STEVEN BOUMA

DTI FIBER TRACT SIMPLIFICATION AND VISUALIZATION

DTI FIBER TRACT SIMPLIFICATION AND VISUALIZATION

STEVEN BOUMA

university of groningen / faculty of science and engineering

MSc. Thesis in Computing Science

Faculty of Science and Engineering
University of Groningen
June 2018 – version 0.6 [FINAL DRAFT]

First supervisor: **prof. dr. A.C. Telea**
Second supervior: **J.F. Kruiger**

I think the main thing a musician would like to do is give a picture to the listener of the many wonderful things he knows of and senses in the universe..
That's what I would like to do.
I think that's one of the greatest things you can do in life, and we all try to do that in some way. The musician's is through his music.

— John Coltrane [9]

ABSTRACT

DTI fiber tracking is the process of approximating neural pathways in the human brain by tracing streamlines using reconstructed diffusion tensors and estimates of anisotropy obtained through diffusion tensor imaging (DTI). In this thesis, we propose methods for the geometric simplification and visualization of complex DTI fiber tract structures, in order to convey a clear impression of the main structure, shape, and topology of neural pathways.

Our contribution is two-fold: First, we demonstrate an approach to the simplification of DTI fiber tracts that is based on CUBu edge bundling. Our simplification approach shows promising computational performance through GPU parallelization, and allows various ways to control the simplification process, leading to a clearer impression of the type of anisotropy within the fiber structures.

Secondly, we propose a new illustrative rendering method for DTI fiber tracts, and compare this method to several other pre-existing methods. Our work was tested using DTI datasets obtained from a variety of subjects.

We have to abandon the idea that schooling is something restricted to youth. How can it be, in a world where half the things a man knows at 20 are no longer true at 40 - and half the things he knows at 40 hadn't been discovered when he was 20?

— **Arthur C. Clarke** [72]

ACKNOWLEDGMENTS

First of all I would like to thank Gera & Sookie for unwavering support and cuddles..

I would also like to express my sincere gratitude to Alex Telea for his patience, elaborate feedback and encouragement.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ABP Alpha Blended Pointsprites

AD Alzheimer's Disease

ADT Apparent Diffusion Tensor

CSF Cerebrospinal Fluid

DDH Depth Dependent Halos

DTI Diffusion Tensor Imaging

DWI Diffusion Weighted Imaging

DW-MRI Diffusion Weighted MRI

FMRI functional Magnetic Resonance Imaging

FPS Frames per Second

GPU Graphics Processing Unit

GM Gray Matter

KDE Kernel Density Estimation

MRI Magnetic Resonance Imaging

MVP Model-View-Projection

NDC Normalized Device Coordinates

NPR Non-Photorealistic Rendering

OIT Order-Independent Transparency

PCA Principal Component Analysis

ROI Region of Interest

RQ Research Question

SNR Signal-to-Noise Ratio

SRS Stereotactic Radiosurgery

TCS Tessellation Control Shader

TES Tessellation Evaluation Shader

WM White Matter

1 INTRODUCTION

DTI is a neuroimaging technique based on Magnetic Resonance Imaging (MRI) which makes it possible to estimate the location and orientation of white matter tracts in the human brain in vivo. By measuring the diffusivity of water molecules through the process of Diffusion Weighted Imaging (DWI) it is possible to estimate diffusion tensors and implicitly the anisotropy of the diffusion data. Clinical applications of DWI and DTI include the localization of tumors or white matter lesions and pre-operative assessment (surgical planning).

DTI fiber tracking is the process of approximating neural pathways by tracing streamlines using reconstructed diffusion tensors and estimates of anisotropy obtained through DTI. The fiber tracking process can produce large sets of dense polylines that are challenging to visualize due to their scale and structural complexity. Traditionally, filtering and selective fiber tracking in a Region of Interest (ROI) have been applied to limit the extent of the data to more manageable quantities. A more recent set of techniques such as graph- and trail bundling, that are based on the notion of geometric simplification, allow for a different approach: by simplifying dense sets of fiber tracts it becomes possible to present whole-brain tractography in a more insightful manner.

The most common approach to fiber tractography is to trace streamlines in the major diffusion direction since this is a strong indicator for the orientation of white matter fibers. There are however many areas within the human brain where diffusion is non-linear, but exhibits for instance plane-like diffusion[2, 53, 62]. These structures may occur in areas with e.g. crossing or branching fibers, which conventional fiber tracking does not convey very well. Alternative approaches to conventional fiber tracking include stream-surface tracking [67] and multi-tensor tractography methods such as UKF-tractography [35]. Although these methods also provide more insight into regions of higher planar anisotropy, output produced by such methods may still suffer from occlusion.

High density fiber tractography of the entire brain can be a requirement depending on application. For instance, in a healthy subject where the anatomy is known, the location of interesting bundles and structures can reasonably be determined. However, in patients, existing insight about possible underlying structure of the brain may not apply and therefore manual seeding of fiber tracts can miss important structures [67]. Rather than placing seedpoints selectively, or by filtering tracts, it is therefore more suitable to trace an entire DTI image in order to create the largest possible coverage of the brain. Consequently this produces very dense path data with a lot of occlusion (or clutter). . Hence rises the possible need for a simplified representation of whole-brain tractography data that exhibits reduced occlusion.

Visualization and analysis of (planar) anisotropy is of use in topics where the under-

lying structure of neural fiber tracts is of interest. Teipel et al. [63] for example have linked Alzheimer's Disease (AD) dementia to a loss of crossing fibers, and Speckter et al. [59] have indicated that planar anisotropy is a predictor for the responsiveness of meningiomas to Stereotactic Radiosurgery (SRS). Wedeen et al. [70] have proposed that fiber pathways in the human brain form sheet structures, a phenomenon that is actively studied as the organizing principles of cerebral connectivity remain unclear. Developing adequate visualization methods could provide more insight in the underlying structure of the human brain and facilitate further research in this area.

We propose a two-fold approach to clearly showing structural information of dense DTI fiber tracts that is based on simplification of these tracts in conjunction with a rendering method. Geometric simplification of the fiber tracts is applied to reduce occlusion and adequate rendering can complement this to clearly convey structural information and data attributes within the simplified set of fiber data.

1.1 PROBLEM STATEMENT

Tensor fields, such as for example emerging from DTI scanning techniques, are commonly visualized by two main techniques: streamlines (fiber tracts) and glyphs. Both techniques have specific advantages and disadvantages [64], but a major problem for both glyphs and fibers is occlusion.

A relatively recent set of techniques for reducing occlusion relates to the idea of geometric simplification of the spatial structure created by either fibers or glyphs. The aim of such simplification is to reduce the adverse effects of occlusion and clutter, but to still convey a clear impression of the main structure, shape, and topology existent in the DTI data. Within geometric simplification, bundling techniques have proven recently high potential: they can indeed reduce clutter by effectively '*thinning*' a fiber-set; they can be combined with glyphs to yield a more compact, and therefore more clutter-free and easier to understand, rendering of the shape of the fiber-set; and they can be controlled in various ways to limit and/or guide the amount of deformation.

Structural simplification also requires adequate complementary rendering, for which there is a variety of methods available. Even when the shapes embedded in a DTI dataset are approximated by a (simplified) set of fibers, it is possible to represent (render) these shapes by drawing more than the set of fibers itself. Essentially, this bypasses the modeling limitations of a set of $3D$ curves, by adding extra information to them during rendering, so they better approximate the underlying data and shapes. Current examples hereof include the depth-dependent halos in [15, 17]. The key question to address here is: how can we adapt/extend these mechanisms to convey a clearer or better impression of the type of anisotropy the simplified shapes have. In other words how to render curved surfaces, tubular fiber bundles, and individual 'stray' fibers respectively, so that these different structures are easier to recognize in the simplified visualization?

1.1.1 *Research Question*

Based on the above problem statement, we define the Research Question (RQ) to be: *"How can we depict a large and dense set of 3D paths obtained by tractography in a simplified way, so that key structures are easy to see?"*

We address the RQ along two technical axes: *simplification* and *rendering*. To establish a framework for our research we have defined several requirements for both technical axes. We will refer to these requirements throughout the document as they are fundamental to the explanation of rationale and also allow qualitative assessment of the proposed methods.

1.1.2 *Simplification Requirements*

We define two main requirements by which we validate the simplification process. The simplification should:

- S1: Reduce occlusion in dense sets of fiber tracts, while preserving important structures.
- S2: Quickly generate a simplified model, ensuring that interactive exploration of the data is possible.

1.1.3 *Rendering Requirements*

Once simplification is performed, an appropriate rendering method is required to display the simplified model. Based on analysis of related DTI visualization methods and the properties that makes these methods particularly suitable for DTI visualization (see also section 2.4), we define four (functional- and non-functional-) requirements, by which we compare existing and new rendering approaches.

In order to complement the simplification process, a rendering method should:

- R1: Show the nature of local brain structures. Tube-like structures should be rendered as such and surface-like structures should be rendered as clear surfaces without gaps.
- R2: Clearly convey shape and orientation of fiber structures. The method should show convexity and concavity of structures and allow a user to distinguish the orientation of different structures with respect to each other.
- R3: Allow depiction of data attributes through e.g. scalar color coding.
- R4: Allow rendering at interactive framerates.

1.2 THESIS STRUCTURE

The structure of this document is as follows; in the related work chapter (chapter 2) we describe DTI, fiber tractography and visualization methods related to our research.

Chapter 3 covers fiber tract simplification based on edge bundling and consecutively a new DTI rendering method is discussed in chapter 4. An overview of results, including various approaches to rendering is presented in chapter 5. Finally, in chapter 6 we discuss these results and compare several DTI fiber tract rendering methods, and in particular how these are applicable to the simplified fiber tract model. We conclude our work in chapter 7 in which we summarize our findings and present directions for further research.

2 RELATED WORK

In order to place our work on fiber tract simplification and visualization into context, we describe various topics related to DTI, DTI-visualization and edge bundling in this chapter. In section 2.1 we discuss basic fundamentals of DTI and how this is derived from DWI. Next, in section 2.2 we discuss tensor shape classification which provides metrics that are essential to our methods. Several related DTI visualization techniques that are conceptually similar to our approach are discussed in section 2.4. Finally, this chapter is concluded with an overview of current edge bundling applications in the area of functional- and anatomical connectivity visualization of the human brain.

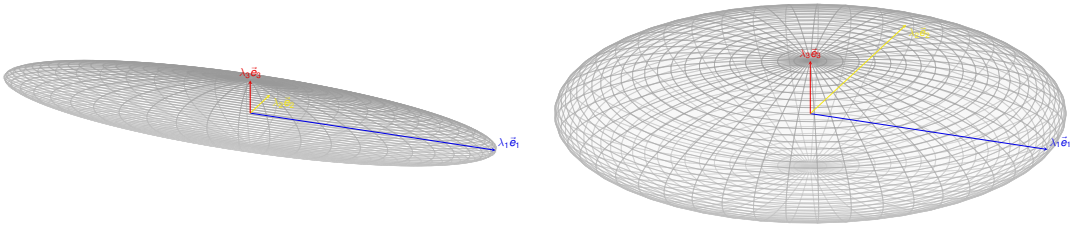
2.1 DIFFUSION TENSOR IMAGING

Conventional structural imaging techniques such as T1-, T2- and proton density–weighted imaging generally create high contrast between major tissue groups in the brain, which are: Gray Matter (GM), White Matter (WM), and Cerebrospinal Fluid (CSF). Such structural imaging techniques are generally well suited for the study of tissue macrostructure yet provide little insight into the *orientation* of white matter fibers.

DWI [32, 38] is a variant of conventional MRI based on the tissue water diffusion rate, which is better suited for the study of white matter pathways. While DWI refers to the contrast of the acquired images, DTI is a specific type of modeling (or abstraction) of the DWI datasets, in which diffusivity of water molecules is represented by tensors[28]. As an in-vivo non-destructive technique that requires no chemical tracers, DTI is presently one of the most promising methods for the study of white matter architecture in living humans [56].

DTI provides quantitative estimates of white matter integrity and orientation by measuring molecular diffusion of water molecules (or Brownian motion). It is based on the phenomenon of diffusion anisotropy in the nerve tissue: water molecules diffuse faster along the neural fiber direction and slower in the fibertransverse direction [30]. WM neural fibers consist of bundles of myelin covered axons of which the location and orientation can be approximated by measuring the diffusion from multiple directions through DWI.

In the single-tensor DTI model, diffusion at each voxel within the imaging volume is represented by a 3×3 diffusion tensor matrix (also known as a Hessian matrix, Eqn. 2.1) and is often referred to as the Apparent Diffusion Tensor (ADT)[28]. The term *apparent* is conventionally applied to emphasize the fact that the tensor is derived from observed (or measured) diffusivity, which is subject to measurement conditions [42].



(a) Linear diffusion ($\lambda_1 = 0.75, \lambda_2 = 0.15, \lambda_3 = 0.1$).

(b) Planar diffusion ($\lambda_1 = \lambda_2 = 0.45, \lambda_3 = 0.1$).

Figure 2.1: Diffusion tensor ellipsoids.

As the frame of reference of the ADT corresponds to the MRI scanner (also referred to as the laboratory frame of reference) and therefore does not represent the local orientation of anatomical structures, this is generally transformed to a representation independent of any coordinate system through Principal Component Analysis (PCA)[64], also known as tensor reconstruction. Standard tensor reconstruction algorithms include linear least squares and the potentially more accurate weighted linear least squares method [66].

The resulting rotationally invariant representation of the ADT is commonly visualized as an ellipsoid (shown in figure 2.1) defined by three eigenvectors (the principal axes) and three eigenvalues. The eigenvectors $\boldsymbol{e_1}$, $\boldsymbol{e_2}$ and $\boldsymbol{e_3}$ are mutually orthogonal unit vectors that represent directions in which the diffusion has extremal value (also known as principal directions) proportional to the corresponding eigenvalues λ_1, λ_2 and λ_3.

For each voxel within the imaging volume, direction $\boldsymbol{e_1}$ (the major eigenvector) is the direction of maximal diffusion by λ_1, and $\boldsymbol{e_3}$ (the minor eigenvector) is the direction of minimum diffusion by λ_3. This information allows for numerous tensor visualization techniques, most notably fiber tractography.

$$D = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{pmatrix} \tag{2.1}$$

The DWI acquisition process involves measurement of diffusivity by applying pulse sequences from multiple directions around a subjects body, at various b-values. This sequence of gradients (consisting of diffusion sensitizing directions and b-values) is referred to as the gradient table. To estimate the diffusion tensors, a minimum of seven diffusivity measurements is required (six diffusion directions and one $b0$-image for a non-diffusion-weighted baseline), yet modern scanning usually involves many more of these diffusion sensitizing directions to obtain more accurate results (see also appendix A.1 for datasets in the context of this document). Application of higher b-values produces more diffusion-weighted contrast but also reduces the Signal-to-Noise Ratio (SNR)[28].

DWI has low SNR, low resolution and is very susceptible to motion distortion [48], for this reason the image acquisition pipeline generally involves a sequence of preprocessing steps. These steps include algorithms for noise correction, motion correction, phase correction, and eddy-current correction [13, 20].

Although such additional processing can mitigate some of the distortions commonly affecting the DWI process, these are not always effective at completely removing all artefacts, and as such the SNR of DWI is relatively low despite the extensive corrections. Brain masking is a commonly applied method for DWI- and DTI datasets to limit the adverse effects of signal noise in areas outside the brain (see also section 3.1). As a postprocessing step, subject facial features are also commonly stripped or obscured to preserve subject anonymity[41].

The tensor information as obtained through PCA allows for a variety of useful quantitative parameters to be derived from it. Various of these DTI-derived parameters (described in the next section) that are based on the eigenvalues of the diffusion tensor ($\lambda_1, \lambda_2, \lambda_3$) provide measures of the degree of diffusion anisotropy. These scalar measures combined with the estimates of fiber direction provide a means to WM tractography, which leads to a unique multiscale description of tissue architecture and organization[28].

2.2 TENSOR SHAPE CLASSIFICATION

Westin et al. have defined a number of geometrical diffusion measures to describe the shape of tensors reconstructed from DWI [71]. Any tensor within the DTI dataset can be classified as being a linear-, planar-, or spherical diffusion-tensor (not mutually exclusive). These measures, that are based on the relation between the three ordered eigenvalues ($\lambda_1, \lambda_2, \lambda_3$ with $\lambda_1 \geq \lambda_2 \geq \lambda_3$) are defined as follows:

$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.2}$$

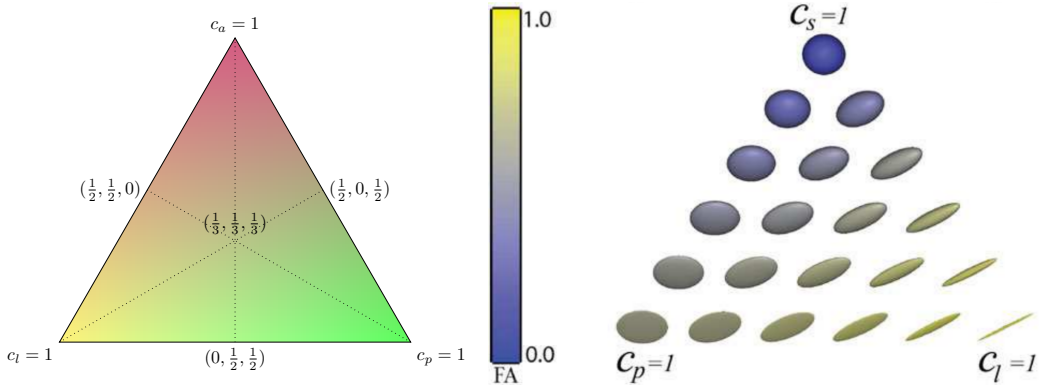
$$c_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.3}$$

$$c_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.4}$$

Equations 2.2, 2.3 and 2.4 are commonly referred to as the linear-, planar- and spherical certainty metrics. All three measures are normalized to the range of $[0, 1]$ and form barycentric coordinates [74] as is depicted in figure 2.2a.

From these estimates follows a measure for anisotropy (Eqn. 2.5) which can be used to indicate regions in the data where the diffusion is planar or linear (fiber-like) as opposed to isotropic (i.e. equal in all directions):

$$c_a = c_l + c_p = \frac{\lambda_1 + \lambda_2 + 2\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} = 1 - c_s \tag{2.5}$$



(a) Barycentric coordinates (c_a, c_l, c_p) depicted on an equilateral triangle.

(b) Tensor ellipsoids with *FA* colormap, image from[53].

Figure 2.2: Barycentric space of tensor shapes.

Fractional anisotropy is another widely used measure for anisotropy[4], and can be used as a measure for deviation from isotropic diffusion. Fractional anisotropy is defined as:

$$FA = \sqrt{\frac{1}{2}} \frac{\sqrt{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{2.6}$$

Figure 2.2b shows a clear example of how *FA* maps to the various tensor shapes. The image shows that highest values of *FA* apply to linear tensors (the *cigar* shaped ellipsoids) and that lowest *FA* values for the spherical ellipsoids, which represent more isotropic diffusion.

An important aspect of these equations (2.2),(2.3),(2.5),(2.6) is that the eigenvalues are ordered by convention ($\lambda_1 \geq \lambda_2 \geq \lambda_3$). For this reason the eigenvalues are also referred to as the major- (λ_1), medium (λ_2) and minor (λ_3) eigenvalues. The metrics listed above are generally very useful because the more complex tensors are reduced to much simpler scalar attributes.

Besides various clinical applications in medical image analysis[28], anisotropy measures are also essential to the reconstruction and visualization of neural pathways (white matter tractography). Fractional anisotropy (*FA*, Eqn. 2.6) for instance is a strong indicator for white matter and is therefore commonly used in fiber tractography seeding criterions (discussed in the next section). On the contrary, diffusion within CSF is unrestricted in all directions (isotropic, with $c_s \sim 1$) which is one of the strongest indicators for absence of neural fibers [56].

2.3 FIBER TRACTOGRAPHY

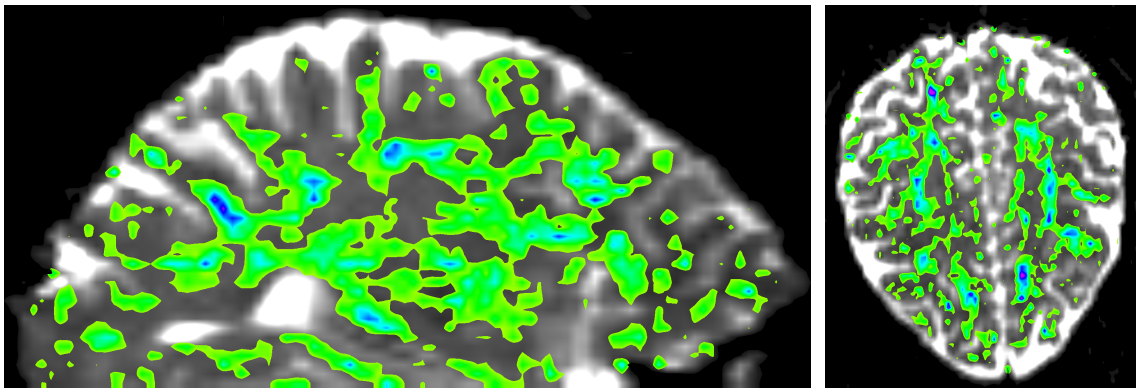
The orientation information of the diffusion tensor may be used to reconstruct estimates of white matter pathways in the human brain. The class of methods that performs such reconstruction is referred to as *white matter tractography*, *fiber tractogra-*

phy or *fiber tracking*.

From a meta perspective fiber tracking techniques can be categorized as deterministic or probabilistic methods. Deterministic methods use the directional information described by the diffusion tensor directly, assuming that this is the best estimate of the underlying fiber orientation, while probabilistic methods provide some degree of confindence or a measure for error in the tracking procedure[28].

A primary assumption of many tractography algorithms is that the direction of greatest diffusivity (the major eigenvector, e_1) is roughly parallel to the local white matter fiber bundle direction. The simplest tractography algorithms integrate streamlines in the direction e_1 [19, 43], yet more elaborate approaches include streamlines defined by tensor deflection[28], bi-tensor modelling [35] and streamsurface tracking[67].

Streamline propagation using only the direction of the major eigenvector has drawbacks since this approach is not able to accurately generate crossing or branching fibers. These fibers (illustrated in figure 2.4) occur in regions characterized by higher planar anisotropy that are present troughout the human brain. Figure 2.3 shows an example of such regions as indicated by the c_p measure (Eqn.2.3).



(a) Sagittal slice. (b) Axial slice.

Figure 2.3: Planar estimate (Eqn2.3) displayed on slices of DWI (grayscale). Values are thresholded, showing only regions of higher planarity ($c_p \geq 0.2$).

Tractography approaches vary in their methods for integration of the streamlines, seeding and stopping criteria, and in which constraints are applied on the streamline propagation. Research in the area of anatomical validation of fiber tractography has shown that tractography algorithms can produce anatomically correct results, yet are also limited by image noise, low SNR and crossing fibers.

Fiber tracking has several applications, including noninvasive visualization of white matter pathways, segmenting of specific tracts in the brain for image analysis, and relating white matter tract anatomy to brain tumors and lesions in patients who are candidates for neurosurgery[28].

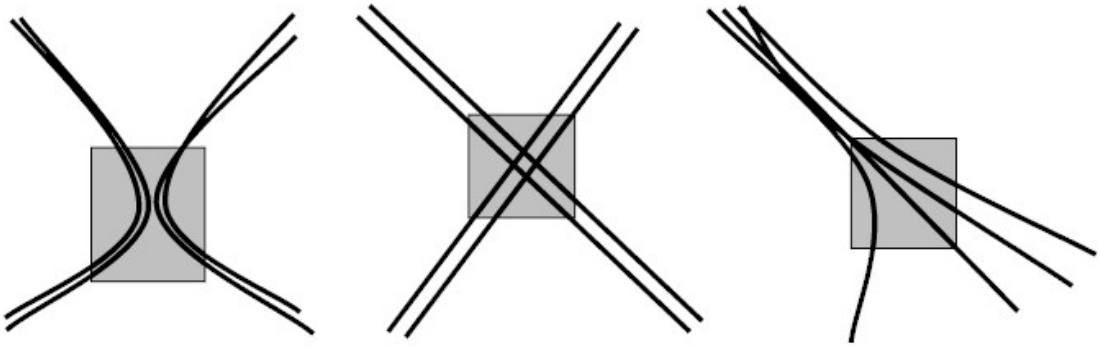


Figure 2.4: Viewer aligned illustration of fibers in regions where anisotropy is planar: (left) kissing fibers, (middle) two fiber bundles crossing and (right) diverging fibers. Gray regions have a tensor with planar diffusion. Image from [68].

It is important to note that DTI fiber tracts present a higher level abstraction of neural pathways rather than individual neural fibers. For example, axons in the human brain have a varying diameter of somewhere between $0.2\mu m$ and $20\mu m$ [1, 3], while current DTI typically produces a much larger voxel size in the order of a few mm^3 ($\sim 0.5..2mm^3$), and as such DTI represents generalized properties of *all* nerve tissue contained within a voxel.

2.4 DTI VISUALIZATION

Conventional line rendering is a simple and efficient baseline rendering technique that has been employed since fiber tracking was first introduced[43, 74]. More extensive illumination[45], ambient occlusion[11] and alpha blending[64] have been applied to line rendering of fiber tracts to further improve the shading of the lines.

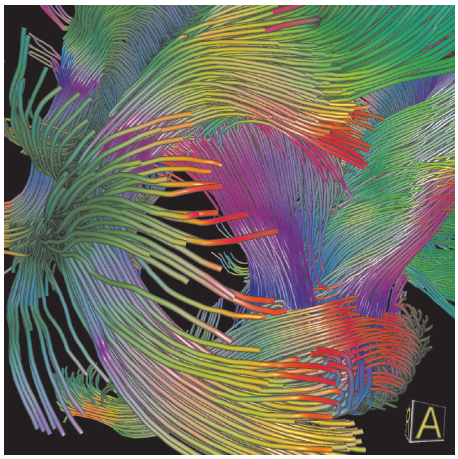
Rendering $3D$ tubes with Phong shading, such as the labeled tuboids by Petrovic, Fallon, and Kuester [46], is based on more detailed geometric modelling than line rendering, and can produce even higher quality visualizations. The methods by Stoll, Gumhold, and Seidel [60] and Merhof et al. [39] (shown in figure 2.5a) produce similar shading using screenspace techniques and incorporate illustrative rendering techniques as well. Hyperstreamlines [53, 64] are an extension of cylindrical tubes that provide a richer representation of the diffusion tensor field.

Various proposed methods for DTI- and fiber tract visualization are based on the notion of combining or merging geometry in order to show an enhanced image of the local nature of fiber bundles. These methods include: streamsurface tracking by Vilanova, Berenschot, and Pul (in which surfaces and tubes are combined, see figure 2.5c), merging ellipsoids [10, 57] (in which tubes or surfaces are formed by merging tensor ellipsoids), and Depth Dependent Halos (DDH) by Everts et al. [15] (which groups colinear fibers with halos).

Several of the previously mentioned related methods [15, 17, 60, 64] apply view-aligned rendering (or billboarding) in the context of fiber tract visualization, which is a concept that is also applied by our proposed method (further explained in chapter

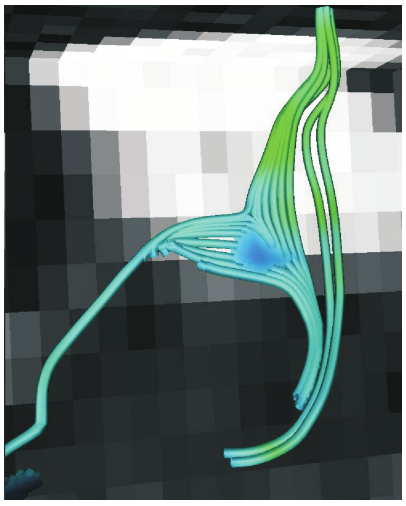
4). Illustrative techniques such as the rendering of silhouettes[44], contours, outlines and halos[15, 60] have also been proposed to provide additional visual cues in the context of complex fiber tract visualization.

Surface reconstruction or extraction from DTI datasets have also been applied as a means for indirect volume visualization[53]. For instance Merhof et al. [40] have shown that encompassing fiber tract bundles with isosurfaces yields a more preferable representation for use in neurosurgery, and *FA* ridge and valley surfaces [53] are demonstrated to capture the cores of sheet-like tracts.



(a) Hybrid visualization for white matter tracts, image from [39].

(b) Depth-Dependent Halos, image from [15].



(c) DTI Visualization with Streamsurfaces, image from [67].

(d) Labeled tuboids, image from [46].

Figure 2.5: Various related DTI fiber tract visualization methods.

2.5 EDGE BUNDLING AND FIBER TRACT SIMPLIFICATION

Edge bundling is a mechanism for geometric simplification of large sets of dense path data, such as large scale graph drawings. By applying edge bundling techniques to complex data, it is possible to generate a simplified view that shows underlying structures in an easier to understand manner. Such simplification can lead to new insight as certain features of the data become more apparent. Applications of edge bundling include graph drawing simplification[23, 75], trajectory exploration[34], eye-tracking analysis[47] and streamline bundling[73].

Image-based edge bundling techniques, including SBEB[12], ADEB[47], KDEEB[23], CUBu[75], and FFTEB[33], are amongst the most performant bundling algorithms[34]. In this class of methods, CUBu [75] is presently one of the fastest proposed edge bundling algorithms and has been demonstrated to achieve sub-second running times for large scale $2D$ graphs. Through Graphics Processing Unit (GPU) parallelization, the algorithm surpasses the computational performance of related methods presented so far[75]. Besides the GPU-based design that addresses scalability, Zwan, Codreanu, and Telea [75] have shown various visual styles, directional bundling and relaxation[24] as examples of how additional constraints can be incorporated to control the shape of resulting bundles.

Edge bundling has also been applied in the context of DTI and fiber tract visualization as a method to simplify dense sets of fiber tracts or connectivity data. For example, Bottger et al. [8] have proposed mean-shift bundling of $3D$ connectivity graphs obtained through functional Magnetic Resonance Imaging (fMRI)[7, 8]. Simplification of anatomical connectivity using edge-bundling is a relatively new topic, and one example thereof is KDEEB[23] applied to fiber tracts [25, 64]. Other examples of fiber tract simplification methods include: *multi-scale local fiber tract contraction* by Everts et al. [16] and *two-dimensional neural maps* by Jianu, Demiralp, and Laidlaw [27].

2.6 CURRENT CHALLENGES

The main limitations of current methods that are related to our goal of exploring and visualizing dense sets of DTI fiber tracts are in the areas of simplification and rendering.

As listed in the previous section, there are various edge bundling methods for general $3D$ polylines and some for fiber tracts specifically. These methods are generally not adapted in order to recover *plausible* anatomical structures. Instead, these bundling methods often produce $1D$ bundles, and create gaps in key areas. Control of the fiber tract bundling is therefore an important challenge.

Computational scalability of the proposed simplification is another important concern, as geometric simplification is generally a computationally extensive process. Another major challenge is therefore to allow *interactive* exploration of all simplification parameters. GPU parallelization of algorithms has shown promising results, yet

not all of these have been applied to $3D$ simplification.

Control of the rendering, so that the emerging structures are as close as possible to suggesting the *'underlying'* anatomical elements, such as bundles, sheets and manifolds of fibers is also not trivial using methods presented so far (see section 2.4). Various of the mentioned DTI visualization methods address this in one or more ways, yet to fulfill all requirements mentioned in subsection1.1.3 is challenging.

In section 4.1 we discuss several challenges associated with fiber tract rendering in detail, as part of an introduction to our rendering method presented in chapter 4, which we propose to address those challenges.

3

FIBER TRACT SIMPLIFICATION

As we have outlined in chapter 1, we aim at both simplifying large sets of DTI tracts, and rendering this simplification so that one can easily recover the (simplified) anatomical structures from them. In this chapter, we present our proposals for simplification (rendering is discussed separately in chapter 4).

Zwan, Codreanu, and Telea [75] have demonstrated the CUBu method to be a highly parallelizable algorithm, achieving sub-second running time for the bundling of large $2D$ datasets. We propose to extend this bundling method to $3D$ and to apply it in the context of DTI fiber tracts in order to simplify dense sets of fiber tracts.

Figure 3.1 shows the proposed simplification process as a part of the complete the visualization pipeline. As indicated in this diagram, the flow of data starts with the Diffusion Weighted MRI (DW-MRI) and is followed by several processing steps (*brain maksing*, *tensor reconstruction* and *fiber tracking*) which together yield the input dataset for the simplification process.

Next in the pipeline is our proposed simplification approach that consists of two aspects: *tract bundling*, which performs modification of the tracts, and *reseeding* which ensures sufficiently dense fiber placement in all regions of interest. Once simplification is performed, the visualization is concluded with a rendering step for which various possible approaches are available (further discussed in chapters 4 and 5).

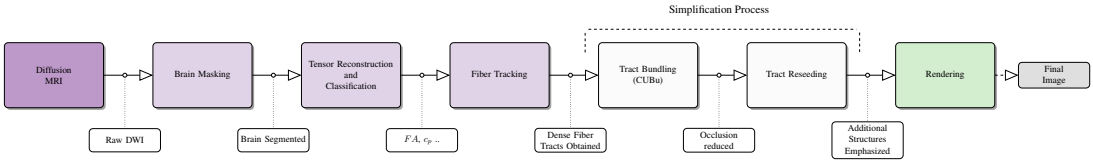


Figure 3.1: Flowchart of the visualization pipeline.

The structure of this chapter is as follows: prior to any fiber tracking or simplification, a brain mask must be obtained, this is briefly discussed in section 3.1. Next in section 3.2 we discuss the CUBu[75] bundling algorithm as the foremost means to our simplification approach. Bundling shape control is discussed in section 3.3, which covers various ways to control the bundling specifically in the context of DTI, these are endpoints bundling (subsection 3.3.3 and DTI edge profiles (subsection 3.3.4). Next to the bundling algorithm, our fiber tract simplification approach also relies on selective fiber tracking in specific regions of higher planar-anisotropy, this *reseeding* principle is discussed in section 3.4. The extension of the CUBu method to $3D$ introduces various options and trade-offs in terms of computational performance and memory requirements, these are discussed in section 3.5.

3.1 BRAIN MASKING

Indicating areas of brain tissue within the DTI dataset is an important prerequisite for fiber tracking and subsequent simplification (see also Fig. 3.1). Since the input volumes contain a variety of areas that are not used in the context of fiber tracking, but for which diffusion *is* measured, it becomes important to partition the data. Such regions include for instance bone tissue of the skull and extracorporeal areas (i.e. where there is no tissue), which should be excluded from the tractography- or tensor reconstruction process for the reason that these contain no nerveous tissue.

A simple and effective method to distinguish between relevant and irrelevant voxels within the dataset, is to apply a binary mask image ($M : \mathbb{R}^3 \rightarrow \{0,1\}$) that indicates brain tissue. On occasion a preconstructed mask is available as part of the input dataset (see appendix A.1), but otherwise this volume can be constructed from the DWI signal using a clustering-based image thresholding algorithm such as Otsu's method [54].

Once obtained, the mask image serves a two-fold purpose: first it is used the determine potentially suitable locations for seedpoints placement, thereby preventing the tracking of false fiber tracts in regions outside the brain. Secondly we can use the bounding volume indicated by the mask image, that is the smallest possible axis-aligned bounding box that will contain all voxels that indicate brain tissue, to optimize memory usage (this is discussed further in section 3.5).

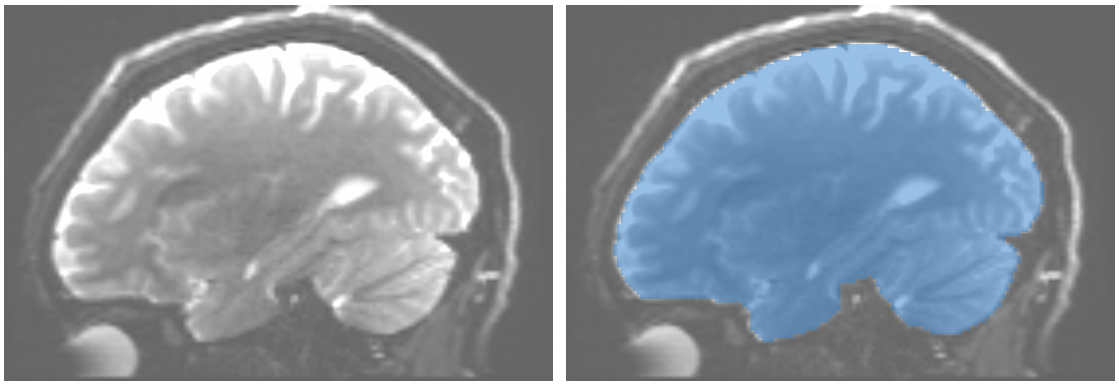
Since mask M is used for both the seeding- and stopping criterion during fiber tracking (explained earlier in section 2.3), areas for which $M \equiv 0$ are guaranteed to not contain any fiber tracts. We can therefore safely clip the space defined by the diffusion data based on the bounds indicated by mask M, without discarding any information that is relevant to the creation, simplification or visualization of fiber tracts.

As is listed in table 3.1 the maximum partition of relevant data for fiber tracking within the tested DTI volumes is approximately 25%, which is indicated by the percentage of voxels for which $M \equiv 1$. Obviously the organic shape of the human brain is not cuboid and some redundancy is therefore expected when sampling DTI data on a structured grid (in this case a cuboid volume). Clipping the volumes however does yield a significant size reduction of about 35..50% for our datasets, as is also listed in table 3.1.

	dataset #1	dataset #2	dataset #3	dataset #4
Volume dimensions	$128 \times 128 \times 94$	$96 \times 96 \times 50$	$140 \times 140 \times 96$	$140 \times 140 \times 96$
Masked brain bounds dimensions	$88 \times 111 \times 82$	$61 \times 82 \times 42$	$84 \times 98 \times 83$	$86 \times 116 \times 88$
% of voxels indicated as brain tissue	365980 / 1540096 voxels in mask (23.76%) of entire volume	113725 / 460800 voxels (24.68%) of entire volume	345004 / 1881600 voxels in mask (18.34%) of entire volume	384297 / 1881600 voxels indicated as brain by mask (20.42%) of entire volume
% of original size	47.99 % of original size	54.41 % of original size	63.69 % of original size	53.34% of original size
Size reduction %	52.01%	45.59%	36.31%	46.64%

Table 3.1: Brain masking metrics for all datasets.

Figure 3.2 shows a mask constructed from the DWI volumes of dataset #4 (see table A.1) on a sagittal slice. The mask that is visible in the right image (indicated in blue where $M \equiv 1$) clearly shows that the skull can be stripped by applying mask M.



(a) Sagittal slice of a DWI volume.

(b) Same image as 3.2a, with binary mask M overlayed in blue.

Figure 3.2: Binary mask constructed from a DWI signal, depicted on a sagittal slice.

3.2 FIBER TRACT BUNDLING USING CUBU

Starting with a set of fiber tracts (a set of sampled polylines) the objective of tract bundling is to produce a simpler, easier to understand layout of these same tracts. Much like related image-based bundling methods (most notably KDEEB and FFTEB), CUBu is based on the notion of Kernel Density Estimation (KDE) [23]. By estimating density of sampling points within a local neighbourhood (commonly labeled ρ, Eqn.3.1) we can establish suitable locations for alternate tract placement, which al-

lows the generation of the simplified layout of these tracts.

In our proposed method, fiber tracts are iteratively shifted towards nearby regions where ρ is already high (many paths exist) by advection in the direction of the density gradient $\nabla\rho$. This process essentially trades off clutter for overdraw, as the number of intersections of bundles should be significantly lower than the number of intersections of input paths[34].

For consistency with the modeling proposed in[75] we apply the same terminology, and for this reason *fiber tracts* are also called *edges*. Starting with a set of fiber tracts (G), consisting of edges $E \subset \mathbb{R}^3$, the algorithm's first phase creates a bundling $B \subset \mathbb{R}^3$of G.

Edges e_i are modeled as uniformly-sampled polylines $e_i = \{x_j\}$, for which $\|x_j - x_{j+1}\|$ is a user-given sampling- step σ. The control points x_j of these polylines are referred to as *sites*, and the first- and last points $\{x_0, x_n\}$ as *endpoints*.

To generate bundling $B \subset \mathbb{R}^3$ of G, several iterations are performed until a stable bundling is achieved. Each iteration incorporates four steps, these are: *density estimation*, *advection*, *smoothing* and *resampling*. Figure 3.3 shows these four steps (discussed next), as part of the iterative KDE bundling process.

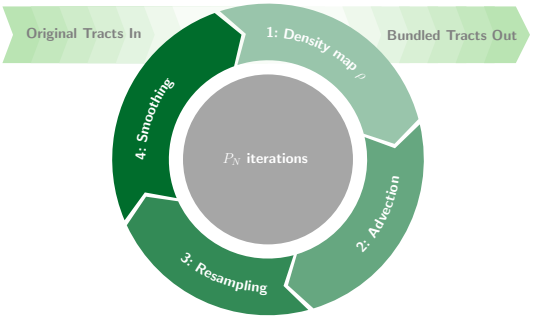


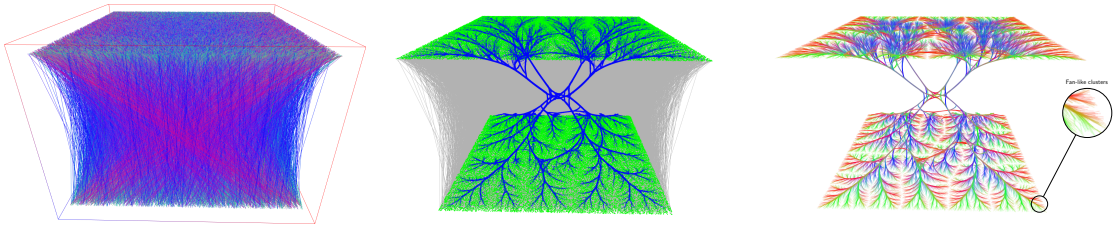
Figure 3.3: Iterative KDE bundling steps.

DENSITY MAP As a first step, a density map ($\rho : \mathbb{R}^3 \to \mathbb{R}^+$) is constructed for the drawing of G. To compute ρ, the set P of all sites x_j on all edges in E is convolved with a spherical non-linear kernel K of radius p_R. Common choices for K are an Epanechnikov kernel or a Gaussian kernel[23, 75], however we found that $K(x) = e^{-4\|x\|}$ also yields high quality results in the context of fiber tract bundling.

To estimate the density, first a (per-pixel) site count map $C : \mathbb{R}^3 \to \mathbb{R}^+$ is computed for P. Next the count map C is convoled with kernel K to produce the density map according to equation 3.1:

$$\rho(y) = \sum_{x \in T(y)} K\left(\|x - y\|\right) C(x) \tag{3.1}$$

With $T(y)$ being a sphere or radius P_R centered at y. After each iteration, the kernel radius P_R is reduced which gradually results in a stable bundling state.
Density map ρ is efficiently computed in parallel using a gathering strategy[75]. The $3D$ convolution (Eqn. 3.1) can be split into three $1D$ passes (for each row, column and slice), as the density kernel (K) is symmetric and separable. This approach leads to increased performance (discussed further in section 3.5), which is rather important



(a) Original edges with bounding box ($140 \times 140 \times 96$).

(b) Bundled edges with sites (blue) and edge endpoints (green).

(c) Bundled edges with directional colormap ($\alpha = 0.2$).

Figure 3.4: Bundling of a pseudo-random dataset containing $50K$ crossing edges.

since density estimation is one of the more computationally expensive aspects of the bundling algorithm.

ADVECTION Next, the sites x_j are advected upwards in the normalized gradient of ρ with a distance p_R, according to equation 3.2. The gradient $\nabla\rho$ is computed from ρ using a finite differences method, for the fiber tract simplification, we found that central differences is a sufficient method.

$$x_j^{new} = x_j + p_R \frac{\nabla\rho}{\|\nabla\rho\|} \tag{3.2}$$

Equation 3.2 can be further adapted to exibit control over the bundling, for instance using edge profiles[75], which is discussed in subsection 3.3.4. During advection, edge endpoints are generally constrained (i.e. not advected). However, for application to fiber tracts we propose an alternate approach for the bundling of endpoints specifically, which is furter discussed in subsection 3.3.3.

SMOOTHING In order to produce smooth bundles, Laplacian smoothing according to equation 3.3 is applied after advection. Smoothing removes small scale jitters that arise from imprecisions in the density estimation.

$$x_i^{smooth} = (1-\phi)x_i + \phi\frac{\sum_{j=i-L}^{i+L} x_j}{2L+1} \tag{3.3}$$

In equation 3.3, ϕ is the smoothing factor, and L determines the number of neighbouring sites for which the window average is computed.
Although it is an essential step, smoothing should applied with some restraint, as oversmoothing may remove an undesirable amount of curvature from the fiber tracts. For fiber tract bundling $\phi = 0.2$ is generally sufficient when smoothing once each bundling iteration.

RESAMPLING Edges are next resampled to get an uniform and dense spatial distribution of the sites x_j over G , which is needed for a good kernel density estimation. A sensible default value for the user-given sampling distance is $\sigma \sim 1$ (close to 1 voxel).

To provide a simple example, bundling of a pseudo-random dataset using the approach described above is shown in figure 3.4. The first figure (Fig. 3.4a) shows the original dataset and bounds, and the next images (figures 3.4band 3.4c) show the main topology obtained through bundling. For this bundling result, a relatively large

starting kernel radius $p_R = 51$ was applied ($\phi = 0.25$, $P_N = 40$) and edge endpoints where blocked. The third subfigure also clearly shows fan-like clusters that occur when edge enpoints are kept in place (see also subsection 3.3.3 for details).

Each of the above steps (again see figure 3.3) is performed in parallel on the GPU which yields a significant performance boost compared to the earlier CPU-bound method KDEEB[23] that follows a very similar approach.

3.3 BUNDLING SHAPE CONTROL

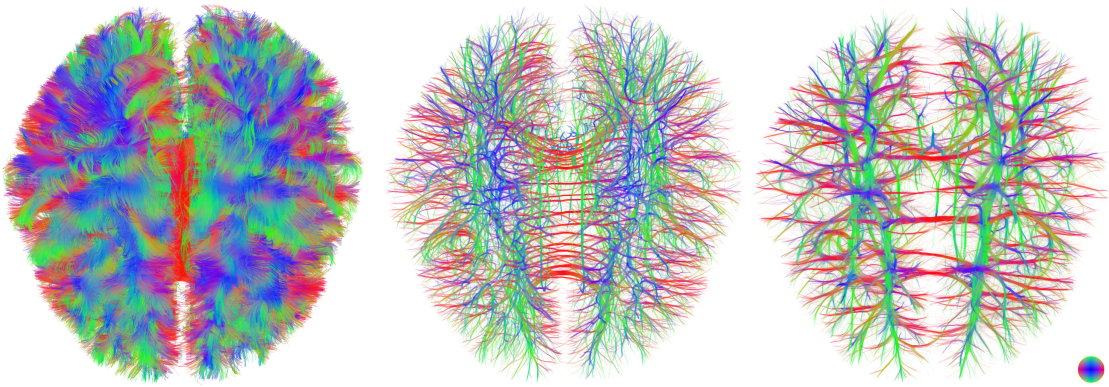
Various mechanisms for shape control of tract bundling as discussed in the previous section (section 3.2) are possible. In this section we first describe two mechanisms originally proposed by Zwan, Codreanu, and Telea [75], which are: choice of *kernel radius* (subsection 3.3.1) and *relaxation* (subsection 3.3.2). Next we also propose two additional mechanisms for shape control, being: *endpoints bundling* and *DTI edge profiles.*

The bundling of endpoints can yield further reduction of occlusion, which is beneficial in the context of fiber tract bundling. DTI edge profiles are an adaption of the edge profiles described in [75], in which DTI tensor shape classification measures (section 2.2) are incorporated to preserve certain structures during the simplification process.

3.3.1 *Kernel Radius Controls Bundling Coarseness*

The initial kernel size p_R (equations 3.2 and 3.1), specified in voxels, controls bundling coarseness. Increasing p_R includes sites from within a larger spherical region in the density estimation, which results in a courser bundling result as sites from a wider area are advected towards higher-density locations.

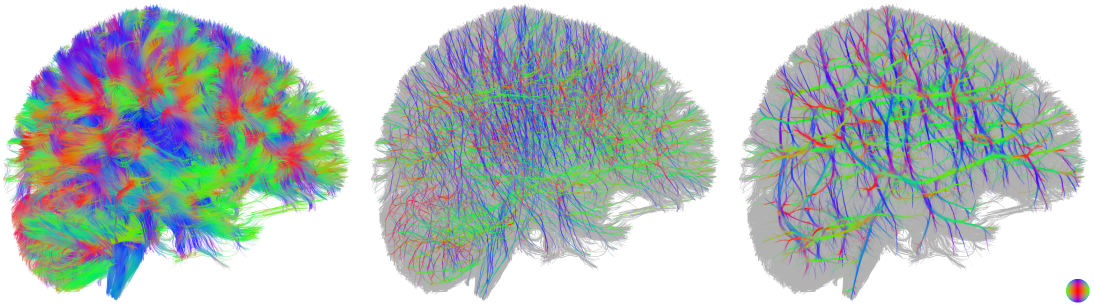
Figure 3.5 shows a side-by-side comparison of fiber tract bundling using two different kernel sizes. The left column (Fig. 3.5a and 3.5d) shows the original fibers (viewed along the sagittal axis and the transverse axis), the middle column (Fig. 3.5b and 3.5e) shows a finer bundling result produced by a relatively small starting kernel size $p_R = 9$, and the right column (Fig. 3.5c and 3.5f) shows coarser bundling produced by selecting a larger starting kernel size ($p_R = 51$).



(a) Original fiber tracts.

(b) Finer bundling using smaller kernel ($p_R = 9$).

(c) Courser bundling using larger kernel ($p_R = 51$).



(d) Original fiber tracts.

(e) Finer bundling using smaller kernel ($p_R = 9$).

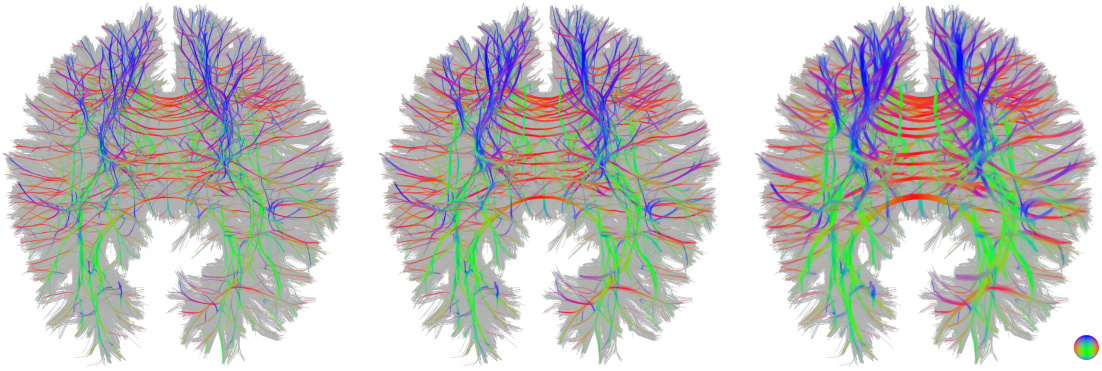
(f) Courser bundling using larger kernel ($p_R = 51$).

Figure 3.5: Course-to-fine bundling produced by varying density kernel radius p_R.

3.3.2 *Relaxation Controls Bundle Thickness*

In addition to the adjustment of the kernel radius P_R (as described in the previous subsection), *relaxation*[24] is another approach to bundle shape control which allows for the control of bundle *thickness*. Bundling relaxation is the process of interpolating between the bundled fiber tracts and their original counterparts, thereby *relaxing* the bundled output towards the original input. The application of a moderate amount of relaxation (generally 5..20%) results is slightly thicker (more pronounced) bundles that can be easier to observe.

As an example, figure 3.6 shows a bundled set of fiber tracts with three levels of relaxation applied, rendered in context of the original fiber tracts (in gray). The first subfigure (Fig. 3.6a) displays very fine bundles without any relaxation applied, which is the bundling result as created using te algorithmn described in section 3.2. Comparing this to the next two figures (Fig. 3.6b and 3.6c) shows that bundles become more pronounced as relaxation is applied. The amount of relaxation is a user-controlled parameter that can be adjusted interactively to determine a suitable level.



(a) No relaxation (fine bundles). (b) 10% relaxation (slightly thicker bundles). (c) 20% relaxation (thicker bundles).

Figure 3.6: Relaxation applied to bundling result (100K fiber tracts).

3.3.3 *Endpoints Bundling*

Keeping edge endpoints in place is an important aspect of the CUBu bundling method. In general bundling applications, endpoints may represent important information such as a specific location on a map, or a node of a graph, in which case it may be undesirable to alter the representation of such a node. Another important motivation for keeping endpoints in place is that this approach prevents the underlying structure of the data from shrinking during bundling. There is a side effect to keeping all endpoints in place: upon completion of the bundling process, various groups of endpoints are typically visible as fan-like clusters (see for example Fig. 3.4c and Fig. 3.8b). This is a common phenomenon for various bundling algorithms that employ a similar strategy for edge endpoints, also referred to as the FDEB style [34, 75].

Due to the fact that they are intentionally unaffected, the reduction of occlusion is not optimal for these fan-like clusters of endpoints. For 2D trail- or graph datasets this generally poses less of a problem, as the point of view usually remains unchanged. For interactive exploration of our 3D fiber tracts however, this becomes more problematic as the user changes the viewpoint. Any occlusion that is not reduced due to keeping all endpoints in place can have a negative impact on the visualization of the bundling result. For example clusters of endpoints closer to the skull may prevent us from looking inward into the simplified set of fibers.

Various rendering approaches that de-emphasize low-density areas (such as the groups of endpoints described above) are available, these are discussed in more detail in chapter 4. In essence, these rendering methods implicitly tune out low-density areas by means of i.e. alpha-blending[22]. Another option for decreasing the occlusion further, is to prune the endpoints; thereby effectively trimming the fan-like clusters described above. This trimming approach does yield a cleaner visualization, but this is at the cost of shortening bundles and possibly omitting important information. Therefore both of these options have drawbacks in terms of conveying the best possible simplified model of the neural fiber structures, i.e. we either impose additional

requirements on our rendering approach (such as the need for alpha-blending) or we present an image that may not adequately reflect the underlying structure of the original fibers in the best possible way due to shortening.

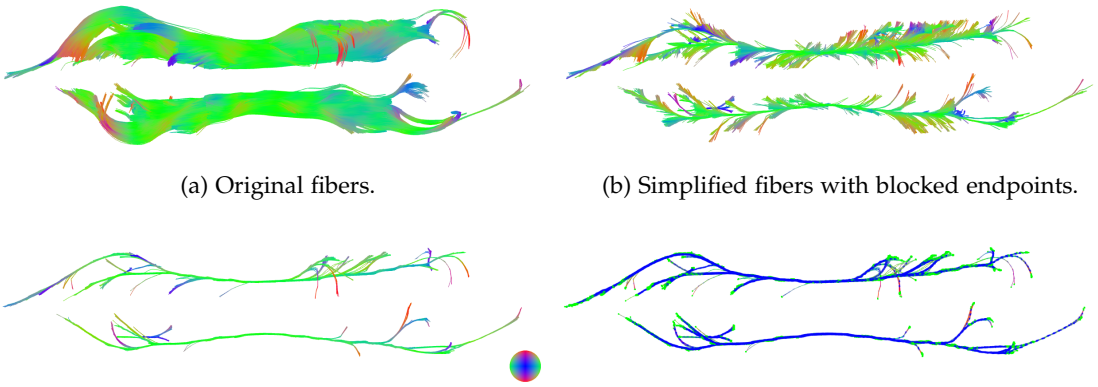
An effective solution to the problem described above is to include all endpoints in the advection process, similar to all other sample points, while also controlling the deformation of the line segments in such a way that no fibrous structures are unnecessarily shortened. Similar to the approach that limits advection in the direction of the local tangent (tangent drift) to improve sampling quality of edges while bundling [75], we can also use the tangent direction $\tau(x_j) \in \mathbb{R}^3$ (eq. 3.4) to guide the advection of all fiber endpoints.

$$\tau(x_j) = \frac{x_{j+1} - x_j}{\|x_{j+1} - x_j\|} \tag{3.4}$$

Based on the observation that displacement of endpoint x_j in the direction of $\tau(x_j)$ is guaranteed to shrink a fiber, we can assume that using a direction (d') that is perpendicular to the tangent will likely *not* do so. In the case of $2D$ bundling we have $\tau(x_j) \in \mathbb{R}^2$ for which $d' \perp \tau(x_j)$ can be determined directly. In the case of fiber-bundling we have $\tau(x_j) \in \mathbb{R}^3$ and therefore there are infinite possible d' and we need to establish a suitable plane containing $d' \perp \tau(x_j)$ first. To achieve this we can use the displacement (d) according to for example equation 3.2(i.e. $d = x_j - x_j^{new}$) and project this vector d onto the alternate displacement vector d' that is in the same plane as tangent $\tau(x_j)$ as well as displacement d, but also orthogonal to $\tau(x_j)$.

Equation 3.5 shows how we compute the projection of d onto d', where we refrain from advection when the displacement is nearly parallel to the tangent ($d \parallel \tau(x_j)$), do not project the vector when it is already orthogonal ($d \perp \tau(x_j)$) and project d otherwise. A schematic example of such a projection is given in figure 3.7 where the $w = \frac{d \times x_j}{\|d \times x_j\|}$ (the normal of the plane containing the orthogonal bundling direction) and $u = \frac{w \times \tau(x_j)}{\|w \times \tau(x_j)\|}$.

$$d'(x_j, d) = \begin{cases} 0 & if \ \frac{d}{\|d\|} \cdot \tau(x_j) \approx 1 \\ d & if \ \frac{d}{\|d\|} \cdot \tau(x_j) \approx 0 \\ (d \cdot u) \cdot u & with \, u = \frac{d \times x_j}{\|d \times x_j\|} \times \tau(x_j) \, otherwise \end{cases} \tag{3.5}$$



(a) Original fibers.

(b) Simplified fibers with blocked endpoints.

(c) Bundling applied to all sampling points.

(d) Endpoints (green) after bundling.

Figure 3.8: Endpoints bundling demonstrated on bundles of the left- and right cingulum.

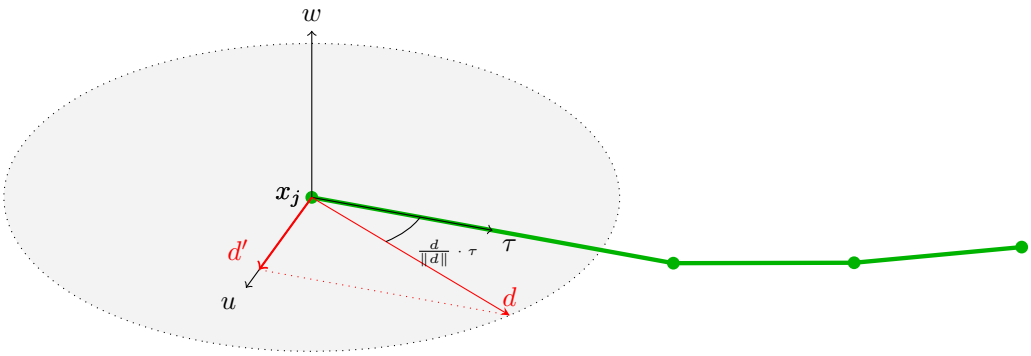
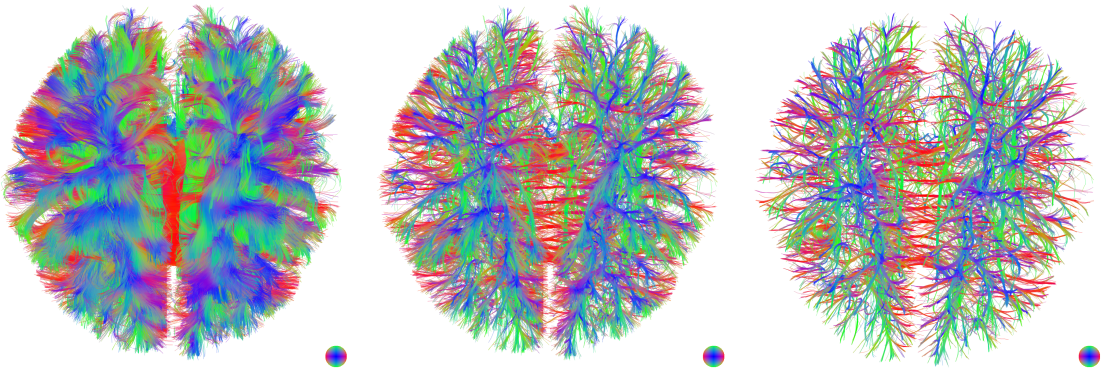


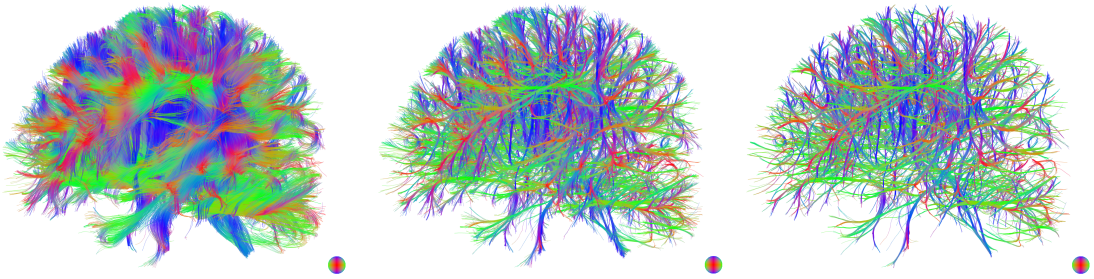
Figure 3.7: Schematic view of projection of displacement d of endpoint x_j of the green polyline onto d'. Vectors u, τ, d and d' are all in the plane defined by normal w (indicated by the gray ellipsoid), with $w \perp u \perp \tau$.

The effect of endpoint bundling using equation 3.5 is demonstrated on a local scale in figure 3.8 where we compare original fibers to simplified fibers with and without endpoints bundling on fibers from the left- and right cingulum. Figure 3.8a shows these bundles using simple line rendering with directional color coding (eq. A.1) viewed along the vertical axis. Figure 3.8b shows the exact same set of fiber-tracts with simplification applied according to equation 3.2, while in figure 3.8c alternate advection was applied according to eq. 3.5. In this last image (fig. 3.8c) it clearly visible that the fan-like structures appearing in fig. 3.8b are more folded into the main bundle, which leads to a cleaner simplification result.

Comparing figures 3.8b and 3.8c shows that occclusion can be reduced much further when applying constrained advection to *all* sampling points (as opposed to only sites), while ensuring that the resulting bundles are not shrunk unnecessarily. An important side note is that although we apply alternate *advection* to all endpoints they are still exempt from any Laplacian smoothing (as is the case for any bundling style described in[75]) since this would also shrink the fibers by definition (eq. 3.3).



(a) Original fibers. (b) Endpoints blocked. (c) Endpoints bundled.



(d) Original fibers. (e) Endpoints blocked. (f) Endpoints bundled.

Figure 3.9: Side-by-side comparison of endpoints bundling on a set of $50K$ fibers covering the entire brain.

From a global perspective the difference is also very apparent as is shown in figures 3.9a through 3.9f. Again these images show a side-by-side comparison between the original fibers, bundling with blocked endpoints and bundling with guided advection of endpoints. The top row shows a view along the vertical axis (alpha blended lines with directional colormap eq. A.1 and $\alpha = 0.6$) and the bottom row shows the same set of fibers viewed along the transverse axis with the exact same rendering parameters. The differences are most noticeable in regions where white matter transitions into other tissue, e.g. regions closer to the skull, since endpoints are typically densely distributed in these areas due to the fact that a stopping-criterion is met during fiber-tracking.

3.3.4 *DTI Edge Profiles for Constrained Advection*

To further control the bundling specifically in the context of DTI, we modulate site advection (Eqn. 3.2) by an edge-profile function[75]. We apply such a profile function in order to restrain bundling based on scalar DTI measures, in particular *FA* (Eqn. 2.6),c_p (Eqn. 2.3) and c_l (Eqn. 2.2).

Our profile is defined as:

$$\lambda_{FA}(x_j) = \begin{cases} 1 & FA(x_j) \geq t_{FA} \\ 0 & otherwise \end{cases} \tag{3.6}$$

$$\lambda_{c_l}(x_j) = \begin{cases} 1 & c_l(x_j) \geq t_{c_l} \\ 0 & otherwise \end{cases} \tag{3.7}$$

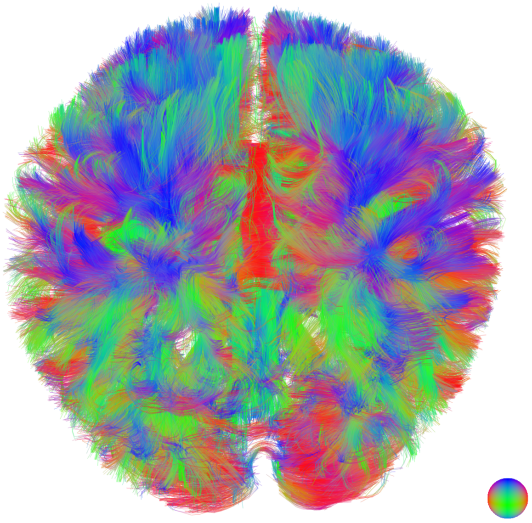
$$\lambda_{c_p}(x_j) = \begin{cases} 1 & c_p(x_j) \geq t_{c_p} \\ 0 & otherwise \end{cases} \tag{3.8}$$

Where t_{FA}, t_{c_l} and $t_{c_p}$are user-defined thresholds for each scalar measure (FA,c_l and c_p).

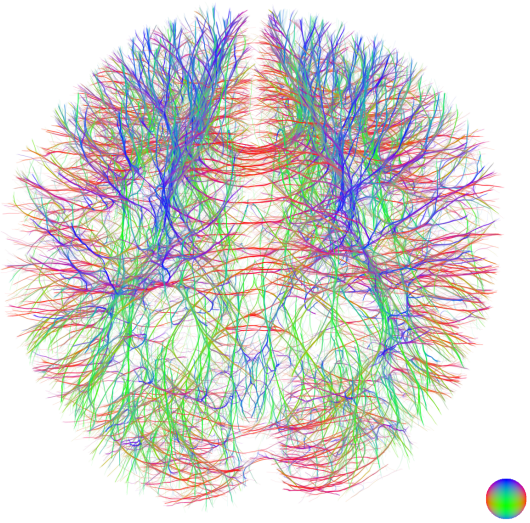
As besides advection (Eqn. 3.2), each site x_j is subject to smoothing and resampling as well (which all may alter a site in some way). To completely restrain the bundling, we also limit smoothing based on the above edge profiles.

Figure 3.10 shows bundling constrained by FA using equation 3.6. By comparing e.g. subfigures 3.10b and 3.10d, we can observe that several key structures within the fibertracts are preserved, most notably the *corpus callosum.*

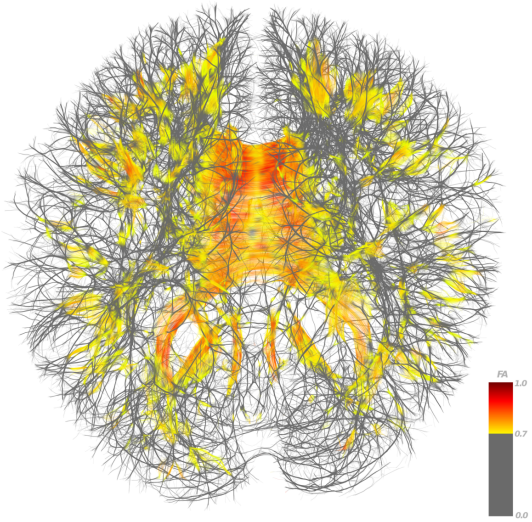
A similiar result is shown in figure 3.11 where a c_l-constrained edge profile was applied ($c_l \geq 0.4$) and a larger kernel radius ($P_R = 23$) was used. Constrained structures are shown using a rainbow colormap in context of the bundled fibers (dark gray) and original fibers (light gray). This image highlights the preserved *corpus callosum* (amongst other structures) as well.



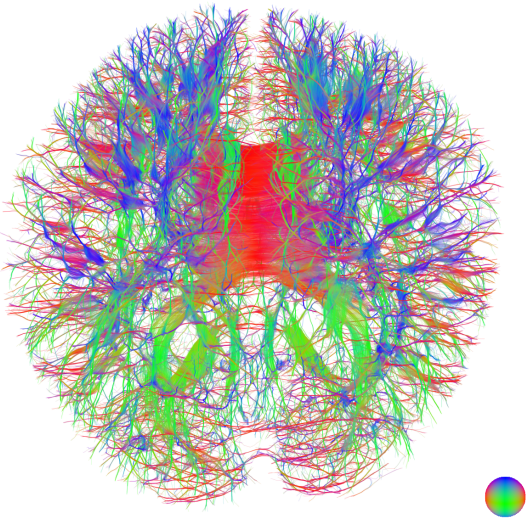
(a) Original (100*K* fibers).



(b) Bundled (uniform edge profile).

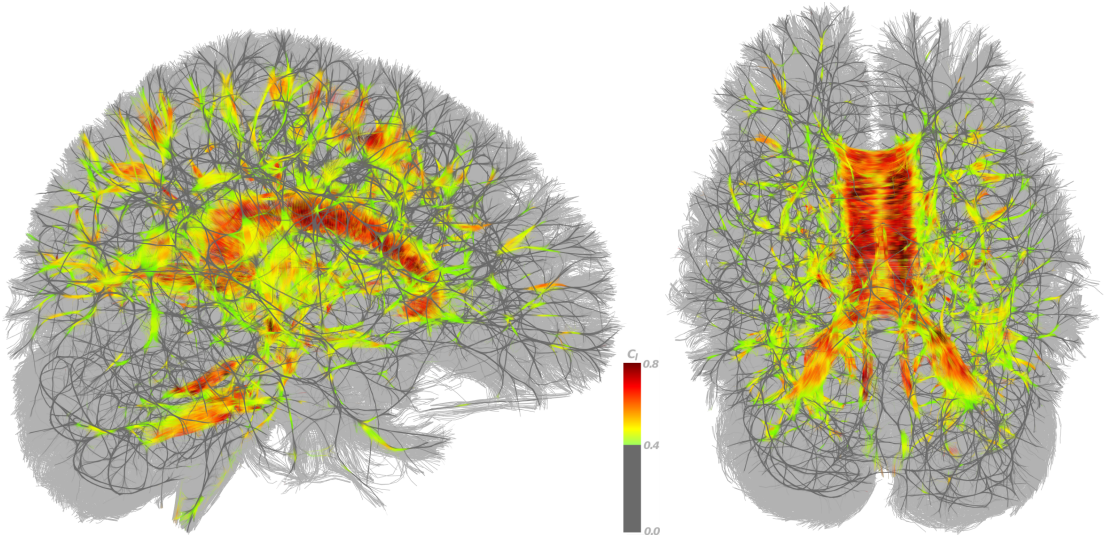


(c) Constrained bundling where $FA \geq 0.7$ (thresholded colormap shows affected fibers).



(d) Same as figure 3.10c with directional colormap.

Figure 3.10: *FA*-constrained edge profile (100*K* tracts).



(a) Transverse axis view. (b) Viewed along the vertical axis.

Figure 3.11: Bundling using c_l-constrained edge profile rendered in context of original fibers ($c_l \geq 0.4$, $100K$ tracts).

3.4 RESEEDING

To complement the bundling as described in the previous section, our simplification approach features additional fiber tracking in specific regions of higher planar anisotropy, which we call *reseeding*. The purpose of reseeding is to reveal or uncover sheets of fiber tracts that are characteristic for areas of higher c_p. Since the input dataset may not contain all these (required) fiber structures, we ensure this aspect by performing additional tracking of fibers in these regions.

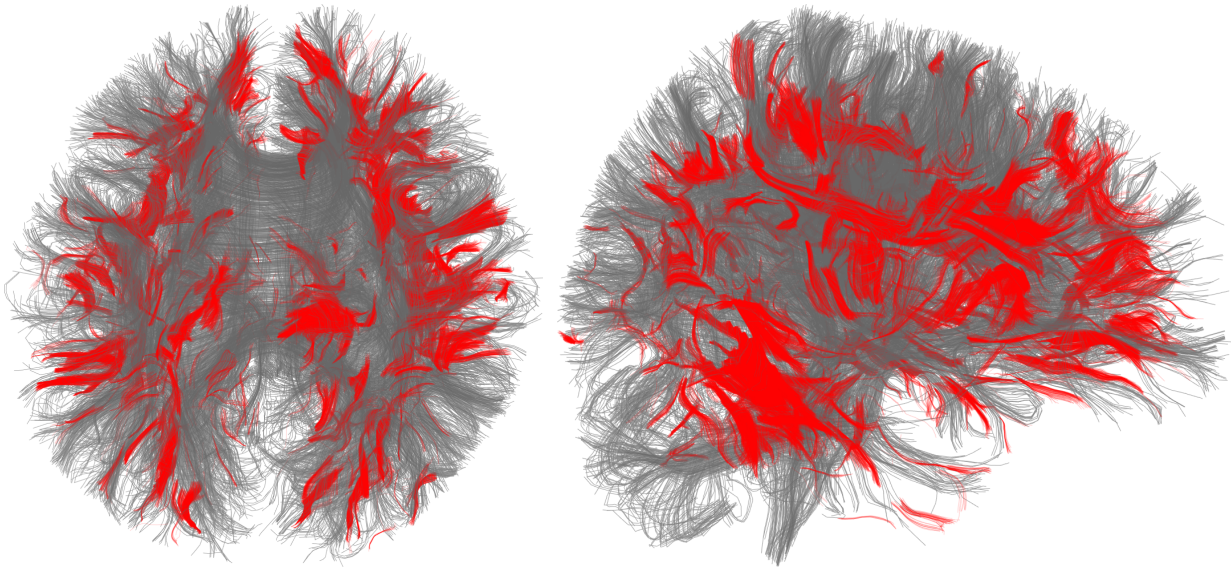
The fiber tract reseeding, like tract bundling, is part of the simplification process as shown in figure 3.1.

Similiar to the approach proposed by Vilanova, Berenschot, and Pul [67] we apply a (re)seeding and stopping criterion based on: $c_p \geq 0.25 \wedge c_l \leq 0.2$. This criterion indicates regions of higher planar certainty (c_p, Eqn. 2.3) yet not too high linear certainty (c_l, Eqn. 2.2).
Fiber tracking is performed using a deterministic method (see section 2.3) that applies Euler integration, trilinear interpolation of the eigenvectors and scalar certainty metrics, as well as random seepoint placement within each voxel.

An example of such reseeding is shown in figure 3.12 where a relatively small set of fiber tracts ($\sim 15K$, indicated in gray) is complemented with approximately $65K$ additional fibers (indicated in red).

dataset	tract count	average #points	average iteration time	total running time
random data	50.000	3.271.979	30*ms*	444*ms*
dataset #3	50.000	1.441.486	11*ms*	170*ms*
dataset #2	50.000	4.145.785	8*ms*	314*ms*
random data	500.000	32.762.628	254*ms*	3812*ms*
dataset #2	500.000	12.639.524	57*ms*	858*ms*
dataset #3	500.000	14.850.241	69*ms*	1029*ms*

Table 3.2: Measured bundling running times on a system with a single GTX-1080 GPU (2560 CUDA cores).



(a) Anterior view.

(b) View along the transverse axis.

Figure 3.12: Original fiber tracts (grey) reseeded with additional fiber tracts (red) in planar regions ($c_p \geq 0.25$).

3.5 SCALABILITY

Scalability is important for interactive DTI visualization and has two sub-aspects. First, data scalability refers to ways in which we can handle large amounts of data points (e.g. voxels in the DTI volume, but also sample-points in the set of fiber tracts). Secondly, computational scalability refers to ways in which we can process such datasets quickly in order to generate the final simplified models.

To provide an indication of the computational scalability of the bundling (section 3.2) we show a listing of running times for several datasets in table 3.2, as obtained from a single GTX-1080 GPU system. Measurements include roundtrip CPU to GPU transfer of fiber tracts (yet do not include timings of aspects related to rendering).

As described in section 3.2 we use separable convolution to compute ρ (Eqn. 3.1). Doing so requires a buffer for reading and writing during three consecutive passes

dataset / scale factor	×1 *	×2	×4	×8
dataset1	88 × 111 × 82 (3.2 MB)	176 × 222 × 164 (25.63 MB)	352 × 444 × 328 (205.05 MB)	704 × 888 × 656 (1.64 GB)
dataset2	61 × 82 × 42 (840.34 KB)	122 × 164 × 84 (6.72 MB)	244 × 328 × 168 (53.78 MB)	488 × 656 × 336 (430.25 MB)
dataset3	84 × 98 × 83 (2.73 MB)	168 × 196 × 166 (21.86 MB)	336 × 392 × 332(174.91 MB)	672 × 784 × 664 (1.4 GB)
dataset4	86 × 116 × 88 (3.51 MB)	172 × 232 × 176 (28.09 MB)	344 × 464 × 352 (224.74 MB)	688 × 928 × 704 (1.8 GB)

Table 3.3: Memory usage for storing a 32-bit precision density map (per dataset).

(one pass for each dimension). This approach is performant, yet requires 2× the amount of memory during density estimation. Alternatively we can gather $\nabla\rho$ from a sphere of size P_R from within site count map $C(x)$. This only requires a single buffer, yet is also much slower as it is an $O(P_R^3)$ operation, while separable convolution is $O(P_R)$. By applying separable convolution, we therefore trade off memory for computation, which is favorable if sufficient memory is available.

** original resolution of brain mask M.*

As oppossed to normalizing the input tracts to fit the volume resolution P_I (as proposed in[75]), we set P_I to the axis-aligned bounding box indicated by brain mask M (since all tracts are contained within), and scale this resolution as needed. This approach allows for easier mapping to the DTI data during simplification.
Example memory requirements for a single density buffer, based on this approach, are listed in table 3.3. Besides the density buffers, fiber tract information and DTI data are also required to be stored in device memory. For example, handling $1M$ tracts with on average 100 sample points requires approximately 1.2 GB of memory. Adding this to the figures shown in table, we can see that a GPU that has 4 GB of memory is sufficient to handle a dataset of this magnitude when using our proposed simplification method.

As pointed out by Zwan, Codreanu, and Telea [75], the output volume size P_I controls the accuracy of density estimation (Eqn. 3.1), and thus also of the gradient ($\nabla\rho$) estimation by finite differences. In order to obtain the best possible result using the smallest possible volume resolution, we sample ρ using trilinear interpolation (with 32-bit precision).

4 FIBER TRACT RENDERING

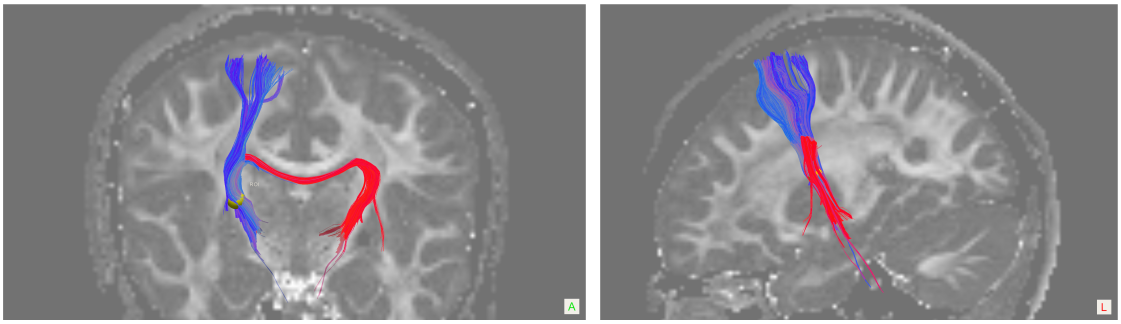
Following the simplification approach (as discussed in the previous chapter 3) we have a (simplified) set of fiber tracts that requires adequate rendering. We propose a number of rendering techniques that aim to produce effective views of the *actual* fiber structures within the brain. By effective, we mean views which add more information cues to the original tracts (3*D* polylines) in such a way that the final images more suitable for providing insight. Such insight relates to the reduction of clutter; understanding of local anisotropy (e.g. planar vs. linear); filling in small gaps that exist even in densely sampled sets of fiber tracts, and simplifying the view by highlighting the coarse-scale structure of the underlying geometries.

4.1 CHALLENGES RELATED TO FIBER TRACT RENDERING

Fiber tracts are commonly visualized using illustrative rendering- or Non-Photorealistic Rendering (NPR) techniques (see also section 2.4). Fiber tracts differ from streamlines originating from e.g. aerodynamic trail data or fluid flow data in the sense that they have a clearer mapping to physically existing biological tissue - WM bundles[17]. DTI fiber tracts generally represent complex, organic shapes and have many more structural variations than streamlines originating from other forms of vector field visualization.

One detailed example of such a complex structure is given in in figure 4.1, where bundles from the *corona radiata* and the *corpus callosum* passing through a spherical ROI (indicated by a yellow sphere) are displayed. This image clearly shows branching tubular structures within the bundles of fibers as well as surface-like structures. The *corona radiata* bundle (in blue) contains a flat, surface-like part with several branches, and it also merges into the U-shaped bundle from the *corpus callosum* area (red).

To provide some context on the location of the bundles each image contains a greyscale slice of fractional anisotropy (Eqn. 2.6), which was also the scalar attribute that was used in the seeding- and stopping criterion. Such complex fibrous structures are characteristic for the human brain and can be found throughout the result of a whole-brain tractography (previously given examples are figures 3.9a and 3.9d).



(a) Bundles with coronal *FA* slice.

(b) Bundles with sagittal *FA* slice.

Figure 4.1: Bundles from the corona radiata and the corpus callosum depicted with slices of *FA* (Eqn. 2.6) from two viewpoints.

Traditional rendering methods for the aforementioned DTI structures are line rendering and shaded 3*D* tubes [36, 64]. Though both have proven to be effective rendering methods and are therefore commonly used [18, 36, 69], they do not fulfill all of our rendering requirements (as listed in subsection 1.1.3). This is mostly due to the fact that we are trying to convey the local nature of fiber bundles, and thus intend to also create a surface-like appearance from fiber tracts wherever applicable (see requirement R1). Even if there is a dense surface-like *distribution* of fiber tracts, creating the appearance of a surface from this (without gaps) can be challenging when applying techniques such as line rendering. In essense this originates from the modeling limitations of a set of 3*D* curves, which raises the question how to bypass this by rendering more than just the fiber tracts in order to obtain a more preferable approximation of the underlying structure.

Basic line rendering offers limited options in terms of shading and scaling, which can be improved by rendering 3*D* tubes instead. In the case of tube rendering, fibers are modelled with 3*D* geometry, which naturally offers more options for shading and scaling but at the cost of much more complex modelling. Screenspace rendering techniques applied in the context of DTI, such as [39, 60], address this by recreating 3*D* tube shading with simpler modelling. Though this approach may solve certain performance issues and provides interesting options for shading as well, this still raises the question how to merge or blend geometry (visually or otherwise) in order to create solid looking structures with smooth shading. The DDH method [15, 17, 61] is an example of a method that performs this *merging* of structures quite well, as colinear fibers are visually combined into thicker bundles with illustrative halos. The major downside of the DDH method is that the black and white rendering (a stylistic choice motivated by NPR) offers limited scalar visualization capabilities.

In this chapter we describe an alternative DTI rendering technique that is able to depict a wide variety of fiber structures and that provides various options for shading and color mapping, while also rendering at interactive framerates.

4.2 INTRODUCTION TO ALPHA BLENDED POINTSPRITES

Provided that we have sufficiently dense sampling, it is possible to create solid-looking geometry by rendering (circular) $2D$ sprites at all sampling points of the fiber tracts. For $3D$ curves this can be achieved by rendering the curves using *billboarded pointsprites* [6], which are $3D$ elements rendered as $2D$ sprites.

Billboarding refers to a rendering approach where rendered elements have a $3D$ position, but are not modeled with a particular orientation, and instead are always oriented towards the viewer. Elements may also be scaled to create the suggestion of perspective projection. In the case of billboarded pointsprites, all sprites are modelled as equally oriented squares that are always tangent to the viewing plane (facing the viewer). The geometric representation of a billboarded pointsprite is a point in $\mathbb{R}^3$ (a worldspace position) with a certain attributed pointsize that yields the screenspace size of the sprite upon rasterization. The purpose of billboarding in this case is to present solid structures to the viewer (i.e. close to orthogonal to the viewing plane).

By rendering pointsprites along densely sampled fiber tracts, we can create both shaded tubes as well as surface-like objects (requirement R1), yet conveniently we do not need to explicitly distinguish between these types of structures. It is also possible to smoothly transition between those two categories of shapes, for instance a tube-like structure transitioning into a surface-like structure (like the example previously given in figure 4.1), all while using the same geometric primitive (a polyline, rendered using only sprites).

Figure 4.2 shows this concept applied to the exact same set of fibers that was used in the introduction of this chapter (Fig. 4.1). The left image shows simple line rendering (1 pixel linewidth), and the right image shows the same set rendered using pointsprites. All fiber tracts in the second image appear slightly thicker, because additional scaling of the rendered elements was applied to create solid appearing bundles (this is discussed further in section 4.5).
In addition to clearly showing the local nature and orientation of fiber tracts, each individual sprite can easily be colored, which allows for color coding as required by e.g. scalar visualization.

Our pointsprites rendering approach is closely related to surface-splatting techniques [26, 29] in the sense that splats (or sprites) are used to create the appearance of a surface from an underlying datastructure that does not represent such a surface explicitly, yet also different due to the fact that besides surfaces we also aim to render tube- or line-like bundles. Since each sprite can be regarded as a simple form of glyph, a relation to DTI glyph rendering techniques [52, 57, 64] is also apparent. Similar to glyph rendering, each sprite can encode DTI attributes through e.g. scalar color coding or scaling, yet unlike glyph rendering individual elements are indistinguishable from another, since these are densely placed (and blended) in order to create solid structures. For these reasons this method could be considered a hybrid of glyph-based techniques, surface splatting and conventional streamline visualization.

(a) Lines (1px). (b) Alpha blended pointsprites.

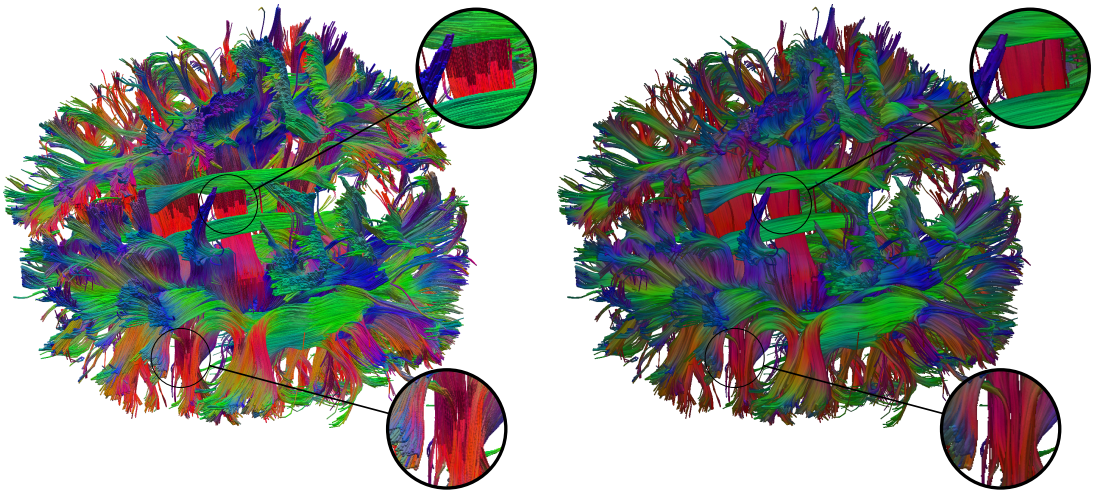
Figure 4.2: Bundles from the *corona radiata* and the *corpus callosum*.

There are various steps involved in the rendering as described above. In order to achieve smooth shading without artefacts we render all sprites using alpha blending. This blending and associated render order details are discussed in section 4.3 in which we propose a multi-pass rendering method for smooth shape reconstruction. Each of the sprites can be textured in various ways to produce distinct shading that enhances shape perception, we describe several shading profiles in section 4.4. To produce solid looking fibers and surface-like structures without gaps, sufficient overlap between rendered sprites is required. While billboarding in general is a proficient mechanism to achieve such overlap as rendered elements always face the viewer, we can also achieve such overlap by either introducing additional points in areas with insufficient density (through e.g. on the fly resampling, see section 4.6) or by increasing the size of the rendered sprites which is discussed in section 4.5. We optionally complement our pointsprites rendering with illustrative distance attenuated outlines in order to convey a simplified visualization, which we cover in detail in section 4.8. Finally, an overview of implementation details for all of the above is provided in section 4.7.

4.3 MULTI-PASS BLENDING FOR INTERACTIVE SMOOTH SHAPE RECONSTRUCTION

Rendering all sprites in one pass using conventional depth buffering can produce visually appealing results (shown in fig. 4.3a). This single-pass approach may however suffer from certain rendering artefacts. This is particularly the case when the camera orientation is changed or wherever sprites with different color encoding coincide. We can improve the visual quality of the rendering further by *blending* the sprites,

which results in smoother shading of all rendered structures. Figure 4.3 shows the difference between these two variants on the same dataset. Figure 4.3a is rendered without blending, while figure 4.3b is the same image with the improved shading that is based on alpha blending.



(a) No alpha blending applied.

(b) Improved shading with blending.

Figure 4.3: Point sprites with- and without blending. Demonstrated on a slice-filtered set of fibertracts using a directional colormap (eq. A.1).

Blending the sprites introduces a complication: because the blending operation is not commutative, the order in which fragments are rendered becomes important [50]. One possible solution is to render all elements from back to front, that is: starting with the elements furthest away from the camera and ending with those that are closest. This can be done by transforming all vertices to eye-space using the model-view matrix associated with the current camera setup, and then by sorting them by vector length (since the *"eye"* in eye-space is located at the origin).

Although explicit sorting will likely ensure a correct render order, it is also potentially a very expensive procedure when the camera setup is subject to frequent change. Ideally we would like to render all sprites with blending in an arbitrary order, yet with a final result that is equal to- or at least very similar to sorted rendering.

The class of techniques that deals with this problem is generally referred to as Order-Independent Transparency (OIT)[5, 14]. There are various proposed methods that all share a common objective: to allow the rendering of geometry with alpha blending yet without an explicit render order.

OIT techniques such as Weighted Blended Order-Independent Transparency [37] are generally designed for scenes containing highly transparent objects, such as overlapping layers of glass or smoke particles. For our usecase we do not nescessarily need complete transparency of all rendered objects, since our goal is to blend these pointsprites and create smoothly shaded geometry while still maintaining some level of occlusion culling as a mechanism for showing relations between fibertracts (requirement R2). In other words: we do not need to blend all overlapping fragments, yet we would like to blend as many topmost fragments as needed in order to create

smooth shading.

To solve the render order problem described above we propose a multi-pass rendering method that applies a custom depth test procedure, in order to blend fragments that have a similiar depth value. The custom depth test essentially allows for *peaking into* or *probing* the set of all available fragments, selectively blending only the topmost fragments within a certain depth range. This rendering approach allows for the use of alpha blending for the creation of smooth surfaces as well as tube-like shading without explicitly sorting the geometry.

OVERVIEW OF MULTI-PASS RENDERING PIPELINE A higher level overview of the proposed multi-pass rendering pipeline for the alpha blended pointsprites is displayed in figure 4.4. This diagram shows how the final image is produced from an input set of fiber tracts in three consecutive rendering passes, being: a depth sampling pass, a shading pass and a compositing pass.

In the first pass we construct a depth buffer and create a flat-shaded background for the blending operation. In the second pass we apply shading and blending based on the custom depth test, using the previously constructed buffers. Next, in the third pass, we composite the final image and color the optional depth attenuated outlines (see section 4.8). During this last pass, the final image is rendered to the screen by means of a viewport sized rectangle. Two render targets (textures) are used to share information between the rendering passes, these are an RGBA texture ($RT_{rgb\alpha}$) and a depth texture (RT_d). We discuss further implementation details of this multi-pass rendering approach in section 4.7.

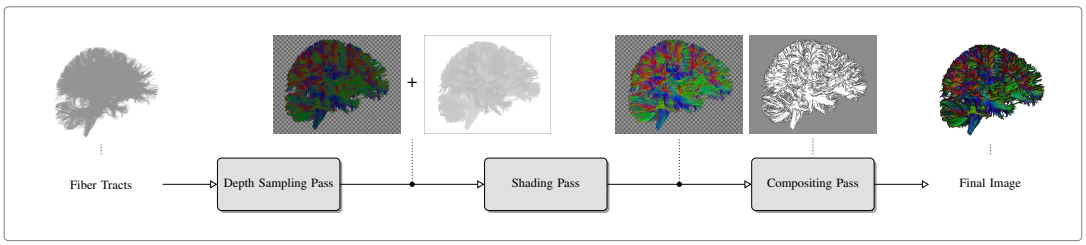


Figure 4.4: Diagram of alpha blended pointsprites rendering passes.

4.4 SHADING FOR ENHANCED DEPTH AND SHAPE PERCEPTION

To produce assorted shading effects, such as visible in the previous example figure 4.3b, each individual sprite is textured using a $2D$ splat profile $f(x) : \mathbb{R}^2 \rightarrow \mathbb{R}^+$, that determines color luminance. As the pointsprites rendering is a screenspace technique, and no particular lighting model is applied, the splat profile choice and parameterization of this profile are important factors for the final shading result. For instance, we can choose a profile that emphasizes fiber orientation with tube-like shading, or we can choose a profile with lower contrast for less pronounced (diffuse-like) shading of for instance surfaces.

A splat profile can be applied to the sprites in a sampled form (e.g. as a single channel texture), or in an analytical form (e.g. through evaluation in the fragmenta-

tion stage). While the first option may be beneficial in terms of performance, the latter is more convenient for further parameterization of the shading, making it easier to adjust the shading per sprite as needed. For example an asymmetrical profile could be rotated to align with a certain axis, or we could transform $f(\vec{x})$ to obtain a particular level of brightness or contrast. In this section we discuss several splat profiles and describe how these can be parameterized. Each splat profile is transformed to a desired shading range $[s_{min}, s_{max}] \in [0,1]$ accoriding to:

$$f(\vec{x}) = (s_{min} - s_{max})\, p(\vec{x}) + s_{min} \tag{4.1}$$

where

$\vec{x} \in \mathbb{R}^2$	Texture coordinate within the unit circle
$s_{min} \in [0,1]$	Shading range lower bound
$s_{max} \in [0,1]$	Shading range upper bound, with $s_{min} \leq s_{max}$
$p(\vec{x}) \in [0,1]$	Splat profile function

To match the texture coordinates generally used in graphics frameworks (e.g. OpenGL), the domain of Eqn. 4.1 is within $[-1,1]$. The range of Eqn. 4.1 is within $[0,1]$ which is convenient for color modulation used in for instance scalar color coding: to produce a colored sprite $f(\vec{x})$ is simply multiplied with an RGB color. The profile function $p(\overrightarrow{x})$ can be substituted by a suitable function that produces the desired shading. In the next subsections four of such profiles are discussed, being: the *flat-*, *conical-*, *smooth-* and the *Gaussian* profile.

4.4.1 *Flat Splat Profile*

The simplest profile is created by defining $p(\vec{x}) = 1$, which results in uniform color luminance. The flat splat profile is defined as:

$$p_{flat}(\vec{x}) = 1, \|\vec{x}\| \leq 1 \tag{4.2}$$

, where $\vec{x}$ is constrained within the unit cirle in order to produce a disc shaped sprite (fragments not within the unit circle are discarded).

4.4.2 *Conical Splat Profile*

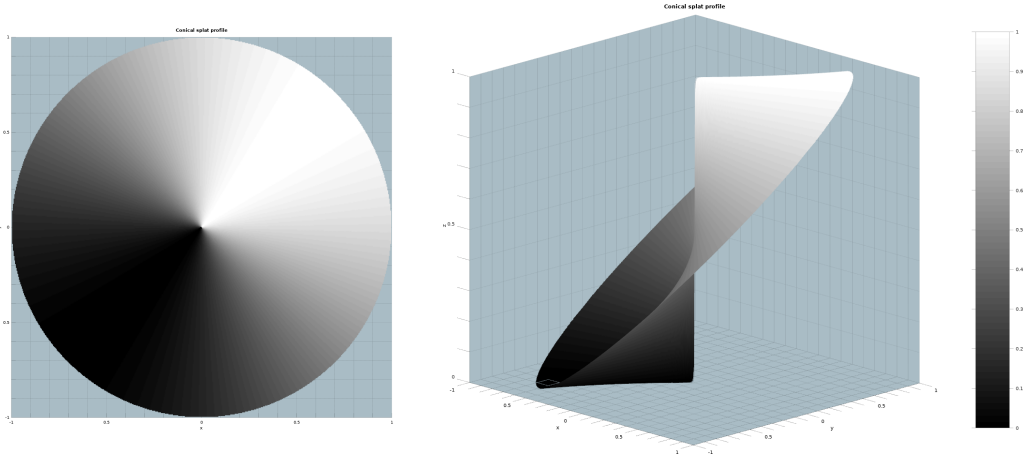
A very effective splat profile is created by using the angular distance between each texture coordinate and a predefined shading axis $\vec{a}$, relative to the center at $(0,0)$. This can be computed by taking the vector dot product of a normalized texture coordinate $\frac{\vec{x}}{\|\vec{x}\|}$ with $\vec{a}$. We name this profile the "*conical*" splat profile as it resembles a shaded $3D$ cone observed along the z-axis (shown in figure 4.5a). Use of the round conical profile generates fake shading that emphasizes the orientation of fibers and can provide depth cues similar to (for example) $3D$ tubes rendered with traditional Phong shading [60]. The profile is defined as:

$$p_{cone}(\vec{x}) = \left(\frac{1}{2} \cdot \left(\frac{\vec{x}}{\|\vec{x}\|} \cdot \vec{a}\right) + \frac{1}{2}\right)^d, \|\vec{x}\| \leq 1, \|\vec{a}\| = 1 \tag{4.3}$$

where

$\vec{x} \in \mathbb{R}^2$	Texture coordinate within the unit circle
$\vec{a} \in \mathbb{R}^2$	Normalized shading axis
d	Shading power

The conical splat profile is defined with several parameters that allow for further finetuning of the shading. Figure 4.5b shows a surface plot of eq. (4.3) with $s_{min} = 0$ and $s_{max} = 1$, this surface is transformed to $0 \leq s_{min} \leq s_{max} \leq 1$ (using Eqn. 4.1), which allows for the adjustment of the overal brightness to a desired level. Shading power d is applied to make the shading more *pronounced*, where $d = 0$ is equal to applying a flat profile , and higher values of d emphasize darker shades.



(a) Viewed along the z-axis the plot of Eqn. 4.3 has the appearance of a shaded cone.

(b) 3D plot of Eqn. 4.3.

Figure 4.5: Plots of the *conical* splat profile (Eqn. 4.3) using a grayscale colormap.

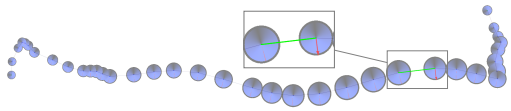
The conical profile has a single axis of symmetry (see also Fig. 4.5a), which can be aligned with a specific direction to enhance the produced shading. We propose two schemes for determining shading axis $\vec{a}$: *uniform* and *tangent-perpendicular*.

Choosing a uniform $\vec{a}$ (meaning the same direction for all rendered sprites) is the simplest approach and requires no rotation or adjustment per sprite. A uniform shading axis may however in some occasions produce shading that does not show local fiber orientation in the best possible way, due to the fact that all sprites are aligned equally.

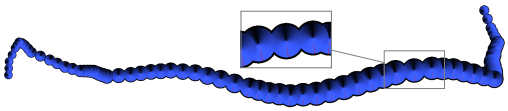
Alternatively, aligning axis $\vec{a}$ with a direction that is *perpendicular* to the tangent of the line segment (in screenspace), produces shading that emphasizes local orientation consistently throughout the image. As a result of orienting the sprites in this fashion along the polyline, a shading is obtained for which one side of the line tangent receives darker shades, and the other is shaded lighter (thereby emphasizing the line orientation visually).

Figure 4.6 shows this perpendicular splat orientation scheme on a local scale and in particular how the conical profile generates the final tube-like shading of a single fiber tract. Subfigure 4.6a shows a sparsely sampled tract and how $\vec{a}$ is aligned with the

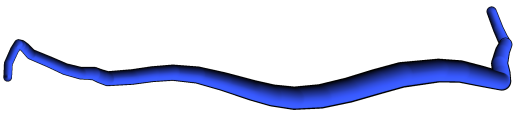
direction perpendicular to the local tangent (orthogonal vectors are shown with red indicators). Next, in figure 4.6b, the rendering is improved by adding more sprites. In this image it also becomes apparent that the shading axis $\vec{a}$ is piecewise constant (i.e. no interpolation is applied), as is further described in subsection 4.4.2.1. The third subfigure (Fig. 4.6c) shows the final shading as a result of dense sprite placement, and finally figure 4.6d shows the same image with color modulation applied.



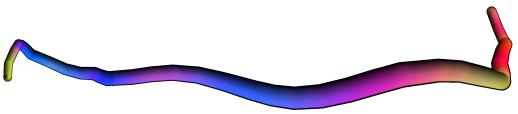
(a) Red indicators show splat orientation.



(b) Additional sprites.



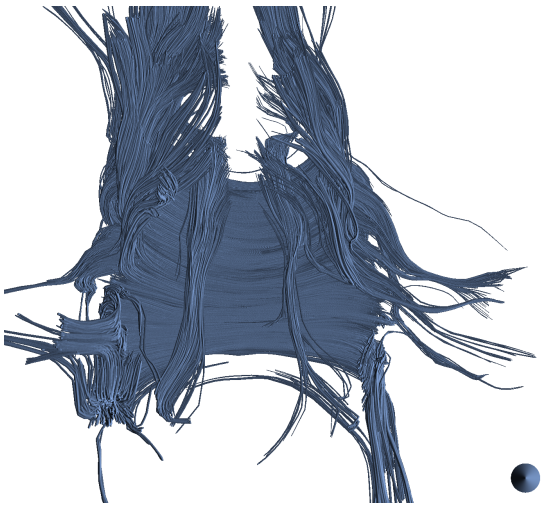
(c) Full shading with outline (uniform color).



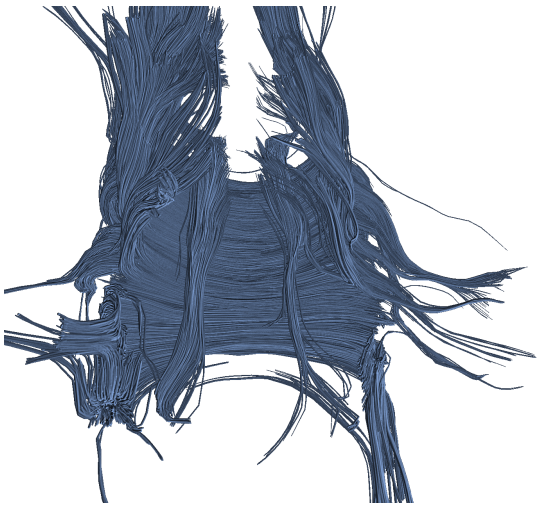
(d) Same as fig. 4.6c with color modulation applied.

Figure 4.6: Single fiber tract rendered with tangent-perpendicular conical splats.

An example of the shading axis concept on a more global scale is displayed in figure 4.7, where the left image (Fig. 4.7a) was created with a uniform $\vec{a}$ (indicated by the legend), and the right image (Fig. 4.7b) shows more consistent cues for the local fiber orientation using the tangent-perpendicular scheme.



(a) Uniform shading axis.



(b) Perpendicular to screenspace tangent.

Figure 4.7: Shading axis schemes for the conical splat profile (Eqn. 4.3), $\alpha = 0.15$.

4.4.2.1 *Computing the perpendicular shading axis*

The perpendicular shading axis $\vec{a} \in \mathbb{R}^2$ as used in Eqn. 4.3 can be computed as follows. First the normalized screenspace tangent of point x_i is determined as:

$$\tau = \frac{P_s(x_{i+1}) - P_s(x_i)}{\|P_s(x_{i+1}) - P_s(x_i)\|}$$

with $P_s : \mathbb{R}^4 \rightarrow \mathbb{R}^2$ being the screenspace projection (Eqn. 4.4) of each point. Next, the direction orthogonal to τ that lies in the viewing plane is derived as $\hat{a} \in \mathbb{R}^3 = \frac{\tau \times \hat{Z}}{\|\tau \times \hat{Z}\|}$, with $\tau_z = 0$ and $\hat{Z} = (0, 0, -1)^T$, which is the viewing plane normal. From this acquired vector $\hat{a}$, the x- and y-components can be used for the $2D$ shading profile: $\vec{a} \in \mathbb{R}^2 = \begin{pmatrix} -\hat{a}_x \\ \hat{a}_y \end{pmatrix}$.

To produce the accurate perpendicular vectors in screenspace (as shown by the red indicators in in figure 4.6a), we apply equation 4.4 to transform homogeneous coordinates to screenspace. This calculation follows a common rendering pipeline procedure for transforming between spaces, by first multiplying with the Model-View-Projection (MVP) matrix, then converting to Normalized Device Coordinates (NDC) and finally transforming to screen space coordinates as determined by the size of the viewport:

$$P_s(\vec{x}) = \left(\frac{1}{2} \frac{M\vec{x}}{\vec{x}_w} + \frac{1}{2} \right) \vec{V} \tag{4.4}$$

for which:

$\vec{x} \in \mathbb{R}^4$	homogeneous coordinate in worldspace $\begin{pmatrix} x & y & z & w \end{pmatrix}^T$
$\vec{x}_w$	w component of $\vec{x}$ for perspective division
M	MVP matrix obtained from the current camera setup
$\vec{V} \in \mathbb{R}^2$	size of the viewport (in pixels)

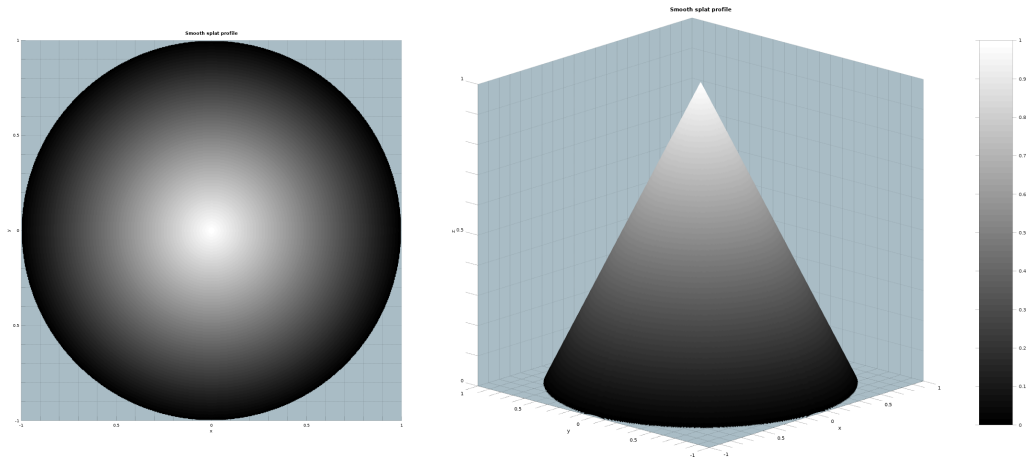
Sensible default parameters for the conical splat profile are: $s_{min} = 0$, $s_{max} = 1$ and $p = \frac{1}{2}$, in combination with the tangent-perpendicular scheme for $\overrightarrow{a}$.

4.4.3 *Smooth Splat Profile*

A smoother profile based on the distance transform

$$p_{smooth}(\vec{x}) = 1 - \|\vec{x}\|, \|\vec{x}\| \leq 1 \tag{4.5}$$

As opposed to the conical splat profile (Eqn. 4.3) the smooth profile is symmetric (across any axis), and therefore rotation of the sprites (as described in 4.4.2) does not apply.



(a) Viewed along the z-axis the plot of Eqn. 4.5 is a symmetrical splat with constant fall-off.

(b) 3D suface plot.

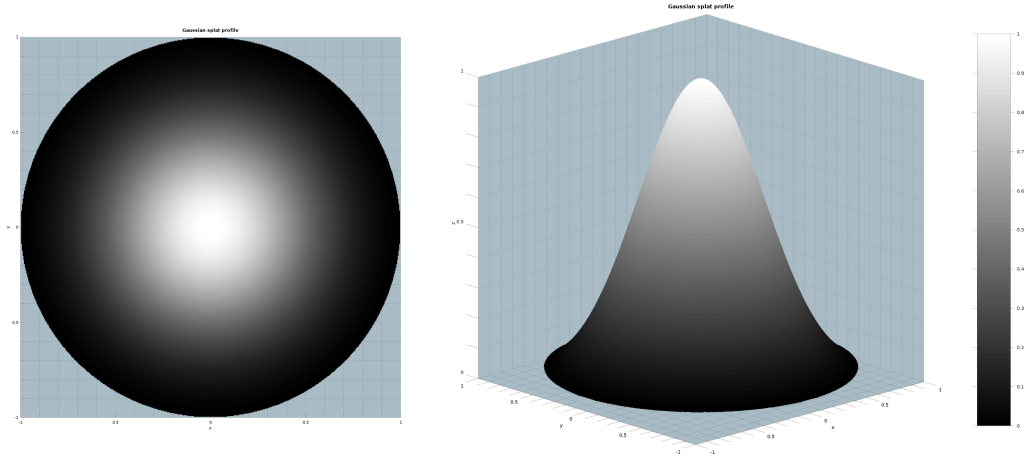
Figure 4.8: Plots of the *smooth* splat profile (Eqn. 4.5) using a grayscale colormap.

4.4.4 *Gaussian Splat Profile*

We can create mellow shading that slightly resembles the diffuse lighting aspect of Phong shading by applying a splat profile that is based on a Gaussian function. Choosing a Gaussian with $\sigma = \frac{1}{\sqrt{2\pi}}$ ensures an amplitude of 1, the proposed Gaussian profile is therefore defined as:

$$p_{gauss}(\vec{x}) = e^{-\frac{\vec{x}\cdot\vec{x}}{\pi}}, \|\vec{x}\| \leq 1 \tag{4.6}$$

Plots of equation 4.6 are displayed in figure 4.9. Like the smooth profile, the Gaussian profile is also symmetrical and can therefore not aligned with a specific axis like the conical splat profile.



(a) Viewed along the z-axis the plot of Eqn. 4.3 is a symmetrical splat with exponential fall-off.

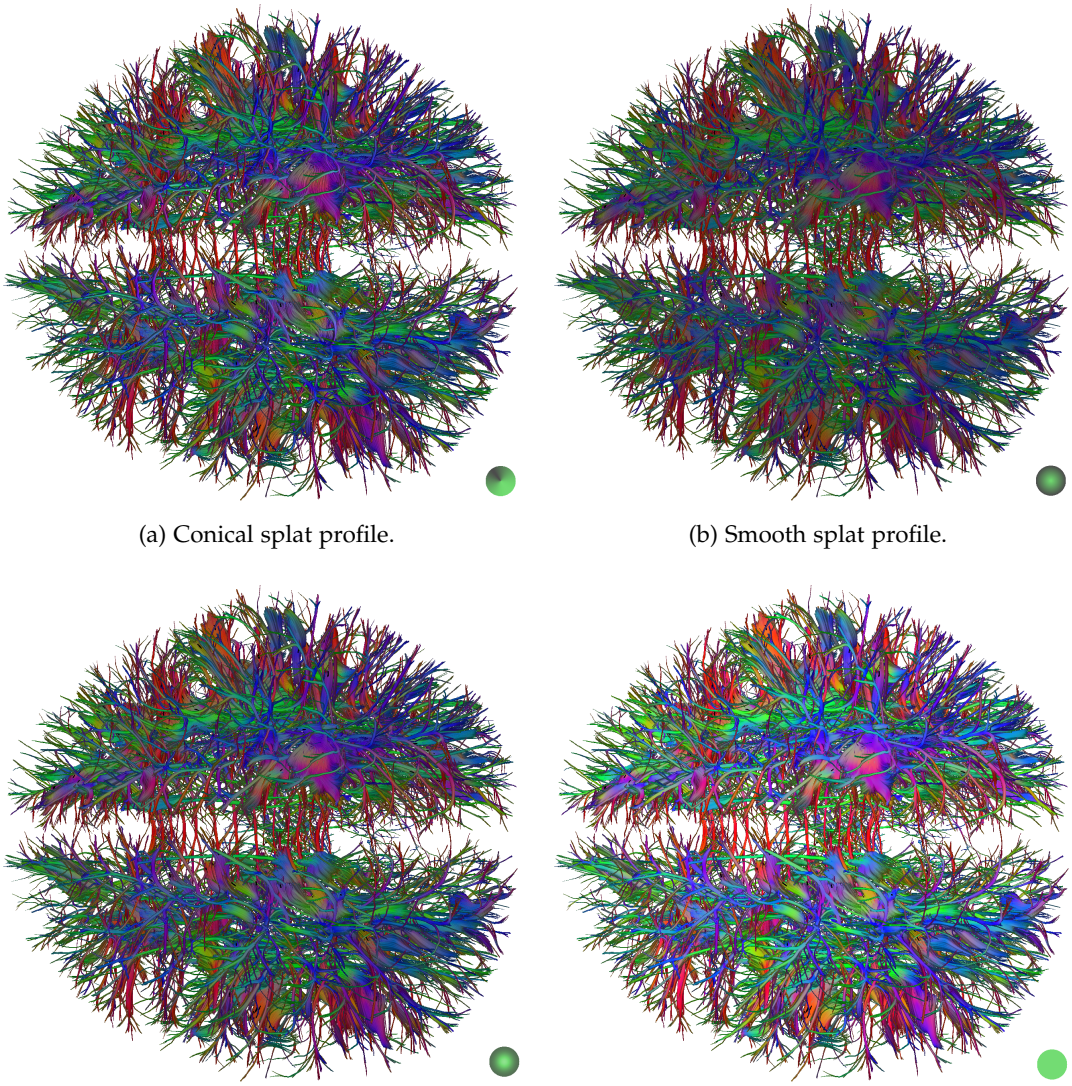
(b) 3D suface plot.

Figure 4.9: Plots of the Gaussian splat profile (Eqn. 4.6) using a grayscale colormap.

4.4.5 *Overview of Splat Profiles*

An overview of the various proposed splat profiles (equations 4.2, 4.3, 4.5 and 4.6). is shown in figure 4.10 in which the same set of simplified fiber tracts is depicted using the four different profiles. In this image it is visible that the conical profile (Fig. 4.10a) produces more pronounced tube-like shading, the Gaussian profile produces subtler mellow shading (Fig 4.10a), and that the Smooth profile provides a mix of those properties. The last subfigure (Fig. 4.10d) shows that even though the flat splat profile (Eqn. 4.2) was applied, the resulting shading is not entirely *flat*, which is a result of blending with the background ($\alpha \approx 0.2$ and $\delta = 1 \times 10^{-4}$). This shows that the final shading is not exclusively determined by the profile $p(\vec{x})$, as it depends on e.g. blending as well. Comparing the flat profile (Fig. 4.10d) with the other profiles (figures 4.10a-4.10c) clearly shows the added value (distinct shading effects) produced by the splat profiles.

In figure 4.11 a more detailed view of the profiles comparison is shown using the same bundle selection that was introduced at the beginning of this chapter (Fig. 4.1). These images show the effect of complementing the shading produced by the various proposed profiles with illustrative dark outlines (these are further discussed in section 4.8).



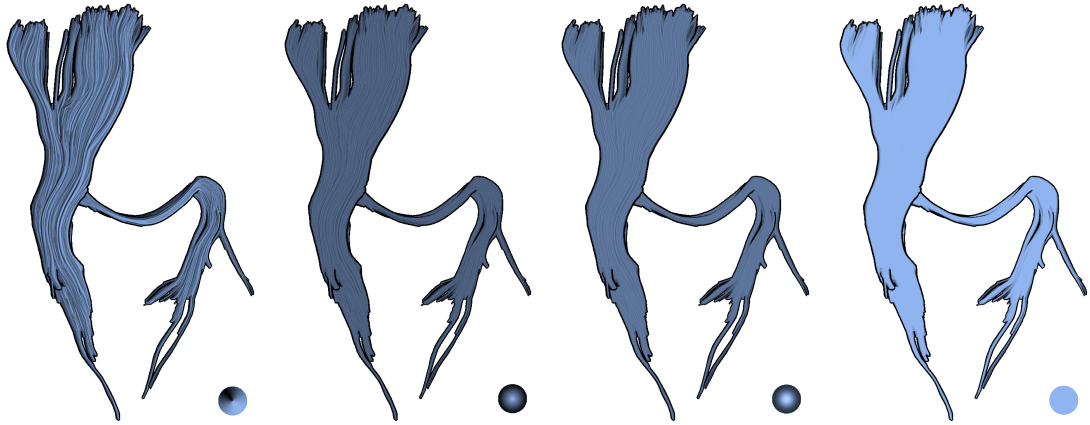
(a) Conical splat profile.

(b) Smooth splat profile.

(c) Gaussian splat profile.

(d) Flat splat profile.

Figure 4.10: Simplified fiber tracts rendered with various splat profiles (directional colormap Eqn A.1, $\alpha \approx 0.2$).



(a) Conical splat profile. (b) Smooth splat profile. (c) Gaussian splat profile. (d) Flat splat profile.

Figure 4.11: Splat profiles with outlines on a filtered bundle (uniform color).

4.5 SPRITE SIZE PARAMETERIZATION FOR COMPACT SHAPE RECONSTRUCTION

As explained earlier, the size of all rendered sprites is important since we want to close gaps in surface-like DTI structures. For this, we can modulate the size of the sprites in screen space based on the DTI data. Moreover, this type of modulation is needed to achieve correct perspective effects, e.g. structures farther from the viewer should be rendered with smaller sprites.

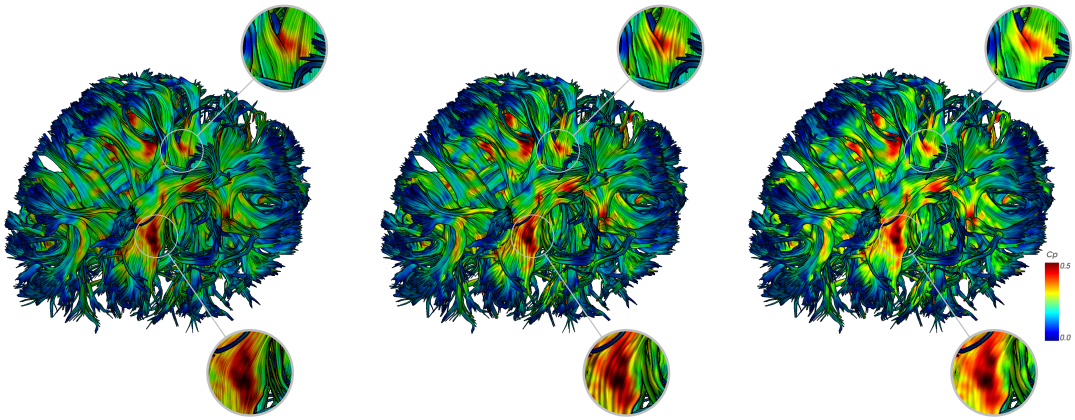
DISTANCE ATTENUATION Since the size of a pointsprite in worldspace is not automatically subject to perspective projection (in modern versions of OpenGL), it is required to explicitly define a method for point size distance attenuation in order to provide adequate depth cueing and to make the rendering more zoomlevel independent. We apply distance attenuation using the following heuristic:

$$a_{dist} = \sqrt{\frac{1}{d + d^2}} \tag{4.7}$$

Where d is the projected depth value obtained after perspective projection. Equation 4.7 is based on distance attenuation commonly used in lighting [55].

PLANAR EMPHASIS In conjunction with the distance attenuation (Eqn. 4.7) we can also introduce an attenuation factor based on scalar data from the DTI dataset, in particular the planar estimate c_p (Eqn. 2.3) which indicates regions of higher planar anisotropy.

By applying this form of scalar attenuation it becomes possible to emphasize these regions of higher planar anisotropy and increase the possiblity of creating the appearance of a surface without gaps in these areas. The complete point size attenuation function $S(\vec{v}) : \mathbb{R}^3 \rightarrow \mathbb{R}^+$ (as performed in the vertex shader, see section 4.7), featuring both scalar- and distance attenuation, is defined as:



(a) Conical splat profile.

(b) With additional planar emphasis ($c_p \geq 0.2$).

(c) With mixed profiles.

$$p(\vec{x}) = \begin{cases} c_p < 0.2 & p_{cone}(\vec{x}) \\ c_p \geq 0.2 & p_{gauss}(\vec{x}) \end{cases}$$

Figure 4.12: Planar emphasis demonstrated on a slice-filtered set of fiber tracts shown using a rainbow (c_p) colormap.

$$S(\vec{v}) = max\left(\left(b_{size} + \begin{cases} c_p \geq t & p_{size} \\ c_p < t & 0 \end{cases}\right) \vec{v}_{size}\, a_{dist},\, s_{max}\right) \tag{4.8}$$

where

$\vec{v}$	Vertex
a_{dist}	Distance attenuation factor (Eqn. 4.7)
$\vec{v}_{size} \in [0,1]$	Relative point size (an attribute of $\vec{v}$)
s_{max}	Uniform maximum point size (in pixels)
b_{size}	Base point size (in pixels), applies to all points
p_{size}	Additional point size (in pixels), applied to vertices that pass the c_p threshold $t \in [0,1]$

Figure 4.12 shows effect of applying the planar emphasis as described above. First, figure 4.12a shows a slice filtered set of fiber tracts using a rainbow colormap (showing c_p) and the conical splat profile for shading. Next in figure 4.12b, the sprite size was incread for regions where $c_p \geq 0.2$. In the third figure the shading was adjusted as well by applying mixed splat profiles. In this case, the gaussian splat was used to shade the sprites in emphasized regions (again $c_p \geq 0.2$) and the conical profile for all other sprites.

LINE TAPERING Similar to the visual enhancements described in [15] we also optionally apply line tapering, a useful construct to de-emphasize polyline endings by gradually narrowing line ends. In this case a point-size modulation is applied to a predefined number of points, $n_{tapered} \geq 1$. For vertex v_i with index $i \in [0,n)$, $taper(v_i) \in [0,1]$ is defined as:

$$taper(v_i) = \min(\min(i, i-(n-1)) \cdot \frac{1-s_{min}}{n_{tapered}}, 1) \tag{4.9}$$

Where s_{min} is the minimum factor applied to both end-points. Equation 4.9 yields an interpolation between $[s_{min}, 1]$ for tapered points and 1 for unaffected points. We found that using $n_{tapered} = 5$ and $s_{min} = 0.5$ generally yields acceptable results.

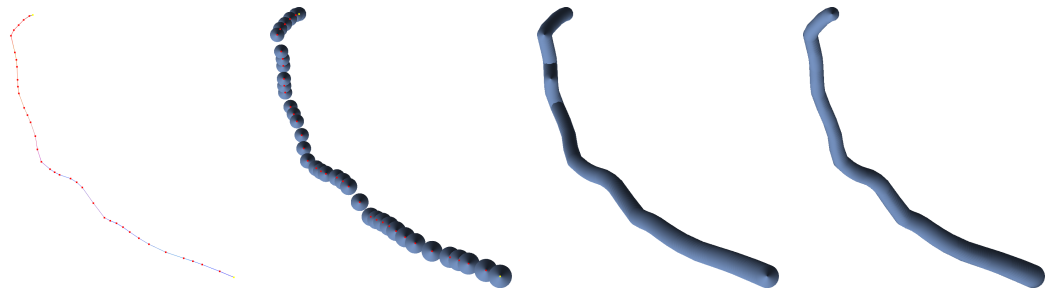
4.6 SPRITE DENSITY MODULATION FOR RENDERING QUALITY CONTROL

As explained at the beginning of the current chapter (see section 4.2), we want to use billboarded sprites to convey the DTI tube and surface-like structures. For this, we however have to decide where to place these sprites, and how many of them we need. Our only $3D$ spatial information comes in terms of the DTI fiber tracts introduced in section 2.3. Hence, we need to place our sprites along these curves. The remaining question is how many sprites we need, or how densely should we sample these curves with $2D$ sprites.

Recall that we represent DTI fibers in an already sampled manner, i.e., as a polylines. However, the sampling density of these polylines may not be optimal, for various reasons. For instance the sampling density may be a given value for the input fiber dataset (originating from parameters such as the integration step size or applied spline-filtering [43, 69]), or it can be determined by the simplification as described in chapter 3. Moreover, since billboarding is a screenspace technique, while these polylines are sampled in world ($3D$) space, we may need to adapt the sampling of polylines with sprites as we interactively zoom-in/out while viewing the data.

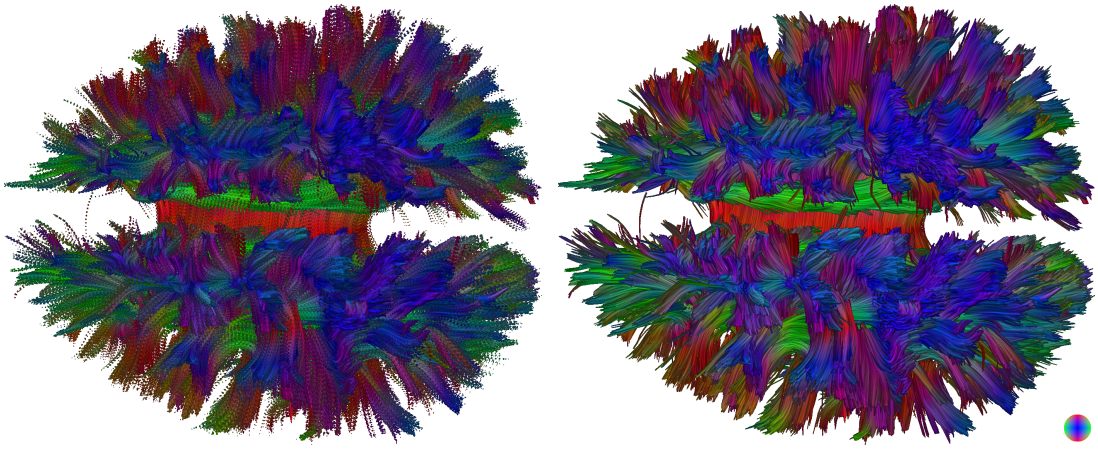
In figure 4.13 the sprite density is demonstrated on a single fiber tract (each subfigure depicts the exact same $3D$ fiber with the various rendering parameters, no colormapping was applied). The first image (Fig. 4.13a) shows the input fiber and samling points for reference (sampling distance is approximately 1 voxel). Figure 4.13b shows the same fiber, with sprites shaded with the conical splat profile (Eqn. 4.3) rendered at these sampling points. As is visible, the the number of sprites was insufficient to create a solid structure. Next in figure 4.13c the sampling density was increased by a factor of 7 (resulting in the output of 7 additional sprites per line segment). In the last figure (Fig. 4.13d), blending was applied as described in section 4.3. With blending enabled the previously shown artefacts disappear and the result is a smoothly shaded polyline.

The sprite placement density requires careful tuning since besides *undersampling*, which may lead to gaps in the rendered structures and low quality visualization (as shown in figure 4.14a), there is also the possibility of *oversampling* which will have a negative impact on rendering performance. We propose to approach this by an on-the-fly resampling of the $3D$ polyline fibers to create splat placement points. A convenient method for adding additional pointsprites is to apply tessellation to the line segments, which is comparable to resampling the polyline on the fly. The implementation of such (optional) tessellation is discussed in the next section (section 4.7). Figure 4.14 shows a side-by-side comparison of an undersampled rendering (Fig.



(a) Line rendering, sites marked as red dots. (b) Pointsprites with insufficient density. (c) Same as (b) but with $\times 7$ more points. (d) Same as (c) but without artefacts due to blending ($\alpha = 0.55$).

Figure 4.13: Pointsprites density demonstrated on a single fiber tract.



(a) Lower quality due to undersampling. (b) Improved rendering quality (4x more sprites).

Figure 4.14: Sprite placement density reflected in rendering quality.

4.14a, interpoint distance $\approx 1 voxel$) and a higher quality rendering with a ($4x$) higher sampling rate (Fig. 4.14b).

4.7 IMPLEMENTATION

We implement all steps described in this chapter efficiently using OpenGL techniques including various shaders, indexed VAO's, tessellation and alpha blending. The multi-pass rendering approach as introduced in section 4.3 (in particular see the diagram in Fig. 4.4) is displayed in a source code snippet (C++) in listing 4.1.

4.7.1 *Line Segments to Points Conversion*

As explained earlier the fiber tract data comes in the form of polylines, which are naturally represented by line primitives. For the sprites to be rendered along a line, the GPU can convert the line primitives to *points* on the fly. This can be achieved by configuring the primitive generator to generate a point primitive for each vertex dur-

Listing 4.1: Alpha blended pointsprites rendering (C++).

```
void depthSamplingPass()
{
    // disable blending, but enable depth-test and depth-write
    glDisable(GL_BLEND);
    glEnable(GL_DEPTH_TEST);
    glDepthMask(GL_TRUE);

    // render fiber tracts as pointsprites with flat shading
    flatShader.enable();
    fiberTracts.render();
}

void shadingPass()
{
    glEnable(GL_BLEND);        // enable blending
    glDisable(GL_DEPTH_TEST);  // disable depth test
    glDepthMask(GL_FALSE);     // don't write to the depth buffer
    glBlendEquation(GL_FUNC_ADD);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    // render fiber tracts as pointsprites with splat profile shading + blending
    splatProfileShader.enable();
    fiberTracts.render();
}

void compositingPass()
{
    compositingShader.enable();
    // render contents to screen by means of a viewport aligned quad
    fullScreenQuad.render();
}

void render()
{
    // multi-pass rendering
    depthSamplingPass();
    shadingPass();
    compositingPass();
}
```

ing primitive assembly. For OpenGL implementations this referred to as point-mode output (a tessellation layout qualifier) which is enabled in the Tessellation Evaluation Shader (TES) (see also fig. 4.15) and can produce equally spaced points along a line segment. Once produced, these points are rasterized as shaded sprites (as described in section 4.4).

An outer tessellation level of 1 will result in a point primitive for the start- and endpoint of a line segment, thereby matching the original sampling density of the fiber tracts. By using a higher tessellation level (> 1) additional points can be generated along each line segment, which is a flexible on-the-fly resampling mechanism in the case that the uniform arc-length sampling is insufficiently dense (as discussed in section 4.6).

4.7.2 *Shaders*

A diagram of the shading pipeline of is shown in figure 4.15. The diagram shows four shading stages (including a vertex shader, a Tessellation Control Shader (TCS), a TES and a fragment shader for each geometry pass), and in particular where the conversion to points (as described in the previous paragraph) fits in.

First the vertex shader performs common per-vertex operations such as screenspace projection (using the applied MVP matrix) and the output of per-vertex attributes. This includes the sprite size (pointsize) parameterization as explained in section 4.5. The second stage is performed in the TCS, which determines per patch output variables (e.g. the local shading axis, section 4.4) and prepares the interpolation performed in the next stage (TES). The TES defines the third stage in which linear interpolation of position, pointsize and color is performed. Finally the fragment shader produces the shading for the rasterized points (which we call sprites), by means of the splat profiles as described in section 4.4.

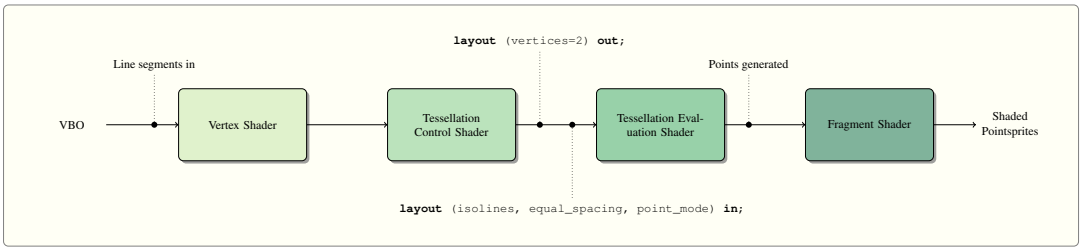


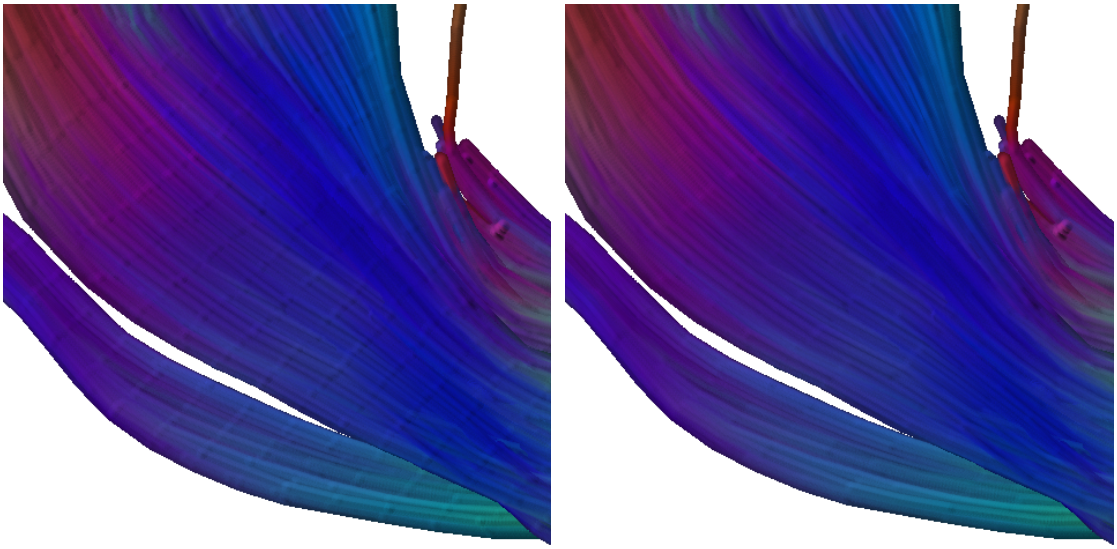
Figure 4.15: Diagram of the alpha blended pointsprites shading pipeline.

4.7.3 *Preventing Redundant Point Output*

Since the fibers are modelled as polylines, and each line segment shares one vertex, it becomes important to prevent duplicate (redundant) point output from the primitive generator. This is particularly the case when applying tessellation as described above in combination with alpha blending, because rendering a sample point twice will result in visual artefacts (see Fig. 4.16 where Fig. 4.16a displays the artefacts, and Fig. 4.16b is the same image with proper tessellation). In order to solve this we set the pointsize for every first tessellated point of a line segment to 0 (thereby preventing output during the fragmentation stage for this vertex), with the exception of the very first line segment of each polyline. As the tessellation is applied to each line segment individually (without information available regarding the polyline) we store the parametric arc-length position $t = \frac{i}{N-1} \in [0, 1]$ as a vertex attribute (with $t = 0.0$ for the first vertex, and $t = 1.0$ for the last), and use this information to prevent duplicate points output in the TES.

4.7.4 *Alternate Depth Testing*

In the first pass (as shown in listing 4.1) all tracts are rendered as sprites to a texture ($RT_{rgb\alpha}$) using conventional depth buffering and flat shading. The depth buffer, which on completion contains the topmost depth value for each frament, is sampled in a secondary texture (RT_d). Figure 4.17a shows the flat shaded framebuffer contents of $RT_{rgb\alpha}$ after the depth sampling pass (again see the diagram of fig. 4.4). During the second pass, the shading pass, these contents are blended with fragments that pass an alternate depth test, which results in shaded rendering as shown in (fig. 4.17b). This custom depth test is defined as:



(a) Rendering artefacts caused by duplicate tessellation output. Each shared vertex is rendered twice.

(b) Skipping redundant points eliminates the artefacts.

Figure 4.16: Visual artefacts as a result of redundant tessellation output.

$$depthtest(\vec{f}) = \left\{ \begin{array}{cc} true & if\ (d' - \delta) < RT'_d \\ false & otherwise \end{array} \right\} \tag{4.10}$$

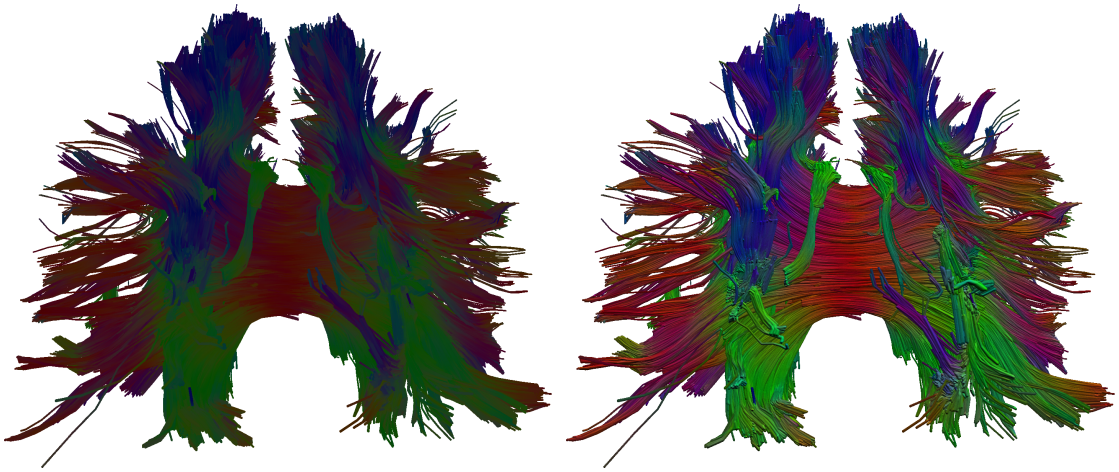
where

$\vec{f} \in \mathbb{R}^4$	Fragment
$d' \in [0,1]$	Linear depth value for fragment $\vec{f}$
$RT'_d \in [0,1]$	Linear depth buffer value for the fragment coordinate of $\vec{f}$
$\delta \in [0,1]$	Uniform probe depth

By applying equation 4.10 all fragments within the range defined by δ and the depth texure $[RT'_d, RT'_d + \delta]$ are blended (in no particular order).

The linear depth value d' is computed from the (non-linear) gl_FragCoord.z, and RT'_d is likewise computed from the non-linear depth that is stored in RT_d (the depth-texture). We use only these linear depth values for depth testing (Eqn. 4.10) in order to make the user-defined probe-depth δ consistently applicable at any camera distance. A conversion is in this case required because GPU hardware depth buffers don't use a linear representation of the distance, a mechanism that allows for increased precision closer to the near viewing plane [31]. The conversion from non-linear depth to linear depth values is performed according to equation 4.11, where D_{near} and D_{far} are the depths of the near and far plane respectively.

$$d' = \frac{d_{near}(d+1)}{d_{far} + d_{near} - d(d_{far} - d_{near})} \tag{4.11}$$



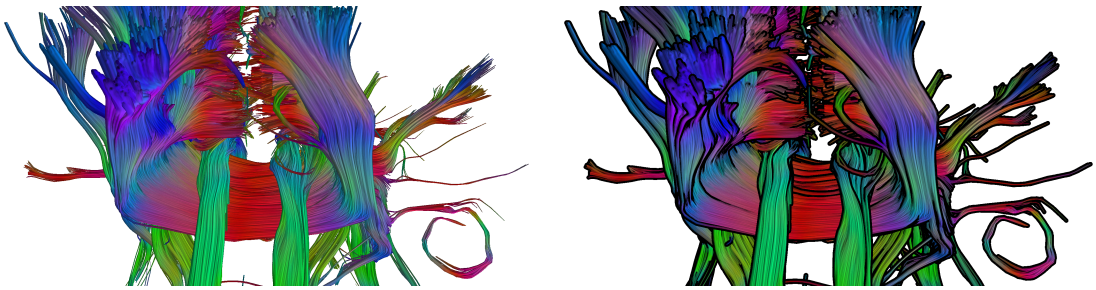
(a) RGBA framebuffer contents after the first pass. (b) Shading after the second rendering pass ($\alpha = 0.1$, $\delta = 5 \times 10^{-3}$).

Figure 4.17: Multi-pass blending (Shown with a directional colormap).

4.8 OUTLINES FOR SIMPLIFIED VISUALIZATION

A final addition to the alpha blended pointsprites method is the rendering of outlines. Rather than creating halos around individual fibers, a common illustrative technique is to create outlines around *bundles* or groups of fibers [15, 44]. The effect of such outlines is that groups of co-linear fibers are emphasized, which is most apparent at the crossing or overlap of bundles.

Outlines provide additional cues to the viewer as to which structure appears in front of another, which is a helpful property for clearly conveying structural relations between bundles. An illustration of combining the pointsprite rendering with additional outlines is given in figure 4.18, where we show various crossing bundles of a slice-filtered set of fiber tracts with and without black outlines. Central in this image are the *left-* and *right cingulum*, that run above and across a section of the *corpus callosum*.



(a) No outlines. (b) Thick outlines ($bsize = 3px, t = 15px$).

Figure 4.18: Outlines applied to crossing bundles.

4.8.1 *Determining Outline Fragments*

We can use the sprite size parameterization described earlier (section 4.5) and the fact that we render all sprites in two consecutive passes, to incorporate the creation of outlines in the rendering method with only some minor adjustments of the rendering parameters. Moreover, rather than spending significant effort on the creation of outlines specifically, in this case much of the required information is already available and it is therefore possible to add outlines with limited overhead.

The outlines can be created by dilating the rendered sprites: if we apply an increased size $S' = S + t$ (with S eq. 4.8) during the first pass (the depth sampling pass), but retain the originally intended size S during the second pass (the shading pass), we can use the difference in size to create outlines around areas of similar depth in the final fragmentation stage.

By sampling the depth texture ($RT_d \in [0,1]$, for which we have defined $d_{near} = 0$ and $d_{far} = 1$) we can discern foreground fragments ($d < 1$) from background fragments ($d \equiv 1$). Beside the depth texture we also have our RGBA rendering target ($RT_{rgb\alpha} \in [0,1]^4$) for which we have set $\alpha = 0$ as a default. Following this analysis, we define our set of outline fragments as:

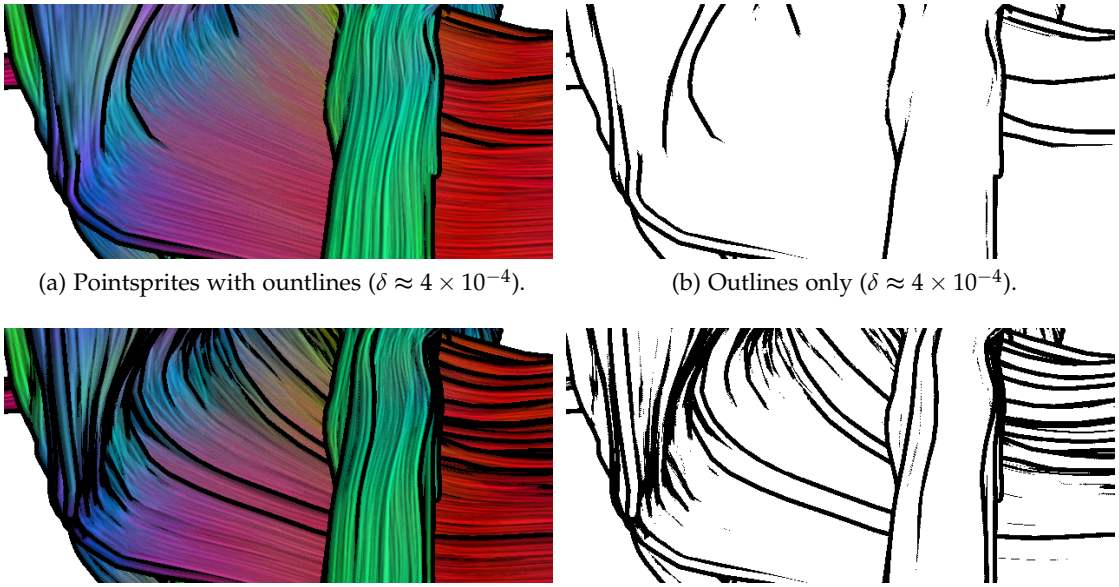
$$F_o = \{\forall f \in RT_{rgba} | \, \alpha = 0 \wedge RT'_d < 1\} \tag{4.12}$$

, being all fragments that were output from the first pass, but that were not shaded in the following pass.

4.8.2 *Outline Shape Control*

The shape of the outlines is controlled by two parameters, first the thickness (t) of the outlines is controlled by the sprite size difference, $t = S' - S$, and secondly the area around which contours are drawn is determined by the uniform probe depth δ (see section 4.7).

Figure 4.21 shows outlines with a uniform black color and a thickness of 2 *pixels* rendered on a slice filtered set of fiber tracts. The left image shows a rendering without outlines ($b_{size} = 4$), and the middle image has outlines added ($t = 2$). For clarity the right image shows only the outlines, which clearly visualizes the difference between the first two images. By increasing the δ-parameter more fragments will pass the depth test (Eqn. 4.10). As a result of this, less of the interior contours will be incorporated in the set of outline fragments (Eqn. 4.12). To illustrate this, figure 4.19 shows a comparison for two levels of δ with $t = 2$ (a detailed view of Fig. 4.18), where the top row has a smaller $\delta \approx 4 \times 10^{-4}$, and the bottom row has an increased probe-depth $\delta \approx 4 \times 10^{-1}$.



(a) Pointsprites with ountlines ($\delta \approx 4 \times 10^{-4}$).

(b) Outlines only ($\delta \approx 4 \times 10^{-4}$).

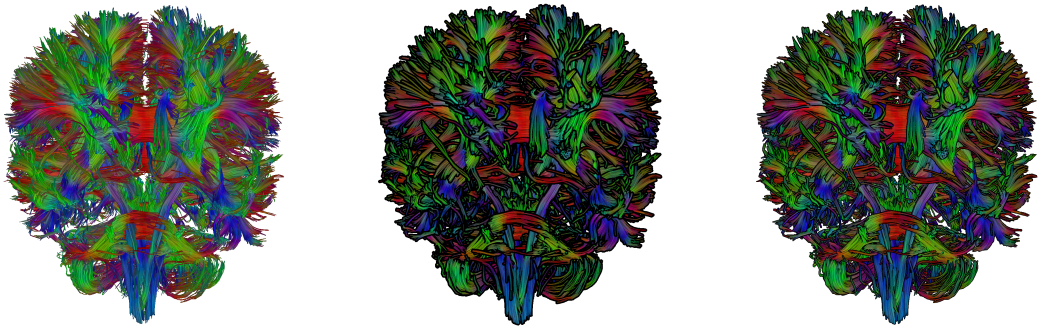
(c) Pointsprites with outlines ($\delta \approx 1 \times 10^{-4}$)

(d) Outlines only ($\delta \approx 1 \times 10^{-4}$).

Figure 4.19: Detailed view of outlines for two levels of uniform probe-depth δ.

4.8.3 *Distance Attenuation*

A uniform outline thickness may in some usecases put to much emphasis on stray fibers or bundles further away from the viewer. This can be resolved by applying distance attenuation (Eqn. 4.7) to sprite size S', which will result in depth-dependent outlines. These distance attenuated outlines are (like all other elements) scaled according to distance, and therefore provide more depth cues to the viewer. A comparison of outline approaches is displayed in figure 4.20, where figure 4.20a has no outlines, figure 4.20b has uniformly sized outlines ($t = 5px$) and figure 4.20c shows the subtler distance attenuated outlines.



(a) No outlines.

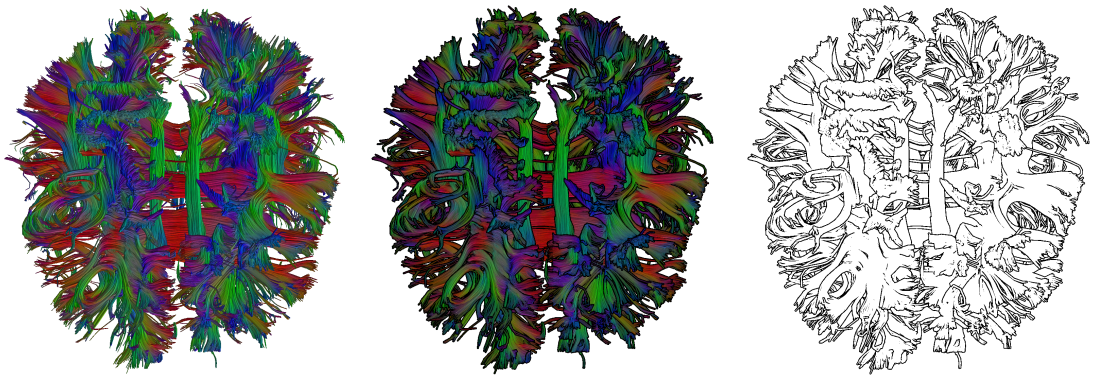
(b) Uniformly sized outlines.

(c) Depth attenuated outlines.

Figure 4.20: Outlines on a whole-brain fiber set.

4.8.4 *Coloring*

For the coloring of outlines we propose two options: 1) applying a uniform color, or 2) reusing the already created framebuffer contents (as in Fig. 4.17a) to color the outlines matching the currently applied colormap. In general it makes sense to use a uniform color (such as black or white) because this can clearly separate outlines from both background- and foreground fragments using a distinct color. Creating outlines that match the colormap (with darker shading) works well with clustering-based colormaps (e.g. as proposed by Otten et al. [44]).



(a) No outlines. (b) Black outlines. (c) Outlines only.

Figure 4.21: Rendering outlines on a slice-filtered set of fiber tracts.

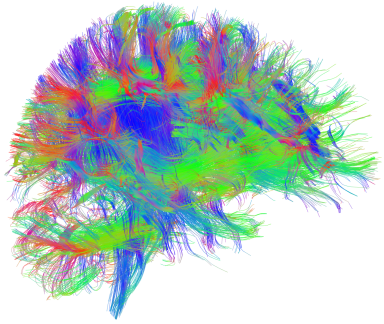
5 RESULTS

In this chapter we present various results of our simplification approach (as discussed in chapter 3), shown using several different rendering methods. These rendering methods are: alpha blended lines (as introduced in chapter 3), DDH by Everts et al. [15] (as discussed in section 2.4, and 4.1), and our own method Alpha Blended Pointsprites (ABP) (described fully in chapter 4).

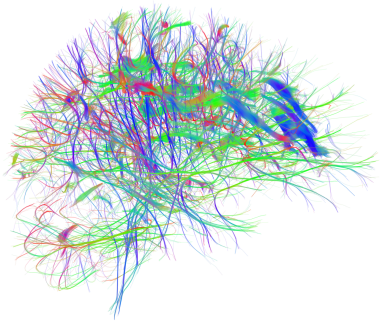
For our pointsprites rendering method (ABP) we show both the multi-pass blending version (as described in 4.3), and a single-pass version for reference. It should be noted that both the single-pass ABP and alpha blended lines are rendered with sorted geometry (ordered back-to-front) without depth buffering, while DDH and ABP do not require such explicit render ordering.

5.1 SIMPLIFICATION WITH PLANAR RESEEDING

Figure 5.1 shows a side-by-side comparison of simplification results using various rendering methods. The left column shows the original set of fibers, and the right column shows the simplified fibers with planar reseeding applied. Each column shows the exact same set of ($60K$) fibers from precisely the same point of view. Applied bundling parameters in this case are: $P_R = 13, P_N = 15$, $\phi = 0.25$ and $\sigma = 1$ (without relaxation, endpoints were bundled). Reseeding was performed in regions where $c_p \geq 0.25 \wedge c_l < 0.2$.

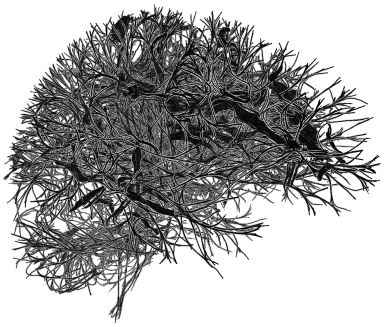


(a) Alpha blended lines (original).

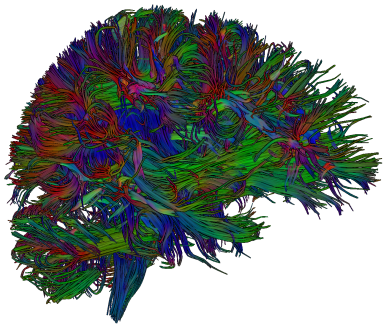


(b) Alpha blended lines (simplified).

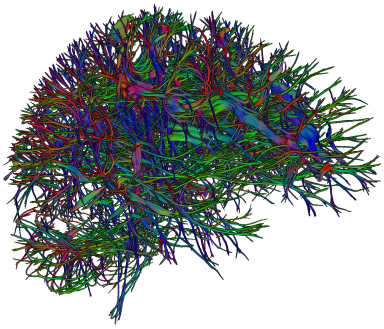
(c) DDH (original).



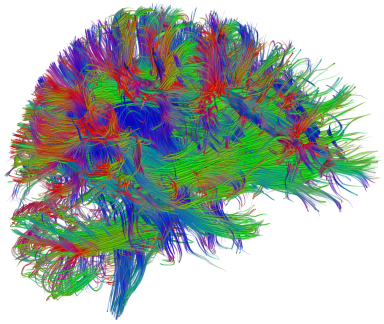
(d) DDH (simplified).



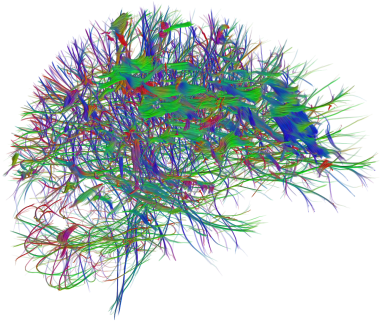
(e) ABP (original).



(f) ABP (simplified).



(g) Single-pass ordered ABP (original).



(h) Single-pass ordered ABP (simplified).

Figure 5.1: Comparison of various rendering methods, shown for original fibers (left column) and simplified fibers with planar reseeding (right column).

As is visible in the top row of figure 5.1, alpha blended lines an effect that can be increased by lowering the uniform α value. Lowering alpha is generally complementary to the bundling, yet has the disadvantage that the reseeded structures become transparent.

The DDH method (second row in Fig. 5.1) clearly emphasizes these densely reseeded structures by visually merging the tracts and displaying halos around the bundles, an effect that can be enhanced by decreasing e.g. the strip width w_{strip}, or increasing the strip depth d_{max} to suitable levels (for details see [15]). Single (or stray) fibers are emphasized by DDH which results in a sub-optimal reduction of occlusion when compared to e.g. blended lines (top row).

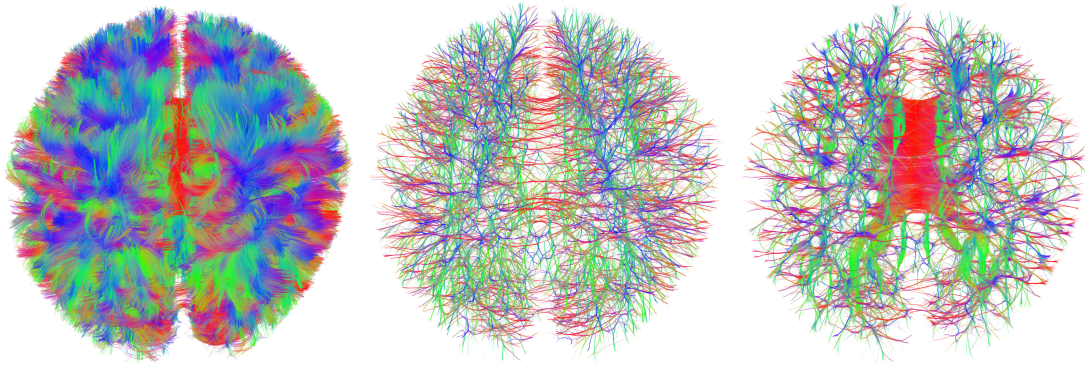
The ABP method (third row) places similar emphasis on individual fiber tracts, yet adds more shading and color coding compared to DDH. The rendered outlines are particularly suitable for emphasizing the reseeded regions, which is most notable in Fig 5.1f. By selecting an appropriate value for δ (in this case, $\delta = 4 \times 10^{-4}$) thicker bundles, such as the reseeded regions, can be highlighted. Lowering δ results in more outlines drawn around specific elements (as described in section 4.8), thereby emphasizing individual tracts, rather than bundles.

The final row shows the single-pass version of ABP. This approach does not feature outlines, and is comparable to blended lines (shown in the top row) in terms of the performed background blending. By varying the blending factor it is possible to de-emphasize low-density areas, which is beneficial for conveying the bundling of the tracts. The single-pass ABP improves over line rendering by adding more options for shading (as discussed in chapter 4) and by enhancing the depiction of solid, surface-like structures.

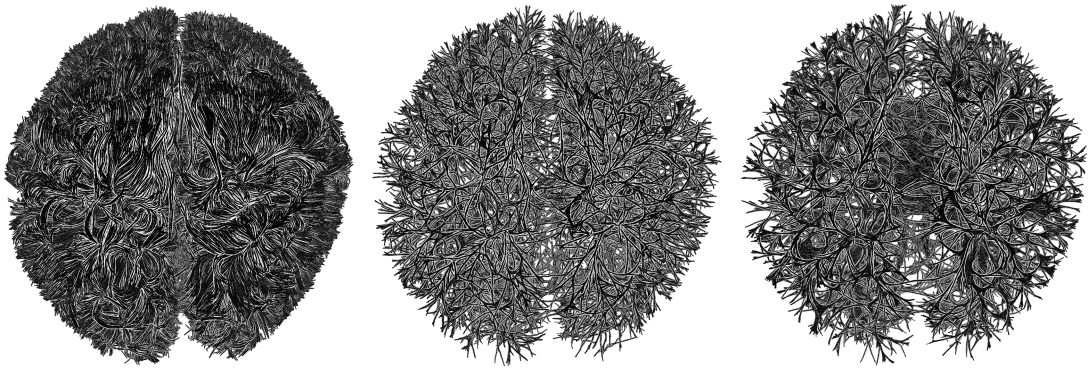
5.2 SIMPLIFICATION WITH DTI EDGE PROFILES

Figure 5.1 shows a side-by-side comparison of simplification results based on DTI edge profiles, using various rendering methods. The left column shows the original set of fibers, the middle column shows the simplified fibers with uniform bundling applied and the right column shows simplification using the *FA*-constrained edge profile.

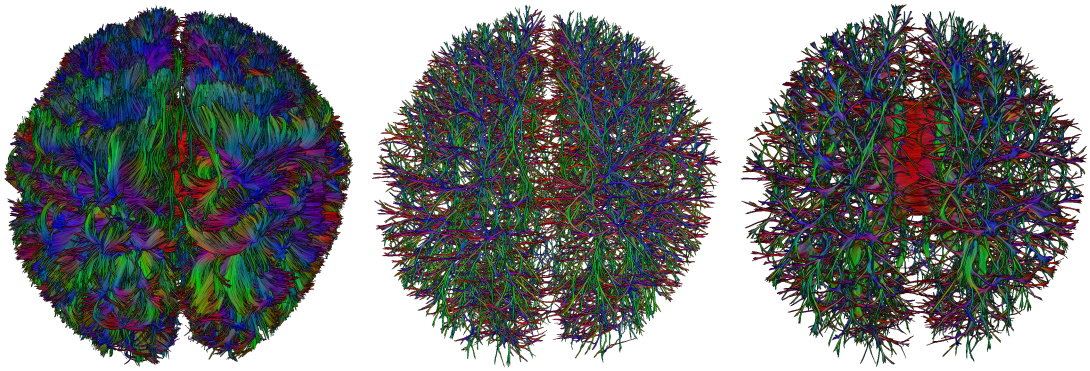
Again each column shows the exact same set of ($100K$) fibers from precisely the same point of view. The middle and right column shown bundling using the parameters: $P_R = 13, P_N = 15$, $\phi = 0.2$ and $\sigma = 1$ (without relaxation, endpoints were bundled). The right column shows the same bundling that was limited using the *FA*-constrained edge profile (Eqn. 3.6, with threshold $FA \geq 0.7$).



(a) Lines (original). (b) Lines (simplified). (c) Lines (*FA*-constrained).



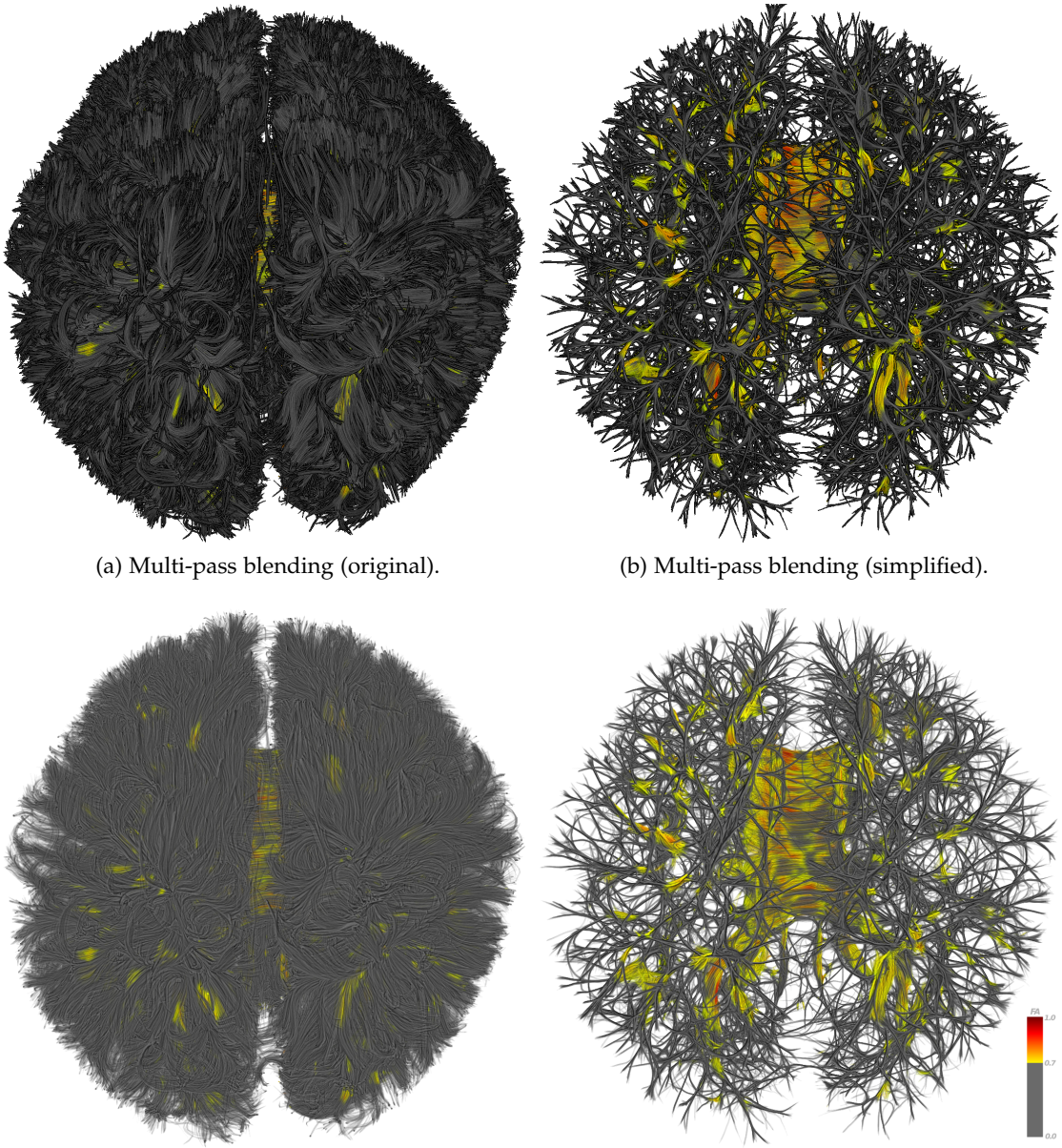
(d) DDH (original). (e) DDH (simplified). (f) DDH (*FA*-constrained).



(g) ABP (original). (h) ABP (simplified). (i) ABP (*FA*-constrained).

Figure 5.2: Comparison of various rendering methods, shown for original fibers (left column), simplified fibers (middle column) and fibers simplified with *FA*-constrained edge profile (right column).

In order to clearly show the difference between ABP and single-pass sorted ABP, figure 5.3 shows the same simplified fibers as before (Fig. 5.2) using both of these methods. The main difference between the two rendering approaches is in the ability to blend background fragments. Whereas ABP emphasizes each fiber and bundle (e.g. Fig. 5.3b), the single pass (sorted) version is able to deemphasize low-density areas (shown in Fig. 5.3d).



(a) Multi-pass blending (original).

(b) Multi-pass blending (simplified).

(c) Single pass blending (original).

(d) Single pass blending (simplified).

Figure 5.3: Alpha blended pointsprites: multi-pass rendering (top) vs. single pass rendering (bottom).

6 DISCUSSION

In this chapter we discuss the various results presented so far along the two major axes of our research, which are: *simplification* and *rendering*.

6.1 SIMPLIFICATION

We found that choosing a relatively small density-kernel size (radius 5..15 voxels) produce nice results that really convey the original topology of the brain. A larger starting kernel size P_R results in more "*abstract*" simplification (e.g. Fig. 3.5c) for which the relation to the original fiber tracts is less clear.

We also found that $P_N \in [15, 20]$ bundling iterations and a sampling distance of approximately $0.5 \leq \sigma \leq 1$ voxels is sufficient to achieve sufficient simplification.

Our approach to endpoints bundling is very useful in the context of fiber tract bundling, in particular for larger sets of tracts created with random seeding within each voxel. As fan-like clusters of endpoints are prominent in such datasets, occlusion is reduced much further, which leads to a cleaner simplification.

Both the planar reseeding approach and the proposed DTI (restrained) edge profiles allow the uncovering of distinct structures, and permit these to be shown in the context of simplified layout of the entire dataset.

The computational performance of our bundling approach is promising, and shows running times under 5 seconds for larger sets of fibers (e.g. $500K$ tracts), as was listed in table 3.2. Sub-second running times ($< 1s$) are also feasible depending on the chosen parameters, which predominantly are: volume resolution, P_R and the number of points in the dataset. The measured running times show that we can quickly generate a simplified model, thereby ensuring that interactive exploration of datasets and simplification parameter space is possible (which fulfills both S1 and S2).

We found that using the original DTI resolution (e.g. as listed in table 3.1) to approximate ρ (Eqn. 3.1) does *not* produce high quality bundling, however scaling this resolution up by a factor of 2..4 is sufficient. Clipping the DTI volumes based on the brain mask (as discussed in section 3.1) is useful, as it helps to prevent density estimation in redundant voxels.

6.2 RENDERING

The various rendering methods shown in the previous chapter (Ch. 5) are only partially comparable, since these have quite varying characteristics. We therefore discuss the methods in terms of their *properties*, rather than on a pixel-precision level, in rela-

	Lines	DDH	**ABP**	**ABP***
R1: Show the nature of local brain structures	+/-	+	**+**	**+**
R2: Convey shape and orientation	-	+	**++**	**++**
R3: Allow depiction of data attributes	+	-	**+**	**+**
R4: Allow rendering at interactive framerates	-	+	**+**	**-**
Complement S1: reduction of occlusion	+	-	**-**	**+**

Table 6.1: Summary of rendering requirements fulfilment.

tion to the defined rendering requirements as listed in subsection 1.1.3.

* *single-pass sorted rendering*

From the comparison showed in figures 5.1 and 5.2 we can make various observations. First we can see that DDH and ABP show the local nature of structures (requirement R1) where the alpha blended lines is less suitable for that purpose, due to limited shading.

Secondly, both DDH and ABP also convey shape and orientation of the fiber structures quite well (requirement R2), by applying halos/outlines and additional shading. ABP slightly improves over DDH in this area, as each surface-like structure is shaded with additional cues and coloring, whereas DDH mostly groups these structures into black bundles with white outlines (this is most notable by comparing Fig. 5.1d with Fig. 5.1f).

Thirdly, a trivial observation can be made that DDH, in contrast to all other methods, does not allow color coding, and for this reason cannot depict data attributes very well. As of such, DDH does not fulfill our third rendering requirement R3.

DDH is by far the most performant (high Frames per Second (FPS)) rendering method of the compared to all others. ABP also allows for interactive framerates, yet employs a more elaborate shading pipeline and thus is slower. Blended lines, and single-pass ABP both use explicit sorting, which is workable, yet not scalable. For this reason, these methods do not allow smooth user interaction and thus do not fulfill requirement R4.

In addition to the above, we can see that sorted rendering with alpha blending shows the bundling result very clearly. This is due to the fact that low density areas are implicitly *tuned out*, which emphasizes the bundling result (bundling creates high density areas) and *de-emphasizes* individual stray fibers. In this regard both single-pass ordered ABP and blended lines complement the reduction of occlusion (requirement S1) much more, whereas ABP and DDH leave room for improvement.

A summary of the above analysis is shown in table 6.1, which highlights that out of all compared methods, the ABP method is the only one that fulfills all rendering requirements, yet requires sorted rendering to fully complement the simplification

(requirement S1).

ALPHA BLENDED POINTSPRITES Out of the various tested splat profiles, the conical splat profile (Eqn. 4.3) is the most useful, as it is most fitted for showing orientation of fiber tracts in comparison to the other profiles (as described in subsection 4.4.2). The profile has one axis of symmetry, which can be aligned specifically which allows for the creation of more extreme, pronounced shading that shows the orientation of the fibers quite clearly. Unlike the other proposed profiles, equation 4.3 contains a singularity at the center (Fig. 4.5b). This singularity becomes mostly disappears in the final image, due to applied blending, or otherwise provides a suitable sprite for trac.

An advantage of the ABP method is that it requires less dense line data to produce solid looking structures, as opposed to e.g. line rendering. It does, on the other hand, require dense uniform arc-length sampling to achieve dense sprite placement.

Using a single primitive to reconstruct both surfaces and tube-like structures (which ABP does, as described in section 4.2) has the advantage that no selection or classification criterion (based on low SNR DTI information) needs to be applied. Mixing geometric primitives complicates the rendering approach since it would require such a selection, e.g. by thresholding c_p, such as applied by Vilanova, Berenschot, and Pul [67].

Another obvious choice are *surfels* or oriented splats[26, 29], which have also been shown to be highly suitable for surface reconstruction. However, such oriented elements can be *looked past*, which makes them less suitable for the depiction of line- or tube-like elements. A supplementary example of this problem is shown in figure A.1, which shows tensor-oriented splats. Besides problematic orientation of line-like elements, this image also shows that inferring splat orientation from (noisy) DTI information can be challenging. Billboarding, as employed by ABP, prevents this problem and allows for a uniform rendering approach.

Due to bundling, any relation to the original DTI information may not be that clear. An advantage of the ABP method is that it does not rely directly on the DTI information to orient rendered elements, and as such can be used in any case to render tracts (applied to bundled- or original fibers, or a mixture of those).

7 CONCLUSION

In this thesis we have addressed our research question that stated: *"How can we depict a large and dense set of 3D paths obtained by tractography in a simplified way, so that key structures are easy to see?"*. Based on six simplification- and rendering requirements (subsections 1.1.2 and 1.1.3), we have proposed several methods that are able to address the RQ.

First, we have demonstrated an approach to the simplification of DTI fiber tracts. Our approach, based on the CUBu method, shows promising computational performance, and allows the generation of a simplified model within a few seconds using standard consumer hardware. We have also shown various ways to control the bundling process and in addition have shown approaches to revealing specific bundles and structures of higher planar- or linear anisotropy.

Secondly, we have proposed a new rendering method for fiber tracts and compared this method to several other pre-existing methods. We found that our pointsprites rendering method (ABP) is particularly suitable for the rendering of complex organic shapes that are intrinsic to DTI visualization.

Using our proposed simplification and rendering methods it becomes possible to convey the main topology of neural pathways within the human brain, and in particular structures of higher planar- or linear anisotropy within the context of the simplified fiber tracts layout.

The OIT aspect of our work remains a difficult topic. As we have shown in the previous chapters, our proposed multi-pass blending is no substitute for blending with sorted rendering, it does however allow for rendering at interactive framerates (without explicit render order) and smooth shading with various illustrative effects. Our pointsprites rendering method emphasizes individual bundles and fibers (particularly when outlines are shown), which is a visual style that may not be applicable in every scenario.

Our comparison of rendering methods also shows that (sorted) alpha blended lines rendering is a method that depicts the $3D$ bundling of fiber tracts quite well, as low-density areas are de-emphasized due to blending with background fragments. We have shown that even though shading options are limited for such lines rendering, it is a viable rendering method for showing bundled fiber tracts.

Alternatively a single-pass version of our alpha blended pointsprites rendering can be used, which improves over line rendering by improved shading and shape reconstruction. The disadvantage of both the single-pass ABP and the alpha blended lines method, is that all geometry is sorted in eye space, and as such these methods are

limited in achieving interactive framerates.

We have tested our work using several datasets from both male and female subjects (listed in appendix A.1) with varying properties such as voxel dimension, resolution and the number of diffusion directions. We found that visualization results are comparable across all tested datasets.

7.1 FUTURE WORK

Running times as demonstrated by our proposed method allow for interactive exploration and simplification of dense fiber tracts. Nevertheless, application of alternate bundling methods such as FFTEB[33] could further improve these running times, and as such should be explored.

Furthermore, additional options for bundling shape control could be explored. For instance, each bundling iteration of our proposed method applies a uniform kernel size, which results in a uniform bundling coarseness level. An approach for varying kernel sizes, in which certain structures would be bundled coarsely whereas others would receive finer bundling, could be an interesting topic of research.

The precise nature of the relation between *simplified* fiber tracts and their *original* counterparts remains a topic for further study. Segmented fiber tracts with anatomical labels could provide more insight in this matter, for instance fiber tracts as obtained from the atlas based method proposed by Guevara et al. [21].

In our research, fairly conventional deterministic fiber tracking was applied, for which many alternatives have been proposed so far[28]. Another direction for future research could therefore be to verify our simplification approach using more elaborate fiber tracking and tensor reconstruction approaches, since such methods could further improve the quality of the input data. An example thereof is the UKF-tractography method proposed by Malcolm, Shenton, and Rathi [35].

Although we have shown the alpha blended pointsprites applied to DTI visualization, the scope of this rendering method is not necessarily restricted to DTI. Application of the method for the rendering of other types of organic shapes, for instance: vascular structures[51] or DW-MRI tractography of the heart[58], could prove to be interesting as well.
More generic surface splatting or point cloud rendering usecases may also be suitable for our rendering method, which could be further explored.

The visual result of the alpha blended pointsprites is to some extent comparable to isosurface hull construction around bundles, a visualization approach that occasionally is preferred over conventional fiber tracking in medical practice[40]. Our rendering method could be explored as an alternative visualization method for such occasions.

Even though our multi-pass blending approach allows for illustrative rendering with outlines at interactive framerates, it does involve *two* geometry passes. In ad-

dition the OIT aspect of the rendering also remains a partly unsolved problem. An approach using per-pixel linked lists[65] could result in a single-pass method with improved background blending capability, which is a direction that was left unexplored.

Part I

APPENDIX

APPENDIX

A.1 DATASETS

All datasets were made freely available to the scientific community, origin URLs are listed in the colophon.

	Dataset #1	**Dataset #2**	**Dataset #3**	**Dataset #4**
Source	3DSlicer [18]	OpenfMRI [49]	HCP [13]	HCP [13]
Subject ID	N/A	Subject 10159	mgh_1001	mgh_1009
Gender and age	N/A	Female, age 30	Female, age 40-44	Male, age 30-34
Accession number	N/A	ds000030	ConnectomeDB S01322	ConnectomeDB S01330
Volume dimensions	$128 \times 128 \times 94$	$96 \times 96 \times 50$	$140 \times 140 \times 96$	$140 \times 140 \times 96$
Voxel size	$1.50^3 mm^3$ (isotropic)	$1.979mm \times 1.979mm \times 2.000mm$	$1.50^3 mm^3$ (isotropic)	$1.50^3 mm^3$ (isotropic)
Number of diffusion directions	42	64	512 (256 used)	512 (256 used)
Number of $b0$-images	1	1	40 (20 used)	40 (20 used)

Table A.1: Overview of used datasets.

A.2 COLORMAPS

A commonly used [64] RGB directional colormap that shows local fiber orientation is defined as:

$$color_{RGB}(x_j) \in \mathbb{R}^3 = \tau(x_j) \tag{A.1}$$

Where $\tau(x_j)$ is the line tangent at x_j (eq. 3.4)

A.3 SUPPLEMENTARY IMAGES

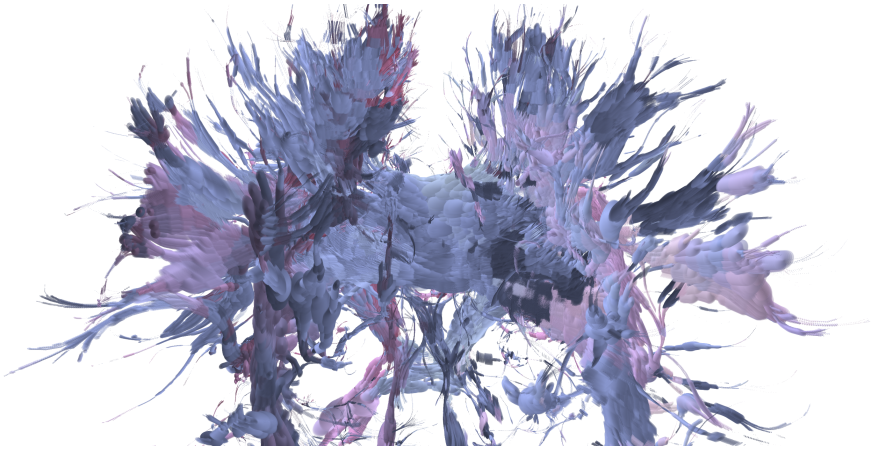


Figure A.1: Example of tensor oriented splats. Each ellipsoid element, scaled by λ_1 and λ_2, is positioned using the minor eigenvector ($\boldsymbol{e_3}$) as surface normal.

BIBLIOGRAPHY

[1] Daniel C. Alexander, Penny L. Hubbard, Matt G. Hall, Elizabeth A. Moore, Maurice Ptito, Geoff J.M. Parker, and Tim B. Dyrby. "Orientationally invariant indices of axon diameter and density from diffusion MRI." In: *NeuroImage* 52.4 (2010), pp. 1374 –1389. DOI: https://doi.org/10.1016/j.neuroimage.2010.05.043. URL: http://www.sciencedirect.com/science/article/pii/S1053811910007755 (cit. on p. 10).

[2] Michael Ankele and Thomas Schultz. "A Sheet Probability Index from Diffusion Tensor Imaging." In: *Computational Diffusion MRI*. Springer, 2017 (cit. on p. 1).

[3] Yaniv Assaf, Tamar Blumenfeld-Katzir, Yossi Yovel, and Peter J. Basser. "Axcaliber: A method for measuring axon diameter distribution from diffusion MRI." In: *Magnetic Resonance in Medicine* 59.6 (), pp. 1347–1354. DOI: 10.1002/mrm.21577. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.21577. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.21577 (cit. on p. 10).

[4] PETER J BASSER and CARLO PIERPAOLI. "Microstructural and Physiological Features of Tissues Elucidated by Quantitative-Diffusion-Tensor MRI." In: *JOURNAL OF MAGNETIC RESONANCE, Series B* 111 (1996), pp. 209–219 (cit. on p. 8).

[5] Louis Bavoil and Kevin Myers. "Order independent transparency with dual depth peeling." In: *NVIDIA OpenGL SDK* (2008), pp. 1–12 (cit. on p. 35).

[6] Alexey Boreskov and Evgeniy Shikin. *Computer graphics: from pixels to programmable graphics hardware*. CRC Press, 2013 (cit. on p. 33).

[7] Joachim Böttger, Ralph Schurade, Estrid Jakobsen, Alexander Schaefer, and Daniel S Margulies. "Connexel visualization: a software implementation of glyphs and edge-bundling for dense connectivity data using brainGL." In: *Frontiers in neuroscience* 8 (2014), p. 15 (cit. on p. 12).

[8] Joachim Bottger, Alexander Schafer, Gabriele Lohmann, Arno Villringer, and Daniel S Margulies. "Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain." In: *IEEE transactions on visualization and computer graphics* 20.3 (2014), pp. 471–480 (cit. on p. 12).

[9] Leonard Brown. *John Coltrane and black America's quest for freedom: Spirituality and the music*. Oxford University Press, 2010 (cit. on p. v).

[10] Wei Chen, Song Zhang, Stephen Correia, and David F Tate. "Visualizing diffusion tensor imaging data with merging ellipsoids." In: *Visualization Symposium, 2009. PacificVis' 09. IEEE Pacific*. IEEE. 2009, pp. 145–151 (cit. on p. 10).

[11] Sebastian Eichelbaum, Mario Hlawitschka, and Gerik Scheuermann. "LineAO—Improved three-dimensional line rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 433–445 (cit. on p. 10).

[12] Ozan Ersoy, Christophe Hurter, Fernando Paulovich, Gabriel Cantareiro, and Alex Telea. "Skeleton-based edge bundling for graph visualization." In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2364–2373 (cit. on p. 12).

[13] David C. Van Essen, Stephen M. Smith, Deanna M. Barch, Timothy E.J. Behrens, Essa Yacoub, and Kamil Ugurbil. "The WU-Minn Human Connectome Project: An overview." In: *NeuroImage* 80 (2013). Mapping the Connectome, pp. 62 –79. DOI: https://doi.org/10.1016/j.neuroimage.2013.05.041. URL: http://www.sciencedirect.com/science/article/pii/S1053811913005351 (cit. on pp. 7, 71).

[14] Cass Everitt. "Interactive order-independent transparency." In: *White paper, nVIDIA* 2.6 (2001), p. 7 (cit. on p. 35).

[15] Maarten H Everts, Henk Bekker, Jos BTM Roerdink, and Tobias Isenberg. "Depth-dependent halos: Illustrative rendering of dense line data." In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009) (cit. on pp. 2, 10, 11, 32, 45, 51, 55, 57).

[16] Maarten H Everts, Eric Begue, Henk Bekker, Jos BTM Roerdink, and Tobias Isenberg. "Exploration of the brain's white matter structure through visual abstraction and multi-scale local fiber tract contraction." In: *IEEE transactions on visualization and computer graphics* 21.7 (2015), pp. 808–821 (cit. on p. 12).

[17] Maarten Hindrik Everts. "Visualization of dense line data." PhD thesis. University of Groningen, 2011 (cit. on pp. 2, 10, 31, 32).

[18] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, et al. "3D Slicer as an image computing platform for the Quantitative Imaging Network." In: *Magnetic resonance imaging* 30.9 (2012), pp. 1323–1341 (cit. on pp. 32, 71).

[19] Pierre Fillard et al. "Quantitative evaluation of 10 tractography algorithms on a realistic diffusion MR phantom." In: *NeuroImage* 56.1 (2011), pp. 220 –234. DOI: https://doi.org/10.1016/j.neuroimage.2011.01.032. URL: http://www.sciencedirect.com/science/article/pii/S105381191100067X (cit. on p. 9).

[20] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. "The minimal preprocessing pipelines for the Human Connectome Project." In: *Neuroimage* 80 (2013), pp. 105–124 (cit. on p. 7).

[21] Pamela Guevara, Delphine Duclap, Cyril Poupon, Linda Marrakchi-Kacem, Pierre Fillard, Denis Le Bihan, Marion Leboyer, Josselin Houenou, and J-F Mangin. "Automatic fiber bundle segmentation in massive tractography datasets using a multi-subject bundle atlas." In: *Neuroimage* 61.4 (2012), pp. 1083–1099 (cit. on p. 66).

[22] Danny Holten. "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data." In: *IEEE Transactions on visualization and computer graphics* 12.5 (2006), pp. 741–748 (cit. on p. 22).

[23] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. "Graph bundling by kernel density estimation." In: *Computer Graphics Forum*. Vol. 31. 3pt1. Wiley Online Library. 2012, pp. 865–874 (cit. on pp. 12, 17, 18, 20).

[24] Christophe Hurter, Alexandru Telea, and Ozan Ersoy. "Moleview: An attribute and structure-based semantic lens for large element-based plots." In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2600–2609 (cit. on pp. 12, 21).

[25] Christophe Hurter, Stéphane Puechmorel, Florence Nicol, and Alexandru Telea. "Functional Decomposition for Bundled Simplification of Trail Sets." In: *IEEE Transactions on Visualization and Computer Graphics* 99 (2017), pp. 1–1 (cit. on p. 12).

[26] Andrei C Jalba, Jacek Kustra, and Alexandru C Telea. "Surface and curve skeletonization of large 3D models on the GPU." In: *IEEE transactions on pattern analysis and machine intelligence* 35.6 (2013), pp. 1495–1508 (cit. on pp. 33, 63).

[27] R. Jianu, C. Demiralp, and D. H. Laidlaw. "Exploring Brain Connectivity with Two-Dimensional Neural Maps." In: *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2012), pp. 978–987. DOI: 10.1109/TVCG.2011.82 (cit. on p. 12).

[28] Derek K. Jones. *Diffusion MRI*. Oxford University Press, 2011 (cit. on pp. 5–9, 66).

[29] Leif Kobbelt and Mario Botsch. "A survey of point-based techniques in computer graphics." In: *Computers & Graphics* 28.6 (2004), pp. 801–814 (cit. on pp. 33, 63).

[30] S Kollias. "Insights into the Connectivity of the Human Brain Using DTI." In: *Nepalese Journal of Radiology* 1.1 (2012), pp. 78–91 (cit. on p. 5).

[31] Eugene Lapidous and Guofang Jiao. "Optimal Depth Buffer for Low-cost Graphics Hardware." In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. HWWS '99. Los Angeles, California, USA: ACM, 1999, pp. 67–73. DOI: 10.1145/311534.311579. URL: http://doi.acm.org/10.1145/311534.311579 (cit. on p. 50).

[32] Denis Le Bihan, Eric Breton, Denis Lallemand, Philippe Grenier, Emmanuel Cabanis, and Maurice Laval-Jeantet. "MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders." In: *Radiology* 161.2 (1986), pp. 401–407 (cit. on p. 5).

[33] Antoine Lhuillier, Christophe Hurter, and Alexandru Telea. "FFTEB: Edge bundling of huge graphs by the Fast Fourier Transform." In: *Pacific Visualization Symposium (PacificVis), 2017 IEEE*. IEEE. 2017, pp. 190–199 (cit. on pp. 12, 66).

[34] Antoine Lhuillier, Christophe Hurter, and Alexandru Telea. "State of the art in edge and trail bundling techniques." In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library. 2017, pp. 619–645 (cit. on pp. 12, 18, 22).

[35] James G. Malcolm, Martha E. Shenton, and Yogesh Rathi. "Neural Tractography Using an Unscented Kalman Filter." In: *Proceedings of the 21st International Conference on Information Processing in Medical Imaging*. IPMI '09. Williamsburg, Virginia: Springer-Verlag, 2009, pp. 126–138. DOI: 10.1007/978-3-642-02498-6_11. URL: http://dx.doi.org/10.1007/978-3-642-02498-6_11 (cit. on pp. 1, 9, 66).

[36] Daniel S. Margulies, Joachim Böttger, Aimi Watanabe, and Krzysztof J. Gorgolewski. "Visualizing the human connectome." In: *NeuroImage* 80.Supplement C (2013). Mapping the Connectome, pp. 445 –461. DOI: https://doi.org/10.1016/j.neuroimage.2013.04.111. URL: http://www.sciencedirect.com/science/article/pii/S1053811913004709 (cit. on p. 32).

[37] Morgan McGuire and Louis Bavoil. "Weighted Blended Order-Independent Transparency." In: *Journal of Computer Graphics Techniques (JCGT)* 2.2 (2013), pp. 122–141. URL: http://jcgt.org/published/0002/02/09/ (cit. on p. 35).

[38] Klaus-Dietmar Merboldt, Wolfgang Hanicke, and Jens Frahm. "Self-diffusion NMR imaging using stimulated echoes." In: *Journal of Magnetic Resonance (1969)* 64.3 (1985), pp. 479–486 (cit. on p. 5).

[39] D. Merhof, M. Sonntag, F. Enders, C. Nimsky, P. Hastreiter, and G. Greiner. "Hybrid Visualization for White Matter Tracts using Triangle Strips and Point Sprites." In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 1181–1188. DOI: 10.1109/TVCG.2006.151 (cit. on pp. 10, 11, 32).

[40] Dorit Merhof, Martin Meister, Ezgi Bingöl, Christopher Nimsky, and Günther Greiner. "Isosurface-based generation of hulls encompassing neuronal pathways." In: *Stereotactic and functional neurosurgery* 87.1 (2009), pp. 50–60 (cit. on pp. 11, 66).

[41] Mikhail Milchenko and Daniel Marcus. "Obscuring surface anatomy in volumetric imaging data." In: *Neuroinformatics* 11.1 (2013), pp. 65–75 (cit. on p. 7).

[42] Ludovico Minati and Władysław P Węglarz. "Physical foundations, models, and methods of diffusion magnetic resonance imaging of the brain: A review." In: *Concepts in Magnetic Resonance Part A* 30.5 (2007), pp. 278–307 (cit. on p. 5).

[43] Susumu Mori, Barbara J Crain, VP Chacko, and Peter Van Zijl. "Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging." In: *Annals of neurology* 45.2 (1999), pp. 265–269 (cit. on pp. 9, 10, 46).

[44] Ron Otten, Anna Vilanova, and Huub Van De Wetering. "Illustrative white matter fiber bundles." In: *Computer Graphics Forum*. Vol. 29. 3. Wiley Online Library. 2010, pp. 1013–1022 (cit. on pp. 11, 51, 54).

[45] THJM Peeters, A Vilanova, and R ter Haar Romeny. "Visualization of DTI fibers using hair-rendering techniques." In: *Proc ASCI*. 2006, pp. 66–73 (cit. on p. 10).

[46] Vid Petrovic, James Fallon, and Falko Kuester. "Visualizing whole-brain DTI tractography with GPU-based tuboids and LoD management." In: *IEEE transactions on visualization and computer graphics* 13.6 (2007), pp. 1488–1495 (cit. on pp. 10, 11).

[47] Vsevolod Peysakhovich, Christophe Hurter, and Alexandru Telea. "Attribute-driven edge bundling for general graphs with applications in trail analysis." In: *Visualization Symposium (PacificVis), 2015 IEEE Pacific*. IEEE. 2015, pp. 39–46 (cit. on p. 12).

[48] Daniel L Polders, Alexander Leemans, Jeroen Hendrikse, Manus J Donahue, Peter R Luijten, and Johannes M Hoogduin. "Signal to noise ratio and uncertainty in diffusion tensor imaging at 1.5, 3.0, and 7.0 Tesla." In: *Journal of Magnetic Resonance Imaging* 33.6 (2011), pp. 1456–1463 (cit. on p. 7).

[49] Russell Poldrack, Deanna Barch, Jason Mitchell, Tor Wager, Anthony Wagner, Joseph Devlin, Chad Cumba, Oluwasanmi Koyejo, and Michael Milham. "Toward open sharing of task-based fMRI data: the OpenfMRI project." In: *Frontiers in Neuroinformatics* 7 (2013), p. 12. DOI: 10.3389/fninf.2013.00012. URL: https://www.frontiersin.org/article/10.3389/fninf.2013.00012 (cit. on p. 71).

[50] Thomas Porter and Tom Duff. "Compositing Digital Images." In: *SIGGRAPH Comput. Graph.* 18.3 (Jan. 1984), pp. 253–259. DOI: 10.1145/964965.808606. URL: http://doi.acm.org/10.1145/964965.808606 (cit. on p. 35).

[51] Felix Ritter, Christian Hansen, Volker Dicken, Olaf Konrad, Bernhard Preim, and Heinz-Otto Peitgen. "Real-time illustration of vascular structures." In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006) (cit. on p. 66).

[52] Timo Ropinski, Steffen Oeltze, and Bernhard Preim. "Survey of glyph-based visualization techniques for spatial multivariate medical data." In: *Computers & Graphics* 35.2 (2011), pp. 392–401 (cit. on p. 33).

[53] Thomas Schultz and Anna Vilanova. "Diffusion MRI visualization." In: *NMR in Biomedicine* (Feb. 2018). DOI: 10.1002/nbm.3902. URL: http:https://doi.org/10.1002/nbm.3902 (cit. on pp. 1, 8, 10, 11).

[54] Mehmet Sezgin and Bülent Sankur. "Survey over image thresholding techniques and quantitative performance evaluation." In: *Journal of Electronic imaging* 13.1 (2004), pp. 146–166 (cit. on p. 16).

[55] Dave Shreiner, Graham Sellers, John M. Kessenich, and Bill M. Licea-Kane. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. 8th. Addison-Wesley Professional, 2013 (cit. on p. 44).

[56] Jose Soares, Paulo Marques, Victor Alves, and Nuno Sousa. "A hitchhiker's guide to diffusion tensor imaging." In: *Frontiers in Neuroscience* 7 (2013), p. 31. DOI: 10.3389/fnins.2013.00031. URL: https://www.frontiersin.org/article/10.3389/fnins.2013.00031 (cit. on pp. 5, 8).

[57] Wei-jie Song, Wei Chen, Hai-dong Chen, Hong-chan Zheng, and Jun-zhi Cui. "Visualizing diffusion tensor fields on streamsurfaces with merging ellipsoids and LIC texture." In: *Applied Mathematics-A Journal of Chinese Universities* 29.4 (2014), pp. 399–409. DOI: 10.1007/s11766-014-3234-y. URL: https://doi.org/10.1007/s11766-014-3234-y (cit. on pp. 10, 33).

[58] David E Sosnovik, Ruopeng Wang, Guangping Dai, Timothy G Reese, and Van J Wedeen. "Diffusion MR tractography of the heart." In: *Journal of Cardiovascular Magnetic Resonance* 11.1 (2009), p. 47 (cit. on p. 66).

[59] H. Speckter, J. Bido, G. Hernandez, L. Suazo, D. Rivera, S. Valenzuela, and P. Stoeter. "Fractional Anisotropy Predicts Responsiveness of Meningiomas to Stereotactic Radiosurgery (SRS)." In: *International Journal of Radiation Oncology*Biology*Physics* 96.2, Supplement (2016). Proceedings of the American Society for Radiation Oncology, E101. DOI: https://doi.org/10.1016/j.ijrobp.2016.06.845. URL: http://www.sciencedirect.com/science/article/pii/S0360301616311713 (cit. on p. 2).

[60] C. Stoll, S. Gumhold, and H. P. Seidel. "Visualization with stylized line primitives." In: *VIS 05. IEEE Visualization, 2005.* 2005, pp. 695–702. DOI: 10.1109/VISUAL.2005.1532859 (cit. on pp. 10, 11, 32, 37).

[61] Pjotr Svetachov, Maarten H. Everts, and Tobias Isenberg. "DTI in Context: Illustrating Brain Fiber Tracts In Situ." In: *COMPUTER GRAPHICS FORUM* 29.3 (2010), pp. 1023–1032. DOI: 10.1111/j.1467-8659.2009.01692.x (cit. on p. 32).

[62] Chantal MW Tax, Tom Dela Haije, Andrea Fuster, Carl-Fredrik Westin, Max A Viergever, Luc Florack, and Alexander Leemans. "Sheet Probability Index (SPI): Characterizing the geometrical organization of the white matter with diffusion MRI." In: *NeuroImage* 142 (2016), pp. 260–279 (cit. on p. 1).

[63] Stefan J Teipel, Michel J Grothe, Massimo Filippi, Andreas Fellgiebel, Martin Dyrba, Giovanni B Frisoni, Thomas Meindl, Arun LW Bokde, Harald Hampel, Stefan Klöppel, et al. "Fractional anisotropy changes in Alzheimer's disease depend on the underlying fiber tract architecture: a multiparametric DTI study using joint independent component analysis." In: *Journal of Alzheimer's Disease* 41.1 (2014), pp. 69–83 (cit. on p. 2).

[64] Alexandru C Telea. *Data visualization: principles and practice*. CRC Press, 2014 (cit. on pp. 2, 6, 10, 12, 32, 33, 71).

[65] Nicolas Thibieroz. "Order-independent transparency using per-pixel linked lists." In: *GPU Pro* 2 (2011), pp. 409–431 (cit. on p. 67).

[66] Jelle Veraart, Jan Sijbers, Stefan Sunaert, Alexander Leemans, and Ben Jeurissen. "Weighted linear least squares estimation of diffusion MRI parameters: strengths, limitations, and pitfalls." In: *NeuroImage* 81 (2013), pp. 335–346 (cit. on p. 6).

[67] A. Vilanova, G. Berenschot, and C. van Pul. "DTI Visualization with Streamsurfaces and Evenly-spaced Volume Seeding." In: *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*. VISSYM'04. Konstanz, Germany: Eurographics Association, 2004, pp. 173–182. DOI: 10.2312/VisSym/VisSym04/173-182. URL: http://dx.doi.org/10.2312/VisSym/VisSym04/173-182 (cit. on pp. 1, 9–11, 28, 63).

[68] A. Vilanova, S. Zhang, G. Kindlmann, and D. Laidlaw. "An Introduction to Visualization of Diffusion Tensor Imaging and Its Applications." In: *Visualization and Processing of Tensor Fields*. Ed. by Joachim Weickert and Hans Hagen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 121–153. DOI: 10.1007/3-540-31272-2_7. URL: https://doi.org/10.1007/3-540-31272-2_7 (cit. on p. 10).

[69] R Wang, T Benner, AG Sorensen, and VJ Wedeen. "Diffusion toolkit: a software package for diffusion imaging data processing and tractography." In: *Proc Intl Soc Mag Reson Med*. Vol. 15. 3720. 2007 (cit. on pp. 32, 46).

[70] Van J Wedeen, Douglas L Rosene, Ruopeng Wang, Guangping Dai, Farzad Mortazavi, Patric Hagmann, Jon H Kaas, and Wen-Yih I Tseng. "The geometric structure of the brain fiber pathways." In: *Science* 335.6076 (2012), pp. 1628–1634 (cit. on p. 2).

[71] C.-F. Westin, S. Peled, H. Gudbjartsson, R.Kikinis, and F. A. Jolesz. "Geometrical Diffusion Measures for MRI from Tensor Basis Analysis." In: *ISMRM '97*. Vancouver Canada, 1997, p. 1742 (cit. on p. 7).

[72] Todd Whitaker and Dale Lumpa. *Great quotes for great educators*. Routledge, 2014 (cit. on p. ix).

[73] Hongfeng Yu, Chaoli Wang, Ching-Kuang Shene, and Jacqueline H. Chen. "Hierarchical Streamline Bundles." In: *IEEE Transactions on Visualization and Computer Graphics* 18.8 (Aug. 2012), pp. 1353–1367. DOI: 10.1109/TVCG.2011.155. URL: http://dx.doi.org/10.1109/TVCG.2011.155 (cit. on p. 12).

[74] Leonid Zhukov and Alan H Barr. "Oriented tensor reconstruction: Tracing neural pathways from diffusion tensor MRI." In: *Visualization, 2002. VIS 2002. IEEE*. IEEE. 2002, pp. 387–394 (cit. on pp. 7, 10).

[75] Matthew van der Zwan, Valeriu Codreanu, and Alexandru Telea. "CUBu: Universal real-time bundling for large graphs." In: (2016) (cit. on pp. 12, 15, 18–20, 22–25, 30).

COLOPHON

All datasets used in this thesis were obtained from:

- the Human Connectome Project https://db.humanconnectome.org
- the OpenfMRI project https://openfmri.org
- 3*DSlicer* https://www.slicer.org

The complementary implementation for this research was built using OpenGL, NVIDIA CUDA, the Nifti-1 C libraries and Qt.

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede, and in particular the LyX template by Nick Mariette and Ivo Pletikosi'c, `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Final Version as of June 29, 2018 (`classicthesis` version 0.6 [FINAL DRAFT]).