use statistics that will allow for automatic detection of abnormalities in cerebral blood flow (on MR perfusion studies). Our idea was to create a statistical model of the cerebral circulation in a healthy brain. In other words, the model would tell us how blood flow should behave in a given part of the brain. Patient studies suspected with the cerebral ischemia are compared with the model, and all deviations, from the norm can be pinpointed and brought to the physician's attention. As this is a powerful statistical model, it can also give us much information [5].

Together these two methods produce very good, promising results. This has been proved by the preliminary testing. To fully assess these techniques, from the medical

point of view, further clinical studies are required. They will be carried out by the hospital we are cooperating with and are intended to be the subject of our future investigations. At this point, we are not able to provide very accurate data regarding statistical model, as it requires many healthy patient studies to create. This takes time and requires deep involvement of the medical team. At the moment, the model was built using several data sets.

From the computer science point of view, the project has been successfully realised. Approaches have been designed, implemented and tested. We would like to stress that many algorithms used in this thesis present new quality. They have received favourable reviews and were published in national and international, highly regarded journals. Some citations are provided in the following section, where we describe, briefly, each chapter.

1.1.2 Contents

I have divided this thesis into thee chapters and three appendixes. All images used here were prepared by me using software developed as part of my research. Whenever I used pictures from other sources (few cases), I have put appropriate references.

In the following part of this chapter, medical and physical basics of the perfusion imaging will be presented. I have included only most important knowledge in this chapter. This should give the reader more information about these issues and the importance of the MR perfusion diagnostic and this research.

In the next, second chapter, we are presenting segmentation methods designed and used during our research. Some of these methods are modifications of well known algorithms and some are new approaches, designed strictly for this research.

Some space is also dedicated to the development of accurate and effective interpolation methods for two and three-dimensional images. This is a very important issue, as interpolation methods play a significant role in 3D data reconstruction and visualisation and our research has yielded some interesting results on that score [6, 7].

Particularly noteworthy are the "Combined T1 and T2 MR brain segmentation", which produces very accurate results [8, 9], and a very interesting method, based on cellular automata, which has many practical applications [10, 11, 12, 13]. The third method developed by us is the automatic eyes segmentation and detection algorithm. This is a recent development and has not been publicised yet.

The third chapter is the most important one. It is completely devoted to the perfusion analysis. We present here some very interesting and new algorithms that were developed to reach the goal. First is the innovative method that reduces noise influence in calculation of hemodynamic parameters [14, 15]. This makes visualisation of perfusion parameter maps very accurate and easier for further analysis.

As mentioned, our semi-automatic technique, among other things, compares perfusion parameters between two hemispheres, because a stroke often affects only one of them, while the other is functioning properly. To do such a comparison we needed a good algorithm that searches for symmetry lines on low resolution perfusion images [16].

Next we present the methodology used to automatically evaluate the statistic perfusion brain model (SPBM), which we consider our main achievement in this thesis. The main difficulty of this subject is to find the way to compare brains of completely different people. To do this, we have developed a method that finds three characteristic planes of the head, without using any external markers - only anatomical structures in the head. These three perpendicular planes (called by us Brain Characteristic Vectors) together with their intersection (the Brain Characteristic Point) allow to accurately align completely different brains.

We are also presenting algorithms that allow us to express position of the 3D point in one brain as a function of the point coordinates in the second, completely different, brain. This technique allows us to find for each voxel in one study a corresponding point in shape and anatomy aware manner in the second study. Our method proves to be very accurate in comparing different brains, which may be found useful not only for the purpose of this research. We want to stress that this method takes into consideration the shape of the brain and its anatomy; therefore, this is a unique technique that does not involve any additional external markers or study acquisition routines. It uses standard studies acquired during daily work.

Finally, we present the SPBM and results of our test. At the end of the third chapter we present final results and conclusions. In this case, we also have not managed to publish them yet.

In appendix A we present some features of the application (DICOM workstation) created as a tool during our research. This is a short presentation from the software engineering and educational point of view. This part of our research was publicised [17, 18].

In appendix B, DICOM standard is presented. This is purely a theoretical chapter. The most important part is a discussion and a description of the *scanner / patient*

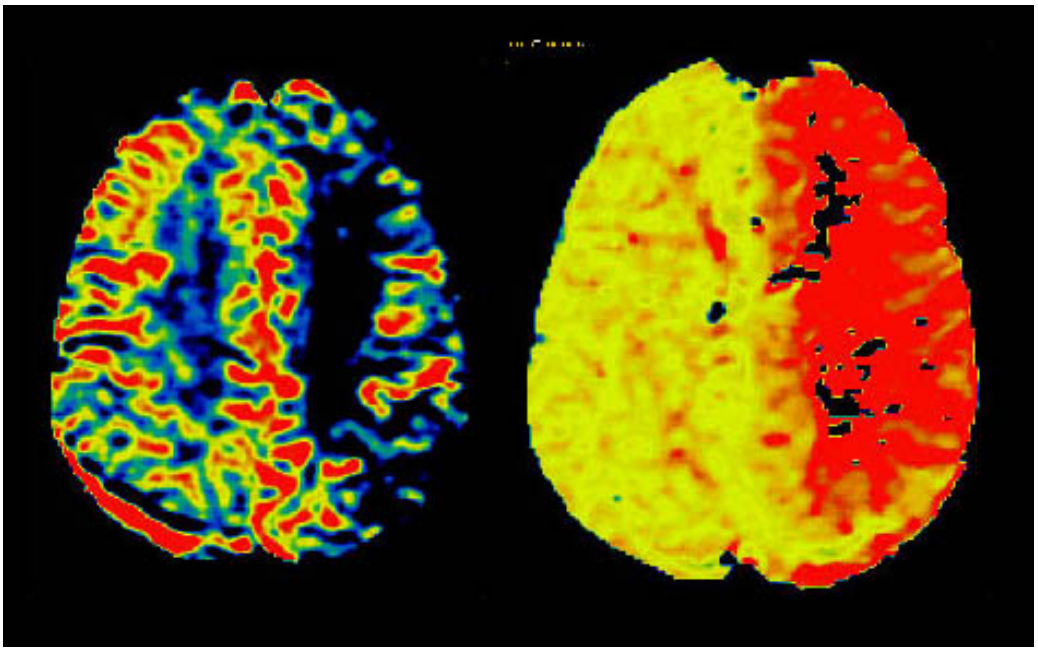

Figure 1.1: Left: Perfusion-weighted MRI of a patient who presented 1 hour after onset of stroke symptoms. Right: Mean transfer time (MTT) map of the same patient. [23]

coordinate system vs. *image coordinate system*.

Appendix C is strictly devoted to digital image processing, its fundamentals, frequently used techniques and algorithms. It should be stressed that all algorithms and operations presented in this chapter were implemented and used in our research. Furthermore, not all implemented methods have been described here, because they are not directly related to the thesis‘ subject.

1.2 Perfusion imaging - short medical background

Perfusion MR studies provide an absolute and/or relative measurement of the cerebral microvascularisation parameters: regional blood volume, mean transition time, regional blood flow [19, 20, 21, 22]. They rely on the use of a tracer / contrast agent that may be exogenous (Gadolinium) or endogenous (spin labelling). Figures and show model MR images in acute stroke.

The first pass technique uses the magnetic susceptibility effect of the Gadolinium chelates, with T2* weighting or T2 weighting, in GE-EPI or SE-EPI sequences. Very important for a good examination are the quality of the injection and the timing of acquisition. The drop of the signal during the first pass of agent makes it possible

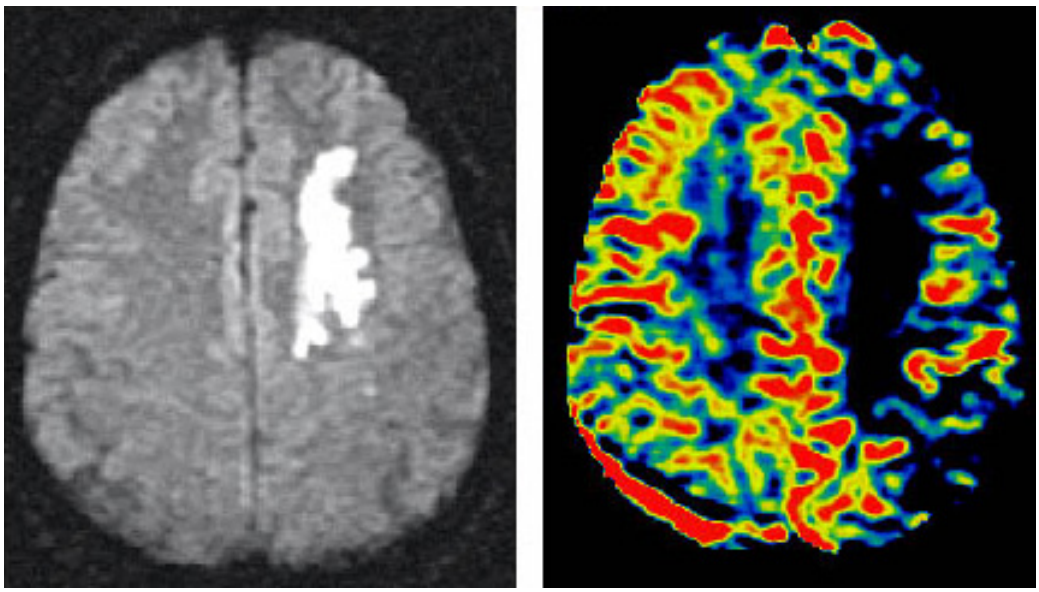

Figure 1.2: Diffusion-perfusion mismatch in acute ischemic stroke. The perfusion abnormality (right) is larger than the diffusion abnormality (left), indicating the ischemic penumbra, which is at risk of infarction. [23]

to extract perfusion parameters after post-processing. The spin labelling technique [24, 25, 26] uses saturated blood just upstream of the slice of interest as a tracer. Saturation causes variation in received signal in relation to one without prior saturation. The estimation of the local hemodynamic parameters can be evaluated by comparing these two signals. Imaging is carried out in echo planar sequence and different labelling techniques are possible. It produces only weak signal-to-noise ratio and is restricted to the analysis of a limited ROI (Region Of Interest). The main applications of the first-pass perfusion MR are vascular pathologies (vasospasm, ischemic strokes) and tumoural pathologies.

Perfusion MR gives us access to the information about the capillary microcirculation in tissue. The main quantitative parameters we measure are: blood volumes and temporal data (mean transit time, time to contrast peak...). The main goal of perfusion MR is to measure and assess the blood flow in the explored organ, which is expressed in millilitres per one hundred gram of a tissue per minute. This corresponds to micro-circulatory tissue perfusion rather than the flow of the main vascular arteries.

The possibility to differentiate between perfused and non-perfused tissue depends on the use of an intra-vascular tracer that can be:

- exogenous: contrast agent is injected (most often non diffusive, present only in the vascular space and will not cross the normal blood-brain barrier).

- endogenous: hydrogen nuclei in water circulating in the blood is traced (tracer is diffusive, there is an exchange between extra- and intra-vascular parts).

1.2.1 First-pass contrast-enhanced dynamic perfusion imaging (DSC MRI)

Principles Gadolinium chelates are used in perfusion MRI because of their magnetic susceptibility effect at high concentration: the contrast agent in vessels causes the diversity in the magnetic field, which leads to decrease in relaxation times T2* and T2 of surrounding tissues.

The reduction of the signal measured, during the first pass of the contrast agent, depends on its concentration in vessels, diameter and number of vessels (per volume unit) and type of signal weighting. T2-weighted sequences (ES-EPI) are more specific to the microvascular part than T2*-weighted sequences (GE-EPI) that also take into consideration larger vessels.

The injection is done at fast rate to deliver a contrast agent as quickly as possible (as close to instantaneous delivery as possible). At the same time a dynamic acquisition in echo planar sequence (ES-EPI, GE-EPI) is done. This measures the signal before (baseline), during and after the first-pass of the bolus. The signal is monitored over time with typical temporal resolution of the order of seconds (see fig. 1.3). The bolus itself will pass entirely through the head in around ten seconds.

As the size of capillaries is unmeasurable, all analysis techniques assume a direct relationship between the signal loss and the total contrast volume within a voxel. The assumption of a linear relationship between relaxation rate and concentration of the contrast agent has been shown to be valid both by experiment and simulation for the blood volume fractions in the physiological and pathological range [27].

To calculate perfusion parameters, post-processing of signal curves must be done. Figure 1.4 shows visualisation of graphical interpretation of the calculated perfusion parameters. There are two main techniques used:

- Modelling of the signal curve (by gamma curve fitting) that eliminates the contribution of the tracer recirculation - this technique gives only a relative quantification of perfusion parameters.
- Deconvolution with the arterial input function (AIF) that takes into account the diffusion of the bolus in time and the patient‘s cardiac function, if with errors caused by the distal spread of the bolus and differences in the hematocrit between large and small vessels. This causes variations in proportions of the plasma volume in which the tracer is diffused.

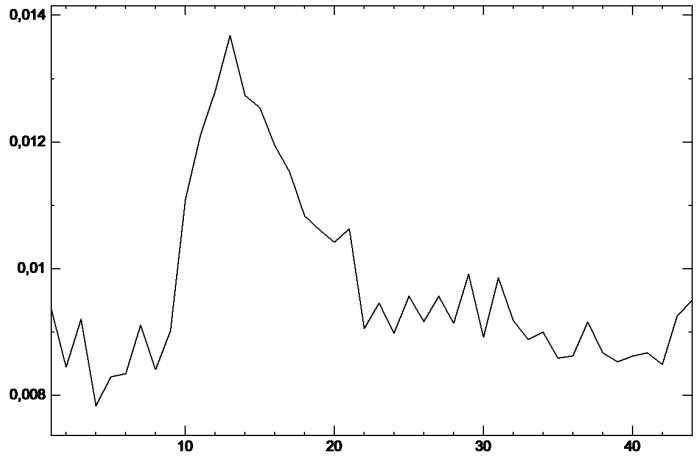

Figure 1.3: Example of concentration time curve. The X-axis represents time (consecutive MR scans) and the Y-axis shows the CTC value calculated at a given time (bolus).

The mathematical formula on the calculation of the most important perfusion parameters are presented in the chapter 3.2.

Parameters measured

- TA: time of arrival of the contrast agent in the slice after injection
- TTP (Time To Peak): time corresponding to the maximum contrast variation
- MTT (mean transit time)
- Peak amplitude: percentage loss of intensity of the maximal signal
- rCBV (regional cerebral blood volume): index of cerebral blood volume determined by the area below the signal curve
- rCBF (regional cerebral blood flow): index of cerebral blood flow corresponding to the rCBV/MTT ratio.

Values rCBV and rCBF are relatively measured, providing the ratios between the pathological region and a healthy one (that serves as a reference). Relative quantification of these measurements uses a deconvolution by the AIF (arterial input function) that is obtained from the signal of the large arterial vessel present in the image. This takes into consideration the imperfect nature of the bolus (not instantaneous) and the patient‘s cardiac function.

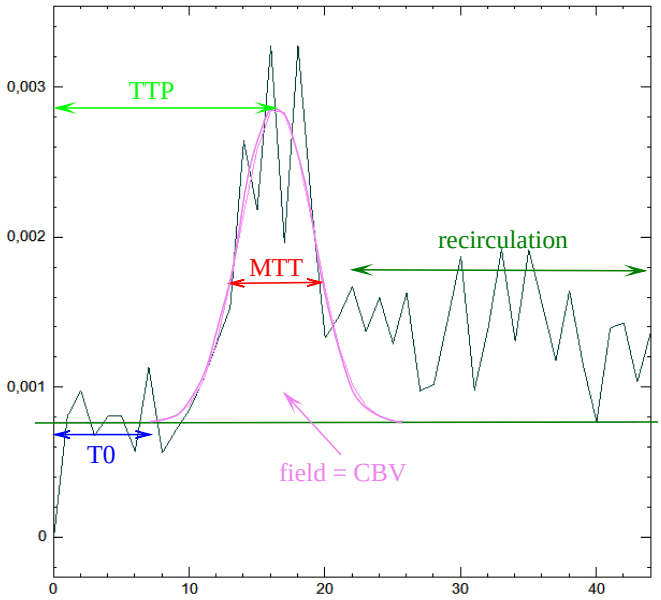

Figure 1.4: Gauss curve fitted into most significant part of the bolus with most important perfusion characteristics marked. The X-axis represents time (consecutive MR scans) and the Y-axis shows the CTC value calculated at a given time (bolus).

Optimization The quality of perfusion imaging can be affected by many factors:

- The bolus must be injected very quickly. A strong concentration of the tracer will cause a more considerable drop of signal. The venous access quality, injection speed and the patient‘s hemodynamics must be taken into consideration.
- The total acquisition time must include reference images before agent injection, the first-pass of bolus in the whole volume, without taking too long (recirculation effect).
- The patient must be stable, movements must be avoided, as it can cause calculation errors in the parametric images.
- Gadolinium chelates do not normally cross the blood-brain barrier. In pathological situations, where this barrier is crossed, the tracer produces extravasation that reduces the T1 of the tissue. It spoils the perfusion signal and leads to the risk of miss-calculating the cerebral blood volume.
- The sequence type (GE-EPI / ES-EPI) will affect microvascularisation sensitivity (better in ES-EPI, as TE raises, to the detriment of the signal-to-noise ratio) and the signal-to-noise ratio (better in GE-EPI).

Main applications of perfusion MRI Thanks to the possibility of studying microvascularisation with perfusion MR, it is used in [28]:

- vascular diseases: ischemic stroke, study of vasospasm in subarachnoid hemorrhage
- tumoural diseases: perfusion MR is used to assess neoangiogenesis and tumoural vascularisation - the diagnosis or aftercare treatment of some nervous tumours (high grade glioma, lymphoma meningioma, metastasis, pilocytic astrocytoma...)
- infectious or inflammatory diseases, one of whose characteristics can be studied in perfusion MR.

Perfusion MR techniques, in the first-pass and by spin-labelling, require high temporal resolution, an appropriate signal-to-noise ratio with the highest spatial resolution available.

Chapter 2

MR segmentation and processing methods

What I have learnt is but a handful of earth. What is left unlearnt is the earth itself.

- Tamil proverb

Contents

In this chapter we present some image processing and segmentation methods used during our research. We present methods developed by us to reach the thesis‘ goal. Segmentation methods shown here are used later to find exact borders of the brain or remove unnecessary background from MR images. We have developed methods that are automatic or semi-automatic and can be used depending on our current requirements.

Image analysis combines techniques that compute statistics and measurements based on the grey-level intensities of the image pixels [29, 30, 31]. In the case of medical image processing of MR data, we are dealing only in cases where no color information is acquired during study acquisition. Thanks to image processing segmentation techniques [32, 33, 34, 35] we can extract important features or information from the image that can be used later by more complex algorithms.

Before we move on to very important segmentation techniques, we will discuss very interesting research into interpolation techniques used while dealing with two- and three- dimensional medical data. We have used this knowledge while developing improvements to perfusion parameters calculation algorithms and the Statistical Perfusion Brain Model evaluation.

2.1 Two- and three-dimensional interpolation of medical images

Diagnostic devices such as CT, MRI, PET play an important role in modern medicine. They produce two-dimensional sections of human body, which may be later processed and analysed. Many algorithms and methods require execution of operations such as enlargement (which alone is very important for every radiologist), rotation, deformation, etc. All these transformations require calculation of a pixel value in a real point between two discrete ones (see Figure 2.1).

Two-dimensional images are often not sufficient enough for diagnostic purposes. There are series of applications in which three-dimensional representation is needed. Determinations of mutual spatial situations of different body structures, or calculation of their size (volume, surface) are situations when the standard slice is insufficient. Of course the more slices are available the more accurate the mapping is.

The optimal number of slices is when the real size of pixel is almost equal to the distance between slices (isotropic resolution). Unfortunately, in practice it is not possible to make studies precise (dense) enough due to pure technical reasons and patients‘ safety (radiation dose). The typical approach is to interpolate additional slices, so that the smallest indivisible cuboid (called voxel) achieved from a set of two dimensional images will have a shape resembling a cube.

2.1.1 Interpolation fundamentals

The quality of the interpolation method used can be judged in many different ways - starting with an assessment of the visual effect, and ending with statistic measurement. We would like to present some results of our previous research concerning the above subject [6, 7]. First, let us discuss the general idea of interpolation.

Lets assume that we want to count the function value (not necessarily discrete) in $x \in R$, knowing its values at discrete points [36]. We may write the value of function I as follows:

$$I(x) = \sum_k I(k) \cdot K(x - k) \tag{2.1}$$

where $K(x)$ is a continuous function called interpolation kernel. We are summing after all k points neighboring x, furthermore we may assume that the distance between each point k is unit length.

It's a general formula for calculation of a value of one dimensional discrete signal in a real point. Because in most cases kernels are symmetrical and separable, we can write instance of two dimensional signal:

$$I(x, y) = \sum_k \sum_l I(k, l) \cdot K_{2D}(x - k, y - l) \tag{2.2}$$

where $K_{2D}(x, y) = K(x) \cdot K(y)$, and analogically for three dimensions

$$I(x, y, z) = \sum_k \sum_l \sum_m I(k, l, m) \cdot K_{3D}(x - k, y - l, z - m) \tag{2.3}$$

where $K_{3D}(x, y, z) = K(x) \cdot K(y) \cdot K(z)$.

In our deliberations, we will understand signal $I(k, l)$, or $I(k, l, m)$ as intensity, in grey scale, of a pixel in a discrete point (k, l) for two-dimensional image and a voxel at a point (k, l, m) for three dimensions.

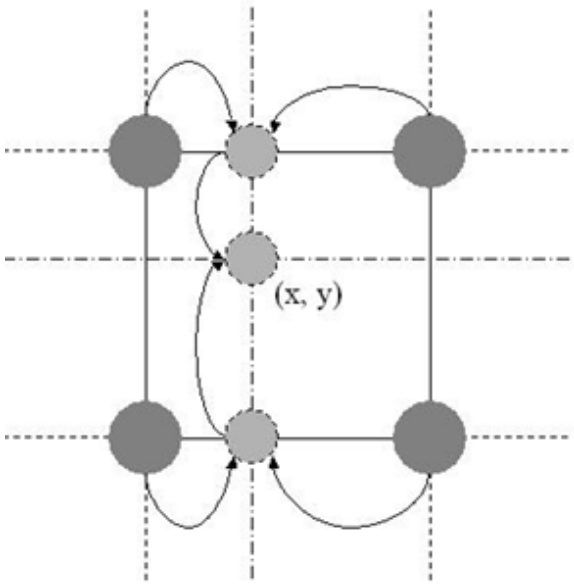

Figure 2.1: One dimensional decomposition of two dimensional interpolation NxN at a point (x, y).

In Figure 2.1, we can see an illustration of the method for calculation of a pixel value at a point (x, y) based on neighboring four pixels. First, intensities at a points $(x, \lfloor y \rfloor)$, $(x, \lceil y \rceil)$ are estimated. Secondly, they are used to interpolate pixel at (x, y). In the three-dimensional case, the operation is similar: calculation of voxels in a plane above and below the searched point is done in the way described earlier and interpolation along the z axis is executed.

2.1.2 Methods survey

In our research we have tested many interpolation kernels: nearest neighbor, linear, cubic, cosine, Hermit (in original, James McCartney's and Laurent de Soras version), spline (in six-points, cubic, B - basic, Catmull-Rom version) and sinc (with windows: rectangular, Blackman - Harris, Lanczos, Welch, Parzen, Hann - Hamming, Gauss). We will present some of them in table 2.1.2.

These are most commonly used functions for data interpolation. They estimate a value at a point according to the intensities of various number of neighbors. The simplest ones are based on two points, whereas the more complex (for instance sinc) on almost an infinite number of neighbors. As far as sinc function is concerned, it is defined to be infinite; therefore, it must be multiplied be an adequate window function. Research has shown that best results are obtained for sinc windows of different parity size [44]. Unfortunately, there is one disadvantage, it executes rather slowly, compared to other, simpler methods, including spline (see Figure 2.2). There-

Method name	Formula	Ref.
Linear interpolation	$K_{lin}(x) = \begin{cases} 1-\lvert x\rvert, & 0 \le \lvert x\rvert < 1; \\ 0, & \text{else.} \end{cases}$	[37]
Cosine interpolation	$K_{cos}(x) = \begin{cases} \frac{(1-(1-\cos(\lvert x\rvert\cdot\pi)))}{2}, & 0 \le \lvert x\rvert < 1; \\ 0, & \text{else.} \end{cases}$	[38]
Cubic interpolation	$K_{cub}(x) = \begin{cases} (\alpha+2)\lvert x\rvert^3 + (\alpha+3)\lvert x\rvert^2 + 1, & 0 \le \lvert x\rvert < 1; \\ \alpha\lvert x\rvert^3 - 5\alpha\lvert x\rvert^2 + 8\alpha\lvert x\rvert - 4\alpha, & 1 \le \lvert x\rvert < 2; \\ 0, & \text{else.} \end{cases}$	[39]
Spline Catmull-Rom interpolation	$I(x) = 0.5 \cdot \begin{pmatrix} 1.0 & x & x^2 & x^3 \end{pmatrix} * \begin{pmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{pmatrix} * \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$	[40]
Sinc interpolation	$K_{sinc}(x) = \frac{\sin(\pi\cdot x)}{\pi\cdot x}$ $K_{Nsinc}(x) = \begin{cases} (K_{sinc}(x)\cdot w(x), & 0 \le \lvert x\rvert < \frac{N}{2}; \\ 0, & \text{else.} \end{cases}$	
Welch window	$w_{welch}(x,r) = \begin{cases} 1-(\frac{x}{r})^2, & \lvert x\rvert < r; \\ 0, & \text{else.} \end{cases}$	[41]
Hann-Hamming window	$w_{hh}(x,r,\alpha) = \begin{cases} \alpha + (1-\alpha)\cdot\cos(\pi\cdot\frac{x}{r}), & \lvert x\rvert < r; \\ 0, & \text{else.} \end{cases}$	[42]
Lanczos window	$w_{lanczos}(x,r) = \begin{cases} \sin(\pi\cdot\frac{x}{r})/(\pi\cdot\frac{x}{r}), & \lvert x\rvert < r; \\ 0, & \text{else.} \end{cases}$	[41, 43]

Table 2.1: Survey of interpolation methods, tested in our research, with corresponding references

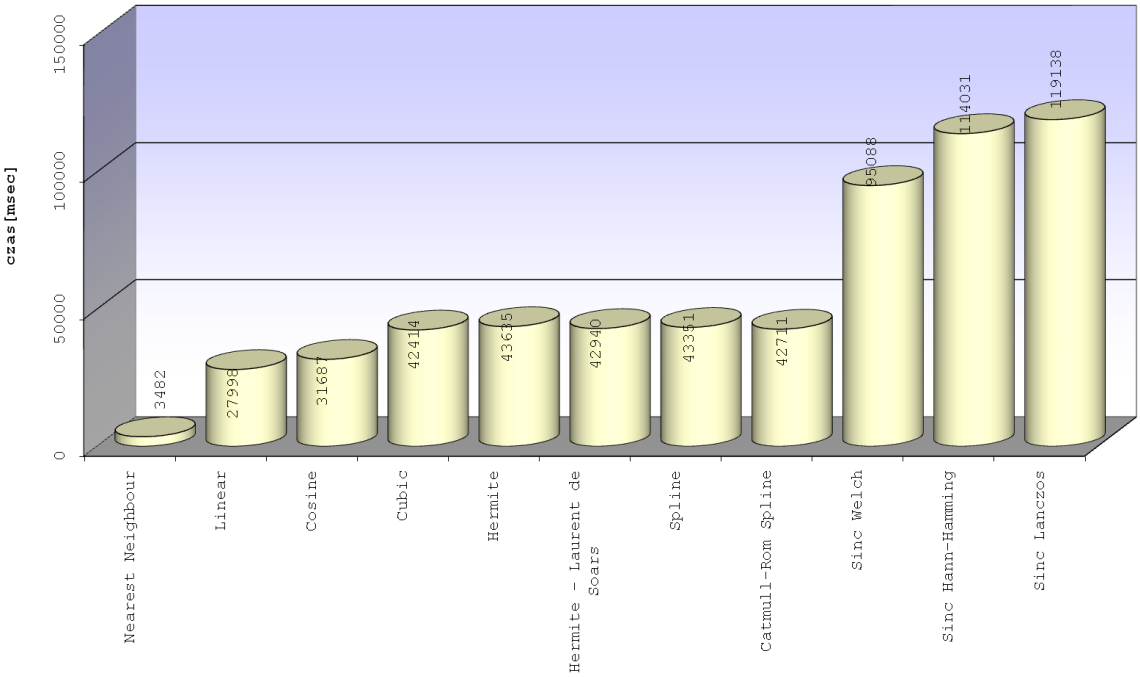

Figure 2.2: Comparison of execution times for several interpolation methods (from our previous investigation [7]).

fore, Catmull-Rom interpolation is often considered to be more profitable than sinc because of shorter execution time and similar results [45].

2.1.3 The search for the best interpolation method

The aim of our previous research [6, 7] was to search for optimal interpolation technique to calculate volume of anatomical structures, as well as development of an accurate enlargement method. Research was based on 231 CT head scans with spatial resolution of 512x512, from Visible Human Project [46]. The object of interest was the volume of a skull segmented by the thresholding method. According to the main data set, several smaller sets were interpolated. These sets were then again interpolated to original size. From images obtained this way, voxels belonging to the skull were selected. The quotient of the volume from the interpolated sets to the original volume was calculated. The average distance of voxels belonging to segmented region to those from original set was also measured (average Euclidean distance of point from set).

Apart from the obvious conclusion the more slices we have, the closer we get to original volume, with the average distance nearing to zero, the results obtained show that for this goal selection of interpolation technique doesn't have important

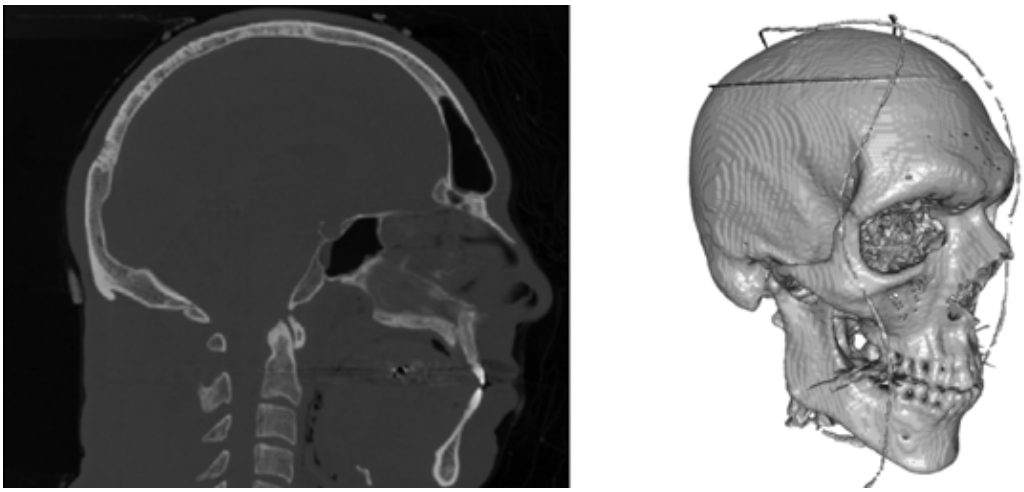

Figure 2.3: Original data: crosswise section of CT set and surface reconstruction according to segmented data (from our previous investigation [7]).

meaning. Interesting fact is, that for a small number of slices any interpolation significantly increases the difference between original and calculated volume. There are several reasons for this. Firstly, interpolation is done only along the z axis, without taking into consideration any deviations, which leads to the interpolation of values of different structures and background between one another and to the appearance of numerous distortions. For large real distances between slices (corresponding to the real size of pixel) the use of interpolation methods that takes larger number of neighbours introduces significant errors (not to mention computation cost, see Figure 2.2). In such cases, their use is pointless and even wrong.

The 'stairs' effect visible between source slices cannot be removed by such an approach to interpolation. However, in many situations an image visually presents itself better, but as calculations have shown, information contained in the data is demoted. For that reason, a new approach to interpolation must be found. Methods based on a segmented object, not on the whole set, can give better results. It removes the problem of structures intensities 'mixing', but the problem of interpolation along the one z axis remains unsolved.

The main aspect differentiating the above described interpolation methods from the method based on the shape is that this method does not operate according to the intensity of the whole set, but only on segmented region of interest. The method consists of several stages:

- obtaining contour of segmented structure (thresholding + morphological operations)
- generation of distant maps for each slice

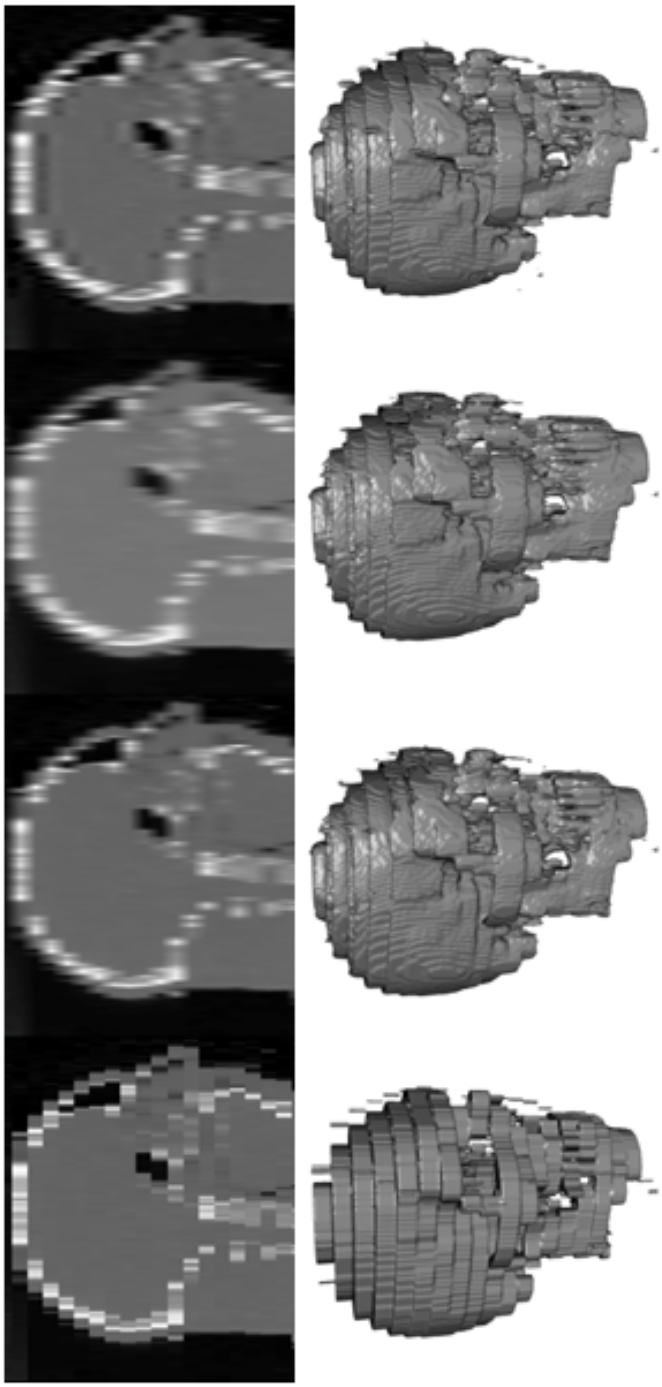

Figure 2.4: Samples of images created based on 20 slices using standard interpolation approach, together with 3D reconstruction. Interpolation kinds(from left): nearest neighbour (lack of interpolation), linear, spline, sinc Welch. Despite better looking surfaces, after using interpolation methods, calculated structure volume differs from original in degree more than for non interpolated set (from our previous investigation [7]).

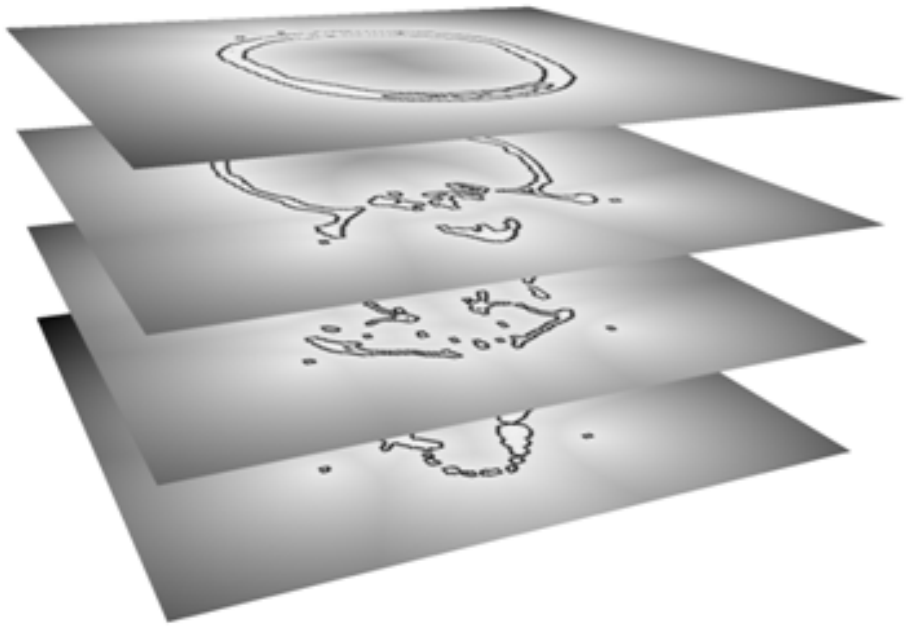

Figure 2.5: Sections containing contours of anatomical structures together with distant maps (from our previous investigation [7]).

- linear interpolation of slices to the desired data set size

First, contour of segmented structure is generated, creating binary image. Next, for all points in the slice its distance to the nearest point of contour is counted. Points inside contour have positive values, while those outside are negative (see Figure 2.5).

As a result, we obtain something shaped as three dimensional distance map, which after setting threshold to zero and segmentation gives the final object. Although interpolation was done only for z axis, thanks to using distance maps it was in fact done in all three, perpendicular directions. Results of obtained volumes show that this method is better suited for anatomical structures shape reconstruction from thin slices. The shape of the structures resembles the original, and the 'stair' effect was greatly reduced or at least softened. However, it is not a perfect method because for some cases it may cause unnecessary connecting (disconnecting) of voxels earlier disconnected (connected), which may undermine the image. Besides, it works only on binary, previously segmented objects, which greatly limits the scope of its application.

A method based on distant maps may be an interesting alternative. There are works confirming usefulness of this technique, and often it is the basis for more advanced techniques allowing processing not only of binary images, but also original, not processed data. Currently we are working on different interpolation methods

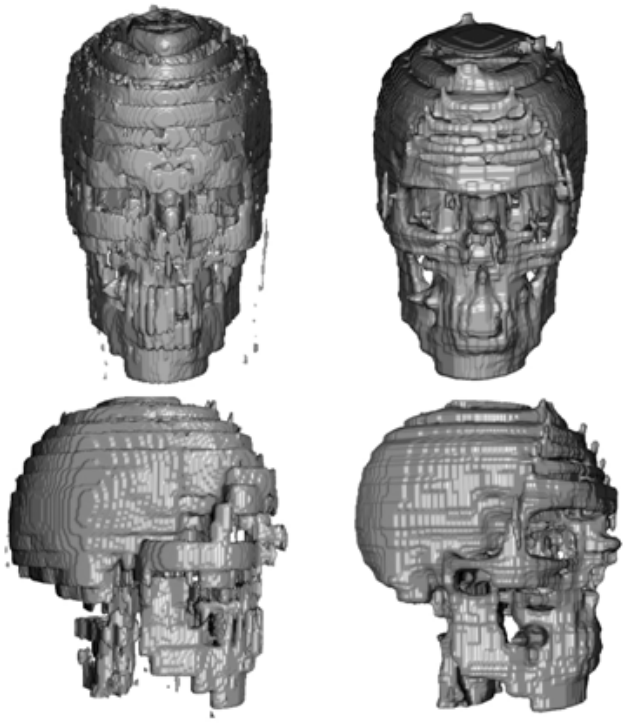

Figure 2.6: Comparison of results of conventional interpolation (left) and interpolation based on the shape (right) (from our previous investigation [7]).

for medical data sets, not requiring pre segmentation. Results are appearing to be promising.

2.2 Thresholding

Grey level images thresholding is the simplest segmentation approach. Image regions or objects can be described by their brightness/intensity or density (medical images). This method is based on a simple idea. We use a parameter T called the threshold and apply it to the image $I(x, y)$ in the following way

$$\begin{array}{ll} I(x,y) \geq T & for I(x,y) = I(x,y) \\ I(x,y) = B & else \end{array} \tag{2.4}$$

where B is a background pixel value (mostly equal to 0).

The output will be an image containing only pixels (or voxel) of brightnesses greater than or equal to T, the rest of the pixels will have value equal to the background (B). Of course, the test conditions can differ and be based on different features, but the idea is clear [47].

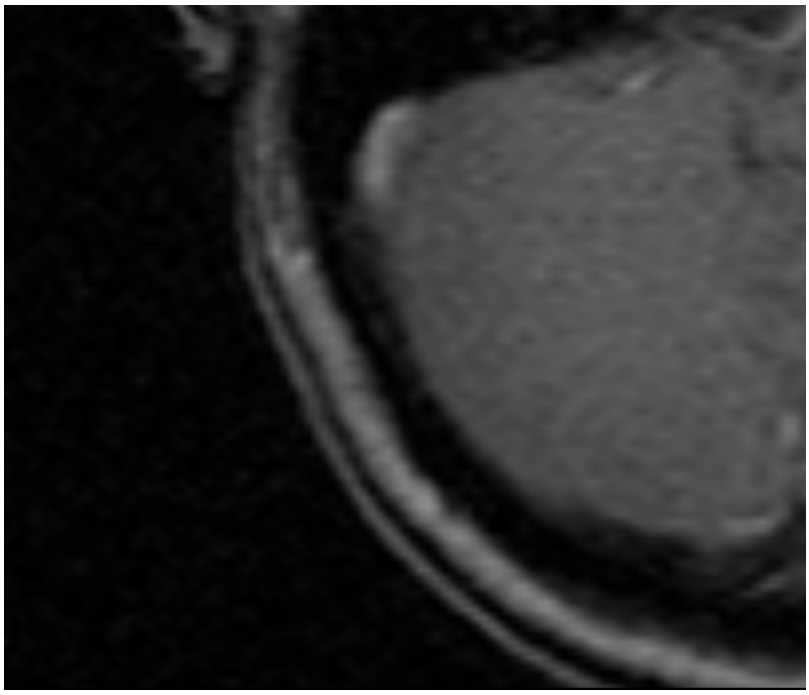

Figure 2.7: Sample MR brain study, compare with Fig. 2.9 after thresholding.

Chow and Kaneko [48] have developed a variation of thresholding method where image is divided into non-overlapping regions. A threshold is calculated in each region and threshold values obtained are then interpolated to form a thresholding surface for the whole image.

2.2.1 MR background removal

While dealing with MR brain images, the first and most important and simple step is to remove background noise of the image. By noise, in this case we understand all the "static" that can by observed outside the body tissues (see Figure 2.7). The best way to get rid of it is to apply a simple thresholding operation to each slice in the display set. To choose which intensities should be removed we've analyzed histograms of those slices. Generally, their shape ia as shown in Figure 2.8.

The high peak on the left side of the plot represents numerous low frequency pixels of the background (also meninges between brain and bone-skin tissues). They are followed by minimum denoted in Figure 2.8 as t_b. Next, is the rest of the image data. Therefore t_b is the best place to set the end of the thresholding range, which, of course, starts with 0 intensity.

The method for finding this local minimum is rather straightforward. Considering the average shape of the histogram plot, this minimum is always located in the first

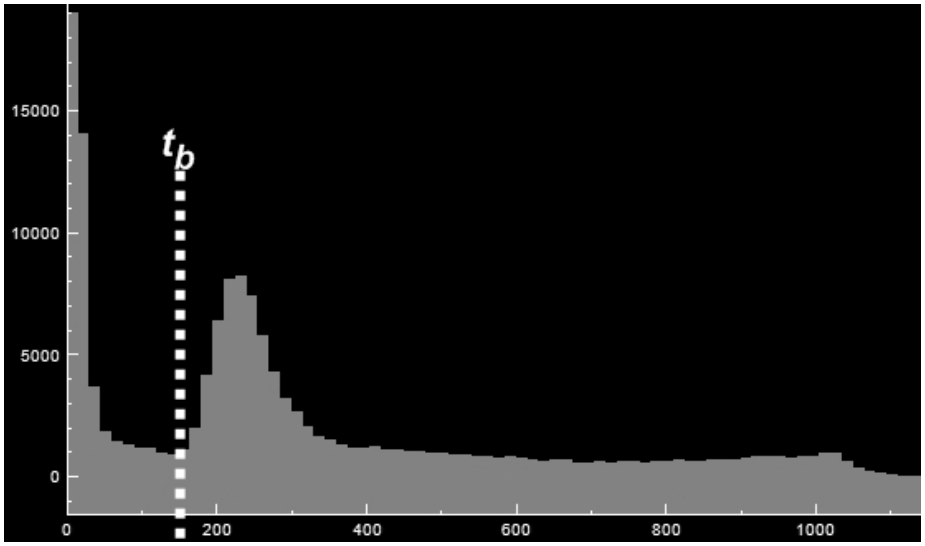

Figure 2.8: General shape of the histogram plot of MR brain slices. The X-axis represents pixel intensity and the Y-axis shows the number of pixels.

1/5 part of the diagram. For this part it is a local minimum.

2.2.2 Perfusion MR background removal

This is an upgrade to the previous method (see Section 2.2.1). On the general histogram of the MRI perfusion slice we can find two main peaks. The first peak represents numerous low frequency background pixels. With an increasing intensity of pixels (to the right of the diagram) the number of pixels decreases reaching local minimum (S_0) and increasing again to reach another local maximum (S_b). In a general case of MRI study, segmentation at the level S_0 would be sufficient, but in the case of the perfusion images we need a "deeper" segmentation. Therefore, we have set the threshold value at $S_P = S_0 + \frac{(S_B - S_0)}{4}$.

2.3 Cellular Automata Tissue Segmentation

In this section we present a new approach to the MRI brain segmentation. We have used it in our research to segment data that could not be automatically segmented using our other methods. It also proves to be very useful in physician daily work.

The method is based on the Cellular Automaton (CA). It is truly interactive, and produces very good results comparable to those achieved by Random Walker [49] or Graph Cut [50] algorithms. It can be used for CT and MR images, and is accurate for various types of tissues. The version of the algorithm discussed here is developed especially for the purpose of the MR brain segmentation. The method is extensible,

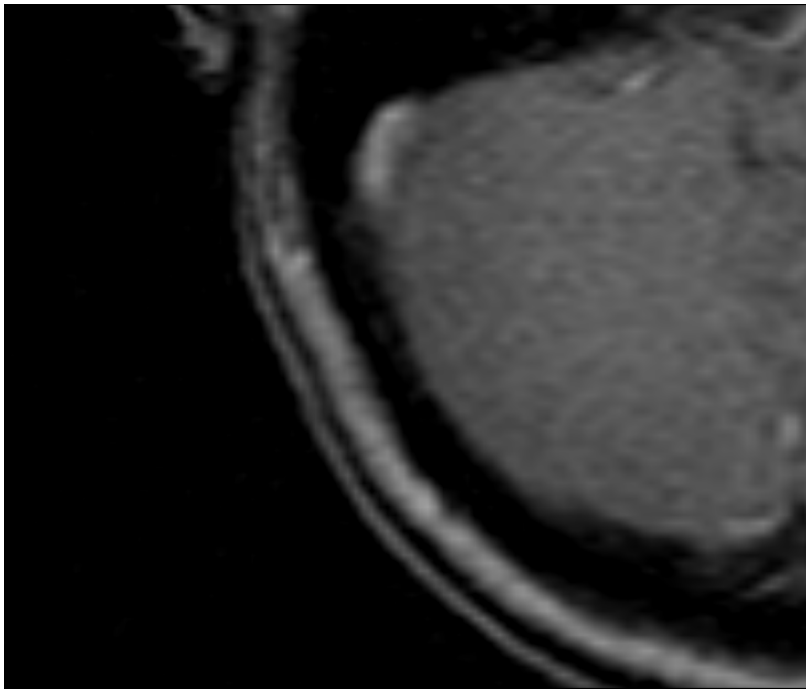

Figure 2.9: MR brain slice after thresholding.

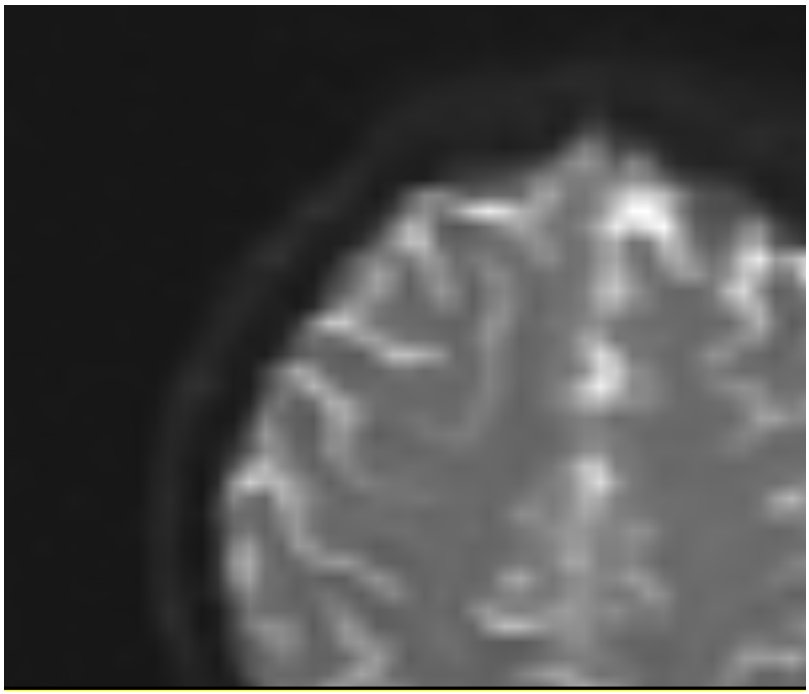

Figure 2.10: Sample MR perfusion brain study, compare with Fig. 2.12 after thresholding.

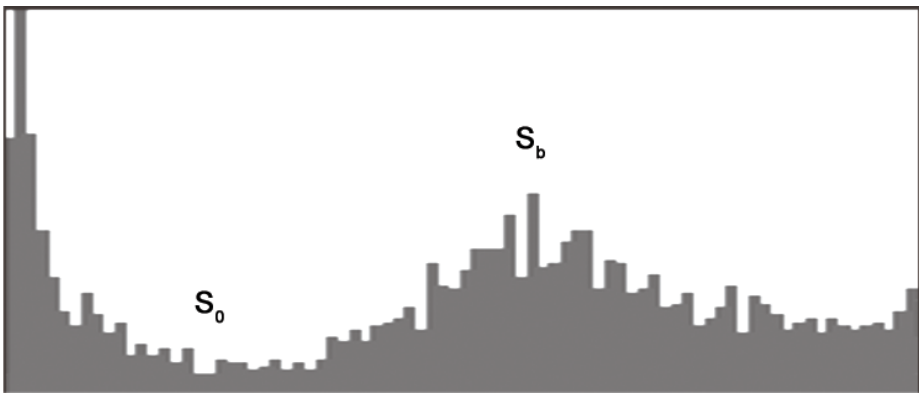

Figure 2.11: General shape of the histogram plot of perfusion MR brain slices. The X-axis represents pixel intensity and the Y-axis shows the number of pixels.

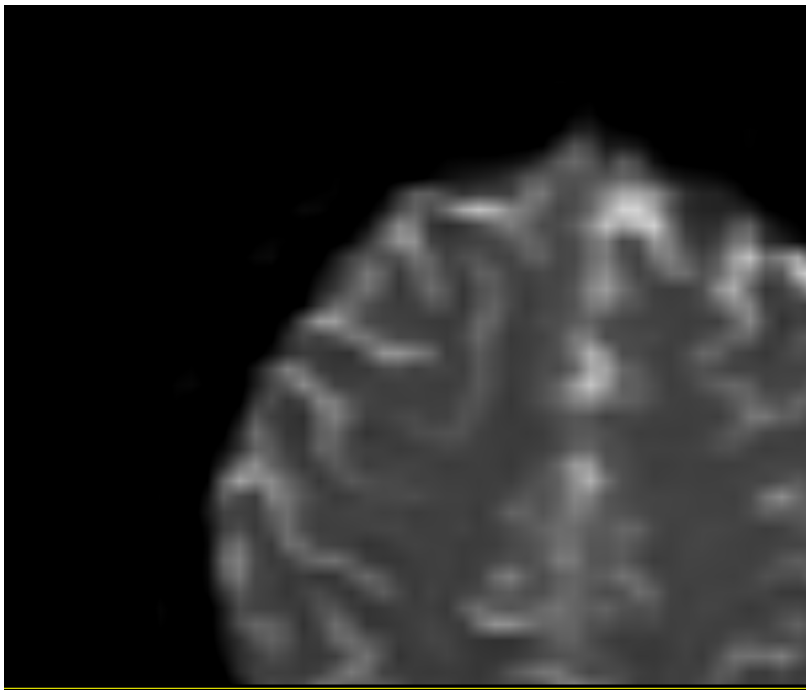

Figure 2.12: Perfusion MR brain slice after thresholding.

allowing simple modification of the algorithm for a specific task. It will also be seen to be very accurate for two-dimensional medical images. Three-dimensional cases require some unsophisticated data post processing [51], or making some modifications in the manner in which the automaton grows into the third dimension from the two-dimensional layer. To prove quality of this method, some test results are presented at the end of this section.

2.3.1 Introduction

The method is based on a cellular automata, first introduced by Ulam and von Neumann in 1966 [52]. It can be used to solve difficult segmentation problems; furthermore, it is multi-label - segments many object simultaneously (computation time does not depend on the number of labels). It is interactive: requires the user to provide starting points for the algorithm (not many seeds are needed and their entering them is not laborious), but in turn it makes it possible to observe the segmentation process and make modifications in it. Interactivity is very important for physicians who like to have some (often large) influence on medical images processing. Furthermore, a radiologist will be able to place seed points very accurately and in characteristic places of a specific organ (the reason why will be explained later), and will check the correctness of segmentation afterwards.

2.3.2 Cellular automaton

A cellular automaton is a discrete model studied in mathematics and theoretical biology [53]. It consists of an infinite, regular array of cells. Each cell can be in one state. The grid can have any finite number of dimensions. Time in the model is also discrete, and the state of a given cell in time t is a function of the states of neighbouring cells at time $t-1$. These neighbours are relative to the specified cell and do not change. Although the cell itself may be in its neighbourhood, it is not usually considered a neighbor. Each cell has the same rule for updating, based on the values in this neighbourhood. Each time the rules are applied to the whole grid a new generation is produced.

Formally, cellular automata is a triplet

$$A = (S, N, \delta); \tag{2.5}$$

where: S is non empty states set, N is the neighboring system, δ ia a transition function that describes how cell state in time $t+1$ is calculated based on state of its neighbors in time t.

State of the cell p also consists of three values:

$$S_p = (l_p, \Theta_p, I_p) \tag{2.6}$$

where: l_p is the current cell label, Θ_p is the cell strength, I_p is a value of the pixel (or voxel) in image corresponding to the given cell p.

2.3.3 Application of cellular automaton for segmentation

In our case, for two dimensional images, cellular automaton is a mesh where for each pixel there is a corresponding cell of automata. In three dimensional cases we would have a set of two dimensional sheets placed one on another. Because this algorithm is multi-label, each cell may be in several states: the number of different areas, labels, we are segmenting plus neutral territory. During the evolution of an automata, label cells are slowly conquering neutral territory. Obviously, each cell has eight neighbors on the same plane, and if we are dealing with a three dimensional case, there are also eighteen ones on the planes above and beneath. In a more general instance, we may use, for example, von Neumann's:

$$N(p) = q \varepsilon Z^n : \|p - q\|_1 := \sum_{i=1}^{n} |p_i - q_i| = 1 \tag{2.7}$$

or Moor's

$$N(p) = q \varepsilon Z^n : \|p - q\|_\infty := \max_{i=1..n} |p_i - q_i| = 1 \tag{2.8}$$

neighboring system.

The method requires the user to provide starting points (seeds) for segmentation. In the simplest case, two kinds of seeds should be provided: seeds corresponding to the object we are segmenting and its surroundings. Of course, more than two classes can be given to segment several objects at once. See Figure 2.13, where the black pixels are neutral territory, the white ones belong to ROI's cells and the grey ones to cells surrounding ROI - background. In each time step every cell, of each label, tries to impound the neighboring pixels/voxels of a different label. After every step each group grows, and after dozen of steps, each pixel of the image has a certain label. The evolution process stops when no new cells have been conquered, in one of the steps. We used the following formula to evaluate if a given cell p has been taken by a cell q in a time step t:

$$g\left(|I_p - I_q|\right) \theta_q^t > \theta_p^t \tag{2.9}$$

where g is a monotonous decreasing function on $[0, 1]$. For example:

$$g(x) = 1 - \frac{x}{I_{max}} \tag{2.10}$$

where I_{max} is the maximal density in the data set.

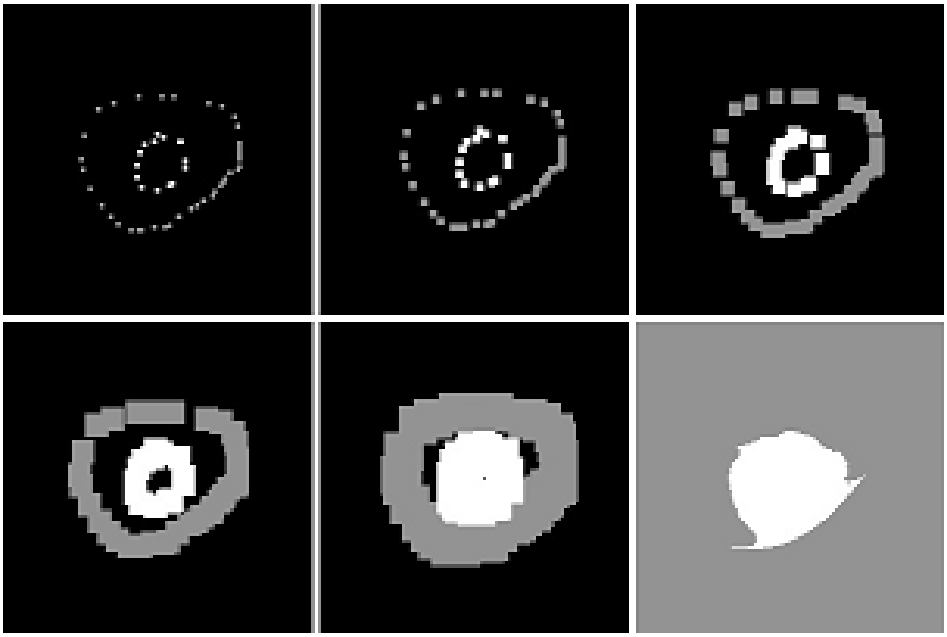

Figure 2.13: Evolution of automaton segmenting tumour from 2.14.c: black - neutral territory, white - organ labeled bacteria, grey - background labeled bacteria. Evolution time steps are as follows (from upper left): 1, 3, 6, 11, 20, 40 (lower right); (from our previous investigation [10, 11, 12]).

The stop condition, as mentioned, can be a case when during one time step the cells state ceases to change. Unfortunately, this approach can lead to the execution of a lot of time steps, some of which are completely unnecessary. Because in most cases we wish to segment one organ which is a small part of the image, the better way is to narrow down the calculations to a small box containing the interesting part of the image, thus shortening the evolution process. A further improvement can be made by considering changes (to be more precise: lack of state changes) close to the object boundaries. We put seed points (the ones belonging to the object and the ones belonging to its surroundings) near the boundary of the ROI (on both the inner and the outer side), so during the evolution the situation on the ROI boundary is quickly stabilised and the process can be stopped, and the interior of the ROI (if not yet conquered by cells corresponding to it) can be automatically filled. Another way to save time is to execute a fixed number of time steps. How many? It depends on the type of segmentation we perform (number of seed points, distance between outermost points, etc.), and can be estimated empirically. For example, results presented in this article (for two-dimensional cases) were achieved with forty time steps (no noticeable differences have been found between forty and, for example, one hundred steps).

2.3.4 Results of the two-dimensional segmentation

We will now present some results of the cellular automata segmentation (see Fig. 2.14). As can be seen, it is a very accurate method allowing to segment different tissues from various types of medical images. It should be stressed that accuracy of segmentation depends strongly on the appropriate choice of seed points.

2.3.5 Results of the three-dimensional segmentation

When we deal with three-dimensional data sets we would like to set seed points for one layer and let the automata segment the rest of the organ. The presented algorithm can be easily applied to two and three dimensions. If seeds for only one layer are given, 3D segmentation turns out to be rather accurate, though sometimes nearby tissues are recognised as a part of the segmented organ (see Fig. 2.15). This problem can be easily fixed by post processing of the data set. Applying morphological operations [54], dilation and erosion filtering, destroys small connections between the main organ and the over-segmented tissue.

Next, connected components labeling [55, 56] is performed to select the only organ of interest to us. As we can see in Fig. 2.16, such processing is very effective. Apart from post processing of the data set, some pre-processing (morphological operations) can also be necessary (for example, thresholding as described in section 2.2.1). The main difficulty may be finding a proper place to put the seed points at.

2.3.6 Observations

Let us first discuss the correct introduction of seed points. In the case of 2D this is rather simple and does not need additional explanation. One should remember to mark (as seeds) all (most of) characteristic pixels of the object and its surroundings - background. When this is done properly, we may be sure that each point will grow in proper direction and the boundary will be found correctly. In 3D the same rules apply, but some further guidelines must be given. When choosing layers in the data set to place seeds in, we should choose the ones which have the most characteristic features of the tissue we are segmenting and its surroundings. Seeds of the background ought to be placed not only near the boundary of the object, but also on different tissues surrounding it (even far away from the object). We must remember that we are dealing with a three-dimensional data set and tissues (organs) distant from the object on the current slice can come into contact a few centimetres above. This fact should also be taken into consideration when selecting a place for seed points.

As we could see from the effects of the two-dimensional segmentation, sometimes the segmented boundary is ragged (see Fig. 2.14). When we need to capture the

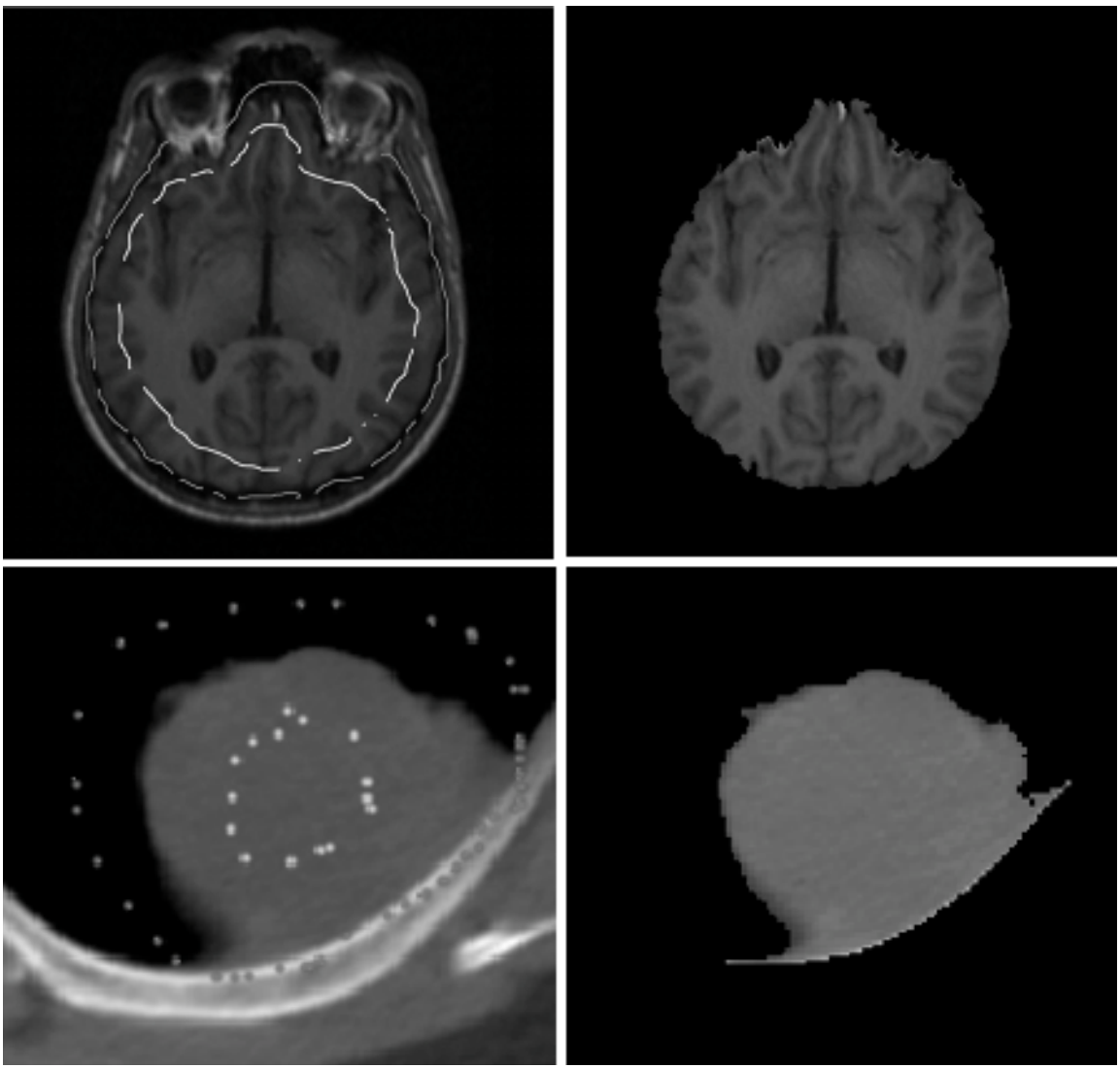

Figure 2.14: (a) Seeds for MRI brain segmentation (top left); (b) Segmented MRI brain(top right); (c) Seeds for lung tumour segmentation (lower left); (d) Segmented lung tumour (lower right); (from our previous investigation [10, 11, 12]).

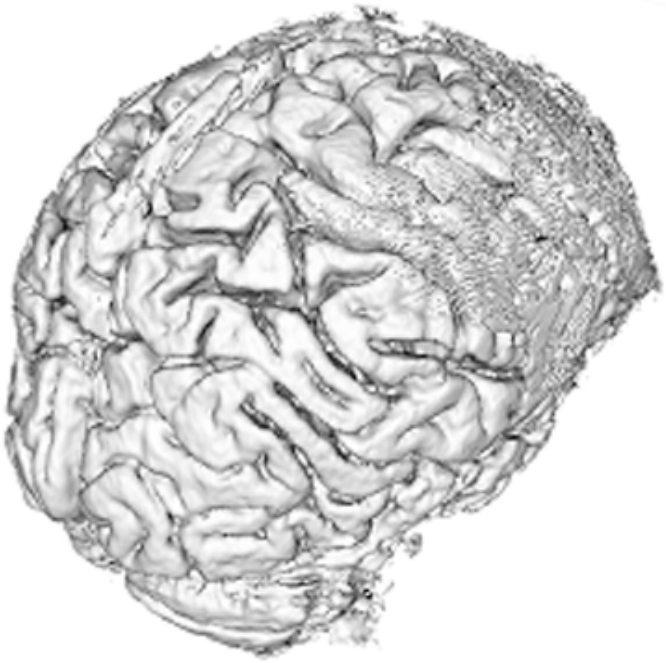

Figure 2.15: 3D MRI Brain segmented by the pure CA - cellular automata; (from our previous investigation [10, 11, 12]).

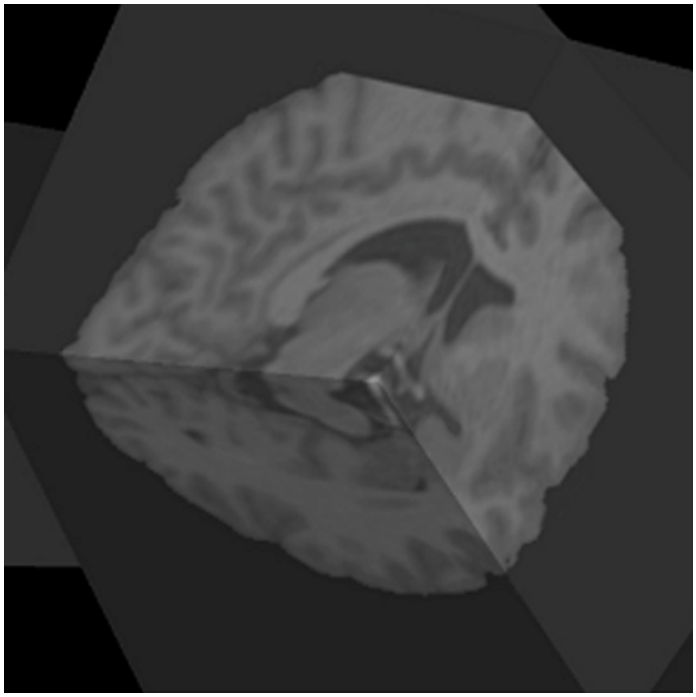

Figure 2.16: 3D MRI Brain segmented by CA and post-processed; (from our previous investigation [10, 11, 12]).

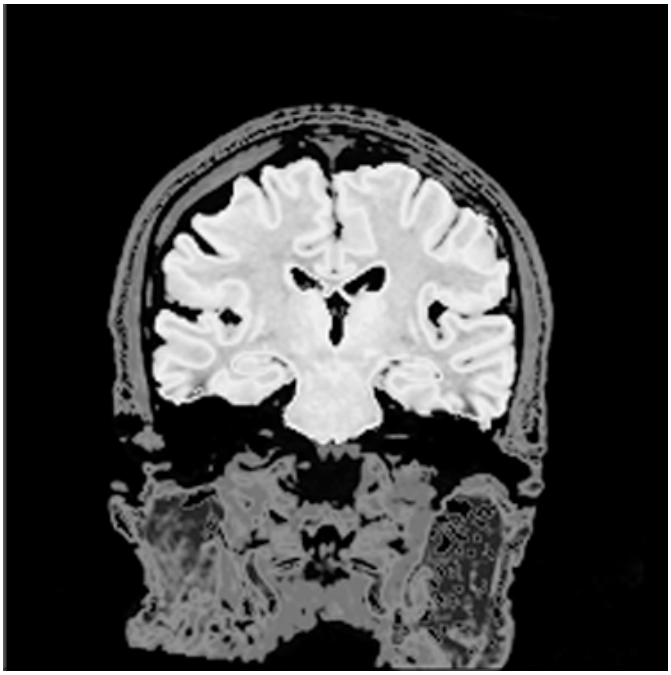

Figure 2.17: Segmented 2D brain (MRI).

smallest detail, this is acceptable in most cases, but it may also be an unwanted artifact. To achieve smoother boundaries a slight alteration of the transition rule can be put forward. Let us call the cells of a different label than the examined one, the enemies of that cell. Now, cells having more enemies than, say, $E1$, are prohibited from attacking their neighbors, and cells that have more than $E2$ enemies are automatically conquered by the weakest of their neighbors. Values $E1$, $E2$ control boundary smoothness and should have a value from 6 to 9 (no smoothing) for the Moor neighboring system. This modification has not been tested for three-dimensional cases.

2.3.7 Tests and conclusions

To objectively estimate quality of the cellular automaton segmentation we have compared its results with segmentation done by an expert - physician. The test data was obtained from The Internet Brain Segmentation Repository (IBSR) [57]. This data consists of MRI brain scans of several patients. For each scan there is a second one with the brain segmented by a radiologist. The testing method is, therefore, obvious. We have segmented the brain from several dataset using the cellular automaton approach and compared them with model results by a simple image subtraction (see Fig. 2.17 and fig. 2.18). Next, we have compared the number of over- and under-segmented pixels/voxels (real quantitative difference) with the exemplary pixels/voxels number. For a few dozen of two dimensional images we have obtained an average difference of 3,3% (both over- and under-segmentation). For

Figure 2.18: Comparison of result acquired by CA segmentation with the one done be an expert (bright pixels mark over-segmentation - 434 pixels, darker under-segmentation - 155 pixels); relative difference to the model is 2,4%.

three dimensional datasets the difference ratio was slightly greater and was equal to 4,7%. As we can see the dissimilarity is very small, and often not visible to the naked eye.

To summarise, advantages of the algorithm presented are as follows: accuracy (results comparable with those provided by human expert), interactivity, possibility of segmenting multiple object at once, scalability. Its drawback is a long execution time, but as we could see this can be levelled by making some simple modification. In the three-dimensional case, post processing is required to erase artifacts which could sometimes appear during segmentation. Nonetheless, the method itself is very promising and should be developed further to improve its performance and find appropriate modifications for specific purposes.

2.4 Eyes segmentation

In the following section we present the algorithm developed to precisely segment and detect eyes on the MR studies. Because the method was developed to serve the needs of the brain axis detection that will be discussed in Section 3.4, it works with T2 MR series, where such detection is simplest. We did not need the method for

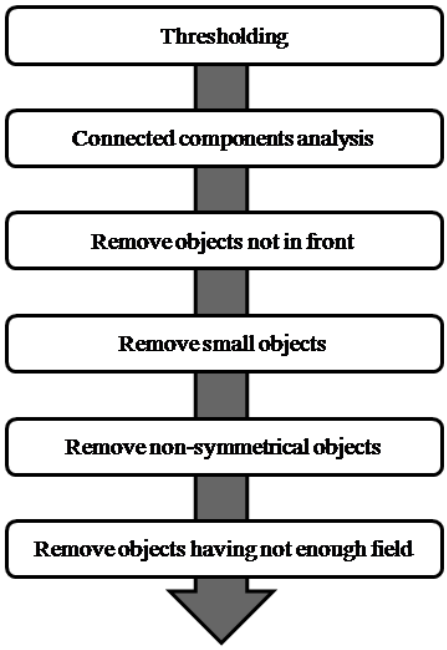

Figure 2.19: Steps of the eye segmentation algorithm.

segmenting eyes on T1 series. In general, each MR study should contain T1 and T2 series as it complies with study acquisition methodology.

2.4.1 Algorithm

We have divided the algorithm into several steps shown on 2.19.

Thresholding First of all, thresholding (see Section 2.2) needs to be done to remove all low intensity voxels from the data set (we are dealing with three dimensional set of axial-plain brain slices). Voxels corresponding to eyes have high intensity values on the T2 series; therefore, they can be easily separated from surrounding tissues by the means of thresholding segmentation. After analysing several T2 series of different patients, we have found that best results are obtained by removing voxels smaller than 40% of the largest value in the data set. Sample results of this operation are shown in fig. 2.20.

Connected components analysis After segmentation is done, only thing left to do is select objects corresponding to eyes. To do this, we've used connected components analysis that scans whole data set and gives each separate set of pixels a different label (see Section C.3). This analysis was done for each slice separately and 8-connectivity was used. We didn't treat components as three-dimensional object because it would increase the complexity of the algorithm, while not giving enough

Figure 2.20: Results of the thresholding operation.

additional data in return.

Each label / object found on the slice is described, inter ilia, by two points: upper-left and lower-right vertex of the smallest rectangle containing this object. When all separate objects on each slice were detected, algorithm removes objects that are not located in the front of the head (at 40% of its length). Then objects smaller than 15 pixels in diameter (diagonal of the containing rectangle) are removed. The next step is to remove objects that have containing rectangle vertical and horizontal size differ more than 20% (eyes generally should be symmetrical - with some small deviations).

Unfortunately, this last condition is not always enough. The last step is to remove all objects that are not fully filling the space of the containing rectangle. The algorithm compares the field of the rectangle with the number of voxels associated with a given label. If an object is smaller than 60% of the rectangle, it is removed.

2.4.2 Results

The presented algorithm has proved to be very effective and correct. Some sample results are shown in 2.21. As can be seen, eyes have been found and segmented correctly.

2.4.3 Tests and conclusions

To test the presented algorithm we have asked physicians from hospital we are cooperating with to manually segment eyes from fifteen different T2 series. We have then used our algorithm to do the same thing and compared our results with

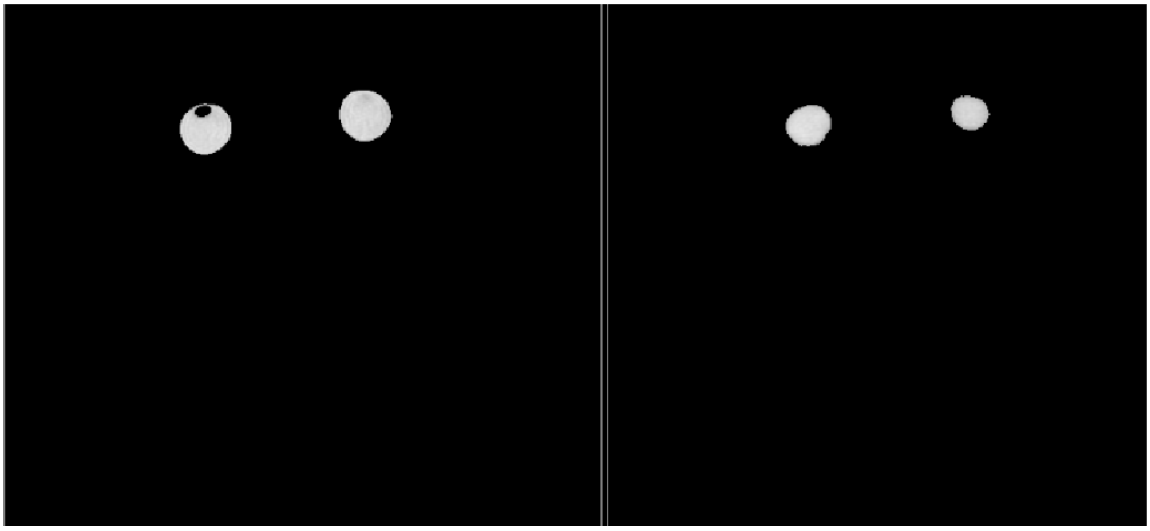

Figure 2.21: Results of the eye segmentation algorithm.

the ones obtained by experts, by subtracting one form another. Next the number of over- and under-segmented pixels was analysed and compared (real quantitative difference) with the exemplary pixels number. For a tested series, we have obtained average difference of 4,2% (both over- and under-segmentation). As we can see, the dissimilarity is very small, and often not noticeable to the naked eye.

In conclusion this rather simple method has proved to be a very effective tool for quick and accurate eye segmentation and detection from MR studies.

2.5 Combined T1 and T2 MR brain segmentation

In the following section we present a new approach to segmentation of brain on MR studies. The method is fully automated, very efficient, and quick. The main point of this algorithm is subtraction of T1 series form T2 series (that's why we've called it combined), preceded and followed by a few image processing steps. The algorithm is adopted to methodology of study acquisition used in hospitals we are cooperating with. The method has been tested and graded by experts, also the segmentation results were compared numerically to those produced by experts. The results of these tests point to the great effectiveness of the presented algorithm.

2.5.1 Method

At this point, the presented method, works on the axial plane and requires T1 and T2 axial series of the same part of the brain to be present in the MR study. The algorithm is divided into several stages, as shown in figure 2.22. It uses two data

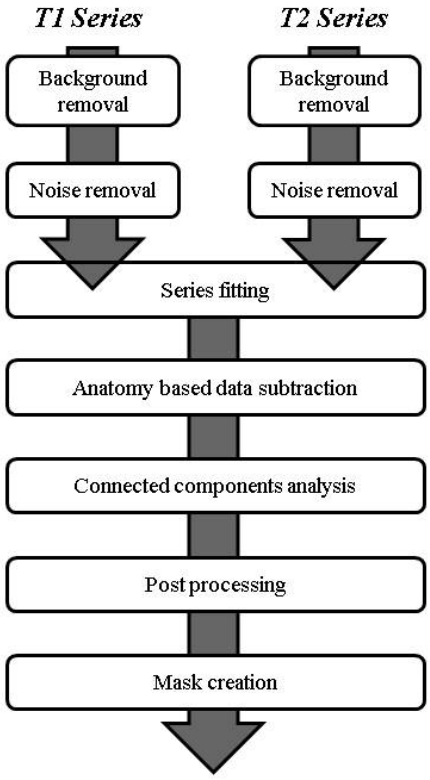

Figure 2.22: Steps of the segmentation method.

sets, first T1 series and second, T2 series. Each of them covers a certain part of the brain, as stated in a DICOM header. From this header information on slice location in space is obtained. Generally, in hospitals we cooperate with studies are acquired so as to ensure that slices in a different series are in the same location.

Information from both series is used to produce a general segmentation mask for a given study. This mask can then be used with each series in a study.

Background removal This is a preliminary stage which is applied to both T1 and T2 series. Its purpose is to remove all background noise from images. See Section 2.2.1 for details.

Noise removal This is a simple filtering operation which removes remaining single pixels from the image [58].

Series fitting

As mentioned before, in most cases T1 and T2 series are acquired at the same patient position. However, in some few cases it's different. In such cases, it is

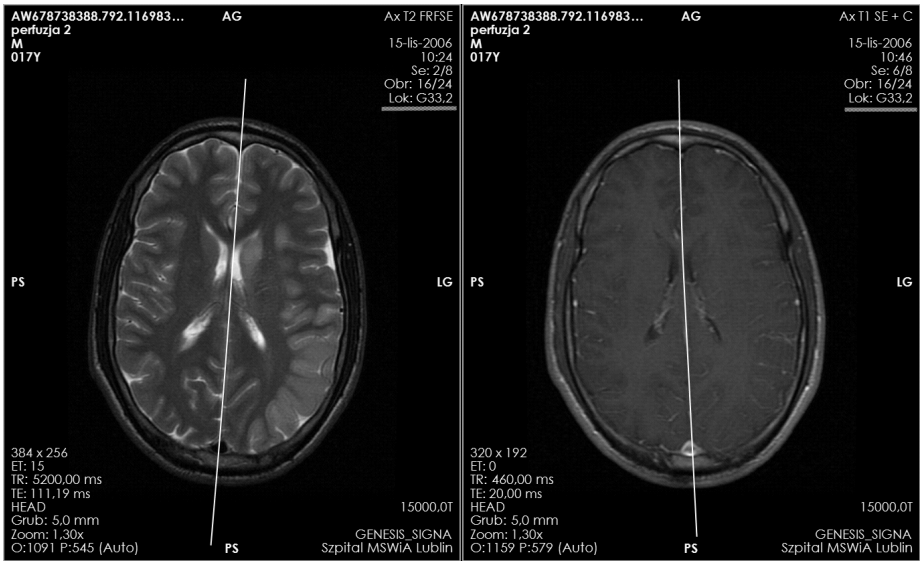

Figure 2.23: Slices of T1 and T2 series of the study where the patient was slightly rotated between series acquisition. It didn't affect slices location, as can be read from DICOM header (underlined value). White lines denote symmetry planes. Screenshot from my application. Actual patient study thanks to Michał Siczek, MD, MSWiA Hospital, Lublin.

necessary to transform one data set to match the other (see Fig. 2.23).

To detect a situation when position of the patient is different in both needed series, symmetry planes [59] are found for both of them and compared. Next, one of the series is rotated (and translated, if needed) to fit the second.

Anatomy based data subtraction This operation is the central point of the whole algorithm, when two series are subtracted. To be precise, T1 series are subtracted from T2 series. However, we must remember that the brain is a complex structure and simple subtraction of one voxel from another will do us no good. (Our main goal is to obtain a contour of the brain as accurate as is possible.)

A first look at a T2 MR brain slice gives us very important information. The gray matter (which lies on the surface of the brain) is very well produced as high intensity voxels. Those voxels will form a brain contour - our goal - in the next steps of the algorithm. Of course, the anatomic structure of the brain differs in different parts of the brain. Therefore, we must make subtraction aware of the brain's anatomy details. To accomplish this, we have divided the brain into several regions, according to how tissues are distributed on a given slice. Figure 2.24 shows those regions. For each region a ratio was chosen on an empirical basis (through subtraction analysis for twenty different studies).

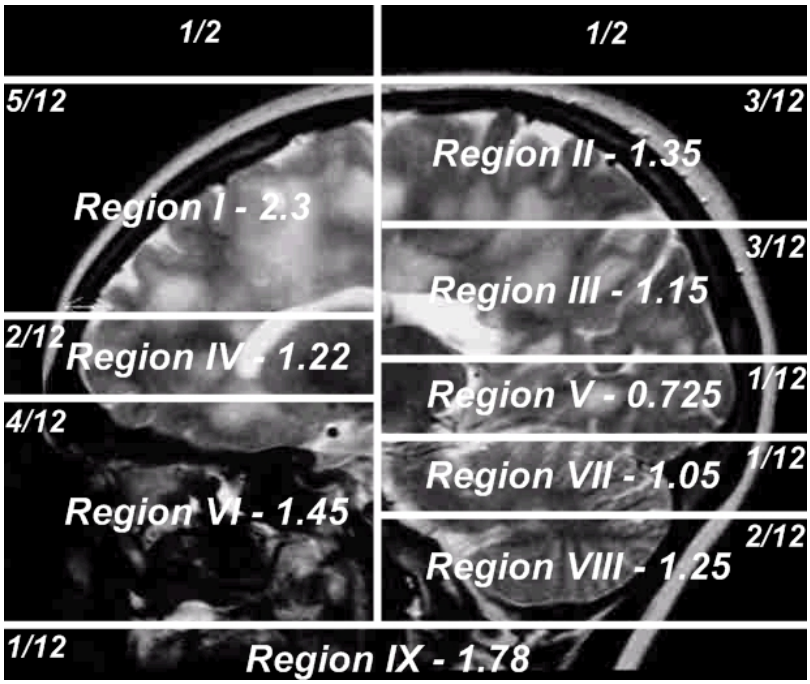

Figure 2.24: Region distribution throughout the brain volume. For each region its ratio is given along with its size. Size is provided as a fraction of the whole brain data height and width.

During subtraction, voxels in each region are treated differently, according to the following formula:

$$V_M = V_{T2}^R - ratio^R * V_{T1}^R \tag{2.11}$$

where V_M is a result voxel of subtraction, V_{T2}^R is a voxel in region R of T2 image, V_{T1}^R is a voxel in region R of T1 image, and $ratio^R$ is ratio for region R.

As a result of this step, we will get well segmented brain structures, surrounded by other tissues, in most cases not in direct contact with the ROI - brain. Only a few groups of voxels may be "disturbing" the quality of the segmentation. These anomalies will be removed in the penultimate step of this segmentation method.

Connected components analysis In our case connectivity is determined by the 8-connectivity and an algorithm is applied for each slice separately.[60]. After labelling is done, only certain components are left on each slice, i. e. the largest (having most voxels) and those no smaller than 15% from them. This removes all unnecessary objects leaving only the brain itself (also the spinal cord and cerebellum).

Post-processing As mentioned before, there remains the issue of a few anomalies at the brain edge. To fix this, we have looked at the distribution of voxel intensities along the lines - intersections of slices. Consider Figure 2.25. Plot A shows distribution of voxels values (along showed lines) on a well segmented boundary, whereas plot B re a badly segmented part of the brain border.

An ideal border intersection has steep, linear outer slope, while the border with artifacts is disturbed with additional local peeks and roughness. The obvious thing to do is to remove them by smoothing intensities values, as shown on Fig. 2.26.

In our algorithm, it is done for vertical and horizontal intersections, plus for each analysed line, with two neighboring lines also taken into account. Thanks to this continuity of border is ensured, and there is no situation when there is insertion in borders (as the effect of too steep smoothing).

Mask creation The last step of the segmentation involves filling shapes, contours of structures acquired during the previous segmentation steps. We have used flood fill algorithm with 4-connectivity. As a starting point, we have used inner border of the shape boundary (see Fig. 2.27).

2.5.2 Results and tests

At this point we have tested our method on several dozen (about twenty) studies from two different hospitals we are cooperating with. The method produces very good "visual" results. Boundaries of the brain tissue are well marked, smooth and not rugged. The spinal cord (often shown on first few slices in series), along with cerebellum is also well segmented. Figures 2.28 and 2.29 show several examples of the method outputs.

Of course, visual estimation is not authoritative, especially when done by a non physician. Therefore, we have chosen two indicators to measure the effectiveness of the method.

First: We have asked experts - physicians to rank results of segmentation according to their quality and precision. Each segmented data set could be graded as: perfect (there are no mistakes in segmentation, I wouldn't do it better), very good (there are some minor mistakes that don't affect in any way diagnostic quality of data), good (results are average, some segmentation may have been done better; however mistakes, would not affect diagnosis), bad (only some slices are well segmented, but also contain some mistakes, diagnosis could be affected), very poor (almost nothing is done well, why should one use this method?). Each description had a corresponding numeric value from 5 to 1, respectively.

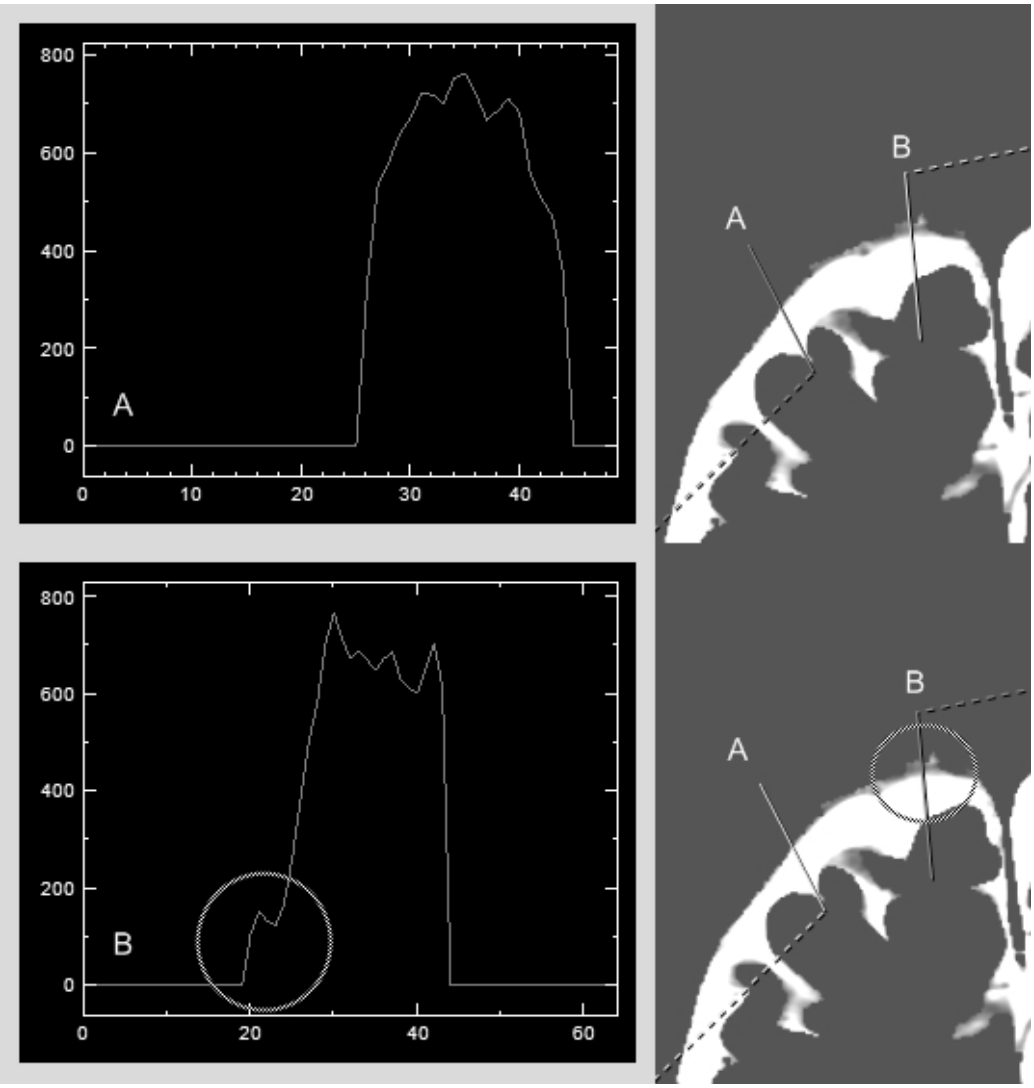

Figure 2.25: Distribution of voxels intensities along lines marked for sample slice being the result of previous algorithm operations. The X-axis represents voxel position on the line and the Y-axis its intensity.

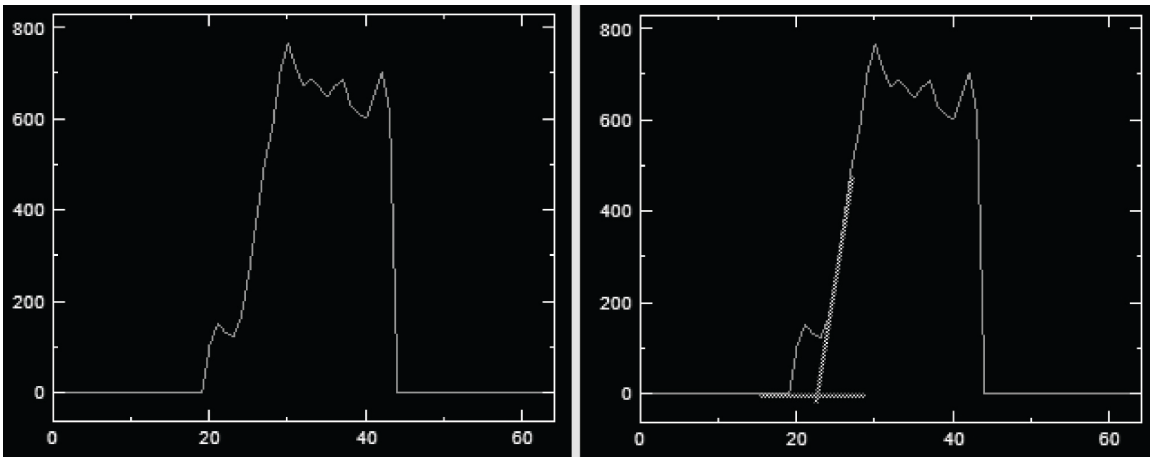

Figure 2.26: Post-processing, border smoothing visualisation. The X-axis represents pixel position on the line and the Y-axis its intensity.

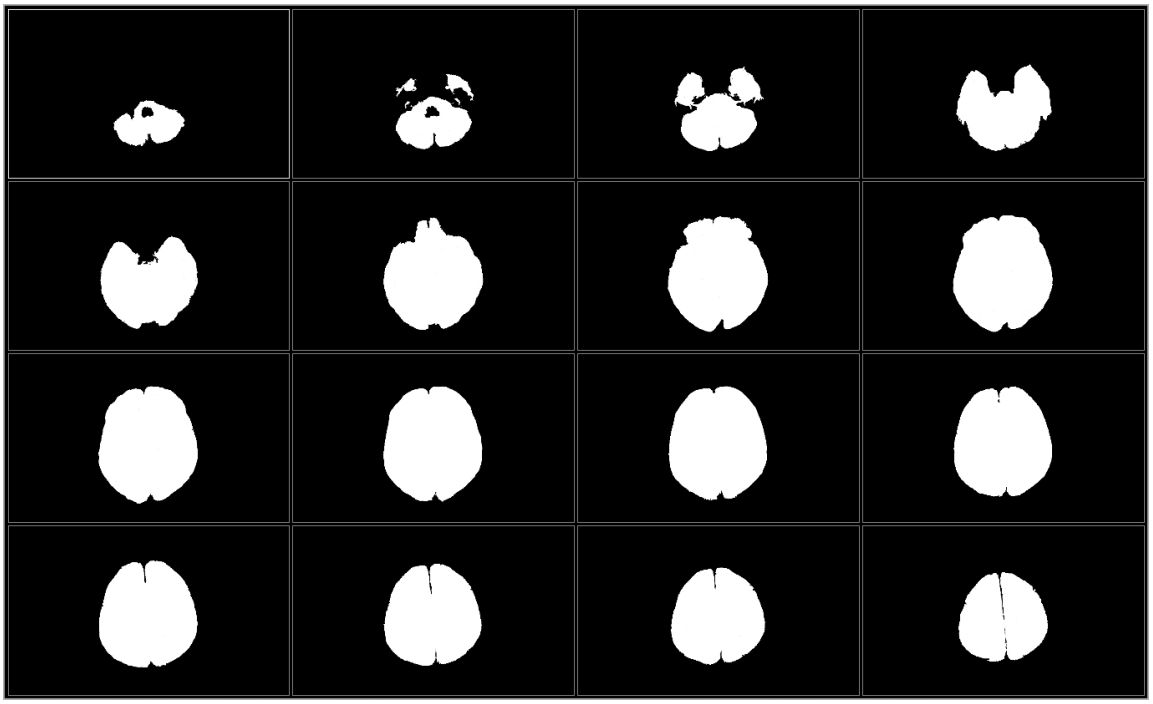

Figure 2.27: Mask created by the algorithm.

	DS 1	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7
Expert 1	4	5	4	4	4	4	4
Expert 2	4	4	3	3	4	5	4
Expert 3	5	4	4	4	5	4	4
Expert 4	4	5	3	4	4	5	4

Table 2.2: Results of the first test (ranking of results performed by experts) for seven data sets

	DS 1	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7
aver. over-seg. %	2.1%	2.0%	2.8%	2.2%	1.9%	1.8%	2.3%
aver. under-seg. %	1.5%	1.7%	2.1%	2.0%	1.7%	1.7%	1.9%
std. dev. of over-seg. %	0.6%	0.6%	0.9%	0.8%	0.7%	0.9%	0.8%
std. dev. of under-seg. %	0.7%	0.6%	0.8%	1%	0.7%	0.9%	0.8%

Table 2.3: Results of the second test (comparison with segmentation performed by experts) for same seven data sets.

Second: We have asked our experts to manually segment a few data sets for us. These data sets were then segmented with our method and the results were compared with ones acquired manually. We have measured the number of over- and under-segmented pixels to the whole number of pixels. An over-segmented pixel is one that has been included into segmentation mask by our algorithm, but hasn't been included by experts. An under-segmented pixel, on the other hand, is such a point that was included by the physicians but not by our method.

Tables , show results obtained for seven data sets. These are some of the best, worst and average results.

Generally, for twenty data sets the method has achieved the average of 4.3 points in the first test. In the second test, the average under-segmented pixel percentage was 2.6% with a standard deviation of 1.7%, and the average over-segmented pixel percentage was equal to 3.1%, with a standard deviation of 1.65%.

As we can see, the method is very accurate. Furthermore, it is not time consuming. The average time of its execution is about one, one and a half minute on a rather standard PC configuration. Of course, it might be faster but in our case precision is more important.

There is however, a small problem. The results seems to be to good for such a simple method. The reasons for these are rather obvious. First of all, as mentioned before, most of test data didn't require series fitting. Secondly, ratios for brain regions were estimated according to data sets / data acquisition techniques used in one of the hospitals we are cooperating with. Therefore, they form a specific kind of "mean brain". This "mean brain" proved to be correct for data in other hospitals, and some online studies (released to public domain). We are aware of the fact that for other data it may not be so precise. There are many parameters, the technician acquiring a study can set, which in some configuration can produce brain image not fitting the model presented. We want to solve this issue in the future.

We have noticed that changes in ratios up to 10% have no effect on the final result; therefore again we can say that in most cases the method presented should work and produce very good results.

2.5.3 Conclusions

As we could see, the method presented brings very good results. Its drawback is requirement for T1 and T2 series to be present in a study, and that they cover the same part of the brain. Fortunately, in most cases this is satisfied. Nevertheless, the algorithm is rather simple and could be applied in each situation when accurate segmentation of the brain is required.

Also, there is some doubt if such precise results can always be achieved - is the method universal? Maybe not, but the general approach definitely is. In most cases, such a tool is used in one hospital (not in many at the same time); therefore, it can be adjusted to a general acquisition technique used in a given unit, by choosing a different set of ratios.

Nevertheless; the algorithm is simple, very promising and could be applied in each situation when accurate segmentation of a brain is required. In our future work we will focus on improving the method's effectiveness and efficiency. We also want to implement a system that would automatically compute region ratios for aware series subtraction. Before using this tool one should learn it by providing a set of studies he or she is working with. By analysing these data, system would estimate ratios values in each region. These solution would solve the issue of universality.

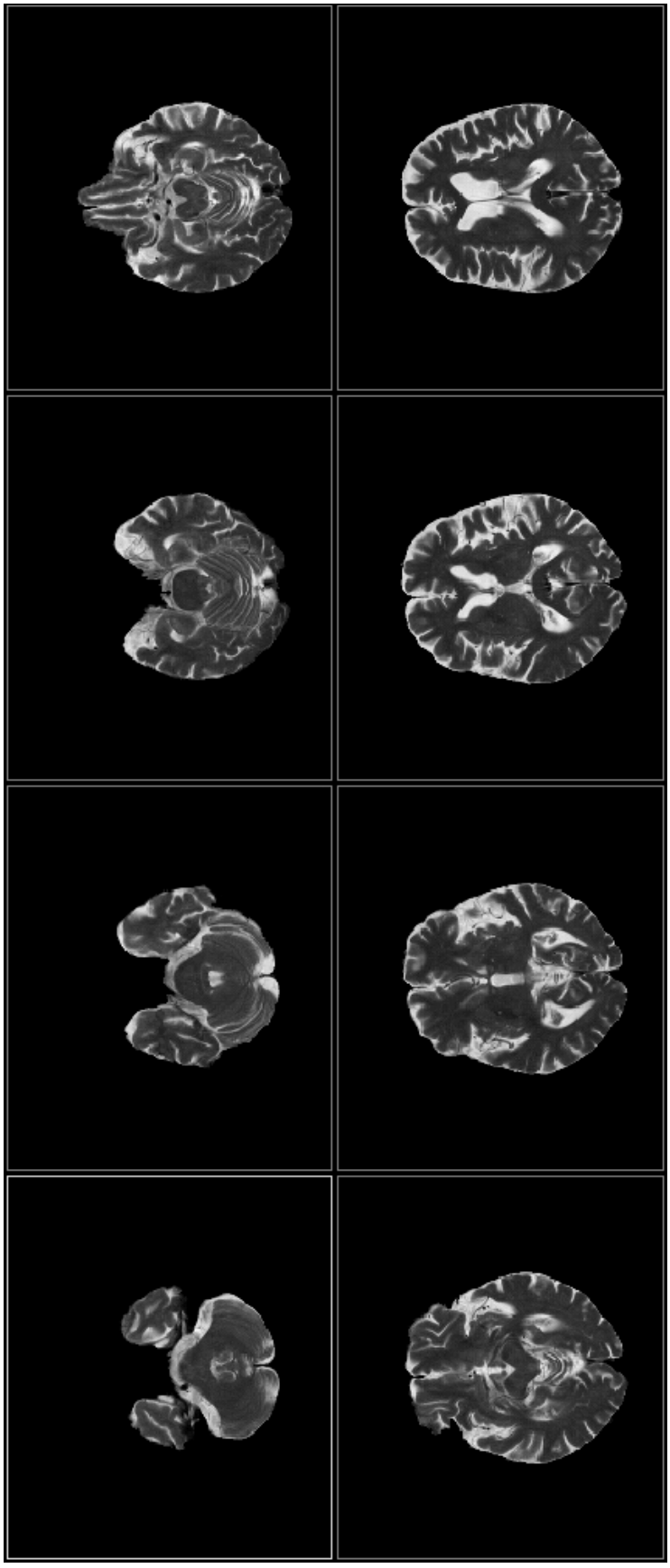

Figure 2.28: Results of the segmentation, part 1.

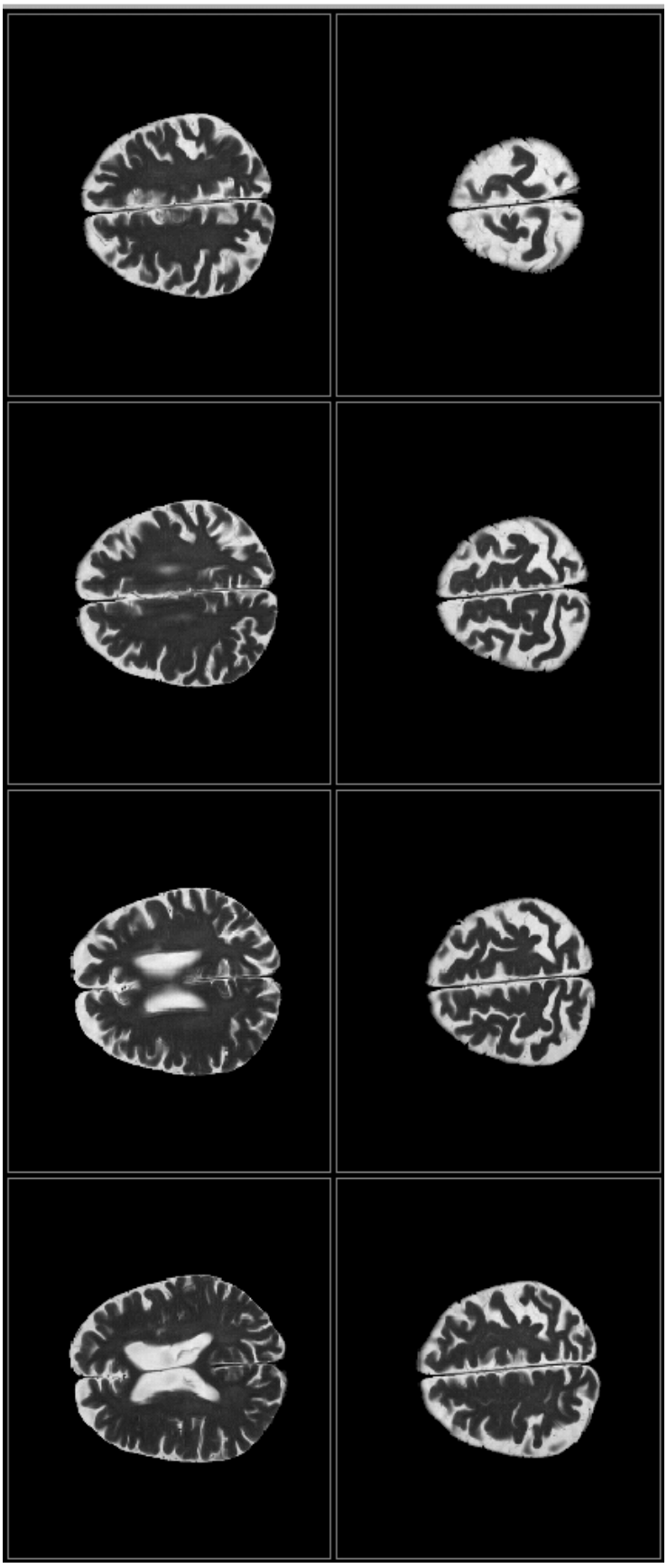

Figure 2.29: Results of the segmentation, part 2.

Chapter 3

Perfusion analysis

If a man will begin with certainties, he shall end in doubts; but if he will be content to begin with doubts, he shall end in certainties.

- Francis Bacon, Advancement of Learning Bk. I

Contents

3.1 Symmetry plane of the brain on perfusion MR images

In this section we present an algorithm whose purpose is to estimate real symmetry lines of the brain on the MR perfusion studies. Because perfusion slices are acquired in low resolution and larger distances than standard MR studies, symmetry plane of the brain must be evaluated on the slice-by-slice basis. The algorithm presented, therefore, searches for symmetry line, taking into consideration only anatomical information on each slice, ignoring the adjacent slices.

3.1.1 Introduction

Perfusion MR study [61, 62] is based on the use of injected contrast agent that changes the magnetic susceptibility of blood and thereby the MR signal, which is repeatedly measured during bolus passage. Because images must be acquired very quickly and frequently, the quality must be reduced and the distance between slices increased. As a result of one study, we get about 400 - 500 images (about 10 images per 45 passes). Fig. 3.1 and 3.2 shows a model set of images acquired by a single pass.

It is common knowledge that the brain consists of two symmetrical cerebral hemispheres. Often to check for abnormalities, differences, two adjacent hemispheres on the same image are compared. Symmetry information is also required when calculating perfusion parameters (see section 3.2), as different Arterial Input Functions (AIF) are take for each hemisphere. To automatise this process, there is a need to find the symmetry plane of the brain. In this chapter, we are using the term *symmetry*

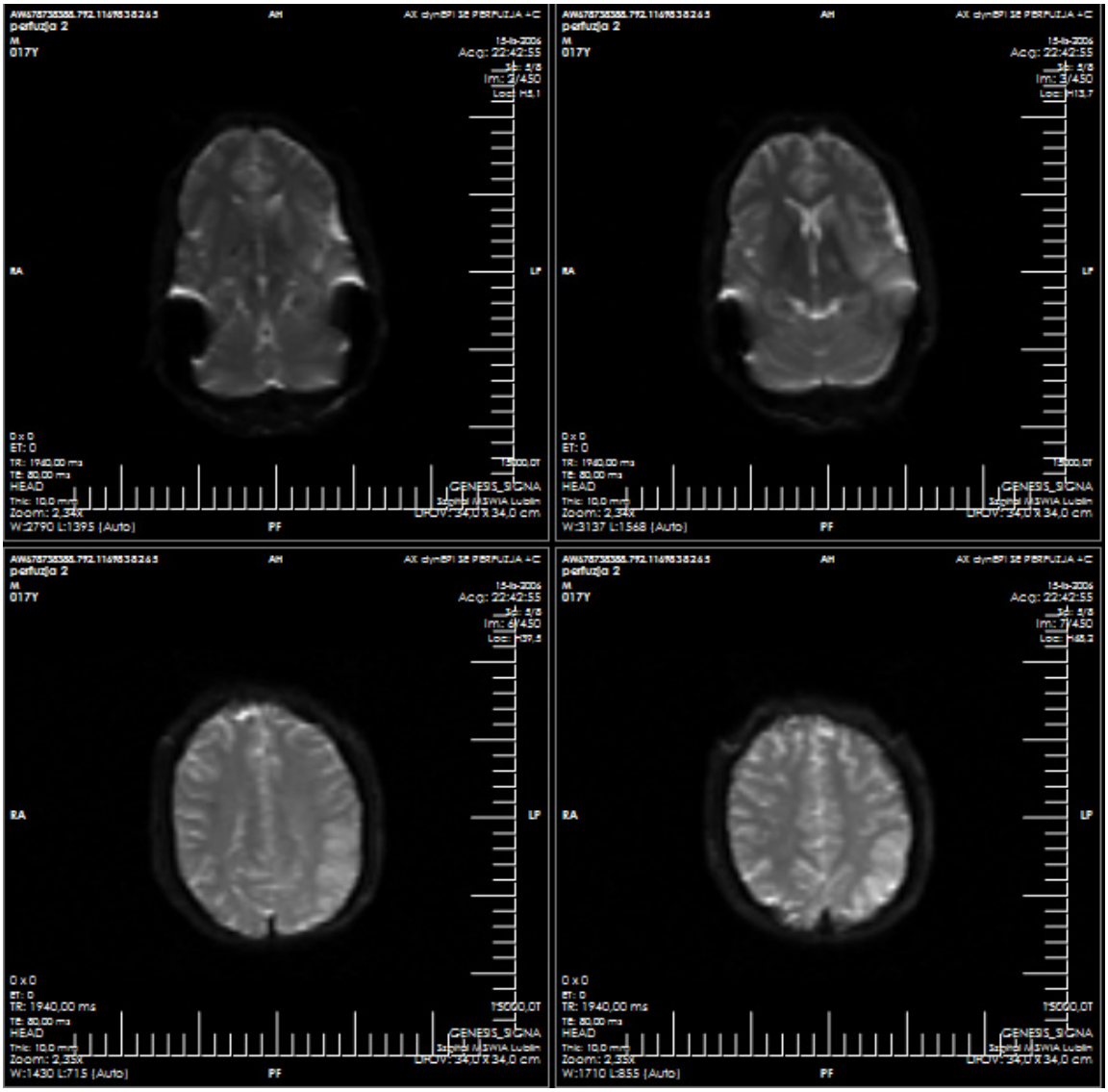

Figure 3.1: Model MRI perfusion study (first pass) - part 1. Screenshot from my application. Actual patient study thanks to Michał Siczek, MD, MSWiA Hospital, Lublin.

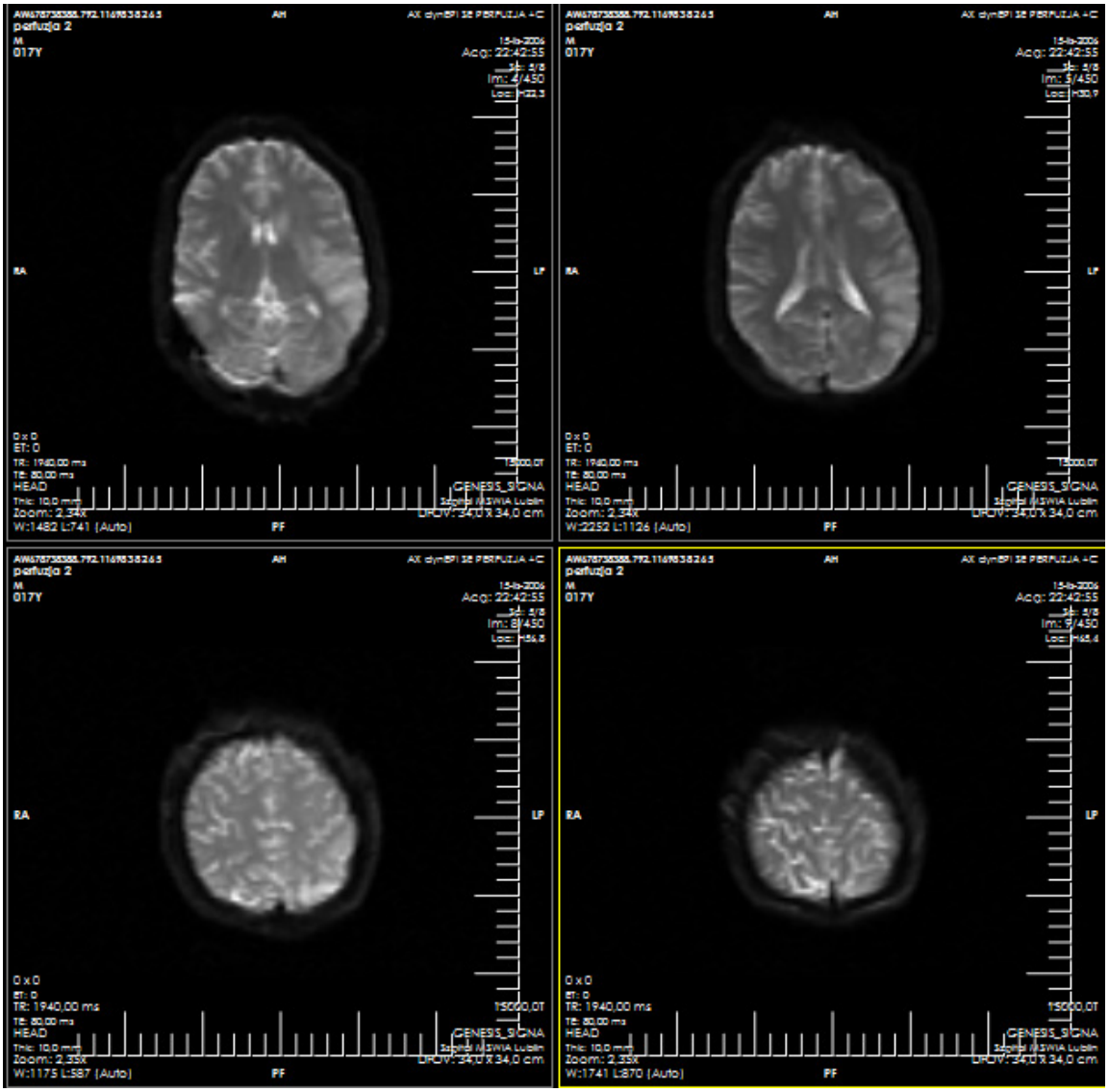

Figure 3.2: Model MRI perfusion study (first pass) - part 2.

line/plane to describe a symmetry plane of the brain itself, not the symmetry line of the data on image. Because the image acquisition plane is not perpendicular to the brain symmetry plane, slices acquired rarely show symmetrical image (see Fig. 3.1 and 3.2). Therefore the only chance to find a symmetry plane is to find anatomical structures that determine it.

3.1.2 Perfusion Symmetry Line algorithm (PSL)

Due to the distance between the slices, we are evaluating symmetry lines for each slice separately - we are not treating set of slices as a complete volume. The PSL method can be divided into the following steps:

1. Data preprocessing
2. Finding the Characteristic Points (CP)
3. Linear regression

One of the main assumptions of this algorithm is that the brain symmetry line is not deviated more than about 15^0 from the Y axis in the scanner coordinate system.

Data preprocessing The first step of the presented algorithm is to remove background noise and other tissues surrounding the brain. This can be achieved by simple thresholding to the certain value (see Section 2.2.2).

Finding Characteristic Points To have some starting point for estimation of symmetry line, the algorithm finds notches between hemispheres in the front and back of the brain (see Fig. 3.3). To find them, the image is scanned line-by-line from beginning to end. According to assumption of brain not being too much rotated, we are narrowing the search to the 25% middle of the image (vertical *trench*). In the case of an average slice having 120x120 pixels, this *trench* would be 30px wide. Generally we are looking for the deepest (closest to the middle of the brain) notch in the front and back tissue of the brain. If there are several notches found at the similar *deepness* and the difference (in deepness) is no greater than 2% (of the whole image size), then the problem is not resolved and no CP is found. The same situation is when the *notch* is too wide (more than 4%) - de facto, there is no notch (see Fig. 3.3.)

For obvious reasons, finding these points is not always possible - on some slices there are no notches detectable. Also, sometimes these points can be badly found, so they cannot be the only CP used to fit symmetry line into. Therefore, we have also incorporated a method for finding other anatomical structures that can describe symmetry plane of the brain.

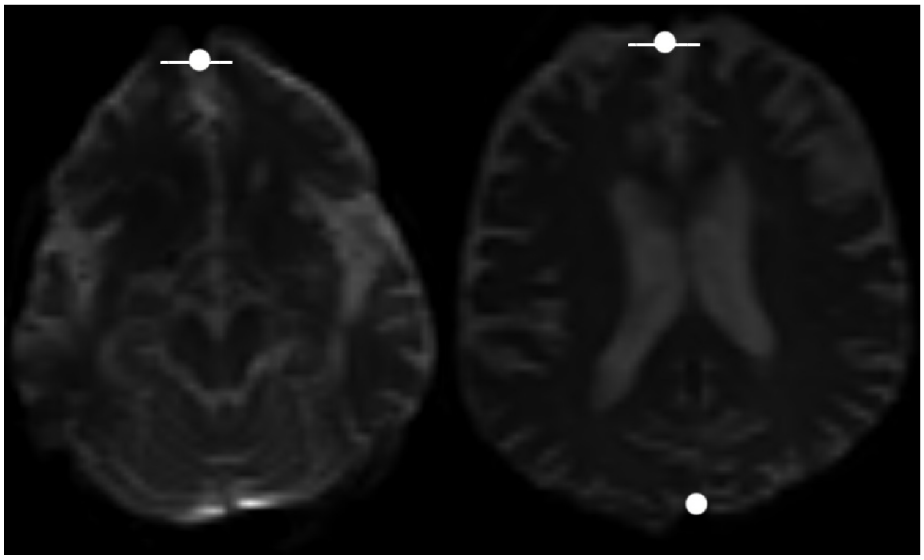

Figure 3.3: First two CP (on the right). Case when one of the CP cannot be found (on the left). White segments mark scan lines shown in Fig. 3.4.

To do this, we are continuing to scan-line the *trench* between the first two CP or places where the previous step has ended. The algorithm is searching for local minima's or maxima's on these scan lines that have local maxima's or minima's respectively, on both of its sides (see Fig. 3.4). In this figure one can see ideal cases that do not leave place for misinterpretation. And only such cases are used later. We do not need a lot of points in the later step of the algorithm - linear regression. Most important is quality and accuracy of these points.

Of course, in many cases good points will be found on adjacent scan lines. For each set of six points that are adjacent (occupy six neighbouring horizontal lines) the middle one is selected as CP to represent all six points. Thanks to this step, only reasonable number of CP will be used in regression.

Perpendicular Regression Given a set of Characteristic Points, it is possible to fit a straight line into this set by using linear regression [63, 64, 65]. When we perform a regression fit of a straight line to a set of (x, y) data points, we typically minimise the sum of squares of the *vertical* distance between the data points and the line. However, in this case we have to optimize the perpendicular distances to the line.

One way of tackling this problem is to find the *principle directions* of data points. Let us assume that we have the (x, y) coordinates of n points. First, we compute the average of the x and y values, naming them X and Y respectively. The point (X, Y) is the centroid of the set.

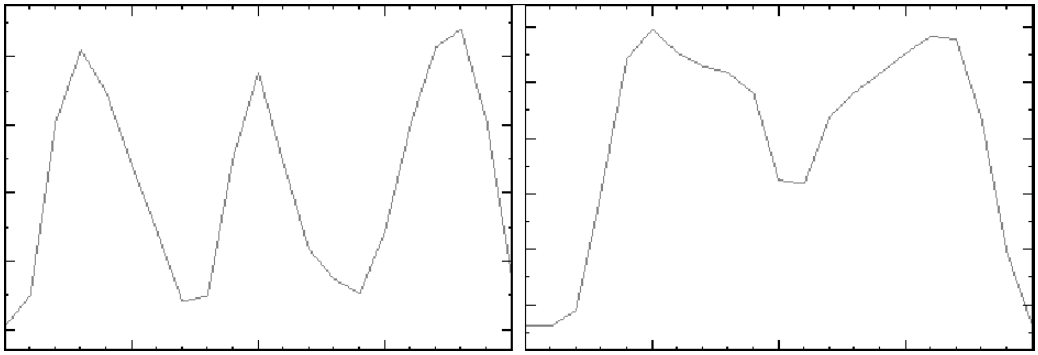

Figure 3.4: Horizontal scan lines of the trench for two different slices. The X-axis represents pixel position on the line and the Y-axis its intensity.

Then we subtract X from each of the x values, and Y from each of y values, which gives us a list of n data points, having centroid: $(0, 0)$.

To find the principle directions, let's imagine rotating (angle q) the entire points set about the origin. This translates the point (x, y) to the point (x', y') where

$$x' = x * \cos(q) + y * \sin(q) y' = -x * \sin(q) + y * \cos(q) \tag{3.1}$$

For any fixed angle q, the sum of the squares of vertical heights of n translated data points is $S = \sum [y']^2$. We are searching for the angle q that minimises this sum. We can interpret this as rotating the regression line so that perpendicular distance corresponds to the vertical. To do this, we take the derivative with respect to q and set it equal to 0. The derivative of $[y']^2$ is $2y'(\frac{\partial y'}{\partial q})$, so we have

$$\frac{\partial S}{\partial q} = 2 \sum [-x * \sin(q) + y * \cos(q)][-x * \cos(q) - y * \sin(q)] \tag{3.2}$$

This must be equal to 0, so we can immediately eliminate the factor of 2. Next, we expand the product and collect terms into separate summations:

$$[\sum x * y] * \sin^2(q) + [\sum (x^2 - y^2)] * \sin(q) * \cos(q) - [\sum x * y] * \cos^2(q) = 0 \tag{3.3}$$

Dividing by $cos^2(q)$, we get a quadratic equation:

$$x * y * \tan^2(q) + x^2 - y^2 * \tan(q) - x * y = 0 \tag{3.4}$$

where the *curly braces* indicate the sum of the contents over all n points (x, y). Dividing by the $x * y$ gives

$$\tan^2(q) + A * \tan(q) - 1 = 0 \tag{3.5}$$

where $A = \frac{x^2 - y^2}{x * y}$. Solving this quadratic function for $\tan(q)$ leads to two solutions that correspond to the *principle directions*, that is the directions where the spread is maximum and minimum. We are searching for minimum.

Having the angle q, we can find symmetry line, which is passing through the centroid of the data.

3.1.3 Results and tests

Sample results, obtained by the method presented are shown in Fig. 3.5 and 3.6.

We have tested this method on ten data sets from two sources - Clinical Hospitals. In 92% of cases (failed on 8 images out of 100 - 10 first pass images from 10 studies), the method has managed to find more than four CP and perform regression. We must note that images on which the method has failed were difficult cases located in the lower parts of the brain. Generally, all abnormalities that must be located by the perfusion study are located in higher parts of the brain. Therefore, for tested data sets, the method proved to find CP in all diagnostically meaningful cases.

To evaluate the quality of symmetry lines found by the algorithm, we have compared its results with the ones provided by the experts. We have asked three physicians to mark symmetry lines on first pass images from each study. Each expert received six data sets. Studies were divided so that three data sets were common for each expert. This was our control set to estimate average error between experts. Because expert skills are subjective - each one will mark symmetry line a little differently - we wanted to check what difference is not significant to them.

As indicators for comparing two symmetry lines we have compared the angles (from the vertical axis in terms of percent of 45 degrees) and the distance between these lines [66].

First of all, we have compared results for control sets. The maximal difference in distance, between experts, was about 5% (of image width in pixels). The difference in case of the angles was about 2,5% (2% of 45 degrees = 0,9 degree). The mean difference for the distance was 4% and the angle about 2%.

Next, we have compared expert results with those acquired by the method in question. Comparison showed mean difference in distance of 5,5% and 3,5% in angle. On an image, these differences mean about 6-7 and 1.5 pixels.

We have also asked our experts to rate the results obtained with our method. In their opinion, the algorithm produces satisfying result (only in a few cases they noted that it could have been done better).

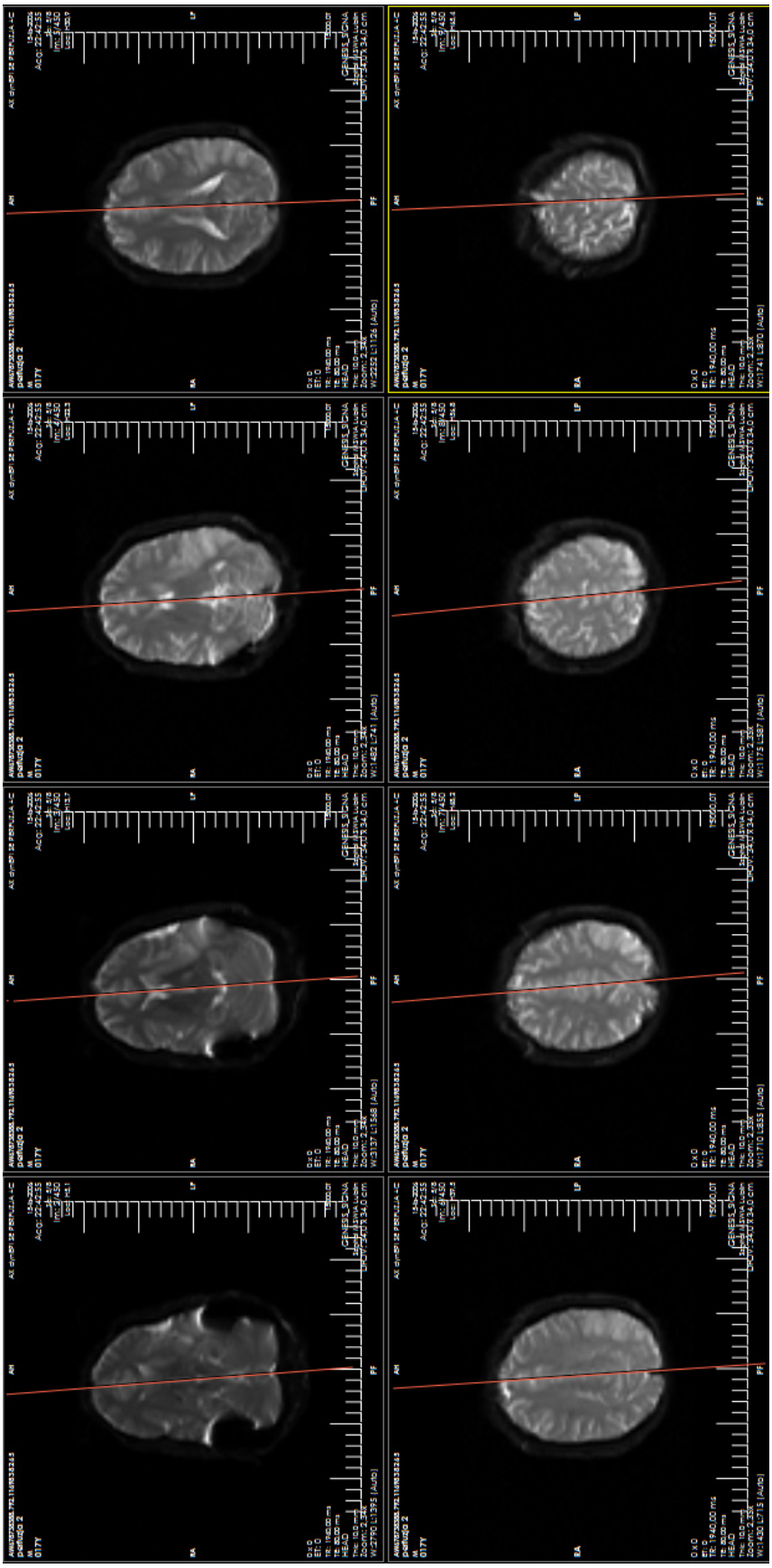

Figure 3.5: Model results of symmetry line finding for the whole set. Red lines denote symmetry line for the given slice.

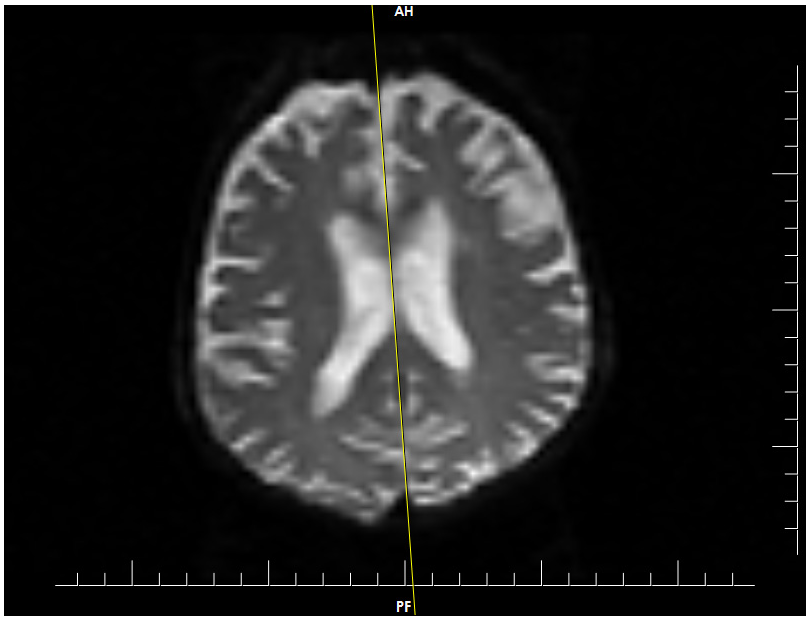

Figure 3.6: Model results of symmetry line finding for the whole set. Yellow line denotes symmetry line.

As mentioned at the beginning of this article, the algorithm will be used as part of other methods that need to distinguish between two cerebral hemispheres; therefore, the quality of symmetry lines found, even in *worst* cases, is quite satisfying.

3.1.4 Conclusions

The algorithm presented in this article may be used for finding symmetry lines on low resolution, sparse images of MRI brain perfusion studies. Because these images are acquired in a specific manner, the only way to locate a line between cerebral hemispheres is to analyse brain anatomy. Therefore; the method is trying to find characteristic anatomical places that describe the middle point of the brain. Using linear regression, a line is then fitted into a set of characteristic points found. Of course, some presumptions are made for this method. First of all, deviation of symmetry line from the vertical axis cannot be greater than 10-15% (which may occur only in special cases). We also do not require symmetry lines found to be perfect; a small deviation from the ideal case is possible.

After comparing the results acquired with results given by experts, we have come to the conclusion that the method produces satisfying results for diagnostically meaningful cases.

3.2 Perfusion parameters calculation and noise reduction

In this section we present our research into the subject of reducing the influence of noise on evaluation of perfusion parameters, such as CBF, CBV or MTT. Noise can be present on some pixels of study slices; therefore, it can lead to artifacts in calculated concentration time curves and blur the final results.

To minimise any influence from these factors, we propose a method that is different from those commonly used. Generally, noise reduction is done by filtering (smoothing, blurring), which dose not always produce good results, as much information from an image is lost. Therefore, it is more effective to use the interpolation methods.

We have studied different interpolation techniques and compared them numerically. Tests have proved that using our method leads to better, more accurate estimation of perfusion parameters. It also seems that large window *Sinc* interpolation gives the best results.

3.2.1 Classic approach

Perfusion weighted MR imaging is a technique that can provide information about functional status of cerebral tissue (as described in Chapter 1). It has been studied in various diseases of the brain since 1989 [24, 67]. Using the signal change that brain tissue experiences over time following administration of contrast agents, important hemodynamics parameters, such as cerebral blood volume (CBV), cerebral blood flow (CBF), mean transit time (MTT) can be relatively measured and mapped [68].

Quantification by means of the deconvolution method uses the arterial input function (AIF) obtained from major cerebral arteries and is thought to be highly susceptible to signal noise and technical error [27]. While evaluating perfusion parameters, algorithm analyses time steps images on a pixel-by-pixel basis. It is obvious that a small noise can change pixel intensity and thus values of concentration time curve (CTC), which leads to artifacts in perfusion parameters maps.

A classic and widely used approach to reduce noise on perfusion images is filtering (smoothing or blurring). Often very large kernels are used, even 5x5. These have a great effect on the quality of the image, which already is rather small (128x128 pixels). Such radical smoothing/blurring of the acquired data leads to a lot of information being lost. Therefore, we propose a technique, based on interpolation, that does not affect the image itself but uses values of pixels in the neighbourhood of the one being analysed to calculate a value of CTC at a point.

3.2.2 MRI Data Analysis

The perfusion study consists of series of images acquired in a short period of time after administration of the contrast agent. To perform such quick image acquisition, magnetic resonance is set to lower matrix (128x128) and interslice gap is increased. As a result, we obtain several hundreds of images, divided among several dozens of time steps (for example 45 time steps, 10 slices each).

In the classic approach, the first step of analysis is noise removal, by smoothing with a 5x5 uniform smoothing kernel [69]. In our case, this step is omitted and background removal is performed. This can be achieved by a simple thresholding to the certain value (see chapter 2.2.2).

AIF function was obtained from circular ROIs placed manually on the middle cerebral artery (MCA). To reduce the partial volume effect (PVE), we have searched the ROI for the maximal intensity value and chosen only pixels with intensity $>$ 99% of this value. Mean concentration time curve (CTC) was then calculated from these pixels and used as the AIF. We used left MCA AIF to calculate parameters of the left hemisphere and the right AIF MCA for the right cerebral hemisphere. To discriminate between hemispheres, we have used our Perfusion Symmetry Line algorithm from Section 3.1.

The signal intensities in the brain parenchyma were converted to transverse relaxation rates on the voxel-by-voxel basis. The signal time curves $I(t)$ were used to evaluate CTC ($C(t)$) with the following equation:

$$C(t) = -K * \ln \frac{I(t)}{I_0} \tag{3.6}$$

where: I_0 is the baseline signal, $I(t)$ is the signal intensity at time t after injection of contrast agent; $K = \frac{1}{TE}$ - TE is the echo time.

This data was then modelled assuming the Gamma Variate function by fitting the early part of the signal attenuation to avoid recirculation effect (see Fig. 3.7):

$$C(t) = A(t - t_0)^{\alpha} e^{\frac{-(t-t_0)}{\beta}} \tag{3.7}$$

where: α and β are shape parameters for the bolus. t_0 is extracted directly but other parameters must be extracted using the function fitted.

Next, $C(t)$ was deconvoluted with AIF ($AIF(t)$) using fast Fourier transformation to obtain $C_d(t)$ [70, 71] using the equation below:

$$C_d(t) = F^{-1}(\frac{F(C(t))}{F(AIF(t))}) \tag{3.8}$$

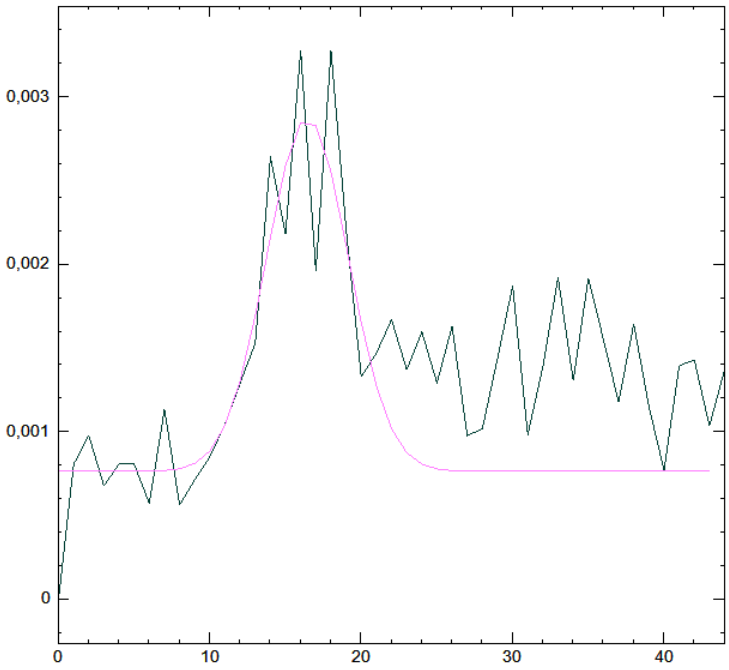

Figure 3.7: Gauss curve fitted into most significant part of the bolus. The X-axis represents time and the Y-axis shows the CTC value calculated (bolus).

The following formula was used to calculate CBV:

$$CBV = \frac{k_h}{\rho} \frac{\int_0^\infty C(t)dt}{\int_0^\infty AIF(t)dt} \tag{3.9}$$

where k_h is a factor (0.73) correcting the difference in hematocrit between the capillaries and large arterial vessels [72] and ρ is the density of the brain tissue ($1.04 g/cm^3$).

CBF is given by:

$$CBF = \frac{k_h}{\rho} C_{max} \tag{3.10}$$

where C_{max} is the maximal value of $C_d(t)$.

MTT was calculated as the ratio:

$$MTT = \frac{CBV}{CBF} \tag{3.11}$$

Time To Peak (TTP) is the time (value on the x axis, Fig. 1.3) when CTC reaches its peak value (maximal value on the y axis, Fig. 1.3).

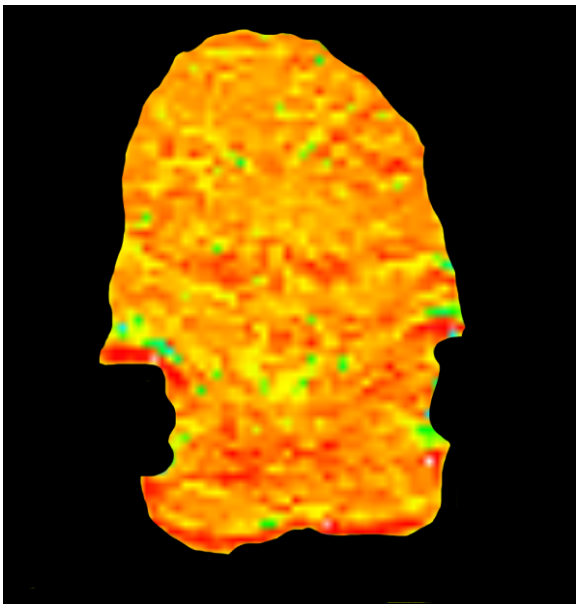

Figure 3.8: Visualization of perfusion parameter (CBV) for sample data set before noise removal. We have used LUT that enhances unusual, noise pixels. Colours represent different values of the CBV parameter in a given point. Green coloured pixels correspond to low value of the CBV, in this case mostly noise. Red and white ones are high values of this parameter.

These values where calculated for each slice acquired for the analysed period of time (in our case each study had 10 slices per time step) on the pixel-by-pixel basis. These lead to making spatial maps of perfusion parameters (see Fig. 3.8). Unfortunately, without noise removal maps contained some artifacts, which made it difficult to interpret the results of perfusion analysis. To reduce them, we propose the following method which has to be applied at the moment of sampling the signal time curve (equation 3.7).

3.2.3 Interpolated pixel sampling

Lets assume that we want to evaluate function value at a point $x \varepsilon R$ knowing its values in surrounding points. Function value I is:

$$I = \sum_{k} I(k) \cdot K(x - k) \tag{3.12}$$

where $K(x)$ is a continuous function named the interpolation kernel. Summation is done for k points neighboring with point x. Without losing the generality and to simplify calculations, we assume that the distance between consecutive k points is 1.

A general formula for calculation of discrete signal value at a real point is given by 3.12. We are use symmetrical and separable interpolation kernels [36], so we can

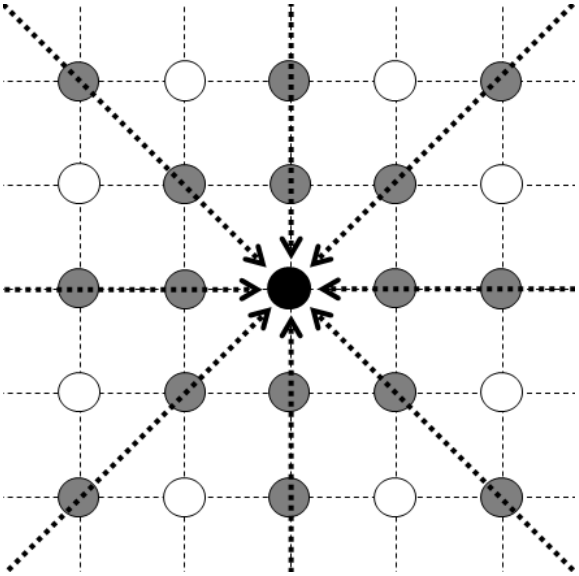

Figure 3.9: Interpolated pixel sampling.

write the formula for interpolation of the two dimensional signal as :

$$I(x,y) = \sum_k \sum_l I(k,l) \cdot K_{2D}(x-k, y-l) \tag{3.13}$$

where $K_{2D}(x,y) = K(x) \cdot K(y)$. In our deliberations we will understand $I(x,y)$ as the intensity of voxel of perfusion image at a point (x,y).

As mentioned earlier, image sampling for the calculation of concentration time curve 3.7 is done on a pixel-by-pixel basis. To ensure that sampled pixel P value is not affected by the noise, we analyse its neighbourhood as shown in Fig. 3.9. We consider pixels aligned on the horizontal, vertical and two diagonal lines passing through the analysed pixel on the plane of the slice. Note: because of the great distance between slices in the series, it is not possible to consider voxels from below and above the slices.

We assume that the P is not known and interpolate its value from four one dimensional directions. In the case of the diagonal neighbourhood, we assume distance between consecutive pixels to be equal to $\sqrt{2}$. As a result, we get four values of intensities. After excluding minimum and maximum value, an average of the two remaining values is calculated. This should minimise the influence of possible noise in surrounding pixels. Next, this average is compared with the actual signal intensity in point P. If the difference is larger than 10%, the interpolated value in chosen as the value of point P, otherwise, the original signal value is taken.

In the case of borders of the brain (that is in a situation when some consecutive pixels on the analysed lines are equal to 0), the method is not used, as it could bring more harm than benefits. Especially when information from the edge of the brain rarely presents data of diagnostic importance. The conditions for applying the above method are as follows:

- interpolation along the 1D line can be performed if at least $\frac{2}{3}$ of significant (belonging to interpolation kernel) points are > 0
- interpolation along at least 1 line is possible.

3.2.4 Methods tested

Because the portfolio of the existing interpolation kernels is very wide, we have chosen for analysis (only several methods). Table 2.1.2 (from Chapter 2.1) shows the methods we have tested. We have used simple and more advance methods. These are most widely used interpolation methods. They are evaluating function value using different number of neighboring pixels. The simplest use only two neighbors, but more complex (for example *sinc* family) can use larger number of points. Because *sinc* function, by definition, is infinite it must be multiplied by specific window. Research shows that the best results are obtained by odd sized kernels [44].

3.2.5 Test results

To objectively test our method, we have created a set of ideal slices without any noise distortion. We drew on the data from one of the clinics we are cooperating with and manually removed each abnormal pixel, creating almost clear images. The imaging protocol used consisted of perfusion study [echo planar imaging (EPI), TR/TE: 1940/80 msec, 10 slices with a 5mm thickness and 10mm interslice gap, matrix: 128x128]. For this ideal data set we have calculated all hemodynamics parameters and prepared a map of CBV, CBF and MTT values. Next, we have introduced artificial noise (of different strength) [73] into the test study and tried different methods of noise influence reduction. We have used noise grains of one and two pixels in diameter that were placed randomly with a probability of 5%, 10% and 20%. Noise pixels had maximal acceptable value. Results were then compared with the ones obtained for the ideal study. The mean distance between values of each parameter (for each corresponding pixel) was calculated and used as the criterion for quality assessment. Model results for these tests are show in the table 3.2.5.

After analysing average distances between values of CBF, CBV and MTT, together with standard deviations, we have come to the conclusion that interpolation methods are very effective in reducing noise effect, especially when compared with the classic filtering approach. Lack of noise reduction leads to visible artifacts in perfusion parameter maps, which in turn can lead to misinterpretation. Therefore, in fact,

method name	kernel size	grain ø1			grain ø2		
		5%	10%	20%	5%	10%	20%
Linear	2	0,0131	0,0231	0,0412	0,1022	0,1632	0,2843
Cosine	2	0,0113	0,0201	0,0356	0,1054	0,1778	0,2654
Cubic	4	0,0094	0,0121	0,0285	0,0687	0,1238	0,1843
Spline	4	0,0095	0,0163	0,0232	0,0621	0,1183	0,1743
Sinc Welch	5	0,0043	0,0082	0,0143	0,0354	0,0568	0,0872
Sinc Hann-Hamming	5	0,0035	0,0073	0,0112	0,0293	0,0432	0,0712
Sinc Lanczos	5	0,0038	0,0076	0,0121	0,0295	0,0456	0,0781
Filtering	3	0,0211	0,0356	0,0543	0,0844	0,1424	0,2172
	5	0,0308	0,0451	0,0811	0,1232	0,1804	0,3244
no noise reduction		0,0413	0,0803	0,1514	0,1651	0,3223	0,6050

Table 3.1: Results of test - average distance between values of CBV (normalized)

some action must be taken to minimise these artifacts. The traditional approach makes it possible, but has a degrading influence on the image quality and in our opinion should be avoided, especially when proposed interpolation sampling produces very good results. Even such a simple method as linear interpolation produces better results than the said filtering. Small kernel interpolation functions prove to be less effective when dealing with larger noise grains, but were very good for little, single pixel noise. For small and larger noise grains most universal are very effective *Sinc* family method, that use 5-pixel windows. Differences between *Sinc* windows are minimal, but Hann-Hamming window gives the best results (see Fig.). This method is unfortunately the slowest one; therefore, Welch window should be chosen, as it produces similar results and runs 60% more quickly.

3.3 Symmetry information for detection of ischemic changes

In this section we would like to present our proposition for the method allowing automated detection of ishemic changes on perfusion MR studies - the Symmetry Based Detection Tool (SBDT). Our approach uses the algorithm described in Section 3.1. It is important to note that it is not a three-dimensional approach like the algorithm described in the following section, 3.5. Here, we use information only from directly measured perfusion study slices. Therefore, this method is simpler and can be used and treated as a mid-step to more complex, statistical approach presented later (SPBM).

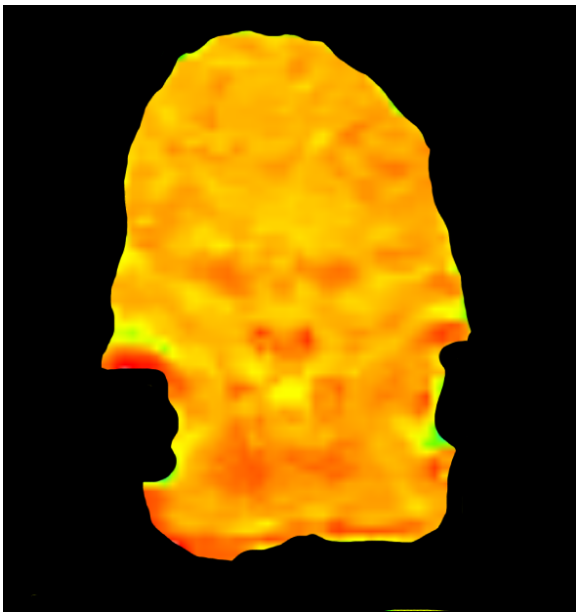

Figure 3.10: Visualization of perfusion parameters for sample data set after interpolated noise removal with Hann-Hamming *Sinc* kernel. Compare with Fig. 3.8.

The fact that this algorithm is two dimensional or less complex does not mean that it is worse than SPBM. It is just different. As our test showed, more than often this simple technique produces very good, sufficient results, in fact improving the diagnostic process. Additionally, it is not time consuming and can yield instant information about possible abnormalities in perfusion parameters in a certain parts of the slice. Of course, the method is not perfect or reliable. Like all semi-automatic diagnosis tools, it is just an aid to the physician. It can show something that is not seen at once, but will never replace a human.

3.3.1 Method description

The idea is that ischemic changes most often strike only one side of the brain. Additionally, if we compare perfusion parameters on both sides of the brain, we should be able to pinpoint the location of any abnormalities outside the given margin. Following this idea, we have developed the previously described algorithm (section 3.1) to find anatomical symmetry line on perfusion slices.

The first step is to find the symmetry line for a given scan. We assume here that segmentation and pre-processing was done as part of the PSL (perfusion symmetry line) algorithm. As a result, we get a line dividing two cerebral hemispheres on a given section of the brain (see Fig. 3.11).

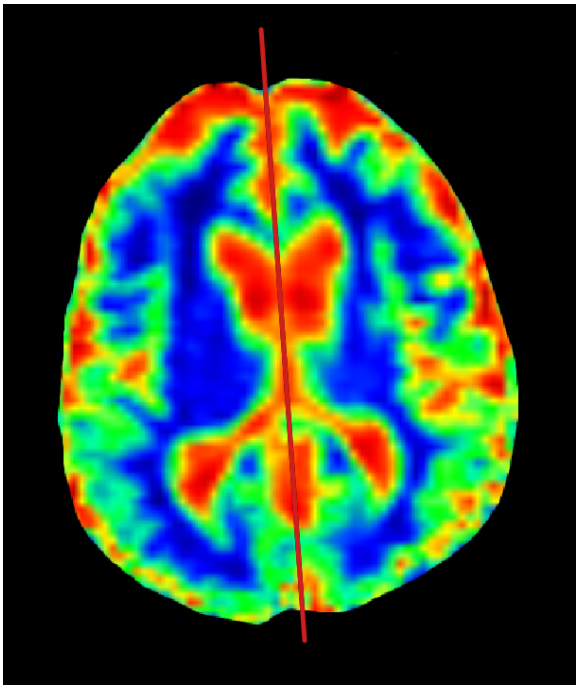

Figure 3.11: Perfusion slice with symmetry line marked

Next, we need to compare information on both hemispheres. It is not that obvious, as sizes and shapes of the left and the right one can differ, as a result of the image plane positioning in the 3D space. This issue was already discussed in the previous sections. Here we will use a method similar to the one we are using in the SPBM algorithms (see Section 3.5).

In order to compare parts of the brain (on the slice plane), we need to find a convex hull for the whole brain image (see Chapter C.4.3). This is necessary to level influence of inequalities of the brain surface. Next, we find the middle (X) of the segment created by the intersection points of the symmetry line with the convex hull. This symmetry line divides the brain into two parts - hemispheres. To make their comparison accurate, we divide their surface into n triangle shaped sections. Division lines pass through point X and points on the half's convex hull border. Points on the border divide it into n segments of equal length ($\frac{1}{n}$ of the total half's border length). Owing to the way we create these segments they reflect differences in shape of both parts and each "triangle" on one side has a corresponding one on the second. See Figure 3.12 for a visualisation of this idea.

For each triangle-shaped section, we define two vectors created by their sides (green dashed lines) and common point X (see Fig. 3.12 and 3.13). These vectors, denoted in Fig. 3.13 as v_0, v_1 and v'_0, v'_1, form a local basis of our barycentric coordinate system described in Section C.1.5. Each point on the plane can be described as a

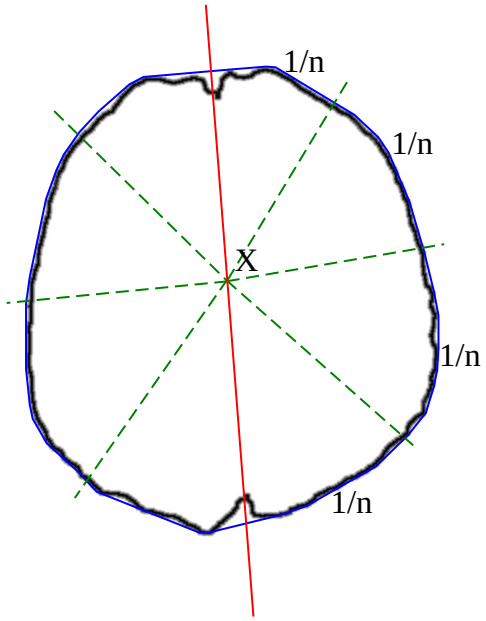

Figure 3.12: Sample borders of the brain. Blue lines represent convex hull over the brain border. Red line denotes symmetry line and green doted lines mark corresponding sectors on both halves

combination of these vectors:

$$P = X + u * v_0 + v * v_1 \tag{3.14}$$

We will, however, limit ourselves to the points within the angle created by the said vectors (that is $u \geq 0$ and $v \geq 0$). For each pixel P on one half of the brain, algorithm finds corresponding point P' on the second half (see section C.1.5). Point P is calculated as a function of values u and v (equation 3.14). An important advantage of such a solution, is that these values are normalised and describe proportions. Therefore, they can be used to calculate position of point P' in relation to point X, in space v'_0, v'_1, of the corresponding section on the second cerebral hemisphere.

$$P' = X + u * v'_0 + v * v'_1 \tag{3.15}$$

The higher the number n is, the more fine segmentation of the half's border is and the more triangle-shaped subsections are determined. The more triangles there are the more correct our approximation of the cerebral hemisphere slice is. Our empirical research showed that $n = 20$ is a good compromise between exactness and computation time, as the more triangles there are, the more operations must be carried out. On the other hand, a small number of triangles may not express the differences in the shape of the border of the brain.

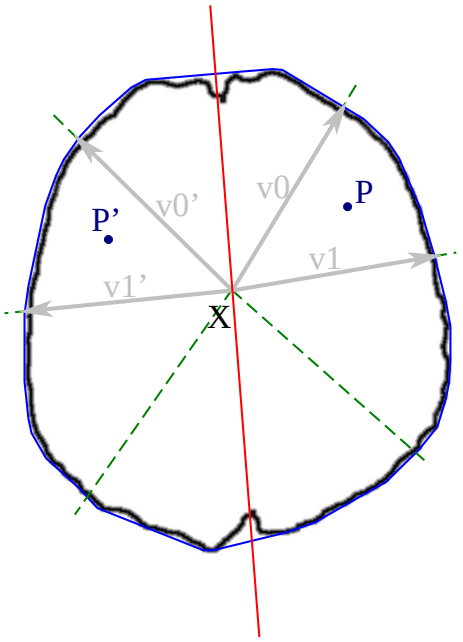

Figure 3.13: Brain parts/hemispheres with triangle-shaped sections marked. Each section has two vectors v_0, v_1 that can be used to describe point P inside it.

We are dealing here with maps, for example, of CBV, MTT, etc. It means that in each pixel of the slice there is one value of a given parameter. To find potential abnormalities in perfusion parameters, both cerebral hemispheres are compared. The algorithm chooses the larger one (in terms of pixels associated) and for each of its pixels P searches for a corresponding pixel P' on the second half. To do this, we must know in which triangle the point lies (see section C.1.5) and calculate its position in basis v_0, v_1. Next, we calculate the position of the point P' in the corresponding triangle on the other half, using equation 3.15. From this, we can get the position of the point in the standard coordinate system of the slice, thus getting the pixel we are searching for. Or not exactly yet. After translating position from v_0', v_1' coordinates to the standard ones, we will get a floating point value, which must be translated to integer values of pixel coordinates. We have chosen to use linear interpolation with kernel size = 2. See Section 2.1 for more information about interpolation techniques.

A few words of comment about the interpolation method chosen. We are drawing here on our research into interpolation of 2D images [6] and on our experience in this area. The operation we are performing here can be compared to texturing and the whole point is to acquire best visual effect. Therefore, the nearest neighbor interpolation was excluded from our deliberations as it produces rather poor results. A lot better is linear interpolation that gives us form of "mean" value of four pixels. This is also the method most often used in texture sampling. We could also use more

sophisticated methods, with larger kernels, but have come to the conclusion that it wouldn't improve the quality or could even give worse results. The reason is that we are dealing with small low res images, and introducing data from pixels far away could blur the real value, and a more natural approach is to use closest pixels.

Now, when for each pixel on one half we have anatomically evaluated a corresponding value on the second half, we can start to search for abnormalities. It is not that simple as there are differences in values caused by the anatomy (the comparing algorithm and the data to compare are not perfect). Therefore, the best way to tackle this problem is to use statistical methods. The idea is to compare deviation at a current point with the mean value of deviations in the whole image. Based on this mean and the maximal deviation, we have introduced a percentage above which deviations are treated as abnormalities:

$$DT(p) = D_{mean} + (D_{max} - D_{mean}) * p \tag{3.16}$$

where: DT stands for Detection Threshold, D_{mean} is mean value of deviations, D_{max} is maximal deviation from D_{mean} and p is the percentage - parameter. All differences in values of perfusion parameter between two corresponding points on both hemispheres, greater than DT at a given p are treated as abnormalities and denoted as such on the resulting image. This is a basic, simple, form of our method that can and will be developed in our further research.

3.3.2 Results and tests

Parameter p, from equation 3.16, is used as a way to regulate the correctness and sensitivity of our method. The only way to determine what its value should be is empirical research. This should also test our method and show if it is correct or not. Can this tool really be used to help in detection of ischemic changes? Sample results obtained with our method are shown in Figures 3.14 and 3.15.

When it comes to validation, results and tests, we want to show that our method is correct form the technical point of view and that it is accurate in detecting abnormalities in perfusion parameters. To test the first issue, we must check the correctness of the algorithm in comparing two differently shaped slices of cerebral hemispheres. To do this, we have prepared artificial images of the cerebral hemisphere. We have used masks of the real brain from one of our test studies, with symmetry line marked. Each pixel of the slice contained the information about its location on the plane. Next we have applied some predefined warping [74, 75, 76, 77] to the data in such a way that the symmetry line was a fixed area of the transformation and remained a reference point. During warping we have used a linear interpolation in the sense that the calculated point value was its floating-point coordinates between four discrete pixels (note: containing coordinates in 2D space).

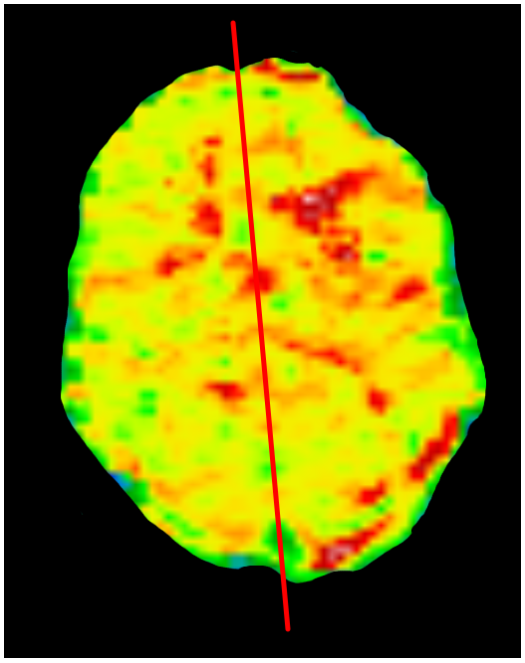

Figure 3.14: MTT map with the symmetry line marked. Abnormalities can be easily noticed - dark red regions.

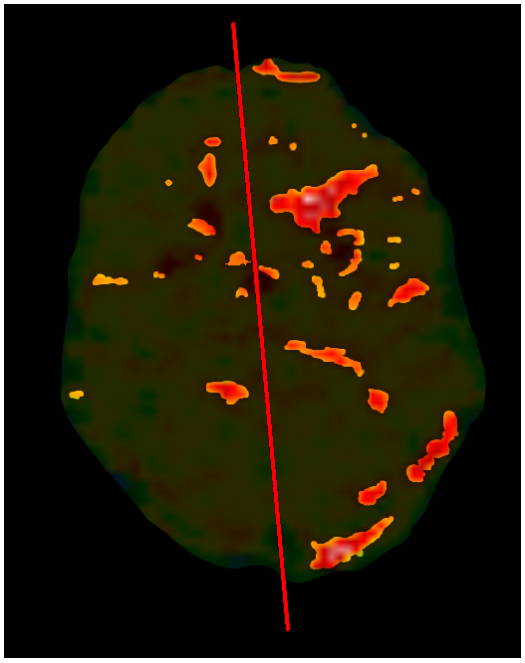

Figure 3.15: Sample results of the method used to detect ischemic changes with $p = 0.6$. Red, not dimmed pixels represent regions detected as abnormal.

We have used five predefined warping meshes on three different artificial slices. After traditional, well tested, warping of the cerebral hemisphere slice, we have used our technique to compare morphed slice with the original one. Thanks to the information about position of each pixel in the space we could asses what the mean distance of the twice morphed point from its starting position is. In a perfect case, we would return to the original position, but nothing is so perfect. For all cases analysed, fifteen data sets in total, we have obtained the average distance of 1.9 pixel. The minimal distance was 0.0 and the maximal 3.9 pixel (in most distorted images). The standard deviation from the mean was 0.8. In our opinion, these are very good results, taking into consideration the fact that this is after two different warping / approximation operations. These numbers also show that visual effects that look very good to the "naked" eye are in fact justified.

We cannot objectively test the usefulness and correctness of this tool from the medical point of view, as this requires expert knowledge. We have asked physicians form our cooperating clinic to help us with this. In the case of this tool, study acquisition technique, equipment used, etc. also play a major role and have a direct impact on the resulting images and thus on our tool. The best working value of p for one hospital can be incorrect for another. Therefore, we have implemented this method in a way allowing direct control over parameter p, from the GUI (Graphic User Interface) level. The user can define a default value that is most useful to him.

To find this value for our partner, we have asked doctors to use our tool to find ischemic changes on perfusion studies. We had nine studies at our disposal, each containing ten slices. Of course, hemodynamic abnormalities were present only in some images. The task was to find, for each image, a scope of the p value (gradation was 0.05, that is 5%), at which changes were highlighted in the most accurate way. We have also asked physicians to note values of p at which changes that are difficult to notice are shown by our tool. In total, we had 25 slices that were analysed by two radiologists. After getting all results, we have found out that the most useful range of the p value is $(25 - 70\%)$.

Following the physician's suggestions, we have omitted regions too small to be a real abnormality and we did not take them into consideration in our result analysis. Using our tool, physicians were able to find most abnormalities in all cases, and all ischemic changes in 92% of analysed cases. In all these cases, regions of abnormal behavior, of perfusion parameters, marked with our algorithm, were in correlation with the ones marked by the physicians. Of course, they were not always exactly marked, but we have treated algorithm indication as correct if a physician said it was ok. In 72% of cases, there were also some regions marked as abnormal, while they were not. They amounted to 12% of the total pixels marked (mean value). It was mostly caused by a badly located perfusion slice that was not perpendicular

to the brain symmetry plane; thus a slice showed different parts of both cerebral hemispheres, not corresponding anatomically (see Fig. 3.23 and 3.24). This should be taken into consideration while performing perfusion study acquisition.

The acquired range of the p value was rather widely spread, unfortunately, but it did not vary much from our predictions, as it is obvious that calculated perfusion maps contain many different values and can vary from case to case. More importantly we showed that our approach can aid physicians. In several cases doctors were able to see changes that normally would only be noticed after a more detailed examination. This is because visualisation of perfusion parameters dose not reveal all information. Spread of data is limited to the value from 0 to 255. Our tool takes into consideration the original data and even if it is not automatic, it can quickly pinpoint regions that should be taken into consideration as possible places of ischemia. Error ratio is rather low, too.

Unfortunately, we had access only to studies and experts from one hospital. In our further research, we want to test our approach in other clinics. We also want to test it, asking more physicians for help. Now we only could ask two doctors. As always in such cases, this is caused by their in availability - when they can devote some of their time to us.

3.4 Brain characteristic planes - BCV algorithm

In our further research into the subject of perfusion analysis, we need to compare brains of completely different persons. Of course, brains will have different sizes and their images will be acquired differently. Therefore, there is a need to find some way to write coordinates of the point in one brain in terms of coordinates in another. In other words, some form of registration is required that will fit one brain to another. In some research projects it was done by attaching fidicual markers to a skull before imaging [78, 79, 80]. These markers were then used to fit one modality image to another. This, however, was limited to different modalities of one patient. Our problem is more complicated.

Because registration in this form (multi-modal, multi-patient) would be very difficult (would require special metric and optimiser, which are difficult to develop), we wanted to approach this problem differently. We want to find characteristic planes of the brain / head, together with their intersection, that will be a starting point to compare two brains. Such a point together with plane would allow us to align brains of two different persons, so that one brain could be "morphed" into another without any need of additional rotation whatsoever. It is very important to say that we do not require a very accurate alignment, as perfusion slices have very small resolution and potential error while fitting them is greater than that possible here. This issue

will be extended and explaneed better in the following sections, for now we want to focus on the method we have developed to find the planes mentioned.

3.4.1 Eye middles

The starting point for the algorithm is detection of natural markers on the head - eyes. These are very good markers, as from the anatomical point of view [81, 82] they are similar (in shape an of size) among people and located in the same place. The dimensions differ among adults by only one or two millimetres. The vertical measure is generally lower than the horizontal one and is about 24mm among adults and about 16-17mm at birth. The eye is not a perfect sphere, it is rather a fused two-piece object. The smaller unit in the front called the cornea is linked to the larger object called the sclera. The cornea is typically about 8 mm in radius. The sclera constitutes the remaining $\frac{5}{6}$, and its radius is about 12 mm. The sclera and cornea are connected by the limbus.

We use the previously described segmentation method (see section 2.4) to find eyes in the T2 series of the study. An obvious problem is that slices may be acquired on such a plane that it is not passing through eyes horizontally, therefore sizes of eyes, shown on the slice, may differ. Another problem is that image acquisition plane may not pass through the real middle of the eye. To find it, we assume that eye is a sphere. It is a good assumption since, as mentioned before, eyes are nearly a sphere. Of course, the middles found can lay in the part of the head not directly shown on slices.

Best fit sphere When eyes have been segmented, their contour is analysed and sampled to acquire points for fitting. At this point, it is very important to take into consideration the real position of the voxel in the scanner coordinate system (see chapter B.3) and slice thickness (distance between two consecutive slices). Having ImagePositionPatient, PixelSpacing and ImageOrientationPatient values (from the DICOM file), one come easily calculate the real voxel coordinates. The next assumption is that we do not sample eyes in the front $\frac{1}{3}$-th of the eye, because this is the anatomical location of the cornea, which would interrupt fitting.

A sphere can be described by its centre (x_0, y_0, z_0) and radius r_0. Any point on the sphere satisfies the equation

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2. \tag{3.17}$$

A minimizing function must be identified to get an initial estimate for the centre and radius. Let us consider the function

$$f_1 = r_i - r \tag{3.18}$$

where $r_i = \sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}$. Differentiating this with respect to x_0, y_0, z_0 and r_0 will give equations that are difficult to solve. Therefore, we use the following function

$$f_2 = r_i^2 - r^2. \tag{3.19}$$

This function can be rewritten as

$$f_2 = (r_i - r)(r_i + r) \approx 2r(r_i - r), \tag{3.20}$$

since $r_i + r$ can be approximated as $2r$. Differentiating this with respect to x_0, y_0, z_0 and r_0 gives initial estimates for the centre and radius.

Thus, the minimizing function (to find initial estimates for a sphere) is $f = r_i^2 - r^2$. Expanding this function, we get $f = (x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2 - r^2 = -(2x_ix_0 + 2y_iy_0 + 2z_iz_0) + \rho + (x_i^2 + y_i^2 + z_i^2)$ where $\rho = (x_0^2 + y_0^2 + z_0^2) - r^2$. ρ was introduced to make the equation linear. The above set of n equations can be expressed in the form of matrix:

$$\begin{bmatrix} -2x_1 & -2y_1 & -2z_1 & 1 \\ -2x_2 & -2y_2 & -2z_2 & 1 \\ \dots & \dots & \dots & \dots \\ -2x_n & -2y_n & -2z_n & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ \rho \end{bmatrix} - \begin{bmatrix} x_1^2 + y_1^2 + z_1^2 \\ x_2^2 + y_2^2 + z_2^2 \\ \dots \\ x_n^2 + y_n^2 + z_n^2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{bmatrix} \tag{3.21}$$

After getting the initial estimates for the centre and radius r, the Gauss Newton method was used to calculate values for the centre and radius [83, 84]. The minimizing function is given by $d_i = r_i - r$ where $r_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2}$.

3.4.2 YZ characteristic plane

Applying the above algorithm to both segmented eyes gives us their real centre. Both centres create *image eye segment*. We could create a *YZ characteristic plane* at this point, taking into consideration the fact that it should pass through the middle of the *eye segment* and be perpendicular to it. Tests shows, however, that in some cases some deviation is caused by some inaccuracies in eye middles evaluation. Such errors are visible only in faraway places of the data set. Nonetheless, we propose a solution to this problem.

We go back to the 2D slices of eyes. The algorithm searches for the middles of eyes (centroids - see Section C.1.4), on a slice plane, where eyes are represented by circles larger and less different in size. As a result, we get two points representing *slice eye segment* (see Fig. 3.16).

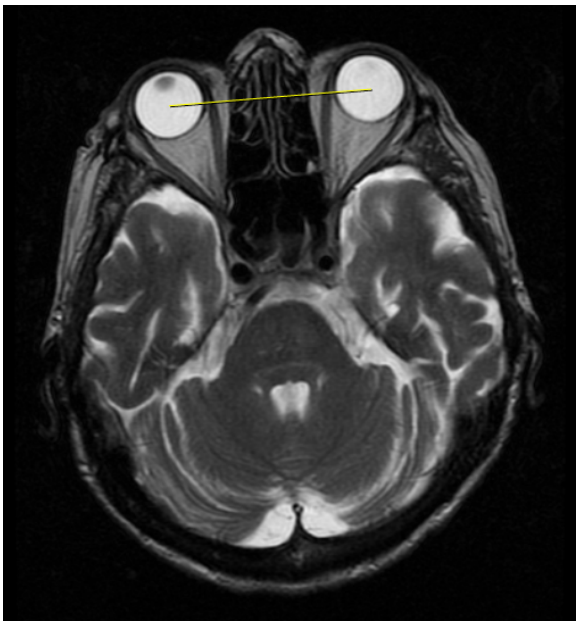

Figure 3.16: MRI slice with *slice eye segment* marked

The next middle of the *slice eye segment* is found. Also, bisection of this segment is evaluated. Next, the algorithm searches for the anatomical object placed on the symmetry line (or near it) as the second point describing the *YZ characteristic plane*. First, thresholding of the T2 data set is done. Values smaller or equal to 45% of the maximal voxel value in the set are removed. As a result, we obtain a set of connected components - isolated objects on the slice. Next, the algorithm removes:

- all objects outside the rectangle described by two points: $(slice\_width * 0.3, slice\_height * 0.3)$ and $(slice\_width * 0.6, slice\_height * 0.8)$,
- all objects whose containing rectangle is smaller (in diagonal) than 10 pixels,
- all objects whose containing rectangle is larger (in diagonal) than 50 pixels,
- all objects not intersecting with the *slice image eye segment* bisection line.

and leaves only objects whose centroid is closest to the bisection line. A sample result is shown in the Fig. 3.17.

The centre of this object, after calculating its coordinates in the scanner coordinate system, is used as the second point characterising the *YZ characteristic plane*. For the third one (three points are sufficient to describe a plane), the algorithm searches on the XZ plane that passes through the *eye segment* and is perpendicular to the XY scanner plane - see Fig. 3.18. Most important is the image on the top right-hand side, showing coronal section of the dataset - the XZ analysis plane mentioned. Algorithm

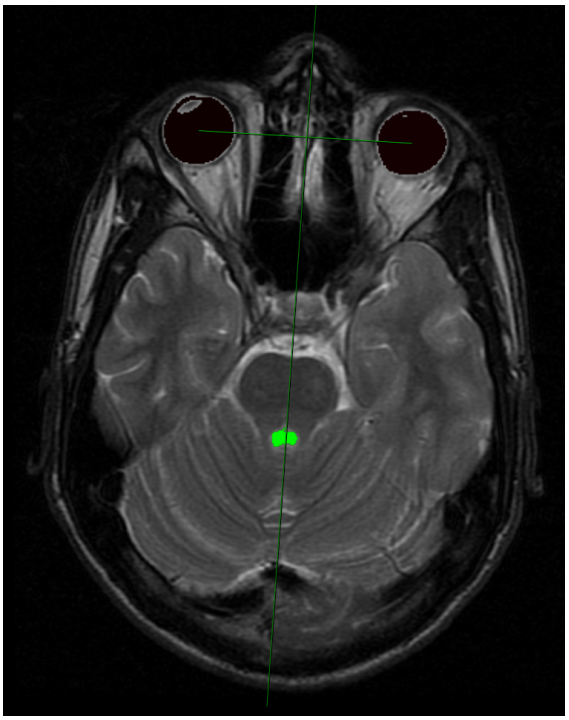

Figure 3.17: Symmetry line described by the middle of the *image eye segment* and centroid of the anatomical object.

analyses pixel values along the projection of the *eye segment* onto image plane on different slices. Fig. 3.19 shows, as an example, distribution of voxel intensities along the line shown in Fig. 3.18. We can see that near the middle of the segment there is a characteristic local minimum, surrounded by local maxima. This notch, low contrast pixels, represent a space between two brain hemispheres. From the anatomical point of view, such spacing between hemispheres should be visible somewhere on the analysed section.

The analysis is started on the slice located in the middle of the vertical distance between the eyes and the top of the brain (last slice in the data set). If the local minimum described is not found near the middle of the scanned segment, the higher (closer to the top of the head) slice is analysed, and so on. Loop stops when minimum is found. Its spatial position in the scanner coordination system is calculated and used as the third point to estimate the *YZ characteristic plane*. Knowing three points on the plane, the algorithm calculates it using formula from chapter C.1.2.

3.4.3 XY and XZ plane

At this point, when one plane has been evaluated using the anatomical structure of the head, some additional assumptions must be made to find the last two planes.

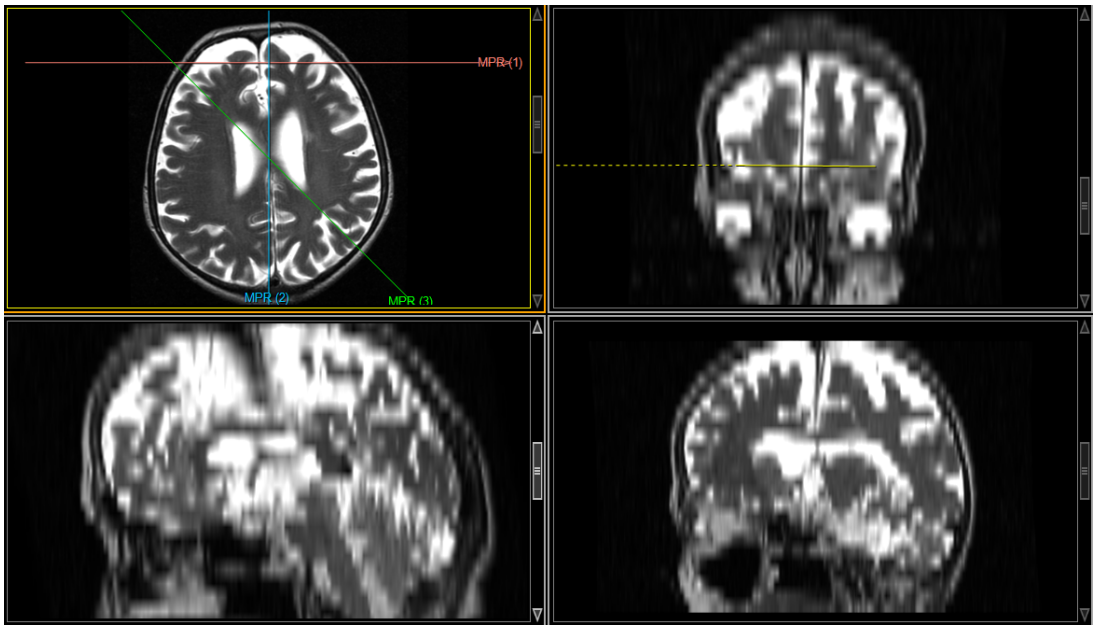

Figure 3.18: MPR visualisation of the sample dataset.

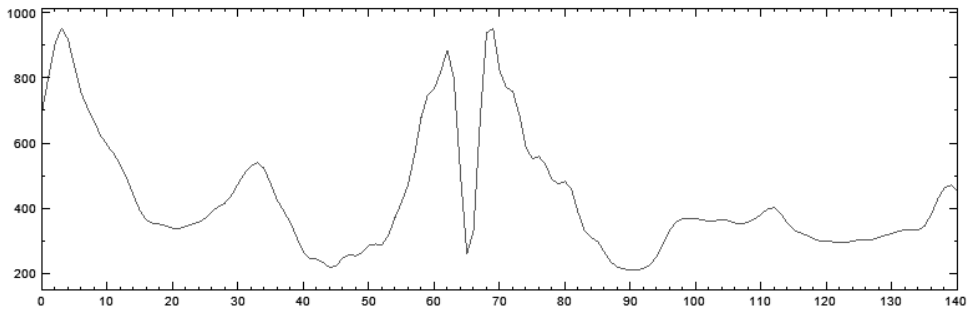

Figure 3.19: Line histogram of the segment shown in the Fig. 3.18. The X-axis represents pixel position on the line and the Y-axis its intensity.

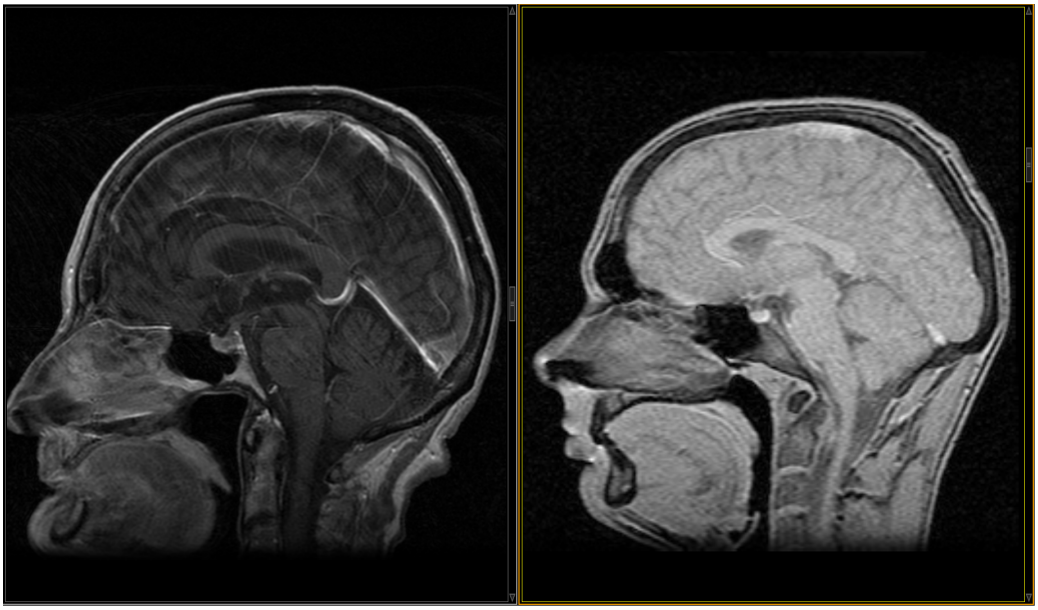

Figure 3.20: The sagittal view of heads of two different patients

We can make a good assumption that in most cases analysed all patients will be positioned in the way ensuring that *XZ characteristic plane* (for each patient) is perpendicular to the scanner's XY plane (see chapter B.3). This is almost always the case because while performing a scan,the patient is lying on his back with the head even with the back. A different alignment would require the patient to shift his head (i.e. unnaturally getting his chin close to the chest), which is a very inconvenient way to lie. See the Fig. 3.20 showing two different patients in sagittal section. One can see that the heads are aligned similarly in the scanner coordinate system. Any deviation form this should not affect any further comparison between different brains.

We expect the *XZ characteristic plane* to pass through the *eye segment middle point* and to be perpendicular to the *YZ characteristic plane* and the XY scanner plane. To calculate such a plane the algorithm creates two vectors, described by three points from the previous section (calculation of the *YZ characteristic plane*) with the common point being *eye segment middle point*. Next, an orthogonal vector to these two vectors is found by simple cross product ($A \times B$) calculation. This way the *XZ characteristic plane* has been described.

The Calculation of the last characteristic plane is now formality. It is perpendicular to both previous planes and passes through the *eye segment middle point*. Again, the cross product of two vectors is calculated. Obviously, vectors normal to YZ and XZ planes are used.

This way, real anatomical position of the brain in the scanner coordinate system can be calculated. We have called this set of characteristic planes the Brain Characteristic Vectors (BCV) and their intersection Brain Characteristic Point (BCP), and will refer to them as such in the following text.

3.4.4 Results and validation

Figure 3.21 shows characteristic planes intersections with different head slices. As can be noted, visual effects are good. They, however, should be verified. We have come to conclusion that there are two ways of doing this. First of all, we can use expert knowledge and compare our results with those obtained by the physicians. Such a test, for obvious reasons, is subjective. Another way, more objective, is to prepare a semi-artificial study that has predefined characteristic planes.

We have chosen the latter, objective method. We have prepared an artificial 3D data set containing eyes, and other structures in the head used by our algorithm. To prepare such complex data, we have used very well aligned and dense real study (real voxel size in the Z dimension was the same as in the other dimensions). It was slightly modified by us to become symmetrical and have all the needed structures well defined. Also all characteristic planes were evaluated for this basic data set. It was easy as all structures were "perfectly" arranged and prepared.

Next, we have applied some random rotations and translation to the data set, generating twenty different sets. Because the BCV algorithm in the form presented assumes no rotation along the Y axis, such transformation was not applied during tests. To make tests as close to reality as possible, we have used a combination of translation and rotation along the Z and X axis. The allowable angle was up to $\pm 15^0$ and translation vector length up to $\pm 10\%$ of the size of the set in the current direction.

Parameters of transformations were chosen randomly, maybe not exactly randomly, because we have analysed generated values and taken only the ones covering the entire range of allowable angles and vectors. Next, to each transformed data set, the BCV algorithm was applied and characteristic planes were evaluated. As we exactly knew how each set was obtained, we could calculate what characteristic planes should look like in each case. Therefore, we have a reference we can compare our results with. For obvious reasons (many approximations, rounding etc.), a perfect result cannot be achieved.

For all analysed cases we have calculated:

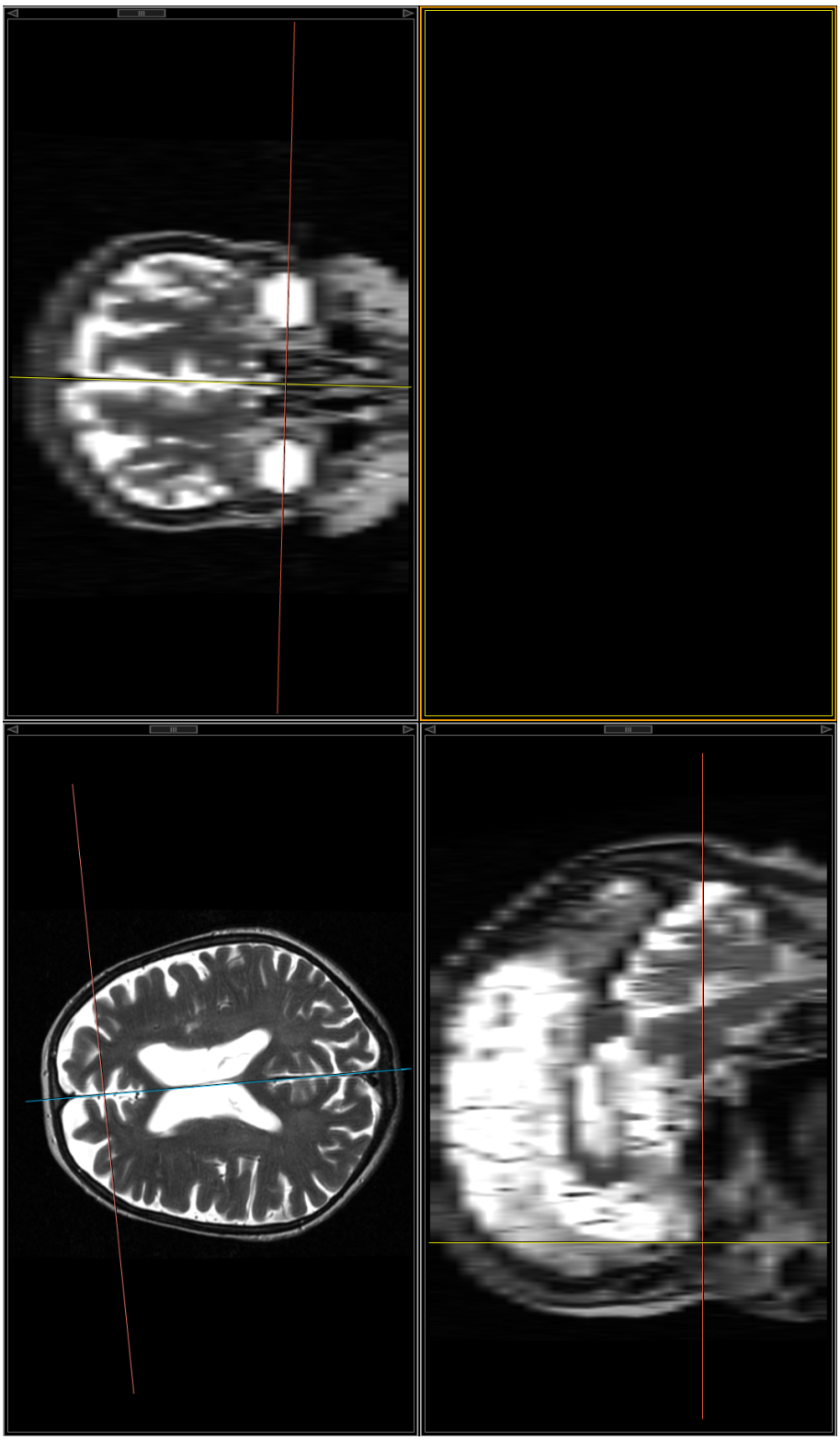

Figure 3.21: Characteristic planes shown on different MPR views of the head

1. the mean value of differences between the original BCP and the calculated one, in terms of vector created by pairs of points (the original and the calculated one)
2. the standard deviation for the above values
3. mean values of deviation between the original BCV and the calculated ones, in terms of angle difference in each direction
4. the standard deviation for the above values

As a result, we obtained:

1. the mean difference in the BCP placement: $(1.1\%, 1.5\%, 1.4\%)$ (percentage was calculated relative to the largest data set size in one direction - usually 512)
2. the standard deviation for the difference in BCP: $(0.5\%, 0.6\%, 0.5\%)$
3. the mean difference in the BCV angles: 0.8^0 for the X axis, 0.7^0 for the Z axis
4. the standard deviation for the BCV angles: 1.1^0 for the X axis, 0.9^0 for the Z axis

These values show that the method works pretty well. The BCP can be calculated within several pixels accuracy. The same situation is with the BCV angles that can differ up to several degrees. Of course, a lot depends on the data set, but inadequates are within acceptable limits. Therefore, the BCV algorithm developed can be treated as a useful and exact way to find characteristic planes of the brain.

The usefulness of the method presented, to compare two different brains will be assessed in the next section. In our further research, not described in this thesis, we would like to extend this method in order not to be dependent on the assumption about lack of the rotation along the Y axis.

3.5 Statistical Perfusion Brain Model

To make analysis of perfusion studies easier and more correct, we propose to use widely known and respected statistical methods [85, 86]. The idea is to calculate a model data set of perfusion parameters for the healthy brain. Such a model would tell us what value a given hemodynamic parameter should have in a given part of the brain. To be more precise it would give us statistical distribution of a parameter in a 3D point plus information about blood flow behaviour in time (fourth dimension).

Before we get to the description of the methods used to calculate SBPM (Statistical Brain Perfusion Model) one assumption must be made. The model will be very sensitive, as hemodynamic parameters are sensitive to technique of study acquisition. Therefore, it would be best to use studies made in one hospital, under a given methodology. It will be very difficult to get studies acquired with the same technique from two hospitals, as they use different equipment and methodology - unification is almost impossible. The best solution is then to restrict oneself to one hospital, its equipment and methodology. As mentioned earlier, when segmentation methods were described, software would be used in one hospital anyway, so the program could be "trained" using studies from that institution.

The SBPM can be acquired by analysing sets of perfusion studies of a healthy brain. Of course, the more studies we analyse, the more correct the model will be. As mentioned in Chapter 1, there are four main parameters describing perfusion in the brain: TTP (Time To Peak), MTT (Mean Transit Time), CBV (Cerebral Blood Volume) and CBF (Cerebral Blood Flow). Our model will describe statistical distribution of values for each of these parameters.

From the point of view of computer science, the SBPM is a set of three dimensional arrays (though they will be implemented as a single dimensional array) where a single field represents a statistical distribution - a list of observed values of the perfusion parameter and the CTC in a given place of the brain. Thanks to such a approach, we will get values that the parameters have reached in all cases analysed, which is a great input for further statistical analysis. Because, as mentioned earlier, perfusion images are acquired with a smaller matrix, typically 128x128, the SBPM matrices will be relatively small. To recapitulate, we are dealing here with three dimensional matrices (about 128x128x100) of lists of parameters values.

3.5.1 General approach

The SBPM generation algorithm has been implemented in the way ensuring full automatism of its calculation. As the input, it takes list of studies, and as a result, it returns SBPM matrices. Of course, later further studies can be added to the model.

The first step is to choose the model shape of the brain. To do this, brains from studies are segmented using the algorithm from Section 2.5, measured and the largest brain is chosen. The reason why we decided to use the biggest brain and not for example an average one, will become clearer once the filling of the model has been explained. Generally, it is due to technical reasons (interpolation) and it facilitates model's construction. Also, it seems to make no big difference as variations of sizes of analysed organs are not significant.

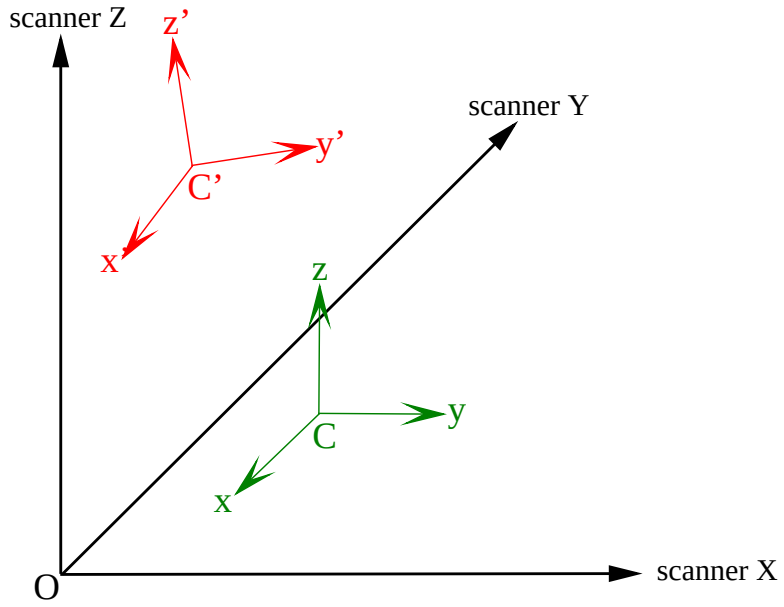

Figure 3.22: Model position of two different BCV's in the scanner coordinate system

At this point, differences in brain shapes are also important and must be taken into consideration. Generally, the problem is how to link position of the point in one brain with coordinates of the point in the second brain. It can be achieved using the algorithm for finding the head characteristic planes described in Section 3.4. These three perpendicular planes (BCV) together with one point - their intersection (BCP) are unique for each brain and always pass through the same anatomical structures in the head. Therefore, they can be used to compare brains of two completely different people.

3.5.2 Transform

Let us call BCV's of two brains we want to compare: BCV and BCV'. We are looking for a transformation that will rotate and translate one vector set into another; in other words, describe point P written in basis BCV as a point P' in basis BCV' and vice versa. The basis of two coordinate systems BCV and BCV' can be written as follows (see Fig. 3.22):

$$BCV = \{x :< x_x, x_y, x_z >; y :< y_x, y_y, y_z >; z :< z_x, z_y, z_z >\} \tag{3.22}$$

$$BCV' = \{x' :< x'_x, x'_y, x'_z >; y' :< y'_x, y'_y, y'_z >; z' :< z'_x, z'_y, z'_z >\} \tag{3.23}$$

To make calculations easier, let us assume that the BCV coordinate system is the main one and we will only move between these two systems. To transform point P

in BCV coordinate system to a point in the BCV' system, it is enough to calculate the dot product of P with x', y' and z' respectively. This is the matrix of the transformation:

$$M_{BCV'} = \begin{bmatrix} x'_x & y'_x & z'_x & 0 \\ x'_y & y'_y & z'_y & 0 \\ x'_z & y'_z & z'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

If we now take point $P(a, b, c)$ and transform it against this matrix to point $P'(a', b', c')$, we get

$$P' = P * M_{BCV'} = \begin{bmatrix} a & b & c & 1 \end{bmatrix} \begin{bmatrix} x'_x & y'_x & z'_x & 0 \\ x'_y & y'_y & z'_y & 0 \\ x'_z & y'_z & z'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

$$P'((a * x'_x + b * x'_y + c * x'_z), \tag{3.26}$$

$$(a * y'_x + b * y'_y + c * y'_z), \tag{3.27}$$

$$(a * z'_x + b * z'_y + c * z'_z), \tag{3.28}$$

$$1) \tag{3.29}$$

So the dot product elements we are searching for are:

$$P' = (P \cdot x', P \cdot y', P \cdot z') \tag{3.30}$$

The last step is positioning. We need to make two operations: translation and rotation $T^{-1}_{CBV} R^{-1}_{CBV}$. The first matrix is translation, the second rotation. We already have a rotation matrix called $M_{BCV'}$, so we only need a translation matrix:

$$T^{-1}_{BCV} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -t_x & -t_y & -t_z & 1 \end{bmatrix} \tag{3.31}$$

where: $t = (t_x, t_y, t_z) = C' - C$, where C and C' are the centre points of BCV and BCV' coordinate systems respectively. Finally, we get:

$$T_{BCV'} = T^{-1}_{BCV} * M_{BCV'} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -t_x & -t_y & -t_z & 1 \end{bmatrix} * \begin{bmatrix} x'_x & y'_x & z'_x & 0 \\ x'_y & y'_y & z'_y & 0 \\ x'_z & y'_z & z'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.32}$$

$$= \begin{bmatrix} x'_x & y'_x & z'_x & 0 \\ x'_y & y'_y & z'_y & 0 \\ x'_z & y'_z & z'_z & 0 \\ -(t \cdot x') & -(t \cdot y') & -(t \cdot z') & 1 \end{bmatrix} \tag{3.33}$$

The above matrix applied to a point in the BCV' coordinate system will transform it to the point in the BCV coordinate system. To get inverse translation we just need to inverse the matrix $T_{BCV'}$, that is $T_{BCV} = T^{-1}_{BCV'}$. Using this two transformations we can easily move between the BCV and BCV' coordinate systems and as a result express any point P in one system as a function of point P' in the other. This only tells us how to align two datasets in the space so the $BCV's$ of both of them wil be in the same place. This does not solve the issue of different brain sizes.

3.5.3 Localise slices

To realise exactly what we are dealing with, see Figures 3.23 and 3.24. Figures show typical location of perfusion slices with respect to the whole brain. As we see, they are located differently than general axial slices in a normal study and far away from each other on the Z axis. The $SPBM$ we want to create will be more detailed and contain information about the whole volume of the brain. It is obvious that it will not be populated entirely at the beginning, as each study gives us information only about ten slices in the brain, while there is a place for a lot more. Very often different studies will provide data about the same brain parts, so there will be places where we have more information and places that are a form of "white spots" on the map. Therefore, we see that many studies must be used to create accurate model covering for the whole brain. However, taking into consideration that in the same hospital studies are performed in the repetitive way, the odds are that the early $SPBM$ will have enough information to provide an accurate answer whether or not the study analysed has some abnormalities on the hemodynamic parameters map.

Let us get back to the description of how the model is filled. First of all, we need to know what the differences in brain sizes are on the Z axis, between two brains compared. To do this, the tip of the brain is found. The algorithm is rather simple. The YZ characteristic plane is scanned top-down line by line, until the brain tissue voxel is found. It is important to note that lines that are being scanned are parallel to the XY-YZ characteristic planes intersection line. This way, when we compare distances of the voxels from the respective XY plane on both data sets, we will get size proportions between them.

Our next idea is to find a plane in 3D space of the $SPBM$ that corresponds to a given perfusion slice in the data set added to the model. We do it for each of about ten different slice positions that are acquired during perfusion study (Fig. 3.23 and 3.24).

As usual, the position of each of perfusion slices in the study is described in the DICOM header. Therefore, we know the exact location in the scanner coordinate system. From this, we can describe its position in the BCV basis. Next, we search

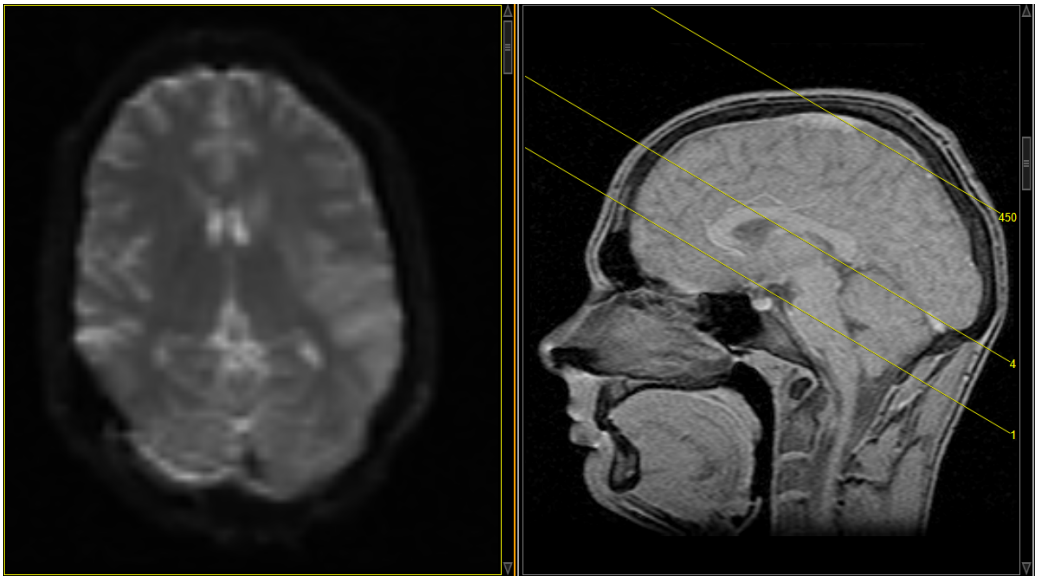

Figure 3.23: Typical arrangement and localization of perfusion slices in the brain (sagittal plane)

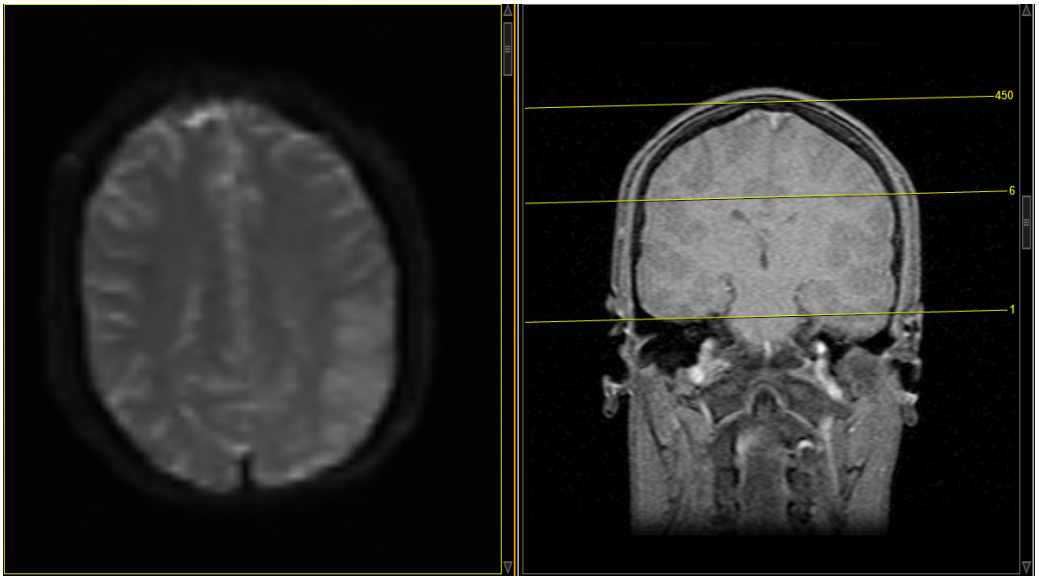

Figure 3.24: Typical arrangement and localization of perfusion slices in the brain (coronal plane)

for points that describe a given plane in the BCV system. We are interested in four intersections of the YZ and XZ characteristic planes with borders of the brain. After the brain has been segmented this is a very simple task - just scan the intersection lines (intersections of characteristic planes with the slice plane form two intersecting perpendicular lines) from both sides, till we find a brain voxel.

Using the previous notation, we are in the BCV' space. Using the transformation matrix $T_{BCV'}$ we shift our four points into the BCV space of the model data set. Here, using these four points we find a plane corresponding to the slice plane on the added data set. The points transformed should form a plane, but to be sure that any rounding did not influence their correctness, we use a suitable algorithm (see chapter C.1.3) to find best fit plane for these set of points.

3.5.4 Compare slices

We will be filling this plane in the model data set with the data from the corresponding slice in the added data set. That is, for each point $P(x, y, z)$ on this plane that is within the model brain we will look for a corresponding point $P'(x', y', z')$ within the brain on the analysed slice of the added data set. To do this, we also need to find four intersection points, on the slice plane, of the XZ and YZ characteristic planes with borders of the model brain. Because calculated section / plane in the BCV space may have some floating-point coordinates, there will be a need to calculate voxel values in discrete space using interpolation. Using information from Chapter 2.1, we have chosen the Nearest Neighbor interpolation.

Next, we find borders of brains on both slices - the model and the added one. In the case of the one that is being added there is no problem as we are dealing with the real slice that contains acquired data. In the case of the model brain, as mentioned in the previous paragraph, we are somewhere in the 3D space that may not contain real data and is located in a floating-point, not a discrete space. Therefore, we use a modified border tracking algorithm. The algorithm starts form the top of one of the four intersection points we have found earlier, and during each step analyses the neighbourhood of the current point using 3D nearest neighbor interpolation (see chapter 2.1) to check where the next border voxel is on the plane. After this step, we have two two-dimensional slices in two different coordinate systems, with the know border of the brain and a cross formed by characteristic plane intersection, on each of them (see Fig. 3.25).

In order to compare two slices, we need to find a convex hull for each of them (see section C.4.3). This is necessary to level influence of inequalities of the brain surface. Next, we find the middle (X) of the segment created by the intersection points of the YZ plane intersection line with the convex hull, and evaluate the perpendicular line

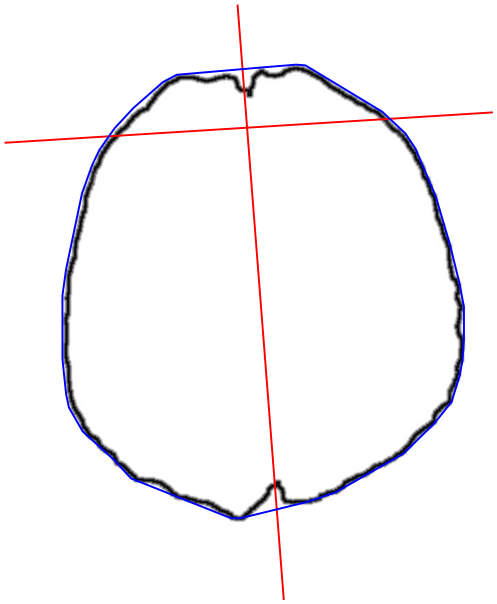

Figure 3.25: Brain borders on the intersection plane. The red cross marks the intersection of the slice plane with the XZ and YZ characteristic planes. The blue lines represent the convex hull over the brain border.

crossing through this point (see Fig. 3.26). This way we have divided both brain slices into four corresponding quarters. To make calculations more reliable they are further divided into n triangle-shaped sections (in Fig. 3.26 each quarter is divided into three sections). Division lines pass through point X and points on the quarter's convex hull border. Points on the quarter's border divide it into n segments of equal length ($\frac{1}{n}$ of the total quarter's border length). As a result, we get $4 * n$ triangle-shaped sections on both slices. Owing to the way we create them, they reflect differences in shape of both slices and each "triangle" on one slice has a corresponding one on the other.

Now let us define for each triangle-shaped section two vectors created by their sides (green dashed lines) and the common point X (see Fig. 3.26 and 3.27). These vectors, denoted in Fig. 3.27 as v_0, v_1 and v_0', v_1', form a local basis of our barycentric coordinate system described in Section C.1.5. Each point on the plane can be described as a combination of these vectors:

$$P = X + u * v_0 + v * v_1 \tag{3.34}$$

We will, however, limit ourselves to points within the angle created by the vectors discussed (that is $u \geq 0$ and $v \geq 0$). From now on, we will treat the model slice in the BCV coordinate systems as a standard discrete pixel map. For each point / pixel P on this slice, the algorithm checks in which section it is located (see section C.1.5)

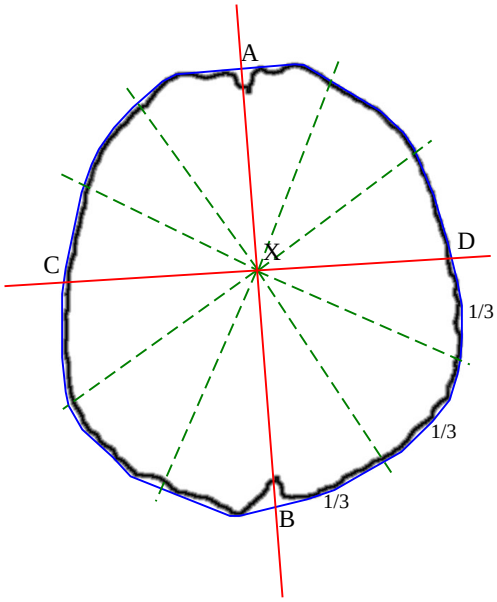

Figure 3.26: Brain borders on the intersection plane. The red cross marks the intersection of the slice plane with the XZ and YZ characteristic planes. The blue lines represent the convex hull over the brain border. The green doted lines mark sub-sectors on slice quarters.

and calculates its position as a function of values u and v (equation 3.34). A strong point of such a solution is that these values are normalised and describe proportions. Therefore, they can be used to calculate position of point P' in relation to point X' in a space v'_0, v'_1 of the corresponding section on the added slice.

$$P' = X' + u * v'_0 + v * v'_1 \tag{3.35}$$

Using parameter n we can modify the correctness of the transformation. The greater the n is the more triangles we get. The problem with few triangles is that information about the border of the brain may be outside the triangle created by v_0 and v_1 vectors (see Fig. 3.26); thus, information may by fitted badly to the target shape. We would have a perfect situation if all the significant image data were inside the triangle. On the other hand, high n value increases computation complexity and is not always needed as perfect accuracy is not necessary. As what we need to find the "golden mean". Our preliminary research showed that $n \in < 10, 15 >$ gives best results in acceptable time. In our further research, we will analyse it further as it may have implications in the case of not standard studies.

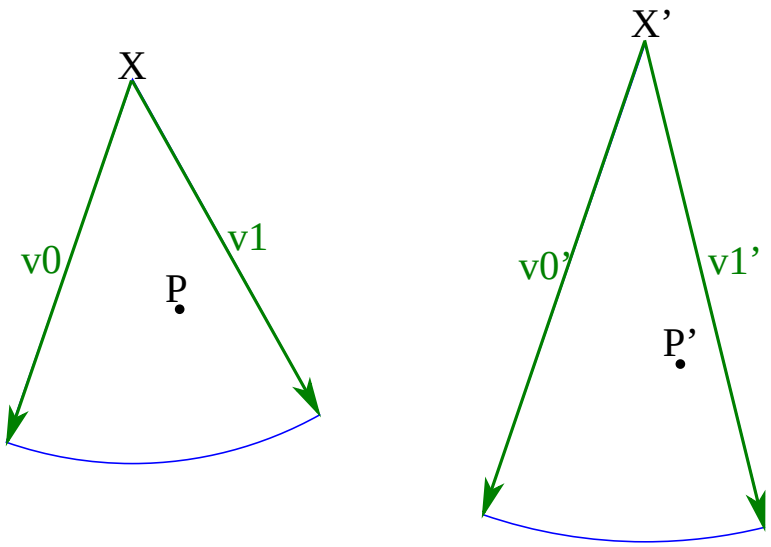

Figure 3.27: Two corresponding segments from two slices. Green vectors mark the local basis of coordinates used to describe positions of points P and P' in relation to points X and X' respectively.

3.5.5 Update the model

This time, however, point P' will not have discrete but only floating-point coordinates. Therefore, as described in Section 2.1, we must use interpolation techniques to calculate a value of a point based on values of its neighbors (see Fig. 2.1). The nearest neighbor approach is used again. In our further research, we will try using different methods to check their influence on the final result. Please remember, however, that we are dealing with complex data, and point P' is just a reference to get very detailed information about all hemodynamic parameters calculated for the study. Real interpolation methods would require complex calculation that might result in to mistakes.

The final step is to include new data in the $SPBM$. We know the position of point P in triangle section of the slice. From this, we can calculate its 3D position in the BCV and the scanner coordinates of the model.

3.5.6 Method verification

We have divided verification of our method into steps. By showing that each step included in the algorithm is correct and transitions between steps are accurate, we will verify the whole algorithm. Note that we want to show that all geometric operations are correct and, in fact, two completely different brains can be compared using this approach. Correctness will be analysed later.

The algorithm is divided into the following steps:

1. Localization of a study in a scanner coordinate system
2. Transform between two BCV coordinate systems and find corresponding 2D slices
3. Find corresponding points on two slices (shape and proportions maintained)

1. Localization of a study in a scanner coordinate system This method was tested in chapter 3.4 while the BCV algorithm was discussed. The method's accuracy can be estimated to be about 3% (or higher) of the data set size in the largest dimension.

2. Transform between two BCV coordinate systems and find corresponding 2D slices The correctness of this method is partly guaranteed by mathematic calculations, as this is a basic arithmetic and geometry vector. To test its accuracy, we have used similar methodology as in Section 3.4.4.

We have generated an artificial dens data set. Instead of a standard voxel data, in each point of three-dimensional space we have put information about the location of a current voxel in the scanner coordinate system. Like in Section 3.4.4, we have taken this data set and applied random transformation to it, generating twenty different (rotated and translated) data sets. We have used the same transformations as in the section mentioned. We may treat these data sets as studies that are to be added to the model - original set.

Now we assume we know the BCV and the BCP for each data set. With a given plane in a transformed study the methods in this algorithm's step should accurately provide us with a corresponding plane in the original one. Ideally we should receive 1 to 1 match when we compare points of these two planes (each one contains coordinates of the voxel in the original data set). Of course, this is not the case as we used interpolations, rounding in vector calculations, etc. The best we can do is to make the difference as negligible as possible.

For each case analysed we have taken four random (withing reasonable scope found in real life) planes (slices) to analyse. For each point on the plane found (in the original data set) we have calculated its geometric distance in 3D to the corresponding point in the data set that is to be added to the model. For all planes tested we have obtained the mean distance of 2.1 voxels, and a standard deviation of 0.9 voxels. The maximal difference we got was 2.5 voxels and the minimal 0.5. It is important to note that we have only taken into consideration significant points - within brain borders.

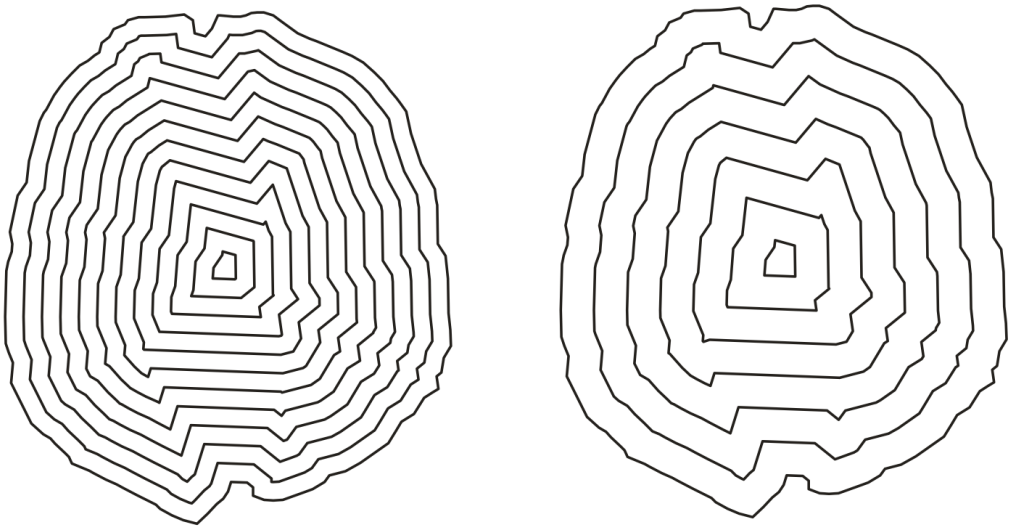

Figure 3.28: Sample test images for the 2D corresponding point searching algorithm.

This shows that, generally, the approach is correct but not perfect. We can get planes wrongly positioned, with a variance of up to 3 voxels (5 voxels worst). This is $< 5\%$ (of the whole size). Taking into consideration other approximations that must be included, this is rather small error and will not influence whole methodology or the $SPBM$. As mentioned, perfusion studies are not very accurate themselves and small rounding here will not affect the final performances of the model.

3. Find corresponding points on two slices This is a very important part of the algorithm as it performs shape-aware computations. Because it is done on a two-dimensional plane, the technique can be verified easier than in the 3D case. To check this part of the algorithm, we have generated artificial two-dimensional images of borders of the brain. The borders used were a real outline obtained in the previous step. Next, we have generated filling of that shape in form of decreasing inner contours, with different steps (see Fig. 3.28).

For five different borders we have generated inner contours, with two different steps. On the other side, 5 different borders were chosen to be filled with data from the first ten. For each image we have marked a cross needed by the method, using real coordinates from the previous steps. We have then launched our method for our ten data sets and morphed them into five borders, getting a total of fifty results. Thanks to the images we have prepared, it is easy to see if the algorithm works properly. After using it to fill given shapes, all the contours should remain closed and their mutual position should be similar to the one on the input image (see Fig. 3.29).

Figure 3.29: Sample test result of image morphed with the algorithm analysed.

Because it is difficult to analyse and interpret resulting images numerically, we have chosen to use experts. This is a very subjective approach but in this case it will be sufficient, as this is a high level context and shape analysis that can be effectively done only by humans. We have, therefore, asked four of our colleagues to look at the pictures and grade the quality of the morphing. Each image could be graded as: good (everything looks great and fitting is perfect or almost perfect), average (results are average, there are some mistakes but generally it is neither very good nor bad), bad (almost nothing is done well, why should one use this method?). Each description had a corresponding numeric value from 3 to 1 respectively. We chose such a narrow range of values to avoid a situation when the person questioned would hesitate, for example: is it 5 or 4? We have three clear steps that can accurately describe an image.

Each of the fifty resulting images was ranked by our four experts. The worst marks were given to images with small contours step that were fitted into small shapes (difference in diameters was more than $1 : 3$). They got average ranking of 2.2. In the case of similar shaped images, which is a realistic case that will be met in our method, or when images were enlarged, the mean rating was very high: 2.7. The fact that the lowest rating was 2 shows that our method proves to be quite correct and can be used to compare differently shaped brains.

3.5.7 Conclusions

Summing up, we have shown that the cross-patient brain comparison algorithm that we have developed is working. It produces accurate (withing acceptable limits) results, and can be used to compare information from corresponding voxels of two completely different MR brain studies. We have used the algorithm presented here to build the Statistical Perfusion Brain Model that can hold complex data about perfusion parameters (CTC, CBV, CBF, MTT, TTP). This model is a compilation of information gathered from any number of studies. Each new study is analysed and actually measured information about hemodynamic parameters enriches the model with new information. It is very important and unique that the model is updated with data from studies of completely different patients. The methods we have developed and tested make it possible to add this data in a shape and anatomy aware manner.

3.6 Results, summary and conclusions

It is time to show the final results of our research and how they can be used in practice. Our target was to propose the method that would help detect abnormalities in hemodynamic parameters measured with MR perfusion imaging. To reach our main and secondary goals, we have developed and implemented methods that could aid physicians in making diagnoses.

3.6.1 Perfusion parameter maps visualisation

First of all, we have enhanced classically used methods for calculation of hemodynamic parameters. Our interpolation based noise reduction method (section 3.2) has proved to significantly reduce artifacts, while at the same time important information is not lost. In the standard approach, image filtering - blurring is used to reduce the strong influence of noise on the final results. Such convolution operations work degradingly on the quality of already low-res image. Our technique removes only noise and other pixels that do not match the surroundings (brain tissues are rather homogeneous and all pixels standing out are artifacts). Additionally, it tries to find the value that should replace the wrong one, based on the information from the surrounding tissues.

Thanks to this improvement, we can provide physicians with more accurate visualisations and data about perfusion parameters in a certain area of the brain. Additionally, we have prepared special Look Up Tables (LUTs), used at display stage, that assist physician by emphasising certain values and levelling others. This, in turn, leads to easier and better diagnosis and can help detect ischemic changes in their earliest stages. Figures 3.31, 3.32, 3.33, 3.34 show sample images of perfusion parameters in different LUTs, generated by our software for study 3.30.

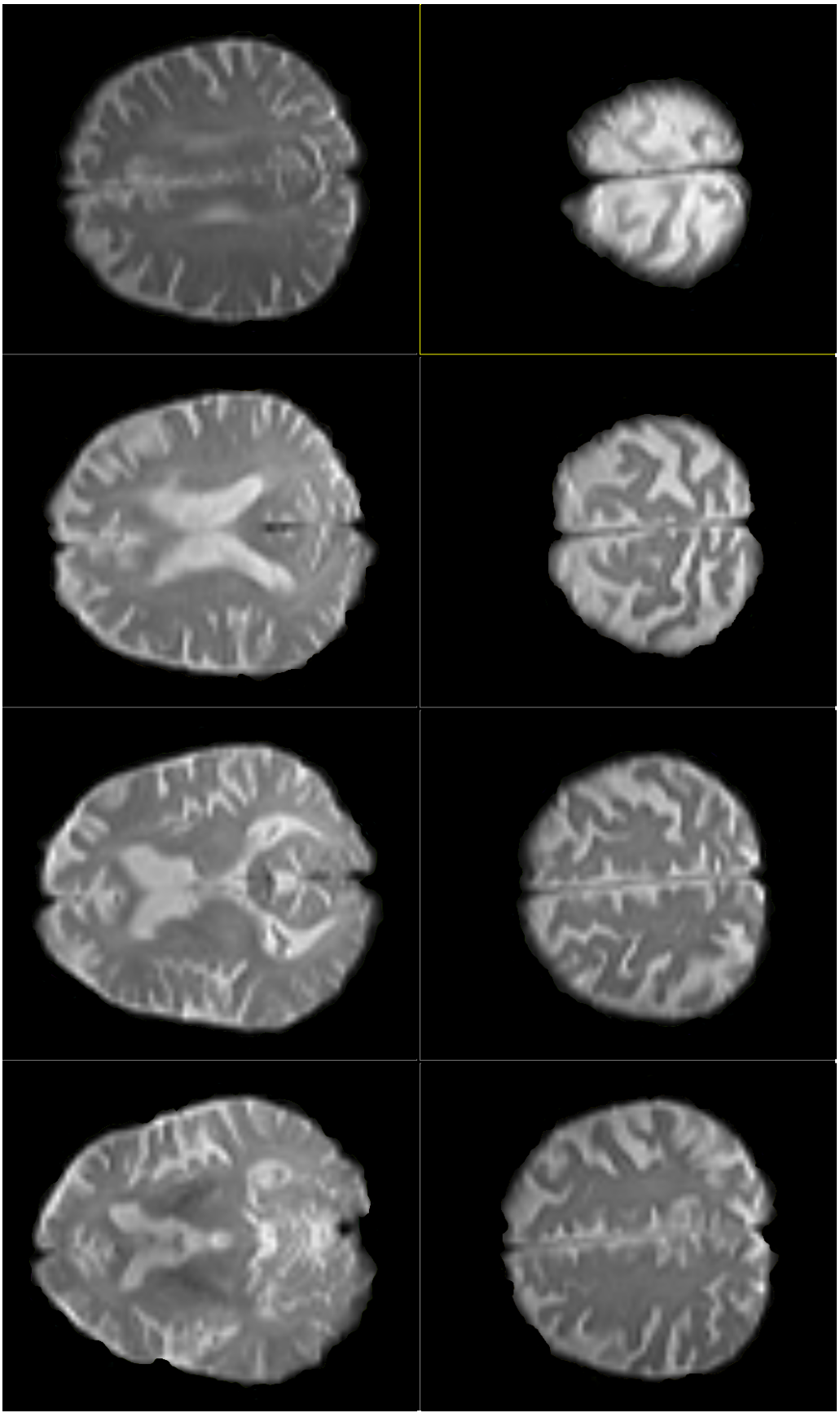

Figure 3.30: One of the passes of the sample study.

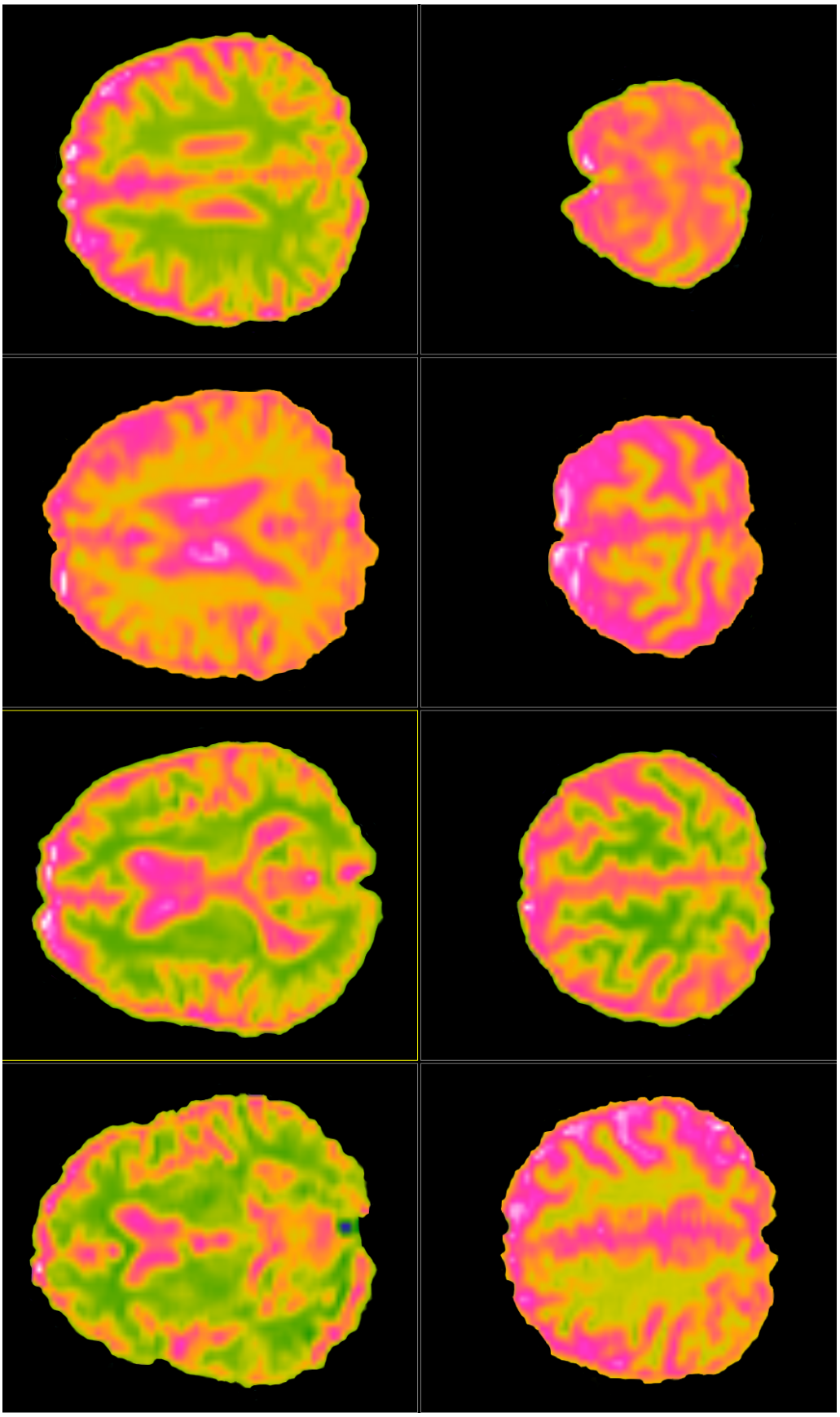

Figure 3.31: Sample visualizations of the CBV map.

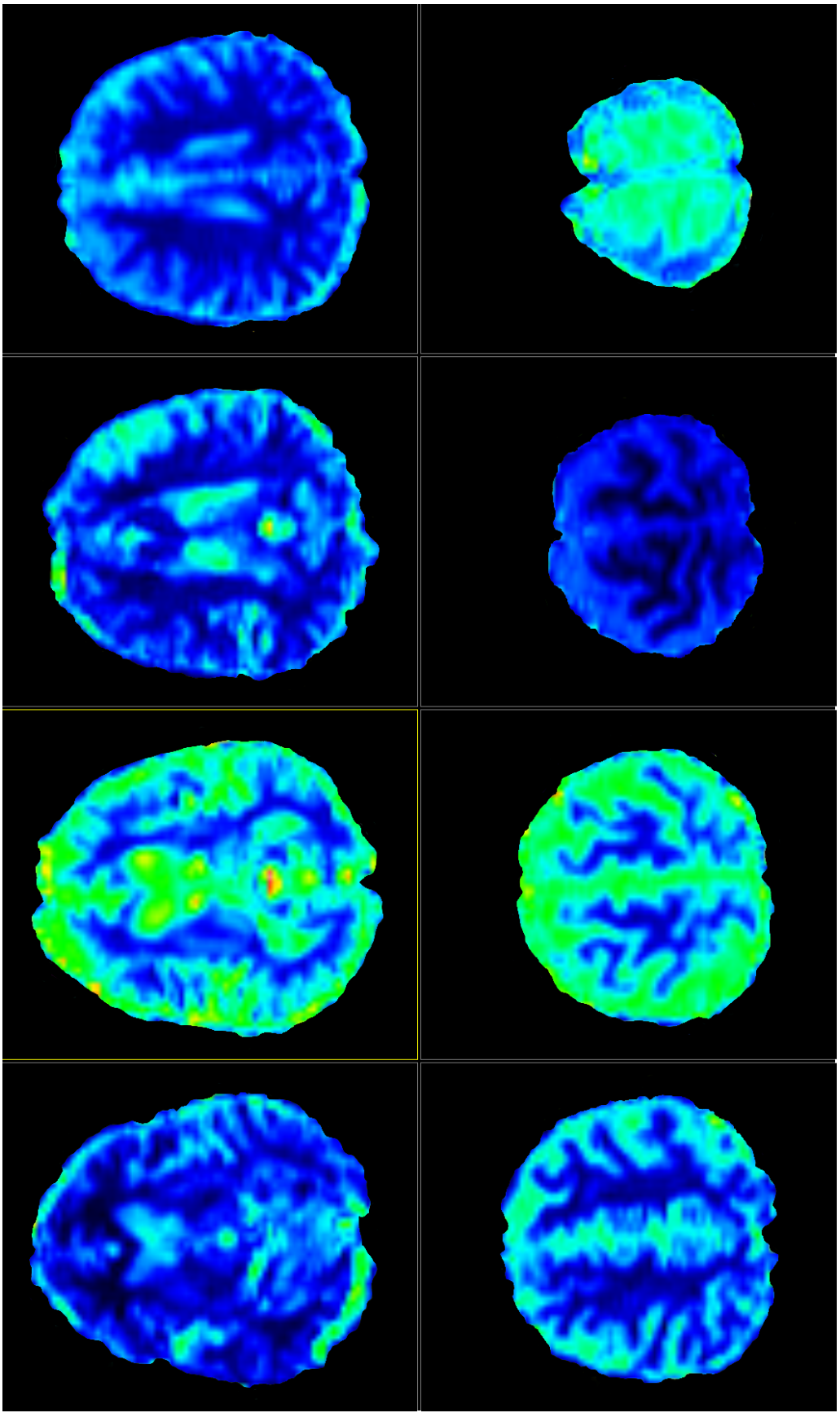

Figure 3.32: Sample visualizations of the CBF map.

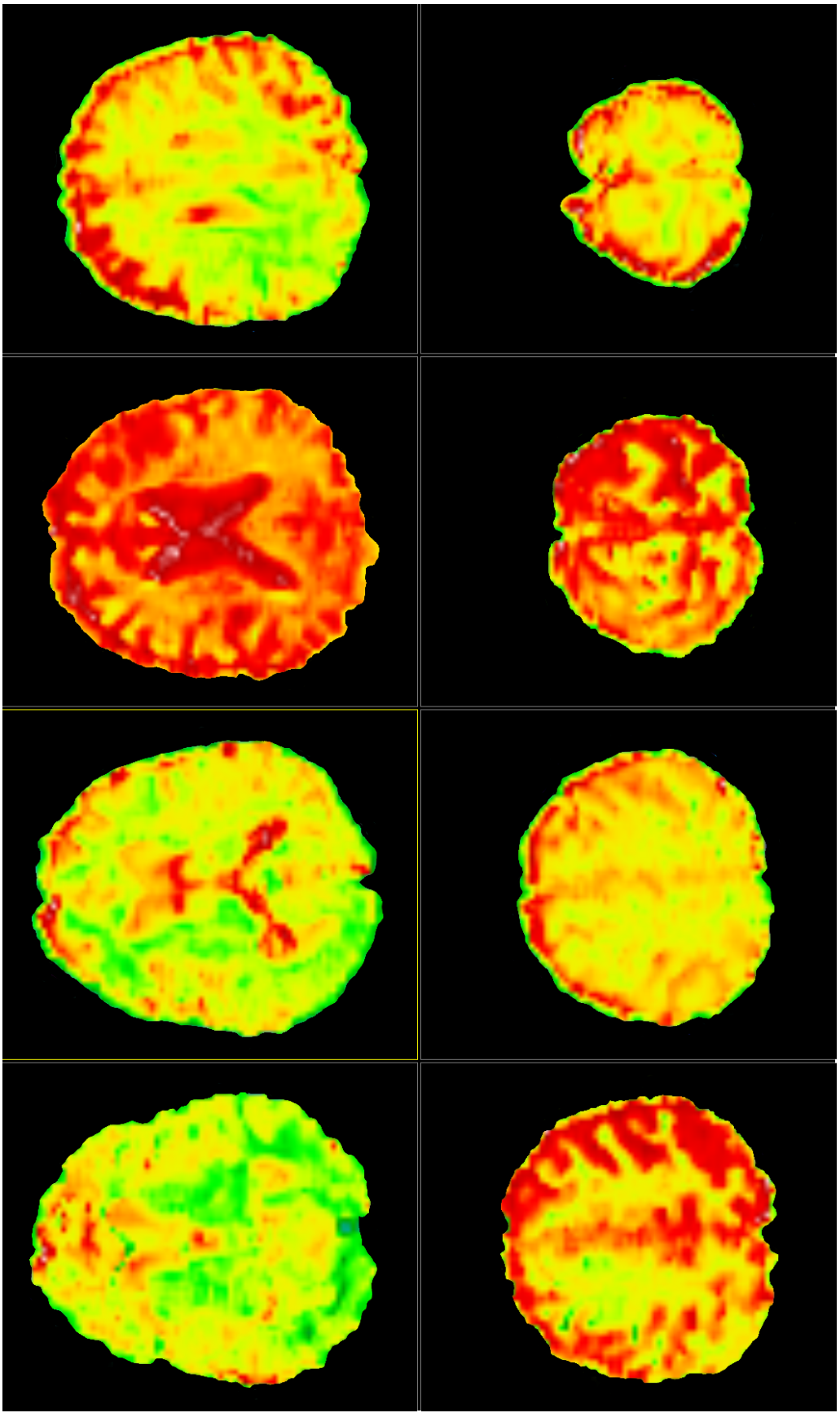

Figure 3.33: Sample visualizations of the MTT map.

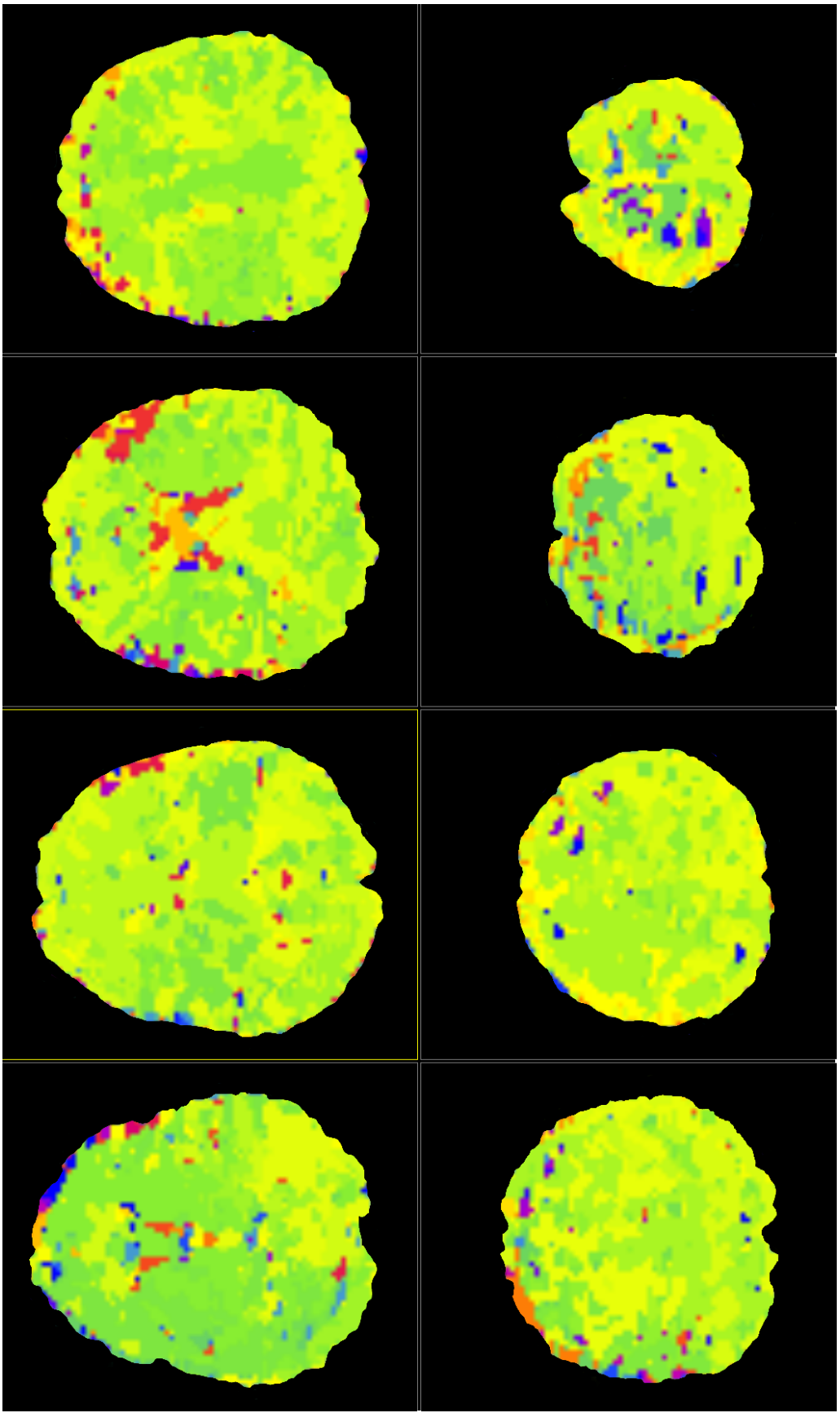

Figure 3.34: Sample visualizations of the TTP map.

3.6.2 Detection using symmetry information

The next technique we propose to use to improve the diagnostic process is to compare information from both cerebral hemispheres. The idea is that ischemic changes most often strike only one side of the brain. So if we compare both sides of the brain, we can detect abnormalities in hemodynamic parameters (see section 3.3). This method produces good results, although it is not fully automatic. It provides physicians with a tool that can pinpoint places of changes in perfusion parameter behavior. Preliminary tests look very promising as in most cases the algorithm helps to detect regions of possible change and notice spots where abnormalities are found only after detailed analysis. At the same time the algorithm does not yield many erroneous over-detected regions. In the opinion of physicians testing this tool, it is very useful and if skilfully used, significantly increases the diagnosis process.

Due to the limited availability of medical staff and studies from other clinics, our tool requires further testing. We will focus on confirming the results and conclusions from one source by using this method on a wider range of studies, acquired in other hospitals.

Sample results are shown in the Figure 3.35 (it is the same set of images as in the previous part of this section).

3.6.3 Detection using Statistical Perfusion Brain Model

In Section 3.5, we have tested our approach by comparing transformed artificial data with their originals. This shows the correctness of our algorithm and its importance from the point of view of computer science. To create the Statistical Perfusion Brain Model, apart from the IT tool, we need many real studies of healthy patients to fill the model with.

We had access to twenty five studies in total, of which nine could be treated as healthy and can populate the model. This is because it is not common to make perfusion studies on patients that are not suspected of a stroke or other ischemic disorder. The studies treated by us as healthy are, in fact, cases where no changes were detected, and patients were just lucky. To obtain more studies to fill the model, complex medical research is required. This, in turn, requires funds that must be obtained from grants. All this will take time so, unfortunately, we cannot present the final outcome here, as the medical team couldn't start working before we have finished the IT part of the project and done preliminary testing

We are aware that the number of studies we have used to fill the model is inadequate to make any final conclusions about its medical significance. For the time being we have provided physicians with a tool allowing them to check how the given part

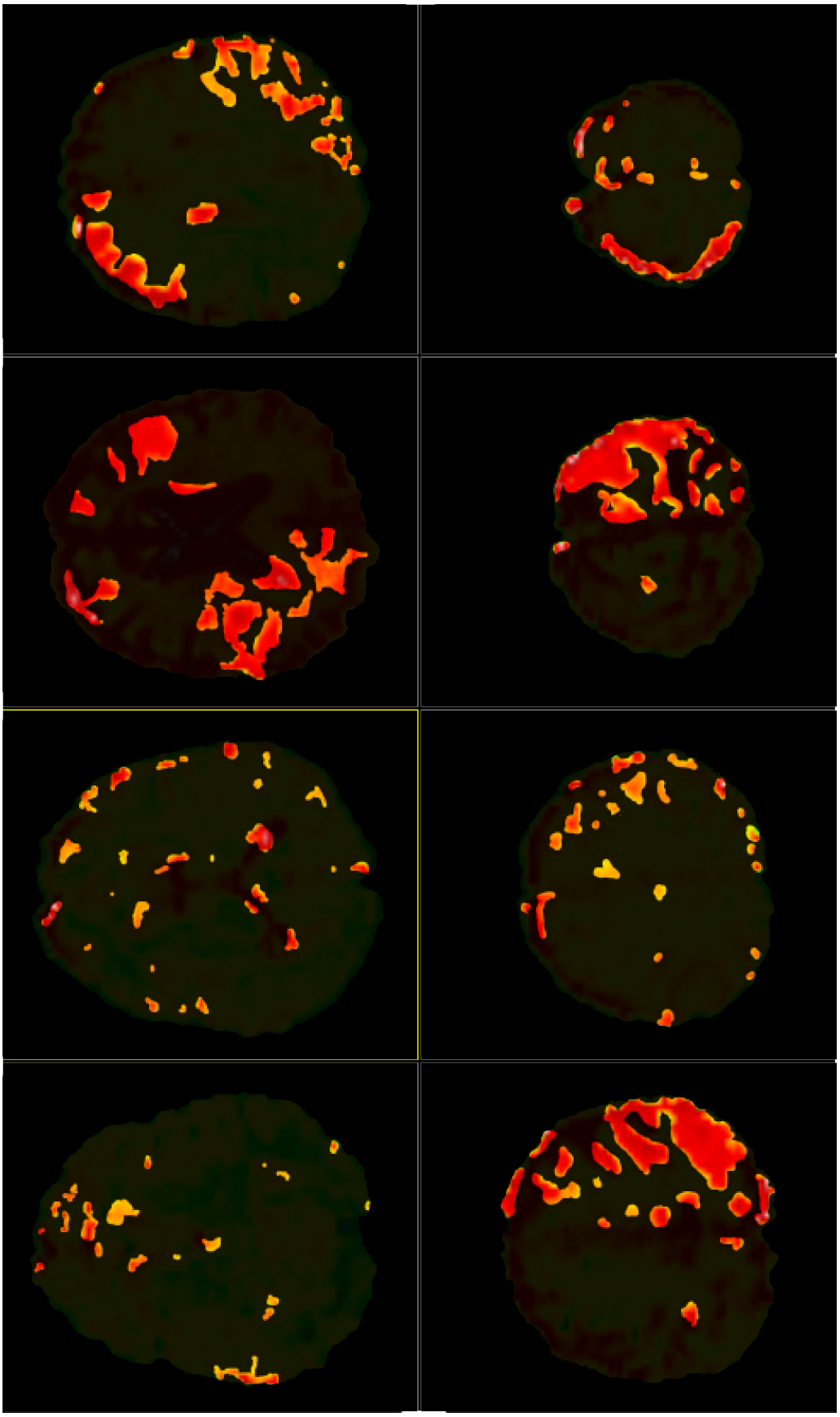

Figure 3.35: Sample visualisations of the MTT map, with regions detected by the method described in section 3.3 marked. Please note that not all slices are diagnostically significant.

of the brain is behaving while compared to the $SPBM$ (3.36). Software presents diagrams of distribution of each hemodynamic parameter at a given point or a ROI. We predict, and this also partly shown by preliminary tests, that distribution will take on the form of a Gauss curve. We will have many values around the mean and the more distant a given value is from the mean, the less often it is observed.

We have noticed that the $SPBM$ works very well with the symmetry based detection tool. Physicians are firstly limiting the suspected regions with the Symmetry Based Detection Tool ($SBDT$) and then use the $SPBM$ on a problematic region. Seeing this, we have prepared an extended tool that combines the $SBDT$ and $SPBM$. The user can define two thresholds, lower and upper. If a given perfusion parameter in a given point of the region detected by the $SBDT$ is below the lower threshold (percent of the mean value) or above the upper threshold, it is treated as abnormality and denoted as such (see Fig. 3.37).

Unfortunately, with so few studies, the model does not contain information about some parts of the brain. In other parts there is only data from one patient, etc. Therefore, our test limited to some studies, where perfusion slices, fitted mostly populated parts of the model. Preliminary results and opinions of doctors are very good and promising. We are, however, drawing mainly on their subjective judgement, as objective tests are not applicable here yet.

From the medical point of view, the $SPBM$ makes sense and, in fact, should work and be useful. There are some problems that will have to be solved as there are some differences in blood flow, between humans, near cerebral ventricles and brain surface. This can be solved by using different tolerance levels for these parts of a brain. Therefore, automatic, or at least less semi-automatic, detection of changes in blood circulations within the brain, it is possible to use our Statistical Perfusion Brain Model. Additional methods that will have to be developed to reach this stage are difficult to predict, as they must be preceded by long medical research. We are planing to test high level computational techniques that should be useful here [87, 88].

3.6.4 Conclusions

This work was devoted to four dimensional analysis of perfusion in the brain. In our research, we have used MR perfusion studies acquired in the clinic hospital we are cooperating with. Perfusion studies are done after administration of a contrast agent. This shows how the blood flows through the volume of the brain. Therefore, we deal with the true four dimensional analysis.

The main goal of our research was to develop a new effective methodology for automatic (or semi-automatic) detection of the acute cerebral ischemia in a MR

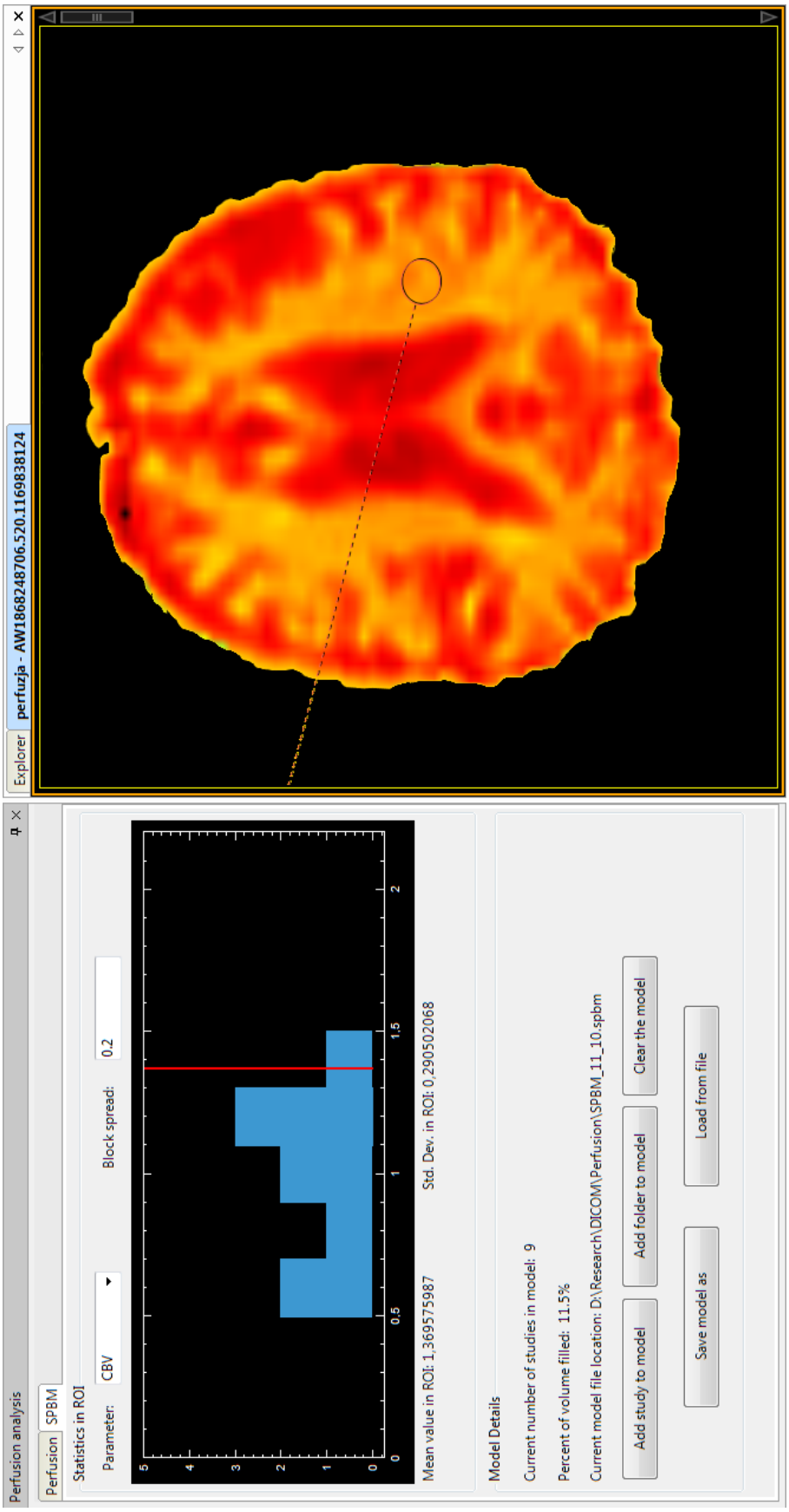

Figure 3.36: Diagram provided by our software, comparing perfusion parameter from the tested study with the SPBM

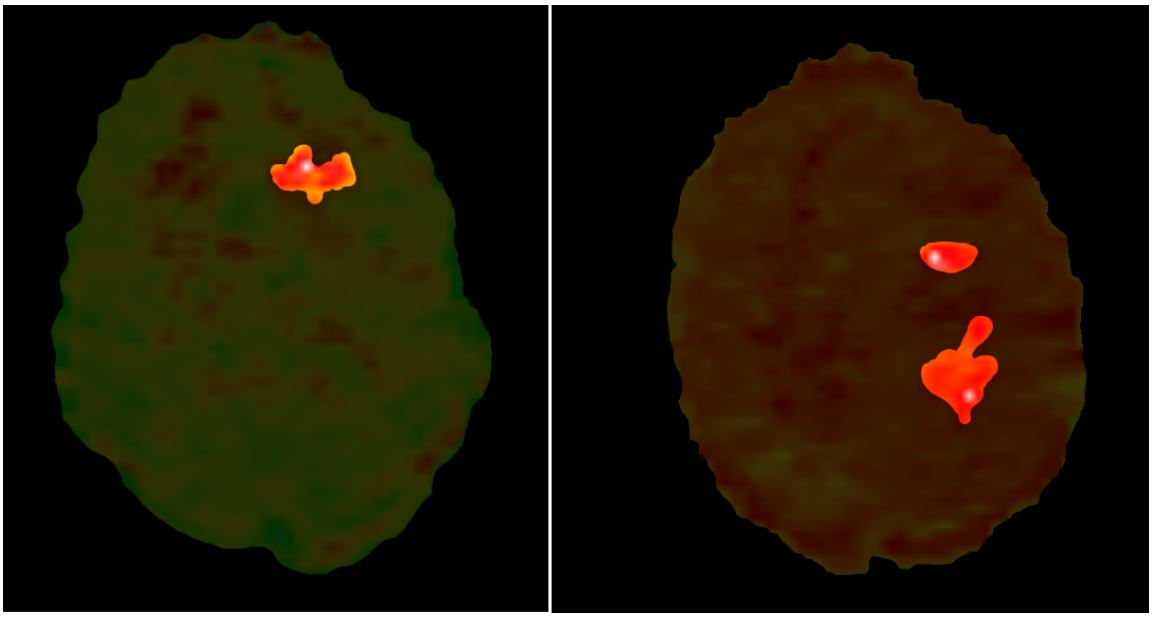

Figure 3.37: Sample slices with regions denoted as abnormal by a combined use of the SBDT and SPBM

perfusion study. We have proposed two semi-automatic detection and assistant tools. Our Statistical Perfusion Brain Model has a potential to become an automatic way to detect abnormalities. Additionally, we have improved the correctness of the currently used algorithm for calculation of hemodynamic parameters and prepared special color pallets for better visualisation of perfusion parameters maps.

All the tools were verified and preliminarily tested. The results are very promising, but because this is an important issue - human health and life - they must be deeply tested from the medical point of view. We have provided physicians with effective and powerful IT tools, now it's their turn to deeply evaluate them. We will, of course, work closely together with medical doctors to improve our methods and develop new ones.

While developing the Statistical Perfusion Brain Model, we have designed several algorithms that can be used to accurately and automatically compare two completely different brains in a way that takes into consideration shape and anatomical structure of the human head. This was our secondary objective and the main issue, from the medical image processing point of view. We have objectively and subjectively tested these methods showing the correctness of our approach, even when dealing with low-res perfusion images.

During our research, we have also developed many innovative algorithms to aid us in reaching our goal and making results more accurate. For example, in chapter 2, we have presented several different segmentation approaches. These algorithms are very interesting, presenting different complexity and producing good results. We have tested all the methods presented, for use with the $SPBM$ and $SBDT$. Which one should be used depends on the situation and the correctness we require at the time.

Whenever we presented a new algorithm, it was supported by test results and validation. Generally, it was our approach throughout this thesis as we present everything in the bottom-up order. First we present a method, test and discuss it and then we show how we put all this together to get a more complex algorithm. Sometimes, thanks to such an approach, we can show that a larger method is valid, because "bricks" making it up work very well.

It is important to note that many results of our work in this research were published, after reviews, in national and international, highly regarded, journals ([6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]). The results of recent research await publication.

Implementing all the techniques presented here, we have shown that it is possible to develop a non-expensive commercial-quality MR perfusion analysis application with very limited resources. Using only a single-person development team, we were able to design and implement software that can compete with commercial applications. It has same functionality and some tools that are even more complex and innovative. We show some functions and sample features in chapter A. It can have a considerable impact on the availability of the software and the popularisation of MR perfusion examination. In turn that will directly translate into the safety and well being of patients.

I never see what has been done;
I only see what remains to be done.

- Marie Curie

Appendix A

DICOM Workstation

In this appendix we present universal and very functional medical diagnostic application architecture. It was developed based on years of my experience: scientific research in Information Technology Department UMCS and practical knowledge gained while working for company developing professional solutions for radiology (PACS, RIS, Dicom workstations, etc.). I have also used the most effective development techniques. The project has been realised as a fully functional workstation, which finds many applications in daily work of radiologists and developers working on medical images processing.

Application architecture is based on the MVC pattern. It presents a successful compromise between effectiveness and scalability (plugin structure). Adding new function is rather straightforward and does not require a deep knowledge of the programming language, while providing wide control over applications framework functions. Thanks to a well thought-out application architecture and realising it on the .Net platform, extensions can be written under various programming languages (C#, C++, VB or Java).

Thanks to such an approach, I have drastically lowered the enter barrier of the project. Even young and inexperienced programmers can start developing new plugins to the application, after a short familiarisation with its structure.

Due to the size limit and the size of the application alone, in the chapter, we have decided to present only its most important features. Apart from a rather short and general architecture description, we also analyse it - we check if enter barrier was in fact lowered. We present methodology for researching this subject and initial results and conclusions.

A.1 Architecture

The main reason why many computer science projects fail or develop too slowly is, inter alia, a high participation barrier. This may be caused by bad design, lack of documentation, or before actual participation one must become acquainted with a large part of the program codes. The architecture proposed allows:

- simple joining of the project

- simple adding of common functionality to the application, while more complex customisation is also possible
- participation of programmers at different levels of skills

Easiness of participation is very important. The most demanding moment for the programmer who wants to participate in a project is to gain practical knowledge of the program code. To avoid this, we have chosen a plugin architecture. It means that new functionality may be added during runtime, as separate binary modules, which do not disturb the main, base code. A good example are such applications as the Firefox internet browser, or the integrated programming environment Eclipse for Java.

Easiness of use and flexibility. The simple API and object model are good, because they are simple to use. On the other hand, if we want to do something unusual, which was not predicted by the API's authors, it may be very difficult, if not impossible for less experienced programmers, due to the number of abstractions and indirections used. Most reasonable solution that we've used is to combine a rather flexible API and hiding most of the complexity behind easy to use interface and code templates [89, 90].

Adjusting to programmer's skill. Not everyone has experience in programming in the same language. Some people prefer C++ or C# typed languages, others Java or VB. From the project management point of view, it is not vise to narrow down to one programming language. Therefore, we have chosen Microsoft .NET environment, which allows programmers to write plugins in a wide range of languages provided by the .NET framework [91, 92].

Extensibility. To ensure best flexibility, plugin architecture is designed so that everything is a plugin, and plugins may extend plugins. The only executable is the boot loader, which launches first plugin.

Model/view separation. The kernel of the application was designed using Model/View/Controller pattern (MVC) [93]. It means that domain object (data Model), objects representation on the User Interface (View), and object's manipulating domain objects (Controller - system logic) are separate. This approach, along with plugin structure ensures modular software development. Separating the View from the Model is very crucial, because presentation layers are still changing. For example, Windows Vista introduces WPF (Windows Presentation Foundation). It is to supersede what we know as Windows Forms. If Model and View where too close together, adding support for the WPF would be very difficult. Therefore, in our solution, model and view are implemented as separate plugins; thus, adding support for the WPF comes down to writing a plugin.

Plugin / extension. [94] As we have mentioned, functionality is added to the platform through plugins, which allows to extend application's possibilities infinitely. Thanks to extension points and extensions, the application knows what to do with a code in the plugin. In general, a plugin is only useful if the existing plugin knows what to do with it.

Plugin A → Plugin B → Plugin C

The above shows that *pluginA* knows how to use *pluginB*, which, in turn, knows what to do with *pluginC*. We say that *pluginB* extends *A*, and *pluginC* extends *pluginB*. However, for the plugin to be extended, it has to have a mechanism to declare how it may be extended. It means that it has to declare one or more extension points. An extension point can be extended by many plugins, which can extend one or more extension points of one plugin.

From the program's code point of view, the extension process is achieved by using interface implementation mechanism. It means that an extension point will always be described by interface, which the extension must implement. An extension is a class that implements a given interface.

At the application's startup, the framework loads plugins and searches them for defined extension points and extensions.

At the same time, the main application's plugin asks the framework for all extensions of the *DesktopToolExtensionPoint* extension point (this extension point allows to add new elements to the UI). The framework then tries to create an instance of each extension of this extension point, in the current plugin set, and return the object to the Model. Because each extension implements interface *ITool*, Model knows how to cooperate with the tool extension and how to integrate it in the application.

ImageViewerToolExtensionPoint is similar to *DesktopToolExtensionPoint*. Both define *ITool* interface. *ImageViewerToolExtensionPoint* allows one to define image viewer tools, which can be used in the DICOM viewer, for example a stack tool - for navigating through data set, or image magnification tool.

A different tool can extend different extension points. They must always implement *ITool* interface directly or indirectly (through class which derives from *ITool*). An extension class must extend a proper extension point in order for the host plugin (Model) to know how to treat it.

Application component. [95] Application component is a functionality which adds a new user interface element to the desktop. It consists of the component (implementation of the *IApplicationComponent*) and corresponding view (implementation of the *IApplicationComponentView*). The main function of the component is encapsulation of the logic and acting as a presentation model for the view. Because the application component plays a role of the model in the model-view paradigm, the view must be thin and unintelligent.

Application component is executed within a host (implementation of the *IApplicationComponentHost*). The host provides a component with an interface to the environment in which it is executed. It is important to note that the view knows about the component, but the component does not know about the view. On the other hand, the relation between host and component is bidirectional.

Actions and actions model. *IAction* interface represents a single, atomic action that can be taken by the user. For example, *IClickAction* - a single mouse click, or *IDropDownAction*.

In the user interface, menu elements or buttons on the toolbar are organised in a hierarchical structure. The menu contains a sub-menu, which can also contain a sub-menu. The hierarchical menu and button structure can be modelled as a tree. It is the most obvious and legible solution. Likewise, an action model is simply the *ActionModelNode* object tree which stores references to the *IAction* objects. Only tree leaves contain such references.

One of the ways to describe an action model is to describe a path through a tree for each *IAction* object. If a path is associated with an action, then for each action set, a corresponding model can be constructed. Therefore, *IAction* has a property Path, which describes action position in the action model.

For the tool to be useful, the user must be able to interact with it. Therefore, tools deliver action sets, which are integrated with the User Interface. *ITool* provides property Actions, which return a set of *IAction* objects, linked with methods of tool objects. Tools are the basic way to add functionality to the framework.

After a tool extension is defined, application component must be created, during runtime, instance of each class extending this extension point. ToolSet class delivers comfortable mechanism for this. Component simply initialises the ToolSet object with an extension point and reference to the concrete tool. The ToolSet class creates instances of each tool and initialises it with a concrete context.

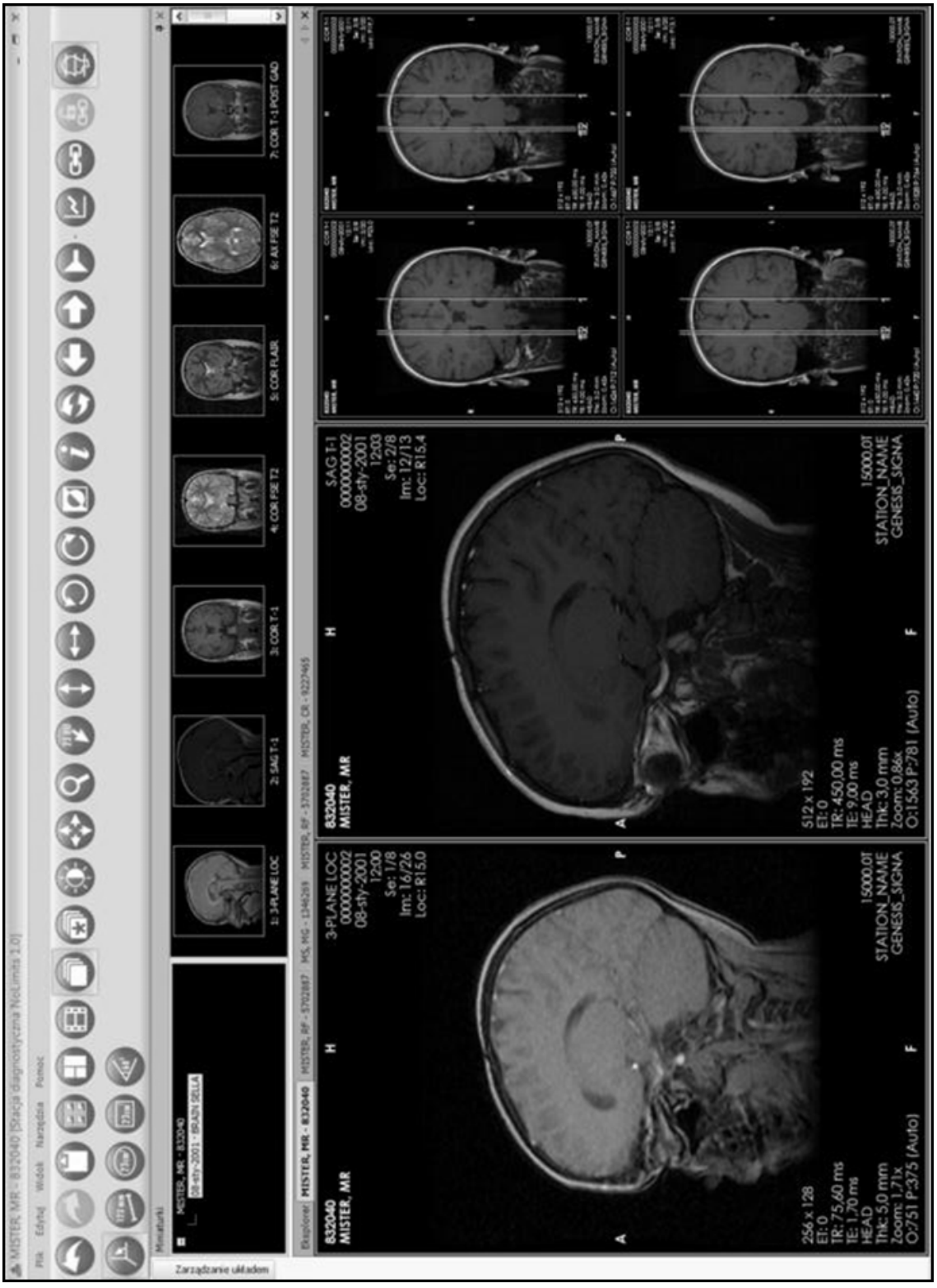

Figure A.1: Screenshot of the application.

A.1.1 Discussion

To sum up, our goal was to develop application (DICOM viewer - see Fig. A.1) architecture in such a way that the participation barrier is very low. We want to gather as many eager students as possible. We do not want to set participation requirements too high as it will cause the project to die natural death, because there will be no one to develop it. Each participant should be able to start work almost instantly. It is important to mention that the participants do not and will not work together. In fact, the only common thing is developing application - its framework and target. Each programmer is working on his own module, his own part, and this work is not coordinated with others. Of course participants can use modules that have been developed earlier, but their main particular target is to achieve their goal (in most cases it is to explore the master's thesis subject).

Theoretically speaking, taking into consideration application architecture, and its possibilities, the goal was achieved. But does practice bear it out? We would like to present a method for checking and measuring real practical flexibility and easiness of the application extension.

The reduction of differences between work efficiency is the first indicator showing if the application: is simple and flexible at the same time and allows for participation of programmers writing in different languages, as well as different level and set of skills, having different knowledge about the application code and architecture. What we mean is that if the application is, in fact, simple and not demanding (in terms of adding new functionality), then the effectiveness of a programmer knowing the application, and the one just starting the work shouldn't differ much. Additionally, a newbie should be able to quickly catch up in terms of effectiveness with people working longer with the application. Under the term *effectiveness* we understand relation between the effects of the work, that is, ready modules fragments (project + implementation), and the time spent on their creating. In other words, we want to check how big is the difference in the time spent on realisation of similar programming tasks, between inexperienced and experienced participants, and how fast potential differences are nullified. (We do not research into code effectiveness, or optimization. Test subjects are not very complicated; therefore, at the present state of technology there is no sense in deliberating tenth parts of the second). The problem formulated in this way can be easily and objectively measured.

An introduction to the analysis of the above indicator is studying the level of knowledge and skills of the persons entering the project. For this purpose, standard tests, conversations, and study course analysis are sufficient. Next, we consecutively analyse (through tests) conversations, focus groups and work effects analysis, improvements of the project participants. This gives us an idea on how fast they learn the application architecture and programming language, and what effects they

achieve. Also, it should give us some feedback about the effectiveness of our training process.

The second indicator, no less important, though subjective, is the analysis of participants‘ "feeling". Do they like the application architecture? Is the work with it simple and clear? Does it make it possible to realise all the purposes intended? And, most importantly, is the work not tiring and not exhausting? These are only a few questions that we ask programmers to answer. Of course, we should treat those questions as a sort of general indicators, which should be tested with a suitable set of questions, made into a survey. The survey should also contain open questions, allowing respondents to state what they think should be changed, or what changes should be made to the application architecture. Such propositions can then be analysed in focus groups, and if the team would decide that they are right, they should be implemented.

Appropriate tools (surveys, tests, forms) were developed and introduced, according to the methodology presented. We have used them on the first group of the project's participants. Because the project has just been launched, the results that we present are not formally competent - the sample is too small. However, they show a tendency and in the authors‘ feelings this should not change dramatically. We must also take into consideration the fact that it is just the beginning and we are all still learning.

At this moment, we have five participating programmers. Three of them are using C++ language, two C#. For clarity, we have described their programming skills with a three-grade scale (basic, intermediate, advanced). There is no need for a more sensitive scale, because it is not so important. Getting back to results, two C++ programmers and one C# programmer were at the intermediate level, one C# programmer was at the advanced level and one C++ programmer at the basic level.

In short, the effectiveness analysis (participants were presented with the tasks at different difficulty levels) showed that the difference between intermediate and advanced programmers was not as big as between beginner and intermediate ones. It is probably due to the fact that the application requires from the programmer a certain "set of knowledge and skill", and persons at the intermediate and advanced levels had mastered most of those attributes, whereas the beginners hadn't. But, again, because the sample is too small, we cannot find certain conclusions. Interestingly, the same analysis showed that after approximately 32h of work all participants, have reached a similar efficiency level. In other words, they became acquainted with the application and learnt its architecture, and didn't have to use the documentation. At this moment, the only advantage of more experienced programmers is that they write a more optimal code, but the level of skill does not significantly affect the production speed.

Considering the opinions and feelings of the participants reveals that they like this architecture, which is legible and easy to work with. Due to the level of the project's advancement, it is hard, however, to estimate the usefulness of the framework for creating more complex algorithms.

A.1.2 Conclusion

In this section, we have presented our proposition for a very scalable application (DICOM viewer) architecture. The description was rather short and presents only the highlights. The project has a very low participation barrier, and can unite programmers writing in different languages and having different skill levels. This is a very important feature, because it extends the possible range of project's participants, and does not limit them only to a few very good and patient programmers.

We have also presented a methodology that allows one to estimate in objective and subjective manner the quality of architecture, its usefulness and the effectiveness of working with it. We have also presented first results of our research with this methodology. They show that, in fact, the architecture proposed is very flexible and one can almost instantly start working on developing it.

In the future, we would like to develop a training program for new participants, which would quickly get them to work. Currently, we are delivering training, but it is not what we want. Based on research results of the persons joining our project, training needs should be assessed, and according to this analysis the best form and training subject must be chosen. Training itself must be targeted at filling certain "gaps". It is obvious that we will use elements of e-learning, creating a knowledge base in a form similar to Wikipedia. Next, training evaluation must be done. It is only a matter of time, because after a few such "cycles" (need analysis, training planning, training, evaluation) a training standard, suitable for each new participant, should be elaborated. Thanks to this, the current time of 32h needed for a full introduction into the project can be reduced to, say 8h.

A.2 Sample tool - Teleradiology consulting system

In this section we would like to present a tool, recently developed by us. Its purpose is to allow a physician to tele-consult a radiological study with another specialist. The main advantage of this solution, apart from providing secure image transfer and real time work of two users with the same data set, is that it is only a module of the DICOM viewer (diagnostic application). There is no need to use additional programs or web services. The physician works with the same application he or she uses every day. Whenever necessary, he or she can connect to the remote consultant.

A.2.1 Background

The recent development and dissemination of information and communication technologies have contributed to the creation of a global infrastructure for wider use of information sharing for medical needs. The use of computer networks to transfer information and images allows for fast data exchange between medical staff.

An inevitable part of a diagnostic and therapeutic process is the exchange of information between health units at different levels of reference. Consultation carried out in a traditional way involves the transport of the patient and medical records between health care institutions. Implementation of the consultation through a network that ensures the quality of the media can accelerate this process. This is particularly important when the image is the primary diagnostic material.

Before we get to discuss our solution we would like to emphasise that the target group are radiologists, doctors of 'first contact'. They evaluate several studies per day and their time and accuracy of diagnosis is most important. They will be consulting doctors at a higher or equal reference level, working in the medical centres in different parts of the country.

A.2.2 Objectives

It is therefore reasonable to develop and implement a solution that would allow for remote consultations of digital radiological studies. Before work starts, one must realise the specificity of the work of doctors and radiologists and analyse the process of "production" of the study description. A description - correct diagnosis is the most important element both for the patient and the physician. Thanks to it, the former has a chance of recovery, and the latter fulfils his mission.

Therefore, we have defined the two most important actors in our system, although only one - the doctor - is to use it actively. Let us see what the typical diagnostic process looks like (see Fig. A.2). At the same time, let us suppose that the diagnostic tool is computed tomography.

We ignore (at this juncture) the whole process of registering patients and their way to the laboratory, because it is not important from our point of view. Depending on the degree of computerisation of the hospital, pictures from the digital device (CT) go to the hospital PACS archive, from where they are collected by the workstation; or directly to the workstation (or simultaneously to both places). Explanation: PACS archive is a database that stores not only image information but also patient data (according to the DICOM standard, see Appendix 1).

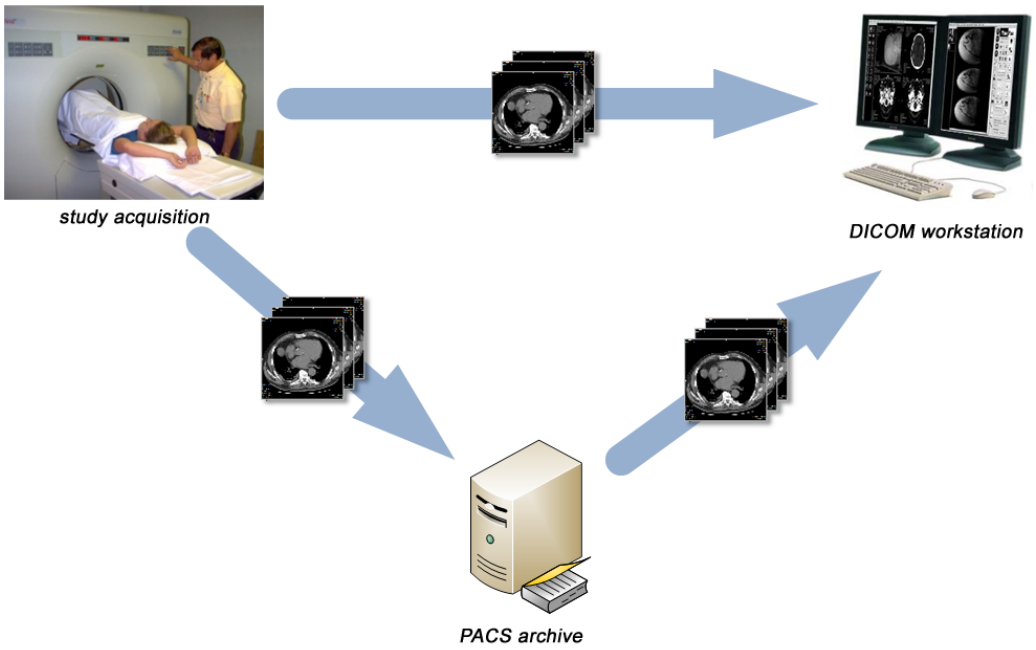

Figure A.2: Study acquisition and description workflow.

From the outside a diagnostic workstation, is a computer with high-quality diagnostic monitors. But more important is what is inside - specialised software that makes it possible to view and analyse images generated by diagnostic equipment. This is the place where the radiologist spends most of this time. Using his own knowledge and experience, he attempts to detect any anomalies, things that are not right in the study images. Of course, he also has access to patient records through the hospital system (digital or "bureaucratic").

This is where doubts could arise. It is logical and understable that various medical centres employ specialists of different classes. They are divided into the so-called reference levels. If the doctor on a given level of reference is not sure of the diagnosis - cannot cope with a case - he must ask for a medical opinion at a higher level of reference. The person at the highest level, cannot do this and must make a diagnosis alone.

Using this example, it can be clearly seen that in the case of a doubt at the time of diagnosis, teleconsulting could be great of help. The solution presented by us is to be used exactly in this place. Its purpose is to provide a platform enabling doctors to work on one data set in various places around the world (as if they were sitting at one diagnostic monitor). They can use the same tools and see the results of activities of everyone participating in the consultation.

A.2.3 Assumptions

During the analytical work, we came to the conclusion that the best way to introduce an effective teleconsulting platform is to use the existing DICOM image viewer (see Fig. A.3) and add a module allowing remote consultations.

Why? The reason is very simple. During the diagnostic work the physician often relies on tools [96, 97] such as measurement of length, area, angle, level / window control, MPI, MPR, or even VR (volume rendering). Adding such tools to the teleconsulting platform developed earlier would be a very costly and long process, and "forgetting" about some tools may be disadvantageous to the solution - it would not be used. It is much easier and faster to develop the extension of the existing diagnostic applications. For such a "base" we have chosen the being developed by the one of the authors of this article.

Another assumption / conclusion flowing from discussions with medics is that the consultations do not need to be carried out between more than two physicians (peer-to-peer scheme). The reason is simple - cost. In any situation where the doctor needs a second opinion from another centre, there arises the question: who will pay for it? Perhaps the consultations will be held under a contract between units or between doctors. Anyway, this is a real problem and the conclusion is obvious: it is easier and cheaper to consult one doctor already checked than, for example, two ones (a double cost). Of course, there could be a situation when you need to consult another expert, but you can then repeat the session. Such situations are quite rare, doctors are usually familiar with their skills and know who to turn to with a problem.

These assumptions, about the structure and nature of the application are subjective conclusions of its authors. However, there are assumptions that must always be met, regardless of the platform.

First of all : security. We must remember that the patient data, that will be sent is protected by the law. At the same time physicians must be confident that the data with which they work has not been distorted (intentionally or accidentally).

Please also remember the legal issues - diagnosis can only be made on the basis of the DICOM quality images.

Second : speed. Consultation must take place in real time, with delays being negligible.

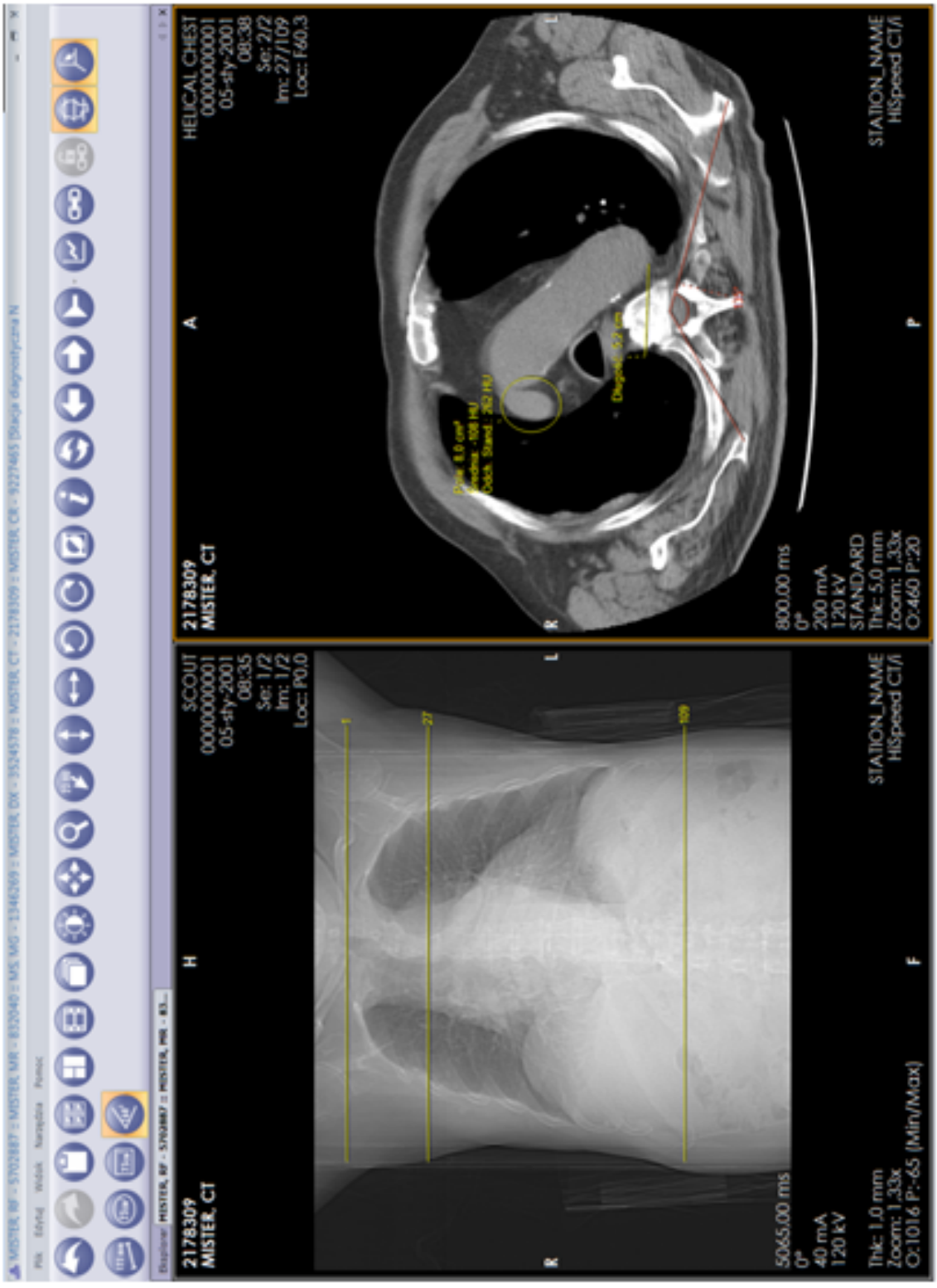

Figure A.3: Screenshot of our application.

A.2.4 Use cases

The first step (after analysis) to begin work on any computer system is to define use cases, that is to answer the question who and what for will use the application. The actor in our system, as mentioned, is the radiologist who:

- can connect to any other user of the application (must know the IP address and the password to the partner's application)
- can communicate with the partner through a voice conversation or a text chat (for example, Skype)
- can also watch his interlocutor through the webcam

For the activity to be successful, other conditions must also be met:

- radiologists can work on the analysis of the same study, by performing all the operations provided by the modern diagnostic software
- results of operations appear simultaneously on both participants workstation windows
- possibility to transfer studies
- in addition it is possible to securely transfer files between partners (for example, scans of the order or patient chart).

A.2.5 Implementation

Now we will discuss, in general, how we have realised each functionality (see Fig. A.4).

As mentioned, we wanted to avoid communication through a server, and only used the peer-to-peer network architecture. It is therefore necessary to have external IP addresses of both sides of communication. In the absence of such an address and connecting through the gateway, it may be necessary to define the tunnel to a specific computer (and port) in the subnet. The parties will exchange their addresses to be able to connect. Of course, the user is informed of an incoming call and can accept it or not.

We used a sound library [98] to capture and send audio over the network, and streaming classes to transfer an image captured from the camera [99], and text chat [100]. This part of the tool is relatively simple and quick to implement as it is well described and often used.

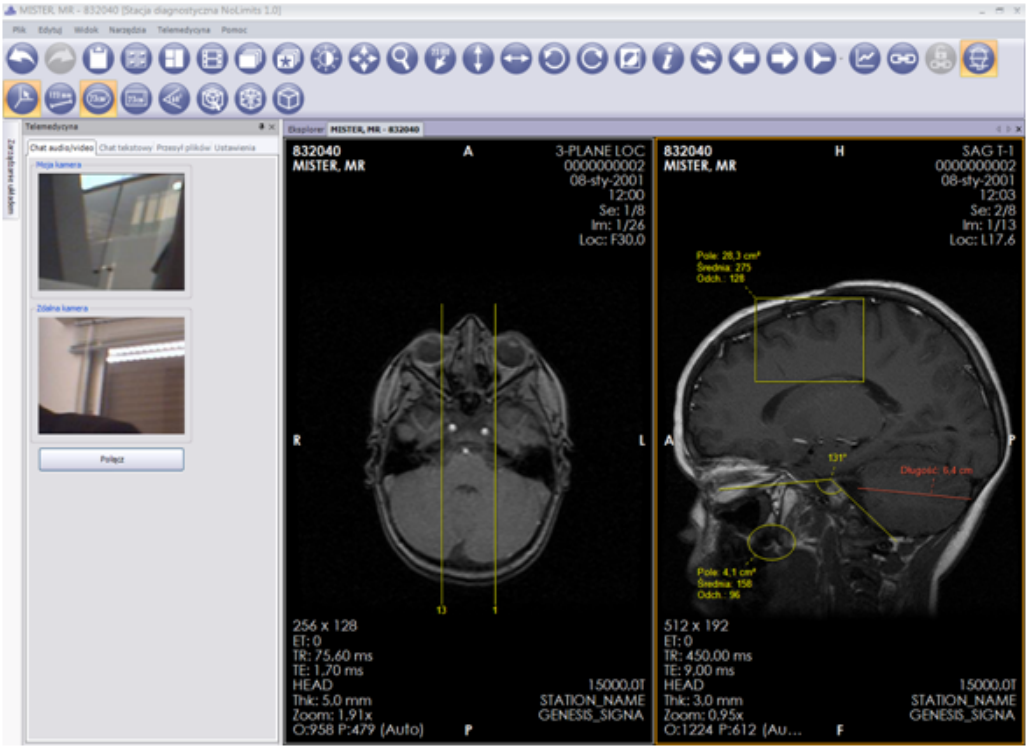

Figure A.4: Application with an activated teleconsulting tool.

What about a completely different work on a common data set? The problem can be divided into two separate sub-problems. The first is the transfer of the image in the sense of a DICOM file; the second is the transfer of any information about transformation and subsequent layers over the image (all markers, measurements of length, angle, area, etc).

Transmission of images is not as simple as it may seem. The main problem is their size. As mentioned, the diagnosis can only be done based on the diagnostic quality images - such as DICOM. The size of a single slice can vary from half MB (for CT) to several MB (for X-ray). Of course, the total size of the study also differs (see Appendix 1). Standard CT contains over a hundred layers, while the X-ray study usually contains only a few photos. This gives us an overall view of the size of the study which must be sent.

For the studies for which individual slices do not exceed a certain size the slice to be sent first is the one currently observed by the initiator of the communication. Subsequent layers are downloaded in the background, so that physicians can begin work as soon as possible with the first image. For studies with larger slices, such as X-ray examination the first to be sent is what the user can see - only the real portion of the compressed image. Then more information about the image is sent.

The user on the other side sees this as gradual improvement of the quality in the images observed. He may also begin to perform his graphic operations.

As for the transmission of images, after each modification it would involve a considerable burden on the network; therefore, in our solution only information about the operations performed is sent. It is simpler (because of the underlying architecture of our application) to synchronise stacks of graphical operations that are then performed on both sides.

The problem occurs when the users employ different versions of the application, for example, if one party does not have a tool that was used by the other side. In such a case the effect of the operation will be sent - the modified data set. The security of the image and information transfer between parties is ensured by the public-key encryption (SSL) [101]. Of course, this way you can also send documents bearing a digital signature. This way you can send a description - diagnosis, which thanks to qualified digital signature gains legal force and it is not necessary to send the "paper" version of the document.

A.2.6 Results and conclusions

Preliminary test results of the solution presented are promising and demonstrate its high efficiency. It ensures the safety, optimal speed of communication and work on the study in real time. See Fig. A.5 for some screen-shots of the tool.

The platform presented is an extension of fully functional diagnostic software. Applications of this type are used employing workstations at which radiologists work describing the studies. Owing to this, the doctor does not need to use an extra software in addition to that he is used to working with every day. In the near future, we intend to continue testing communication performance, depending on the type of study and quality of the internet connection. Surely, there is room to optimise and improve the potential slowdowns in the communications.

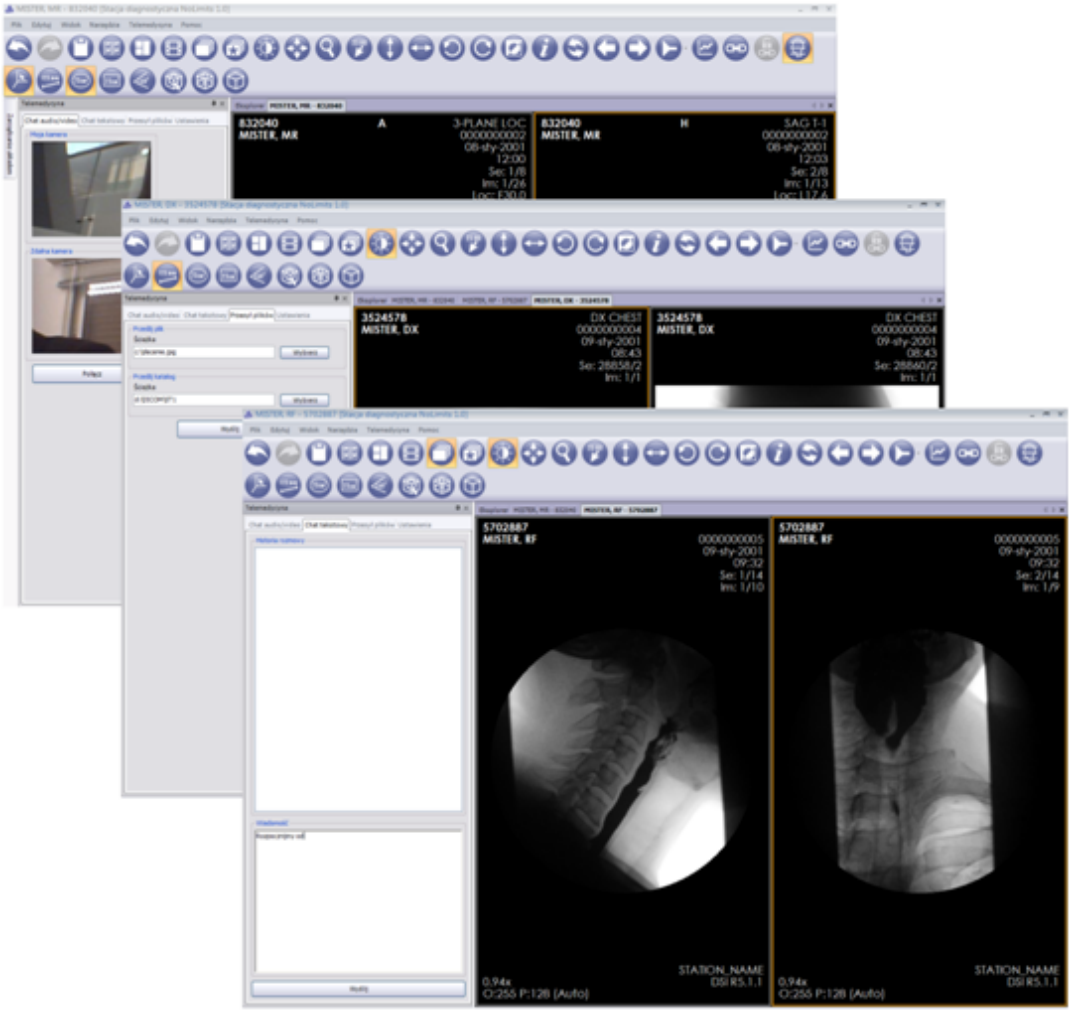

Figure A.5: Features of the teleconsulting tool.

Appendix B

Digital Imaging and Communications in Medicine

The DICOM - Digital Imaging and Communications in Medicine - is a standard that describes and unifies handling, storing, printing and transmitting information in medical imaging. It defines a file format and a network communications protocol (application protocol using TCP/IP for communication). DICOM files can be exchanged between two entities. Both sides must be capable of receiving patient data or/and images in a DICOM format. The National Electrical Manufacturers Association (NEMA) holds the copyright to this standard [102]. It was developed by the DICOM Standards Committee, the members of which partly are also members of the NEMA.

Thank to the DICOM scanners, servers, workstations, printers and network hardware from multiple manufacturers can be integrated into PACS - picture archiving and communication system. Medical devices come with the DICOM conformance statement that states the DICOM classes given device supports. The DICOM has been widely adopted by clinics and is starting to be used even by smaller units like dentist's and doctor's offices.

B.1 Parts of the Standard

The DICOM is known as the NEMA Standard PS3, and as ISO Standard 12052. The DICOM standard is divided into independent parts [103]:

- PS 3.1: Introduction and Overview
- PS 3.2: Conformance
- PS 3.3: Information Object Definitions
- PS 3.4: Service Class Specifications
- PS 3.5: Data Structure and Encoding
- PS 3.6: Data Dictionary
- PS 3.7: Message Exchange

- PS 3.8: Network Communication Support for Message Exchange
- PS 3.9: Retired (formerly Point-to-Point Communication Support for Message Exchange)
- PS 3.10: Media Storage and File Format for Data Interchange
- PS 3.11: Media Storage Application Profiles
- PS 3.12: Storage Functions and Media Formats for Data Interchange
- PS 3.13: Retired (formerly Print Management Point-to-Point Communication Support)
- PS 3.14: Grayscale Standard Display Function
- PS 3.15: Security and System Management Profiles
- PS 3.16: Content Mapping Resource
- PS 3.17: Explanatory Information
- PS 3.18: Web Access to DICOM Persistent Objects (WADO)

B.2 Data Format

The DICOM data object contains a number of attributes, including such items as name, ID, patient‘s name, etc. It also includes an attribute with the image pixel data. The main object has no "header", only a list of attributes containing pixel data (see Figure B.1). One DICOM object can only contain one attribute with pixel data. For many modalities, this is equal to a single image. This attribute can, however, contain multiple "frames", so cine loops or other multi-frame data can be stored. Three- or four-dimensional data can also be stored here, for example NM data, that by definition is a multi-dimensional multi-frame image. A variety of methods can be applied to compressed pixel data, for example JPEG, JPEG Lossless, JPEG 2000, Run-length encoding (RLE) and LZW (zip). The latter method can be used not only for pixel data but for the whole data set.

B.3 Coordinate system

The DICOM uses the patient-centred coordinate system convention. In this convention, the coordinate axes are oriented in the same direction, regardless of the patent's position in the scanner - they are fixed to the scanner (see Fig. B.2 and B.3). DICOM files contain tags that provide information about image position and orientation in the scanner coordinate system.

Group-Element	Tag Name	VR	Length	Value
(0002, 0001)	File Meta Information Version	OB	2	1\1
(0002, 0002)	Media Storage SOP Class UID	UI	26	1.2.840.10008.5.1.4.1.1.4
(0002, 0003)	Media Storage SOP Instance UID	UI	50	1.2.840.113619.2.5.1762583153.215519.978957063.162
(0002, 0010)	Transfer Syntax UID	UI	18	1.2.840.10008.1.2
(0002, 0012)	Implementation Class UID	UI	28	1.2.276.0.7230010.3.0.3.5.4
(0002, 0013)	Implementation Version Name	SH	16	OFFIS_DCMTK_354
(0002, 0016)	Source Application Entity Title	AE	2	!!
(0008, 0005)	Specific Character Set	CS	10	ISO_IR 100
(0008, 0008)	Image Type	CS	22	ORIGINAL\PRIMARY\OTHER
(0008, 0016)	SOP Class UID	UI	26	1.2.840.10008.5.1.4.1.1.4
(0008, 0018)	SOP Instance UID	UI	50	1.2.840.113619.2.5.1762583153.215519.978957063.162
(0008, 0020)	Study Date	DA	8	20010108
(0008, 0021)	Series Date	DA	8	20010108
(0008, 0022)	Acquisition Date	DA	8	20010108
(0008, 0023)	Content Date	DA	8	20010108
(0008, 0030)	Study Time	TM	6	120022
(0008, 0031)	Series Time	TM	6	121759
(0008, 0032)	Acquisition Time	TM	6	121759
(0008, 0033)	Content Time	TM	6	121759
(0008, 0050)	Accession Number	SH	10	0000000002
(0008, 0060)	Modality	CS	2	MR
(0008, 0070)	Manufacturer	LO	18	GE MEDICAL SYSTEMS
(0008, 0080)	Institution Name	LO	0	
(0008, 0090)	Referring Physician's Name	PN	0	
(0008, 1010)	Station Name	SH	12	STATION_NAME
(0008, 1030)	Study Description	LO	12	BRAIN SELLA
(0008, 103e)	Series Description	LO	10	COR FSE T2
(0008, 1060)	Name of Physician(s) Reading Study	PN	0	
(0008, 1070)	Operators' Name	PN	0	
(0008, 1090)	Manufacturer's Model Name	LO	14	GENESIS_SIGNA
(0009, 0010)	Private Creator Code	LO	12	GEMS_IDEN_01
(0009, 1001)	Private Tag	UN	14	(0009,1001) Private Tag
(0009, 1002)	Private Tag	UN	4	(0009,1002) Private Tag
(0009, 1004)	Private Tag	UN	6	(0009,1004) Private Tag
(0009, 1027)	Private Tag	UN	4	(0009,1027) Private Tag
(0009, 1030)	Private Tag	UN	8	(0009,1030) Private Tag
(0009, 1031)	Private Tag	UN	4	(0009,1031) Private Tag
(0009, 10e3)	Private Tag	UN	32	(0009,10e3) Private Tag
(0009, 10e6)	Private Tag	UN	2	(0009,10e6) Private Tag
(0009, 10e7)	Private Tag	UN	4	(0009,10e7) Private Tag
(0009, 10e9)	Private Tag	UN	4	(0009,10e9) Private Tag
(0010, 0010)	Patient's Name	PN	10	MISTER^MR
(0010, 0020)	Patient ID	LO	6	832040
(0010, 0030)	Patient's Birth Date	DA	0	
(0010, 0040)	Patient's Sex	CS	0	
(0010, 1010)	Patient's Age	AS	0	
(0010, 1030)	Patient's Weight	DS	0	
(0010, 21b0)	Additional Patient History	LT	0	
(0011, 0010)	Private Creator Code	LO	12	GEMS_PATI_01
(0011, 1010)	Private Tag	UN	2	(0011,1010) Private Tag

Figure B.1: Sample header information of the MR image DICOM file.

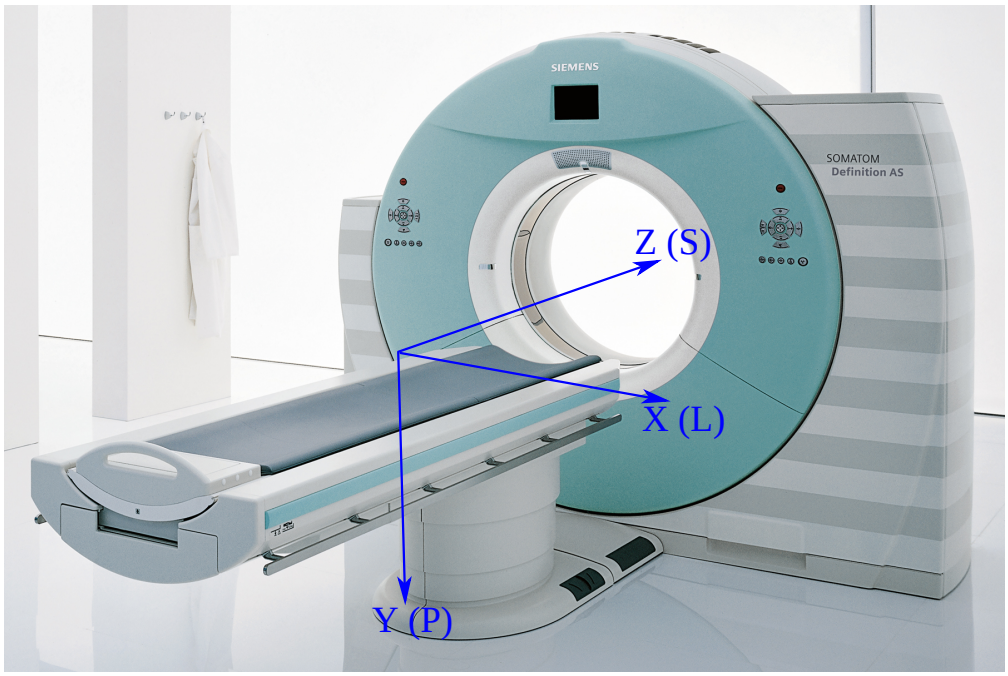

Figure B.2: Scanner coordinate system visualized (supine - face-up - patient). L - left, P - posterior, H - head (Superior).

1. **Image Position** $(0020, 0032)$: specifies the (x, y, z) coordinates of the upper left hand corner of the image (first voxel transmitted).

2. **Image Orientation** $(0020, 0037)$: specifies the direction cosines of the first row and the first column with respect to the patient. The direction of the axes is defined by the patient's orientation - LPS system (x-axis increasing to the left hand side of the patient, y-axis increasing to the posterior side of the patient and z-axis increasing toward the head of the patient) (see Fig. B.2)

3. **Patient position** $(0018, 5100)$: descriptor relative to the equipment. Possible values: HFP= head first-prone, HFS=head first-supine, HFDR= head first-decibitus right, HFDL = head first-decubiturs left, FFP = feet first-prone, FFS, FFDR, FFDL. Required for CT and MR images.

The DICOM standard explains these as follows [104]:

- **C.7.6.1.1.1 Patient Orientation.**
  "The Patient Orientation $(0020, 0020)$ relative to the image plane shall be specified by two values that designate the anatomical direction of the positive row axis (left to right) and the positive column axis (top to bottom). The first entry is the direction of the rows, given by the direction of the last pixel in the

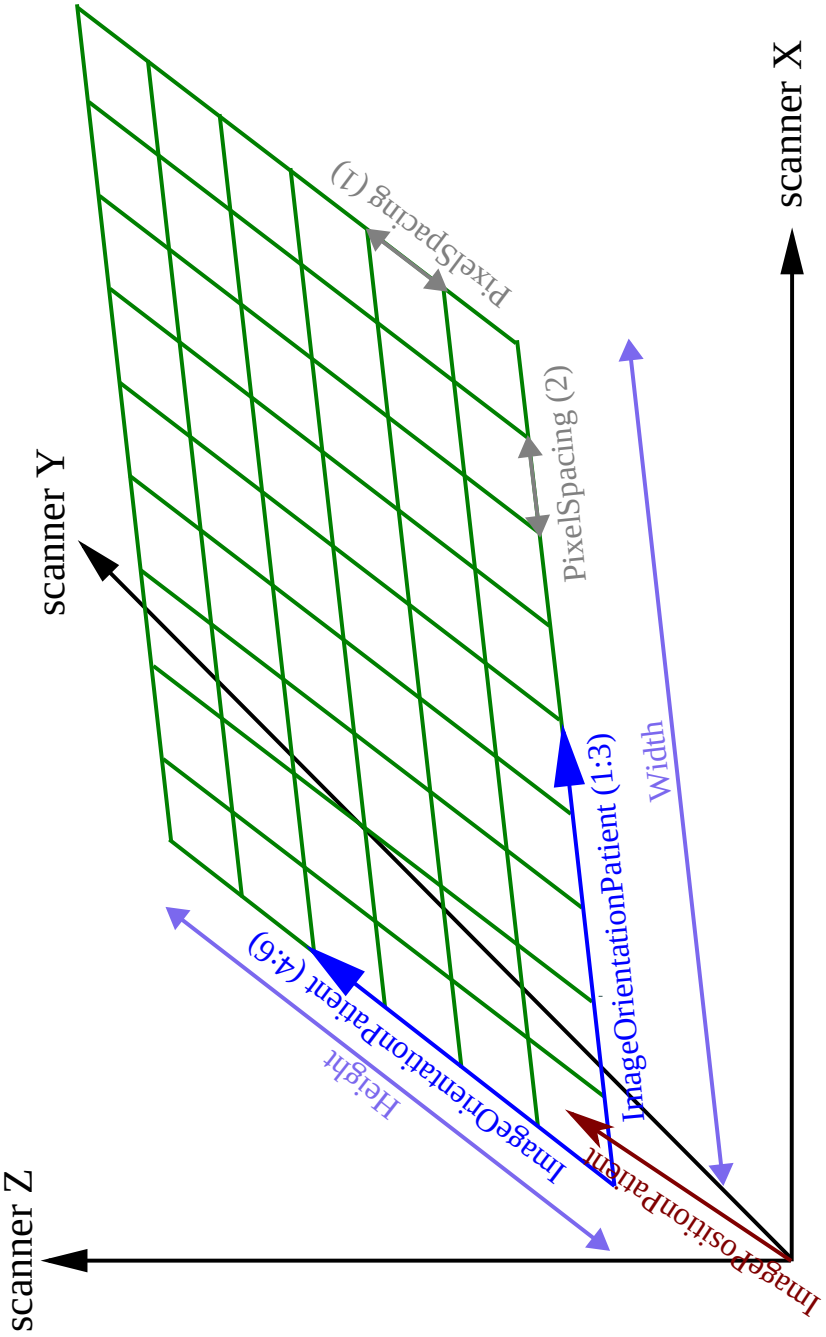

Figure B.3: Visualization of the image coordinate system in the scanner/patient coordinate system.

first row from the first pixel in that row. The second entry is the direction of the columns, given by the direction of the last pixel in the first column from the first pixel in that column. Anatomical direction shall be designated by the capital letters: A (anterior), P (posterior), R (right), L (left), H (head), F (foot) (see Fig. B.4). Each value of the orientation attribute shall contain at least one of these characters. If refinements in the orientation descriptions are to be specified, then they shall be designated by one or two additional letters in each value. Within each value, the letters shall be ordered with the principal orientation designated in the first character."

- **C.7.6.2.1.1 Image Position And Image Orientation.**
  "The Image Position $(0020, 0032)$ specifies the x, y, and z coordinates of the upper left hand corner of the image; it is the centre of the first voxel transmitted. Image Orientation $(0020, 0037)$ specifies the direction cosines of the first row and the first column with respect to the patient. These Attributes shall be provided as a pair. Row value for the x, y, and z axes respectively followed by the Column value for the x, y, and z axes respectively. The direction of the axes is defined fully by the patient's orientation. The x-axis is increasing to the left hand side of the patient. The y-axis is increasing to the posterior side of the patient. The z-axis is increasing toward the head of the patient. The patient based coordinate system is a right handed system, i.e. the vector cross product of a unit vector along the positive x-axis and a unit vector along the positive y-axis is equal to a unit vector along the positive z-axis."

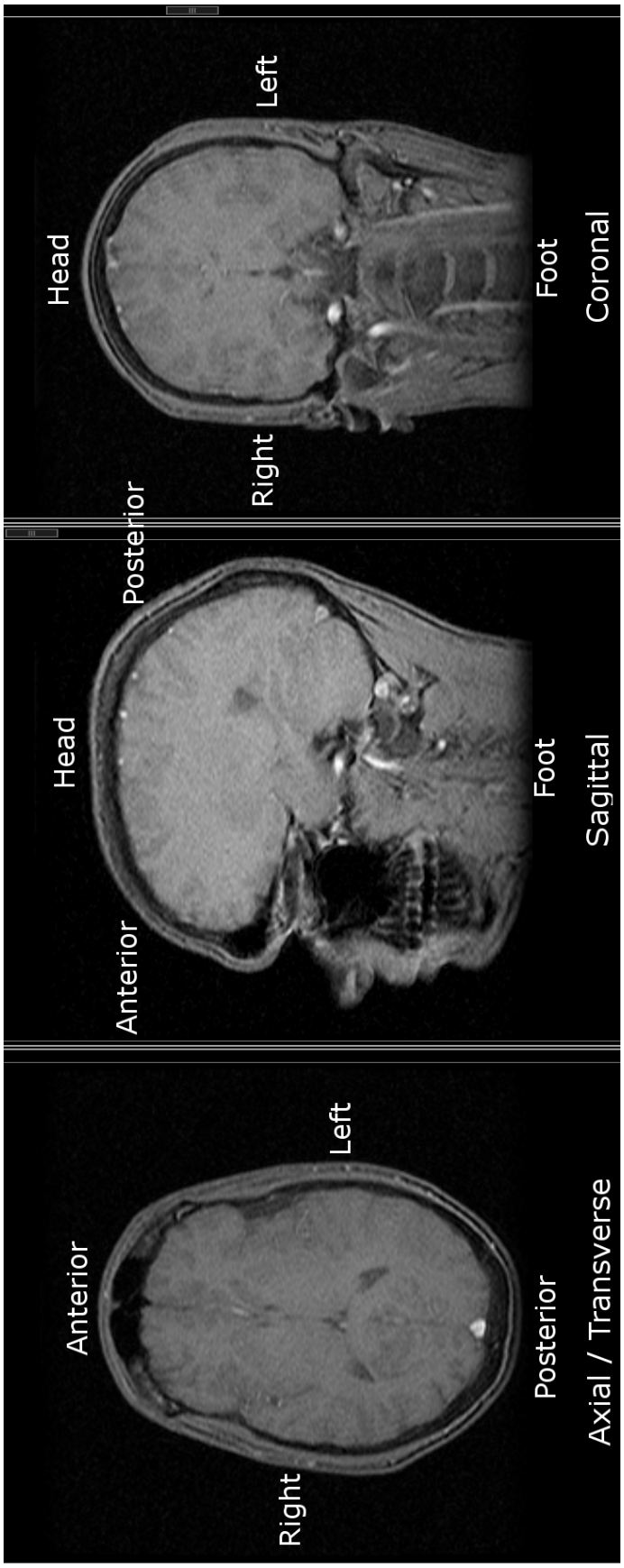

Figure B.4: Radiographic presentations

Appendix C

Digital image processing

C.1 Geometry

C.1.1 Lines

A line on a Cartesian plane [105, 106] can be written as a linear equation. In two dimensions, the description can be given in several forms:

- general form
$$Ax + By = C \tag{C.1}$$
where neither A nor B is equal to zero.

- slope-intercept form
$$y = mx + b \tag{C.2}$$
where m is the slope and b is the y-intercept.

- point-slope form
$$y - y_1 = m(x - x_1) \tag{C.3}$$
where m is the slope and (x_1, y_1) is any point on the line.

- two-point form
$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{C.4}$$
where (x_1, y_1) and (x_2, y_2) are two points on the line that $x_1 \neq x_2$.

Line in the three dimensional Cartesian space is described by parametric equations:

$$x = x_0 + At \tag{C.5}$$
$$y = y_0 + Bt \tag{C.6}$$
$$z = z_0 + Ct \tag{C.7}$$

where x, y, z are functions of the independent variable t. (x_0, y_0, z_0) is any point on the line. The vector (A, B, C) is parallel to the line.

Distance between a point and a line The distance between point $P = (x_p, y_p)$ and a line, given in the general form, can be written as:

$$d = \frac{|Ax_p + By_p + C|}{\sqrt{A^2 + B^2}} \tag{C.8}$$

Segment bisection Bisection of the segment described by two points $A = (A_x, A_y)$ and $B = (B_x, B_y)$ is given by:

$$(2x - A_x - B_x)(A_x - B_x) + (2y - A_y - B_y)(A_y - B_y) = 0 \tag{C.9}$$

After some transformations, we get a bisection equation in the general form:

$$2(A_x - B_x)x + 2(A_y - B_y)y + (B_x^2 + B_y^2 - A_x^2 - A_y^2) = 0 \tag{C.10}$$

C.1.2 Plane

A plane can be described by a point on it and a vector normal to it. Vector N is the normal to the plane and p are all points that:

$$N \cdot p = k \tag{C.11}$$

where $\cdot$ is the dot product between the two vectors: $A \cdot B = (A_x, A_y, A_z) \cdot (B_x, B_y, B_z) = A_x B_x + A_y B_y + A_z B_z$ Given any point a on the plane

$$N \cdot (p - a) = 0 \tag{C.12}$$

A plane can also be also described in a standard form:

$$Ax + By + Cz + D = 0 \tag{C.13}$$

The normal to the plane is the vector (A, B, C).

Given three points in space (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), the equation of the plane through these points is given by the following determinants.

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \tag{C.14}$$

$$B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \tag{C.15}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \tag{C.16}$$

$$D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \tag{C.17}$$

Expanding the above gives

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \tag{C.18}$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \tag{C.19}$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \tag{C.20}$$
$$-D = x_1(y_2z_3 - y_3z_2) + x_2(y_3z_1 - y_1z_3) + x_3(y_1z_2 - y_2z_1) \tag{C.21}$$

If the points are collinear, then the normal vector (A, B, C) will be $(0, 0, 0)$. The sign of $s = Ax + By + Cz + D$ determines which side of the point (x, y, z) lies in relation to the plane. If $s > 0$, then the point lies on the same side as the normal (A, B, C). If $s < 0$, then it lies on the opposite side, if $s = 0$ then the point (x, y, z) lies on the plane.

Intersection of two planes Two planes can be parallel or intersect making a line. Let us define two planes as follows

$$N_1 \cdot p = d_1 \tag{C.22}$$
$$N_2 \cdot p = d_2 \tag{C.23}$$

where N are normal vectors.

Any point in the space can be written as

$$p = c_1N_1 + c_2N_2 + c_3(N_1 \times N_2) \tag{C.24}$$

since $\{N_1, N_2, (N_1 \times N_2)\}$ is a basis. c_3 is the line's parameter, c_1 and c_2 are constants.

Taking the dot product of the C.24 with each normal, gives two equations with c_1 and c_2 as unknowns.

$$N_1 \cdot p = d_1 = c_1N_1 \cdot N_1 + c_2N_1 \cdot N_2 \tag{C.25}$$
$$N_2 \cdot p = d_2 = c_1N_1 \cdot N_2 + c_2N_2 \cdot N_2 \tag{C.26}$$

Solving for c_1 and c_2

$$c_1 = (d_1N_2 \cdot N_2 - d_2N_1 \cdot N_2)/det \tag{C.27}$$
$$c_2 = (d_2N_1 \cdot N_1 - d_1N_1 \cdot N_2)/det \tag{C.28}$$
$$det = (N_1 \cdot N_1)(N_2 \cdot N_2) - (N_1 \cdot N_2)^2 \tag{C.29}$$

Firstly, we should check if planes aren't coincident or parallel. The easies way to do this is:

$$N_1 \times N_2 = 0 \tag{C.30}$$

Intersection of three planes The intersection of three planes can be a point, a line, or there is no intersection. Planes can be described by:

$$N_1 \cdot p = d_1 N_2 \cdot p = d_2 N_3 \cdot p = d_3 \tag{C.31}$$

Intersection point P can be expressed as:

$$P = \frac{d_1(N_2 \times N_3) + d_2(N_3 \times N_1) + d_3(N_1 \times N_2)}{N_1 \cdot (N_2 \times N_3)} \tag{C.32}$$

The denominator is equal to 0 if $N_2 \times N_3 = 0$ - planes are parallel. Or N_1 is a linear combination of N_2 and N_3.

C.1.3 Fitting a plane into a set of points

Let $p_i = (x_i, y_i, z_i)$ be a set of N points in a 3D space. The distance, R_i, of point p_i from plane C.13 is

$$R_i = \frac{Ax_i + By_i + Cz_i - D}{\sqrt{A^2 + B^2 + C^2}} \tag{C.33}$$

if the plane is normalised, i.e. $A^2 + B^2 + C^2 = 1$, $R_i = Ax_i + By_i + Cz_i - D$.

R_i can be *positive* or *negative*. To change the sign of the R_i, it can always be multiplied by -1.

Least Squares Plane We wish to minimise:

$$Q(A, B, C, D) = \sum_{i=1}^{N} R_i^2 \tag{C.34}$$

$$Q(A, B, C, D) = \sum_{i=1}^{N} (Ax_i + By_i + Cz_i - D)^2 \tag{C.35}$$

Now we take the partial derivatives of Q with respect to A, B, C, D and set them to zero. This leads to four equations to be solved for A, B, C and D.

$$\frac{\partial Q}{\partial A} = \sum_{i=1}^{N} 2x_i(Ax_i + By_i + Cz_i - D) = 0 \tag{C.36}$$

$$\frac{\partial Q}{\partial B} = \sum_{i=1}^{N} 2y_i(Ax_i + By_i + Cz_i - D) = 0 \tag{C.37}$$

$$\frac{\partial Q}{\partial C} = \sum_{i=1}^{N} 2z_i(Ax_i + By_i + Cz_i - D) = 0 \tag{C.38}$$

$$\frac{\partial Q}{\partial D} = -2\sum_{i=1}^{N} (Ax_i + By_i + Cz_i - D) = 0 \tag{C.39}$$

The last equation can be rewritten as (noting that $\sum_{i=1}^{N} D = ND$)

$$D = Ax_0 + By_0 + Cz_0 \tag{C.40}$$

where $x_0 = \frac{\sum_{i=1}^{N} x_i}{N}$, $y_0 = \frac{\sum_{i=1}^{N} y_i}{N}$ and $z_0 = \frac{\sum_{i=1}^{N} z_i}{N}$.

Equation C.40 shows that the best fit plane passes through the centre of mass. Subtracting each point from the centre of mass and substituting it into C.36 leads to a set of equations:

$$WP = 0 \tag{C.41}$$

$$W = \begin{bmatrix} \sum(x_i - x_0)^2 & \sum(x_i - x_0)(y_i - y_0) & \sum(x_i - x_0)(z_i - z_0) \\ \sum(x_i - x_0)(y_i - y_0) & \sum(y_i - y_0)^2 & \sum(y_i - y_0)(z_i - z_0) \\ \sum(x_i - x_0)(z_i - z_0) & \sum(y_i - y_0)(z_i - z_0) & \sum(z_i - z_0)^2 \end{bmatrix} \tag{C.42}$$

$$P = \begin{bmatrix} A \\ B \\ C \end{bmatrix} \tag{C.43}$$

The above must be solved to get the least squares planes. This equation can have an easy solution $A = B = C = 0$. To avoid this trivial solution, we need to set some conditions. The most common is:

$$A^2 + B^2 + C^2 = 1 \tag{C.44}$$

It can be shown that, with this pre-condition, the solution becomes an Eigenvalue problems:

$$WE = VE \tag{C.45}$$

where: V are the eigenvalues and E the eigenvectors.

Equation C.45 will return 3 eigenvalues and a 33 matrix (eigenvectors). The eigenvalues are the sums of the squares of the distances. The eigenvectors will contain three sets of A, B and C values - in columns of matrix E. Lastly, the set of A, B, C corresponding to the smallest eigenvalue is chosen.

C.1.4 Polygons

Polygon area Let us consider a polygon built with line segments between N vertices (x_i, y_i), $i = 0...N-1$. The last vertex (x_N, y_N) is the same as the first - the polygon is closed. The area of the non-self-intersecting polygon is:

$$A = \frac{1}{2} \sum_{i=0}^{N-1} (x_i y_{i+1} - x_{i+1} y_i) \tag{C.46}$$

Polygon centroid - 2D case The centroid is also called the "centre of gravity" or the "centre of mass". The position of the centroid, if the polygon is from a material of uniform density, is as follows:

$$c_x = \frac{1}{6A}\sum_{i=0}^{N-1}(x_i + x_{i+1})(x_i + y_{i+1} - x_{i+1}y_i) \tag{C.47}$$

$$c_y = \frac{1}{6A}\sum_{i=0}^{N-1}(y_i + y_{i+1})(x_i + y_{i+1} - x_{i+1}y_i) \tag{C.48}$$

Polygon centroid - 3D case The centroid of a 3D object created by a collection of N triangles with vertices (a_i, b_i, c_i) is given by:

$$C = \frac{\sum_{i=0}^{N-1} A_i R_i}{\sum_{i=0}^{N-1} A_i} \tag{C.49}$$

$$R_i = \frac{(a_i + b_i + c_i)}{3} \tag{C.50}$$

$$A_i = \|(b_i - a_i) \otimes (c_i - a_i)\| \tag{C.51}$$

where: R_i is the average of the vertices of the i'th triangle and A_i is the double area of the i'th triangle. Triangles are assumed to be of uniform mass; they do not need to form a solid object or be connected.

C.1.5 Position of a point inside a triangle

In the barycentric coordinate system the location of a point is specified as the centre of mass of masses placed at vertices of a simplex (for example, a triangle). The idea was introduced in 1827 by August Ferdinand Möbius [107, 108]. In the case of a triangle, barycentric coordinates of point P (with respect to triangle ABC - see Fig. C.1) are proportional to the areas (signed):

$$P = \frac{1}{\triangle\ ABC}((\triangle\ PBC)A + (\triangle\ PCA)B + (\triangle\ PAB)C) \tag{C.52}$$

Now we will present calculations that enable us to calculate the position of points with respect to triangle vertices. We pick one of the three points and describe all other positions on a plane as relative to that vertex. We choose point A to be the origin. Next, basis vectors must be defined to describe coordinates of all locations on the plane. We choose vectors created by point A and other vertices:

$$v_0 = C - A \tag{C.53}$$

$$v_1 = B - A \tag{C.54}$$

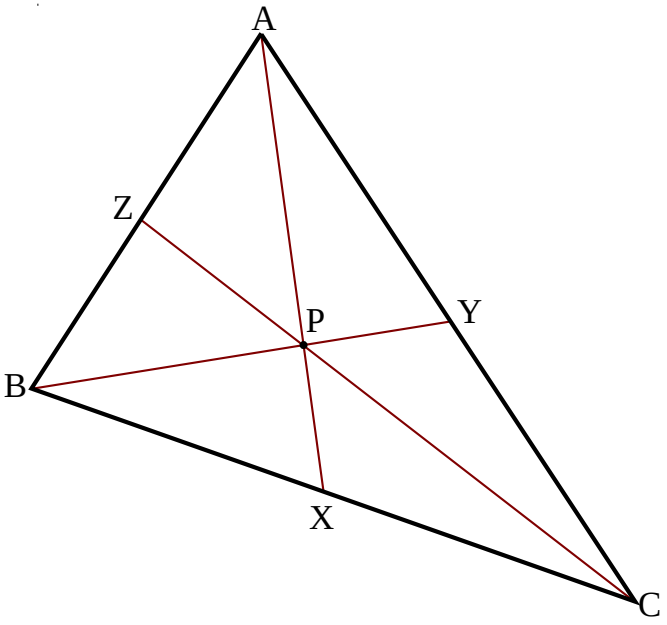

Figure C.1: Triangle ABC with point P marked.

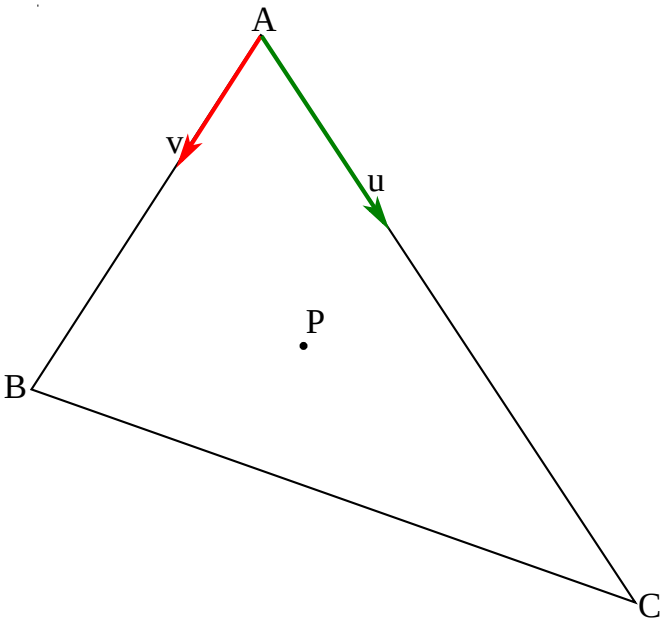

Figure C.2: Vectors u and v describing position of point P relative to point A.

All points can now be reached by moving along vector v_1 and then along vector v_2 (see Fig. C.2). Therefore, any point on the plane can be written as:

$$P = A + u * v_0 + v * v_1 \tag{C.55}$$

Important attributes:

- if u or v are < 0, point P is outside the triangle
- if u or v are > 1, point P is outside the triangle
- if $u + v > 1$, point P is also outside the triangle (crossed BC edge)

Having u and v, it is easy to evaluate point P with equation C.55. Next, a formula to calculate u and v for a given point P is needed. From C.55 we get

$$v_2 = u * v_0 + v * v_1 \tag{C.56}$$

where $v_2 = P - A$. There are two unknowns: u and v. Therefore, two equations are needed to find them. To find them, we *dot* both sides by v_0 and v_1:

$$v_2 \cdot v_0 = (u * v_0 + v * v_1) \cdot v_0 = u * (v_0 \cdot v_0) + v * (v_1 \cdot v_0) \tag{C.57}$$
$$v_2 \cdot v_1 = (u * v_0 + v * v_1) \cdot v_1 = u * (v_0 \cdot v_1) + v * (v_1 \cdot v_1) \tag{C.58}$$

Solving these equations, we get:

$$u = \frac{(v_1 \cdot v_1)(v_2 \cdot v_0) - (v_1 \cdot v_0)(v_2 \cdot v_1)}{(v_0 \cdot v_0)(v_1 \cdot v_1) - (v_0 \cdot v_1)(v_1 \cdot v_0)} \tag{C.59}$$
$$v = \frac{(v_0 \cdot v_0)(v_2 \cdot v_1) - (v_0 \cdot v_1)(v_2 \cdot v_0)}{(v_0 \cdot v_0)(v_1 \cdot v_1) - (v_0 \cdot v_1)(v_1 \cdot v_0)} \tag{C.60}$$

C.2 Image filtering

Filtrating functions use a small neighbourhood of a pixel in an input image to calculate a new pixel value on the output image. This image processing technique has been widely discussed in literature [109, 110, 111, 112]. These methods can be divided into groups according to the aim of processing:

- *Smoothing* - is to suppress noise or other small fluctuations in the image (this is equivalent to suppressing high frequencies in the frequency domain). This operation, unfortunately, also blurs sharp edges that carry important information about the image.
- *Edge enhancement* (gradient operators) - is based on local derivatives of the image function. They are greater where the image changes rapidly - borders (suppression of low frequencies in the frequency domain). If we apply a gradient operator to an image, the noise level will be increased.

The above techniques have conflicting aims. Fortunately, some filtration methods solve this problem and allow for simultaneous edge enhancement and smoothing. There is also a different classification of filtering algorithms according to transformation properties: *linear* and *nonlinear*.

Linear operations calculate the value of the output image pixel $I_{out}[x, y]$ as a linear combination of values of local neighbours of pixel $I_{in}[x, y]$ of the input image. The influence of the pixel is described by the h function.

$$I_{out}[x, y] = \sum_{i \in O} \sum_{j \in O} h[x - i, y - j] I_{in}[i, j] \tag{C.61}$$

This equation is equal to discrete convolution with the h kernel - a convolution mask. Rectangular neighbourhood O very often has an odd number of pixels in columns and rows, so the middle pixel can be chosen.

C.2.1 Image smoothing

As mentioned earlier, smoothing causes sharp edges to become blurred. We shall therefore present some edge preserving methods. The main idea of these methods is that the average is computed only when neighbours of a pixel have similar properties as the processed one. Local image smoothing effectively eliminates small noise and degradations visible as thin stripes; however, it doesn't work for thick stripes or large blob like artifacts.

Averaging assumes that the noise value (v) at each pixel is an independent variable, having zero mean and standard deviation σ. Such an image can be obtained by capturing the same scene (static) several times. The resulting smoothing is an average of the same n pixels in the images $g_1, ..., g_n$ with noise values $v_1, ..., v_n$:

$$\frac{\sum_{i=1}^{n} g_i}{n} + \frac{\sum_{i=1}^{n} v_i}{n} \tag{C.62}$$

Thus, the smoothing can be acquired without blurring if n images (of the same scene) are available:

$$I_{out}[x, y] = \frac{1}{n} \sum_{i=1} n g_k[x, y] \tag{C.63}$$

If only one image is available, averaging is done in a local neighbourhood. We can accept results if the noise is smaller than the smallest ROI in the image, but still blurring is a disadvantage.

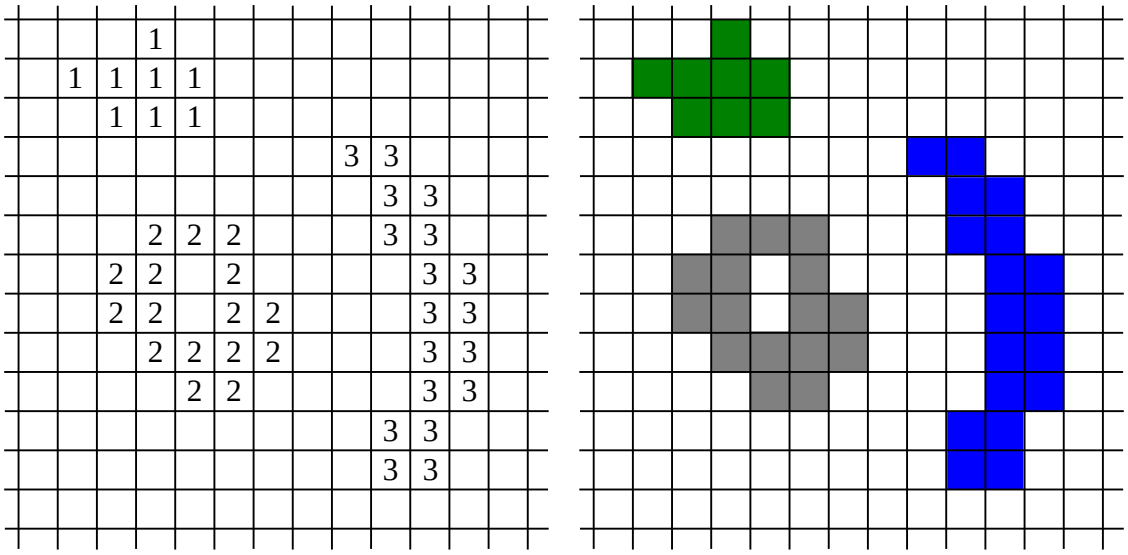

Figure C.3: Connected components and corresponding labels.

C.3 Region identification

C.3.1 Connected components labelling

A connected component is defined as a set of pixels where each pixel is connected to other pixels. A connection may be understood as a common feature between pixels in the same region. Generally, connected components labelling follows image segmentation, when an image is divided into separate regions of interests (ROIs). A component labelling algorithm finds all connected components in an image and assigns a unique label to all points in the same component (see Fig. C.3).

We will now present a basic algorithm for finding connected components for a two dimensional image. This is a simple method, but there are algorithms optimized and modified to be faster, for example [113] (this algorithm is used in our project). Figure C.4 shows masks used for region identification.

First pass searches the whole image I row by row assigning a non-zero value v to each non-zero pixel $I(x, y)$. Value v is chosen based on the values of the neighbours of the pixel.

- if all the neighbours are background pixels, $I(x, y)$ is assigned a new unused label
- if there is one neighbouring pixel with a non-zero label, this label is assigned to the pixel $I(x, y)$

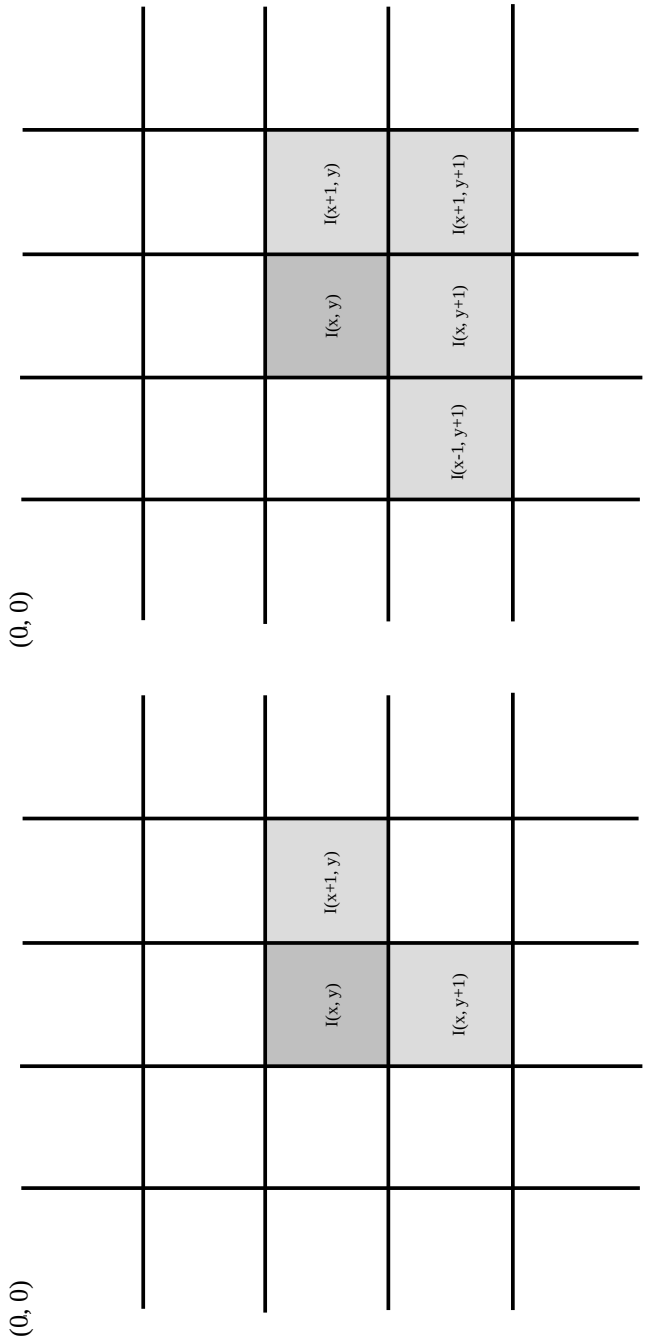

Figure C.4: Masks for region identification: in *4-connectivity* (on the left) and in *8-connectivity* (on the right).

- if more than one neighbours have a non-zero value, assign the label of any one of them to pixel $I(x, y)$. If the labels of the neighbours differ (*label collision*) remember the label pair as being equivalent.

Second pass all of the region-pixels have been labeled, but in some regions there are pixels with different labels (*label collision*). The whole image must be scanned again and the pixels re-labeled according to the equivalence information.

C.3.2 Connected components labeling in 3D

Sometimes, there arises a need of connected components labelling in three dimension. We have, therefore, developed a suitable method to face this task.

The technique is rather simple:

1. Search the entire image (data set) for the first non-background voxel not yet labeled. This voxel V_0 is a starting point for next step of labelling
2. Starting from V_0, launch the region grow procedure [33] accepting all VOIs voxel but ignoring the background.
3. If there are no other non-labeled and non-background voxels, stop computation, or else go to point 1.

C.4 Border tracing

Borders can be detected if regions have been marked on the image. Most often, this is achieved by some form of segmentation that produces a set of ROI on the image. First, we must assume that the image is either binary or all ROIs have been labeled. We will present several algorithms detecting region borders in both 4- and 8-connectivity (they work for regions larger than one pixel, because this case is trivial). An inner border is a subset of the region, while an outer border is not.

C.4.1 Inner boundary tracing

1. Scan the image from top-left till a pixel belonging to a new region is found. This pixel, P_0, is a starting pixel of a region border. We define P_{-1} as a variable holding the previous border pixel (initially it is set to the pixel on the left).
2. The 3×3 neighbourhood of the current element is searched in a clockwise direction, starting from the pixel next to P_{-1} in the clockwise sense (on the neighbourhood circle, Fig. C.5). That is, if $P_{-1} = (5, 5)$ and $P_0 = (6, 6)$, we

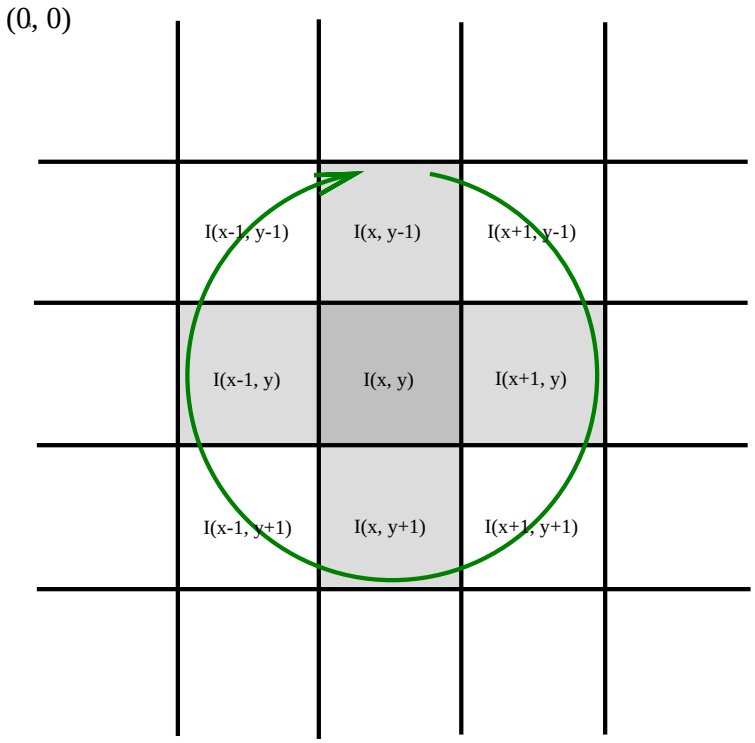

Figure C.5: Direction of scanning current pixel neighbourhood. Grey pixels represent 4-connectivity and together with the white ones they will give 8-connectivity.

start from $P = (6, 5)$. A new boundary element P_n is the first element found of the same value as the current element.

3. The algorithm stops when the current boundary pixel P_n is equal to the second border pixel P_1; furthermore, if the previous border pixel P_{n-1} is equal to P_0. Otherwise, go to the step 2.

4. Elements $P_0 ... P_{n-2}$ are representing the inner border.

C.4.2 Outer boundary tracing

1. Trace the inner region boundary in 4-connectivity.

2. The outer boundary consists of all non-region pixels that were tested during the search process.

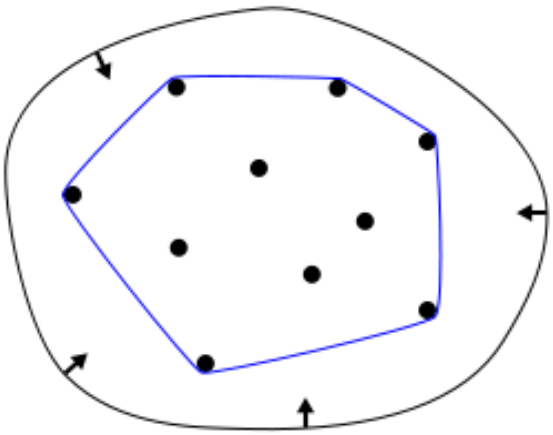

Figure C.6: Convex hull simple example

C.4.3 Convex hull

The convex hull of a set of points X is the minimal convex containing X [114] (see Fig. C.6).

To compute a convex hull for set of points in 2D; the following algorithm is used:

1. Find the least point A (with minimum y coordinate) as a starting point.
2. Find B where all points lie to the left of AB. To check this, we calculate the area of the triangle $P_0P_1P_2$ where $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$:

$$\frac{1}{2}\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix} \tag{C.64}$$

   If the area is positive, then P_2 is to the left of the line P_0P_1 (anti-clockwise order). If the area is negative, then the order is clockwise. If the area is 0, the points are collinear.
3. Repeat the above step till the top of the set is reached and continue down ensuring that all points lie to the right of the vortex analysed.

Bibliography

[1] J. Smietanski, R. Tadeusiewicz, and E. Luczynska. Texture Analysis in Perfusion Images of Prostate Cancer - a Case Study. *International Journal of Applied Mathematics and Computer Science*, 20(149-156), 2010. 2

[2] J. Kawa, P. Szwarc, B. Bobek-Billewicz, and E. Pietka. Multiseries MR Data in Brain Tumours Segmentation. *Advances in Intelligent and Soft Computing, Information Technologies In Biomedicine (Springer-Verlag)*, 2(69):53–64, 2010. 2

[3] M. E. Rossi, E. Jason, S. Marchesotti, P. Dastidar, J. Ollikainen, and S. Soimakallio. Diffusion Tensor Imaging Correlates with Lesion Volume in Cerebral Hemisphere infarctions. *BMC Medical Imaging*, 10, 2010. 2

[4] E. Pietka. *Zintegrowany system informacyjny w pracy szpitala*. PWN, 2004. 3

[5] A. C. Davison. *Statistical Models*. Cambridge University Press, 2003. 4

[6] R. H. Kartaszyǹski and P. Mikołajczak. Use of Interpolation Methods for Two-Dimensional Medical Images Enlargement (in Polish). *Fourth Conference of Computer Science Enthusiasts*, pages 49–58, 2005. 5, 15, 18, 68, 105

[7] M. Chlebiej, R. H. Kartaszyǹski, and P. Mikołajczak. Interpolation Role in Volumetric Medical Images Reconstruction (in Polish). *Varia Informatica. Calculations and Technologies, PTI*, pages 79–88, 2005. 5, 15, 18, 19, 20, 21, 22, 105

[8] R. H. Kartaszyǹski and P. Mikołajczak. Combined T1 & T2 MRI Brain Segmentation. *Advances in Intelligent and Soft Computing, Computer Recognition Systems 3 (Springer-Verlag)*, (57):423–430, 2009. 5, 105

[9] R. H. Kartaszyǹski and P. Mikołajczak. Combined MR Brain Sgmentation. *Annales UMCS Informatica*, 2009. 5, 105

[10] R. H. Kartaszyǹski and P. Mikołajczak. MRI Brain Segmentation Using Cellular Automaton Approach. *Lecture Notes in Computer Science (Springer-Verlag)*, 6374:17–24, 2010. 5, 29, 31, 32, 105

[11] R. H. Kartaszyǹski and P. Mikołajczak. CABRS - Cellular Automaton Based MRI Brain Segmentation. *Proceedings of the XI International Conference MIT 2006*, 2006. 5, 29, 31, 32, 105

[12] R. H. Kartaszyǹski and P. Mikołajczak. CATS - Cell Automata Based Tissue Segmentation of Two and Three Dimensional CT and MRI Medical Images. *Annales UMCS Informatica*, 2006. 5, 29, 31, 32, 105

[13] R. H. Kartaszyǹski and P. Mikołajczak. Volumetric Analysis of Tumours and Their Blood Vessels. *Advances in Intelligent and Soft Computing, Information Technologies in Biomedicine (Springer-Verlag)*, (47):184–191, 2008. 5, 105

[14] R. H. Kartaszyǹski and P. Mikołajczak. Noise Influence Reduction in Estimation of CBF, CBV and MTT, MRI Perfusion Parameters. *Advances in Soft Computing, Image Processing & Communications Challenges II (Springer-Verlag)*, 84:231–238, 2010. 6, 105

[15] R. H. Kartaszyǹski and P. Mikołajczak. Interpolation Sampling for Noise Reduction in Estimation of Weighted Perfusion Parameters. *International Conference on Information Communication Technologies in Health (INEAG Greece)*, pages 238–245, 2010. 6, 105

[16] R. H. Kartaszyǹski and P. Mikołajczak. Symmetry Plane of the Brain on Perfusion MR Images. *Advances in Intelligent and Soft Computing, Information Technologies In Biomedicine (Springer-Verlag)*, 2(69):65–72, 2010. 6, 105

[17] R. H. Kartaszyǹski and P. Mikołajczak. Thought-out Application Architecture as the Source of the IT Project Success. *Polish Journal of Environmental Studies*, 18(3B), 2009. 6, 105

[18] R. H. Kartaszyǹski and P. Mikołajczak. Teleradiology Consulting System. *International Conference on Information Communication Technologies in Health (INEAG Greece)*, pages 63–70, 2010. 6, 105

[19] Barbier and Lamalle. Methodology of Brain Perfusion Imaging. *J Magn Reson Imaging*, 13(4):496–520, Apr 2001. 7

[20] Luypaert and Boujraf. Diffusion and Perfusion MRI: Basic Physics. *European journal of radiology*, 38(1):19–27, Apr 2001. 7

[21] Ostergaard. Principles of Cerebral Perfusion Imaging by Bolus Tracking. *J Magn Reson Imaging*, 22(6):710–7, Dec 2005. 7

[22] Wu and Ostergaard. Technical Aspects of Perfusion-Weighted Imaging. *Neuroimaging clinics of North America*, 15(3):623–37, Aug 2005. 7

[23] Souvik Sen. Magnetic Resonance Imaging in Acute Stroke: Differential Diagnoses & Workup. http://emedicine.medscape.com/article/1155506-diagnosis, Aug 2010. 7, 8

[24] Detre and Alsop. Perfusion Magnetic Resonance Imaging with Continuous Arterial Spin Labeling: Methods and Clinical Applications in the Central Nervous System. *European journal of radiology*, 30(2):115–24, May 1999. 8, 58

[25] Buxton. Quantifying CBF with Arterial Spin Labeling. *J Magn Reson Imaging*, 22(6):723–6, Dec 2005. 8

[26] Wong. Quantifying CBF with Pulsed ASL: Technical and Pulse Sequence Factors. *J Magn Reson Imaging*, 22(6):727–31, Dec 2005. 8

[27] J. L. Boxerman, B. R. Rosen, and R. M. Weisskoff. Signal-to-noise Analysis of Cerebral Blood Volume Maps from Dynamic NMR Imaging Studies. *Magn Reson Imaging*, 7:528–537, 1997. 9, 58

[28] Duyn and van Gelderen. Technological Advances in MRI Measurement of Brain Perfusion. *J Magn Reson Imaging*, 22(6):751–3, Dec 2005. 12

[29] C. R. Giardina and E. R. Dougherty. *Morphological Methods in Image and Signal Processing.* Prentice-Hall, 1988. 14

[30] A. V. Oppenheim, A. S. Willsky, and I. T. Young. *Systems and Signals.* Prentice-Hall, 1983. 14

[31] D. E. Dudgeon and R. M. Mersereau. *Multidimensional Digital Signal Processing.* Prentice-Hall, 1984. 14

[32] K. R. Castleman. *Digital Image Processing. Second ed.* Prentice-Hall, 1996. 14

[33] J. C. Russ. *The Image Processing Handbook. Second ed.* CRC Press, 1995. 14, 140

[34] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley, 1992. 14

[35] Linda G. Shapiro and George C. Stockman. *Computer Vision.* New Jersey, Prentice-Hall, 2001. 14

[36] Thomas M. Lehmann, Claudia Gönner, and Klaus Spitzer. Survey: Interpolation Methods in Medical Image Processing. *IEEE Transactions on medical imaging*, 18, November 1999. 15, 61

[37] J. A. Parker, R. V. Kenyon, and D. E. Troxel. Comparision of Interpolating Methods for Image Resampling. *IEEE Trans. Med. Imag.*, MI-2:31–39, 1983. 17

[38] K. G. Beauchamp. *Signal Processing.* George & Allen Unwin Ltd., 1973. 17

[39] P. E. Danielsson and M. Hammerin. High accuracy rotation of images. *Computer Vision, Graphics and Image Processing*, 54:340–344, July 1992. 17

[40] Edwin Catmull and Raphael Rom. A Class of Local Interpolating Splines. *Computer Aided Geometric Design*, pages 317–326, 1974. 17

[41] Erik H. W. Meijering. Spline Interpolation in Medical Imaging: Comparison with Other Convolution-Based Approaches. *Signal Processing X: Theories and Applications - Proceedings of EUSIPCO 2000*, IV:1989–1996, 2000. 17

[42] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990. 17

[43] C. Lanczos. *Discourse on Fourier Series*. Oliver & Boyd, 1966. 17

[44] N. A. Thacker, A. Jackson, D. Moriarty, and B. Vokurka. Renormalised Sinc Interpolation. *University of Manchester*, Tina Memo No. 1999-005, 1999. 16, 63

[45] Aili Li, Klaus Mueller, and Thomas Ernst. Methods for Efficient, High Quality Volume Resampling in the Frequency Domain. *IEEE Visualization*, pages 3–10, 2004. 18

[46] V. Spitzer, M. J. Ackerman, A. L. Scherzinger, and D. Whitlock. The Visible Human Male: Technical Report. *Journal of the American Medical Informatics Association*, 3:118–130, 1996. 18

[47] T. W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-8(8):630–632, 1978. 22

[48] C. K. Chow and T. Kaneko. Automatic boundary detection of the left ventricle from cineangiograms. *Computers and Biomedical Research*, 5:388–410, 1972. 23

[49] L. Grady. Random Walks for Image Segmentation. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006. 24

[50] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2), 2004. 24

[51] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992. 27

[52] J. Von Neumann. Theory of Self-Reproducing Automata. *University Of Illinois Press. Ed. And Completed by A. Burks.*, 1966. 27

[53] G.Hernandez and H. J. Herrmann. Cellular Automata for Elementary Image Enhancement. *CVGIP: Graphical Model and Image Processing*, 58:82–89, 1 1996. 27

[54] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982. 30

[55] Berthold Klaus and Paul Horn. *Robot vision*. MIT Press, 1986. 30

[56] T. Heimann, M. Thorn, T. Kunert, and H.-P. Meinzer. New methods for Leak Detection and Contour Correction in Seeded Region Growing Segmentation. *International Archives of Photogrammetry and Remote Sensing*, XXXV:317–322, 2004. 30

[57] http://www.cma.mgh.harvard.edu/ibsr/. 33

[58] P.S. Windyga. Fast Impulsive Noise Removal. *Image Processing, IEEE Transactions*, 10, 2001. 38

[59] A. Tuzikov, O. Colliot, and I. Bloch. Evaluation of the Symmetry Plane in 3D MR Brain Images. *Pattern Rec. Lett.*, 24:2219–2233, 2003. 39

[60] H. Samet and M. Tamminen. Efficient Component Labeling of Images of Arbitrary Dimension Represented by Linear Bintrees. *TIEEE Trans. Pattern Anal. Mach. Intell*, 1988. 40

[61] N. A. Thacker. Tutorial: Functional MRI Analysis. *Tina Memo*, (2001-001), 2002. 49

[62] M. D. Phillips. Brain Perfusion Imaging. *Seminars in Cerebrovascular Diseases and Stroke*, 1(4), 2001. 49

[63] F. S. Acton. *Analysis of Straight-Line Data.* New York, 1966. 53

[64] S. Chatterjee, A. Hadi, and B. Price. *Simple Linear Regression.* Wiley, 2000. 53

[65] A. L. Edwards. *The Regression Line X on Y.* W. H. Freeman, 1976. 53

[66] D. Eberly. *3D Game Engine Design.* 2001. 55

[67] R. R. Edelman, B. Siewert, D. B. Darby, V. Thagaraj, A. C. Nobre, and M. M. Mesulam. Qualitive Mapping of Cerebral Blood-Flow and Functional Localization with Echo-Planar MR Imaging and Signal Targeting with Alternating Radio-Fequency. *Radiology*, 192:513–520, 1994. 58

[68] L. Restrepo, R. J. Wityk, and et al. Ma Grega. Diffusion and Perfusion-Weighted Magnetic Resonance Imaging of the Brain Before and After Coronary Artery Bypass Grafting Surgery. *Stroke*, 33:2909–2915, 2002. 58

[69] K. Kaneko, Y. Kuwabara, and et al. F. Mihara. Validation of the CBF, CBV, and MTT Values by Perfusion MRI in Chronic Occlusive Cerebrovascular Disease: a Comparision with O-PET. *Academic Radiology*, 11(5):489–497, 2004. 59

[70] R. Wirestam, L. Anderson, and et al. L. Ostergaard. Assessment of Regional Blood Flow by Dynamic Susceptibility Contrast MRI Using Different Deconvolution Techniques. *Magn Reson Med*, 43:691–700, 2000. 59

[71] L. Ostergaard, R. M. Weiskoff, and et al. D. A. Chester. High Resolution Measurement of Cerebral Blood Flow Using Intravascular Tracer Bolus Passages. Part I: Mathematical Approach and Statistical Analysis. *Magn Reson Med*, 36:715–736, 1996. 59

[72] D. Bereczki, L. Wei, and et al. T. Otsuka T. Hypercapnia Slightly Rises Blood Volume and Sizably Elevates Flow Velocity in Brain Microvessels. *Physiol*, 264:H1360–H1369, 1993. 60

[73] M. Kalos. *Monte Carlo Methods*. VCH, 2008. 63

[74] T. Beier and S. Neely. Feature-Based Image Metamorphosis. *Siggraph '92*, 1992. 69

[75] A. R. Smith. Planar 2-Pass Texture Mapping and Warping. *Siggraph '87*, 21(4):263–272, 1987. 69

[76] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990. 69

[77] G. Wolberg. Skeleton Based Image Warping. *Visual Computer*, 5(1/2):95–108, 1989. 69

[78] J. Fitzpatrick and J. B. West et. al. Comparision and Evaluation of Retrospective Intermodality Image Registration Technique. *SPIE*, (2710):332–347, 1996. 72

[79] J. Fitzpatrick and et. al J. B. West. Comparison and Evaluation of Retrospective Intermodality image Registration Technique. *Journal of Computer Assisted Tomography*, 21:554–566, 1997. 72

[80] J. Fitzpatrick, J. B. West, and C. R. Maurer. Predicting Error in Rigid-body, Point-based Registration. *IEEE Trans. Medical Image special issue on Image Registration*, 17:694–702, 1998. 72

[81] E. N. Marieb, J. Mallatt, and P. B. Wilhelm. *Human Anatomy*. Pearson Education, Inc., 2005. 73

[82] A. Bochenek. *Anatomia czlowieka*. PZWL, 2006. 73

[83] A. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996. 74

[84] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999. 74

[85] G. W. Snedecor and W. G. Cochran. *Statistical Methods*. Blackwell Publishing, 1989. 81

[86] Hagmann and Jonasson. Understanding Diffusion MR Imaging Techniques from Scalar Diffusion-Weighted Imaging to Diffusion Tensor Imaging and Beyond. *Radiographics*, 26 Suppl 1:205–23, Oct 2006. 81

[87] M. R. Ogiela and R. Tadeusiewicz. *Modern Computational Intelligence Methods for the Interpretation of Medical Image, Studies in Computational Intelligence*, volume 84. Springer-Verlag, 2008. 102

[88] M. R. Ogiela and R. Tadeusiewicz. *Medical Image Understanding Technology, Series: Studies in Fuzziness and Soft Computing*, volume 156. Springer-Verlag, 2004. 102

[89] K. Cwalina and B. Abrams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries*. Microsoft .NET Development Series. 107

[90] B. Wagner. *Effective C#: 50 Specific Ways to Improve Your C#*. Effective Software Development Series. 107

[91] A. Troelsen. *Pro C# 2005 and the .NET 2.0 Platform, Third Edition*. Expert's Voice. 107

[92] C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd Edition. 107

[93] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. 107

[94] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Object Technology Series. 108

[95] J. Lowy. *Programming .NET Components*. 2nd Edition. 109

[96] W. K. Pratt. *Digital Image Processing, 3rd Edition*. A Wiley-Interscience Publication, 2001. 116

[97] R. C. Gonzalez and R. E. Woods. *Digital Image Processing, 2nd Edition*. Prentice Hall Upper Saddle River, 2001. 116

[98] Sound library: http://www.codeproject.com/kb/audio-video/cswavrec.aspx. 118

[99] Video capture library: http://socketcoder.com/. 118

[100] Text chat: http://socketcoder.com/. 118

[101] Ssl: http://www.openssl.org/. 120

[102] http://dicom.nema.org. 122

[103] National Electrical Manufacturers Association. Digital Imaging and Communications in Medicine (DICOM); 18 parts, 2009. 122

[104] National Electrical Manufacturers Association. Digital Imaging and Communications in Medicine (DICOM) Part 3: Information Object Definitions, 2009. 125

[105] S. Selçuk Bayin. *Essentials of Mathematical Methods in Science and Engineering*. John Wiley & Sons, 2008. 129

[106] G. A. Korn and T. M. Korn. *Mathematical Handbook for Scientists and Engineers*. McGraw-Hill, 1961. 129

[107] J. Fauvel, R. Flood, and R. Wilson. *Möbius and His Band*. Oxford University Press, 1993. 134

[108] H. S. M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, 1961. 134

[109] A. V. Oppenheim, R.W. Schafer, and T.G. Stockham Jr. Non-Linear Filtering of Multiplied and Convolved Signals. *Proc. IEEE*, 56(8):1264–1291, 1968. 136

[110] I. T. Young et al. A New Implementation for the Binary and Minkowski Operators. *Computer Graphics and Image Processing*, 17(3):189–210, 1981. 136

[111] I. T. Young and L. J. Van Vliet. Recursive Implementation of the Gaussian Filter. *Signal Processing*, 44(2):139–151, 1995. 136

[112] P. W. Verbeek, H. A. Vrooman, and L. J. Van Vliet. Low-Level Image Processing by Max-Min Filters. *Signal Processing*, 15:249–258, 1988. 136

[113] P. Jung-Me and C. Hui-Chuan. *Fast Connected Component Labeling Algorithm Using a Divide and Conquer Technique*. University of Alabama, 2000. 138

[114] P. F. Preparata and S. J. Hong. Convex Hulls of Finite Sets of Points in Two and Three Dimensions. *Commun. ACM*, 20:87–93, 1977. 142