Technische Universität Kaiserslautern
University of Kaiserslautern

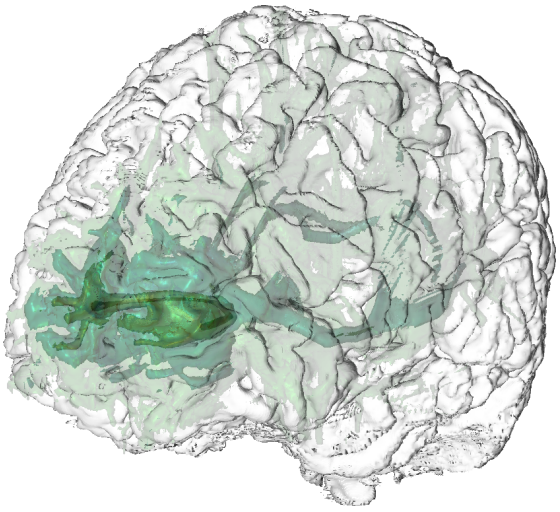
Fachbereich Informatik
Department of Computer Science

Master Thesis

3D In-Context Visualisation for Probabilistic Tractography

by

Andreas Berres
a_berres@informatik.uni-kl.de



First examiner:	Prof. Dr. Hans Hagen
Second examiner:	Juniorprof. Dr. Christoph Garth
Supervisor:	Dipl.-Inform. Mathias Goldau
Handed in:	September 30$^{\text{th}}$ 2011

Abstract

In medical visualisation, the multi-modal display of probabilistic tractograms in anatomical context is a challenging problem. Probabilistic tractograms are scalar fields indicating the probability of connections in the brain; the anatomical context is supplied in the form of structural MRI data. The simultaneous representation of tracts and context leads to difficulties concerning the clear visibility of the respective data. In this work, I present novel approaches aimed at providing a very good depiction of the nature of the tractogram – i.e. its depth, shape, and value distribution – and supplementing it with a meaningful but unobtrusive context. The presented methods improve existing techniques, as, opposed to other volumetric approaches, the nature of the probabilistic tractogram becomes clear in a static image, and the context allows to perceive the cortex structure at the same time as the visualised data inside.

Die multimodale Darstellung probabilistischer Traktogramme in anatomischem Kontext ist ein herausforderndes Problem in der medizinischen Visualisierung. Probabilistische Traktogramme sind Skalarfelder, welche die Wahrscheinlichkeit von Verbindungen im Gehirn aufzeigen. Der anatomische Kontext wird durch strukturelle MRT Daten geliefert. Die gleichzeitige Repräsentation der Trakte und des Kontexts verursachen Schwierigkeiten im Hinblick auf die Sichtbarkeit der einzelnen Daten. In dieser Arbeit präsentiere ich neue Ansätze, die eine sehr gute Darstellung des Traktogramms mit seiner Tiefe, Form und Werteverteilung liefern, und diese durch einen aussagekräftigen, jedoch unaufdringlichen Kontext ergänzen. Die präsentierten Methoden verbessern bestehende volumenbasierte Techniken, da die Eigenschaften des probabilistischen Traktogramms auch in einem statischen Bild klar werden, und der Kontext es ermöglicht, bei der Betrachtung der visualisierten Daten auch die Kortexstruktur wahrzunehmen.

I hereby declare that I composed this work independently. All sources and auxiliary material are indicated in the text, and listed in the bibliography.

Kaiserslautern, September 2th 2025

Acknowledgements

There are a couple of people, without whom this thesis would not have been possible.

Thank you very much...

Hans Hagen, my mentor,
for encouraging me to pursue a great research topic, for fostering my academic career, and for giving me the opportunity to visit my collaborators in Leipzig.

Christoph Garth
for his last-minute advice on academic writing.

All IRTG people
for a pleasant working environment with great colleagues, for gym partners and friends, and for interesting conversations over coffee.

Gerik Scheuermann and the BSV group
for hosting me during my visits, for their technical assistance, and for answering many of my questions.

Mathias Goldau, my supervisor from the BSV group,
for the countless hours he spent listening to my ideas, discussing results, and providing feedback – on the phone, via instant messaging or email, and in person.

My parents
for nurturing my interest in science from the beginning, for raising me to be an independent person, for their continuous support of my studies, and for their interest in this work in particular.

Dagmar
for being my best friend for twenty years, for keeping in touch despite the distance over the last ten years, and for always believing in me.

Til and Betty
for being lovely siblings, for fun discussions over dinner, and for a great sense of humour.

Maxi, my partner,
for enduring my absence, for helping me fight the ~~dragons~~ computers, for always being able to cheer me up, and for keeping me sane.

My friends
for inspiring conversations, for sharing good and bad experiences, and for always being there for me.

Contents

1. Introduction

The main objective of this work is to develop a visualisation with anatomical context, which helps neuroscientists to achieve a better understanding of how the human brain is connected, and which can be used in surgical planning to determine, where cutting would have the least impact on the patient's brain functions. The data they use is acquired with probabilistic tractography. The latter is a technique that generates a scalar field containing the probability, that a given voxel is connected to the seed voxel. Depending on the location of the seed voxel, different probabilistic tractograms can be generated to suit the neuroscientists' needs.

Traditionally, medical imaging data and its context are examined in the form of slices. The main disadvantage of this form of scalar field representation is the loss of information about long-range connectivity between the slices. However, while volumetric representations of probabilistic tractograms preserve this information, they often lack clarity concerning depth and shape of the dataset. A further problem of three-dimensional representations is the choice of values. If all values are displayed, even at low opacity, the main features cannot be visually distinguished from the rest. Therefore, a decision must be made, which values to enhance and which values to mitigate. For the context, the main challenge of volumetric representations lies in neither occluding the data that is visualised, nor distracting from it. At the same time, however, the context should not be occluded by the visualisation or vanish from view.

Medical visualisation improves medical diagnostics and gives a better understanding of human anatomy. Most medical imaging data is noisy and has a rather low resolution of at most one millimetre per voxel, which is too coarse to represent anatomy at cellular level. The data consists of one or more slices, which can be aligned to form a grid containing values at each vertex. Depending on the structure, one obtains a scalar field, vector field, or tensor field.

In addition to visualising a dataset, some form of context must be given to emphasise where the data belongs anatomically. Structural Magnetic Resonance Imaging (MRI) is a very common non-invasive in vivo imaging technique, which is used to acquire images of body parts that require a good differentiation between various types of soft tissue, e.g. the gastrointestinal tract, or the knee. They are also a good basis for brain data, since they provide a good contrast between grey matter and white matter.

My goal for the visualisation is to inform about long-range connectivity while preserving structure and a sense of depth. In order to avoid visual clutter due to too many values, I represented the tractograms as nested, semi-transparent isosurfaces. While it is difficult to understand the dataset structure from just one isosurface, using surfaces for several different isovalues helps to get an overview. Since probabilities convey a sense of significance, where high probabilities are considered more important than low ones, I decided to vary the transparencies of the surfaces depending on the probabilities. Finally, the sense of depth is

obtained by varying the saturation dependent on the depth. Thus, surface parts in front are highly saturated whereas those towards the back are more and more desaturated. This also helps to enhance the surface shape, since the tractograms have a lot of small details, which are easier to separate visually, if their colours have different saturations.

The context must be fundamentally meaningful but non-distracting. I chose a "glass brain" to give anatomical context, which is rendered depending on the surface normal in relation to the view direction. This produces a rendering that is transparent and white on the gyrii (ridges) and opaque and dark in the sulcii (folds). To keep the balance between the visualisation and the context, I introduce a focus-dependent transparency, which can be adapted interactively by the user. As pointed out earlier, the traditional way to analyse medical image data is to study single slices. Since these provide an important clue to the brain anatomy, I integrated them into the context. To set a more specific focus on one region of the visualisation, part of the context can be made opaque in order to reduce the distraction by the other regions.

Overall, the approach I present is very flexible. On the one hand, a flexible environment is vital, since different datasets may need different sets of parameters, which cannot be provided in a static environment. On the other hand, this flexibility also enables the user to adapt the visualisation and its context as required for a task, and to explore the data interactively.

This thesis is structured into several chapters. In Chapter 2, I introduce the medical image acquisition techniques, which were used to acquire the data, before I define some medical terms and finally, give an overview over related work on the topic. Chapter 3 and Chapter 4 discuss the conceptual aspect of the methods I developed to solve the given problem. The implementation of these techniques is described in Chapter 5. Some supplementary excerpts of the implementation are available in Appendix B. Chapter 6 summarises the work and gives insight into possible future works on the subject. In Appendix A, the data is described.

2. Background

Before describing my own work, I first explain some imaging techniques that are commonly in use to acquire the data which is visualised. Then, some medical terms are introduced. And finally, I give an overview over related work.

2.1. Imaging Techniques

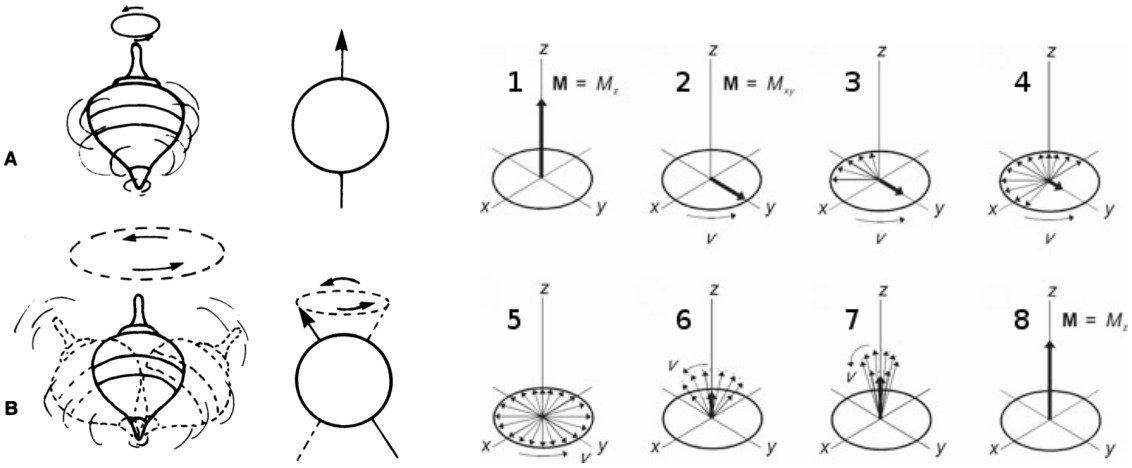
There are many different medical imaging techniques. The most prominent in vivo techniques are Computed Tomography (CT), several variants of Magnetic Resonance Imaging (MRI) and Ultrasound [5]. A large number of techniques falls into the category of MRI, since it does not require to expose the patients to radiation (as opposed to CT which uses X-rays) and it produces reasonably large images that are less susceptible to noise than those acquired with Ultrasound. In the following sections, I introduce three MRI-related techniques, namely MRI itself, Diffusion Tensor Imaging (DTI), and tractography.

2.1.1. Magnetic Resonance Imaging (MRI)

The human body is composed of about 60% of water. Different tissues of the body contain different amounts of water. An imaging technique focusing on hydrogen produces images that have a very good contrast between different kinds of soft tissue, since they lead to different intensities (represented in the images as different shades of grey).

The neutrons and protons in an atom's nucleus are continuously in motion. They spin around their axis with a magnetic moment, which aligns with an external magnetic field. For atoms that have an even number of protons and neutrons, the magnetisation cancels out. Hydrogen has a single proton and therefore generates a small magnetic field. This field can be represented by a vector along the spin axis, as seen in Hendee et al.'s illustration (Figure 2.1a).

Since the brain consists of tissue structures with different amounts of water, one can use MRI to acquire images. It is common to align the external magnetic field M_z along the z-axis with $M_{xy} = 0$, i.e. there is no rotation in the xy-plane. If a pulse of the right frequency is applied to the field, the magnetic moment is tilted and begins to precess. This means, that it rotates around its original axis in a gyroscopic fashion in order to return to the original alignment. This is called *precession* and it is depicted in Figure 2.1a. The rotation causes the emission of a radio signal at the same frequency as the precession, which is unique for each type of atom. This also implies that all hydrogen atoms rotate with the same frequency.



(a) Precession. (b) Relaxation.

Figure 2.1.: The impact of radio pulses on hydrogen atoms. 2.1a: A radio pulse of the right frequency causes precession, Hendee et al. [11, Fig. 2]. 2.1b: Following the radio pulse, the atoms undergo two phases of relaxation, Deserno [5, Fig. 1.9].

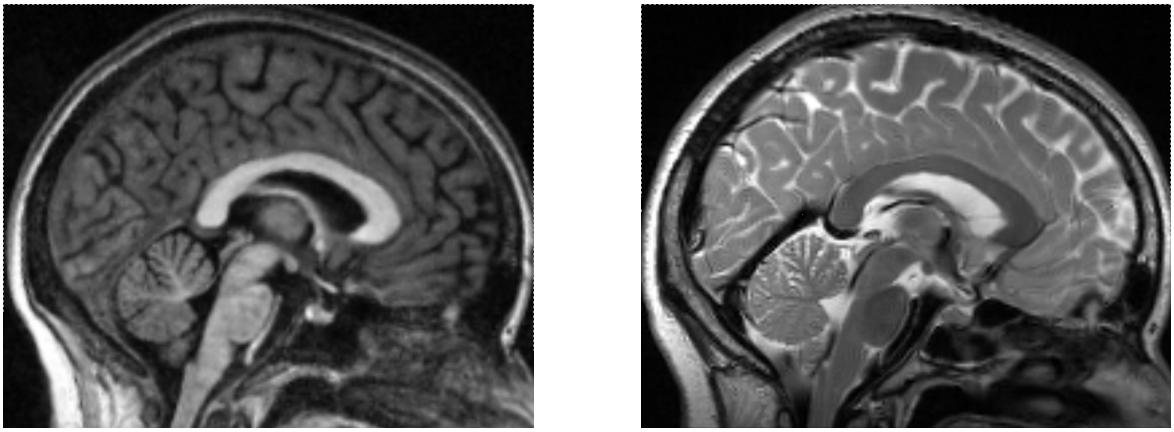
Deserno's illustration in Figure 2.1b visualises the process that takes place. At first, the magnetic field is aligned with the z-axis as seen in time step 1. Then, the MRI is started using a radio frequency (RF) impulse to excite the spinning protons. In time step 2, there is a 90° RF impulse, which turns the magnetic field into the xy-plane where it rotates with precession ν. After this impulse, the system returns to its equilibrium state during *exponential relaxation*. It consists of two independent phases:

1. During the *spin-spin relaxation* in time steps 2 to 5, the precession breaks apart through dephasing and fans out in all directions. The intensity of the radio signal decreases with the separation of the magnetic moment into components. The relaxation time T_2 ranges from 40 – 200 ms for water-based tissues.

2. During the *spin-lattice relaxation* in time steps 5 to 8, the precession realigns with the original magnetic field until the system reaches its equilibrium state. The intensity of the radio signal decreases with the shrinking angle between the magnetic fields. The relaxation time T_1 ranges from 0.4 – 1.2 s for water-based tissues.

During the relaxation, energy is released as photons. The photons are detected as an electromagnetic signal and are then transformed into an image. Since different tissues return to the equilibrium state at different speeds, depending on the hydrogen saturation, the amount of energy varies spatially. To obtain images that have a high quality, sequences of different RF impulses are used and both T_1 and T_2 are measured.

There are three main classes of MR images, which all combine the relaxation times T_1 and T_2, and the proton density M_0 with different weights. The data set used in this work (cf. Appendix A) consists of T_1-weighted images that were created using *inversion recovery*, which consists of a two-pulse sequence. First, an inverting pulse rotates the magnetic field by 180° from $+z$ to $-z$. After a fixed time τ, during which the field relaxes back towards +z, the



(a) T_1 image. (b) T_2 image.

Figure 2.2.: Two images from an unpeeled dataset of the test subject, so the skull and skin are visible. Appendix A describes the test subject dataset. In T_1 images, water is darker and fat is brighter. In T_2 images, the opposite is the case, so fat is darker and water is brighter.

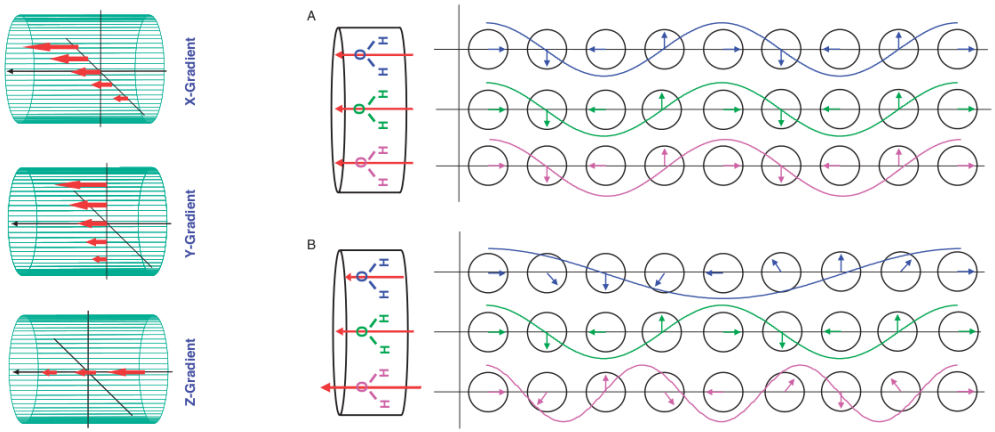
second pulse occurs. This time, a 90° pulse is used and the signal is read immediately after the pulse. The resulting image is T_1-weighted, because the relaxation after the first pulse is a spin-lattice relaxation and thus depends on T_1. With different sequences of pulse angles and waiting times, one can also produce M_0-weighted and T_2-weighted images. Figure 2.2 shows a comparison between a T_1 and a T_2 image.

The MR image is obtained slice-wise. To select a single plane, a gradient is applied to the magnetic field, centred around the plane of interest. This causes the atoms outside the plane to speed up or slow their precession frequency. The intensity at each voxel in the slice of interest can be retrieved using two-dimensional Fourier Transform and selecting the original frequency.

More in-depth information about MRI can be found in Deserno's Biomedical Image Processing textbook [5] and Hendee et al.'s MRI paper [11].

2.1.2. Diffusion Tensor Imaging (DTI)

The signal-transmitting part of a neuron (axon) can be over one metre long, but is only one micrometre thick. Diffusion Tensor Imaging (DTI) can only provide a resolution of about two millimetres per voxel, which is obviously a different order of magnitude. However, axons are known to align in parallel into major fibre bundles, which are large enough to be detected. Diffusion is much more likely to occur inside each cell (along the axon) than through the cell membrane and the Myelin sheath, which is a layer around the fibre that acts as an electrical barrier. Hence, a common assumption is, that there is more diffusion along a bundle than perpendicular to a bundle, as long as there are no complex bundle configurations like crossings.



(a) Gradients along axes.

(b) Homogeneous (top) and gradient (bottom) field.

Figure 2.3.: The gradients are applied along the axes (2.3a). In a homogeneous field, all protons spin with the same frequency and phase, but in a gradient field, their frequency (and phase) varies depending on the field strength in their location. Illustrations by Mori [17, Fig. 1.4/1.8].

The fundamental idea of Diffusion Tensor Imaging (DTI) is to measure diffusion of water molecules in different directions. They have to be fixed, and evenly distributed on a sphere, in order to avoid artefacts and estimation errors [14]. Along the axes of the different directions, gradients are applied, i.e. the strength of the magnetic field varies linearly, as displayed in Figure 2.3a. There are various sources of error in DTI, e.g. noise, patient motion, blood diffusion, or Eddy currents.

As explained in Section 2.1.1, protons spin with different frequencies depending on the magnetic field strength. Figure 2.3b demonstrates the difference between a homogeneous and a gradient magnetic field. All protons spin with the same frequency in a homogeneous magnetic field. Once a gradient is applied to the magnetic field, their frequencies vary with the field strength.

This effect can be used to determine the diffusion along the axis as depicted in Figure 2.4. At first (t_1), the magnetic field is homogeneous and all protons spin with the same frequency and phase. During *dephasing* (t_2), a gradient is applied to the field. As a result, the protons in different positions in the field spin with different frequencies. Afterwards (t_3), there is another time period with a homogeneous field. Now all protons spin with the same frequency again, but their phases are shifted due to the different frequencies in the previous step. Finally, the inverse gradient of the one used in t_2 before is applied during *rephasing* (t_4). Protons that spun more slowly during dephasing, spin more quickly now and vice versa. The rephasing ends, when all protons face in the same direction, and the magnetic field is made homogeneous again. The example in Figure 2.5 illustrates, why diffusion changes the outcome.

The first example shows, what happens for a coherent motion, e.g. patient movement during the scan. Dephasing changes the phase, all molecules move on in some direction, and rephasing

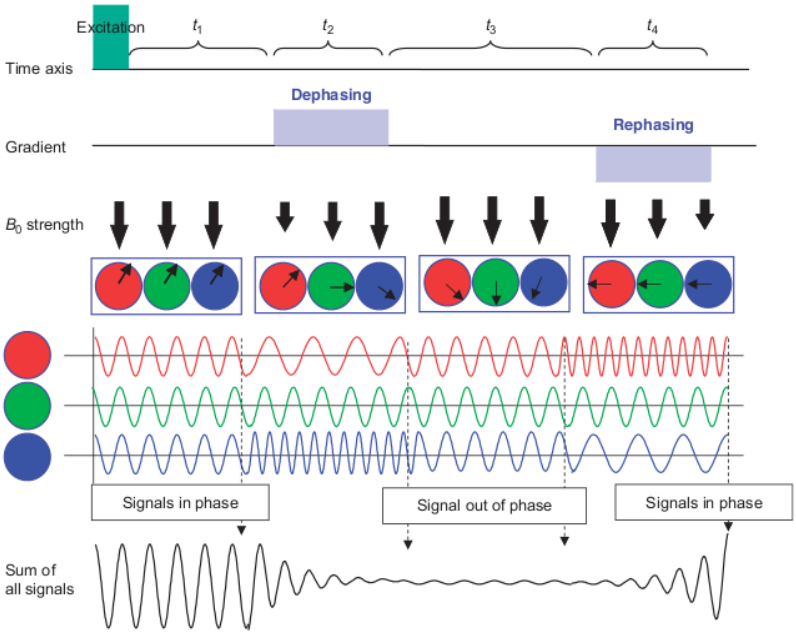


Figure 2.4.: Phase disruption illustrated by Mori [17, Fig. 1.9]. During dephasing and rephasing, opposite gradient fields are applied.

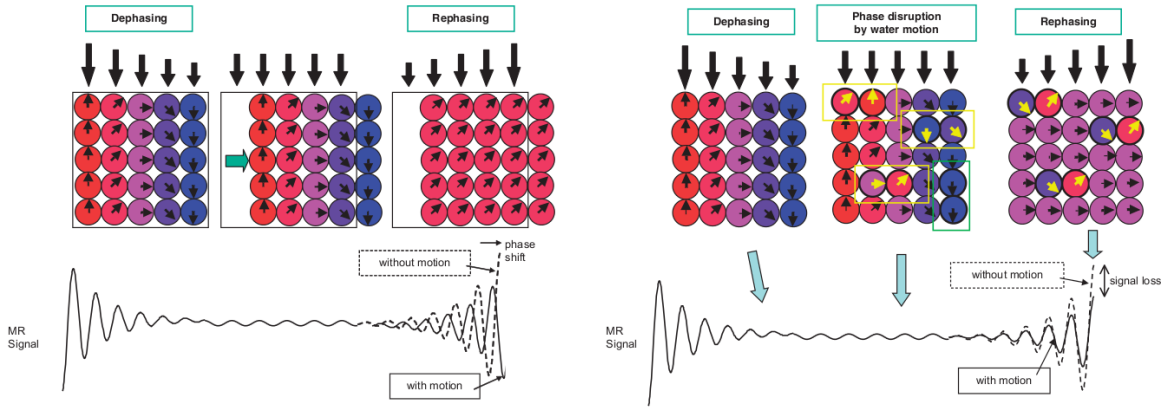
turns all protons back into the same direction. As the drawing of the signal shows, the signal is merely shifted.

In the second example, the movement is not coherent. Instead, there is some diffusion, i.e. the protons in yellow boxes swap their positions. During rephasing, the "wrong" gradient is applied to the protons that have changed their position. This can be registered in the scan as signal loss, as seen in the drawing of the signal.

This signal is recorded from different directions in order to reproduce the three-dimensional value distribution. Ideally, the gradient directions are distributed evenly on a sphere. Depending on the number of gradient images (e.g. over 50 for HARDI-like measurements), one can estimate tensors of different orders, which contain the amount of diffusion in a different number of directions. If only a small number of gradient directions are available, one can only estimate a second order tensor. The resulting data has a different orientation and resolution than the MRI data.

In order to be able to combine the DTI data with structural MRI context, the diffusion data is registered to MRI space. This is achieved using a linear transformation, which can be represented by a matrix. If data derived from DTI, such as probabilistic tractograms, has to be registered, it is usually derived first in order to avoid artefacts. Afterwards, the linear transformation is applied to the derived data.

To register the DTI resolution to the structural MRI, the intermediate values are interpolated. Rotation, distortion, and scaling can be represented by matrix multiplication. Once the matrix is computed, it can be applied to any dataset derived from the DTI data.



(a) Coherent movement.

(b) Diffusion.

Figure 2.5.: For a coherent movement, the process causes all protons to be in the same phase after rephasing. If there is diffusion, the protons that changed their position with respect to the other protons are rephased incorrectly and are therefore out of phase after rephasing. Illustrations by Mori [17, Fig. 1.10].

Additional information can be found in Mori's textbook about DTI [17].

2.1.3. Tractography

Tractography is a procedure to measure the long-range connectivity of major fibre bundles. There are two main approaches, local and global tractography, which can be executed in a deterministic or probabilistic fashion, with more or less complex models of the diffusion process.

Deterministic or *streamline tractography* generally refers to local deterministic tractography. A seed point is selected and the streamline is constructed from local vector information. The vector field contains the eigenvector corresponding to the largest eigenvalue, i.e. the highest amount of diffusion at a given point. Between grid points, the direction is interpolated. There are three major sources for error: imaging noise, modelling error (fibre bundle crossings cannot be represented in the field), patient movement, and integration error. The integration is stopped, if the fractional anisotropy, i.e. the disparity of the magnitudes of the eigenvectors, becomes too small (high uncertainty), or if the curvature of the streamline becomes too large (anatomically implausible).

Probabilistic tractography also refers to the local variant. It addresses the problem of uncertainty, which arises due to patient movement, modelling error, and image noise. For a probability x at voxel B in the probabilistic tractogram, one can assume that

> [given] the model and the data, [one can have] x% confidence that the path of least hindrance to diffusion from seed point A passes through [voxel] B.
> (Behrens et al. [1]).

Instead of stopping in regions of high uncertainty (i.e. low fractional anisotropy), the direction is decided probabilistically. There are different ways to obtain the probabilistic tractogram. One example is described below.

Uncertainty is characterised by an uncertainty orientational density function (uODF), which is derived from a group of datasets. This function can be computed in different ways, e.g. using bootstrapping (extraction from a small number of datasets) or Bayesian models.

To calculate the probability that a given voxel is connected to the seed voxel, every possible path has to be considered. This can, for example, be done by starting the path at the seed voxel and moving along orientations sampled from the uODF. Since computing all possible paths is computationally expensive, only a manageable number of paths is sampled. In the end, the probability can be calculated as

$$p(\text{voxel}) = \frac{\text{number of paths through voxel}}{\text{number of all paths}}$$

An example of a tractogram is displayed in Figure 2.6.

(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure 2.6.: The probabilistic tractogram. Voxels that have probability zero, are clipped, and the values are interpolated. The colour map used is a heat map, where the regions of high probability are white or yellow, and those with low probability are red or dark red.

It is important to note that a probabilistic tractogram does *not* represent the distribution of connections from the seed point. Instead, it only represents the confidence for a single path.

Global tractography, takes global connectivity into account. As before, deterministic global tractography generates paths, whereas probabilistic global tractography generates a volume. The two main ideas are optimisation of connectivity information or paths, and search for geodesic paths.

Behrens et al. [1] described the methods to acquire probabilistic tractograms in great detail.

2.2. Anatomy

Since this work focuses on the visualisation of human brains, I will introduce some terminology.

2.2.1. Directions and Planes

To introduce a coordinate system, which is independent of the patient's pose, the following directions are used to describe anatomical relationships and orientation (cf. [19] and Figure 2.7).

Ventral and dorsal refer to the abdomen and back side of the body. In a standing human, these are front and back.

Superior and inferior refer to the head and feet side of the body. However, since humans walk upright, these directions would be identical to ventral and dorsal.

Left and Right (or sinister and dexter) refer to the left and right side of the body (with respect to the person, not to the viewer!).

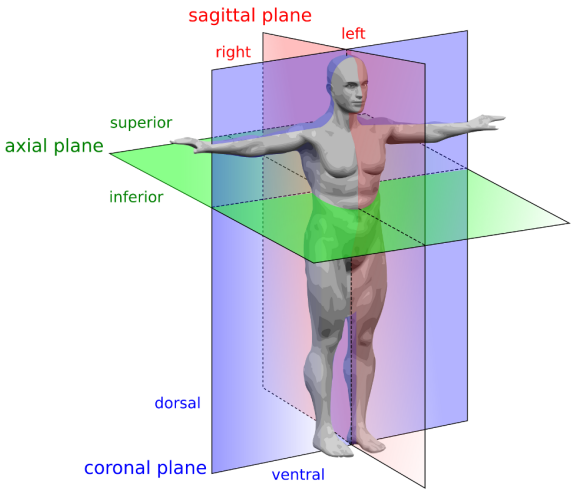


Figure 2.7.: The anatomic directions and planes, by Mrabet [18]

To describe the anatomical orientation of actions or slices, the following terms are used:

The **axial plane** (or transverse plane) divides the superior and inferior side.
The **coronal plane** divides the ventral and dorsal side.
The **sagittal plane** divides the left and right side.

2.2.2. Anatomy of the Human Brain

The human brain is divided into lobes by gaps called *fissures*, e.g. the medial longitudinal fissure, which separates the brain into a left and a right hemisphere, and the lateral cerebral

fissure, which separates the temporal lobe from the rest. The outermost layer of brain tissue is called the *cortex*. Furthermore, the brain obtains its unique structure from the gyrii and sulcii. A *gyrus* is a ridge on the surface of the brain, whereas a *sulcus* is a valley. The most prominent structures of the brain are labelled in the following illustration from Gray's anatomy textbook [10]:

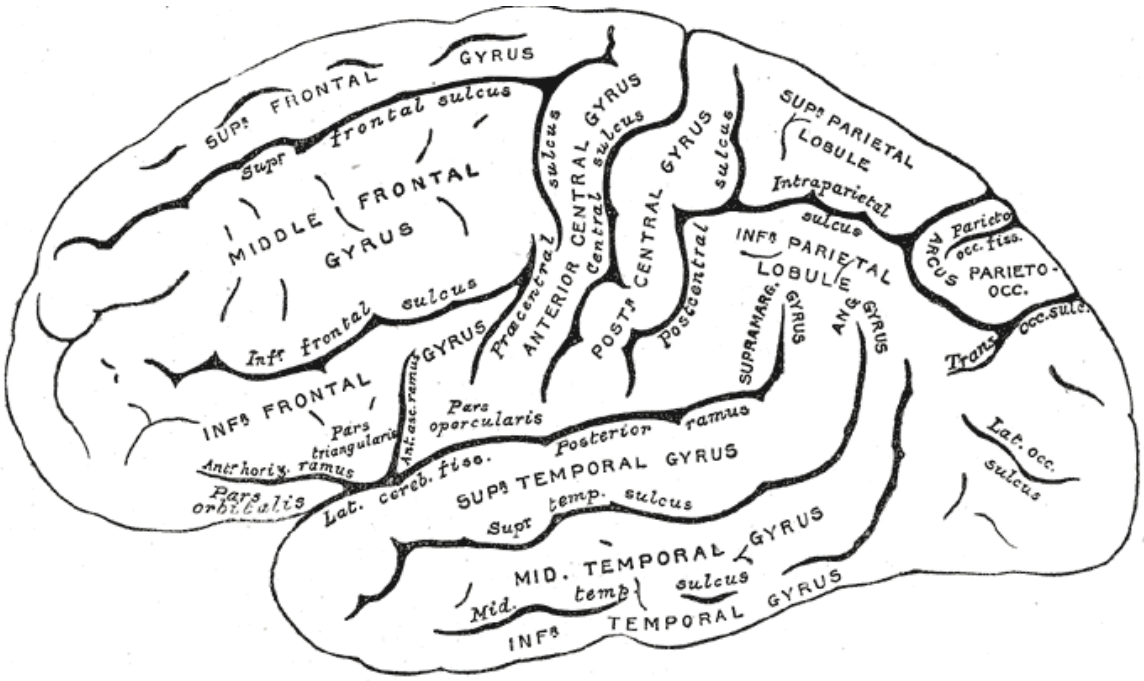


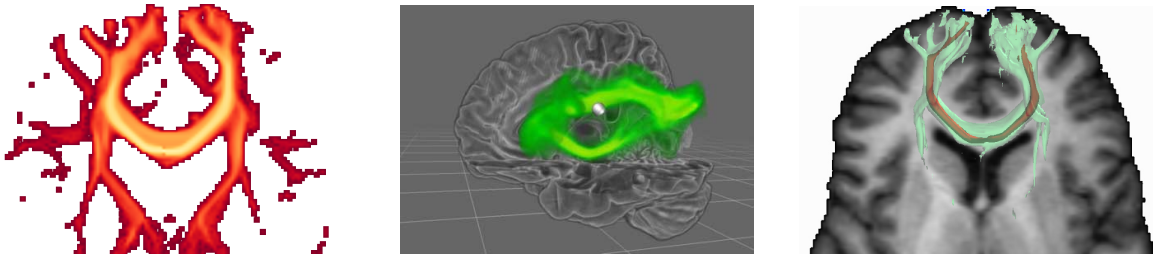
Figure 2.8.: The most important fissures, gyrii and sulcii of the human brain. [10, Fig. 726]

2.3. Related Work

In the following, I discuss previous work, that relates directly to the presented methods in the areas of probabilistic tractograms and context visualisation. This thesis is based on various previous works. First of all, probabilistic tractograms have to be visualised. Second, there needs to be some context. Last but not least, general visualisation concepts should not be ignored. In general, there is a large amount of work on the visualisation of fibre pathways from DTI data; I refer the interested reader to the overview by Preim et al. [20].

2.3.1. Visualisation of Probabilistic Tractograms

There are three main approaches to visualising probabilistic tractograms: slice-based, direct volume rendering, and isosurfaces.



(a) Slice-based visualisation.

(b) Direct volume rendering by Rick et al. [21].

(c) Isosurfaces by Goldau et al. [8].

Figure 2.9.: Slice-based (2.9a), direct volume rendering (2.9b) and isosurface (2.9c) approaches to visualising probabilistic tractograms.

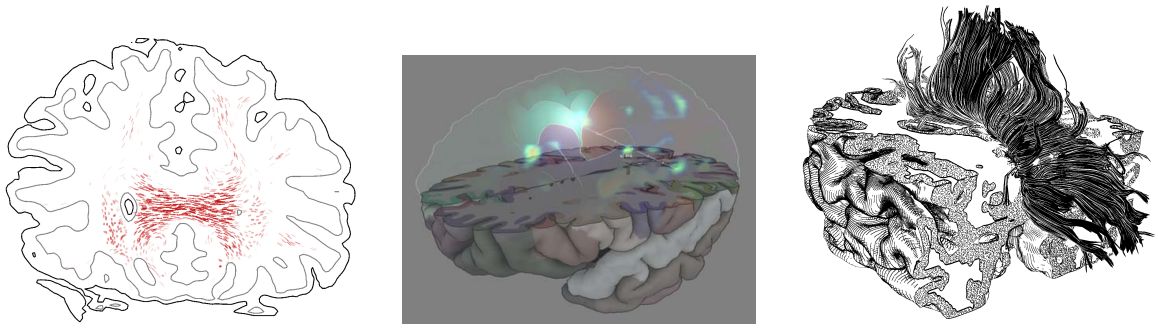
Traditionally, textured slices are used to view and navigate imaging data. Since the source images usually have a very low resolution, the constantly coloured voxels imply edges that do not exist in the real data. This can be cured by interpolating the data. While the original data may have sharp edges, they usually do not concur with the voxel borders. If the original data has sharp edges, they may be lost due to smudging. Otherwise, no information is lost during interpolation.

Rick et al. [21, Fig. 2] employed a *direct volume rendering* approach, where they mapped the probability to colour and opacity. Low probabilities have darker, more transparent colours, whereas high probabilities have brighter, more opaque colours. For multiple fibre tracts, it is mixed in a way that crossing fibres add up to a brighter colour. The visualisation focuses on displaying several tractograms at the same time. In addition, Rick et al. also provide context. It consists of slices and an opaque rendering of the brain with a cut-away. While the brightness of the context competes with the brightness of the probabilistic tractogram visualisation, the main problem lies in the fact, that a reasonable spatial impression is difficult to realise in a static image. This problem can be alleviated to some extent, if the user can interact with the visualisation (e.g. through rotation or zooming), or explore it in a virtual reality environment.

Instead of applying direct volume rendering, other techniques make use of nested isosurfaces, such as the one presented by Goldau et al. [8, Fig. 3]. They try to mitigate the problem of choosing an adequate isovalue by drawing several surfaces. However, they conclude that it is difficult to provide reasonable anatomical context besides slices.

2.3.2. Visualisation of Anatomical Context

Anatomical context for brain data is usually presented in the form of slices, by a semi-transparent "glass brain", or by cut-away views of an opaque brain.



(a) Slice-based context by Goldau et al. [8].

(b) "Glass brain" by Jainek et al. [13].

(c) Cut-away view by Svetachov et al. [23].

Figure 2.10.: Slice-based (2.10a), "glass brain" (2.10b), and cut-away views (2.10c) as context.

A good example of *slice-based* context for probabilistic tractograms is provided by Goldau et al. [8, Fig. 3]. They do a grey and white matter segmentation on MRI slices. A T_1 image can provide more accurate information, however, it also produces visual clutter. Then, they draw fibre stipples of different shape and opacity to illustrate the diffusion direction. The shape represents the fibre direction: it is circular if the fibre lies orthogonal to the slice, and it is long and thin if it lies in the slice. The probability influences the opacity and density: a higher opacity or more densely placed stipples signify a higher probability. While the visualisation provides a very good sense of local connectivity, the impression of long-range connectivity is rather hard to obtain due to the nature of slices.

Jainek et al. [13, Fig. 14] and Born et al. [3] introduce a visually appealing "glass brain" context for brain activity information from functional MRI (fMRI). This data is acquired by presenting the patient with passive (visual, acoustic, or haptic) and active stimuli (tasks, e.g. finger tapping), and observing the brain activity. They assign different dull colours to different gyrii of the pre-segmented brain. One part of the brain is then clipped away, and the cutting plane is textured with an illustrative grey and white matter rendering. Above the clipping plane, the silhouette of the cortex is shown. Near the focus area, they display fMRI information by drawing a semi-transparent colour-coded hull. Their context does not distract from the information of interest, since the only bright, colourful part is the fMRI data. Since the information is integrated seamlessly into the context representation, the hull of the brain is left empty. However, if more information has to be displayed, the mapping to the "glass brain" surface is cluttered and therefore, it is difficult to understand the visualisation.

Another illustrative approach is presented by Everts et al. [7] and Svetachov et al. [23, Fig. 1]. They create line drawings of the brain, using hatching to create a shading for the surface, and employing stippling to contrast the grey matter against the white matter. This is combined by rendering a cutting plane with stipples and drawing part of the brain's cortex. Their context is given for deterministic fibres, which they surround with a halo depending on the immediate neighbourhood. By restricting the fibres that are displayed to those crossing a user-specified region of interest, they manage to avoid visual clutter and draw the viewer's attention to the fibres rather than the context. The context is suitable for deterministic fibre tracts, which can be displayed as lines. But probabilistic tractograms, which are strictly volumetric, cannot easily be represented by a pure line drawing.

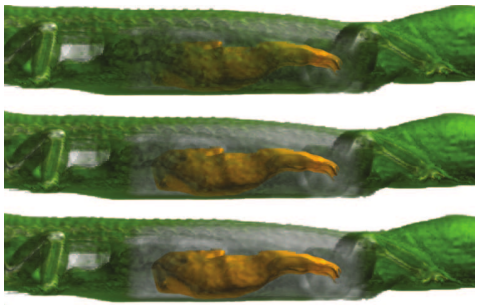
2.3.3. General Visualisation Techniques

Not all visualisation techniques are specifically designed for scalar fields or brains. There are interesting approaches to importance-driven visualisation, multilayer surfaces from flow visualisation, tensor field visualisation and optimisation for choosing isovalues.

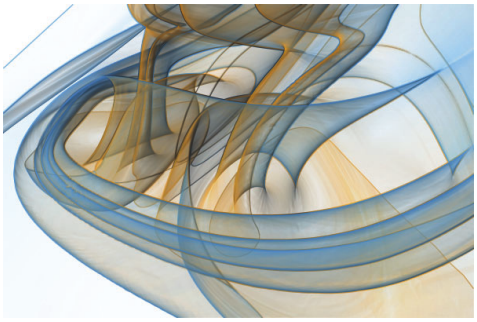
Whenever a lot of information has to be visualised, there is a high chance of visual clutter. To avoid this clutter, one needs to take the importance of different elements into account, as Viola [24, Fig. 4.8] did in his PhD thesis on importance-driven visualisation. While he is not specifically concerned with tractography or brain visualisation, he makes use of volume rendering techniques to visually separate objects of interest from the context. Viola evaluates different representations for context that lies in front of the most important object. He compares different levels of sparseness for a combination of opacity and colour saturation modulation, screen-door transparency and volume thinning. His techniques are based on pre-segmented data and use CPU-based rendering. Since the brain has a lot of structure, and probabilistic tractograms have very small details, applying screen-door transparency or volume thinning can result in confusing representations.

Hummel et al. [12, Fig. 3] introduced new techniques to render multilayer surfaces. While their focus is on integral surfaces from flow visualisation, their techniques can also be applied to nested isosurfaces. In particular, they vary the transparency depending on the angle between the surface normal and the view vector, or depending on an image space approximation of the curvature. This creates high opacities near the boundary of a surface and low opacities elsewhere. Additionally, they use adaptive textures that change density in texture space and appear constant in image space. Since their method is aimed at integral surfaces, it needs to be adapted if used for medical data.

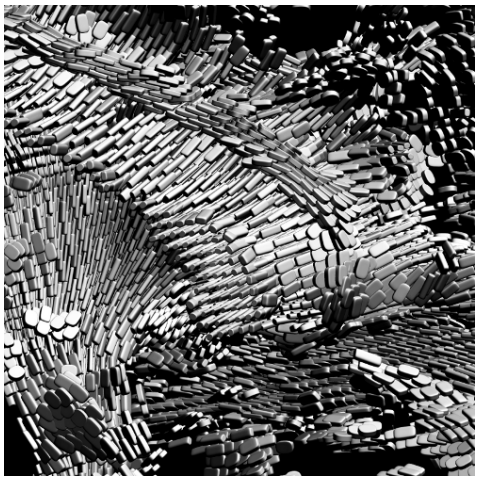
In order to display not just one eigenvalue but a whole diffusion tensor of a DTI dataset, one can use *glyph-based representations*. Kindlmann [15, Fig. 11] uses superquadric glyphs to visualise tensor fields based on the anisotropy. This conveys a very good sense of shape and diffusion direction. However, the inner three-dimensional structure cannot be seen properly because the glyphs in front occlude most of what lies beyond. Another problem with glyphs is, that large ones may dominate a scene even if they are not the object of interest, and it is rather hard to provide context.



(a) Importance-driven visualisation by Viola [24].



(b) Multilayer integral surfaces by Hummel et al. [12].



(c) Tensor field visualisation by Kindlmann [15].

(d) Optimisation of isovalue choice by Bruckner et al. [4].

Figure 2.11.: Importance-driven visualisation (2.11a), multilayer surfaces (2.11b), tensor field visualisation (2.11c), and optimisation of isovalue choice (2.11d).

Drawing isosurfaces requires the user to pick an adequate isovalue. Bruckner et al. [4, Fig. 3] developed a method that automatically determines ideal isovalues. This is achieved by selecting the isovalues, which are most representative of the whole dataset, with the restriction that the chosen values are as dis-similar to each other as possible. The selection is based on histogram analysis and is computationally expensive. Therefore, the computation of the isosurface similarity map is not possible at interactive speed.

3. Probabilistic Tractogram Visualisation

Traditionally, probabilistic tractograms are represented as slices, as detailed in Section 2.3. While medical staff are trained to understand data from such representations, separate slices provide very little information about long-range connectivity. Therefore, an understandable three-dimensional representation is desirable.

Goldau et al. [8] tried to obtain a three-dimensional representation of a probabilistic tractogram using nested isosurfaces combined with an MRI slice. However, they felt that they could not provide an adequate three-dimensional context for the data. In Section 3.1, I discuss the techniques needed to provide a nested view of the isosurfaces as seen in their work. Based on this approach, I developed two new techniques to make the topology of the dataset more easily perceivable for the viewer:

The first new technique adapts the transparencies of the nested surfaces to the dataset. I use the probability (or isovalue) of the probabilistic tractogram for the given surface. This enhances the sense of significance of a given value or surface. The technique is explained in Section 3.2.

The second new technique I developed changes the surface saturation depending on the depth of a given point on the surface. If the saturation of all points on a surface is uniform, it is difficult to distinguish surface parts in front from those further back, due to the high amount of folding and detail in the probabilistic tractogram. Using high saturation close to the viewer and low saturation further away results in a better understanding of the surface geometry. Section 3.3 outlines this method.

3.1. Nested Isosurfaces

Given an isosurface, i.e. a surface consists of all points that have the same value (or isovalue), it is possible to achieve a first impression of a dataset. However, even if the user can manipulate the isovalue with a slider to explore the dataset, it is difficult to obtain a good impression. The restriction to one value results in a loss of information since the topology does not become clear. Moreover, to explore a dataset using just one surface, a user has to look at the dataset from different sides. But a good visualisation should be able to convey the main features of a dataset in a single image.

To address this problem, one can draw multiple isosurfaces instead of one. This does not address the problem of choosing representative isovalues, but it is possible to identify a rough estimate of the values in the dataset.

Isosurfaces are computed using a ray caster that stops once it has found the first match for an isovalue. This is achieved by starting a ray at each pixel and stepping through the volume until a point on the ray has the correct isovalue. The surface features are enhanced by using OpenWalnut's Phong shading function. One can also use a complete ray caster, i.e. one, which traverses through all points on the ray. However, this necessitates the use of a transfer function which decides for each pixel, which voxels along the ray are visible. An entirely different approach would be to extract the isosurface geometrically, e.g. using the Marching Cubes algorithm. But it takes more time to compute a transparent isosurface, since every time, the view is rotated, the triangles have to be ordered by depth. Therefore, the program would be too slow to display the surfaces interactively.

These nested isosurfaces need to be composited. This can done using a standard back-to-front compositing method like the one in Bender et al.'s book [2], where $c = (r, g, b)$:

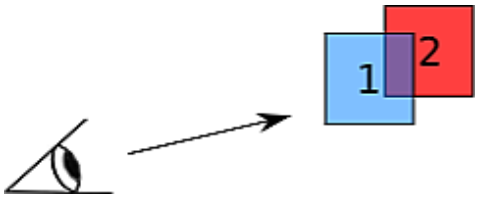


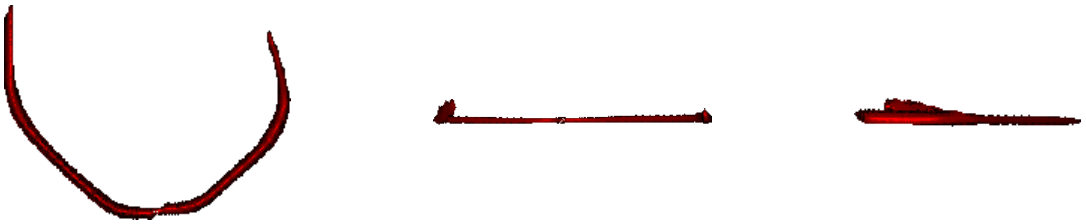
Figure 3.1.: Compositing for two objects: The blue object has $c_1 = (0.0, 0.5, 1.0)$ and $\alpha_1 = 0.5$. The red object has $c_2 = (1.0, 0.0, 0.0)$ and $\alpha_2 = 0.75$. They add up to a rather opaque purple with $c' = (0.375, 0.25, 0.5)$ and $\alpha' = 0.875$.

For two objects with colours c_1 and c_2 and transparencies α_1 and α_2, the composited colour c' and transparency α' are computed as

$$c' = \alpha_1 \cdot c_1 + (1 - \alpha_1) \cdot \alpha_2 \cdot c_2$$

$$\alpha' = \alpha_1 + (1 - \alpha_1) \cdot \alpha_2$$

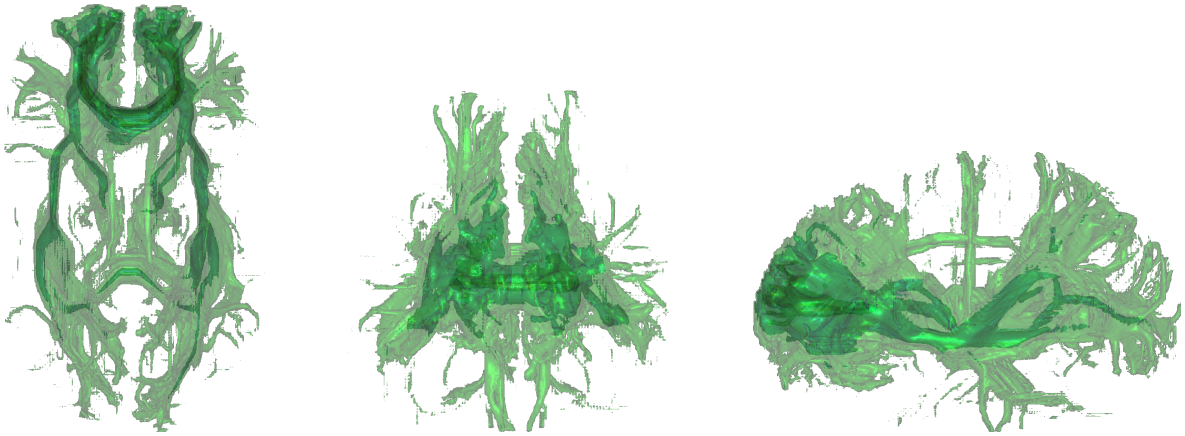
An example is provided in Figure 3.1. In practice, the second object is an image of the isosurfaces that has already been computed in the fragment shader. Then, another layer containing the new isosurface is added in front.



(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure 3.2.: A single surface cannot give an adequate impression of the dataset. The one presented here has the probability 0.8 and is opaque.

The nested view in Figure 3.3 gives a slightly better impression of the dataset than the single isosurface in Figure 3.2 could give. However, it is not clear, how probable each surface is.



(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure 3.3.: Nested surfaces give a better impression of a dataset. The nested surfaces have the probabilities $(0.8, 0.6, 0.4, 0.2)$ and a uniform transparency of 0.5.

First of all, a set of isovalues has to be chosen. In this thesis, this was done manually by exploring the dataset and choosing a set of representative isovalues. But this can also be done automatically as described by Bruckner et al. [4]. The second aspect one has to consider is the number of surfaces. While a single surface cannot give an adequate overview over the data, a large number of surfaces leads to visual clutter. Depending on the nature of the dataset, up to four surfaces can be distinguished visually.

3.2. Adaptive Transparency

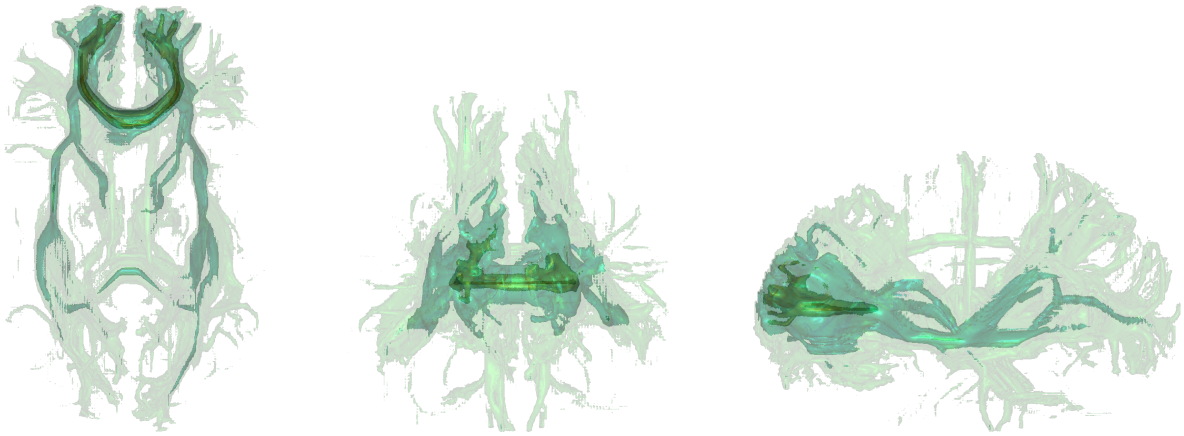
In order to improve the visual impression of the probability distribution, it makes sense to use different levels of transparency for different probabilities. Surfaces that are further inside the body should be more opaque than peripheral surfaces. In addition, the probability should be taken into account for more than just the isovalue. As seen in Figure 2.6, the probabilities grow towards the centre of the shapes.

It is possible to combine both ideas by varying the transparency as a function of the isovalue. For each of the picked isovalues $x_i \in I = [a, b]$, the transparency for x_i can be computed as follows:

$$\alpha_i = \frac{x_i - a}{b - a} \in [0, 1]$$

Instead of using the automatically computed transparencies, it is also possible for the user to pick custom alpha values via the Graphical User Interface (GUI).

With the varied transparency in Figure 3.4, it is a lot easier to see, which parts are more and which are less probable (or important). However, the high transparency and the strongly folded shape of the surfaces make it hard to get a good impression of the depth and the surface shape.



(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure 3.4.: The adaptive alpha makes the inner layers more visible. The nested surfaces have the probabilities and transparencies $(0.8, 0.6, 0.4, 0.2)$.

3.3. Focus-Dependent Saturation

Emphasising the part in focus and de-emphasising the other parts of each surface can help to improve the visual impression of such a dataset. To achieve this goal, the colour is desaturated depending on the distance of the surface from the eye point.

First, an appropriate grey value of the colour is approximated from the single colour channels. In intuitive, hue and saturation-based colour models, i.e. HSV, HSL and HSI, the third component (value, lighting or intensity) corresponds to the overall intensity of a pixel. While the intensity of all three models represents the grey value of a pixel, the way they are computed differs between the models. The HSI model has the most straight-forward method: according to Gonzalez et al.'s textbook [9], the intensity i is computed as

$$i = \frac{r + g + b}{3}$$

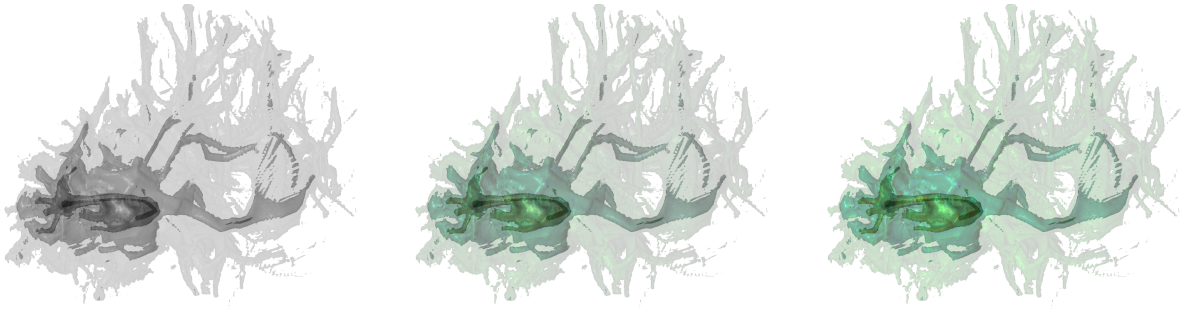
Then, a weighting factor is computed as $w = d^s$, where $d \in [0, 1]$ is the distance between the surface and the point of the dataset's bounding box that is closest to the eye point, and $s \in [0, 2]$ is a modifier. This results in a $w \in [d^2, 1] \subseteq [0, 1]$, so the colour can be changed from completely desaturated to almost completely saturated[1].

In the end, the weighting factor is used to compute the new colour c:

$$\text{colour}' = (1 - w) \cdot \text{colour} + w \cdot \text{intensity}$$

The focus-dependent saturation is presented in Figure 3.5. Its effect can be described as follows:

[1] To enable completely saturated colours, s would need to be infinitely large, but as Figure 3.5c demonstrates, $s = 1.5$ already looks very saturated.



(a) $s = 0.0$. (b) $s = 0.5$. (c) $s = 1.5$.

Figure 3.5.: Focus-dependent saturation improves the depth and shape perception. As before, the probabilities are $(0.8, 0.6, 0.4, 0.2)$.

- For $s = 0$, the complete surface is desaturated.
- For $s > 0$, the surface is more saturated in front and less saturated towards the back. If s is large, a bigger part of the surface is saturated.

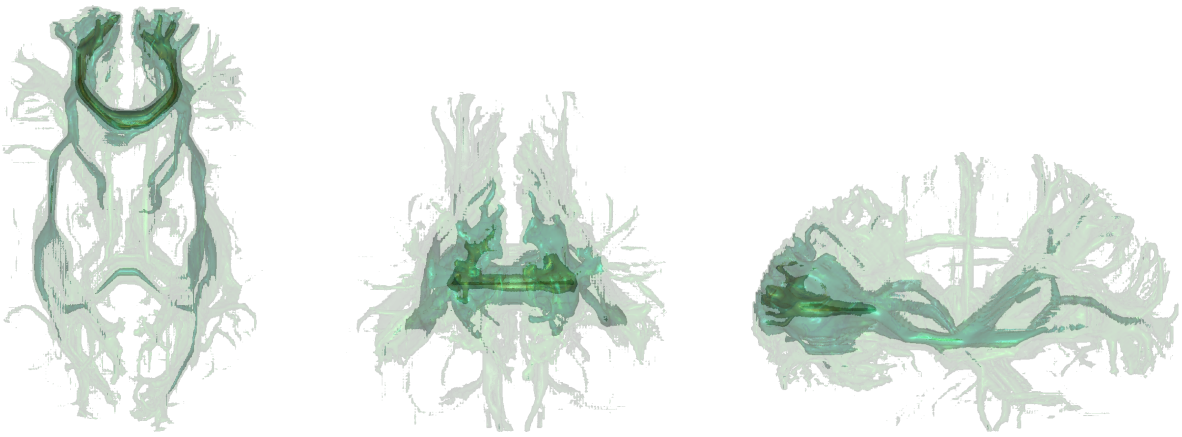
The modifier s can be adjusted by the user. By default, $s = 1$, so the colour and its grey value are interpolated linearly depending on the distance.

3.4. Summary

With the proposed methods, a visualisation is created that consists of multiple semi-transparent isosurfaces as seen in Figure 3.6. The transparency of the surfaces reflects the probability of diffusion from the seed voxel to that point in space. For a better spatial impression, the saturation depends on the focus. Figure 3.7 depicts a visual summary of this chapter. The implementation is discussed in Section 5.2.

The given visualisation provides a significantly better sense of long-range connectivity than slice-based approaches. In comparison to the nested isosurfaces presented by Goldau et al. [8] and others, this visualisation improves both shape and depth perception. Furthermore, using the adaptive transparency enables the user to distinguish more surfaces and get an impression of the significance of the data. Direct volume rendering approaches, such as the one by Rick et al. [21], have the fundamental problem that, in static images, the uniform colour results in a single colour blob. This makes it difficult to convey the shape without changing perspective or using a virtual reality environment. Using nested isosurfaces and adjusting the transparency and saturation helps to reduce this problem.

In the following chapter, the probabilistic tractogram is supplemented with anatomical context.

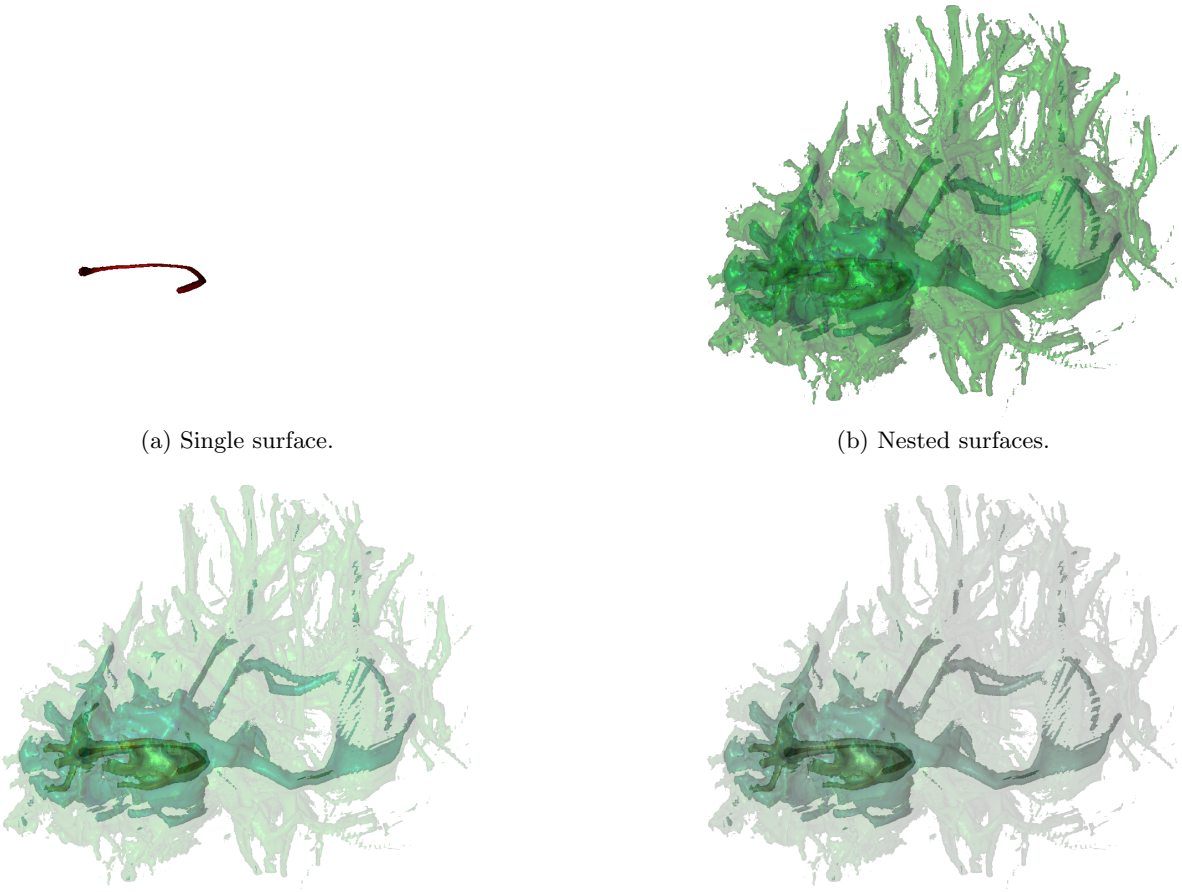


(a) Axial view with $s = 1.75$.

(b) Coronal view with $s = 0.75$.

(c) Sagittal view with $s = 1.25$.

Figure 3.6.: Views of the complete visualisation from the principal directions. Areas, which have a high probability of being a connected to the seed point, are more opaque than areas with a low probability.



(a) Single surface.

(b) Nested surfaces.

(c) Adaptive transparency.

(d) Focus-dependent saturation.

Figure 3.7.: Overview over the progress between the four stages.

4. Context Visualisation

Visualising the probabilistic tractograms alone is not enough to help the neuroscientists. One important goal is to learn, which functional regions of the cortex are connected, and which other white matter structures are involved. To be able to understand the connection between the tractogram and anatomy, they need anatomical context. A good context should give information about how the visualisation is embedded into the brain, but at the same time, it should not be obtrusive. Since the visualisation that is to be embedded is three-dimensional, a lower-dimensional context does not help.

Inspired by the visual appearance of Hummel et al.'s integral surfaces [12], I developed a normal-based initial rendering of the cortex. I decided to use T_1 images of the brain for this purpose, because they provide a good white and grey matter contrast. Then, the normal-based variation is optimised to reflect the properties of the brain shape as explained in Section 4.1. However, as will be seen, this alone does not suffice to create a non-obtrusive context for the probabilistic tractogram visualisation.

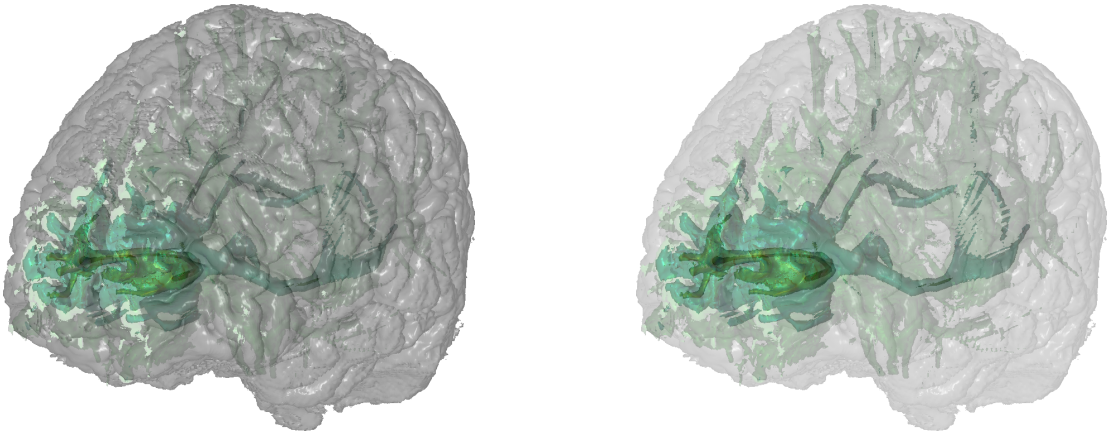
To reduce obtrusiveness of the context, and improve the visual impact of the visualisation inside the context, I reduced the opacity of the front-most parts of the brain so the viewer can see through to the tractogram representation. This is outlined in Section 4.2

Finally, since medical personnel are used to studying slices, this type of visualisation should be incorporated. Section 4.3 presents how I integrated textured slices with the existing context to create a consistent and fully interactive representation of the cortex as context.

4.1. Normal-Dependent Shading

In order to give anatomical context for brain data, the obvious idea is to draw the cortex around it. The easiest approach to create such a brain is to use MRI data and draw a semi-transparent isosurface for an isovalue isolating the cortex. The user is assisted to set the isovalue with a slider. However, such an isosurface would occlude the visualisation on the inside and dull its colours, as shown in Figure 4.1a. To address this problem, the isosurface can be made even more transparent. But as Figure 4.1b shows, this also leads to a loss of structure.

To address these problems, a visualisation is needed, which preserves the brain's unique structure while giving the viewer the opportunity to inspect what lies beyond. This can be achieved by emphasising the sulcii and the silhouette of the brain and making the gyrii appear transparent.



(a) $\alpha = 0.8$. (b) $\alpha = 0.4$.

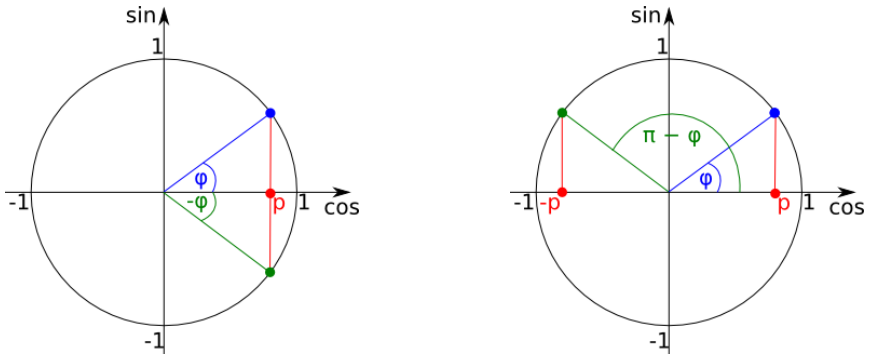
Figure 4.1.: Simple semi-transparent isosurfaces either occlude too much or are not recognisable. The part of the visualisation that is clearly visible can only be seen due to registration error – usually, there should not be any fibre outside the grey and white matter.

Looking at the isosurface of the cortex, one can notice a few properties: On the one hand, the gyrii have normals pointing towards the viewer in the middle of the image, but normals pointing away from the viewer at about 90° near the border of the image. On the other hand, the sulcii near the middle have normals pointing away whereas the sulcii that are obscured by gyrii have normals pointing towards the viewer (or away from the viewer). Since the latter are hidden, they are not considered any further.

For both objectives – emphasising structure and revealing the inside of the brain – the angle between the view vector and the surface normal plays an important role. Let φ be the angle between the view vector $\vec{v}$ and the normal vector $\vec{n}$. The cosine can be expressed by the dot product of two vectors.

$$\cos(\varphi) = \frac{\vec{v} * \vec{n}}{|\vec{v}| \cdot |\vec{n}|} = p \in [-1, 1]$$

Moreover, $\cos(\varphi) = \cos(-\varphi)$ and $\cos(\varphi) = -\cos(\pi - \varphi)$.



(a) $\cos(\varphi) = \cos(-\varphi)$. (b) $\cos(\varphi) = -\cos(\pi - \varphi)$.

Figure 4.2.: The relation between the angle φ and the cosine of the angle $\cos(\varphi)$.

Figure 4.2 illustrates the relation between the angle and its cosine on the unit circle. For the following, let $\cos(\varphi) := p$. As pointed out earlier, the gyrii have normals pointing towards the viewer where they obscure the visualisation beyond. However, close to the silhouette, the normals point away. The silhouette should be emphasised, and the visual obstacle should be removed. Therefore the opacity should be high where the normals point to the sides, i.e. $\varphi \approx n \cdot \pi + \frac{\pi}{2}$ and low where they approach the view vector, i.e. $\varphi \approx n \cdot \pi$ for $n \in \mathbb{N}$.

These conditions are fulfilled by the absolute value of the cosine:

$$|\cos(\varphi)| = \left| \frac{\vec{v} * \vec{n}}{|\vec{v}| \cdot |\vec{n}|} \right| \in [0, 1]$$

This means that φ and its reflections through the origin and the axes result in the same value. The restriction to the absolute value is not strictly necessary, because the orientation of the view vector and the normal are known. Only those surfaces that point towards the viewer are visible, thus the angle between the view vector and the normal is $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, and therefore $cos(\varphi) \in [0, 1]$. However, the absolute value provides the convenient qualities of always being non-negative – both colours and transparencies are positive – and being suitable as weights, as they lie in the interval $[0, 1]$.

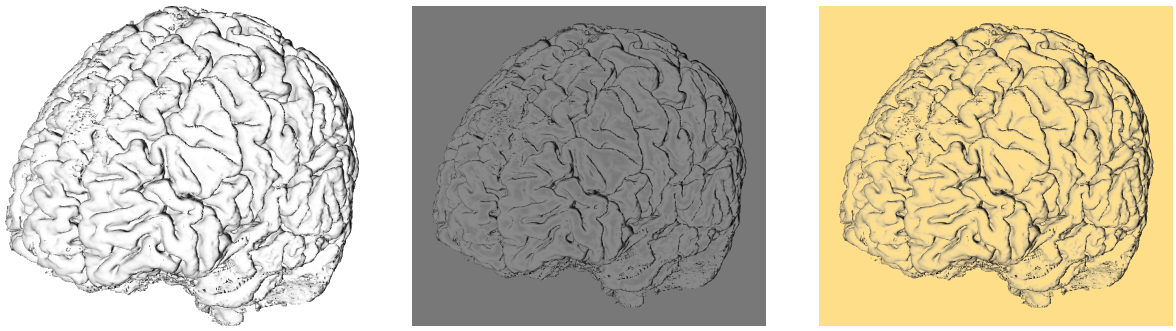
Initially, the surface is opaque and white. Other colours would be possible, but since context should be non-obtrusive, a grey-scale drawing is a good choice. The colour and transparency can be adapted as follows:

$$\text{colour}' = |\cos(\varphi)| \cdot \text{colour}$$

$$\alpha' = (1 - |\cos(\varphi)|)$$

if the colours are represented with values in $[0, 1]$.

Since the colour is initially white, it is bright on the gyrii and dark in the sulcii and near the silhouette. The transparency is initially 0, so the surface is transparent on the gyrii and opaque in the sulcii and near the silhouette. Figure 4.3 displays the visualisation of a "glass brain" using the method described above. To give a better impression of transparency and colour, the same visualisation is repeated with different background colours.

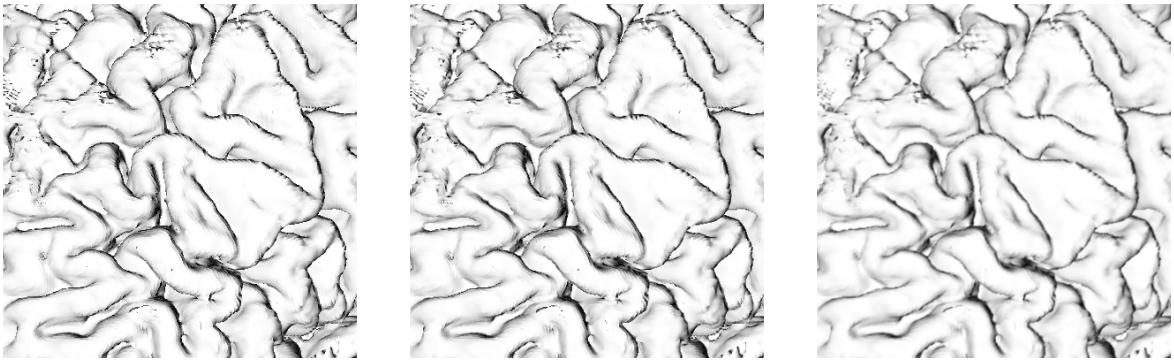


(a) White background. (b) Dark background. (c) Colourful background.

Figure 4.3.: The visualisation is suitable with different background colours.

When the brain is used as a context, a grey-scale background is more adequate since a colourful background can distract from the visualisation inside.

The datasets used in this thesis (cf. Appendix A) have a rather low resolution. Hence, the "glass brain" does not look as smooth as the manually segmented brains in other works (e.g. the ones presented in Section 2.3.2), which have spline curves as boundaries instead of raw image data. This issue can be addressed in two different ways. The first way is to smooth the T_1 slices with a Gaussian filter before the isosurface is computed. Figure 4.4 demonstrates that the surface becomes a tiny bit smoother, but this pre-processing step requires a significant amount of computation time. The second possibility is to apply a Gaussian filter directly to the rendered brain in a post-processing step. While this is done in real-time, it results in a very fuzzy rendering of the brain. Nevertheless, the user can choose to apply either of them if needed.



(a) Umsmoothed. (b) Pre-processed with Gaussian. (c) Post-processed with Gaussian.

Figure 4.4.: The visualisation is suitable with different background colours.

4.2. Focus-Oriented Transparency

Combining the context with the visualisation of the probabilistic tractogram results in a lot of visual clutter. This can be seen in Figure 5.7a.

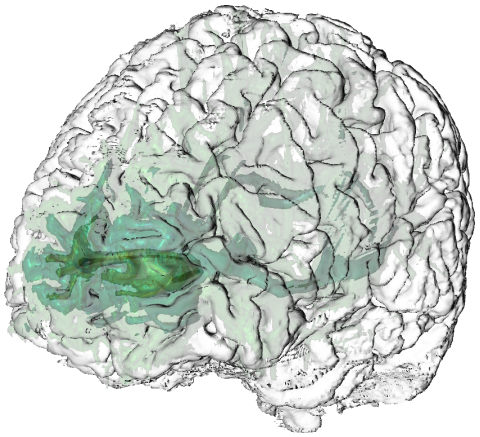


Figure 4.5.: Combining the context with the visualisation can result in visual clutter.

Due to the structure of the context, it is difficult to see the structure of the probabilistic

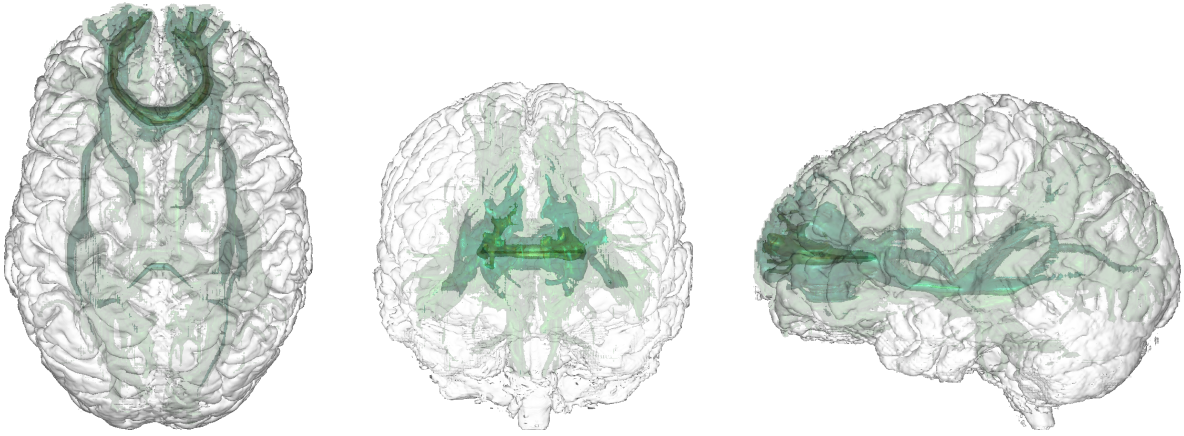
tractogram that lies behind it. To solve this problem, the cortex should be more transparent near the focus of the viewer. Assuming the user turns the view so the region of interest is close to the front, the focus lies on the region with the smallest distance to the eye point[1]. The easiest solution is to scale the transparency with the distance from the eyepoint. For a distance d, this can be done with

$$\alpha' = \alpha \cdot d$$

Since the user may want to influence how much of the context is transparent, there is a parameter $k \in [0, 2]$ to adapt it as follows:

$$\alpha' = \alpha \cdot d^k$$

As a result, the transparency can be changed dynamically from completely opaque ($k = 0$) to a very transparent surface ($k \in [0, 2]$), where the amount of transparency depends on the focus. The main motivation to make this parameter interactive can be observed in Figure 5.7: there is always a trade-off between context and visualisation as there is no value for k optimal for both. Either the context is clearly visible, but it occludes the structure of the visualisation, or the visualisation can be seen clearly but the context is rather weak.



(a) Axial view with $k = 0.5$.

(b) Coronal view with $k = 0.75$.

(c) Sagittal view with $k = 0.5$.

Figure 4.6.: The transparency can be adapted. A more transparent surface leads to a clearer visualisation but a more obscure context.

4.3. Cutting Planes for Context

Medical staff are, as mentioned before, trained to work with slices. Slices textured with T_1 images provide anatomical context, such as the segmentation into grey matter and white matter, and the position of ventricles[2]. Displaying them can help the user to understand how

[1] As pointed out in Section 3.3, I refer to the point on the bounding box, that is closest to the eye point rather than the eye point itself. The main difference lies in the fact, that zooming closer or further away from the visualisation should not change its behaviour.

[2] Ventricles are hollow parts of the brain filled with cerebrospinal fluid, which act as a buffer and provide immunological protection to the brain. They are seen on T_1 images as black shapes.

the visualisation fits into the context of the brain. Thus, there should be a way to incorporate them into context.

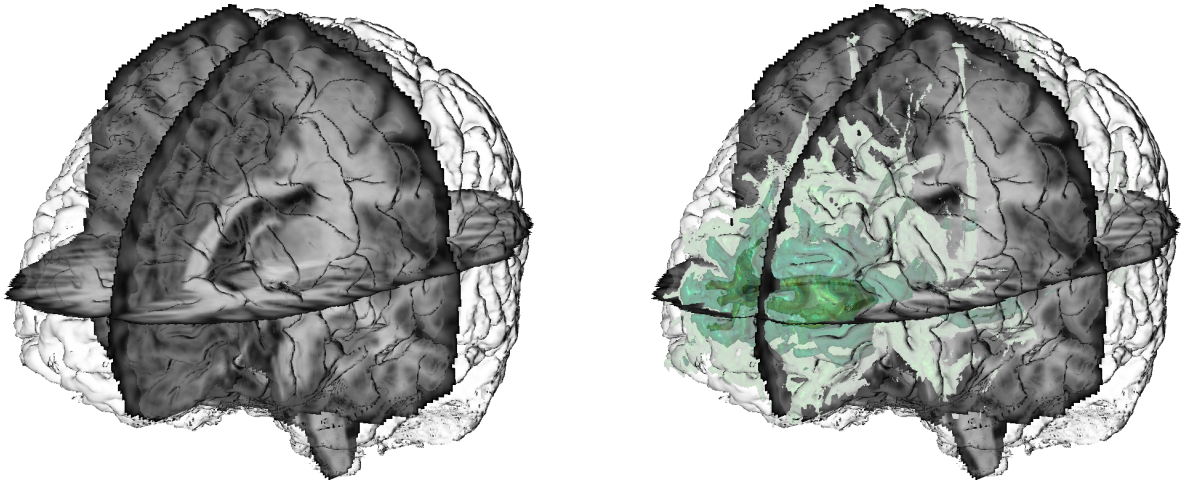


Figure 4.7.: Simply adding slices reduces the clarity of the brain structure.

This is achieved by making parts of the surface opaque. Choosing these parts is restricted to the six possible half-spaces[3] for the principal axes. The user can choose to make one half-space of each principal axis opaque or leave both half-spaces semi-transparent. At most, seven octants can be opaque at the same time.

In order to achieve this effect, some information is needed for each slice $i \in \{\text{sagittal (s), coronal (c), axial (a)}\}$:

- $v_i \in \{-1, 0, +1\}$ captures which half of the brain should be opaque. If $v_i = -1$, the left, dorsal or inferior half is opaque, whereas if $v_i = +1$, the right, ventral or superior half is opaque. $v_i = 0$ means that both halves can be transparent.
- S_i is the position of slice i in the local coordinate system. It can be changed interactively by the user.
- P_i is the current point's coordinate in the local coordinate system[4].

For each principal direction, one can determine whether a point is semi-transparent or opaque by computing

$$\max\{\alpha,\ v_i \cdot \operatorname{sgn}(P_i - S_i)\}$$

where $\operatorname{sgn}(x)$ denotes the sign function, i.e.

[3] A *half-space* consists of all points in (hyper)space that lie on the same side of a (hyper)plane.
[4] For simplicity, the local coordinate system uses the form (s,c,a) instead of (x,y,z).

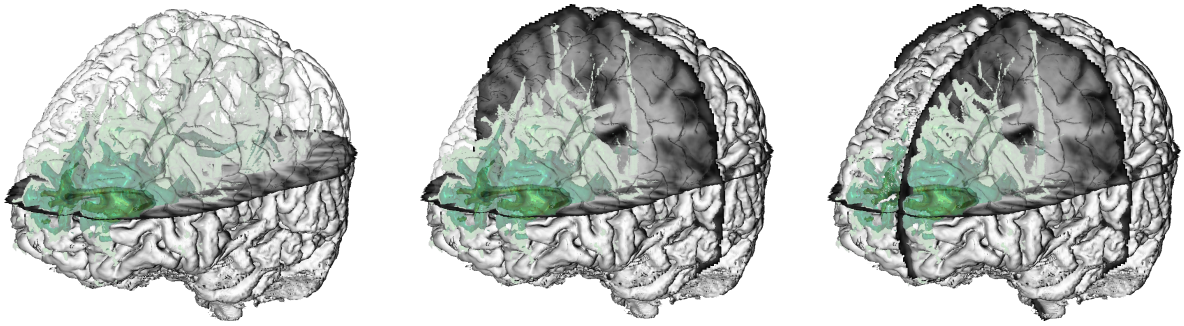
$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ +1 & \text{if } x > 0 \end{cases}$$

and $\alpha \in [0, 1]$ is the transparency as computed in the previous section. By $P_i - S_i$, the distance between the current point and the slice is computed. It is positive for all points in the right, ventral and superior part of the brain, and negative for all points in the left, dorsal and inferior part with respect to the corresponding slice. Hence, the multiplication of v_i with the sign of the difference is positive, if they match and negative, if they do not match. For example, a match is found, if $v_c = -1$ selects the dorsal half-space and the distance $P_c - S_c < 0$, so p lies on the dorsal side. In case of a match, the formula equates to $\max\{\alpha, 1\} = 1$. Then, the half-space in question, e.g. the dorsal part of the brain, is displayed opaquely whereas the ventral part is displayed semi-transparently. Figure 4.8a shows the result of the example at hand.

Since the user may want to combine different opaque half-spaces, the formula needs to be adapted. Instead of taking the maximum of just one parameter and α, one can take the maximum of all three parameters and α:

$$\alpha' = \max\{\alpha,\ v_s \cdot \operatorname{sgn}(P_s - S_s),\ v_c \cdot \operatorname{sgn}(P_c - S_c),\ v_a \cdot \operatorname{sgn}(P_a - S_a)\}$$

For each point in space, this is either α or 1. Using this combined formula, a combination of up to three half-spaces is possible. Figure 4.8 displays different combinations.



(a) The inferior half-space is opaque.

(b) The dorsal half-space is added.

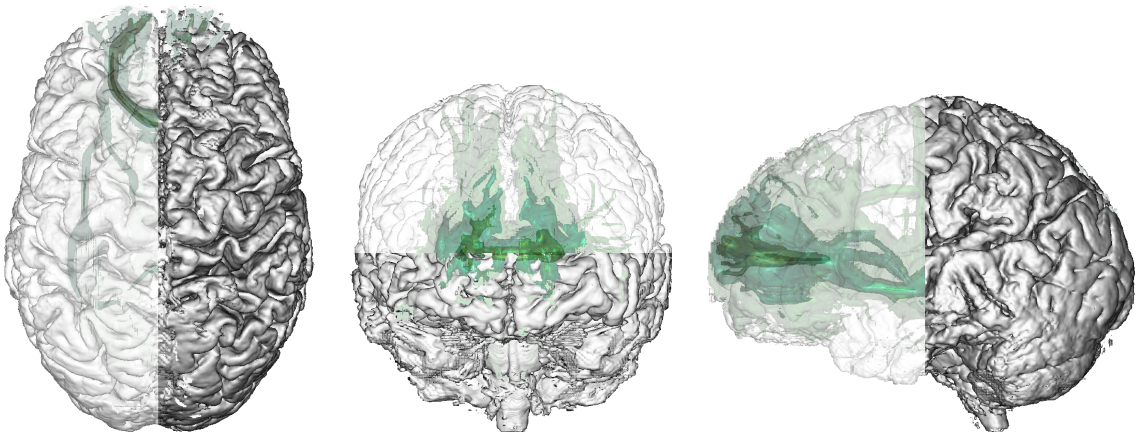
(c) The right half-space is added.

Figure 4.8.: Gradual addition of opaque half-spaces, until only the ventral, superior, left octant remains semi-transparent. In all three images, $k = 0.35$.

4.4. Summary

Using the techniques proposed in this chapter, one can create an anatomical context that reflects the structure of the cortex, as Figure 4.9 demonstrates. By applying a focus-oriented

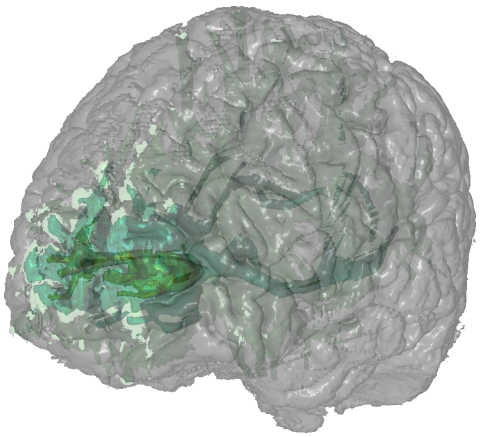
transparency, visual clutter can be avoided; the closest part of the cortex is very transparent, while parts that are further away are more opaque. Finally, slices with opaque half-spaces can be added and combined to create a context that combines the state-of-the-art of medical diagnosis with a three-dimensional representation. Figure 4.10 depicts a visual summary of this chapter. The implementation of said techniques is discussed in the following chapter.



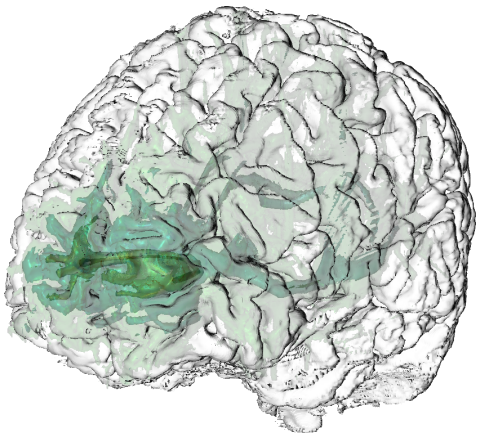
(a) Axial view ($S_a = 1.5$). (b) Coronal view ($S_c = 1.0$). (c) Sagittal view ($S_s = 1.0$).

Figure 4.9.: Views of the complete visualisation from the principal directions.

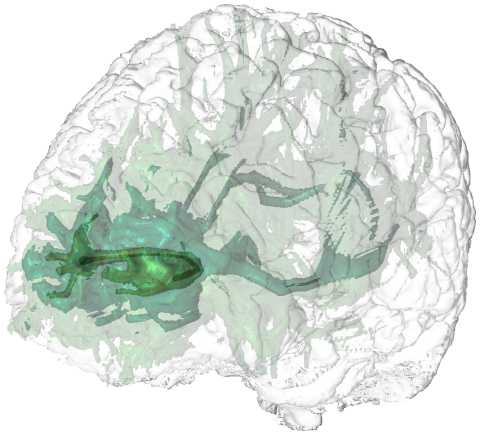
While the concept of normal-dependent shading was inspired by Hummel et al. [12], the formula itself was developed independently. It differs from theirs in that it is adapted to visualising brain data. Compared with slice-based context such as the one presented by Goldau et al. [8], the context at hand provides a more global view of the anatomy surrounding the slice. The "glass brain" presented by Born et al. [3] and Jainek et al. [13] is more suitable to this respect. But since they use their brain to display their visualisation on top, the amount of information that can be displayed is rather restricted. Therefore, the context cannot easily be combined with different visualisations. Svetachov et al. [23] and Everts et al. [7] provide a context that serves the same purpose as the one presented in this thesis. But while deterministic tracts can be represented by line drawings very well, this is not easily feasible for the purely volumetric probabilistic tracts. Since the given visualisation does not match their context stylistically, a different kind of context is needed, like the one at hand.



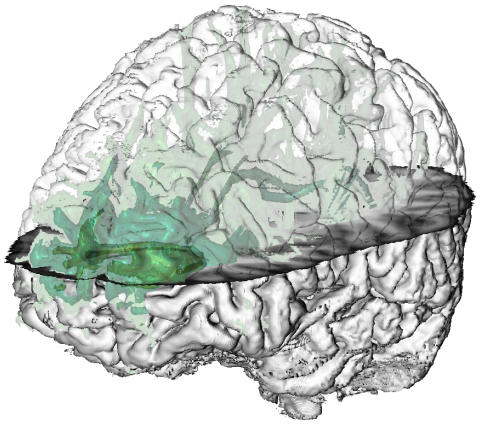
(a) Semi-transparent surface.



(b) Normal-dependent shading with $k = 0.0$.



(c) Focus-dependent alpha with $k = 1.0$.



(d) Cutting planes with $k = 0.35$.

Figure 4.10.: Overview over the progress between the four stages.

5. Implementation

The implementation of this work is part of the OpenWalnut project. OpenWalnut is an open source software for medical visualisation with a special focus on brain visualisation and image processing. It provides a lot of basic functionality like reading datasets, displaying axial, coronal and sagittal slices for these datasets, and interaction. But it also offers more advanced functions like colour mapping, Phong shading and various post-processing techniques.

In addition to the core functions, modules can be added. The user can connect modules to each dataset and to other modules. Furthermore, there are module containers that can contain several modules. Some examples of existing modules are a simple isosurface ray caster, upon which my modules are based, deterministic fibre visualisation, and slice-based probabilistic tractography visualisation (as mentioned in section 2.3).

My thesis provides two additional modules for OpenWalnut. ProbTractVis is a module that implements the visualisation methods described in Section 3. The other module, ProbTract-Context, implements the context methods from Section 4. Both modules can visualise scalar fields, but they are intended for probabilistic tractograms and T_1 images respectively. They can be used independently, so it is possible to provide context for other visualisations as well. In addition to writing these modules, I also added a bit of functionality to the matrix class.

5.1. General Module Design

Modules have a C++ framework, a GUI, and a GLSL shader, which interact as sketched in Figure 5.1.

In the framework, one can add properties that can be modified via the GUI, e.g. colours are set with a colour picker, numbers are modified with a slider, and booleans are set with a check box. All the properties, as well as other elements, can be handed on to the shader as *uniforms*, i.e. elements that are read-only in the shaders.

A shader is a small graphics program that runs on the Graphical Processing Unit (GPU). GLSL, the OpenGL Shading Language [22], is an C-based programming language that can be used with OpenGL. The shader itself consists of two components: a *vertex shader* and a *fragment shader*. The graphic in Figure 5.2 gives a good overview over the rendering pipeline.

At first, the vertex shader processes each scalar from the probabilistic tractogram. It can define *varyings*, which can only be read by the fragment shader, and it has some tasks like transforming vertices, computing their lighting, and generating texture coordinates. Afterwards, the scene is rasterised and processed in the fragment shader. There, all image space

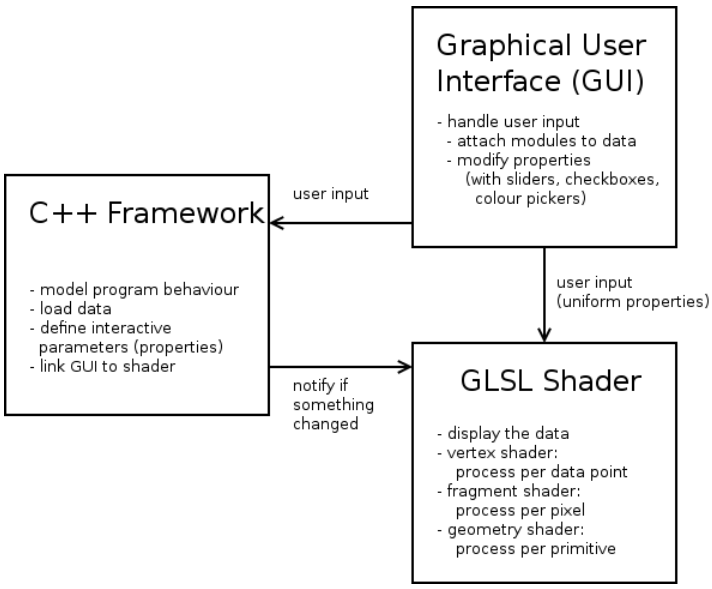


Figure 5.1.: The three main components of OpenWalnut: the C++ framework to model the program behaviour, the GUI to handle user input, and the GLSL shader to display the data.

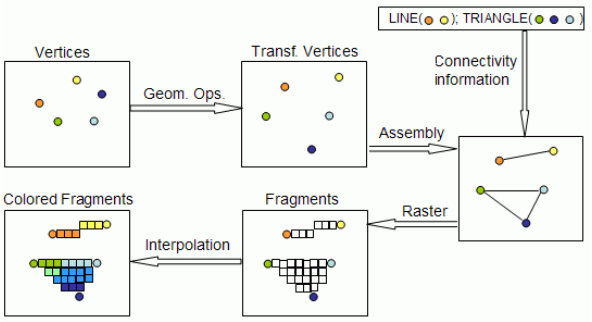


Figure 5.2.: The GLSL pipeline by Lighthouse3D.com [16].

computations take place: for each fragment, or pixel, the fragment shader is carried out in parallel on the GPU. In the end, the fragment has been assigned a colour and a depth.

The basic fragment shader for both modules can be found in Appendix B.

5.2. The Probabilistic Tractogram Visualisation Module: ProbTractVis

As we recall, the probabilistic tractogram visualisation has three main parts. First, there needs to be a way to view nested isosurfaces, then the transparency has to be adapted depending on the isovalue, and finally, the saturation is to be varied depending on the depth.

5.2.1. Nested Isosurfaces

The first step towards displaying more than one isosurface is to introduce a way to render several surfaces at the same time. This is achieved by creating a simple loop in the main function. Instead of calling `rayCast` once, it is called several times, depending on the uniform `surfCount`, which can influence how many surfaces are rendered. The isovalues are now given to the shader as the 4x1 vector `u_isovalue`, and the colours are stored in the rows of the 4x4 matrix `u_isocolour`. In the loop the isovalue and its corresponding colour are selected and handed to the ray casting function.

```
for( int j = 0; j < u_surfCount; j++ )
{
   isovalue = u_isovalues[j]; // select jth isovalue
   isocolour = u_isocolours[j]; // select jth isocolour
   rayCast( curPoint, isovalue, isocolour );
}
```

An iterative solution like in this fragment shader, which calls the ray caster several times, is still more efficient than a nested if statement in the ray caster. A test yielded, that nested if statements are not interactive, even if only two surfaces are rendered. In comparison, the iterative solution can operate with interactive speed even for four surfaces.

The second step that is needed to render several isosurfaces is to blend the colours accordingly. In chapter 3, the following formulae were produced for back-to-front compositing (the new isosurface i lies in front of the fragment f as computed so far):

$$c' = \alpha_i \cdot c_i + (1 - \alpha_i) \cdot \alpha_f \cdot c_f$$
$$\alpha' = \alpha_i + (1 - \alpha_i) \cdot \alpha_f$$

As mentioned before, the ray caster stops after the first match for each fragment. This implies, that back-to-front compositing is identical to inside-to-outside compositing as the values decrease from the core to the borders of the shape.

In the next step of the ray caster, the final fragment colour is constructed as follows, where `rgb` refers to the colour values and `a` refers to the alpha value.

```
wge_FragColor.rgb = vec4( light * colour.rgb, colour.a );
```

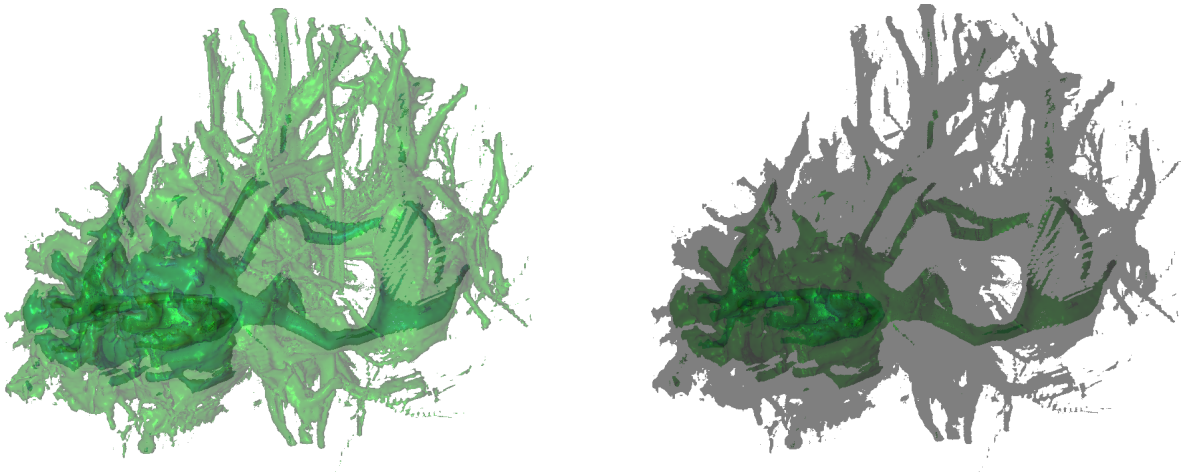
Then, back-to-front compositing is applied. The new isosurface's colour and transparency are part of the vector `colour`, while the old fragment colour and transparency are part of the vector `wge_FragColor`. As a shorthand (and speed-up) for the elements of a vector, one can access any vector elements by appending "swizzles" like `.rgba`. For example, `colour.rb` produces a 2x1 vector containing only the first and third element of the original 4x1 vector. Swizzles are extremely useful, whenever the same operation is applied to some, but not all elements of a vector. In the example above, they are used to modify the colour dependent on the lighting, while leaving the alpha value untouched.

Another way to make code more efficient is to use `mix( one, two, f )`. This function

computes the convex combination $(1 - f) \cdot \text{one} \; + \; f \cdot \text{two}$. Therefore, the back-to-front compositing can be implemented using the `mix` function.

```
wge_FragColor.rgb = mix( wge_FragColor.rgb, light * colour.rgb, colour.a );
wge_FragColor.a = mix( wge_FragColor.a, 1, colour.a );
```

In practice, the visualisation leads to better results in terms of colour representation, if the multiplication with the current fragment transparency α_f is left out. In Figure 5.3, one can observe that multiplying the new colour with the current fragment transparency yields a grey tint. If the multiplication is omitted, the colours are more vibrant.



(a) Without multiplication.

(b) With multiplication.

Figure 5.3.: Multiplication with α_f produces a grey tint.

5.2.2. Varied Transparency

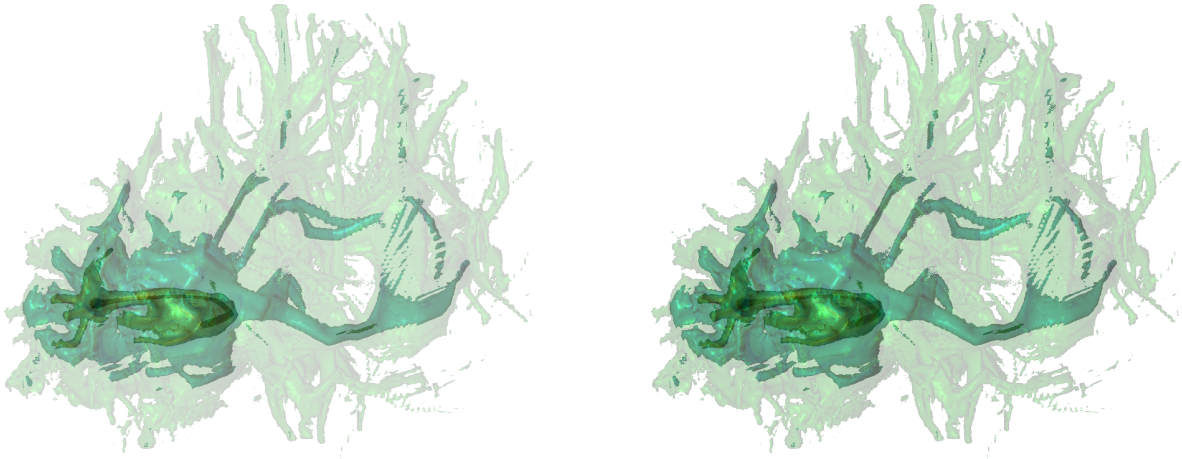
The transparency is varied depending on the isovalue. As seen before, it can be computed as

$$\alpha_i = \frac{x_i - a}{b - a} \in [0, 1]$$

where $x_i \in I = [a, b]$.

Instead of $b-a$, one can also use $x_{max}-a$, since this enhances the colours as seen in Figure 5.4. This is particularly interesting, if the probabilities of the surface are less high, since the resulting visualisation looks very pale due to its low transparency. However, it also conveys a wrong impression of the dataset, since the higher opacity makes the probabilities appear higher than they are. Therefore, $b - a$ is used to normalise the transparency. The isovalues are scaled to $[0, 1]$ in the vertex shader, so it is not necessary to alter them before using them as transparencies.

Prior to computation, the isovalues need to be chosen for each dataset. The user is assisted in choice by pre-set values. Afterwards, the user can also adjust them to any other value. The



(a) $\alpha_i = \frac{x_i - a}{b - a}$ (b) $\alpha_i = \frac{x_i - a}{x_{max} - a}$

Figure 5.4.: Dividing by $x_{max} - a$ instead of $b - a$ results in more vibrant colours, but it also conveys a wrong impression of the dataset.

only restriction is to order them from the highest to the lowest isovalue to ensure the back-to-front compositing is done correctly. The `greaterThan( vec4, float )` function performs an element-wise comparison of the input vector's elements with the given float. This can be used in the shader as follows:

```
// sort the isovalues (and their colours) in ascending order
vec4 isovalues;
mat4 isocolours;
// determine order (largest value first)
ivec4 order = ivec4( greaterThan( vec4( v_isovalues[0] ), v_isovalues ) ) +
              ivec4( greaterThan( vec4( v_isovalues[1] ), v_isovalues ) ) +
              ivec4( greaterThan( vec4( v_isovalues[2] ), v_isovalues ) ) +
              ivec4( greaterThan( vec4( v_isovalues[3] ), v_isovalues ) );
// sort the values and colours according to the determined order
for( int i = 0; i < 4; i++ )
{
   isovalues[i] = v_isovalues[ order[i] ];
   isocolours[i] = u_isocolours[ order[i] ];
}
```

First, the vector `order` is constructed in a way, that each component is assigned its place in decreasing order. This is achieved by comparing each component with the others, and counting how many times it is greater. Afterwards, the isovalues, transparencies and colours are set in decreasing order.

Both the isovalues and the transparencies are handed on to the shaders for further use. In the main loop of the fragment shader, the value-dependent transparency is made optional. If the user chooses to set the transparencies manually, the colour is set to what was defined in the GUI. Otherwise, if the user decides to apply the adaptive transparency, the alpha channel uses the isovalue.

```
for( int j = 0; j < u_surfCount; j++ )
```

```
{
   isovalue = u_isovalues[j];
   #ifdef MANUALALPHA_ENABLED // use user-chosen transparency
      isocolour = u_isocolours[j];
   #else // use adaptive transparency
      isocolour = vec4( u_isocolours[j].rgb, isovalue );
   #endif
   rayCast( curPoint, isovalue, isocolour );
}
```

The rest of the loop remains unchanged.

5.2.3. Focus-Dependent Saturation

Adapting the saturation of the isosurface is part of the colour construction process, i.e. the fifth part of the ray casting pipeline (cf. Appendix B).

In Section 3.3, the first step is to determine grey value of the desaturated colour. This is done by $i = \frac{r+g+b}{3}$. The GLSL equivalent can be seen below. All one needs to do is to compute the mean of the three colour channels and create a vector with the mean value for each component, using the shorthand `vec3( float )`.

```
vec3 grey = vec3( ( colour.r + colour.g + colour.b ) / 3 );
```

Next, the relative distance $d \in [0, 1]$ between the surface and the eye point is computed. For the j^{th} step of the ray caster[1], the relative distance is calculated as follows:

```
float d = stepDistance * j / maxDistance;
```

Finally, the new colour can be determined using the weighting factor $w = d^s$ where $s \in [0, 2]$ is a user-defined modifier that can be adapted by a slider:

$$\text{colour'} = (1 - w) \cdot \text{colour} + w \cdot \text{isocolour}$$

In the implementation, the weighting factor is computed inline using GLSL's power function `pow( base, exponent )`.

```
colour.rgb = mix( colour.rgb, grey, pow( d, u_saturation ) );
```

The exponent s is given to the shader as the uniform `u_saturation`. As pointed out before, the `mix` function is very efficient for computing convex combinations, thus it is applied to determine the colour.

Summary

The finalised module is displayed in Figure 5.5. All modifiable parameters of the module are available in the module settings in the top right of the GUI. The surface colours can be

[1] The number of steps can be changed by the user, and it has a significant impact on the speed. By default, it is set to 250, since it provides a good balance between interactive speed and image quality.

adjusted with a colour picking widget, if the user wants to choose colours or transparencies manually. The number of isosurfaces to be displayed, the isovalue for each surface, and the saturation can be adapted using sliders. Using the "manual transparency" check box, the user can switch between the adaptive transparency and their own choice of transparencies. On the left, the visualisation is rendered. Users can change perspective and viewing distance manually.

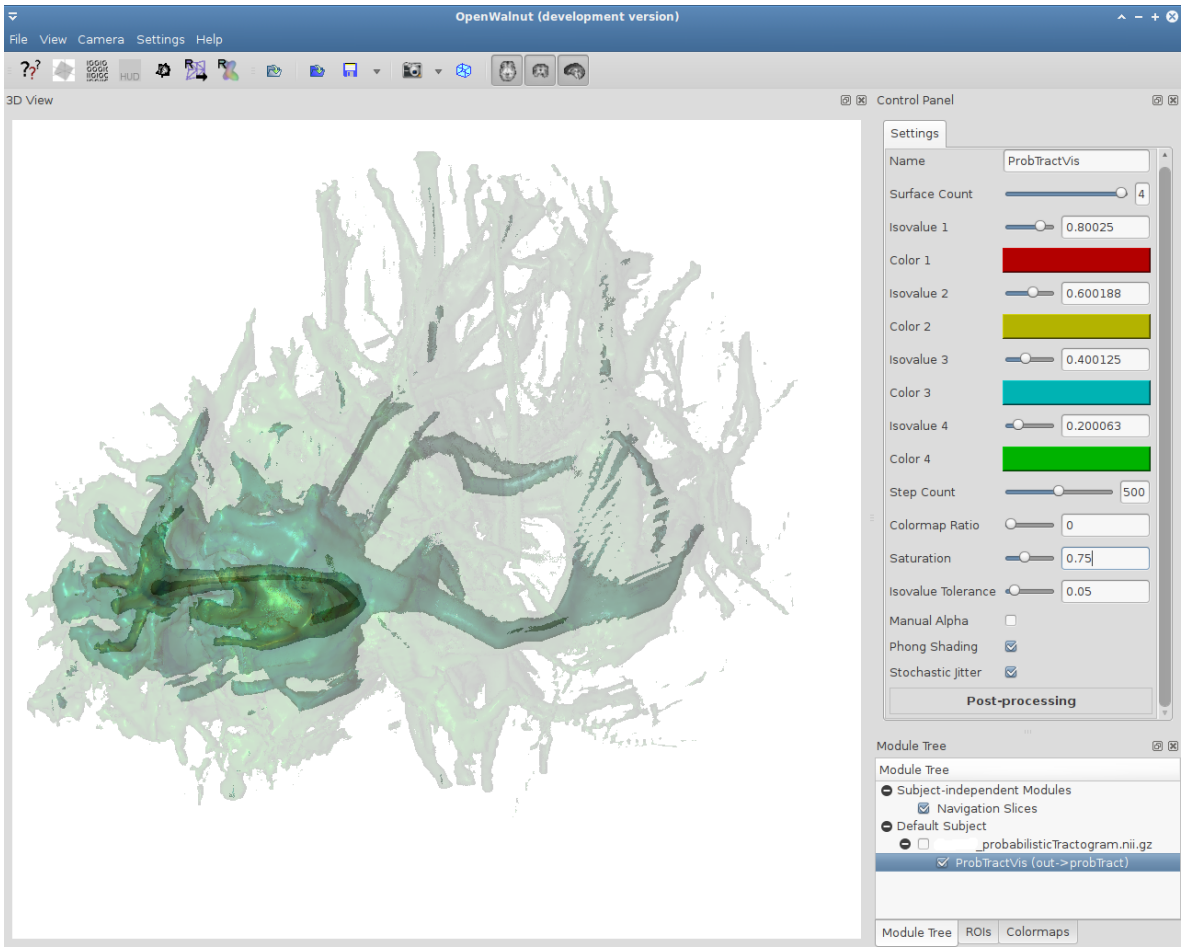


Figure 5.5.: Screenshot of the GUI for the Visualisation module.

As presented in the previous sections, the visualisation gives a very good impression of the probabilistic tractogram since it is sensitive to the probabilities and emphasises depth differences. It helps the user to understand the dataset, and it is fully flexible.

5.3. The Context Module: ProbTractContext

As seen in the previous chapter, the context module consists of three main parts. First, an illustrative brain is rendered using normal-dependent shading. Then, it is made more visually permeable to let the tractogram shine through by applying focus-oriented transparency. Fi-

nally, cutting planes and opaque half-spaces are added to provide more insight into the brain anatomy.

5.3.1. Normal-Dependent Shading

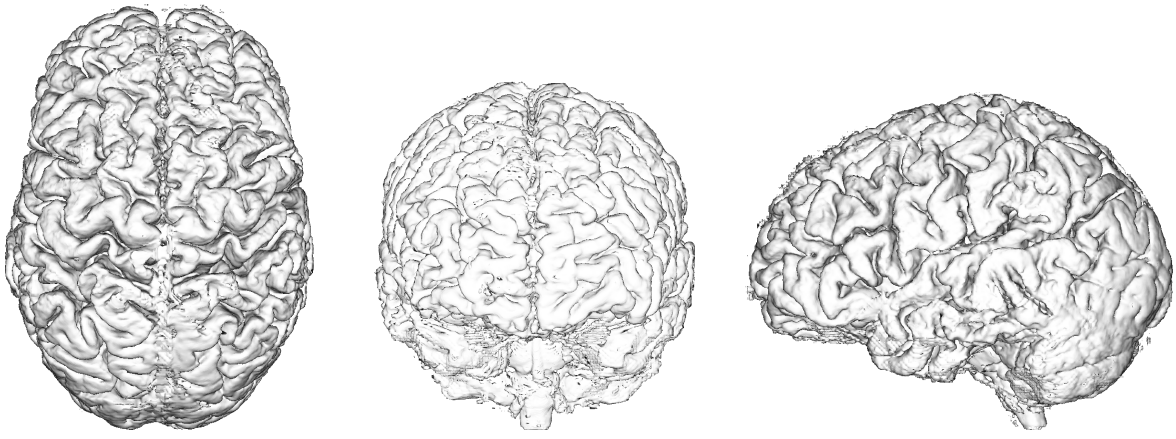
The method I developed for normal-dependent shading uses the absolute value of the cosine between the normal and the view vector $|\cos(\varphi)| = \left|\frac{\vec{v}*\vec{n}}{|\vec{v}|\cdot|\vec{n}|}\right|$. This can be converted into a simple GLSL expression, where `v_ray` is the view vector and `normal` is the surface normal at that point.

```
// Transfer view vector from local to world coordinates
vec4 view = gl_ModelViewMatrix * vec4( v_ray, 0.0 );
float factor = abs( dot( view.xyz, normal ) /
                    ( length( view.xyz ) * length( normal ) ) );
```

Both the colour and the transparency are modified depending on this value. On the one hand, the colour is multiplied with the factor, so it becomes darker as the angle approaches 90°: $\text{colour}' = |\cos(\varphi)| \cdot \text{colour}$. On the other hand, the transparency is multiplied by one minus the factor, so the brain becomes more opaque where the angle approaches 90°, while it remains more transparent where it approaches the view axis: $\alpha' = (1 - |\cos(\varphi)|) \cdot \alpha$.
In GLSL, this can be implemented in one line by setting the intermediate colour as follows:

```
vec4 colour = vec4( u_isocolour.rgb * factor, 1 - factor );
```

Here, `u_isocolour` is the colour used for the brain. By default, it is white to avoid distracting from the tractogram visualisation, but it can be changed interactively. The red, green and blue values are set using swizzles, as explained in Section 5.2.1, while the alpha value is set separately.



(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure 5.6.: Different views of the illustrative brain prior to reducing the transparency dependent on the focus (i.e. $k = 0$).

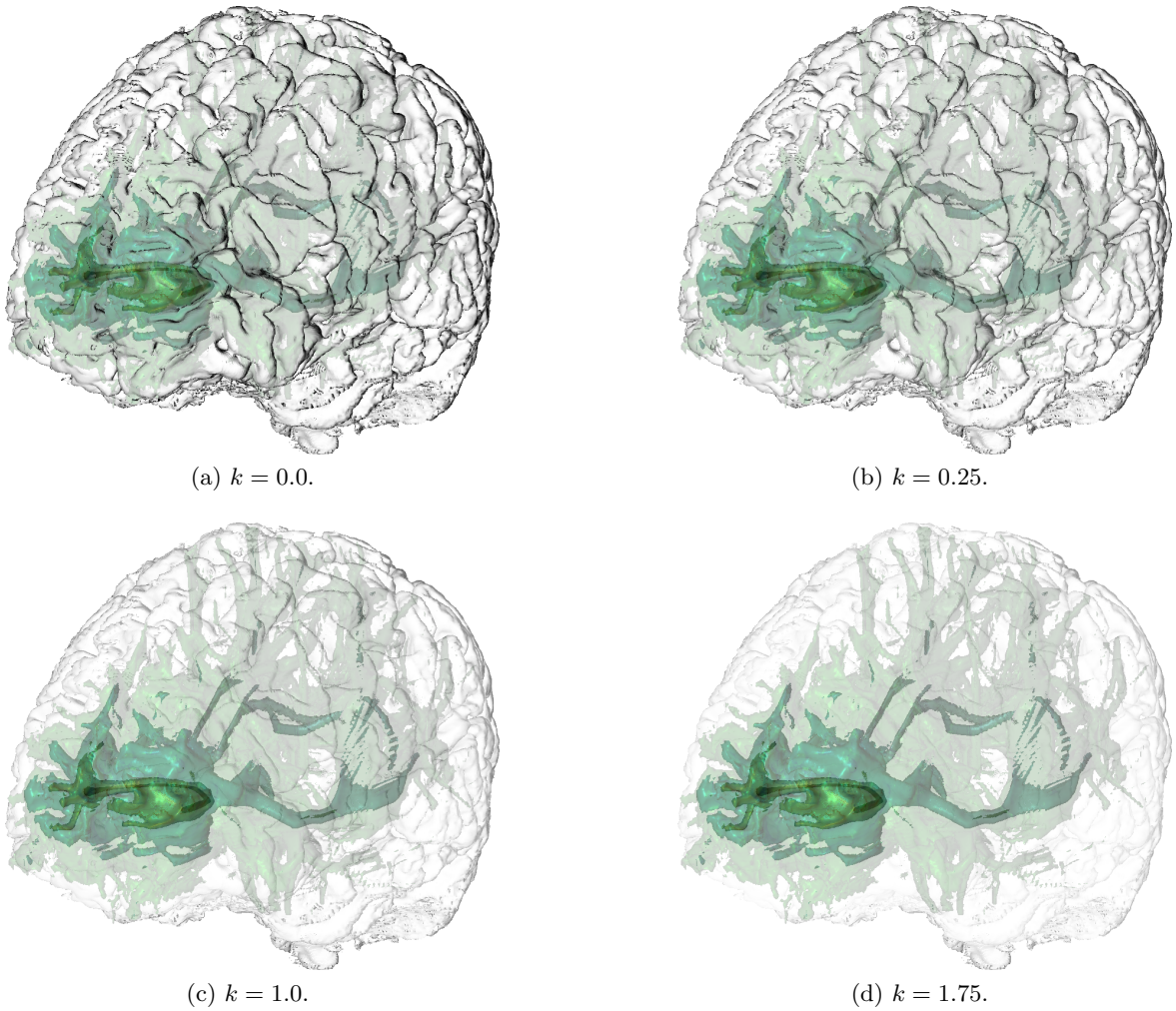
5.3.2. Focus-Oriented Transparency

To prevent the context from occluding the probabilistic tractogram visualisation, the transparency is adapted depending on the distance d to the part of the bounding box, which is closest to the viewer, and a parameter $k \in [0, 2]$ which can be modified interactively: $\alpha' = \alpha \cdot d^k$.

In the shader, this is done using the following two lines.

```
float d = stepDistance * j / maxDistance;
colour.a = colour.a * pow( d, u_alpha );
```

First, the distance is computed like in Section 5.2.3. The parameter k can be varied and is therefore given to the shader as the uniform `u_alpha`. The focus-dependent transparency can be computed by multiplying the alpha value with d^k, which is `pow( d, u_alpha )` in GLSL.



(a) $k = 0.0$. (b) $k = 0.25$.

(c) $k = 1.0$. (d) $k = 1.75$.

Figure 5.7.: The transparency can be adapted using the parameter k. A more transparent surface leads to a clearer visualisation but a more obscure context.

5.3.3. Cutting Planes for Context

In Section 4.3, cutting planes are added and the user can choose to make parts of the cortex opaque. The opaque parts can be combined arbitrarily.

The functionality of viewing and changing positions of slices is already provided by OpenWalnut. Thus, the main challenge lies in obtaining said position and using it for the opaque parts. As pointed out before, the parameter $v_i \in \{-1,\ 0,\ +1\}$ captures, if half-space i of the brain should be opaque (for $i \in \{\text{axial}(a),\ \text{coronal}(c),\ \text{sagittal}(s)\}$). The v_i can be manipulated interactively with discrete sliders. Each slice position is represented by S_i and the current point's local coordinates are P_i.

The half-spaces are combined by

$$\alpha' = \max\{\alpha,\ v_s \cdot \text{sgn}(P_s - S_s),\ v_c \cdot \text{sgn}(P_c - S_c),\ v_a \cdot \text{sgn}(P_a - S_a)\}$$

In the shader, this is achieved by using nested `max` statements. To make the code more readable, the operation is split up into several lines:

```
colour.a = max( u_viewSagittal * sign( curPoint.x - u_sagittal ), colour.a );
colour.a = max( u_viewCoronal  * sign( curPoint.y - u_coronal ),  colour.a );
colour.a = max( u_viewAxial    * sign( curPoint.z - u_axial ),    colour.a );
```

As usual, α is acquired with `colour.a`. The parameters v_i that capture, which parts of the brain should be opaque, are represented by `u_viewAxial`, `u_viewCoronal`, and `u_viewSagittal`. Further, the current point's coordinates P_i are given as `curPoint` and the corresponding swizzle. Finally, the current slice positions S_i can be changed interactively by the user and are therefore handed to the shader as uniforms `u_axial`, `u_coronal`, and `u_sagittal`.

Summary

Figure 5.8 displays the finalised module. Again, all parameters can be changed in the module settings. The focus-oriented transparency can be adapted using a slider. How much it is adapted largely depends on how the probabilistic tractogram is shaped. For only one or two small isosurfaces, it is enough to leave a small viewing window in front, but for a larger number of surfaces, which fill up most of the "glass brain", it makes sense to reduce the visibility of the frontal parts of the brain. The opaque half-spaces can be set using the corresponding sliders. In the screenshot, the dorsal part is opaque (i.e. the slider is set to -1), while the rest is semi-transparent (i.e. the sliders are set to 0). The slices can be turned on and off using the slice icons in the top bar. While they are switched on, they can be moved along their respective axes arbitrarily and they are used to reveal more or less of the visualisation.

As we have seen, the methods presented in this thesis can be used to create a context, which can help neuroscientists to get a better understanding of the data. It may not be possible to create a context that is entirely unobtrusive, and at the same time meaningful. However, the context at hand provides a good balance between revealing the visualisation and providing anatomical structure. Like the tractogram visualisation, it is very flexible since the users can adapt it according to their individual needs.

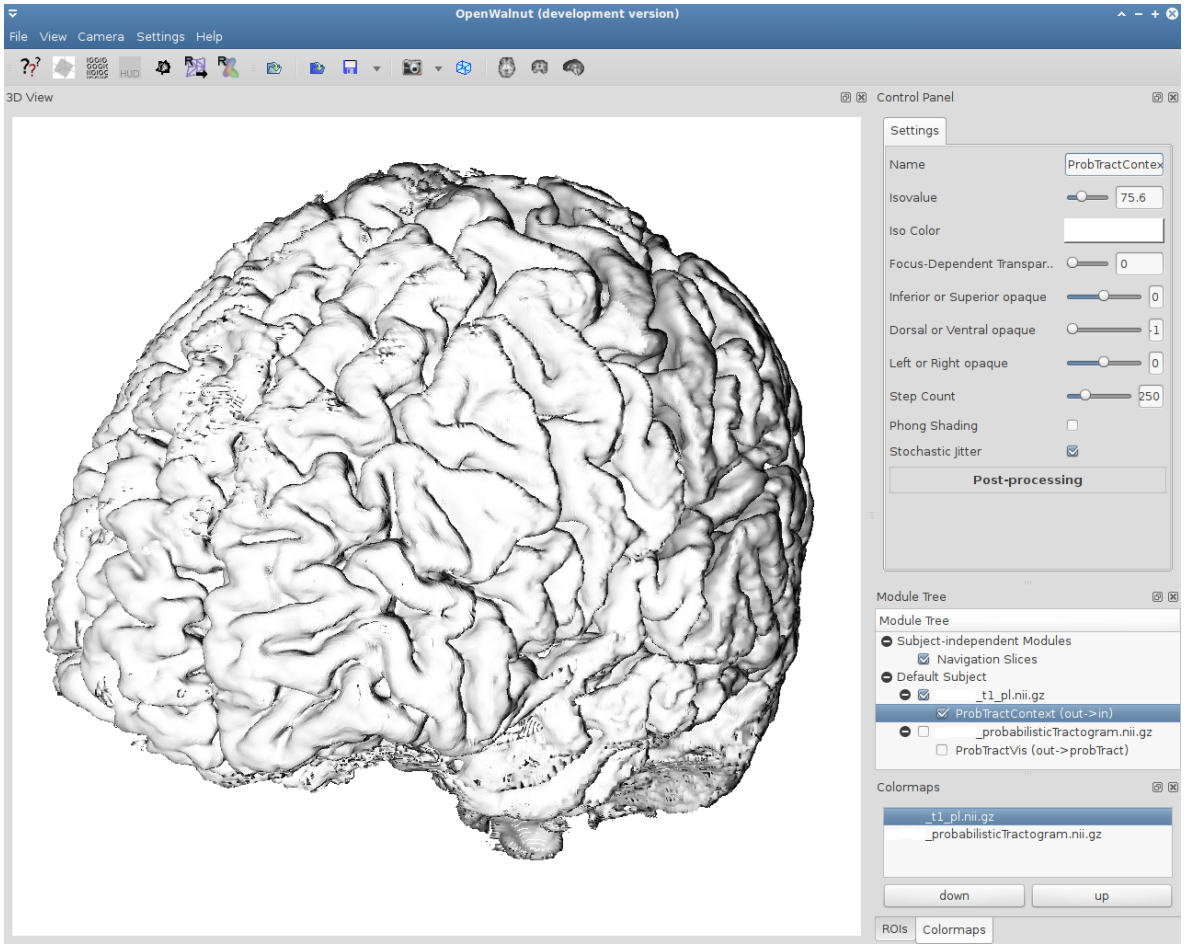


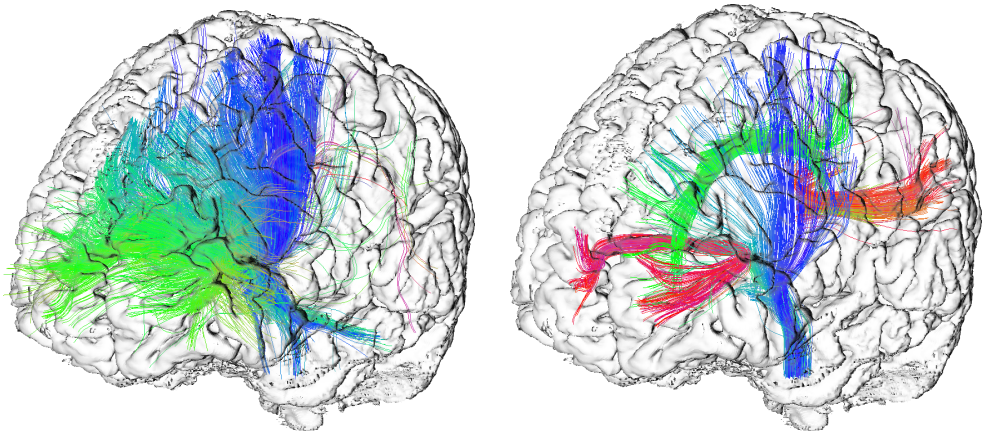
Figure 5.8.: Screenshot of the GUI for the Context module.

5.4. Applications

As pointed out before, the methods I developed are summarised into modules. The most important property of modules is that they can also be used individually, or in combination with other modules.

Context is not only needed for probabilistic tractograms but also for other visualisations of brain data. Figure 5.9 depicts the most closely related type of data: deterministic tractography. As one can see, the context suits the deterministic fibres well, but the vivid colours and the amount of fibres subdue the context. The OpenWalnut module for fibre display offers the user to select regions of interest (ROI) to restrict the amount of fibres of a dataset. To view specific bundles, some additional datasets are available.

Of course, one can also combine the probabilistic tracts with deterministic fibres, as Figure 5.10 demonstrates. The amount of deterministic fibres is restricted to those that run through a ROI box in the region of the probabilistic tractogram, that has the highest proba-



(a) Some fibres selected with ROI.

(b) Fibre bundles.

Figure 5.9.: The context can be used for deterministic fibres. Figure 5.9a contains fibres running through a region of interest in the centre of the dataset. Figure 5.9b contains four fibre bundle datasets: Forceps Major (pink), Forceps Minor (orange), Cingulum (green), and Cortico Spinal Tract (blue).

bilities. As expected, the deterministic fibres are most dense where the probabilities are high, and they grow sparser with decreasing probability.

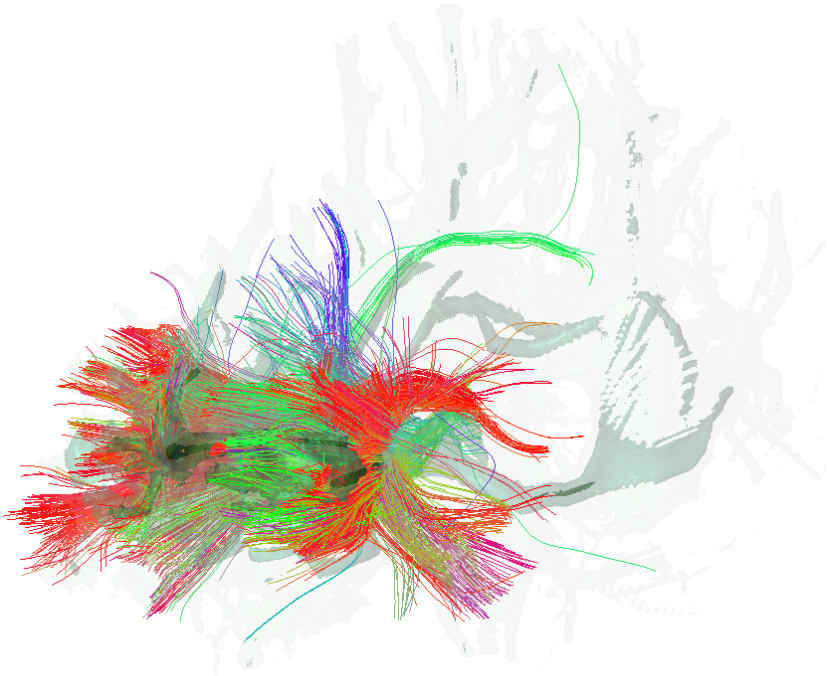


Figure 5.10.: Probabilistic tracts and some deterministic fibres running through the region around the seed point.

6. Conclusion

Before I conclude this thesis with an outlook of future work, I will summarise and discuss the results of this work.

6.1. Summary

As we have seen in the previous chapters, I successfully developed a very flexible visualisation for probabilistic tractograms that achieves the goal to provide a very good impression of the connectivity, shape, and value distribution of the dataset. The overview over the nature and connectivity of the data is achieved by rendering up to four surfaces for interactively chosen values. The user is assisted in understanding the value distribution by employing adaptive transparency, while the shape is enhanced by applying depth-dependent saturation. This also serves to acquire an understandable representation for two-dimensional images such as print.

The visualisation is complemented by a "glass brain" as anatomical context, which meets the goal of being fundamentally meaningful and unobtrusive. It is rendered using normal-dependent shading to darken and enhance the sulcii, and to brighten and fade the gyrii, which creates a clear drawing of the cortex. In order to keep the balance between visualisation and context, i.e. reduce obtrusiveness, the transparency relies upon the area of focus. I supplemented the context with fully integrated slices, which can be moved interactively in axial, coronal and sagittal direction. This also serves to provide more meaningful information. Like the visualisation, the context is very flexible and can therefore not only be used for other probabilistic tractogram datasets, but also to supply context for other brain-related visualisations.

Overall, the methods I presented in this work have several advantages over existing techniques. First and foremost, the main goal of visualisation is met: aiding the user to comprehend the data and its connection to related information, i.e. brain anatomy. The three-dimensional representation of the data is very intuitive, since the sense of depth is enhanced similar to what the user is familiar with: the colours near the viewer are lighter and more saturated, while those further away are darker or fade into grey. In particular, the saturation corresponds to a typical trait of landscape photography, as the colour of mountains in the background is dulled by the atmosphere. As the program is implemented as a shader, the computations are extremely quick. Therefore, it is possible to use the program in real-time, to view the visualisation from different angles and manipulate it immediately. Finally, both modules I developed are very flexible in order to match the variance between different datasets, and grant the user more choice. With so much flexibility, it is vital to assist the user in adapting

the visualisation to the task. This is achieved by offering a set of values initially, and equipping the user with sliders to interactively and intuitively modify the values.

However, the methods also have some limitations. The most significant one lies in the quality of the datasets. For example, if the tractography dataset is not registered correctly to the structural MRI, it can protrude the cortex as seen in Figure 5.7a. It would be possible to clip the parts of the tractogram that are not possible in nature, but if this was done automatically, it could also produce wrong results. Due to the low resolution, the datasets are interpolated. While this removes unwanted sharp edges, it also increases the risk of losing information. Another limitation lies in the use of scalar fields, since they can only represent the amount of diffusion, but not its direction. Finally, the anatomical context is limited to the cortex, and slices displaying grey matter, white matter and ventricles.

Some of the limitations can be removed by future work.

6.2. Future Work

As explained in Section 2.1.3, the probabilistic tractograms are derived from DTI data, i.e. a corresponding tensor field. This field could provide an interesting contribution, since it contains the diffusion direction, and not only its probability. Therefore, it is possible to texture the surfaces according to the diffusion direction, in a similar way to what Hummel et al. [12] and Eichelbaum et al. [6] presented. The diffusion direction could be represented by lines since the interest lies in which parts are connected, and not in the flow orientation. To represent the probabilities, the level of sparseness could be varied. However, an important factor one should keep in mind is avoiding visual clutter, which may be difficult due to the strongly folded, detailed structure of the datasets. If the amount of clutter is acceptable while only one colour is used, one could try to combine the visualisation with an illustrative line-drawing context such as the one presented by Svetachov et al. [23].

In the previous sections, I have pointed out that it is difficult to select suitable isovalues, and that it is left to the user to make a choice. However, given sufficient time for pre-processing, it may be possible to utilise the methods to automatically choose the most representative isovalues, which Bruckner et al. [4] developed.

Alternatively, one could extract isosurfaces beforehand. Again, this requires a significant amount of pre-processing time, or even manual segmentation, before the user can start the actual analysis of the data. However, pre-calculated surfaces would significantly speed up the computation, so it could also be used on rather limited systems, like laptops or older machines.

While automatic isovalue selection or segmentation of the tractograms may help the user, they also require some additional time and effort. A very valuable application of prior segmentation lies in the context. So far, the "glass brain" is drawn as a user-chosen isosurface through the cortex. A prior cortex extraction could make the result more accurate. Slices textured with structural MRI images, as used in this thesis, can provide vital anatomical cues to medical staff, especially if the ventricles are clearly visible. However, there is a lot more information

that can be integrated as anatomical context. For surgical planning, it would be very helpful to render blood vessels, since they must not be severed. Other possibilities would be to supply even more context by rendering the lymphatic vessels, or a volumetric representation of the ventricular system.

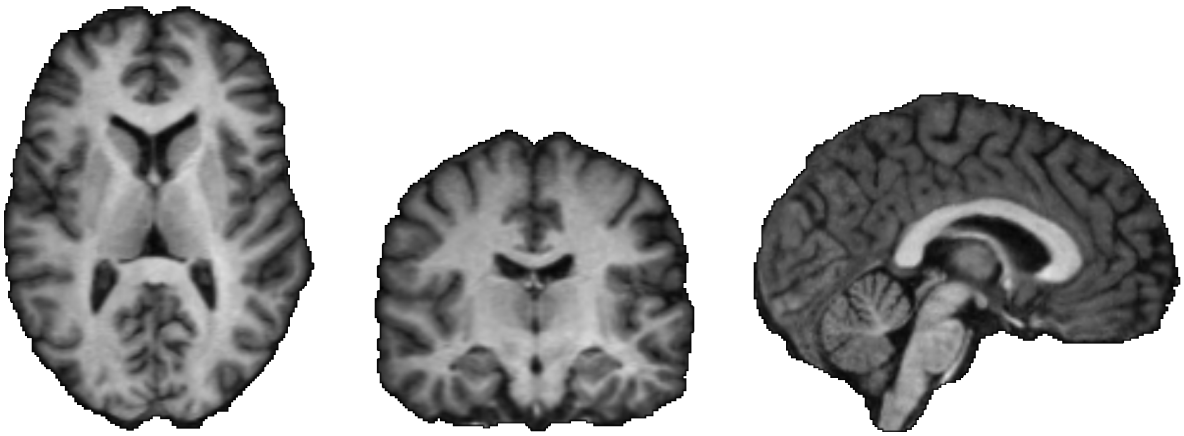
Last, the visual appeal of the renderings in this thesis could be polished by a careful registration of the DTI data to the structural MRI data. This would help to avoid that parts of the visualisation protrude the context.

Overall, I conclude that despite having achieved major improvements compared to previous works, there are still a lot of possibilities to enhance the results of this work.

A. Appendix: Data

All data, which was used in this thesis comes from a single test subject. It was acquired with a 3-Tesla scanner Siemens 3T Trio at the Max Planck Institute for Human Cognitive and Brain Sciences in Leipzig.

The T_1-weighted MRI dataset used for the context has a resolution 160 x 200 x 160 voxels, which corresponds to a voxel with an edge length of one millimetre. Originally, it contained a scan of the subject's whole head, as seen in Figure 2.2. However, for further use, it was clipped to contain only the brain. In Figure A.1, some slices from the principal directions are presented.

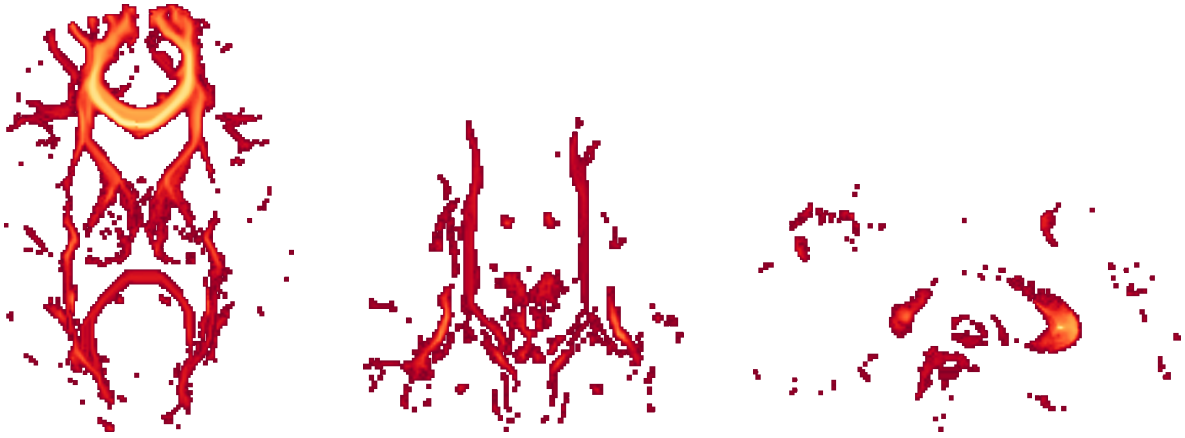


(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure A.1.: The axial, coronal and sagittal T_1 slices through the centre of the dataset. A high proton density is represented by light grey.

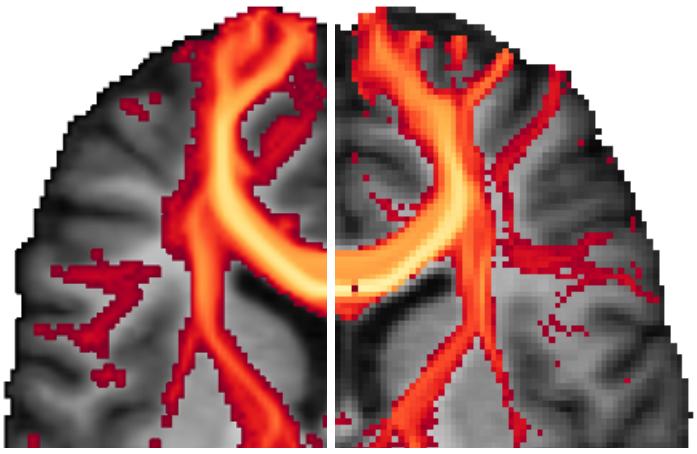
The DTI dataset consists of second order tensors, which were estimated from an error corrected Diffusion MRI with 60 gradients. Registration to structural MRI was carried out linearly. It has a significantly lower resolution than the MRI data with an edge length of 1.72 millimetres for each voxel, so each slice voxel is about five times as big as a structural MRI voxel. The probabilistic tractograms were derived using MedINRIA. Figure A.2 displays some slices of the probabilistic tractogram, which were computed with a random walk method, using the Lipsia toolbox.

Both figures (A.1 and A.2) display the interpolated version of the slices, since this is how the data was used in this thesis. The sharp colour changes between the voxels imply edges that do not exist. These edges are smoothed by trilinear interpolation. The impact of the smoothing is demonstrated in Figure A.3.



(a) Axial view. (b) Coronal view. (c) Sagittal view.

Figure A.2.: An axial, coronal and sagittal slice of the probabilistic tractogram through the centre of the dataset. High probabilities are represented by bright yellow and white, while low probabilities are dark red or transparent.



(a) Interpolated. (b) Raw.

Figure A.3.: Comparison between the interpolated data and the raw data in an axial slice.

B. Appendix: Code

The basic framework for the modules is based on an isosurface ray caster, which is an OpenWalnut module. In this appendix, I only provide shortened versions of the main and the ray casting function. However, the complete implementation is available along with the digital copy of my thesis.

In the main function, I first set the basic fragment colour to an entirely transparent white. Then, the step distance is determined depending on the number of steps chosen by the user, and the current point in the local coordinate system is computed. Finally, the ray caster is called.

```
main()
{
    wge_FragColor = vec4( 1.0, 1.0, 1.0, 0.0 );
    stepDistance = maxDistance / float( u_steps );
    // current point in local coordinate system
    vec3 curPoint = v_rayStart + v_ray;
    // call the ray caster
    rayCast( curPoint, u_isovalue, u_isocolour );
}
```

The ray caster is a lot more complex than the main function. For each point along the ray, it checks, if it is equal (or sufficiently similar) to the isovalue. If this is the case, the colour of the current pixel (or fragment) is modified; otherwise, the point is moved to the next position on the ray.

Now, the standard pipeline is followed in six stages. In the first three stages, the point is transferred and scaled to screen space, and its depth value is computed. Then, the fragment shading is computed with uniform light or Phong shading, depending on the user's choice. In the fifth stage, the colour is determined from a colour map (`mapcolour`) and an interactively chosen colour (`isocolour`). In the end, the shading is applied to the colour, but not to its transparency. After this final step, the program stops.

This program is carried out for every pixel. Since it is carried out on the GPU, it is extremely fast thanks to massive parallelisation.

```
void rayCast( in vec3 curPoint, in float isovalue, in vec4 isocolour )
{
   for( int j = 1; j < u_steps; j++ )
   {
      // get current value
      float curValue = texture3D( u_texture0Sampler, curPoint ).r;

      // is it the isovalue?
      if( abs( curValue - isovalue ) < u_isovalTolerance )
```

```
        {
            // we need the depth value and colour of the current point inside
            // the cube -> standard pipeline

            // 1. transfer to world space and then to eye space
            vec4 curPointProjected = gl_ModelViewProjectionMatrix *
                                     vec4( curPoint, 1.0 );

            // 2. scale to screen space and [0,1]
            // don't need x and y
            curPointProjected.z /= curPointProjected.w;
            curPointProjected.z  = curPointProjected.z * 0.5 + 0.5;

            // 3. set depth value
            gl_FragDepth = curPointProjected.z;

            // 4. shading
            // find normal for a headlight in world coordinates
            vec3 normal = ( gl_ModelViewMatrix *
                            vec4( getNormal( curPoint ), 0.0 ) ).xyz;
            #ifdef WGE_POSTPROCESSING_ENABLED
            wge_FragNormal = textureNormalize( normal );
            #endif
            // full brightness
            float light = 1.0;
            #ifdef PHONGSHADING_ENABLED
            // only calculate the phong illumination only if needed
            light = blinnPhongIlluminationIntensity( normalize( normal ) );
            #endif

            // 5. set colour
            // get colour from colourmap (alpha is set to one since we use the
            // isovalue's alpha value here)
            vec4 mapcolour = colourmapping( vec4( curPoint.x * u_texture0SizeX,
                                                  curPoint.y * u_texture0SizeY,
                                                  curPoint.z * u_texture0SizeZ,
                                                  isocolour.a ) );
            // mix colour with colourmap
            vec4 colour = mix( mapcolour, isocolour, 1 - u_colourmapRatio );
            colour.a = isocolour.a;

            // 6: the final colour construction
            wge_FragColor.rgb = vec4( light * colour.rgb, colour.a );

            break;
        }
        else
        {
            // no it is not the iso value -> continue along the ray
            curPoint += stepDistance * v_ray;
        }
    }
}
```

Bibliography

[1] T.E.J. Behrens and S. Jbabdi. *Diffusion MRI: from quantitative measurement to in-vivo neuroanatomy*, chapter 15: MR Diffusion Tractography. Academic Press, 2009.

[2] M. Bender and M. Brill. *Computergrafik: Ein anwendungsorientiertes Lehrbuch*. Hanser Verlag, 2006.

[3] S. Born, W. Jainek, M. Hlawitschka, G. Scheuermann, C. Trantakis, J. Meixensberger, and D. Bartz. Multimodal Visualization of DTI and fMRI Data Using Illustrative Methods. *Bildverarbeitung für die Medizin 2009*, pages 6–10, 2009.

[4] S. Bruckner and T. Möller. Isosurface Similarity Maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[5] T.M. Deserno. *Biomedical Image Processing - Biological and Medical Physics, Biomedical Engineering*, chapter 1: Fundamentals of Biomedical Image Processing. Springer-Verlag Berlin Heidelberg, 2011.

[6] S. Eichelbaum, M. Hlawitschka, B. Hamann, and G. Scheuermann. Fabric-like visualization of tensor field data on arbitrary surfaces in image space. 2010. Submitted to Dagstuhl Seminar 09302.

[7] M.H. Everts, H. Bekker, J.B.T.M. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1299–1306, 2009.

[8] M. Goldau, A. Wiebel, N.S. Gorbach, C. Melzer, M. Hlawitschka, G. Scheuermann, and M. Tittgemeyer. Fiber Stippling: An Illustrative Rendering for Probablistic Diffusion Tractography. *IEEE BioVis 2011 Proceedings*, 2011. Online Submission ID 129.

[9] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Prentice Hall International, 3rd international edition, 2008.

[10] H. Gray. *Anatomy of the human body*. Lea & Febiger, 20th edition, 1918.

[11] W.R. Hendee and C.J. Morgan. Magnetic Resonance Imaging Part I - Physical Principles. *Western Journal of Medicine*, 141(4):491, 1984.

[12] M. Hummel, C. Garth, B. Hamann, H. Hagen, and K.I. Joy. IRIS: Illustrative Rendering for Integral Surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1319–1328, 2010.

[13] W.M. Jainek, S. Born, D. Bartz, W. Straßer, and J. Fischer. Illustrative hybrid visual-

ization and exploration of anatomical and functional brain data. In *Computer Graphics Forum*, volume 27, pages 855–862. Wiley Online Library, 2008.

[14] D.K. Jones and M. Cercignani. Twenty-five pitfalls in the analysis of diffusion mri data. *NMR in Biomedicine*, 23(7):803–820, 2010.

[15] G. Kindlmann. Superquadric tensor glyphs. In *Proceedings of Symposium on Visualization*, volume 2004, 2004.

[16] Lighthouse3D.com. Pipeline overview: Visual summary of the fixed functionality. http://www.lighthouse3d.com/tutorials/glsl-tutorial/pipeline-overview/, September 2011.

[17] S. Mori. *Introduction to Diffusion Tensor Imaging*. Elsevier Science, 1st edition, 2007.

[18] Y. Mrabet. Planes of human anatomy. http://en.wikipedia.org/wiki/File:Human_anatomy_planes.svg, June 2008.

[19] MadSci Network. Guided Tour of the Visible Human: Planes of Section. http://www.madsci.org/~lynn/VH/planes.html, aug 2011.

[20] B. Preim and D. Bartz. *Visualization in Medicine - Theory, Algorithms, and Applications*. Morgan Kaufmann, 2007.

[21] T. Rick, A. von Kapri, S. Caspers, K. Amunts, K. Zilles, and T. Kuhlen. Visualization of probabilistic fiber tracts in virtual reality. *Medicine Meets Virtual Reality*, 18:486–492, 2011.

[22] R.J. Rost. *OpenGL Shading Language*. Addison Wesley, 2nd edition, 2006.

[23] P. Svetachov, M.H. Everts, and T. Isenberg. DTI in Context: Illustrating Brain Fiber Tracts In Situ. *Computer Graphics Forum*, 29(3):1023–1032, 2010.

[24] I. Viola. *Importance-Driven Expressive Visualization*. PhD thesis, Universität Wien, 2005.