Nerve Fiber Modeling and 3D-PLI Simulations of a Tilting Polarization Microscope

Felix Matuschke

July 2022

Nerve Fiber Modeling and 3D-PLI Simulations of a Tilting Polarization Microscope

Inaugural-Dissertation

zur Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Felix Matuschke
aus Meschede

Düsseldorf, Juli 2022

Heinrich Heine
Universität
Düsseldorf

JÜLICH
Forschungszentrum

Heinrich-Heine-Universität Düsseldorf
Math.-Nat. Fakultät
Universitätsstraße 1
40225 Düsseldorf

Forschungszentrum Jülich
Institut für Neurowissenschaften und Medizin
Strukturelle und funktionelle Organisation des Gehirns (INM-1)
Faserbahnarchitektur
Wilhelm-Johnen-Straße
52428 Jülich

Gedruckt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

1. Berichterstatter	Prof. Dr. Gunnar Schröder Forschungszentrum Jülich IBI-7
2. Berichterstatter	Prof. Dr. Katrin Amunts Forschungszentrum Jülich INM-1

Tag der mündlichen Prüfung:	06.02.2023

Abstract

In the Fiber Architecture group of the Institute of Neuroscience and Medicine, Structural and Functional Organization of the Brain (INM-1), 3D Polarized Light Imaging (3D-PLI) microscopy is used to measure the orientation of nerve fibers in unstained brain sections. Interpretation of the measurement can be challenging for certain regions, for example where fibers cross or are oriented perpendicular to the sectioning plane. To understand the behavior of the measured signal of such structures without further external influences, such as non-ideal optics, simulations are used where each parameter is known. In order to perform simulations, virtual tissue models are needed and a virtual 3D-PLI microscope, being capable of simulating the influence of the tissue on the light.

In order to design realistic models of dense nerve fiber tissue, it must be ensured that individual nerve fibers do not overlap. This is especially difficult to design in advance for interwoven structures, as is occurs in nerve fiber crossings. Therefore, a nerve fiber modeling specialized algorithm was designed in this thesis. The algorithm will check a given volume for overlaps of single nerve fibers, and then slowly push them apart at the affected locations. Thus, a collision-free tissue model is created over time. The pre-existing simulation algorithm of the 3D PLI microscope was completely redesigned as part of this work. The algorithm is now able to run in parallel on multiple CPU cores as well as computational clusters. Thus, a large number of simulations can be performed, allowing for greater statistics in the analyses. These two algorithms were published in the software package *fiber architecture simulation toolbox of 3D-PLI* (fastPLI).

Finally, in this thesis, nerve fiber models consisting of two nerve fiber populations, i. e. two densely packed crossing nerve fiber bundles, were created and subsequently simulated. The results show, that the orientation of the nerve fiber population, which has a higher proportion in the volume, can be determined. With the current resolution of the microscopes used, it is not possible to determine both fiber population orientations individual. The measured orientation seems to follow the circular mean as a function on the proportional volume fraction of the nerve fiber populations, taking into account the decrease of the measured signal due to the increasing tilt angle. In summary, the development of the algorithm for modeling nerve fibers together with the simulation in a toolbox has proven to be a suitable tool to be able to investigate questions quickly through simulations.

Acknowledgement

First and foremost, I would like to thank my supervisor, Prof. Dr. Markus Axer. Without his support and dedicated involvement every step of the way throughout the process, this thesis would never have come to fruition. He has given me wonderful support throughout all these years. I could always count on him no matter what the issues were. Furthermore, I would like to thank my examiners, Prof. Dr. Gunnar Schröder and Prof. Dr. Katrin Amunts. I have always enjoyed our insightful discussions and am glad to have been guided through this work by you.
I would also like to thank Prof. Katrin Amunts in her role as Institute Director of the INM-1 and Scientific Director of the Human Brain Project. I am glad that she gave me the opportunity to participate in both the institute and the HBP.
My special thanks go to my parents and my brother. I would especially like to thank my father for giving me and my brother the curiosity for nature and thus for science. My parents have always supported my brother and me in everything, and we could always rely on them. I would like to thank my brother for always helping me from a very young age, even though it was not always easy for him. Without him, I probably would never have dared to tackle a science degree. As a family, we have had to make very difficult decisions in recent years. These decisions are anything but easy to make, but I am very grateful that we were able to make them together as a family and that we support each other.
I would like to thank all my colleagues at INM-1 at the Forschungszentrum Jülich. I felt very welcome from the beginning, and I still enjoy working with each one of you. My particular thanks go to my dear colleague Miriam Menzel. Without her scientific discussions, I would not have come as far as I have. Unfortunately, she is leaving our working group, but she can continue to count on me, as I have always been able to count on her. My further thanks go to Andrea Brandstetter. She is a true friend and always there when I needed someone to talk to.
Last but not least, I want to thank all of my friends. I appreciate the various board game sessions and the hikes together. Being with you guys is always a delight and a welcome diversion from my work when I need to unwind.

Funding

The project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 720270 (HBP SGA1) and Grant Agreement No. 785907 (HBP SGA2).

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

The graphics in this thesis were generated by the Tikz and PGF Packages developed by Till Tantau and the Pgfplots package developed by Christian Feuersänger.

Contents

Inside you is the potential to make yourself better … and that is what it is to be human. To make yourself more than you are.

— **Jean-Luc Picard**
Captain USS Enterprise NCC-1701-E

1 Introduction

One of the biggest unsolved questions in science today is how the brain, especially the human brain, functions. An important role plays the connectivity of neurons among each other. They may be connected via nerve fibers over a relatively short distance to neighboring neurons or to more distant regions of the brain. Understanding the brain's connectome, the interconnection of brain cells, will allow us to explore and harness the origins of cognitive abilities such as object recognition and memory. Since the brain is such a large and complex structure relative to the size of its individual building blocks, neuroscientists are joining forces in collaborative projects such as the Human Brain Project to solve this problem together [Mar06; She+12; Amu+13; Amu+16].

Techniques in machine learning [Mur13; Goo+16], especially deep learning, profit from a better understanding of the brain. Different types of neural networks are able to solve difficult problems that are almost impossible to solve with a classical algorithm. A better understanding of the neural network as a subset of the connectome can therefore help to design artificial neural networks to efficiently solve tasks like image recognition, and vice versa.

The only technique currently able to measure the human connectome in vivo (i. e., the living brain) is by using diffusion Magnetic Resonance Imaging (dMRI). dMRI provides a rather low resolution of about one millimeter relative to the structural configuration of nerve fiber bundles of several micrometers in vivo. This imaging resolution at the mm-scale can lead to misleading results due to the models and resolution limits involved, in particular in regions of fiber crossings or fiber "kissing" [Mai+17; Sch+21]. Therefore, a higher resolution of fiber pathways and the comprising individual fibers is required. Post mortem, the resolution of dMRI reaches up to the order of 100 µm [Bea+19]. Microscopic imaging techniques on the other side can reach up to the order of micrometer to hundreds of nanometer. The challenge at such low scales is not only to measure a portion of the brain (not to mention an entire human brain), but also to be able to process this massive amount of data. Such datasets will help to learn from and improve the models applied to the lower resolution datasets, which will help with e. g. diagnostics [Yen+21].

3D-Polarized Light Imaging (3D-PLI) is a microscopic imaging technique that allows the measurement of the orientation of nerve fibers inside a brain section [Axe+11a; Axe+11b; Axe+16]. A nerve fiber is the extension of a nerve cell. The nerve fiber consists of an axon, which transmits the electrical signal from the nerve cell. Depending on the nerve fiber, the axon may be surrounded by a myelin sheath, which primarily serves to transport the signal very rapidly. The myelin sheaths generate an optical property called birefringence. Birefringence is the presence of a refractive index that depends on the polarization and propagation direction of light. By analyzing this behavior, the underlying optic axis and thus the orientation of the nerve fiber can be determined. The brain sections are 60 µm thick and the image resolution is in the order of a few micrometers, i. e., at axonal scales. However, due to the fact that the brain needs to be sectioned, the tissue becomes quite fragile which leads to movements of the tissue and can also lead to fractions. These movements have to be later reversed in a process called image registration, allowing to build up an entire 3D dataset from the individual sections. Another challenge comes in steep nerve fiber regions and nerve fiber crossings. Because the signal from an image pixel originates from a volume that contains multiple nerve fiber orientations, the interpretation can be challenging. Steep fibers are e. g. orientated along the propagation direction of the light. Therefore, there is no change for the polarization state of the light. Crossing nerve fibers on the other hand are a mixture of multiple light waves. Since however, only the sum of all light waves reaching the same sensor pixel is measured, the individual information is lost.

To improve the understanding of the underlying architecture including the nerve fiber orientations and its effect on the measured polarization signal, simulations play therefore an important role. Due to the involved spatial sizes and the multitude of possible orientations of nerve fibers and further cells, no phantom currently exists, allowing to investigate all necessary possibilities. Complementary imaging techniques, such as two-photon microscopy, allow the tissue to be examined at higher resolution and lower deformations, but this has the disadvantage of longer measurement time [Cos+20; Cos+21].

For this reason, simulations are used. Simulations make studies of physical effects possible which cannot be easily addressed by experimental setups [Cal+19; Men+20]. Clear benefits of simulations are the repeatability of experiments and the possibility to generate relevant statistics (i. e., large data sets) to meet e. g. the requirements of machine learning [Gin+18; Gin19]. Therefore, the simulation and generation of such datasets should be done as fast as possible to obtain a large training dataset.

This dissertation provides a novel open-source software package, the *fiber architecture simulation toolbox for 3D-PLI* (*fastPLI*), whose main purpose is to provide a method for modeling dense, non-colliding 3D nerve fiber models [Mat+19; Mat+21; Reu+19]. These models are then used to simulate them in a virtual 3D-PLI experiment by calculating the effect of birefringence on polarized light using the Müller-Stokes calculus.

An important issue is the use of supercomputer architectures with efficiently developed algorithms to enable simulations of larger models and volumes. The usefulness will be investigated and limitations will be demonstrated.

Part I

Basics

2 Neuroanatomy

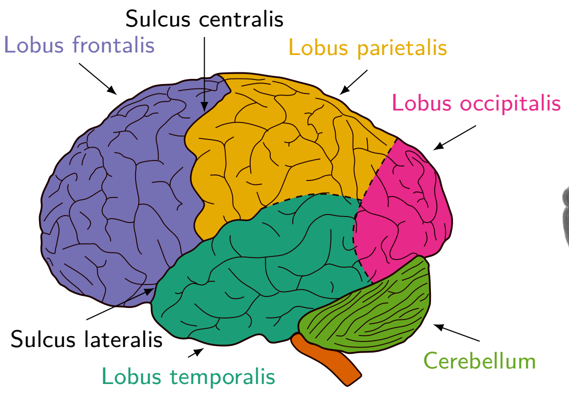
2.1 Introduction

Neuroanatomy is the study of the structure of the brain. It describes the regions and structure of the nervous system in humans and animals. Techniques such as diffusion Magnetic Resonance Imaging (dMRI), fluorescence microscopy, microscopy on stained tissue, autoradiography (to name a few), have been able to study more and more structures from different perspectives in the brain with different resolutions, modalities and contrasts on different species.

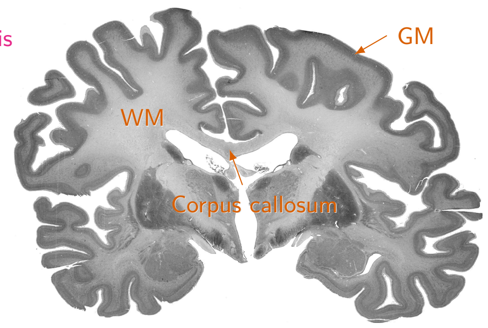
This chapter gives a general overview of the structure of the brain with its most important regions as well as the nerve fiber architecture.

2.2 Brain Architecture

The mammal brain consists of three main parts: the brainstem, the cerebellum and the cerebrum (see fig. 2.1). The brainstem performs various tasks such as controlling the cardiovascular system. It also serves as a connection between the various brain areas and the spinal cord in the lower part of the brain. It can be further subdivided into the midbrain, the pons and the medulla oblongata. The cerebellum is located at the lower rear of the brain. Its most important function is motor control. It is highly folded and therefore has a particularly large surface area. The cerebrum is the largest part of the human brain. As the cerebellum its surface is folded as well. The cerebrum is split into a left and right hemisphere. In addition, the cerebrum can be divided into the following parts: the frontal, parietal, temporal, occipital and insular lobes (see fig. 2.1a). The frontal lobe is responsible for voluntary movements of specific body parts as well as the human personality. The parietal lobe's main functionalities are the processing of the sensory information. The primary function of the occipital lobe is signal processing of the visual system. The temporal lobe contains auditory functions and language perception in addition to visual memories. Beneath the brain surface there are other structures such as the basal ganglia or the thalamus.



(a) Sagittal view of the human brain with its lobes: frontal, parietal, temporal and occipital lobe. The cerebellum and the brainstem are located at the bottom of the brain. [1]



(b) Coronal section stained for cell bodies. The gray matter (GM) is dark while the white matter (WM) is bright. The left and right hemisphere is connected via the corpus callosum. [2]

Fig. 2.1.: Illustration of the human brain and a cell body stained coronal section.

The cerebellum and cerebrum contain a gray matter (GM) structure at the brain surface. This structure is filled with neurons. These cells have the task of processing the information of all signals coming from inside and outside the brain. Such cells are arranged in cortical layers that have different thicknesses, cell types, and densities specific to a brain area. These cells have a relatively high density and are not only locally interconnected with each other, but also connect with different brain areas. Therefore, the folding of the brain is particularly important to increase the surface and therefore the number of cells. In the human brain, there are several billions of nerve cells. There are many types of cells, e. g. granule or pyramidal cells. The highly interconnected structure and arrangement of the various cells is the source of its high number of different functionalities. It is important to investigate the human brain to gain a better understanding of the brain's function and an improved understanding of pathophysiological processes that may lead to improved medical treatment of brain diseases.

2.3 Nerve Fiber Architecture

A typical nerve cell (see fig. 2.2) comprises a cell body, called a soma, that processes incoming information. The information arrives via dendrites, which are star-shaped branches. The axon, or nerve fiber, is the cell's information output. It travels through the brain often associated with nerve fiber bundles to its destination, where it connects with the axon terminal to other neurons.

[1] Drawing of the side view showing the B06 brain collection of the INM-1.
[2] Coronal stained section 4050 from the BigBrain Project [Amu+13].

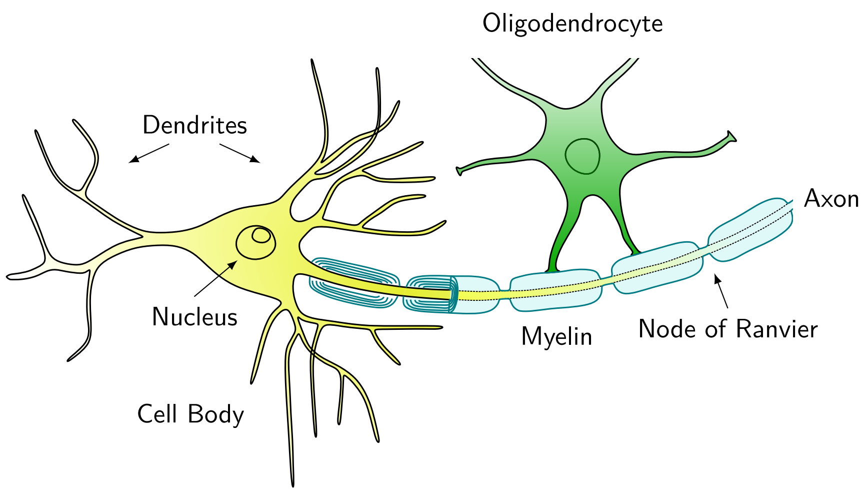
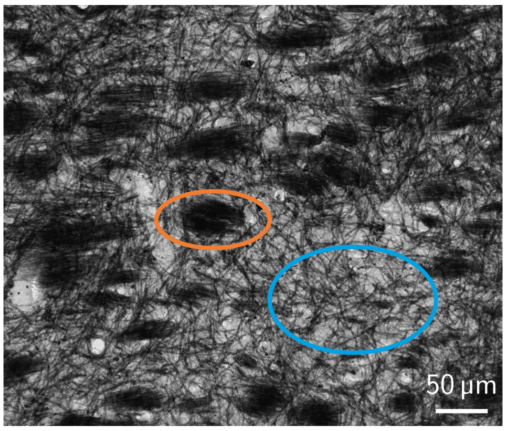


Fig. 2.2.: Illustration of a neuron with axon and oligodendrocytes. The oligodendrocyte build up a layered lipid structure, surrounding the axon. The myelin layers are separated along the axon by nodes of Ranvier.

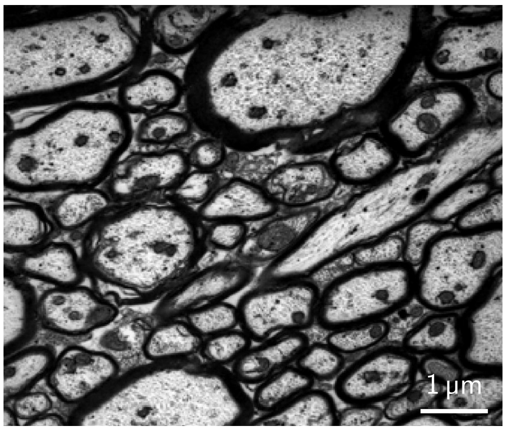
The axon is surrounded by a myelin sheath, a lipid layered structure generated by nearby oligodendrides (see fig. 2.2). The myelin electrically insulates the axon and improves the speed of propagation of the electrical action potential along the axon and also its signal strength. The diameter of the myelin ranges from about 0.5 µm to several micrometers (see table 2.1). There are many types of axons. Some contain a very thick myelin layer, while others have none. The high density of axons and myelin makes the brain appear white and is therefore called white matter (WM), whereas the outer cell bodies appear darker and is called gray matter (GM). This color difference is clearly visible in a Nissl stained histological sections (see fig. 2.1b).

To enhance the contrast of the nerve fibers with respect to the background, staining like Nissl is used to darken the myelin (see fig. 2.3a). This allows to follow small nerve fibers down to individual nerves depending on their myelination degree. Larger nerve fiber bundles are so dark that mostly no orientation can be extracted.

Figure 2.3a shows a Nissl stained brain section. Between neurons, individual nerve fibers form complex reticular structures. Several nerve fiber bundles traverse the thalamus and can be seen as dark structures. A closer look with an electron microscope into a nerve fiber bundle of the corpus callosum of a rodent is shown in fig. 2.3b. The nerve fibers of the 100 nm thin section are densely packed together. Axon diameter and myelin thickness vary from nerve fiber to nerve fiber.



(a) Myelin staining of the human thalamus, sagittal section. Nerve fiber bundle structures are visible as elliptical dark shape. In between net-like structures are formed from individual nerve fibers. `http://brainmaps.org/HBP3/h.sapiens/sag/h5thal-myelin/17a`



(b) Electron microscope image of a 100 nm thin section of rodent corpus callosum [Wal+14]. The myelin is visible as dark surroundings around the inner axon.

2.4 Axon Dimensions

Table 2.1 lists diameters of the nerve fibers in the human brain. The diameter and their standard deviation are region-specific. The g_{ratio} describes the fraction of the axon to the entire nerve fiber diameter. From studies with dMRI and electron microscopy the g_{ratio} is in the range of 0.6 to 0.9, depending on the region (see table 2.2).

area	mean	std
sup. longitudinal fasc.	0.8 µm	0.2 µm
inferior occipital fasc.	0.51 µm	0.05 µm
corpus callosum	0.69 µm	0.04 µm

Tab. 2.1.: Nerve fiber diameter distribution of the human brain. Mean values over three human brains [Lie+14].

study	reference	g_{ratio}
Stikov et al., 2015	[Sti+15]	0.7
Dean et al., 2016	[Dea+16]	0.71-0.9
Mohammadi et al., 2015	[Moh+15]	0.55-0.75
Cercignani et al., 2017	[Cer+17]	0.65-0.8
Berman et al., 2017	[Ber+18]	0.69
Jung et al., 2018	[Jun+18]	0.7-0.8

Tab. 2.2.: human g_{ratio} from in vivo mri studies.

3 Modeling of Light

3.1 Introduction

The following chapter lists the physical theory needed to describe the mathematics behind 3D-Polarized Light Imaging (3D-PLI). These include the basic properties of light, such as its polarization state, the optical properties of nerve fibers tissue, the mathematical description of the 3D-PLI experimental setup and its signal analysis. The chapter drew inspiration from the work presented in [Dem06; Fli12].

3.2 Electromagnetic Waves

Light is an electromagnetic wave. The theory of electromagnetism was first fully described by James Clerk Maxwell, who formulated Maxwell's equations (see eqs. 3.1), generalized from the previous work of Johann Carl Friedrich Gauß, Michael Faraday and André-Marie Ampère and others:

$$\begin{aligned}
\nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} \\
\nabla \cdot \mathbf{B} &= 0 \\
\nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
\nabla \times \mathbf{B} &= \mu_0 \left(\mathbf{j} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right)
\end{aligned} \tag{3.1}$$

where the nabla operator $\nabla := \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$ denotes the three-dimensional gradient operator, $\mathbf{E}$ is the electric field, ρ the electric charge density, ε_0 the permittivity of free space, $\mathbf{B}$ the magnetic field, μ_0 the permeability of free space and $\mathbf{j}$ the electric current density. The first equation states that no electric field can be generated without an electric charge (conservation of charge). The second equation states that there are no magnetic monopoles and the basic unit of a magnetic field is a dipole. The third and fourth equations show the relationship between the electric and magnetic fields in space and time. A change in the electric field results in a magnetic field and vice versa. Equation four also shows a magnetic field's generation from an electric current $\mathbf{j}$. Maxwell's equations satisfy the continuity equation $\operatorname{div} j + \frac{\partial \rho}{\partial t} = 0$, which means that neither an electric field nor a magnetic field can be generated without an electric current or a change in electric potential occurring.

3.2.1 Light in vacuum

In vacuum, eqs. 3.1 simplify with $\rho = 0$ and $\mathbf{j} = \mathbf{0}$ to:

$$\begin{aligned} \nabla \cdot \mathbf{E} &= 0 \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{B} &= \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \end{aligned} \tag{3.2}$$

with

$$\begin{aligned} \nabla \times \nabla \times \mathbf{E} = -\nabla \times \frac{\partial \mathbf{B}}{\partial t} &= -\frac{\partial}{\partial t}(\nabla \times \mathbf{B}) \\ &= -\varepsilon_0 \cdot \mu_0 \frac{\partial^2 \mathbf{E}}{\partial t^2}, \end{aligned} \tag{3.3}$$

the identity $\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2 \mathbf{A}$, $\mu_0 \varepsilon_0 = \frac{1}{c^2}$ and c as the speed of light[1], further simplifies to:

$$\begin{aligned} \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} - \nabla^2 \mathbf{E} = \mathbf{0} \\ \frac{1}{c^2}\frac{\partial^2 \mathbf{B}}{\partial t^2} - \nabla^2 \mathbf{B} = \mathbf{0} \end{aligned} \tag{3.4}$$

which shows that $c^2 \frac{\partial B}{\partial z} = \frac{\partial E}{\partial t} \Rightarrow \mathbf{E} \cdot \mathbf{B} = 0$ and thus the electric and magnetic field components are perpendicular to each other. Furthermore, this implies that space and time are linked, and that light propagates in vacuum with the speed of light c.

[1]can be derived from Maxwell's equations and Lorentz force in a vacuum.

3.2.2 Solving Maxwell's equations in vacuum

Eqs. 3.4 have the form of a wave equation and therefore, it can be solved by

$$\begin{aligned} \mathbf{E}(\mathbf{r},t) &= g(\phi(\mathbf{r},t)) = g(\omega t - \mathbf{k}\cdot\mathbf{r} + \varphi) \\ \mathbf{B}(\mathbf{r},t) &= g(\phi(\mathbf{r},t)) = g(\omega t - \mathbf{k}\cdot\mathbf{r} + \varphi) \end{aligned} \tag{3.5}$$

where g is any well-behaved function (continuous and differentiable) and therefore also its superposition, ω the circular frequency, t the time, $\mathbf{k}$ the wave vector, $\mathbf{r}$ the spacial position and φ the phase. With the help of

$$k = |\mathbf{k}| = \frac{\omega}{c} = \frac{2\pi}{\lambda} \tag{3.6}$$

a planar wave can be described by

$$\begin{aligned} \mathbf{E}(\mathbf{r}) &= \mathbf{E}_0 \cdot e^{-i\,\mathbf{k}\cdot\mathbf{r}} \\ \mathbf{B}(\mathbf{r}) &= \mathbf{B}_0 \cdot e^{-i\,\mathbf{k}\cdot\mathbf{r}} \end{aligned} \tag{3.7}$$

with $\mathbf{k}$ as wave vector pointing in the direction of the propagation of the light wave (see fig. 3.1).

3.2.3 Polarization

Since the light wave propagates in one direction and the electric field and magnetic field are perpendicular to $\mathbf{k}$ and to each other, the orientation of the plane of oscillation is a fundamental property of light called polarization (see fig. 3.1). Without additional information, the polarization orientation is conventionally in the direction of the electric field component. A superposition of x- and y-wave with z-axis equal to the propagation direction gives the general form:

$$\mathbf{E}(z,t) = \begin{pmatrix} E_{0x} \cdot e^{-i\phi_x} \\ E_{0y} \cdot e^{-i\phi_y} \\ 0 \end{pmatrix} e^{-i(kz-\omega t)} \tag{3.8}$$

Figure 3.2 shows an additional representation of the polarization state of a light wave. It shows the component perpendicular to the propagation direction. Thus, the time evolution of the electric field can be represented in the xy-plane. In addition, the states can be described by the Jones or Stokes calculus, which is described in sections 3.2.7 and 3.2.8.

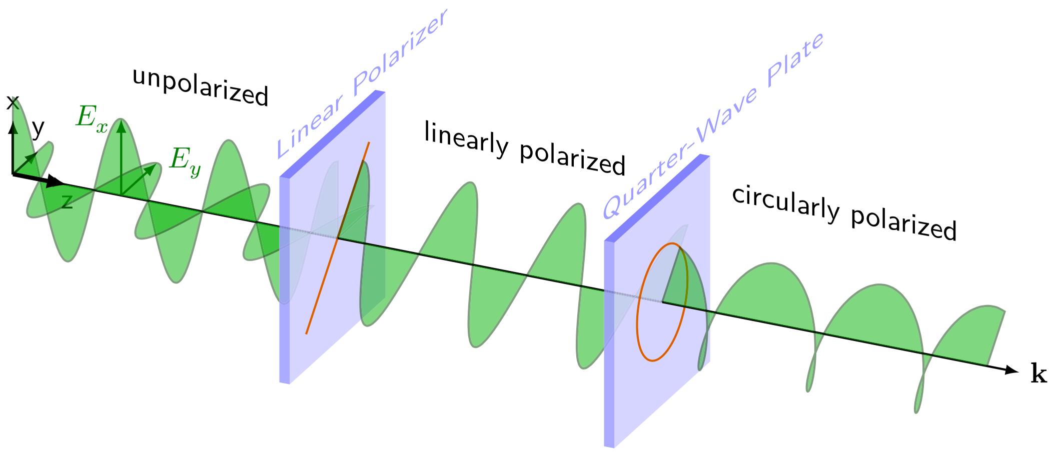
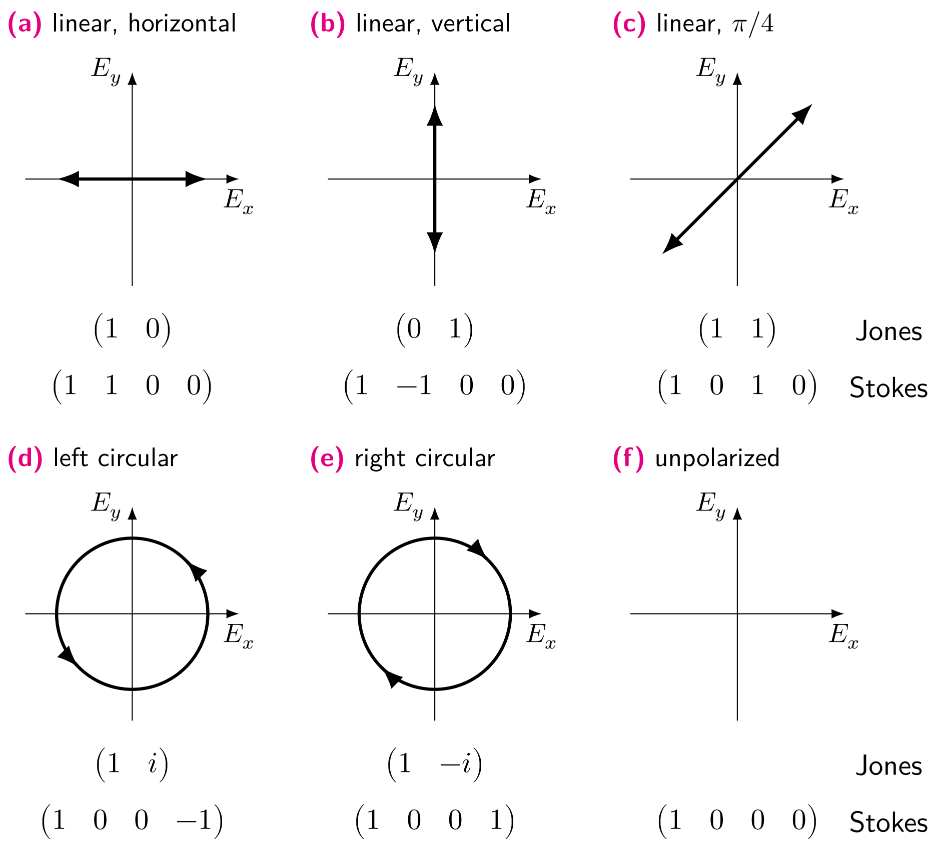


Fig. 3.1.: Illustration of the polarization state of light. Unpolarized light passes through a linear polarizer, polarizing the light in one direction. It then passes through a quarter-wave retarder that converts linearly polarized light (of a specific wavelength) into circularly polarized light, where E_x and E_y are $\pi/2$ in phase.



(a) linear, horizontal
(b) linear, vertical
(c) linear, $\pi/4$
E_y
E_x
$\begin{pmatrix}1 & 0\end{pmatrix}$
$\begin{pmatrix}1 & 1 & 0 & 0\end{pmatrix}$
$\begin{pmatrix}0 & 1\end{pmatrix}$
$\begin{pmatrix}1 & -1 & 0 & 0\end{pmatrix}$
$\begin{pmatrix}1 & 1\end{pmatrix}$
$\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}$
Jones
Stokes
(d) left circular
(e) right circular
(f) unpolarized
$\begin{pmatrix}1 & i\end{pmatrix}$
$\begin{pmatrix}1 & 0 & 0 & -1\end{pmatrix}$
$\begin{pmatrix}1 & -i\end{pmatrix}$
$\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}$
$\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$
Jones
Stokes

Fig. 3.2.: Polarization states in Jones and Stokes convention.

Light in a medium The general Maxwell's equations eqs. 3.1 can be solved similar to section 3.2.2, which yields a special behavior, e. g. the magnetic and electric field component get out of phase. In this work, only the two decisive properties absorption and refraction are described.

3.2.4 Absorption

Absorption is the property of reducing the intensity or energy of an electromagnetic wave passing through a medium. It is described by the Beer–Lambert law of absorption

$$I = I_0 \exp(-\mu x) \tag{3.9}$$

with $\mu = \frac{4\pi\kappa}{\lambda}$ as absorption coefficient, with λ is the wavelength and κ is the imaginary part of the complex refractive index of the medium (see section 3.2.5). If the complex number is inserted into the wave equation, the intensity is reduced exponentially along the path (see eq. (3.9)).

3.2.5 Refraction

Refraction is the change of direction of light as it passes from one medium to another. Refraction can be explained by using the full Maxwell's equations for non-conductive materials, i. e., the differential equation consists of a primary wave with secondary waves induced by the atomic medium, resulting in a decrease in the velocity of the resulting wave. Mathematically, this can be described by a complex number $n = c'/c$. Using this relationship at a boundary surface between two media, one can show that the incident light beam splits into a reflecting and transmitting light wave. The reflecting light wave has the same angle as the incident light beam relative to the surface normal. However, the transmitting light beam, due to the reduction of the velocity, changes its direction described by the Snell's law (see fig. 3.3):

$$n_\alpha \sin\alpha = n_\beta \sin\beta \tag{3.10}$$

The refractive index n can also be described as complex refractive index, where the imaginary part describes the absorption of light along the material (see section 3.2.4):

$$\underline{n} = n + i\kappa \tag{3.11}$$

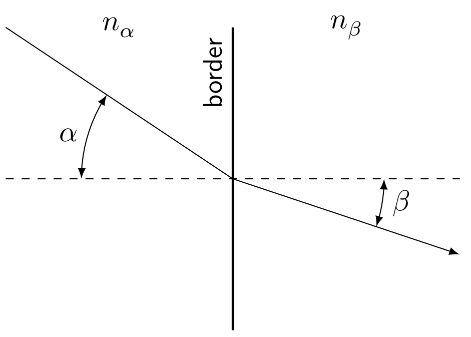


Fig. 3.3.: Illustration of refraction by Snell's law.

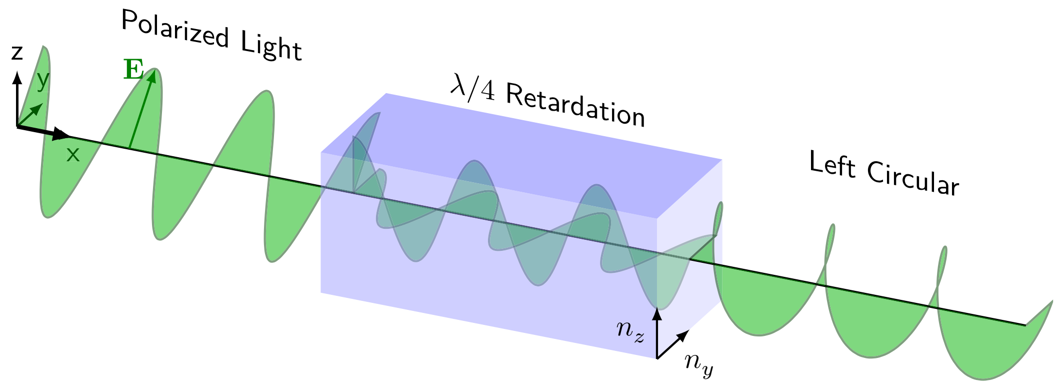


Fig. 3.4.: Illustration of retardation. The linear 45° polarized light wave is decomposed into the x and y components in the birefringent medium. The y component travels faster than the x component. Therefore, a phase shift ϕ between both components occur. This leads in the case of a $\lambda/4$ retarder to a circular polarized light wave.

3.2.6 Birefringence

A translucent material can have a different refractive index depending on the relative orientation and polarization of the light beam. This property is called birefringence. The refractive index can be described by the ordinary refractive index n_o and the extraordinary refractive index n_e, which are perpendicular to each other (see fig. 3.4). Therefore, the light ray can be split into the same perpendicular parts and each can be described by itself. These two light rays can have a different direction due to refraction. If the separation is relatively small, the two light beams (or multiple light beams) can be modeled as they would recombine when leaving the material. The phase change is called birefringence and the physical property is described by:

$$\Delta n = n_e - n_o \,. \tag{3.12}$$

3.2.7 Jones calculus

The Jones calculus, introduced by Robert Clark Jones in 1941 [Jon41], describes the polarization state of a light beam by a complex vector J (see fig. 3.2):

$$\mathbf{J} = \begin{pmatrix} E_x \exp(i\varphi_x) \\ E_y \exp(i\varphi_y) \end{pmatrix} \tag{3.13}$$

The amplitude of the perpendicular components are E_x and E_y with their phase φ_x and φ_y. Optical elements that change the polarization state, such as polarization filters and retarders, can be described by a matrix:

Linear polarizer

$$\boldsymbol{\mathcal{P}}_x = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{\mathcal{P}}_y = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.14}$$

Retarder (fast axis x-axis)

$$\boldsymbol{\mathcal{M}} = \begin{pmatrix} e^{i\delta_x} & 0 \\ 0 & e^{i\delta_y} \end{pmatrix}, \quad \Lambda_{1/4} = e^{\frac{i\pi}{4}} \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \tag{3.15}$$

with δ as retardation and $\Lambda_{1/4}$ as quarter-wave retarder.

Rotation matrix A rotation of an optical element $\boldsymbol{\mathcal{E}}$ can be achieved by a 2D rotation matrix $\boldsymbol{\mathcal{R}}$:

$$\boldsymbol{\mathcal{A}}(\theta) = \boldsymbol{\mathcal{R}}(\theta) \cdot \boldsymbol{\mathcal{A}} \cdot \boldsymbol{\mathcal{R}}(-\theta), \quad \boldsymbol{\mathcal{R}}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{3.16}$$

3.2.8 Müller-Stokes calculus

In analogy to section 3.2.7, the Müller-Stokes formalism, described by George Gabriel Stokes in 1852 [Sto52] and Hans Müller in 1943 [Mue43], also describes the polarization state of a light beam. However, it does not use the absolute electric components, but the relative polarization between both components (see fig. 3.2):

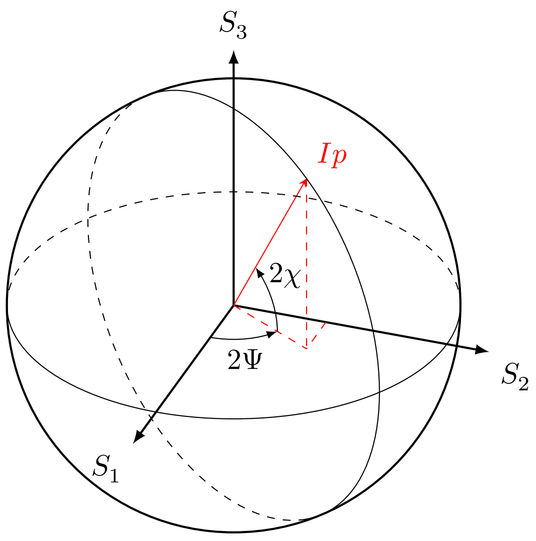


Fig. 3.5.: Poincaré sphere illustrating Stokes component.

Stokes vector

$$\mathbf{S} = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix}, \quad \begin{aligned} S_0 &= I \\ S_1 &= Ip\cos 2\psi \cos 2\chi \\ S_2 &= Ip\sin 2\psi \cos 2\chi &&= 2E_xE_y\cos\delta \\ S_3 &= Ip\sin 2\chi &&= 2E_xE_y\sin\delta \end{aligned} \tag{3.17}$$

where I is the intensity, p the polarization state, Ψ and δ the relative phases between the E_x and E_y components, which can also be described by the two angles Ψ and χ and visualized on the Poincaré sphere (see fig. 3.5). With this description, the phase can no longer be described as in the Jones calculus. However, the relative phase information is stored and can be used to describe unpolarized light or partial polarized light as well as the polarization states of polarization filters, which the Jones calculus cannot describe. Analogous to the Jones calculus, one can formulate the matrices for the optical components:

Linear polarizer

$$\mathcal{P}_x = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathcal{P}_y = \frac{1}{2}\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.18}$$

Retarder (fast axis: x)

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\delta & \sin\delta \\ 0 & 0 & -\sin\delta & \cos\delta \end{pmatrix}, \quad \Lambda_{1/4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.19}$$

with δ as retardation and $\Lambda_{1/4}$ as quarter-wave retarder.

Rotation matrix Analogous to eq. (3.16) rotations of an optical element $\mathcal{E}$ are applied by

$$\begin{aligned} \mathcal{R}(\theta) &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\theta) & -\sin(2\theta) & 0 \\ 0 & \sin(2\theta) & \cos(2\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ \mathcal{A}(\theta) &= \mathcal{R}(\theta) \cdot \mathcal{A} \cdot \mathcal{R}(-\theta) \end{aligned} \tag{3.20}$$

4

3D Polarized Light Imaging

4.1 Introduction

This chapter gives an overview of 3D-PLI. First, the necessary preparation of the brain tissue is described, including its cutting process and the subsequent mounting. The second part describes the 3D-PLI microscopic technique, which allows measuring the orientation of the optic axis that corresponds to the orientation of the myelinated nerve fibers. The foundation of the measured 3D-PLI signal and the optical limits in microscopy are described as a basis for the simulation in the following chapters.

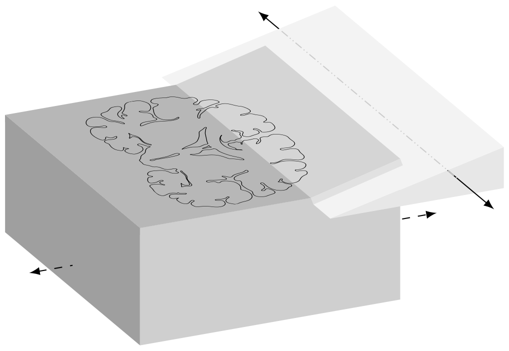
4.2 Brain Tissue Preparation and Sectioning

After the organism death, the brain is removed from the skull within 24 h. In order to prevent the further degeneration of the brain tissue, it is immersed in a solution of 4 % formaldehyde and 20 % glycerin for several days. Then it is frozen at −80 °C as preparation for sectioning. The tissue is fixated with liquid glue on a plate, which allows placing it within a microtome. The microtome consists of a chamber cooled at about −70 °C. A vibrating knife is used to cut the brain into 60 µm sections (see fig. 4.1a). After each cut, each section is manually placed onto a glass specimen, surrounded by glycerin and covered with a thin optical glass (see fig. 4.1b). To preserve the sections for the microscopic measurement, they are placed in a freezer at approximately −70 °C. The 3D-PLI measurement takes place at room temperature. [Axe+11b]

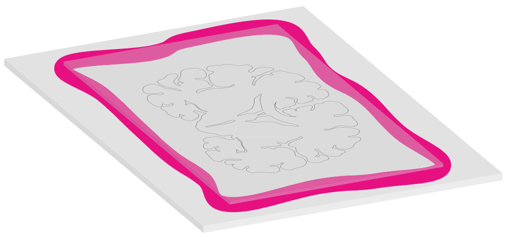
4.3 Experimental Setup

In INM-1, there exist three microscopic setups based on identical physical principles [Axe+11b] (see fig. 4.2). Polarized light with a wavelength of about 525 nm is irradiated through the tissue section. [1] A rotatable polarizer is mounted in front of the tissue

[1]The exact wavelength depends on the microscope.



(a) Illustration of the sectioning process. The brain is fixed with glue to stabilize it for the cutting process. A microtome is used to cut the tissue by using a vibrating diamond blade over which the tissue block is moved.



(b) A section is placed on a glass specimen, to protect it from environmental influences. It is covered with a second, thinner glass plate, which is sealed with nail polish.

Fig. 4.1.: Brain sectioning and sealing illustration.

in the beam path, and a fixed circular polarizer is mounted behind the tissue. By rotating the polarizer, the change in intensity is measured with a charge-coupled device (CCD)-sensor.

The first experimental setup is called large-area polarimeter (LAP).[2] It is used to measure an entire brain section at a resolution of 60 µm. [3]

The large metripol (LMP) microscope allows measuring a tile of 2048×2048 pixels with a pixel size of 1.3 µm. By measuring the tissue with multiple overlapping images, the overlapping tiles can be combined into an overall image with a stitching algorithm. This setup is not able to change the light path. The 3D information can be estimated by analyzing the distribution of retardation and transmittance, however, the inclination sign cannot be detected due to the ambiguity of the signal.

The third setup is the large metripol 3D (LMP3D) microscope [Wie16]. By using a conical light path and a slit, only light of a particular angle can pass and therefore through the tissue. By changing the position of the slit, different light paths can be applied with a maximal tilt angle of 3.9°.

The tilted light beam in the LAP and the LMP3D are measured at four perpendicular orientations: North, East, South and West.

[2] This setup has a slightly different configuration of the optical elements. However, the measurement is the same.

[3] It is also possible to acquire images with different pixel sizes e. g. by changing the focal length, since the setup is not fixed.

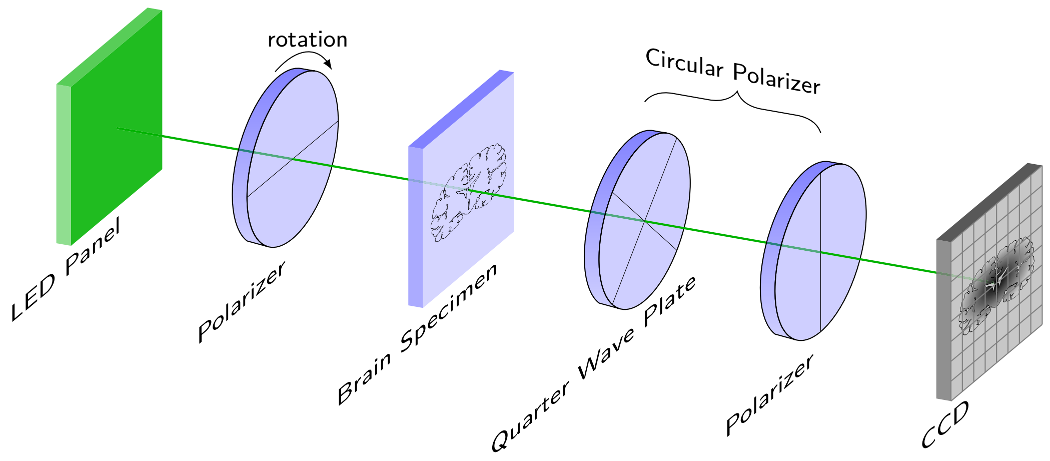


Fig. 4.2.: Illustration of 3D-PLI setup.

The final image is captured by a camera that uses a CCD sensor, consisting of an array of metal oxide semiconductor (MOS) capacitors. Each capacitor stores an electric charge that is released by incident photons using the photoelectric effect. After a readout process, which also includes electrical amplification, the resulting values can be stored as an image. Its value, as long as the capacitors are not saturated or the amplification does not exceed its limits, is linearly correlated with the number of photons. The signal, however, is affected by noise which comes mainly from two parts. First is the thermal noise that can lead to electrical charges in the MOS capacitors. Second, the amplification of the signal underlies a noise usually from a non-ideal direct current and non-ideal electric components like transistors, which is needed for the amplification process. These noise sources combine to produce a Poisson-like distribution due to the nature of digital positive values produced by the analog-to-digital converter. A normal distribution can model the intensity values $\gg 0$.

4.4 Intensity Signal

From the Müller-Stokes matrices (section 3.2.8), the intensity signal, which is the first component of the Stokes vector, follows a sinusoidal curve [Men14; Men18]:

$$I(\rho, \varphi, \alpha, d) = \frac{I_0}{2}\left[1 + \sin(2\rho - 2\varphi) \cdot \sin\left(2\pi \frac{d\Delta n}{\lambda} \cos^2(\alpha)\right)\right] \\ \text{with } \delta := 2\pi \frac{d\Delta n}{\lambda} \cos^2(\alpha) \text{ and } t_{rel} := 4\frac{d\Delta n}{\lambda} \tag{4.1}$$

Eqs. 4.1 describes a sinusoidal signal. For a discrete, equidistant measurement of the rotation angles ρ, one can use the Fourier series with the first three coefficients to describe the signal:

$$\begin{aligned}
\rho &= \left[0, \frac{\pi}{N+1}, \frac{2\pi}{N+1}, ..., \frac{N\pi}{N+1}\right] \\
a_0 &= \frac{1}{N}\sum_{i}^{N} I_i \\
a_1 &= \frac{2}{N}\sum_{i}^{N} I_i \cdot \sin(2\rho_i) \\
b_1 &= \frac{2}{N}\sum_{i}^{N} I_i \cdot \cos(2\rho_i)
\end{aligned} \tag{4.2}$$

The current routine measurements take $N = 9$ images. These are used to calculate the final 3D-PLI modalities (see fig. 4.3):

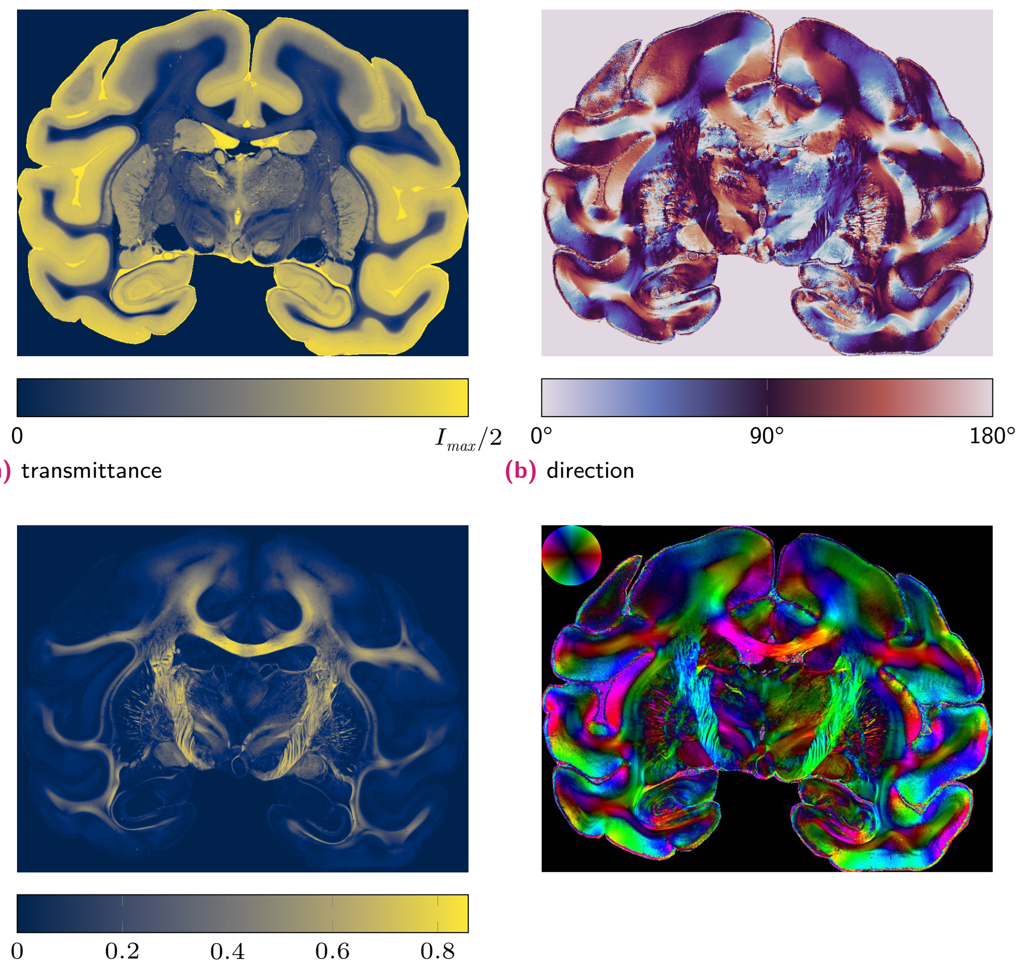
$$\begin{aligned}
\mathit{transmittance} &:= 2 \cdot a_0 && \widehat{=}\, I_0/2 \\
\mathit{direction} &:= 0.5 \cdot \operatorname{atan2}(-b_1/a_1) && \widehat{=}\, \varphi \\
\mathit{retardation} &:= \frac{\sqrt{a_1^2 + b_1^2}}{a_0} && \widehat{=}\, |\sin(\delta)|
\end{aligned} \tag{4.3}$$

The *transmittance* describes the tissue light transmittance, i. e., how much light passes through the tissue. The *direction* describes the signal phase, corresponding to the direction of the macroscopic optic axis and therefore the direction of the nerve fibers. The *retardation* corresponds to the amplitude of the signal, i. e., the strength of the retardation.

4.5 Tilting Analysis

To analyze the optic axis inclination α, one has to distinguish the relative strength of the birefringence from the term $\cos^2(\alpha)$. For this purpose, the tissue is tilted allowing the light signal to be measured through the tissue at a different angle of incidence [Axe+11b; Wie16] (see section 4.3). Therefore, the tissue and its underlying nerve fibers change their orientation due to the tilt angles θ and φ. In addition, the distance the light travels through the tissue increases by $1/\cos(\theta)$ (see fig. 6.6).

Depending on the `pixel_size`, light passes through the same volume but with a different orientation. Therefore, multiple light paths can be measured, and the resulting signals can be used to analyze the inclination α and relative birefringence tissue thickness



(a) transmittance (b) direction

(c) retardation (d) fiber orientation map (FOM)

Fig. 4.3.: 3D-PLI modalities for a coronal section of a Vervet monkey brain.

t_{rel}. The angle of incidence of the light changes the path of the light by Snell's law eq. (3.10). Which results in a perspective shift that a registration process must correct. However, this effect can be neglected in the simulation, since it only adds a parallel offset.

An algorithm was developed under the name *robust orientation fitting via least squares* (*ROFL*) to analyze the tilting signal [Wie16; Sch+18b]. The idea is to fit the measured signals of all light paths simultaneously. Since the change in the signal is proportional to $\cos(\alpha)$, for steep fibers, both the changes of $\frac{\partial}{\partial\alpha}\cos(\alpha)$ and the amplitude of the signal are very small, which leads to the problem of increasing uncertainty while the inclination angle increases.

Another difficulty is that for a smaller `pixel_size`, the light path can no longer be neglected. For the LMP3D-system (with a tilt angle of 3.9° and a tissue thickness of 60 µm), the tilted light path is measured about 4 px away from the non-tilting measurement. This means that other parts of the tissue affect the light from the different tilt views. This effect can be neglected for homogeneous tissue regions, such as parts in the dense WM. However, for single fiber paths, such as visible in the GM or complicated paths like in crossing the effect on the inclination analysis is an open question.

4.6 Optical Resolution

The optical resolution of an imaging system describes the minimum size of an object that can still be resolved. This property is limited by aberration and diffraction. Aberration causes blurring of the image, while diffraction can lead to superimposed diffraction patterns. If diffraction is caused by many small objects in relation to the resolution, this also looks like blurring.

Ernst Abbe was one of the first to describe that the resolution correlates with the light wave λ:

$$d = \frac{\lambda}{2n\sin\theta} = \frac{\lambda}{2\mathrm{NA}} \tag{4.4}$$

d is the minimum resolvable distance, n the refractive index, θ the angle of a light spot, which can be combined to the better known numerical aperture NA. A more common definition is the Rayleigh limit given by

$$d = 1.22\frac{\lambda}{2\mathrm{NA}}\,, \tag{4.5}$$

where d is the distance between two light spots, where the first intensity maxima of the first slit is on the first minima of the second slit (see dashed line fig. 4.4). The wavelength used in 3D-PLI is $\lambda = 525\,\mathrm{nm}$ and a numerical aperture of $\mathrm{NA} \approx 0.15$, which results in a limit of about 2.1 µm [Men18]. In addition, three effects are being applied to the simulated measurement.

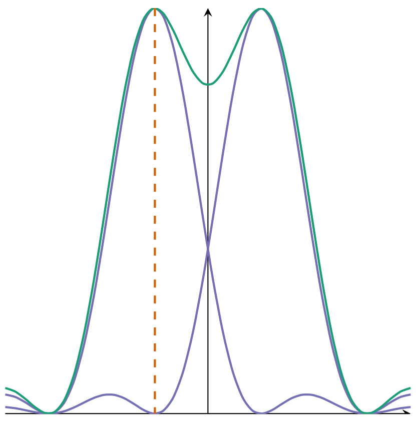


Fig. 4.4.: Rayleigh's criteria. The maxima of the first function is at the the position of the seconds function minima.

Blurring To account for the blurring or out of focus image, the light ray's intensity must be blurred on the CCD-array. This is done via a 2D Gaussian convolution g of the image f:

$$\begin{aligned}(f * g)(x,y) &= \iint f(\tau,\upsilon)\cdot g(x-\tau,y-\upsilon)d\tau d\upsilon \\ g(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\end{aligned} \tag{4.6}$$

Sampling Since the number and final position of the light rays correspond to the voxels (see section 6.2), all intensities of an image pixel must be combined. Here, this is done via a mean value scan:

$$\hat{I}(n,m) = \frac{1}{N_x \cdot N_y} \sum_{i=n\cdot N_x}^{(n+1)\cdot N_x - 1} \sum_{j=m\cdot N_y}^{(m+1)\cdot N_y - 1} I(i,j) \tag{4.7}$$

Unlike resizing, this does not interpolate the image.

Noise The final step is to recreate the noise of the image composition. To account for this, a noise model must be applied to each image pixel. [Wie16] showed that this can be modeled via a normal distribution:

$$I = \mathrm{normal}(\sigma = \sqrt{I \cdot gain}, \mu = I) \tag{4.8}$$

All three effects must be characterized for the system being simulated.

Part II

Software Implementation

5 Dense Nerve Fiber Modeling

In chapter 2 the structure of nerve fibers and the macroscopic structure of white matter (WM) were described. The question is how to represent such a structure in a computer algorithm? For simple fiber configurations, e. g. parallel, linear nerve fibers, this is relatively straightforward. However, it has been shown that irregular, non-symmetric nerve fiber configurations are necessary to obtain a realistic result for microscopy simulations based on the wave nature of the light [Men18]. Therefore, one ideally needs a representation, which allows building any kind of geometry.

Many representations of fiber structures are already available, which are commonly used in graphical visualizations. For example, a rope is represented by a tube object, that is defined by a path with a radius. A surrounding mesh is generated with n-polygonal vertices. For visualization, the mesh is used to apply textures onto the surface. Such meshes are e. g. used in Monte Carlo simulations of diffusion Magnetic Resonance Imaging (dMRI) [Gin+19; Gin19]. Using meshes helps to calculate whether a water molecule travels through the surface of a nerve fiber, i. e., the surface of the mesh. However, this representation is computationally very intensive since the number of triangular faces that must be present in the mesh is quite high.

When creating complex nerve fiber structures, it is important that the nerve fibers do not overlap. To achieve non overlapping fibers, either the user must define such a structure in advance, or a computer algorithm must build such structures automatically. Considering the immense number of configuration possibilities that nerve fibers can have even in a small volume, the task is almost impossible.

The solution found in this work is to initially allow the placement of nerve fibers in any trajectory and without restrictions. The overlap of the fibers will be removed in a secondary step, by checking all fibers for collisions with each other. If a collision occurs, the algorithm tries to remove it by moving the colliding parts slightly away from each other. This step is repeated until the volume has no collisions.

The nerve fibers simulation and all its necessary prerequisites is described in detail in this chapter. First, the representation of a nerve fiber is defined considering the performance for collision optimization. Then, additional *building* functions are designed to help the user define a volume with nerve fibers quickly. Finally, the algorithm for resolving the collisions with the parallelization components is explained in detail. In this chapter, another tissue modeling method developed in collaboration with Neurospin Alternative Energies and Atomic Energy Commission (CEA) is presented, in which neurons such as astrocytes or oligodendrocytes are placed in a volume containing fibers that are relevant for simulations in dMRI.

The algorithms are part of the toolbox *fastPLI* [Mat+19; Mat+21], which is described in detail in chapter 7. The original idea of a collision solving algorithm was initially published in [Mat+19].

5.1 Nerve Fiber Representation

Nerve fibers are tube-like structures surrounded by an electrically insulating lipid layer of myelin (see section 2.3). Both the diameter of the axon and the thickness of the myelin can vary from fiber to fiber (see section 2.4 and table 2.2), which means that the representation of nerve fiber models should account for such variations.

As described in section 2.3, WM consists of nerve fibers packed tightly together in nerve fiber bundles (see fig. 2.3b). These bundles can join with other bundles and traverse the brain to connect one region to another. The bundles can cross each other, either by crossing the individual nerve fibers or bypassing the fiber bundles, forming interwoven structures. [1] Such structures are essential to increase the understanding of 3D-PLI. Therefore, the simulation models must create such configurations without overlapping individual fibers.

To represent individual nerve fibers, a function $\mathbf{f}(t)$ describing a path in 3D space, i. e., a parametrized curve $(x(t), y(t), z(t))$, is used. In addition, a fourth element describes the radius $(r(t))$ of the fiber at the same point:

$$\mathbf{f}(t) \to (x(t), y(t), z(t), r(t)) \mid t \in \mathbb{R} \tag{5.1}$$

However, a continuous representation does not allow simple subsequent changes, such as resolving collisions. For this purpose, individual elements are more manageable, allowing movement or deformation. The fiber is then described by a list of 4D points (x, y, z, r), which can be interpreted as a chain of cylindrical fiber segments (see fig. 5.1):

[1] The terminology "crossing" is relative to the resolution and voxel size.

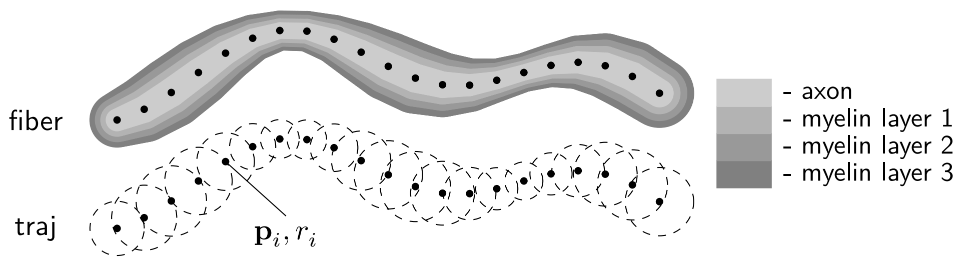


Fig. 5.1.: Representation of a nerve fiber from a list of spheres.

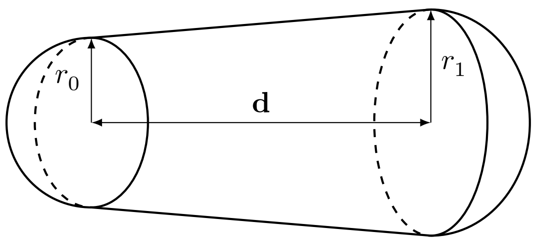


Fig. 5.2.: Schematic of a fiber segment represented by a distance $\mathbf{d}$ and two radii r_0 and r_1 which describes a conical capsule (CC).

$$\begin{aligned} fiber &:= \left\{\mathbf{p}_i = (x_i, y_i, z_i), r_i \mid x, y, z \in \mathbb{R},\, r \in \mathbb{R}^+,\, i \in \{0, 1, ..., N_{points} - 1\}\right\} \\ fiber\_segment_i &:= (\mathbf{p}_i, \mathbf{p}_{i+1}, r_i, r_{i+1}),\, i \in \{0, 1, ..., N_{points} - 2\}\,. \end{aligned} \tag{5.2}$$

Since the fiber radius can change from one point to an adjacent point, the segment can be conical. Therefore, the fiber segments describe a conical capsule (CC) (see fig. 5.2).

This representation has the advantage that much fewer data is needed than a surface's mesh representation, which increases the computational speed for the collision solving algorithm.

5.2 Sandbox

To build dense WM models, one needs to fill a volume with individual nerve fibers. To allow the user to build such volumes, the module `fastpli.model.sandbox` was developed. The user can define a nerve fiber bundle geometry and fill it with a fiber pattern, creating nerve fiber bundles from cylindrical or cubic shapes.

The module `fastpli.model.sandbox` is divided into two submodules. The first module handles the seeding process, while the second part builds up a nerve fiber bundle or volume filled with individual fibers from seed points.

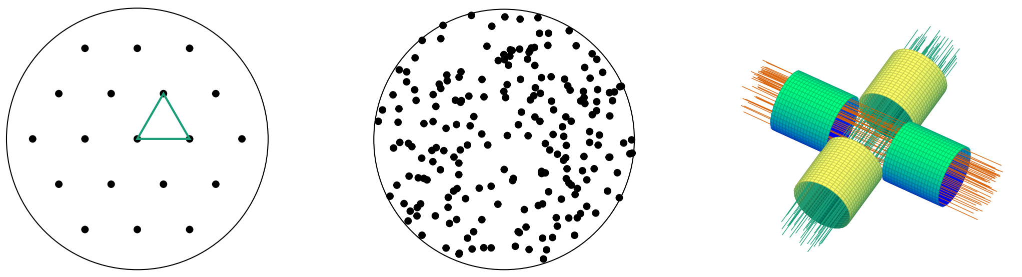

(a) Equilateral triangle grid. (b) Random grid. (c) Populated fiber bundles.

Fig. 5.3.: Populating fiber bundles with 2D seed points.

5.2.1 Seeding fiber bundles

Seed points are stored as a list of 2D points:

$$seeds := \left\{\mathbf{p}_i = (x_i, y_i) \mid x, y \in \mathbb{R},\, i \in \{0, 1, ..., N_{seed\_points} - 1\}\right\} . \tag{5.3}$$

To form particularly dense fiber bundles, a method for generating an equilateral triangular grid is implemented (see 5.3a). Mathematically, this yields the most densely packed pattern for circles with equal radius in a 2D area. However, a regular symmetric grid can lead to unrealistic results (e. g. diffraction pattern in light wave simulation [Men18]). It should, therefore, only be used as an initial configuration. Since the initial configurations are often unknown, it is probably best to choose a random distribution (see fig. 5.3b). For a circular boundary with radius R, this can be sampled from a $\mathrm{uniform}$ distribution:

$$\begin{aligned} \varphi &= \mathrm{uniform}(0, 2\pi) \\ r &= R\sqrt{\mathrm{uniform}(0,1)} \end{aligned} \quad \Rightarrow \quad \begin{aligned} x &= r\cos(\varphi) \\ y &= r\sin(\varphi) \end{aligned} \tag{5.4}$$

5.2.2 Populating fiber bundles

To populate a fiber bundle (see fig. 5.3c), the plane containing the seed points is placed at each fiber bundle point $\mathbf{fb}_i$ along the trajectory. Essentially, for each resulting $fiber_j$, this means:

$$fiber_j = \left\{\boldsymbol{\mathcal{R}} \cdot (x_j, y_j, 0) + \mathbf{fb}_i \,|\, i \in \{0, 1, ..., N_{seed\_points} - 1\}\right\} \tag{5.5}$$

Since a trajectory is only a 2D object, i. e., a line, no unique normal vector can be defined. Theoretically, any torsion can be applied to any 2D curve in space. However, these do not occur in principle in the tissue.

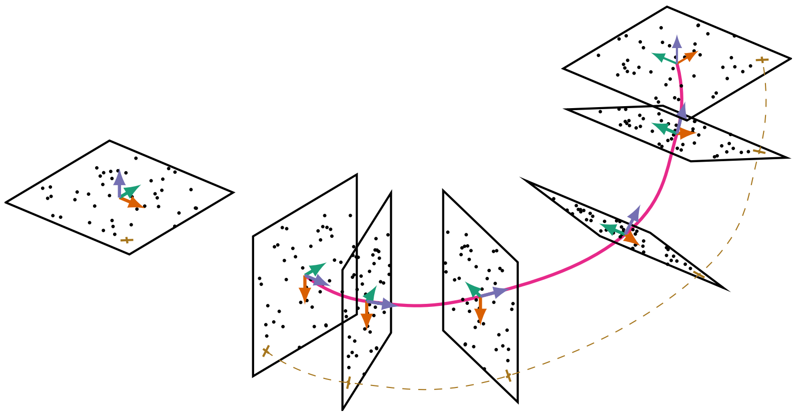


Fig. 5.4.: Left: Plane of seed points. Right: The plane is rotated and placed along the path. At the end, the plane has the same normal vector as the starting plane, but due to the principle of minimum rotation, the plane is rotated along the normal vector.

A more reasonable solution is to perform a rotation such that the rotation along the fiber bundle trajectory is minimal, which can be achieved by computing the rotation matrix that rotates the current tangent vector $\mathbf{t}(i)$ to that of the next point $\mathbf{t}(i+1)$ (see fig. 5.4). The rotation matrix $\mathcal{R}(\mathbf{a}, \mathbf{b})$ for between two nonparallel vectors $\mathbf{a} \nparallel \mathbf{b}$ is

$$\begin{aligned}
\mathbf{a} &= \mathbf{a}/|\mathbf{a}| & \quad \mathcal{U} &= \begin{pmatrix} 0 & \mathbf{v}_z & -\mathbf{v}_y \\ -\mathbf{v}_z & 0 & \mathbf{v}_x \\ \mathbf{v}_y & -\mathbf{v}_x & 0 \end{pmatrix} \\
\mathbf{b} &= \mathbf{b}/|\mathbf{b}| & & \\
\mathbf{v} &= \mathbf{a} \times \mathbf{b} & & \\
c &= \mathbf{a} \cdot \mathbf{b} & \quad \mathcal{R} &= \mathbb{1} + \mathcal{U} + (\mathcal{U} \cdot \mathcal{U}) \cdot (1-c)/|\mathbf{v}|^2
\end{aligned} \tag{5.6}$$

In the case of $\mathbf{a} \parallel \mathbf{b}$ no rotation is necessary.

The presented formulation allows generating a filled nerve fiber bundle. First, the seed point plane is placed at the first fiber bundle point $\mathbf{fb}_0$ and rotated by $\mathcal{R}(\hat{\mathrm{e}}_z, \mathbf{t}_0)$. Then, for each step, the plane is rotated at its origin by $\mathcal{R}(\mathbf{t}_i, \mathbf{t}_{i+1})$ and placed on $\mathbf{fb}_{i+1}$. To smooth the transition, the tangential vector $\mathbf{t}$ at step i is the average of the neighboring points:

$$\mathbf{t} = \frac{1}{2} \frac{\mathbf{fb}_{i-1} + \mathbf{fb}_{i+1}}{|\mathbf{fb}_{i-1} + \mathbf{fb}_{i+1}|} \tag{5.7}$$

Finally, all points belonging to a fiber are stored in a fiber bundle object as $(n \times 4)$-array.

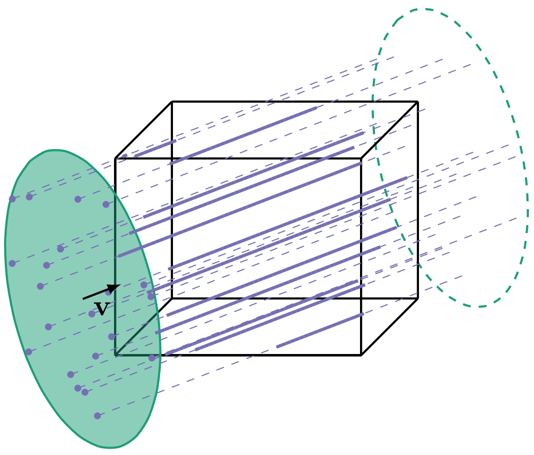


Fig. 5.5.: Populating a cuboid with straight fibers initialized by seed points along the direction $\mathbf{v}$.

5.2.3 Cube models

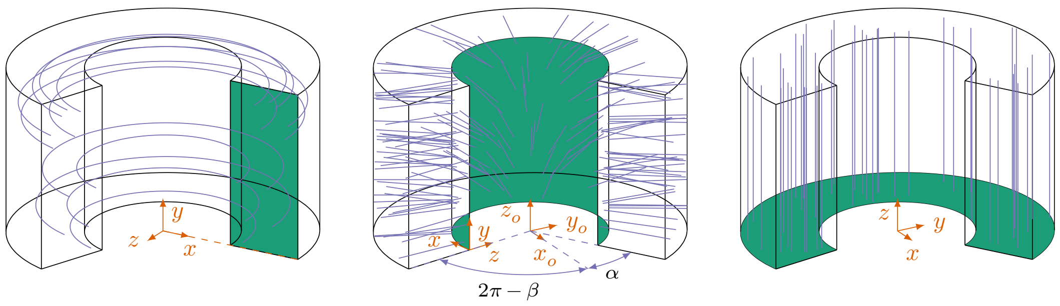
3D-PLI simulations calculate all light vectors inside a cubical volume (see chapter 6). Therefore, a method exists to fill a cube with a fiber population of orientation $\mathbf{v}$ (see fig. 5.5). The individual fibers are initialized with a seed point plane. This plane is placed virtually in front of and behind the cubic volume, with a user-defined orientation. The coordinate origins of the cube and the two seed point planes are on a line. To fill the volume with nerve fibers the seed points are connected. When a line hits the volume, the entry and exit points are calculated and stored as fibers of the returned fiber bundle.

5.2.4 Cylindrical models

The last method allows populating a cylindrical volume with fibers. Since a cylinder has three symmetries, radial, angular and height, those three symmetries have been implemented to provide a way to populate the volume.

The cylinder has an outer radius r_{out} and an inner radius r_{in}. The height h is aligned along the z-axis and starts at $(0, 0, 0)$. In addition, the cylinder can be cut radially between the directional angles α and β to fill only a portion of it. The angles correspond to the cylindrical coordinate system.

The following applies to all cylindrical methods used: If the seed point plane leaves the cross-section plane to be filled, the seed points lying outside are ignored.



(a) Circular population. **(b)** Radial population. **(c)** Parallel population.

Fig. 5.6.: Populating of cylindrical objects. The green area shows the area corresponding to the seed point plane. The coordinate system indicates the coordinate origin corresponding to the seed points origin. The origin of the coordinate system O of the resulting fibers is at the bottom center as shown in fig. 5.6b. Two angles α and β can be used to limit the angular range of the cylinder.

a) circular Mimics a radial path of the cylinder (see fig. 5.6a). The seed points are placed along the surface of the cross-section of the first α direction angle. The origin of the seed point plane is placed on the origin of the cylinder. From there, the fiber is bent circular until the second direction angle β. The step size of the circular path can be customized.

b) radial The seed point plane is placed at the inner wall with the origin at the bottom corner of the first angle α (see fig. 5.6b). The fibers then are generated radially until they meet the outer wall of the cylinder. Thus, the density of the fibers decreases along their path.

c) parallel The fibers are aligned along the cylinder (see fig. 5.6c). The seed points are placed on the lower plane, with identical coordinate origins and orientations (see fig. 5.6c).

Only seed points that are in contact with the cylinder are considered in all methods.

5.3 Solving Fiber Collisions

The following algorithm allows the user to define any fiber path, which allows a wide range of freedom in the initialization process. Since the user initialization will probably produce colliding fibers, the collisions have to be found and solved by moving the affected fiber segments so that a collision-free volume is created by minimal displacement. This allows the user to specify complex interwoven structures such as nerve fiber crossings in

```
    def step():
        # Reset Parameter
        SetSpeed(objects, 0)

        # Building Octree
        octree = Octree(objects)

        # Collision Detection
        for leaf in octree:
            colliding_objs = CheckLeaf(leaf.fiber_list)
            colliding_list.insert(colliding_objs)

        # Separation Process
        MoveObject(colliding_list)

        # Shape Control
        SegmentLength(colliding_list, target_length)
        BendingRadius(colliding_list, target_curvature)

        return colliding_list.is_empty()
```

Alg. 1: Main structure in a single step of the collision checking and shape controlling algorithm.

any configuration. The algorithm allows the user to specify boundary conditions like the mean fiber segment length or minimal bending radius.

A stand-alone algorithm is published in [Mat+19]. The solving algorithm is publicly available as a module of *fastPLI* [Mat+21], `fastpli.model.solver.Solver`. An essential feature is the visualization of the displacement process. This allows the user to see the movement and intervene as early as possible if necessary. This is very important because the solution process can take a lot of time, depending on the volume and the number of objects.

5.3.1 Solver main function

The main function is composed of the following sequential parts (see alg. 1):

- ordering the objects in an octree
- checking each branch of the octree for colliding objects
- separating the colliding objects
- checking the shape of the fibers, i. e., their length and bending radius

After a `step` the user is allowed to interact if necessary. The function returns a boolean value indicating if no more colliding objects were found. Therefore, a simple while

```
def aabb_collide(aabb_0, aabb_1):
for i in dim(aabb):
if aabb_0[i].min > aabb_1[i].max:
return false
if aabb_0[i].max < aabb_1[i].min:
return false
return true
```

Alg. 2: Calculation if a collision between AABBs exists.

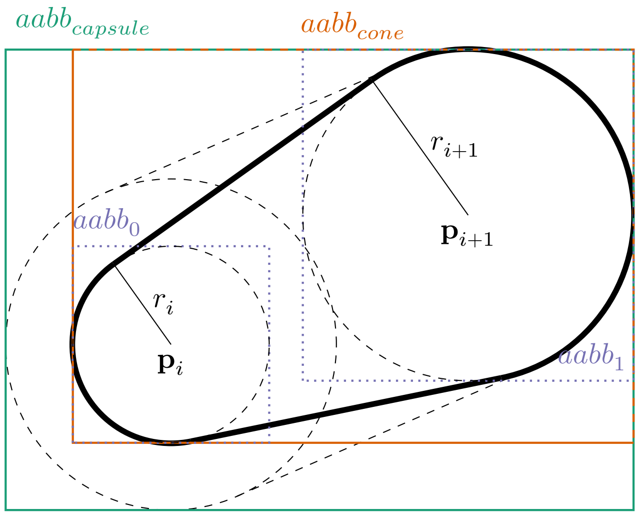


Fig. 5.7.: Fiber segment representations with the AABBs: — CC, -- capsule, -- bounding box.

loop can be used to iterate the collision algorithm until a non colliding configuration is found.

5.3.2 Collision detection

As described in section 5.1, nerve fibers are represented as a chain of spheres, with two adjacent spheres representing a fiber segment that forms a conical capsule (CC) (see fig. 5.7). A collision between two CC involves a few calculations (see alg. 3). To reduce the runtime axis aligned bounding boxes (AABBs) are first checked for collision. This check is fast to calculate (see alg. 2). If a collision occurs between the two AABBs, the actual collision calculation is performed. However, this is a non-trivial task and computationally very intensive. Therefore, it was decided to change the object representation for the collision detection from a CC to a capsule (see fig. 5.7), with a radius equal to the maximum of the two original spheres $r_{capsule} = \max(r_0, r_1)$. This enables the computational effort to be reduced, although some volume is lost. This is of minor consequence, since the change in radius is expected to be relatively small.

```python
def MinDistance(cone_a, cone_b):
   # from https://www.john.geek.nz

   u = cone_a.p1 - cone_a.p0
   v = cone_b.p1 - cone_b.p0
   w = cone_a.p0 - cone_b.p0

   a = np.dot(u, u)
   b = np.dot(u, v)
   c = np.dot(v, v)
   d = np.dot(u, w)
   e = np.dot(v, w)
   f = a * c - b * b

   if f < 1e-5:
      sN = 0.0
      sD = 1.0
      tN = e
      tD = c
   else:
      sN = b * e - c * d
      tN = a * e - b * d
      if sN < 0.0:
         sN = 0.0
         tN = e
         tD = c
      elif sN > sD:
         sN = sD
         tN = e + b
         tD = c

   if tN < 0.0:
      tN = 0.0

      if -d < 0.0:
         sN = 0.0
      elif -d > a:
         sN = sD
      else
         sN = -d
         sD = a
   elif tN > tD:
      tN = tD
      if (-d + b) < 0.0:
         sN = 0
      elif (-d + b) > a:
         sN = sD
      else
         sN = (-d + b)
         sD = a

   if np.abs(sN) < 1e-5:
      sc = 0.0
   else:
      sc = sN / sD
   if np.abs(tN) < 1e-5:
      tc = 0.0
   else:
      tc = tN / tD

   P = cone_a.p0 + u * sc
   Q = cone_b.p0 + v * tc
   length = np.linalg.norm(P, Q)
   return length, P, Q
```

Alg. 3: Collision detection between two capsule objects. A collision occurs if the distance is less than `cone_a.r + cone_b.r > d`. Original: https://www.john.geek.nz/2009/03/code-shortest-distance-between-any-two-line-segments/.

The algorithm for detecting collisions between two capsules is described in alg. 3. It works on the principle that it calculates the shortest distance between two line segments. Three cases can occur. First, the shortest distance is a line perpendicular to the two line segments (see fig. 5.8). Second, only one line segment is perpendicular to the shortest distance line. The other has an anchor point either at the beginning or at the end of the second line segment. Third, the shortest distance is a connection between one of the points of the line segments. For cones, a collision occurs when the distance is less than the sum of the two radii.

The calculation of the collision check is the most complex function. Therefore, the amount of collision checks needs to be reduced as much as possible. A collision check compares each object to all other objects, which leads to a computational cost of $\mathcal{O}(n^2)$, which is not acceptable for large n. Therefore, an octree based strategy was selected to reduce the number of computations.

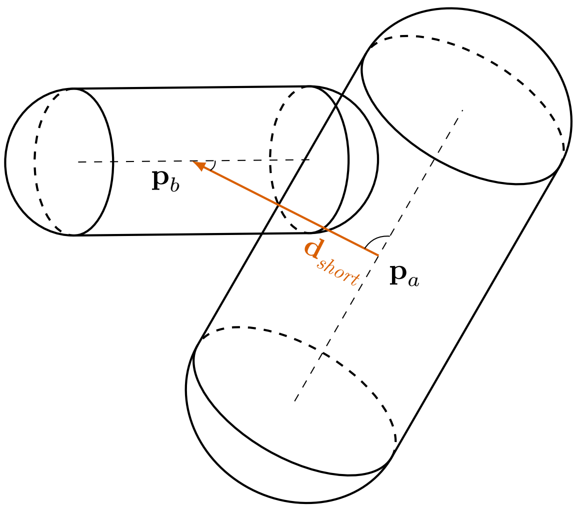


Fig. 5.8.: Shortest distance between two capsule objects. The line must be either perpendicular to the line segments or at least in one of the points of each object.

```python
def octree(volume, objects):
    if num(objects) > threshold:
        sub_volumes, sub_objects = split(volume, objects)
        leaves = [octree(v,o) for v,o in zip(sub_volumes, sub_objects)]
    else:
        leaves = [objects]
    return leaves
```

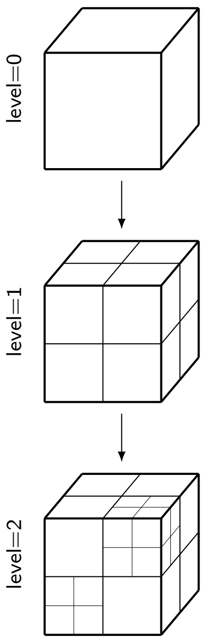
Alg. 4: Recursive generation of an octree.

5.3.3 Octree

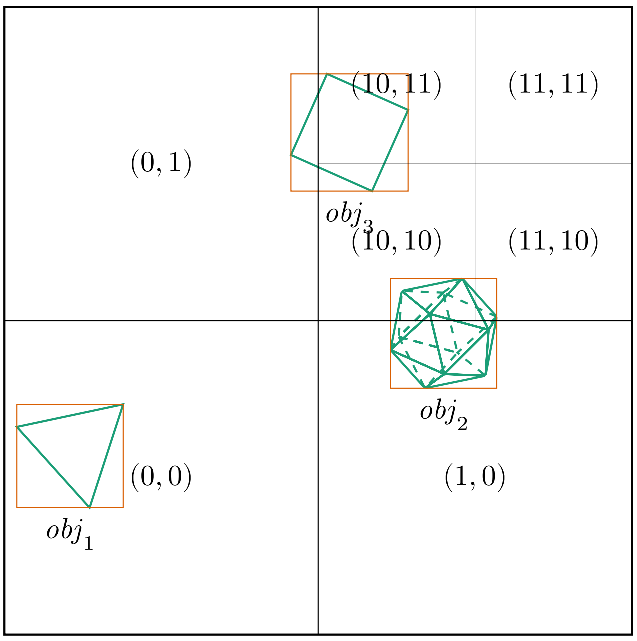
A *tree* is a data structure consisting of a collection of interconnected *nodes*. A *node* is connected to a single parent *node*, and multiple *children* via *branches*. The first parent node is called *root*. The final *nodes* at the end of a *branch* are called *leaves* containing the data. Traversing a uniformly distributed *tree* has the advantage that the cost of traversal is $\mathcal{O}(\log(n))$.

An *octree* is a special *tree* where each node contains eight children, allowing a cubic volume to be divided into eight subvolumes. An example is shown in fig. 5.9a. The length of the subvolumes shrinks exponentially by $(1/2)^{level}$. In this algorithm a recursion function generates the tree structure (see alg. 4).

First, all objects must be sorted into the eight branches of the initial node (if the number of objects is not too small). This means that each object must be checked whether there is a collision with one or more of the eight subvolumes. To reduce the computational



(a) Exemplary octree subdivision of a cube.



(b) Exemplary collision in 2d. The boxes corresponds to the AABBs.

Fig. 5.9.: Exemplary tree subdivision.

costs, as before, the AABBs of the objects are used for the collision check (see fig. 5.9b). Once all objects are sorted into their respective subvolumes, recursion can begin.

Since a branch can be considered a node, the same algorithm can be executed until a desired boundary or property is reached, e. g. maximum number of branches. To speed up the algorithm, the objects for each subvolume are stored in memory linearly. The conditions under which the recursion is to be stopped cannot be defined unambiguously. This depends on the algorithm implemented and the number and position of the objects. The usual approach is to impose the following two constraints:

Maximum number of levels In the case of nerve fibers, it can be assumed that the size of the objects is approximately in the same order of magnitude. Therefore, the largest object in a volume can be used as the lower bound for the smallest volume in the octree. A difference in a higher order of magnitude consequently increases the computational cost to the point where each object must be retested with each object.[2]

Minimal number of objects Once a leaf is created, all objects contained in it must be checked for collisions. As mentioned above, this is a task of $O(n^2)$. However, there is

[2]There are other, more suitable algorithms for this case. However, since this case is not expected, they are not implemented.

also an upper bound of a number of objects for which it is faster to calculate whether they collide than to partition the volume further. This number must be checked at runtime, since it depends on many factors, e. g. central processing unit (CPU) cache size. In the development phase of this algorithm, a value of ≈ 20 for the involved computer architectures was determined.

The collision checking algorithm is executed on each leaf (see section 5.3.2) and all colliding objects are identified.

5.3.4 Separation phase

To resolve a collision between two objects, each point $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ of the two objects is moved. To effectively move the objects apart with a small effort, the fiber objects are translated and rotated away from the other colliding object. The translation is parallel to the shortest distance vector of the collision. For the rotation, the direction of motion is weighted by the distance of each point from the intersection with the smallest distance line (see fig. 5.8 and alg. 5).

```

def push_obj_apart(obj_a, obj_b):
    P, Q = min_distance_points(obj_a, obj_b)

    delta = P - Q
    norm = length(P - Q)

    if norm < 1e-8:
        return random_direction()

    speed = 0.05 * min(r0, min(r1, min(obj.r0, obj.r1)))

    v_obj_a_0 = delta / norm * speed * length(P - p1) /
                length(p1 - p0)
    v_obj_a_1 = delta / norm * speed * length(P - p0) /
                length(p1 - p0)

    v_obj_b_0 = -delta / norm * speed * length(Q - obj.p1) /
                length(obj.p1 - obj.p0)
    v_obj_b_1 = -delta / norm * speed * length(Q - obj.p0) /
                length(obj.p1 - obj.p0)

    return v_obj_a_0, v_obj_a_1, v_obj_b_0, v_obj_b_1
```

Alg. 5: Velocity calculation of colliding objects.

The velocity is stored in an array for each fiber corresponding to the fibers points. All velocities for a single point is summed up.

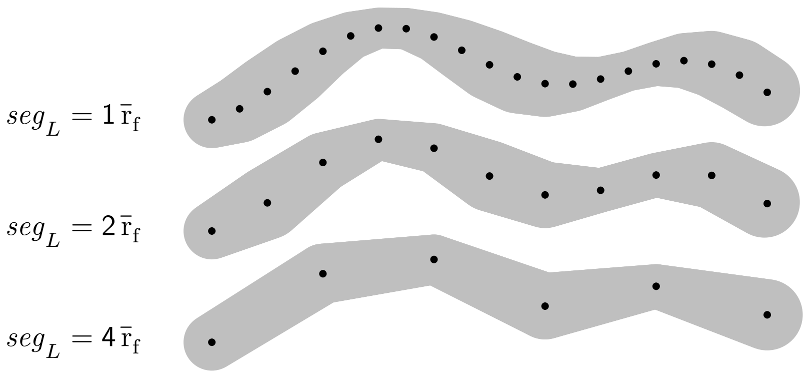

$seg_L = 1\,\bar{\mathrm{r}}_\mathrm{f}$
$seg_L = 2\,\bar{\mathrm{r}}_\mathrm{f}$
$seg_L = 4\,\bar{\mathrm{r}}_\mathrm{f}$

Fig. 5.10.: Different fiber segment lengths as factor of the mean fiber radius $\bar{r}_f$.

A special case is the movement of the first and last point of a fiber. These may only move perpendicular to the first/last segment line, which prevents the fibers from growing to infinity.

5.4 Shape Control

The movement of single points can lead to a distorted fiber model, e. g. two points move very far apart. Therefore, boundary conditions are specified. It was decided to use the following two properties, the mean segment length and the minimum bending radius, as parameters for the shape control. Each parameter can be set to 0 to disable the boundary condition.

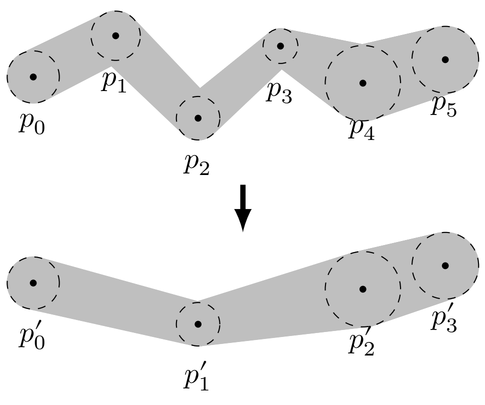
5.4.1 Mean segment length

The average segment length is the distance between the two points of a fiber segment (see fig. 5.10). When the segment length becomes too small/large, the points within a fiber corresponding to the object are merged/separated, and one less point/one new point is removed/added (see fig. 5.11). The segment is allowed to have a length inside $[\frac{2}{3}\bar{d}, \frac{4}{3}\bar{d}]$. Thus, the mean value of the object is:

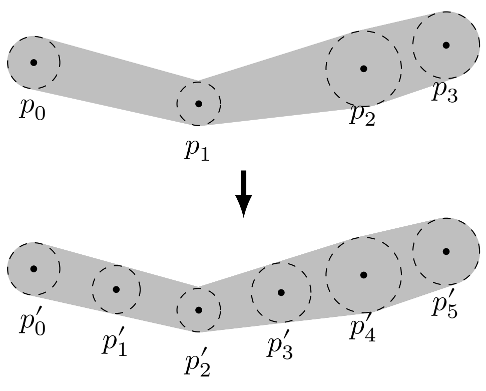
$$\frac{d_{\min} + d_{\max}}{2} = \bar{d} \tag{5.8}$$

If a new point is created when exceeding the maximum limit, the new point $\mathbf{p}_{new}$ with a radius r_{new} and velocity $\mathbf{v}_{new}$ are calculated by linear interpolation:

$$\mathbf{p}_{new} = \frac{\mathbf{p}_i + \mathbf{p}_{i+1}}{2}, \quad r_{new} = \frac{r_i + r_{i+1}}{2}, \quad \mathbf{v}_{new} = \frac{\mathbf{v}_i + \mathbf{v}_{i+1}}{2} \tag{5.9}$$



(a) merge



(b) split

Fig. 5.11.: Length control for the f and f' fibers. Illustrated is the merging and splitting of points, if necessary. The first and last point are never removed.

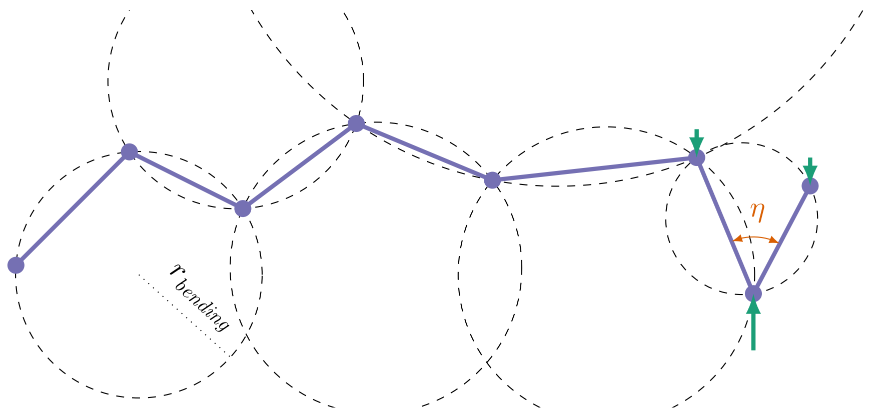


Fig. 5.12.: A fiber along its path can be characterized by circles from three adjacent points. A minimal radius for these circles is applied as boundary condition. Additionally the inner angle has to be $\eta \geq 60°$. If a condition is not fulfilled, a force (arrows) is applied to smoothen the curvature at these points.

5.4.2 Bending radius

The bending radius matches the radius of the circle defined by three adjacent points $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$ (see fig. 5.12). To limit the bending radius of the fiber, a minimal allowed radius r_{min} is defined. Additionally, three adjacent points are not allowed to have an enclosing angle $\eta \geq 60°$. If any point $\mathbf{p}_i$ in a fiber fails these boundary conditions, the three points $\mathbf{p}_{i-1}, \mathbf{p}_i$ and $\mathbf{p}_{i+1}$ will be moved parallel to reduce curvature (see fig. 5.12).

5.4.3 Movement phase

All movements are stored additive in a velocity array before the total movement is executed. The maximum velocity is limited by $v_{\max} = 0.1 \times \min(\mathrm{r_f})$.[3] This is a necessary constraint due to two required properties: First, an object should not be allowed passing through another object, i. e., the velocity should always be $< min(r_f)$. Second, it smooths the motion and thus the maximum achievable density of the resulting models.

After limiting the velocity, the movement of all points is executed. A resistance value is applied to reduce the velocity by the appropriate factor after each step. This can help to reach a collision free volume faster, but the density will be reduced. Its default value is set to 0 so that the velocity is reset to $\mathbf{0}$ after each step.

5.4.4 Optimization and parallelization

The computational architecture to optimize the octree (see section 5.3.3) for solving the fiber collision is presented in this section. The optimization requires millions of objects to be checked for collisions. By using memory alignment and multiprocessing, the computation cost was reduced from $O(n^2)$ to $O(n \log(n))$.

Memory alignment All algorithms use in-memory aligned data types, e. g. `std::vector`, as appropriate, allowing the use of the CPU's prefetcher. During the development process of the octree, it was found that creating a memory-aligned copy of the subset of the objects improves the performance compared to referring to the objects by reference in the different branches. Each optimization was tested for improvements for different volumes and number of objects. The focus is on larger volumes, and a more significant number of objects, since smaller volumes or a smaller number of objects require significantly less computing time.

OpenMP To use multiple CPUs, the Open Multi-Processing (OpenMP) library is used. A structure like an octree suggests the parallelization of 8 cores. Each leaf can be traversed in parallel. In addition, other functions that are thread-safe are parallelized with OpenMP, e. g. moving the points.

[3]This value has proven to be appropriate during the testing phase of this algorithm. Speedup may still be possible by changing this value without significantly changing the results.

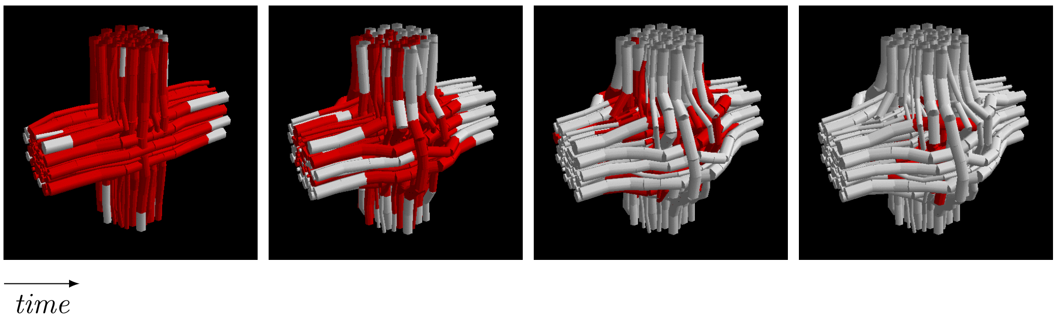


Fig. 5.13.: Exemplary visualization in the collision solving process. The red color indicates that a collision is detected in the fiber segment.

5.5 Visualization

A visualization tool is available to visualize the nerve fiber configuration (see fig. 5.13). This allows the user to get direct feedback, e. g. after each step, to adjust the initial fiber configuration or boundary conditions. It is written in *C++* and *Open Graphics Library* (*OpenGL*) *v2* [@Fou20; @Wik18]. This implementation uses `gluCylinder` to represent a nerve fiber segment. Although not optimal, the visualization is sufficient compared to a single step of the collision solving algorithm.

A further advanced interactive tool is also available as open source, the *Fiber Architecture Constructor* (*FAConstructor*), which was developed by Jan Reuter as part of his bachelor thesis [Reu+19]. It provides additional interactive methods for creating nerve fiber models and was written in *OpenGL v3*.

5.5.1 Transparent objects and cells

In section 5.6 the need of transparent objects is necessary. Not only the outer hull of the myelin, but also the inner axon needs to be seen in the visualization. Additionally, cells, e. g. astrocytes, have to be visualized. For this purpose, the former algorithm was rewritten. For the transparent effects, the algorithm needs to know the order in which the objects or triangular surfaces needs to be rendered. This means, that the triangular surfaces have to be sorted in z-direction, i. e., the view direction. To order the triangular surfaces, they first have to be calculated. The fiber segments are no longer represented as a cylinder, but as CC which consist out of a hexagonal grid (see fig. 5.14).

This is a huge amount of computational resources. It is not yet included in the *fastPLI* package. The resulting visualization is shown in section 5.6.1.

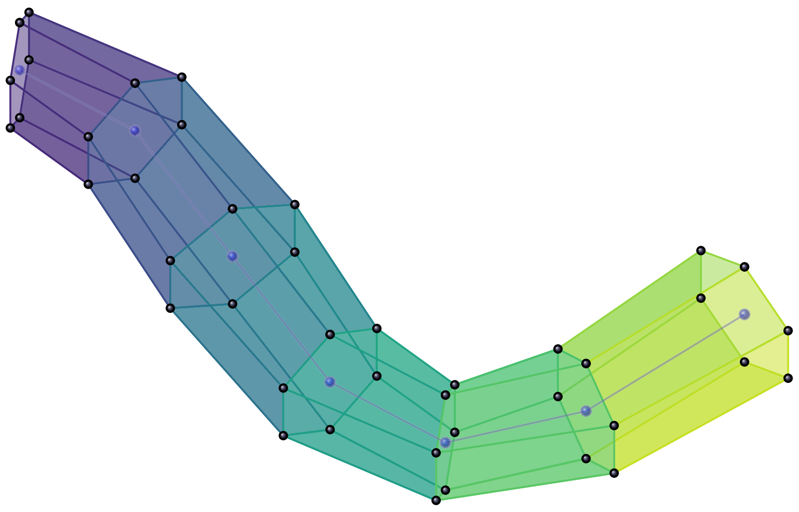


Fig. 5.14.: Generate a mesh for visualization. A mesh perpendicular to the fiber trajectory is calculated from n points. Triangles defined by these points are used as surfaces for the visualization of the object. Normal vectors can be used to smooth the visualization, and flash effects contribute to a more natural representation.

5.6 Sphered Nerve and Cell Modeling

The proposed algorithm for generating dense WM fiber models can also be applied in dMRI. In dMRI, the motion and interaction of water molecules with the fiber models is simulated. When the water molecules collide with the surface of a fiber segment, the molecular behavior, like diffusion through the surface, must be considered. This applies not only to nerve fibers but also to other cell types. Since the WM consists of axons and other cells such as glial cells, astrocytes, and oligodendrocytes, these also change the dMRI signal.

Therefore, an additional algorithm was developed in collaboration with Kevin Ginsburger and Cyril Poupon (Neurospin, CEA). This algorithm called *Microstructure Environment Designer with Unified Sphere Atoms* (*MEDUSA*) [Gin+19] works similarly to the algorithm described in section 5.3, but models the 3D objects as spheres rather than pipe segments. Such formulation has the advantage that the objects are simpler, which leads to faster collision checking calculations. Additionally, the spheres are also used to represent other types of cells. The overall cell volume is represented by the sum of all spheres corresponding to the cell (see fig. 5.15). In this way, any shape is possible. *MEDUSA* additionally produces oligodendrocytes and astrocytes to generate more realistic WM brain tissue.

Another goal of the *MEDUSA* algorithm is to be able to generate a volume from statistical parameters such as density and angular dispersion [Gin+19; Gin19]. Analogous to the section 5.3 algorithm, the collisions of the objects are checked and slowly pushed apart until no more collisions can be detected. The usage of the statistical parameters allows generating a general purpose library of volumes filled with nerve fibers and cells (see section 5.6.1).

The disadvantage of the spheres is that many more spheres and thus objects for collision control are needed for the representation of tubular objects. Therefore, the algorithm was implemented on the graphics processing unit (GPU) with an axis aligned bounding box search algorithm [Kar12].

5.6.1 Algorithm

Since all objects are represented as a collection of spheres (see fig. 5.15)

$$\mathcal{S} = \{(x_i, y_i, z_i, r_i) : i \in \{0, 1, ..., n_{\text{objects}} - 1\}\} \tag{5.10}$$

a collision occurs when

$$d < r_i + r_j \text{ with } d = |\mathbf{p}_i - \mathbf{p}_j| \tag{5.11}$$

However, since adjacent spheres in a fiber collide when the fiber is densely populated, they must be excluded if the length of the partial trajectory is smaller than the sum of the radii of the two spheres:

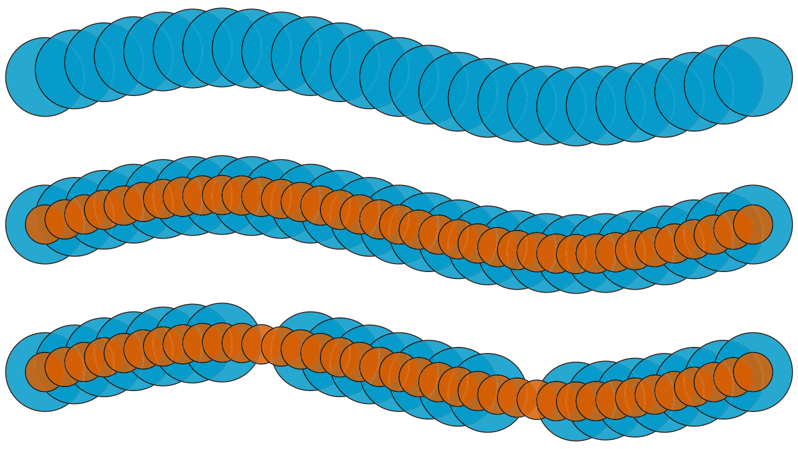
$$\text{ignore if } \sum_{n=i}^{j-1} |\mathbf{p}_n - \mathbf{p}_{n+1}| \leq r_i + r_j \tag{5.12}$$

Spheres inside cell bodies are not checked for collisions because their volume is approximately equal to the volume of the cell.

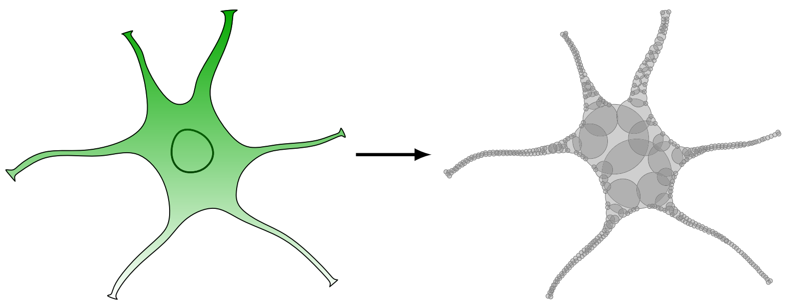
The calculation of the collisions is performed using the GPU architecture. For this first implementation, the algorithm *AxisAligedSortedSearch* [Kar12] is used. It sorts the spheres along an axis and for each sphere i searches for the first and last possible collision on that axis. This results in a list of spheres $\mathcal{C}_i$ to be tested:

$$\mathcal{C}_i = \{s \in \mathcal{S} \mid |s_i.x - s_j.x| < r_i + r_j\} \tag{5.13}$$

The algorithm (see alg. 6) described above is currently used for volumes of about 200 µm for different fiber populations and other properties (see section 5.6.1). For this volume size, the algorithm is fast enough for current use. However, there are more advanced



(a) Modified from [Gin+19]. Outer blue spheres myelin, inner red spheres axon.



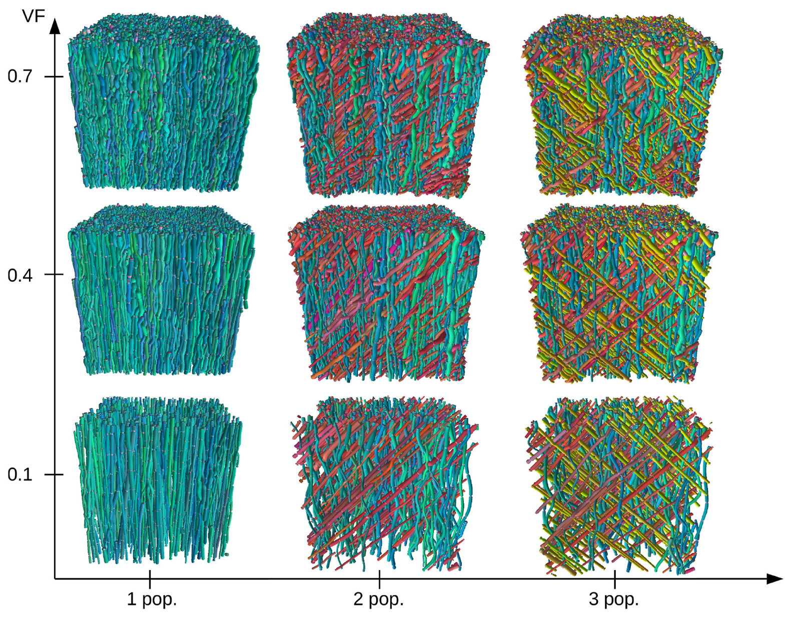
(b) Cell aproximation by overlapping spheres.

Fig. 5.15.: *MEDUSA* sphere approximation.

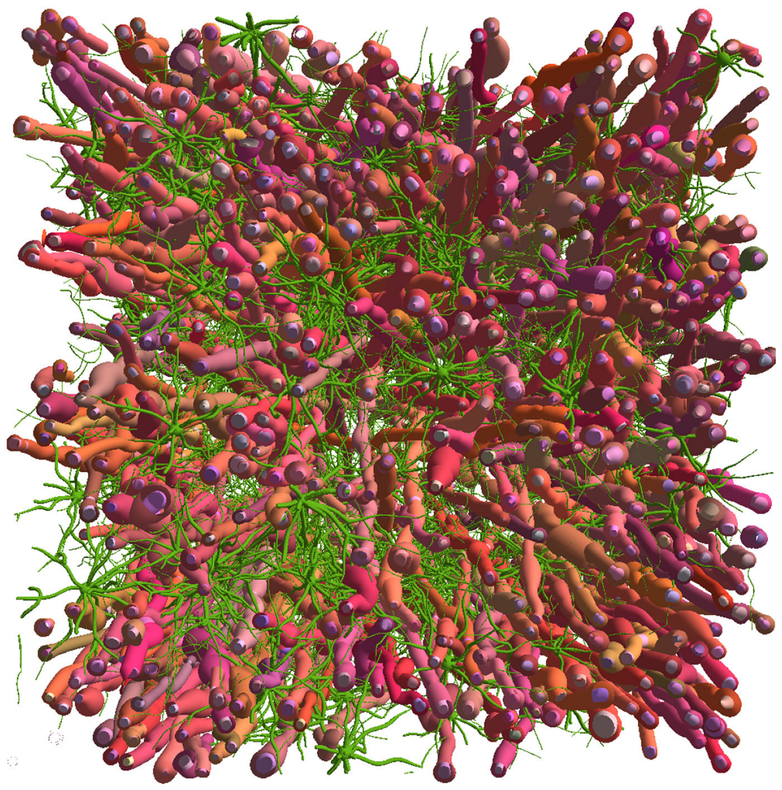
algorithms that can be used here. One promising technique is the use of a *Bounding Box Hierarchy* [Kar12].

```
__global__ void computeRepulsionForces(const Sphere *spheres,
                                       float3 *forces,
                                       const int spheresCount,
                                       const float max_sphere_radius)
{
  int i = blockDim.x * blockIdx.x + threadIdx.x;

  if (i < spheresCount) {
    const Sphere sphere = spheres[i];

    // find first sorted sphere to check
    int s = i;
    while (((sphere.pos.x - max_sphere_radius) <
             (spheres[s].pos.x + max_sphere_radius)) &&
            (s > 0)) {
      s--;
    }
    int s_min = s;

    // find last sphere to check
    s = i;
    while (((sphere.pos.x + max_sphere_radius) >
             (spheres[s].pos.x - max_sphere_radius)) &&
            (s < spheresCount - 1)) {
      s++;
    }
    int s_max = s;

    // check all spheres for collison
    for (int s = s_min; s <= s_max; ++s)
      if (sphere.id != spheres[s].id)
        forces[i] += = computeRepulsionForce(sphere, spheres[s]);
  }
}
```

Alg. 6: *MEDUSA*'s collision checking.



(a) Cubic volumes with different number of fiber populations and volume fractions (VF) are generated.



(b) Exemplary visualization of a volume filled with a single fiber population and astrocytes.

Fig. 5.16.: Original images from [Gin+19].

6 3D-PLI Simulation

Simulations of 3D-PLI have been used to study multiple effects of the microscopic technique involved with brain sections [Doh+15; Men+15; Men+16; Men+20; Men+21; Men14; Men18]. The algorithm presented here for designing new collision-free fiber models allowed to simulate the effect of scattered light in simulations based on a finite-difference time-domain algorithm without superimposed interference signals. These simulations enabled the understanding of scattering effects due to fiber bundle and crossing configurations as well as the transmission change for tilted fiber configurations [Men18; Men+20; Men+21].

The current formulation allows reproducing linear optics simulations coherent with the experimental results of [Doh+15; Men+15; Men+16]. In contrast to [Doh+15; Men+15; Men+16], whose algorithm is computationally and memory intensive due to the pre-computations for discretized tissue volume, this work proposes using the foundations for more efficient parallel computing within a supercomputer architecture presented in [Luc16]. The simulation algorithm was redesigned, considering computational efficiency and the manufacturing design of the new large metripol 3D (LMP3D) microscope. Additionally, the linear optics simulation uses the Müller-Stokes calculus also to take the polarization effects of filters into account.

The 3D-PLI simulation is divided into two sequential parts: the discrete volume generator and the light matter simulation. The discrete volume generator discretizes the virtual nerve fiber models onto a Cartesian grid. Then, the discretized model is used to compute the light matter interaction in the second step. A parallelization technique with *Message Passing Interface* (*MPI*) allows the volume to be partitioned among different CPUs or compute nodes. Due to the tilting approach, the light vector in such a parallelized volume must be able to leave the current volume of a single CPU and traverse to the next volume/process. The computationally intensive algorithms are written in *C++* with an additionally user-friendly designed wrapper function in *Python3*.

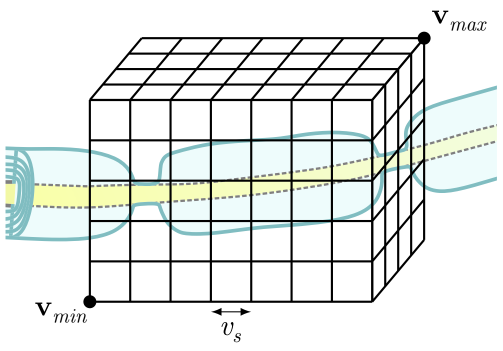


Fig. 6.1.: Discretized tissue volume with a voxel size v_s. The volume is defined by a AABB, which itself is defined by two points $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$.

6.1 Discrete Volume Generator

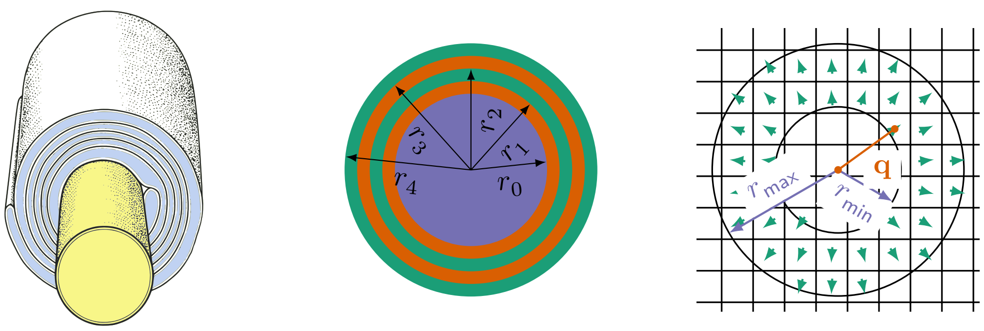
The first part of the 3D-PLI simulation is the calculation of a discretized tissue volume. It represents a discrete, voxel model of the tissue. This helps to drastically speed up the light matter interaction of the next step (see section 6.2), at the cost of a large memory requirement.

The discretized tissue volume represents a cuboid divided into smaller cubes of equal size, called voxels, i. e., 3D pixels (see fig. 6.1). Each voxel contains the physical properties of absorption, birefringence and optic axis orientation of the tissue at its current position. The total volume is bounded by a volume of interest (VOI) defined by a minimum and maximum value: `voi` $= [(x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max})]$. Additionally, the parameter `voxel_size` v_s is set to a floating point number and defines the edge length of the equilateral voxels. In case the division is not integer, the number of voxel of the affected axis is rounded up.

6.1.1 Nerve fiber layers

As described in section 2.3, nerve fibers are axons wrapped by multiple turns of myelin (see fig. 6.2a). Especially for light wave simulations, the myelin windings are an important feature [Men18].

The windings are represented as individual layers (see fig. 6.2b). Which dramatically simplifies the creation process. A layer is defined by a factor between 0 and 1 that scales with the radius of the nerve fiber. For example, 0.75 means that from $0 \leq r < 0.75$ of the radii is interpreted as the first layer (see fig. 6.2b).



(a) Schematic representation of a nerve fiber with axon and myelin sheath.

(b) Cross section through a nerve fiber with layered structure defined by n radii.

(c) Cross section of a discretized nerve fiber with resulting optic axis vectors.

Fig. 6.2.: Discretization of nerve fibers with layered structure.

Each layer requires a number of physical properties in addition to its radius:

- birefringence value: Δn
- absorption coefficient: μ
- optical axis model: $p = parallel$, $r = radial$, $b = background$

The properties are specified as a list of tuples within the algorithm (see alg. 7). Subsequently, the discretized volume generator returns the arrays `tissue`, `optical_axis` and `property_list` for used in the light matter simulation.

```
fbs_properties = [[(r, dn, mu, 'p'), (second layer), ...],
                  [(first layer of second bundle), ...],
                  [...]]
```

Alg. 7: Definition of the properties of fiber bundles.

6.1.2 Discretization of a nerve fiber model

In order to discretize nerve fiber models, nerve fiber segments are discretized individually. Since fibers are a chain of consecutive segments, each voxel inside a fiber segment must be labeled as a tissue with the physical properties at the voxel center position $\mathbf{q}$ (see fig. 6.2). The discretized mesh represents an array, where an element at position $[i, j, k]$ occupies the space from (i, j, k) to $(i+1, j+1, k+1)$ in the unit of $1\,v_s$. To identify all voxels inside a volume, all voxels inside the fiber segments are checked if they are inside the fiber segment. Therefore, a loop over all voxels is performed.

```
for fiber_segment in fiber_bundle:
    for i,j,k in fiber_segment.aabb().voxels():
        min_dist, min_point = calculate_min_distance((i,j,k), cc)
        if min_dist < cc.radius:
            if min_dist < current_distance[i,j,k]:
                optic_axis[i,j,k,:] = get_axis_orientation(
                                            (i,j,k), min_dist,
                                            min_point)
                tissue[i,j,k] = get_layer_id(min_dist)
                current_distance[i,j,k] = min_dist
```

Alg. 8: Pseudocode for filling the discretized volume.

To determine if a voxel is inside the nerve fiber segment, a computation analogously to the collision between two nerve fiber segments is calculated (see alg. 3). The distance vector $\mathbf{d}$ is used to check whether the voxel is inside the fiber segment, and if so, in which layer of the fiber segment it is located, and to calculate the orientation of the birefringence axis.

Two values are stored in the case of a voxel occupied by a fiber segment. The first one is an index within an array *tissue* which will be used later to retrieve the properties from a list with the same index order. The second information is the orientation of the optic axis within the current layer, stored in the array *optic_axis*. The orientation can be parallel to the fiber segment in the case of the macroscopic model or radial in the case of the microscopic model. A layer can also be marked as "background", allowing the user to specify an area without any birefringence.

Two consecutive fiber segments occupy the same space because they have a common point. This is an issue because the processed second fiber segment overwrites the values of the first. This is solved by using an additional array that stores the smallest distance calculated when filling the voxels space. The values are only overwritten when a new distance is calculated that is smaller than the one already stored. This also solves the problem that in a radial optic axis model, the optic axis is star-shaped at the end points of each fiber segment inside the fiber. The first and last point of a fiber, however, are not affected by this.

The algorithm for the discretization loop is shown in alg. 8.

6.1.3 `voxel_size`

The parameter `voxel_size` is an essential property of the light matter simulation. It determines how accurate the volume is represented (see fig. 6.3). Additionally, in

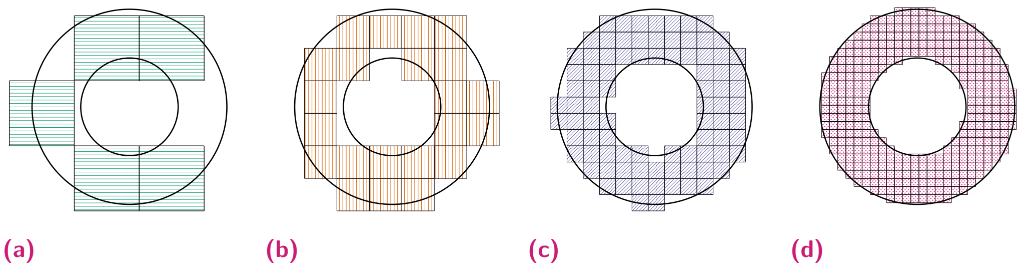


Fig. 6.3.: Discretization error. Cross-section through a single fiber with a myelin layer in the discretized tissue volume. The colored pattern shows the resulting voxels corresponding to the fiber. The smaller the `voxel_size`, the smaller the discretization error.

the light matter simulation one light ray is cast from each voxel of the bottom plane (see section 6.2.1). Therefore, it is also responsible for the sampling of the resulting intensities.

6.1.4 Code optimizations

All arrays are implemented as contiguous c-arrays, accessible externally as *NumPy-arrays* without copying the data. Since these arrays grow with $\mathcal{O}(\frac{1}{v_s}^3)$, the ownership of the data is movable in both the *C++* libraries and *Python3* code. The memory order of the arrays is in the x-y-z direction, so the largest memory shift is in x and the smallest in z. Thus, when the light moves along the z-direction, the information can be read out linearly from the memory, and the acpCPU cache prefetcher can be used effectively.

Two methods are used to parallelize the algorithm on the CPU. The first uses OpenMP to parallelize the filling of the AABB volume of each object. The second uses *MPI* to allow distribution across multiple CPU cores without sharing memory (detailed description in section 6.3.2).

Parallelizing the filling process of the voxels of the discretized volume leads to a race condition when multiple threads want to write or read to the same memory address, i. e., the same coordinate in the volume. A solution with a lock would be suboptimal and since many of the voxels do not need to be overwritten, most of the locks would be unnecessary. To share the work, thread n processes only the memory for the first index i (or volume dimension x) if:

$$i \bmod N_{Threads} == thread_{id} \tag{6.1}$$

Instead of dividing the volume into n subvolumes, each thread loops over every $N_{num\_threads}$ slice of the volume (see fig. 6.4) to distribute the work if the volume

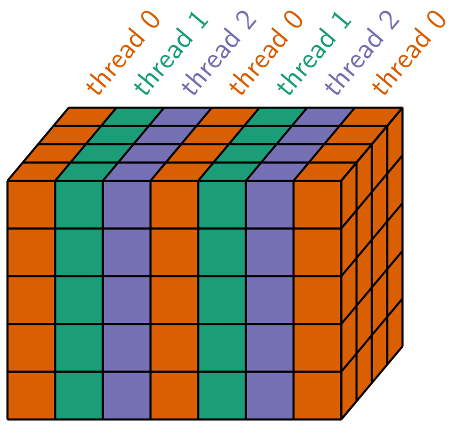


Fig. 6.4.: Discretization volume parallelization with OpenMP. Each thread processes every n-th yz-section. This ensures both thread safety and a more balanced workload.

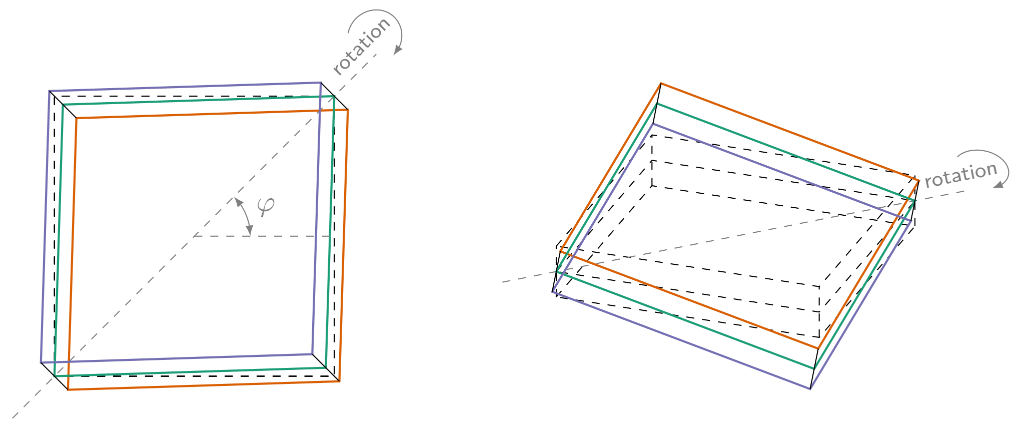
is not filled homogeneously with fibers. This procedure leads to a thread-safe writable operation. One disadvantage of this algorithm is that all threads have to check if the AABB of all fiber segments is inside the current VOI.

6.2 Light Matter Simulation

The light matter simulation algorithm performs the Müller-Stokes calculus (see section 3.2.8) on the previously calculated discrete volume (see section 6.1) for the light rays along their paths. Since no scattering or refraction effects are considered in this simulation, each light path follows a straight line. Initially, the light vector is multiplied by the first polarizer of the optical system (see section 4.3). The path inside the tissue is discretized into steps. The interaction between the light ray and the tissue is calculated after each step according to $\mathbf{S}' = \prod_i \left(\boldsymbol{\mathcal{R}}_i \boldsymbol{\mathcal{M}}_i \boldsymbol{\mathcal{R}}_i^{-1}\right) \cdot \mathbf{S}$ (see section 3.2.8).

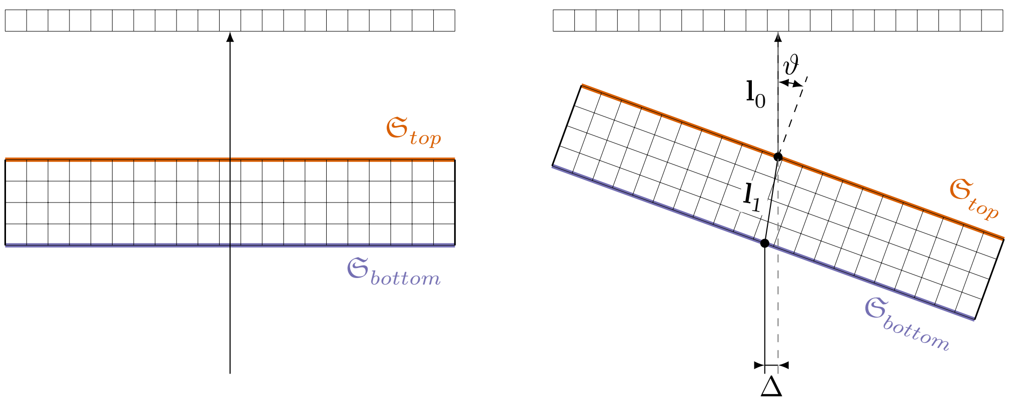
6.2.1 Light ray path

The light matter simulation allows for a tilted light beam. For this purpose, the large-area polarimeter (LAP) uses a tilting stage to which the tissue sections are attached (see fig. 6.5). The LMP3D, on the other hand, has a tilted light beam. This is achieved by a conical light path, from which an aperture is then used to sample the desired light direction [Wie16]. Both methods can be mathematically represented by the same procedure.



(a) camera view **(b)** perspective view

Fig. 6.5.: 3D tilting: around xy-axis, — top, — middle, — bottom, ··· original, — axis of rotation.

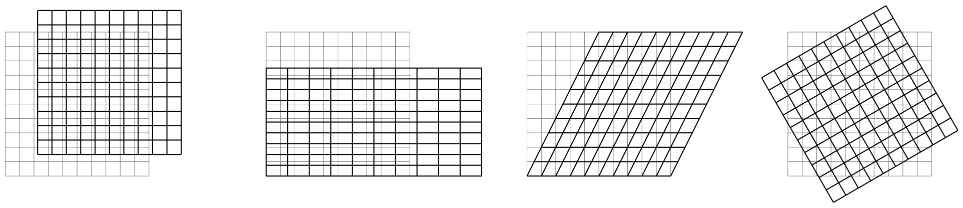


(a) normal **(b)** tilted

Fig. 6.6.: Light ray path for a normal (a) and a tilted (b) case. In the tilted case, the light beam $\mathbf{l}_1$ is tilted within the tissue and thus experiences an optical shift Δ.

An additional effect changing the light path is the refraction at the tissue-air boundary, which is described by Snell's law for isotropic media (see eq. (3.10)). Since this only adds a parallel shift, simulation is only necessary when the effects of the resampling process and image registration is investigated.

The initial position of the light beam is calculated by traversing the light path backwards (see fig. 6.6). This has the advantage that each index inside the charge-coupled device (CCD) always receives exactly one light beam. From the CCD array, the light path can be shifted back to the top plane of the tissue $\mathfrak{S}_{top}$. Subsequently, the light beam $\mathbf{l}_1$ is traced back through the tissue to the bottom plane $\mathfrak{S}_{bottom}$. The point on the lower tissue plane corresponds to the initial position of the light beam. This light path change



(a) translation (b) scaling (c) shears (d) rotation

Fig. 6.7.: Examples of affine transformations.

results in a shift δ along the same direction of the tilting. In the light matter simulation, only light rays are considered, which will go at least partially through the volume. All remaining CCD elements are left with a `NaN` value.

The tilting of the tissue leads to a distortion of the image (see fig. 6.5). This distortion can be described by an affine transformation (see fig. 6.7):

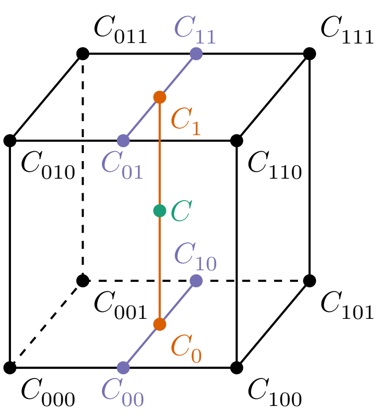
$$f(\mathbf{x}) = \boldsymbol{\mathcal{A}} \cdot \mathbf{x} + \mathbf{t} \tag{6.2}$$

where $\mathbf{x}$ is the coordinate input, $\boldsymbol{\mathcal{A}}$ and $\mathbf{t}$ are the transformation values, and $f(\mathbf{x})$ is the transformed coordinate.

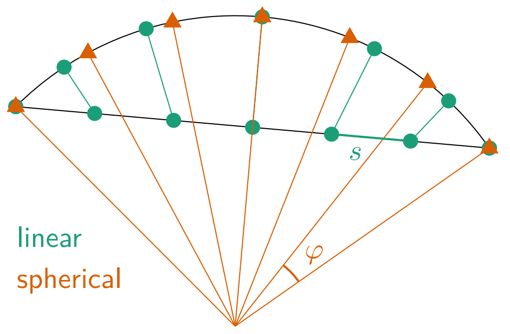
The light matter simulation with the light paths sampling as described above takes this distorted view into account and removes it. The simulation can also sample the light rays so that distortion occurs. Thus, the resulting images have to be registered onto each other by an affine transformation which is also available in the algorithm.

6.2.2 Tissue voxel interpolation

If the `step_size` of the light ray is not equal to the `voxel_size` or if the light path is tilted, the light ray position after a step no longer matches the center of the voxel. In order to correctly recreate the physical properties, interpolation is required (see fig. 6.8). Currently, three interpolation methods are implemented: *nearest neighbor*, *linear interpolation* and *spherical interpolation*. The voxels considered for interpolation are the adjacent eight neighboring voxels, i. e., array indices $(\lfloor x \pm 0.5 \rfloor, \lfloor y \pm 0.5 \rfloor, \lfloor y \pm 0.5 \rfloor)$. The *nearest neighbor* and *linear interpolation* are still present from the development phase. However, they should not be used because they are prone to errors, given that the data represent orientations in the space. Thus, the use of *spherical interpolation* considers the relationship existing in the data components, and it is highly recommended.



(a) trilinear interpolation



(b) spherical interpolation

Fig. 6.8.: Interpolation techniques: Trilinear interpolation can be represented as an axial step interpolation. The difference between linear and spherical interpolation is that linear interpolation has a constant distance s between each point, while spherical interpolation has a constant angle φ between two steps.

```
light_beams = calculate_light_starting_positions()
for light in light_beams:
    light = optical_elements_start * light
    while light.pos in volume:
        properties = get_properties(light.pos)
        light.intensity *= exp(-step_size * properties.absorbtion)
        light = matrix(properties) * light
        light.pos += step_size

    light = optical_elements_end * light
    ccd_array[light.ccd_pos] = light.intensity
```

Alg. 9: Loop over the light vectors for the light-tissue interaction. Their intensity value is stored inside the CCD array.

6.2.3 Simulation of light matter interaction

After the initial light positions are calculated, each light beam is traversed within the tissue. To calculate the light-tissue interaction, the change of the Stokes vector is calculated after each light beam step. The step size is a user defined value, with a default value of the `voxel_size` v_s. To simulate the light beam hitting the volume's boundary, the light beam is multiplied by the matrix of the remaining optical elements of the microscope and the intensity is stored in the CCD-array. The pseudocode is shown in alg. 9. To resample the array to the final size of the CCD sensor and apply the noise model, there are *Python3* functions outside the simulation software available.

6.2.4 Optical system and signal analysis

The image sensor, as described in section 4.3, is a CCD-sensor. The resampling and noise modeling, are implemented according to section 4.6. These calculations are performed on the *Python3* side of the algorithm (`fastpli.simulation.Simpli`) and can be executed with the `multiprocessing` library of *Python3* to use multiple CPU cores. Therefore, when using *MPI* (see section 6.3.2), the intensity image must be reduced to a single process. Since the resulting image is only 2D, there is no need to further speed up the process with *MPI*.

To analyze the signal, the same algorithms are implemented as in the 3D-PLI routine pipeline (see sections 4.4 and 4.5). These include the modalities analysis transmittance, direction and retardation and the tilt analysis performed by *robust orientation fitting via least squares* (*ROFL*). The tilt analysis is a fork of the existing routine analysis developed in python. To parallelize the execution, the native `multiprocessing` library of *Python3* can be used.

6.3 Speedup Strategies

6.3.1 Code design

The following key optimizations are used in the pipeline:

- The order of the tissues stored in memory is along the z-axis (as described in section 6.1.4) so that the light ray *traverses* along the aligned memory.
- The for-loop for the light beams (see alg. 9) is parallelized with OpenMP. These threads are completely separated and there are no race conditions.
- Vector and matrix calculations are optimized for their small sizes by the compiler with the help of tools like *Compiler Explorer*[1] and *C++ Insight*[2].

6.3.2 MPI parallelization

The algorithms for discrete volume generation and light matter simulation can additionally use a parallelization technique. The computation must be split among multiple physical CPUs and nodes for large volumes larger than the local memory size. For this purpose, *Message Passing Interface* (*MPI*) is used. A method is implemented to automatically

[1] `https://godbolt.org/`
[2] `https://cppinsights.io/`

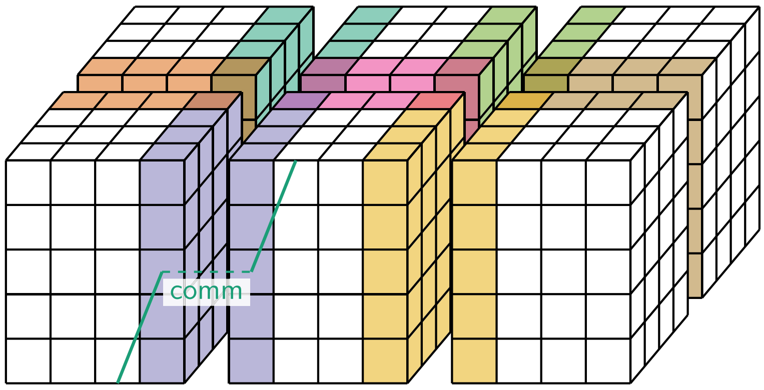


Fig. 6.9.: This example uses six *MPI* ranks to split the entire volume into six subvolumes. To calculate the value at a given point a eight-neighborhood is necessary. Therefore a *halo* area (coloured voxels) with the same information shared by neighboring *MPI* process is necessary.

partition the volume along the x-axis and the y-axis into blocks with the minimal surface area along both axis (see fig. 6.9).

Each CPU can perform the discrete volume generation without the knowledge of each other. The exception is the light matter simulation for tilted light beams. Here, when a light ray leaves the local volume, it must be transmitted to the adjacent volume. *MPI* provides several methods to send information to another *rank*. Since the subvolumes are split along a Cartesian grid, the Cartesian implementations of *MPI* are used (e. g. `MPI_Cart_create`). A problem is the calculation at the edge of the volume, since the light beam needs the information of the surrounding eight voxels for the interpolation. If this voxel information were to be transmitted to the neighboring *rank*, this would mean a large amount of communication, which is much slower compared to the local CPU or random access memory (RAM) instructions. To solve this communication problem, the algorithm uses a *halo*. This is a commonly used concept where the boundaries (in this case the volume) are increased by a certain size so that the same information about the shared regions is available everywhere (see fig. 6.9). Such approach reduces the number of necessary communications between different ranks (see fig. 6.9).

To further speed up the communication process, all outgoing light beams are first stored locally in a communication buffer, and only after all local light beams have been processed the buffer is passed to the neighbors. This ensures minimal communication overhead. The main loop of the light beam algorithm is then restarted on all *MPI ranks* for the communicated light beams. This is repeated until no more communication is required.

Each *MPI* process can additionally use OpenMP to allow multiple cores to benefit from shared memory.

7 *fastPLI*

7.1 Introduction

The previous chapters described the algorithms for creating dense WM fiber models (see chapter 5) and light matter simulation of 3D-PLI (see chapter 6). Both algorithms operate independently, simplifying the use of the algorithms for other domains. For example, the fiber models can be used in dMRI as well ([Gin+19; Gin19]). These implemented algorithms were developed with a focus on computational efficiency and usability. Therefore, an application programming interface (API) is available to provide a friendly user interface. Furthermore, the implementation provides a high level of abstraction within an easily installable framework.

In order to accomplish the usability goals, the *Python3* programming language was chosen as the user interface. Meanwhile, to boost the performance of the optimization steps and heavy computation *C++* was used as described in sections 5.4.4 and 6.3 to ensure efficiency and parallelization.

7.2 fastPLI Toolbox

The here designed *Python3* package is called *fiber architecture simulation toolbox for 3D-PLI* (*fastPLI*). Its source code is publicly available and reviewed in the Journal of Open Source Software (JOSS) [@Mat21; Mat+21]. The software package includes functionalities for analysis and visualization of nerve fiber models as well as for evaluation of simulation analogous to current routine experimental measurements, e. g. tilt analysis (see fig. 7.1).

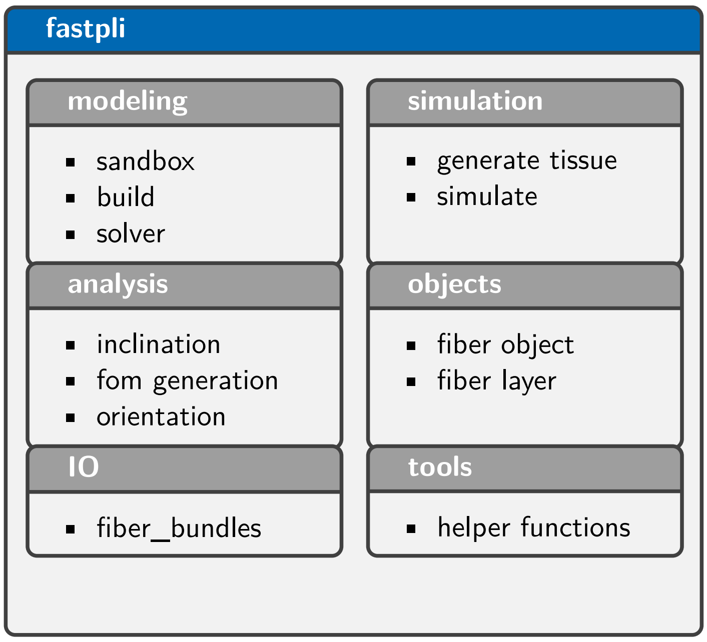


Fig. 7.1.: *fastPLI* package structure.

7.2.1 Dependencies

Python:

NumPy: Base N-dimensional array package [@Vir+19]
`https://numpy.org/`

scipy: Fundamental library for scientific computing [@Vir+19]
`https://www.scipy.org/`

numba: Acceleration of Python Functions [@Lam+15]
`https://numba.pydata.org/`

mpi4py: MPI for Python [@Dal+05; @Dal+08; @Dal+11]
`https://bitbucket.org/mpi4py/mpi4py/src/master/`

h5py: HDF5 for Python [@Col13; @The97]
`https://www.h5py.org/`

C++:

MPI: Message Passing Interface [@For15]
`https://www.mpi-forum.org/`

OpenMP: Open Multi-Processing, API for multi-platform shared memory multiprocessing programming [@Dag+98]
`https://www.openmp.org/`

OpenGL: Open Graphics Library [@Wik18]
`www.opengl.org`

Pybind11: Seamless operability between C++11 and Python [@Jak+17]
`https://github.com/pybind/pybind11`

7.2.2 Installation

The installation instructions are scripted in a `Makefile`. It first starts a *CMake* routine, which searches for all the required libraries and programs. Then the *C++* code is compiled and the resulting *shared object libraries* are stored in the *Python3* routines. Finally, the provided code `setup.py` allows the user to install the compiled package in his environment:

```
git clone --recursive https://github.com/3d-pli/fastpli.git
cd fastpli
make fastpli
pip3 install .
```

Alg. 10: Installation instructions.

7.2.3 Tests, verification and issue tracking

In order to ensure the continuous integration and continuous deployment of the *fastPLI* project, the CI/CD actions were set up within the *Github* repository. The action runs the two latest Ubuntu Long Term Support versions (currently 18.04 LTS and 20.04 LTS) and the most commonly used Python3 versions (currently 3.6 and 3.8) to provide a wide range of standard supported versions.

In addition, the *Github actions* run all test scripts, check tutorial files, check code format and linting for consistency, and publish the latest documentation after a successfully tested release.

Github allows tracking issues. This feature is originally used to document software bugs. However, it is also used to discuss ideas and new features. As part of the open-source release, it was also used to communicate with the reviewers, allowing the development process tracking. [1]

7.3 Modules

The following section describes the different modules that comprise the fig. 7.1 *Python3* package. The modules are listed alphabetically.

7.3.1 `fastpli.analysis`

This module contains all functionalities to analyze the 3D-PLI simulations analogous to the routine measurements, which includes the analysis of the signal to the three image modalities transmission, direction and retardation. Furthermore, it provides the tilt analysis *ROFL* [Sch+18b]. In addition, other helper functions exist that provide methods to convert the direction and tilt results into a fiber orientation map (FOM).

For fiber model analysis, the module provides a few simple helper functions. For example, it allows the user to generate a histogram of the orientations of the fiber segments, like the ones shown in this thesis.

7.3.2 `fastpli.io`

This method provides the read-write routines that allow users to load and save fiber models (i. e., `fiber_bundles`) to or from disk. There are two formats available. The first is a text file with the extension `.dat` (see alg. 11). Here, each (x, y, z, r) tuple of a fiber point is stored as a single line in the file. Two fibers are separated by one blank line, while two blank lines separate two fiber bundles. This data format is provided to allow a straightforward format for manipulating, exchanging and reading the files into other programs.

[1] https://github.com/openjournals/joss-reviews/issues/3042

```
-6.55 -18.93 -64.98 3.75 # x y z r
-5.73 -14.89 -63.37 3.4
-4.42 -13.66 -58.95 3.05
                         # empty line indicates new fiber
-1.96 -10.07 -52.5 2.92
-1.03 -9.4 -48.62 2.93
                         # two empty lines indicates new fiber bundle
3.4 -4.02 -44.76 3.11
6.22 -1.04 -42.45 3.26
```

Alg. 11: Exemplary *.dat* file format.

The second format uses *Hierarchical Data Format v5* (*HDF5*) [@The97], a binary data format. *HDF5* stores the data as *datasets* in *groups*, analogous to a file in an operating system stored in folders. The *HDF5-groups* are used to store the `fiber` in `fiber_bundle` and `fiber_bundles`. The (x, y, z, r) information of each fiber is stored as a 2d-array (see alg. 12).

7.3.3 `fastpli.model.sandbox`

The sandbox module provides all the functions described in section 5.2. The module is divided into two submodules: `fastpli.sandbox.build` and `fastpli.sandbox.seeds`.

`fastpli.sandbox.seeds` contain all the methods for populating a 2d-plane, as described in section 5.2.1. The sandbox .build module provides the methods to populate the fiber from the seeds, which includes all the described functions from section 5.2.2.

7.3.4 `fastpli.model.solver`

The module `fastpli.model.solver` contains the compiled solver algorithm, explained in detail in sections 5.3 to 5.4.4. The solver algorithm is wrapped in the `fastpli.model.solver.Solver` class. This wrapper class provides a higher level of abstraction (see section 7.1). All variables are read and writable by attributes, e. g. `Solver.obj_mean_length`. Each attribute checks if the user input is valid and returns an appropriate warning or error message if necessary. This class also includes a `Solver.get_dict()` method that returns a *Python3* dictionary containing all variables and their values for reproducibility. It is also possible to store the state of the class with the current state of the `fiber_bundles` as an *HDF5* object. Finally, this class also provides the possibility to use a simple visualization of the solver process (see section 5.5).

```
GROUP "/" { # fiber_bundles path
  GROUP "0" { # id of fiber_bundle
      DATASET "0" { # id of fiber
         DATATYPE  H5T_IEEE_F64LE
         DATASPACE  SIMPLE { ( 3, 4 ) / ( 3, 4 ) }
         DATA {
         (0,0): -6.55, -18.93, -64.98, 3.75,
         (1,0): -5.73, -14.89, -63.37, 3.4,
         (2,0): -4.42, -13.66, -58.95, 3.05,
         }
      }
      DATASET "1" { # id of fiber
         DATATYPE  H5T_IEEE_F64LE
         DATASPACE  SIMPLE { ( 2, 4 ) / ( 2, 4 ) }
         DATA {
         (0,0): -1.96, -10.07, -52.5, 2.92,
         (1,0): -1.03, -9.4, -48.62, 2.93,
         }
      }
  }
  GROUP "1" { # id of fiber_bundle
      DATASET "0" { # id of fiber
         DATATYPE  H5T_IEEE_F64LE
         DATASPACE  SIMPLE { ( 2, 4 ) / ( 2, 4 ) }
         DATA {
         (0,0): 3.4, -4.02, -44.76, 3.11,
         (1,0): 6.22, -1.04, -42.45, 3.26,
         }
      }
  }
}
```

Alg. 12: Example structure of the fiber format in *HDF5*. This output is generated with the official `h5dump` tool.

7.3.5 `fastpli.objects`

This module provides a wrapper class for `fastpli.objects.fibers` and `fastpli.objects.layers`. Layers storing the information about the nerve fiber bundles myelin sheet. [2] Essentially, `layers` are a `list` of `layer`, which are a `tuple` of the four attributes `absorption`, `birefringence`, `model`, and `scale` (see section 6.1). This wrapper class contains attributes that allow users to access these values by name.

The class `fastpli.objects.Fiber` stores the 4D points of a nerve fiber as `numpy.ndarray`. Multiple nerve fibers can be grouped into a `list` represented by the class `fastpli.objects.FiberBundle`. Similarly, a fiber bundle is represented by a list of `fastpli.objects.FiberBundle` in the class `fastpli.objects.Fiber`.

[2]Myelin does not consist of individual layers, but of a single one that wraps the axon several times. However, the layered structure allows the structure to be defined in a simple way, with fewer parameters.

Each class provides member functions to `translate`, `rotate`, `scale`, and `cut` the model. This allows users to manipulate and place the objects in space.

7.3.6 `fastpli.simulation`

The `fastpli.simulation` module provides a wrapper for the simulation called `fastpli.simulation.Simpli`, based on the original algorithm [Doh+15; Luc16]. It contains the two algorithms `generator` and `simulation` described in sections 6.1 and 6.2. These two algorithms operate separately, but they coexist within the class since they share several parameters. As in the `fastpli.model.solver.Solver` class, all necessary attributes are available and checked for input errors. Since analysis is always performed on the resulting simulations, they are also available in this class and are performed with the same defined parameters as in the simulation. Methods for saving the variables as `dict` or *HDF5* files are available as well.

The simulation is performed with the same procedure used with the real measurements, using four tilt directions and analyzing the simulated data. `Pipeline` methods exist (see alg. 13), which provide a high level of abstraction to simplify the execution for the users.

7.3.7 `fastpli.tools`

The last module contains a set of helper functions. They provide access to the current version and to the git hash so that all calculations can be reproduced. For fiber modeling, rotation matrices are provided to allow the use of linear algebra.

7.4 Computational Speedup Techniques

Among other specific techniques described in chapters 5 and 6, two essential techniques are used to speed up the calculations.

The computationally intensive code is written in *C++*. There, the `std::vector` has the advantage that the data in memory is linear. The data must be prepared and sent from the RAM to the cache of the CPUs. This takes a relatively long time for a single CPU instruction. The main advantage of the cache is that it is very fast, however its capacities are usually a few MB and quite limited. It is built inside the CPU. The CPU cache prefetcher is a sophisticated directive that requests the element at address i in

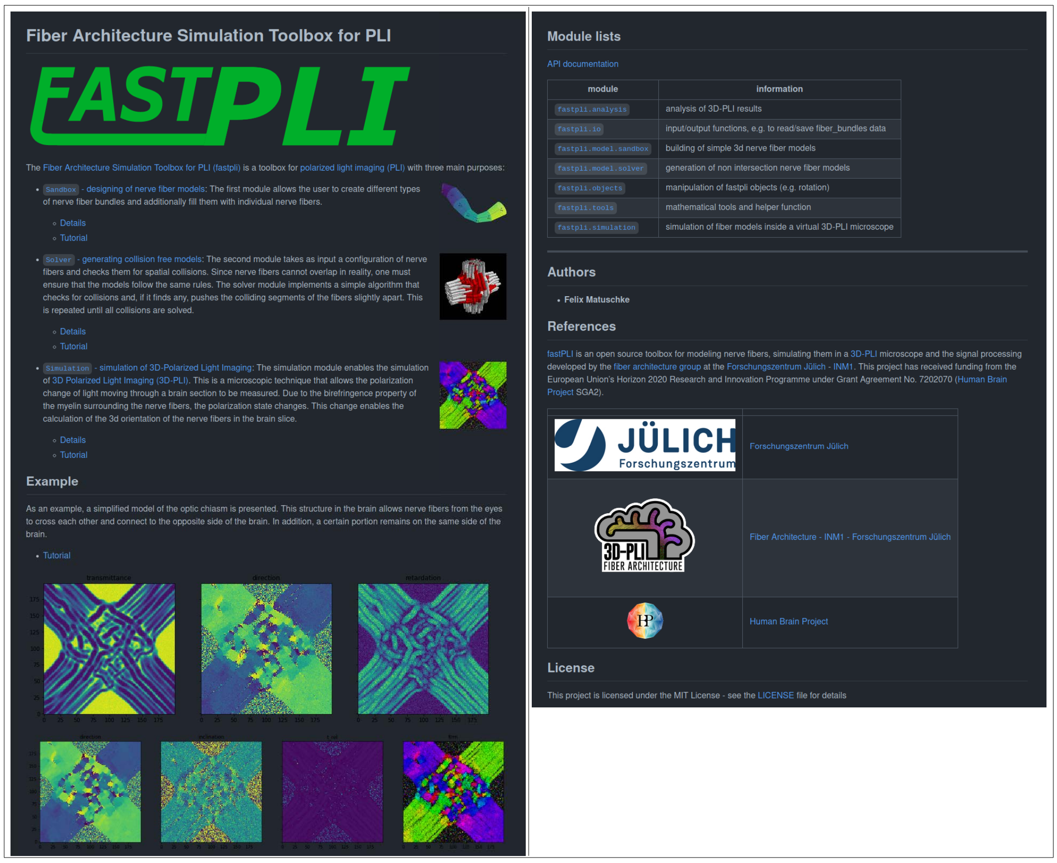
```python
import numpy as np
import h5py

import fastpli.simulation
import fastpli.io

# Setup Simpli for Tissue Generation
simpli = fastpli.simulation.Simpli()
simpli.omp_num_threads = 2

# define model
simpli.voxel_size = 2.0  # in micrometer
simpli.set_voi([-100, -100, -25], [2350, 550, 25])  # in micrometer
simpli.fiber_bundles = fastpli.io.fiber_bundles.load('model.dat')

# define layers (e.g. axon, myelin)
simpli.fiber_bundles_properties = [[(1.0, -0.001, 10, 'p')]]
# (_0, _1, _2, _3)
# _0: layer_scale times radius
# _1: strength of birefringence
# _2: absorption coefficient I = I*exp(-mu*x)
# _3: model: 'p'-parallel, 'r'-radial or 'b'-background

# define pli setup
simpli.filter_rotations = np.deg2rad([0, 30, 60, 90, 120, 150])  # in deg
simpli.light_intensity = 26000  # a.u.
simpli.interpolate = "Slerp"
simpli.wavelength = 525  # in nanometer
simpli.pixel_size = 10  # in micrometer
simpli.tilts = np.deg2rad(
    np.array([(0, 0), (5.5, 0), (5.5, 90),
              (5.5, 180), (5.5, 270)]))  # in deg
simpli.optical_sigma = 0.75  # in pixel size
simpli.noise_model =
    lambda x: np.random.negative_binomial(x / (3 - 1), 1 / 3)

with h5py.File('output.h5', 'w') as h5f:
    with open(os.path.abspath(__file__), 'r') as script:
        tissue, simulation, rofl, fom = simpli.run_pipeline(h5f=h5f,
                                                   script=script.read(),
                                                   save=['all'],
                                                   crop_tilt=True,
                                                   mp_pool=pool)
```

Alg. 13: Simulation pipeline `simpli.run_pipeline`.

memory and the elements next to it ($i+1$ or $i-1$, depending on the algorithm). Since many algorithms traverse arrays, the next element to be computed is typically the next (or previous) element. Therefore, the total time to copy the data from the memory to the cache is reduced. For linear operations on memory, the cache prefetcher reduces the time so much that it behaves as if the CPU had an infinite cache.

Another technique is to use modern compilers such as *Clang v11*[3] or *G++ v10*[4]. They use built-in algorithms to optimize the code for the machine's architecture and much more sophisticated methods. For example, if the number of iterations is known at compile time, a for loop can be *unrolled* to speed up the computations since it no longer needs to check if the conditions are met at the end of each loop cycle. To review these

[3]https://clang.llvm.org/
[4]https://gcc.gnu.org/



Fiber Architecture Simulation Toolbox for PLI

FASTPLI

The Fiber Architecture Simulation Toolbox for PLI (fastpli) is a toolbox for polarized light imaging (PLI) with three main purposes:

- `Sandbox` - designing of nerve fiber models: The first module allows the user to create different types of nerve fiber bundles and additionally fill them with individual nerve fibers.
  - Details
  - Tutorial
- `Solver` - generating collision free models: The second module takes as input a configuration of nerve fibers and checks them for spatial collisions. Since nerve fibers cannot overlap in reality, one must ensure that the models follow the same rules. The solver module implements a simple algorithm that checks for collisions and, if it finds any, pushes the colliding segments of the fibers slightly apart. This is repeated until all collisions are solved.
  - Details
  - Tutorial
- `Simulation` - simulation of 3D-Polarized Light Imaging: The simulation module enables the simulation of 3D Polarized Light Imaging (3D-PLI). This is a microscopic technique that allows the polarization change of light moving through a brain section to be measured. Due to the birefringence property of the myelin surrounding the nerve fibers, the polarization state changes. This change enables the calculation of the 3d orientation of the nerve fibers in the brain slice.
  - Details
  - Tutorial

Example

As an example, a simplified model of the optic chiasm is presented. This structure in the brain allows nerve fibers from the eyes to cross each other and connect to the opposite side of the brain. In addition, a certain portion remains on the same side of the brain.

- Tutorial

Module lists

API documentation

module	information
`fastpli.analysis`	analysis of 3D-PLI results
`fastpli.io`	input/output functions, e.g. to read/save fiber_bundles data
`fastpli.model.sandbox`	building of simple 3d nerve fiber models
`fastpli.model.solver`	generation of non intersection nerve fiber models
`fastpli.objects`	manipulation of fastpli objects (e.g. rotation)
`fastpli.tools`	mathematical tools and helper function
`fastpli.simulation`	simulation of fiber models inside a virtual 3D-PLI microscope

Authors

- **Felix Matuschke**

References

fastPLI is an open source toolbox for modeling nerve fibers, simulating them in a 3D-PLI microscope and the signal processing developed by the fiber architecture group at the Forschungszentrum Jülich - INM1. This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 7202070 (Human Brain Project SGA2).

JÜLICH Forschungszentrum	Forschungszentrum Jülich
3D-PLI FIBER ARCHITECTURE	Fiber Architecture - INM1 - Forschungszentrum Jülich
HP	Human Brain Project

License

This project is licensed under the MIT License - see the LICENSE file for details

Fig. 7.2.: Documentation wiki page of the *Github* repository `https://github.com/3d-pli/fastpli/wiki`.

optimizations, the time critical code was tested with tools like *Compiler Explorer*[5] and *C++ Insight*[6].

7.5 Documentation

All methods contain documentation strings (docstrings) which modern editors automatically display during programming. These docstrings are also used for an automatic release of the API documentation and wiki pages(see fig. 7.2). [7] The wiki is an essential part of the review process for release in JOSS [Mat+21], and it is structured as a guide that walks through the aspects of designing nerve fiber models, applying the collision solver algorithm, visualizing nerve fibers, an introduction in 3D-PLI, and finally applying the models in the simulation.

[5] `https://godbolt.org/`
[6] `https://cppinsights.io/`
[7] `https://3d-pli.github.io/fastpli/`

Both executable *Python3* scripts and Jupyter notebooks are provided examples for a friendly user familiarization with the tool. A nerve fiber crossing is presented as a general example using all presented methods. The example is based on the optic chiasm's anatomy, which is the nerve fiber pathway from the eyes to the occipital lobe.

Part III

Software Application and Evaluation

8 Dense Nerve Fiber Modeling

8.1 Introduction

In the previous chapter chapter 5, the module `fastpli.model` was described for creating non-colliding nerve fiber models. An important question is how the software parameters affect the resulting models.

Three conditions are required for the software to be applicable. First, it must be possible to create a dense volume. This, of course, depends on the original fiber configuration. For example, the density is lower for configurations such as crossings. The second requirement is the runtime. The lower the runtime, the more objects and larger volumes the user can create in the same time. The last requirement is that the original fiber orientations remain intact. Because of the motion phase in the collision solver algorithm, the orientations will change. The question is how much they change and whether the results still meet the user's expectations and anatomical constraints.

To study the behavior of the software, a general data set with simplistic model parameters is needed. These parameters should be able to define a fiber configuration in a volume without having to describe each individual fiber. This description could then be used for both the study of the solver parameters and the subsequent study of the 3D-PLI simulation.

The first part of this chapter is the presentation of the parameterization used to describe the fiber populations. This is followed by an investigation of the collision solver parameters on the behavior of the resulting collision-free nerve fiber models. On this basis, a set of software parameters is determined to create a library of nerve fiber models for 3D-PLI simulations. In the last part, the orientation distribution of the generated models is investigated.

8.2 Designing Fiber Populations

For the 3D-PLI simulation, a volume of size 65 µm × 65 µm × 60 µm is required (see chapter 9). [1] In this work, up to two fiber populations are studied. A fiber population is a fiber bundle with a specific orientation, density, and radius distribution. These parameters of a fiber population have the advantage of being anatomically motivated and describe a nerve fiber volume based on statistical properties. This is for example also used in diffusion Magnetic Resonance Imaging (dMRI) simulations [Gin+18; Gin+19; Gin19]. The number of fiber populations is limited to two to reduce the degrees of freedom within the models.

8.2.1 Orientation and proportion

For the study of a single fiber population, only one model is needed, since the model can be rotated to any 3D orientation. For two fiber populations, the crossing angle θ between the two fiber population orientations is relevant (see fig. 8.1a). By introducing a rotation $R_{\mathcal{F}_1}$ along the first fiber population, the orientation of the second fiber population can be changed (see fig. 8.1b). If the first fiber population is tilted by an angle $\alpha_{\mathcal{F}_0}$, an arbitrary fiber configuration with two fiber populations can be described (see fig. 8.1c).

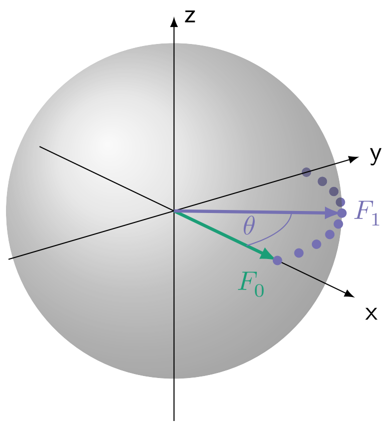
A further parameter is the first population proportion:

$$\Psi_{\mathcal{F}_0} = \frac{N_0}{N_0 + N_1} \tag{8.1}$$

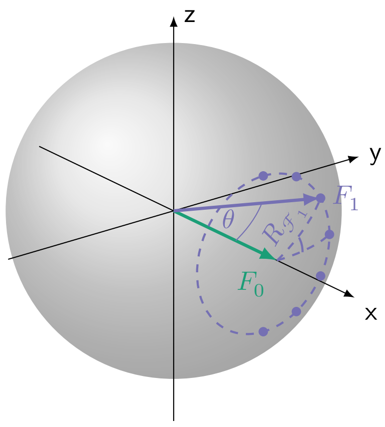
This can range between 0 and 1. Since only dense white matter (WM) phantoms are generated in this study, a lower overall density of 1 is not investigated.

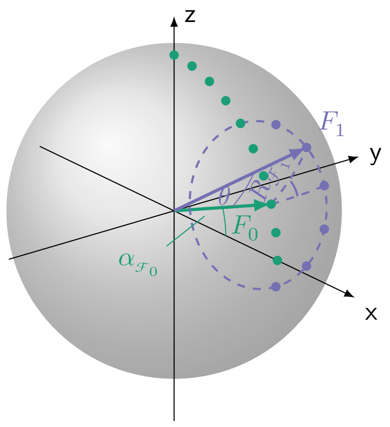
To reduce the number of models required, the parameters are sampled. Ideally, the surface of a sphere is equidistantly sampled. However, this is impossible. There are fairly good approximations, but one would still have to create a model for each sampled point. For simplicity, the intersection angle θ is sampled instead. By rotating the model with $R_{\mathcal{F}_1}$ along the first fiber population axis for all sampled crossing angles θ, one can sample the entire sphere.

[1] 65 µm is divisible by 1.3 µm, the pixel size of the microscope under study.



(a) Initial orientation for two fiber populations $\mathcal{F}_0$ and $\mathcal{F}_1$. The angle between the two populations is θ. The remaining degrees of freedom are accounted for by rotating the entire volume.





(b) The first fiber population is stationary. The second fiber population is described by the opening angle θ and a rotation $R_{\mathcal{F}_1}$ around the first fiber population.

(c) The inclination of the first fiber population is modified by $\alpha_{\mathcal{F}_0}$.

Fig. 8.1.: Parameterization of two mutually relatively oriented fiber populations for the creation of a library.

Therefore, the following parameter samples are chosen:

$$\begin{aligned} \theta &= \{10°, 20°, ..., 90°\} \\ \Psi_{\mathcal{F}_0} &= \{10\,\%, 20\,\%, ..., 90\,\%\} \end{aligned} \tag{8.2}$$

In the case of $\theta = 0°$, $\Psi_{\mathcal{F}_0} = 0$ or $\Psi_{\mathcal{F}_0} = 1$, no second unique orientation exists, so the single fiber population model can be applied. This results in a total number of 82 models that are required.

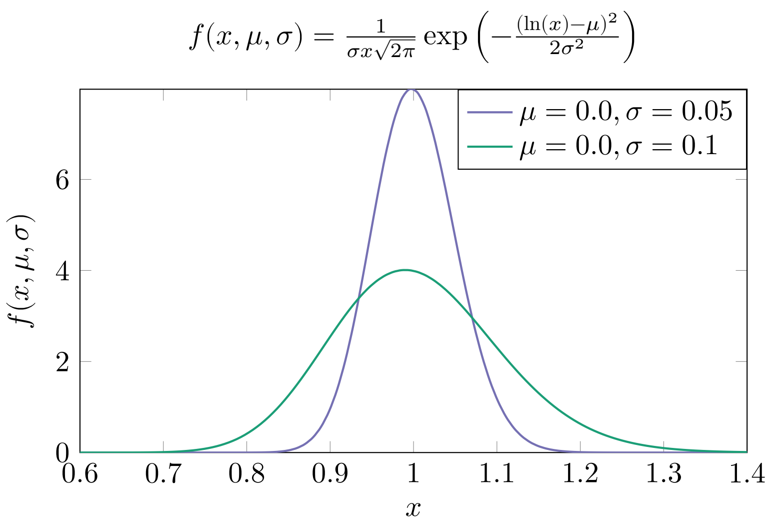


Fig. 8.2.: Probability density function of a multiplicative *log normal* distribution. $\sigma = 0.1$ is used for the initial variation of each fiber radius. $\sigma = 0.05$ for the variation along the fiber for each fiber point.

8.2.2 Fiber placement

To design the individual fiber configurations for each fiber population, the methods described in section 5.2 are used, i. e., the generation of fiber bundles with seed points. Seed points on a 2D plane are generated with a uniform distribution:

$$p_i = (\mathrm{Uniform}(-\frac{1}{2}L, \frac{1}{2}L), \mathrm{Uniform}(-\frac{1}{2}L, \frac{1}{2}L)) \tag{8.3}$$

Since the simulation volumes are cubic and the model is rotated, a spherical boundary with diameter $d_{sphere} = \sqrt{3} \cdot L$ is picked, where $L = 65\,\mu\text{m}$ is the length of the largest edge length of the cube. To ensure that objects exist outside the simulation volume that can build up pressure on the interior, L is increased by two fiber diameters. This also prevents edge effects in the simulation. The number of seed points is set to a value of

$$N_{0,seeds} = \mathrm{round}\left(\Psi_{\mathcal{F}_0} \frac{A_\square}{\pi \cdot \bar{r}_f}\right) \tag{8.4}$$

$$N_{1,seeds} = \mathrm{round}\left((1 - \Psi_{\mathcal{F}_0}) \frac{A_\square}{\pi \cdot \bar{r}_f}\right) \tag{8.5}$$

with $A_\square = L^2$ as the area of the seed point plane. Thus, also with respect to the fiber radius distribution introduced next, there are enough fibers to fill the 2D plane.

These seed points are then used to place straight, parallel fibers within a cubic volume with edge length L. To add a random distribution of fiber radii, the target mean fiber radius $\bar{r}_f$ is multiplied by a random value of the *LogNormal* distribution (see fig. 8.2) to ensure that the mean value is preserved:

$$r_f = \bar{r}_f \cdot \text{sample}\left(\text{LogNormal}(\mu = 0,\, \sigma = 0.1)\right) \tag{8.6}$$

To achieve a random distribution of positions and radii along the fibers, they must initially be divided into fiber segments. The function `Solver.apply_boundaries` is used for exactly such a case. It applies the fiber segment length seg_L to the fiber configuration, which in this case divides the fiber along its trajectory into fiber segments of equal length (except for the last segment). The fiber points thus generated are then randomly shifted in all three dimensions with a normal distribution. The standard deviation of the normal distribution is set to a factor of the average fiber radii $\bar{r}_f$. In addition, the radius is changed with a random multiplicative factor from a *LogNormal* distribution:

$$\begin{aligned} p_i &= p_i + \text{Normal}(\mu = 0, \sigma = 0.05 \cdot \bar{r}_f) \\ r_i &= r_i \cdot \text{LogNormal}(\mu = 0, \sigma = 0.05) \end{aligned} \tag{8.7}$$

8.3 Software Parameters Characterization

The shape control mechanisms used in the algorithm of the collision solver `fastpli .model.solver` (see chapter 5) must be characterized. These are the mean segment length seg_L and the minimum allowed radius of curvature seg_R. It must be ensured that a set of parameters is identified ensuring that the configurations entered by the user remain the same i. e., the orientation distributions remain intact with respect to the fact that the fiber segments must move. In addition, the achievable fiber density for this parameter set should remain high. Finally, it must also be ensured that a parameter set can be chosen that has an acceptable runtime. The runtime should be minimized if possible, since it is usually a limited resource.

To investigate the behavior of seg_L and seg_R, the two relative factor variables ν_l and ν_r are defined:

$$\begin{aligned} seg_L &= \nu_l \cdot \bar{r}_f \\ seg_R &= \nu_r \cdot \bar{r}_f \end{aligned} \tag{8.8}$$

This allows the investigation of the model characterization independent of the fiber radius. The parameter ranges selected to study the effects are listed in table 8.1. To be able to check the results statistically, the generation of the models with different seed points is repeated $n_{repeat} = 24$ times. [2] For characterization, the pairs $\Psi_{\mathcal{F}_0}/\theta$ with $(||) = 1.0/0°$ and $(\times) = 0.5/90°$ are considered since they are edge cases. The number of steps is limited to $n_{max} = 100\,000$ to constrain the computation.[3] A cubic volume of 60 µm × 60 µm × 60 µm is picked. Every 50 steps, the fiber model is cut into a 60 µm $+ 4 \cdot \bar{r}_f$ cube to delete unnecessary fibers and reduce the number of objects and therefore the runtime. To measure the volume fraction, the discretized volume from the simulation is generated by the simulation module. There, the individual label IDs are counted and divided by the total number of voxels to calculate the volume fraction. To examine the final orientation of the model, the volume is reduced to 60 µm. The CPU architecture used to generate the models is an Intel(R) Xeon(R) CPU E5-4657L v2 @ 2.4 GHz, L1d cache: 1.5 MB, L1i cache: 1.5 MB, L2 cache: 12 MB, L3 cache: 120 MB.

8.3.1 Results

Development over time The results of the time measurement for the parameter set table 8.2 are shown in fig. 8.3 for the single fiber population $(||)$ and in fig. 8.4 for the crossing fiber population $(\times)$. They include the number of steps $\#steps$, the number of colliding fiber segments $\#objcol$, the overlap fraction of colliding fiber segments $\#overlapfrac$, the number of objects $\#obj$, and the current step time Δt. The overlap fraction of colliding fiber segments is defined as the average of the minimum distance between two colliding fiber segments divided by the sum of their radii. The results of the additional fiber radii are available in appendix A.

The single fiber populations $(||)$ show a strong linear correlation between the runtime and the number of steps for all parameters. The number of steps increases significantly with decreasing ν_l. A change in the running time when changing the fiber bending radius ν_r is not visible in the logarithmic plot. The total number of steps increases slightly with an increase in the fiber segment length factor ν_l. The total runtime increases

name	variable	values
mean fiber radius	$\bar{r}_f$	$(0.5,\ 1,\ 2,\ 5, 10)$ µm
mean segment length factor	ν_l	$1,\ 2,\ 4, 8$
min segment bending radius factor	ν_r	$1,\ 2,\ 4, 8$
fiber bundle fraction/crossing	$\Psi_{\mathcal{F}_0}/\theta$	1.0/0°, 0.5/90°

Tab. 8.1.: Parameter sets for the characterization of the software parameters ν_l and ν_r.

[2]The chosen central processing unit (CPU) architecture has 24 cores.
[3]In hindsight, this should have been limited by the runtime.

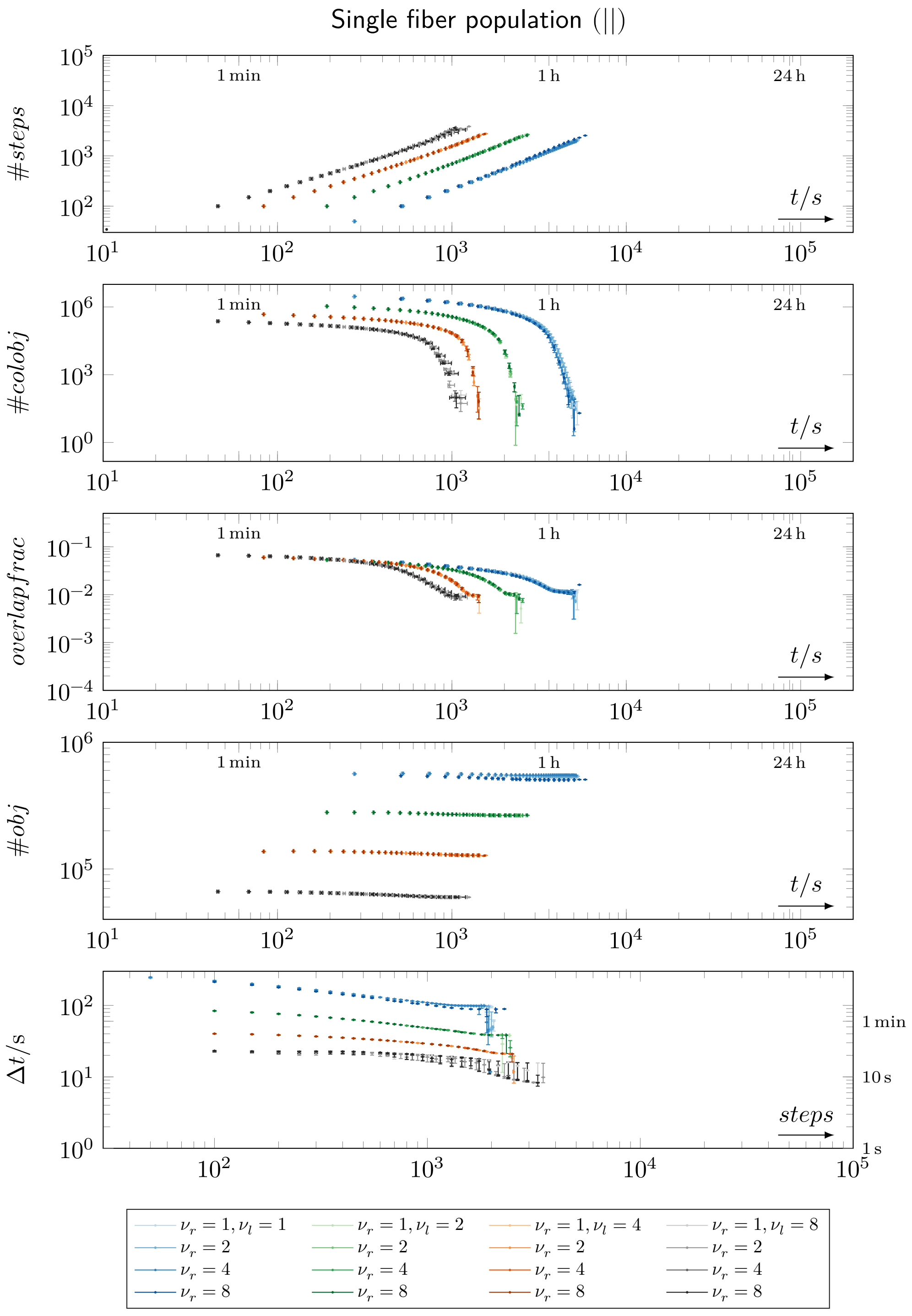


Fig. 8.3.: Time evolution of the model building process of parallel fiber populations. Error bars indicate 25 % and 75 % quantiles.

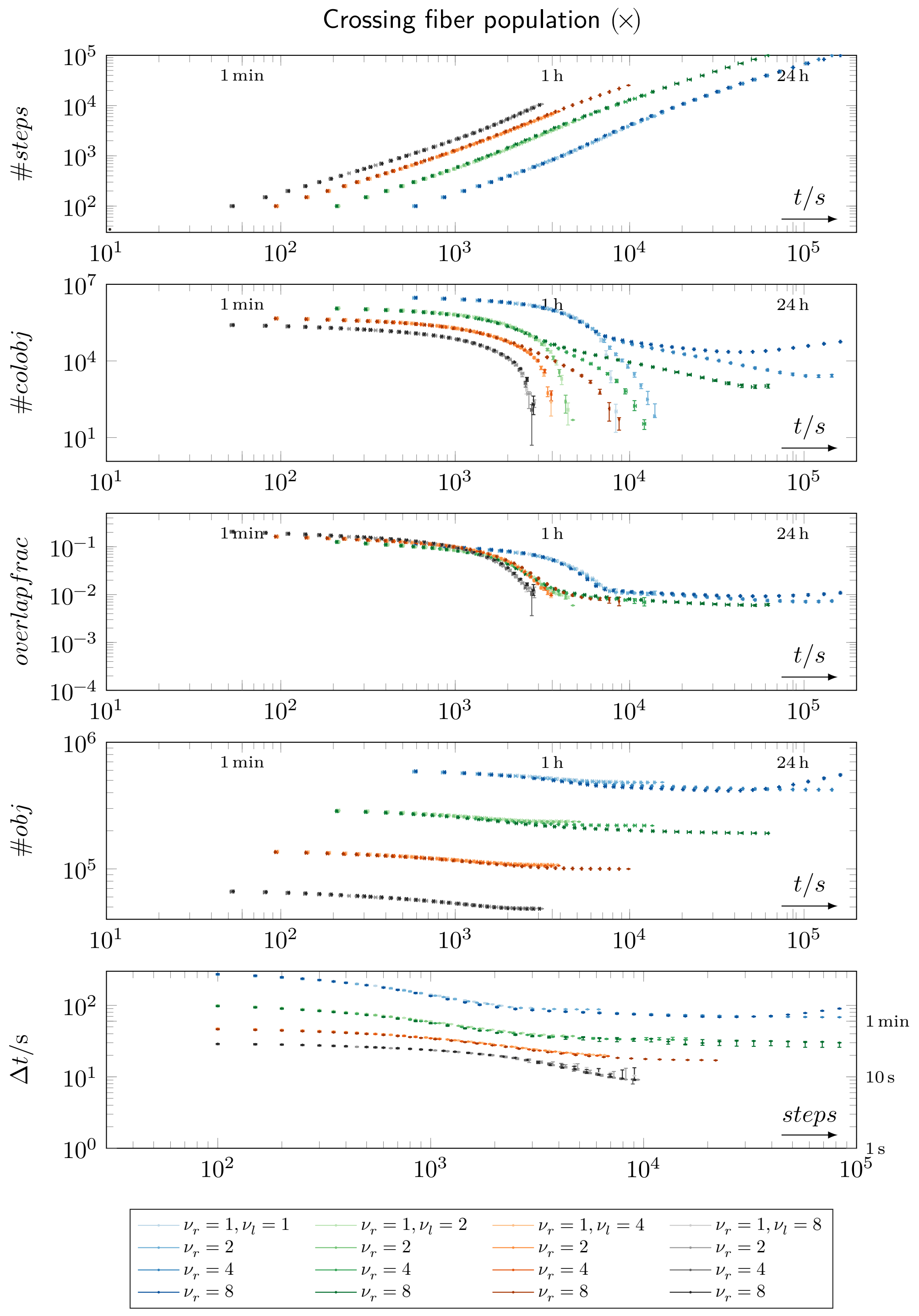


Fig. 8.4.: Time evolution of the model building process of crossing fiber populations. Error bars indicate 25 % and 75 % quantiles.

significantly with a decrease in ν_l. All models of all parameter sets were able to solve the collisions in the maximum number of steps.

The number of colliding fiber segments in the second plot of fig. 8.3 shows a strongly increasing decrease for all parameters with increasing runtime. The total number increases with decreasing ν_l. The ν_r also has no significant effect here.

The overlap fraction for all parameters is about $5\,\%$ at the beginning, and decreases with time to about $1\,\%$. The decrease is approximately linear after the initial phase. At the end of the collision solver process, the variance increases significantly. The curve behavior between the different ν_l is about the same, except of course for an offset in the total runtime. A difference when changing the ν_r is again not visible.

The number of objects decreases slightly over time. A significant increase can be seen by a decreasing value of ν_l. The ν_r is negligible except in the case of $\nu_l = 1$ where the number of objects is increased.

The last diagram shows the average time Δt needed for a single step. The relative length factor ν_l has the biggest influence on the results. A change with ν_r is almost not visible. The step time slowly decreases to about half the time required for the first step. The variance increases as the step size increases and can reach about half an order of magnitude in the case of ν_l.

The total runtime, i. e., the last data point in the graph, increases with ν_l. The other parameters seem to have almost no effect. The runtime ranges from 1000 s to 10 000 s.

The crossing fiber population ($\times$) in fig. 8.4 shows an equally linear behavior for all parameter sets for the number of steps as in the case of the single fiber population. The total runtime is significantly increased compared to the models with a single fiber population. Further, the runtime increases with a decrease in ν_l and decreases with a decrease in ν_r. In the case of $\nu_l = 1$ and $\nu_l = 2$, the maximum number of steps is reached. For $\nu_l = 1$ this has happened after 24 h.

The number of colliding fiber segments shows the same decreasing behavior in the case of the smallest ν_r as in the case of the single fiber population. The key difference is that this behavior changes significantly with an increase in the minimum fiber bending radius factor ν_r. The curves appear to split at a critical number of steps and progressively drop linearly towards higher ν_r and smaller ν_l.

The overlap fraction shows a significant difference for the smaller fiber segment length factor ν_l compared to the single fiber population ($||$) case. For high ν_l values, the curve almost follows the same path as in the single fiber population case, ending in the collision free state. However, for lower values and depending on the fiber segment radius

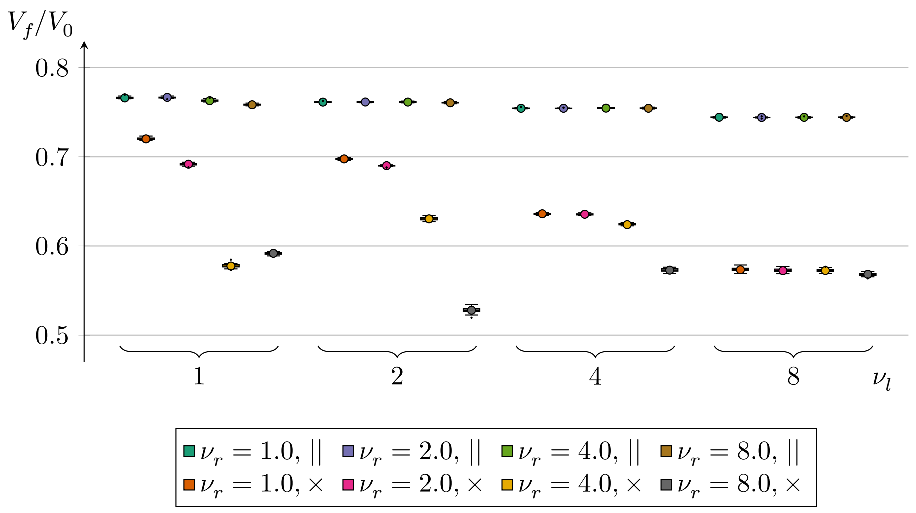


Fig. 8.5.: Volume fractions V_f/V_0 for parallel ($||$) and crossing ($\times$) fiber populations of different relative fiber segment lengths ν_l and relative fiber bending radii ν_r.

seg_R, the curve continues at about $1\,\%$ overlap. In the case of $\nu_l = 1$ and $\nu_r = 8$, it starts to grow slightly again at the end of its lifetime.

The number of objects shows the same behavior as in the case of the single fiber population. There is a slight decrease in the number and a significant increase with a decreasing value of the fiber segment length factor ν_l. The splitting of the curves for the ν_r is also visible, but not as prominent as in the case of the number of colliding objects. In the case of $\nu_l = 1$ and $\nu_r = 8$, an increase in the number of objects is observed in the last phase of the collision solver algorithm.

The time per step decreases for all parameters over the number of steps until it reaches a constant value for each ν_l. The ν_r does not seem to play a significant role. For $\nu_l = 8$, the variance increases again at the end of the runtime, as in the case of the single fiber population, but not as much. In the case of $\nu_l = 1$ and $\nu_r = 8$, one can observe an increase in the step time at the end of the runtime.

The total runtime is influenced by both the fiber segment length factor ν_l and the fiber segment radius factor ν_r. The splitting behavior visible in the graph of the number of colliding objects shows the significant difference in runtime for the ν_r. Increasing ν_r results in an increase in runtime for this parameter by up to an order of magnitude. The differences between ν_l are comparable to the single fiber case.

The other fiber radii show very similar behavior. The strongest difference is a very significant decrease in runtime with an increase in $\bar{r}_f$ (see appendix A).

Volume fraction Figure 8.5 shows the resulting volume fraction V_f/V_0 for the parameter series.

The populations with a single fiber orientation ($||$) have volume fractions greater than 0.74. For constant ν_l, there is no significant change with respect to the fiber bending radius factor ν_r. In the case of $\nu_l = 1$, there is a small decrease in volume fraction with increasing bend radius. There is a significant small decrease in the volume fraction when the fiber segment length factor ν_l is increased.

In contrast, the volume fraction of the crossing fiber population ($\times$) is significantly reduced. For $\nu_l = 1$ and $\nu_l = 2$, the volume fraction decreases sharply between $\nu_r = 2$ and $\nu_r = 4$. The values become more similar for $\nu_l = 2$. For $\nu_l = 8$ the change with ν_r disappears, and the volume fraction reaches only a value around 0.57.

For a mean fiber radius $\bar{r}_f = (1, 2)$ μm the behavior is very similar (see fig. A.6). The variance increases and the median decreases significantly for $\bar{r}_f \geq 5$ μm. The effect of the parameter is not changed otherwise. For mean fiber radii $\bar{r}_f \geq 5$ μm, the volume fraction does not decrease significantly.

8.3.2 Discussion

For parallel fibers, the collision solving process is always faster than for crossing bundles. This is visible in the plot of the overlap fraction figs. 8.3 and 8.4. The overlap is significantly higher for the crossing fibers than for the single fibers. Not only does this mean that the fiber segments that need to be moved out of the way are longer, but also that in higher octree levels the segments cannot be separated anymore and therefore exists in multiple leafs. As a result, the runtime for solving the volume increases significantly.

The segment length factor ν_l is the most important parameter to reduce the runtime for constant mean fiber radii $\bar{r}_f$ because the number of objects is smaller. However, as expected, larger ν_l leads to a lower volume fraction, especially for crossing fibers. For a single fiber population ($||$), it makes sense that the volume fraction does not change, since the direction of motion is radially symmetric along their main orientation axis for all fiber segments to avoid overlap.

The fiber segment bending radius factor ν_r restricts the bending radius, it is to be expected and visible in the results that the volume fraction is strongly influenced since the volume can no longer be filled optimally. As before, changing the ν_r in the case of a single fiber bundle should not lead to any difference. However, crossing fibers will be strongly affected. This is especially evident in the number of colliding objects, where the ν_r splits the data into individual branches over time. Smaller values of ν_r allow more curved geometries here. However, this can lead to unnatural results in terms of anatomical structure. As a result, a large part of the volume would be filled with fibers or rather fiber segments, but the actual number of fibers would be smaller. A non-intuitive effect for smaller ν_l and higher ν_r is visible. The number of colliding objects increases again, as does the total number of objects. Therefore, larger ν_r should be avoided, as this effect also occurs when $\nu_l = 2$ and $\nu_r \geq 4$. Since such a large radius factor represents an unnatural stiffness for a nerve fiber, this is not a major concern for this type of models.

To narrow down the selection of a parameter set, the crossover fiber population ($\times$) is the most important data set. The most essential parameter is the length factor ν_l. From the results of the volume fraction, it can be concluded that ν_l should be as small as possible. However, since the difference between $\nu_l = 1$ and $\nu_l = 2$ is small, and a large jump is visible for $\nu_l = 4$, $\nu_l = 2$ can be used to significantly reduce the runtime.

The choice of the fiber bending radius factor ν_r is an anatomical decision. To counteract excessive deformation, a value of at least $\nu_r = 2$ should be chosen. A larger value of ν_r would significantly increase the runtime and add unnatural stiffness. ν_l should be as small as possible to achieve the highest accuracy. However, the runtime would increase too much for very small values. Therefore, a value of $\nu_l = 2$ was selected, since higher values already have a significant impact on the volume fraction.

In summary, the following parameters are selected:

Tab. 8.2.: Selection of parameters for the creation of the 3D-PLI simulation model.

name	variable	value
mean fiber radius	$\bar{r}_f$	0.5 µm
mean segment length factor	ν_l	2
min segment bending radius factor	ν_r	2

For fiber radii $\bar{r}_f > 0.5$ µm, the most important effect is the shortening of the runtime at constant volume. In addition, boundary effects become apparent due to the limited volume. For example, the mean volume fraction for each configuration decreases significantly and the variance increases. This is the effect of being able to place only a few fibers into a 60 µm thick volume. If they are then randomly arranged and deformed, as in this case, the volume fraction must decrease. If the volume were infinite, there would be no difference between the results, since all parameters are independent of the radii. Therefore, an increase in the fiber radii should be considered if the volume to be calculated is larger or the runtime is finite. It should then be verified that the simulation does not cause significant changes in the results.

Another way to shorten the runtime is to not solve the model completely. Looking at the overlap fraction, a value of $< 1\,\%$ might be feasible. It should be noted that this is only the fraction of the fiber segment that still overlaps. The number of colliding fiber segments decreases steadily for the chosen parameters. This potentially saves an order of magnitude in runtime. Here, the influence on the simulation must also be investigated in advance.

8.4 Nerve Fiber Model Library for 3D-PLI Simulations

With the model parameters selected above (see section 8.3), the simulation models are generated. As described in section 8.2.1, only fibers with $\theta = (10 \text{ to } 90)°$ and $\Psi_{\mathcal{F}_0} = (0.1 \text{ to } 0.9)$ are generated with the addition of $\theta = 0°$, $\Psi_{\mathcal{F}_0} = 1.0$. The volume for the simulation is scaled up to a sphere with a diameter of 135 µm so that a simulated volume can be generated from the sphere in any orientation.

8.4.1 Results

Figure 8.6 shows the orientation distribution of the fiber segments as a polar histogram. The full dataset is available in fig. A.7. The orientation distributions show that the individual fiber segments statistically follow the orientation of the fiber population. The density reflects the population fraction $\Psi_{\mathcal{F}_0}$ of the fiber populations.

Figure 8.7 shows the distributions of direction φ, inclination $\alpha_{\mathcal{F}_0}$, and opening angle Ω, which is the relative angle to the mean fiber population orientation. Since the inclination depends on the direction, the opening angle is also analyzed. The full dataset is available at fig. A.8. The individual measurements of the opening angle distributions are available in table A.1.

The direction and inclination angles are around 0° for the first fiber population $\mathcal{F}_0$ for all parameter combinations. The variance decreases with increasing population fraction $\Psi_{\mathcal{F}_0}$ of the second fiber population. The crossing angle θ does not seem to have any influence. The opening angle Ω has its mean value between about 10° and 20° for all parameter configurations. The variance decreases with increasing fiber population fraction $\Psi_{\mathcal{F}_0}$. The crossing angle θ does not seem to have any influence.

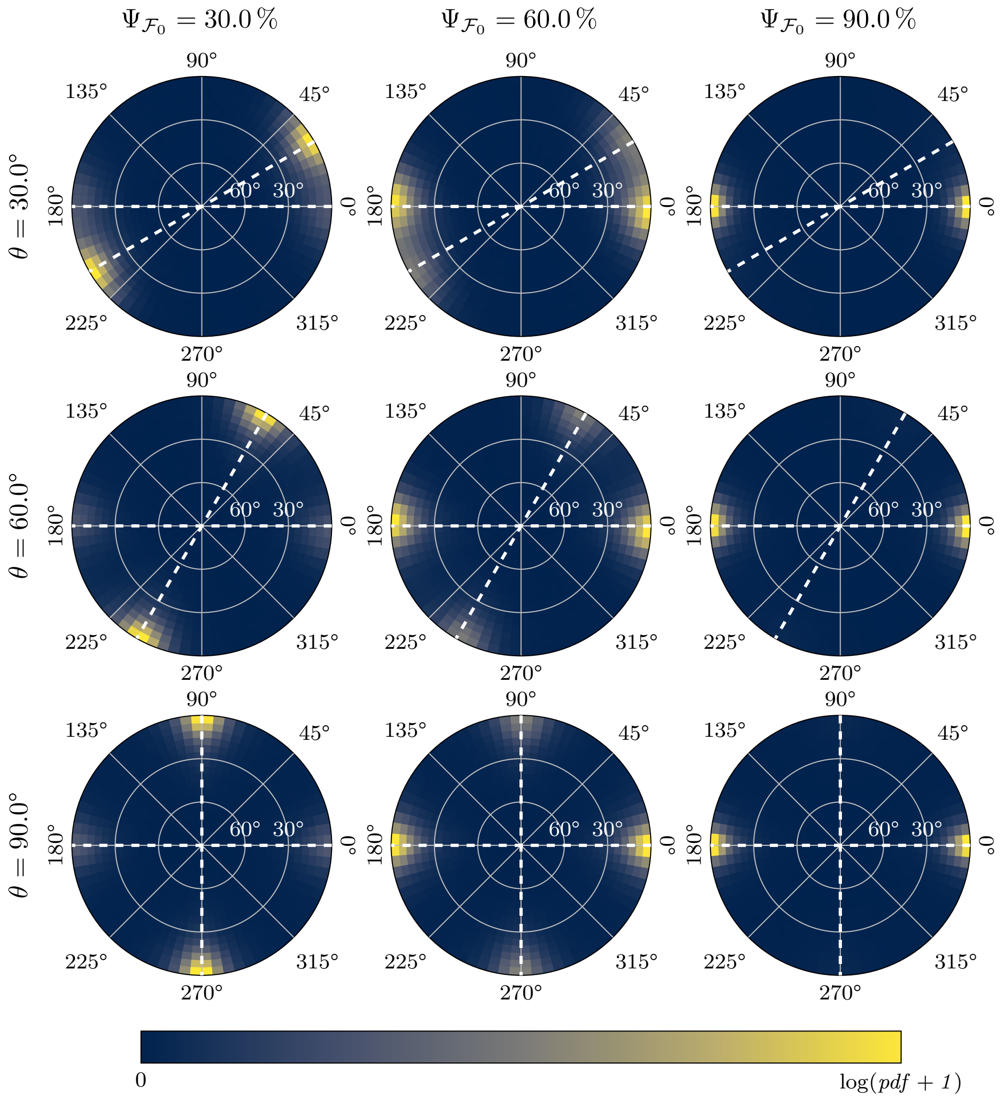


Fig. 8.6.: Density distribution of fiber segment orientation in the simulation models. The value of the segments is weighted according to the area on a spherical surface and normalized so that the integral over one hemisphere is 1. The dashed white line indicates the orientation of the two fiber populations.

The mean direction φ follows the initial crossing angle θ. The fiber population fraction $\Psi_{\mathcal{F}_0}$ does not seem to have a significant influence. The inclination $\alpha_{\mathcal{F}_0}$ is stable around 0° for all parameters. The parameters do not seem to have a significant effect on their distribution. The opening angle Ω is between 15° and 20° for the second fiber population. For smaller crossing angles, the mean values almost do not change. For large crossing angles, the mean value of the opening angle increases with increasing $\Psi_{\mathcal{F}_0}$.

Figure 8.8 shows the visualization of the fiber models. The visualization shows the inner 10 µm × 10 µm × 10 µm cube, so more details are visible. Depending on the population

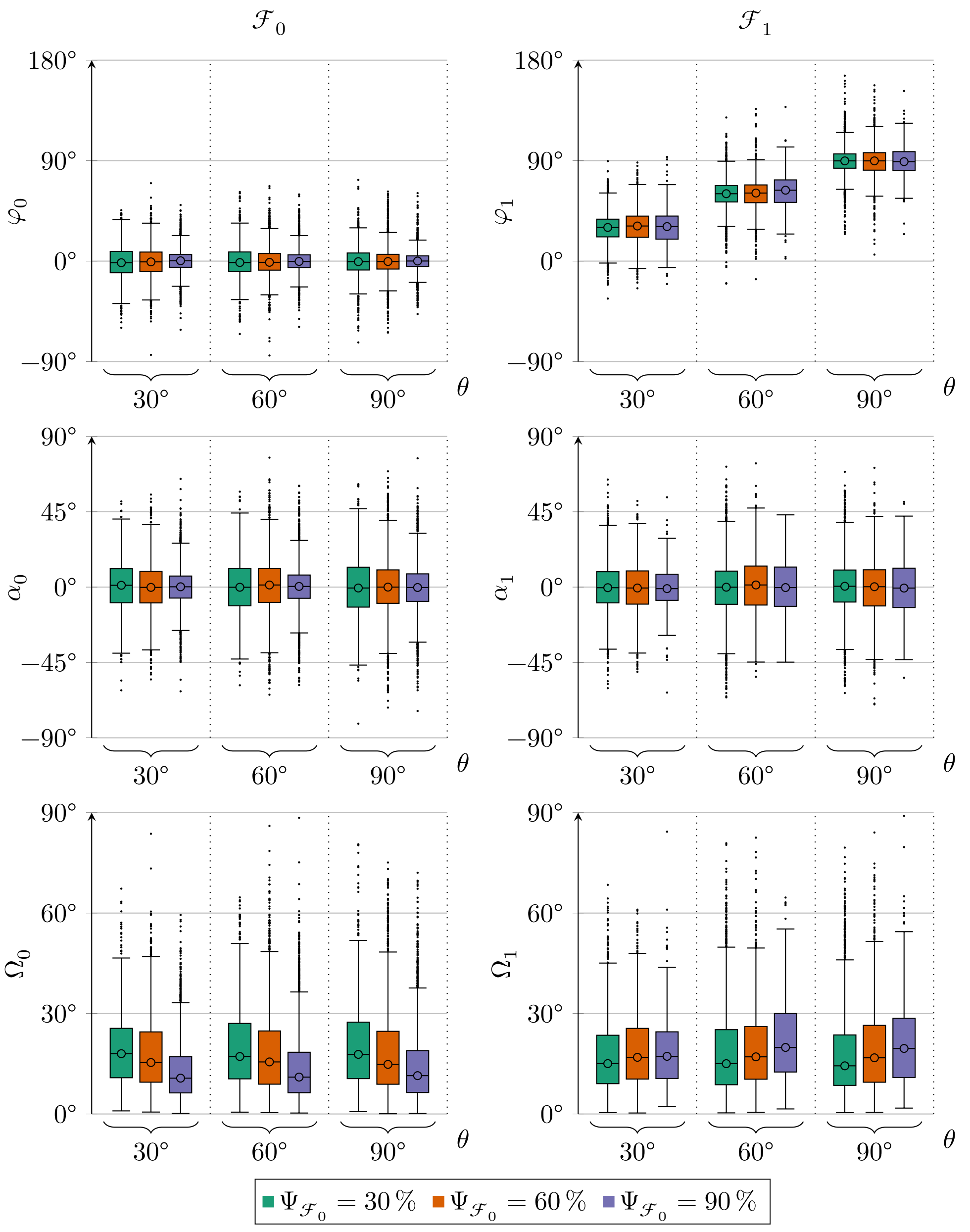


Fig. 8.7.: Direction φ, inclination α and opening angle Ω distribution of the model library.

$\Psi_{\mathcal{F}_0}$, more or less layered distinct structures are visibly parallel to the intersection plane. The full dataset is available at fig. A.9.

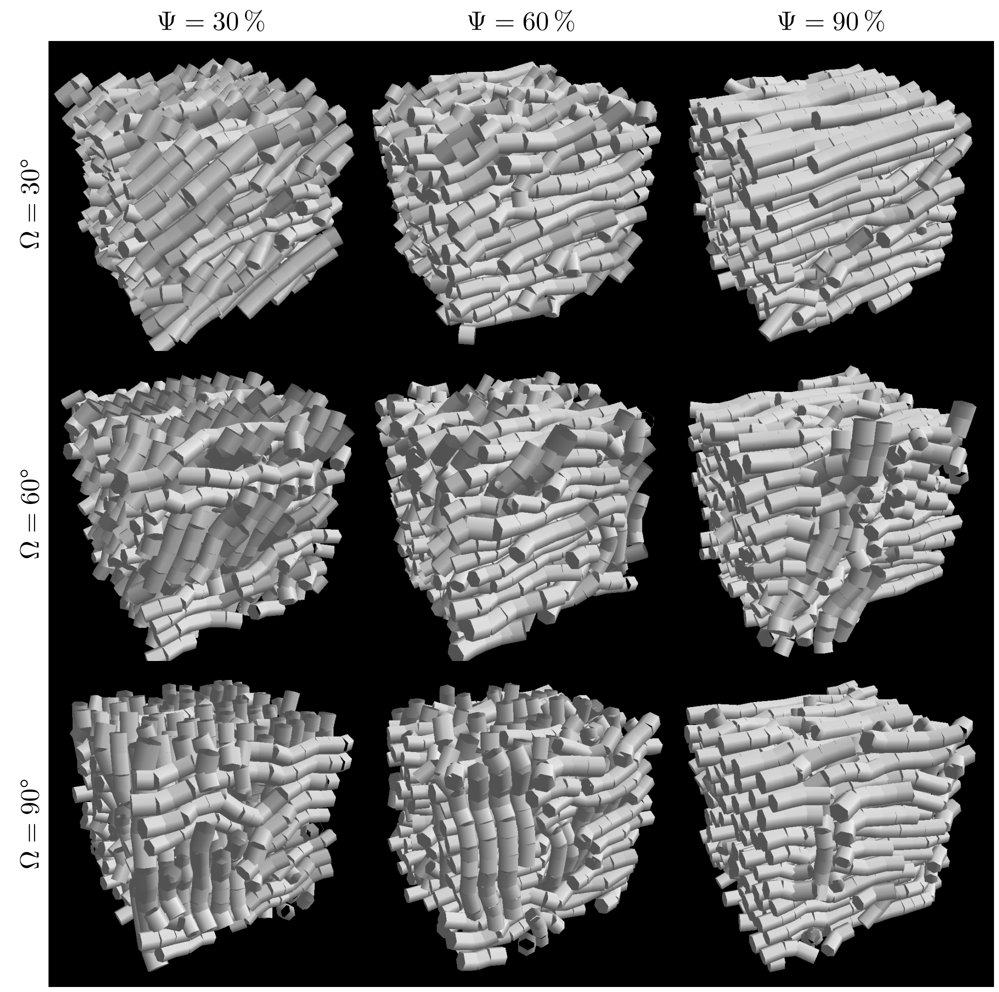


Fig. 8.8.: 3D visualization of the simulation model library 3D-PLI. The inner 10 µm × 10 µm × 10 µm of the volumes are shown.

8.4.2 Discussion

These models have the advantage of producing a naturally inspired angular distribution, which should result in a more realistic distribution. The orientation distribution of the simulation model library is stable at all crossing angles. However, the distribution variance is lower for the fiber population with higher population fraction and vice versa. This is plausible because the main fiber bundle exerts more radial pressure on the orientation, resulting in a more stable configuration for cylindrical objects. This probably has little effect on the simulation results. The stiff models lead to a volume fraction that depends on the crossing angle. Whether this is also true for anatomical tissue remains to be explored. However, since real nerve fibers are not stiff and also not perfectly cylindrical, it can be expected to be higher. Also, the orientation distribution may differ slightly from real tissue due to the stiffness of the model. The layered structure

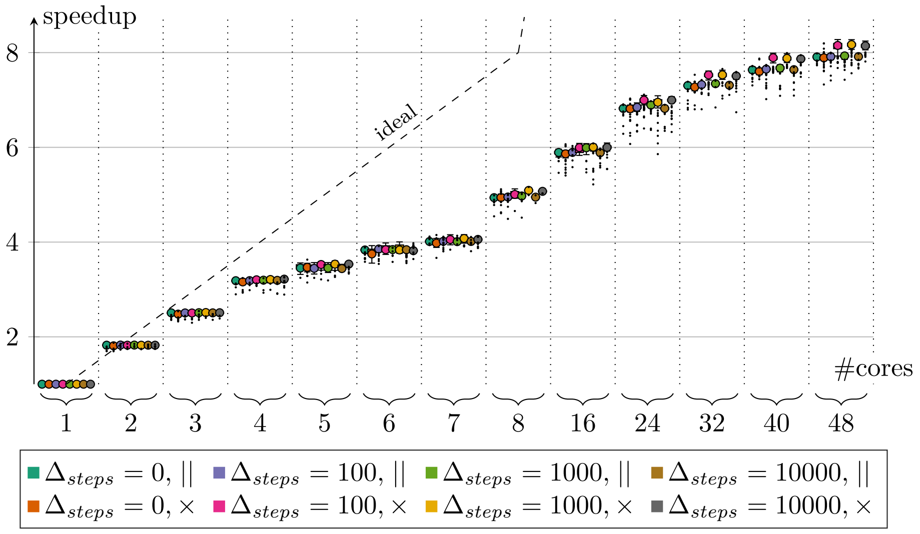


Fig. 8.9.: `model.Sovler` speedup. Time measurements are performed after Δ_{steps} for the next 25 steps for parallel ($||$) and crossing ($\times$) fiber configurations.

can be understood as a randomly introduced pattern. Real fabric will also have a different pattern, e. g. interwoven fiber bundles instead of interwoven single fibers. In linear 3D-PLI simulations, this has no effect because the retardation matrices for the intensity signal are commutative. Or more generally, the order of the tissue matrices is commutative for the intensity signal. In other fields, such as dMRI, additional models are probably required in this regard.

8.5 Multicore CPU Acceleration

As described in section 5.4.4, Open Multi-Processing (OpenMP) is used to accelerate `for` loops in the algorithm. This means that it is currently not possible to use multiple compute nodes.

To investigate the speedup, the runtime for the parameters from table 8.2 is measured for parallel and crossing fibers. The volume used is 60 µm × 60 µm × 60 µm. Speedup is calculated for an average of 100 consecutive steps. To measure whether the speedup changes with time, the measurements are started after a certain number of steps $\Delta_{steps} = (0,\ 100,\ 1000, 10\,000)$ steps. The calculation is repeated for each case $n = 24$ times. For the measurements, the cores were bound to physical CPU cores.

8.5.1 Results

Figure 8.9 shows the speedup for different starting positions Δ_{steps} for the parallel ($||$) and crossing ($\times$) fiber bundles. Up to 4 cores, the speedup increases linearly to a speedup of 3. From 4 cores, the speedup increases more slowly until a significant increase to a speedup of about 5 is observed for 8 cores.

Overall, no significant change is visible for parallel or crossing fibers. Also, for different starting points Δ_{steps}, no change in acceleration is visible. The speedup increases up to 8.

8.5.2 Discussion

Up to 4 cores the speedup is expected with 3. The slower increase and jump for 8 cores can be explained by the structure of the *octree*. Since for 8 cores the parallel alignment on a *octree* (can) be optimal, the speedup increases. More cores are not feasible.

Nevertheless, the data suggests that using a multicore system shortens the runtime considerably. However, for small volumes or low-fiber objects, one should use only one or two cores and prefer to run multiple models in parallel. The results also show that an optimized algorithm is needed, especially when the volume or the number of objects increases. Here, the graphics processing unit (GPU) seems to be the hardware of choice with a more advanced algorithm [Kar12] (see chapter 10).

9 3D-PLI Simulation

9.1 Introduction

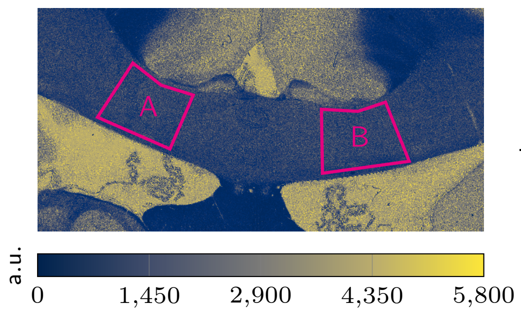
This chapter covers with the simulation of 3D-PLI using the generated nerve fiber library. The first part focuses on the determination of all necessary physical parameters of the tissue and the microscope, as well as on the characterization of the simulation parameters. Then, the simulation of the previously generated nerve fiber models is created and analyzed. The orientation of the simulations is calculated using the routine algorithms implemented in *fiber architecture simulation toolbox for 3D-PLI* (*fastPLI*). The focus of the evaluation is on the accuracy of the tilt analysis for different orientations and crossing configurations.

9.2 Parameter Characterization

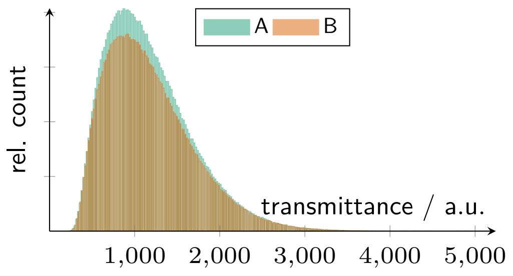
9.2.1 Tissue

The absorption coefficient μ and the birefringence coefficient Δn have to be estimated from a measured section for the simulation. To measure the values, it is important to analyze a homogeneous region filled with flat, dense fibers. In a coronal section, the corpus callosum is suitable for this purpose. It is the main fiber connection between the two cerebral hemispheres (see fig. 2.1b).

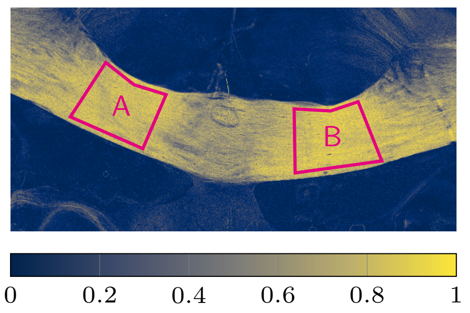
Figure 9.1 shows transmittance and retardation maps of the coronal section of a Vervet monkey (Brain id: PE-2012-00102-V, section: 550). Two region of interests (ROIs) are manually selected at high retardants regions of the corpus callosum (i. e. mostly in-plane parallel fibers). ROI (A) contains 1 064 629 px and ROI (B) sums up to 1 125 858 px. The entire section and the analysis for a human and a rodent section are described in the appendix (see figs. B.1 and B.2).



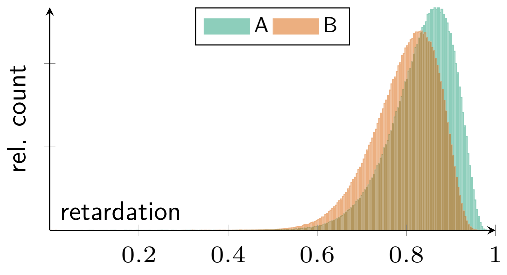
(a) transmittance map



(b) transmittance histogram



(c) retardation map



(d) retardation histogram

Fig. 9.1.: Transmittance and retardation map of coronal section of a Vervet monkey (Brain id: PE-2012-00102-V, section: 550). The absorption coefficient and birefringence is estimated from the measurements in the corpus callosum. Two homogeneous regions are labeled for this purpose. (A) includes 1 064 629 px, (B) 1 125 858 px.

Evaluation To calculate the absorption coefficient from the brain section, the volume fraction of the tissue, i. e., density of the cells, must be considered. For the models $\bar{r}_f = 0.5\,\mu\text{m}$, the volume fraction is about 75 %. The average transmittance in the foreground is about 1000 a.u. and in the background it is about 4500 a.u..[1] This leads to an absorption coefficient of $\mu = 30\,\text{mm}^{-1}$. To approximate the birefringence, the g-ratio, i. e., the myelin density, must also be taken into consideration. The g-ratio of 75 % results in a birefringence of about 0.008 for the radial model when using a median value of the retardation of 0.8.

9.2.2 Optical resolution

The optical resolution depends, among other things, on aberration and diffraction (see section 4.6). They are modeled as described in section 6.2.4. Therefore, the model parameters must be determined for each microscope to be simulated.

[1]The relative transmittance can be changed by adjusting the exposure time of the microscope.

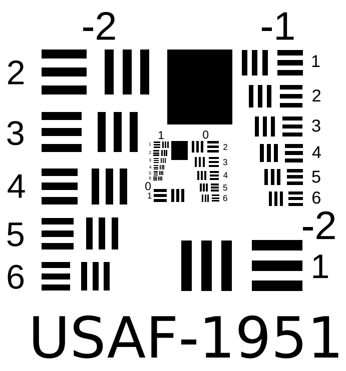
To measure the optical resolution of the microscope, measurements and analyses are repeated [Men14]. For this purpose, the *1951 United States Air Force (USAF) resolution test chart*[2] is used. It consists of several patterns that have three slots with specific spacing and width (see fig. 9.2a). They are arranged in fields of three vertical and horizontal lines. The fields are arranged in a spiral that shrinks from group to group by a factor 0.5. To determine the line width, the fields are numerically ordered according to a main group i and a subgroup j. To determine the resolution of the microscope, it is necessary to determine if the neighboring slits can still be differentiated according to the Rayleigh criterion (see fig. 4.4). For this purpose, the intensity profiles perpendicular to the three vertical and horizontal slits were analyzed.

Evaluation Figure 9.2b shows a section of the USAF resolution test chart taken with the large metripol (LMP) microscope. The highlighted areas show the analyzed groups 7-6 to 8-2. The area of group 7-6 -· has a line width of 2.19 µm, group 8-1 -· of 1.95 µm and group 8-2 -· of 1.74 µm. The intensity line profiles for the vertical and horizontal cases are shown in figs. 9.2c and 9.2d. The Rayleigh criterion can be used to determine the resolution in the second group, which is thus in the range of 1.95 µm. This reproduces the measurements in [Men14]. Therefore, the convolution parameter is set to $\sigma_{\text{optic}} = 0.75\,\text{px}$ in the simulation (see section 4.6).

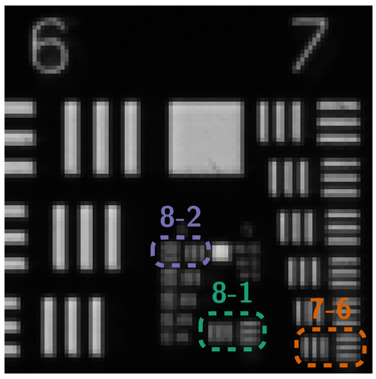
9.2.3 Sensor gain and signal noise

As described in section 6.2.4, the optical noise is modeled with a Gaussian model. The gain factor of a charge-coupled device (CCD) camera describes the linearity between a measured electrical signal and the output signal. This can also be used to calculate the image noise, since it correlates with $\sqrt{I \cdot g}$ [Wie16]. To measure the noise, different intensity values are measured. For this purpose, the sample stage is covered with a fully absorbing cover so that half of the image is dark (see fig. 9.3a). In addition, the focal length is changed so that the light is distributed over the entire image sensor under the cover (see fig. 9.3b). By measuring $N = 500$ images, the variance for the different intensity values is determined.

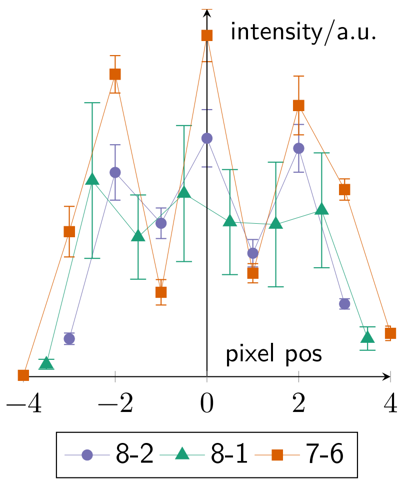
[2]U.S. Air Force MIL-STD-150A standard of 1951.
[3]https://en.wikipedia.org/wiki/1951_USAF_resolution_test_chart



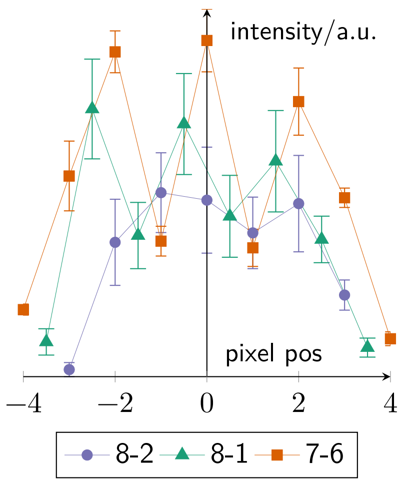
(a) USAF chart from group -2 to 1.[3]



(b) Microscopic image with highlighted groups 7-6 to 8-2.



(c) Centered line plots horizontal slits.



(d) Centered line plots vertical slits.

Fig. 9.2.: Group 8-1 of the USAF chart with a line width of 1.95 µm corresponds to the Raileigh criterium. Therefore an optical convolution of $\sigma_{\text{optic}} = 0.75\,\text{px}$ will be applied in the simulation.

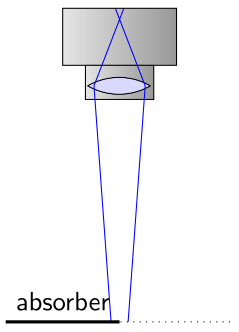
Evaluation The results are shown in fig. 9.3c and show a gain factor of $g_{LMP} = 0.1175(7)$. This gain factor is used to model the integer noise with

$$f(x) = \lfloor \text{normal}(\mu = x, \sigma = \sqrt{gx}) + 0.5 \rfloor \qquad (9.1)$$

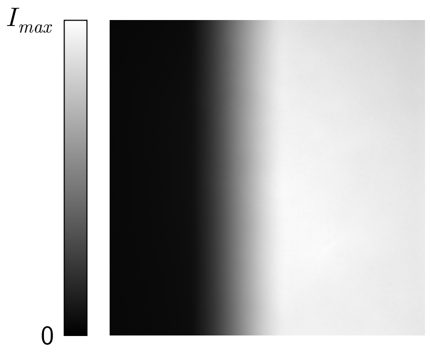
for intensities $I > 0$ and $I \gg \sqrt{gI}$.

9.2.4 `voxel_size` v_s

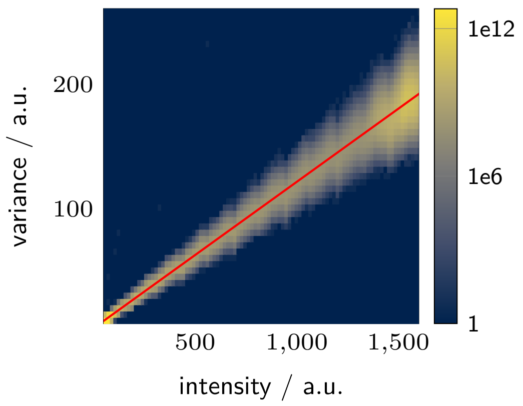
The parameter `voxel_size` v_s is the most important parameter for simulation accuracy, as it determines by how many voxels the models are discretized and how many light



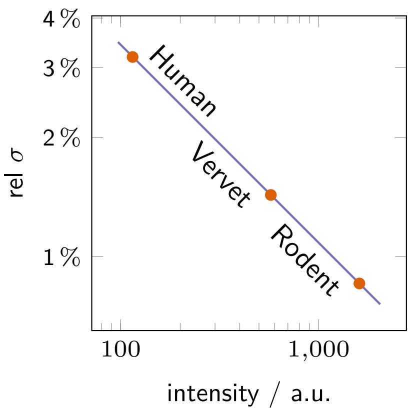
(a) Scheme of unfocused microscopic image.



(b) Microscopic image.



(c) Histogram of intensity vs variance with linear regression. The gain factor results in $g_{LMP} = 0.1175(7)$.



(d) Expected noise range for human, Vervet monkey and rodent. The noise correlates linearly with the light intensity.

Fig. 9.3.: LMP camera noise measurements.

rays are modeled. However, the voxel size cannot be arbitrarily small because the number of calculations and memory consumption increase by $O(n^3_{voxel})$. Therefore, it is recommended to set the voxel size as large as possible without introducing significant errors due to discretization.

To investigate the influence of the voxel size, a simulation with several voxel sizes in the range of $v_s = (0.01 \text{ to } 1.3)\,\mu\text{m}$ is performed. The smallest voxel size $0.01\,\mu\text{m}$ is used as a reference. Since this voxel size is relatively small, the simulated volume is $3 \cdot 1.3\,\mu\text{m} \times 3 \cdot 1.3\,\mu\text{m} \times 60\,\mu\text{m}$ without tilts to limit the memory consumption. During the simulation, the previously determined tissue and noise parameters are used. The models to be simulated are parallel models without inclination $\left((||), \alpha_{\mathcal{F}_0} = 0°\right)$, inclined parallel fiber models $\left((||), \alpha_{\mathcal{F}_0} = 90°\right)$, flat crossing fiber models $\left((\times), \alpha_{\mathcal{F}_0} = 0°\right)$, and crossing inclined fiber models $\left((\times), \alpha_{\mathcal{F}_0} = 90°\right)$ with a fraction of $\Psi_{\mathcal{F}_0} = 0.5$, since

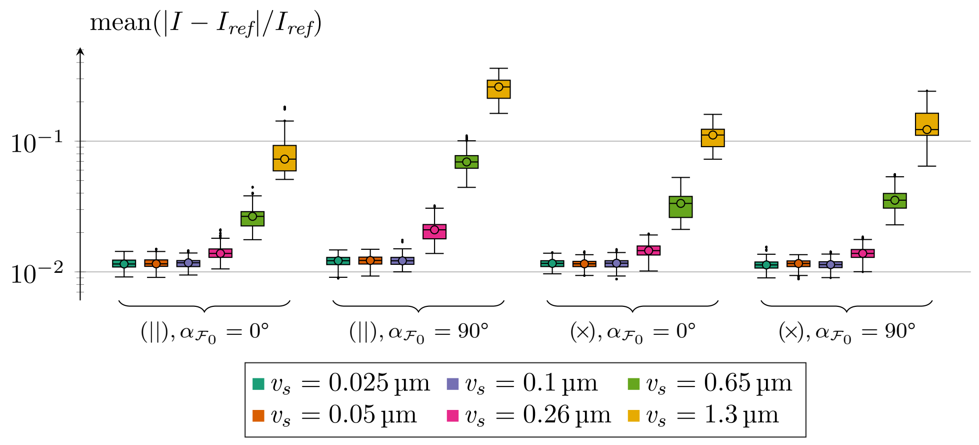


Fig. 9.4.: Comparison of the simulation results with different voxel sizes v_s and fiber configurations.

these configurations represent the extreme for two fiber populations. For statistics, the simulations are repeated 25 times with the volumes center at

$$x = \frac{i \,\%\, 5}{5} \cdot 60\,\mu\text{m} - 30\,\mu\text{m} \qquad y = \frac{\lfloor i/5 \rfloor}{5} \cdot 60\,\mu\text{m} - 30\,\mu\text{m} \tag{9.2}$$

with i from 0 to 24.

Evaluation The results in fig. 9.4 show that the relative difference to the smallest voxel size increases statically significantly from a value of 0.1 µm. The variance of the relative difference increases with increasing voxel size from this value. For a voxel size smaller than 0.1 µm, the difference from the reference simulation does not increase. Therefore, the voxel size v_s = 0.1 µm is a good compromise between runtime and accuracy. However, such assumption is only valid for a pixel size of 1.3 µm and fiber radii of 0.5 µm. Additional parameters are available in fig. B.3.

9.3 Simulation

9.3.1 Setup

As described in chapter 7, *fastPLI* contains a pipeline implementation of the simulation with automatic analysis. This implementation is used to compute the simulation of the signal for the flat and tilted measurements and to apply the optical noise and tilt analysis to the resulting signal using the *robust orientation fitting via least squares* (*ROFL*) algorithm. The input of the simulation is the previously created model library (see section 8.4). The models were rotated according to the previously specified discretization (see section 8.2.1).

Three types of nerve fiber models are simulated. First, a single fiber population. Since there is only a single fiber population, only the inclination of the model is changed. In the second step, simulations are performed with two fiber population orientations: a) flat crossing and b) inclined crossing. The first fiber population is fixed along the x-axis in these models, while the crossing angle of the second fiber population is increased. The last simulation contains two fiber populations where the inclination of the first fiber population is increased and the second fiber population is rotated around it as the crossing angle increases (see fig. 8.1).

Table 9.1 lists the parameters of the simulation in the notation of *fastPLI*. With a pixel size of 1.3 µm and a volume of 65 µm $\times$ 65 µm $\times$ 60 µm, 2500 px are available per simulation for statistical analysis.

To analyze the simulation results the 3D-PLI analysis pipeline, containing the Fourier analysis (see eqs. 4.1) and the tilt analysis (see section 4.5), are performed. From the Fourier analysis, the modalities transmittance and retardation are shown. The

Tab. 9.1.: Simulation parameters.

variable	value
simpli.voxel_size	0.1 µm
simpli.pixel_size	1.3 µm
simpli.voi	$[[-35\,\mu\text{m}, -35\,\mu\text{m}, -30\,\mu\text{m}], [35\,\mu\text{m}, 35\,\mu\text{m}, 30\,\mu\text{m}]]$
simpli.filter_rotations	$(0,\ 20,\ 40,\ 60,\ 80,\ 100,\ 120,\ 140, 160)°$
simpli.interpolate	`"Slerp"`
simpli.wavelength	525 nm
simpli.optical_sigma	0.75 px
tilt angle	3.9°
simpli.light_intensity	8000 a.u.
gain	0.1175
simpli.noise_model	`lambda x: np.floor(np.random.normal(x, np.sqrt(gain * x))+0.5).astype( np.uint16)`
fiber absorption	Vervet: 30 mm^{-1}
fiber birefringence model	'r'
fiber birefringence	0.008
fiber radii scale factor	axon: 0.75, myelin: 1
model inclination step $\Delta\alpha_{\mathcal{F}_0}$	one f. pop.: 5°, two f. pop.: 30°
model rotation step $\Delta R_{\mathcal{F}_1}$	15°

tilt analysis estimates the orientation of each pixel, i. e., direction and inclination, the effective birefringent thickness t_{rel} and the R-value of the fitting model:

$$R = \text{mean}\,|I_i - F_i| \tag{9.3}$$

where I is the signal intensity and F the best solution found by the model. Finally, the opening angle Ω and angular correlation coefficient acc-value between the nerve fiber models orientation and simulation resulting orientation are calculated from the orientations. The acc-value is the angular correlation coefficient, which measures the similarity between two orientation distribution functions (ODFs). The ODFs are calculated by an analytical function from the individual orientations [Ali+20]. The acc-value is then be calculated by:

$$acc = \frac{\sum_i sh_{0,i} \cdot sh_{1,i}}{\sqrt{\sum_i sh_{0,i}^2} \cdot \sqrt{\sum_i sh_{1,i}^2}} \tag{9.4}$$

where sh_i is the i—th spherical coefficient of the orientations ODF [Sch+18a].

To visualize the distribution of the data boxplots are used. The distribution of the angular parameters requires caution, since the values are periodic. To overcome this problem, the values are centered around the circular mean value in the range of $\left[\text{circmean}(\xi) - 90°, \text{circmean}(\xi) + 90°\right)$, where ξ is the angular parameter. The circular mean value is calculated via:

$$\text{circmean}(x) = \text{atan2}\left(\frac{\sum \sin(x_i)}{\sum \cos(x_i)}\right) \tag{9.5}$$

This contributes to the interpretability of the angular results with variances $\ll 90°$. Additionally, for the angular parameters the distribution of the underlying fiber models are plotted for the individual fiber populations $\mathcal{F}_0$ and $\mathcal{F}_1$. The area corresponds to the 25 % to 75 % quantile and will be referred to as $\sigma_{25\,\%}^{75\,\%}$ variance.

9.3.2 Single fiber population

Figure 9.5 shows the orientation distribution of a single population of fibers with increasing inclination angle $\alpha_{\mathcal{F}_0}$. The distributions of the orientations of the model segments are shown on the left, those of the tilt analysis on the right. The results of the tilt analysis show the same mean orientation as the models with a slightly lower variance in comparison.

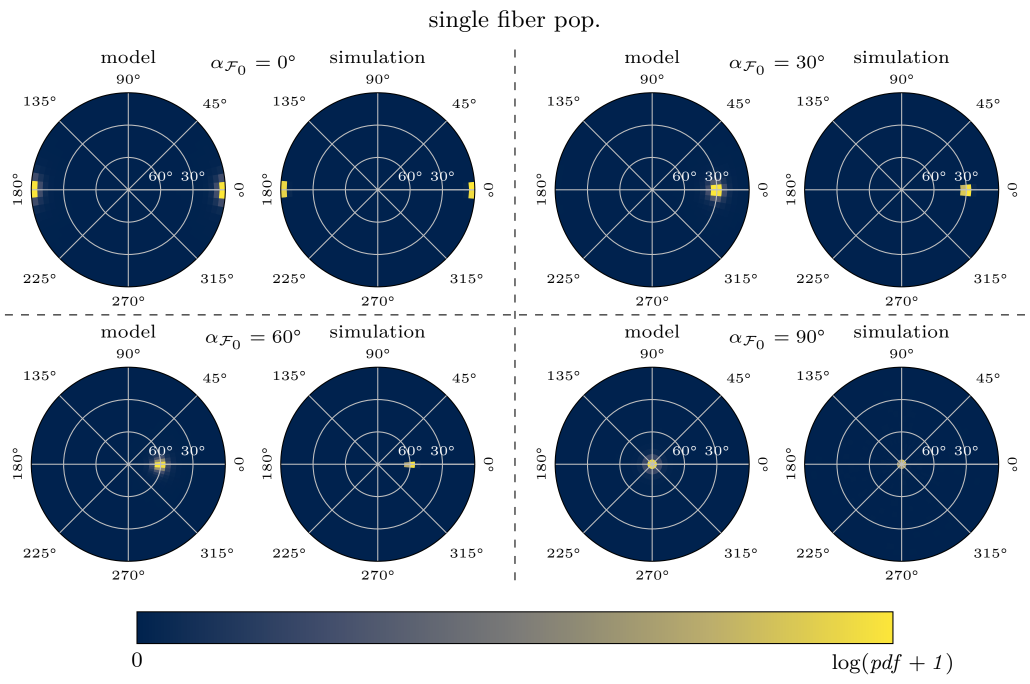


Fig. 9.5.: Single fiber population orientation histograms of the initial fiber model and the simulation results.

Figure 9.6 shows the results of the simulation on the single fiber population with different inclinations.

The transmittance is constant with increasing inclination angle up to a value of $\alpha_{\mathcal{F}_0} < 75°$. For $\alpha_{\mathcal{F}_0} \geq 75°$ the transmittance increases. The variance of the transmittance increases slightly with increasing inclination angle $\alpha_{\mathcal{F}_0}$.

The retardation plot shows the Fourier analysis results as well as a theoretical curve that follows $(\cos(\alpha_{\mathcal{F}_0}) + 1)/2 \cdot \mathrm{mean}\,(\mathrm{ret}(0°))$. The curve is normalized by $\mathrm{mean}\,(\mathrm{ret}(0°))$ for comparison purposes. The retardation of the simulation signal follows the theoretical line. However, at the intermediate inclination angles, the measured retardation is slightly higher than the theoretical line. The variance remains constant for all fiber inclination angles $\alpha_{\mathcal{F}_0}$ with the exception for very steep fibers, where the distribution is shifted towards values > 0.

The next graph shows the measured direction from the tilt analysis. For inclination angles $\alpha_{\mathcal{F}_0} < 75°$ the variance is about a few degrees and remains constant. For larger values, the variance increases. [4] In the background the statistic $\sigma_{25\,\%}^{75\,\%}$ variance of the

[4]It should be noted that the boxplot does not take into account that the values are periodic. Therefore, especially for $\alpha_{\mathcal{F}_0} = 90°$, the values from the $(-90$ to $90)°$ are uniformly distributed.

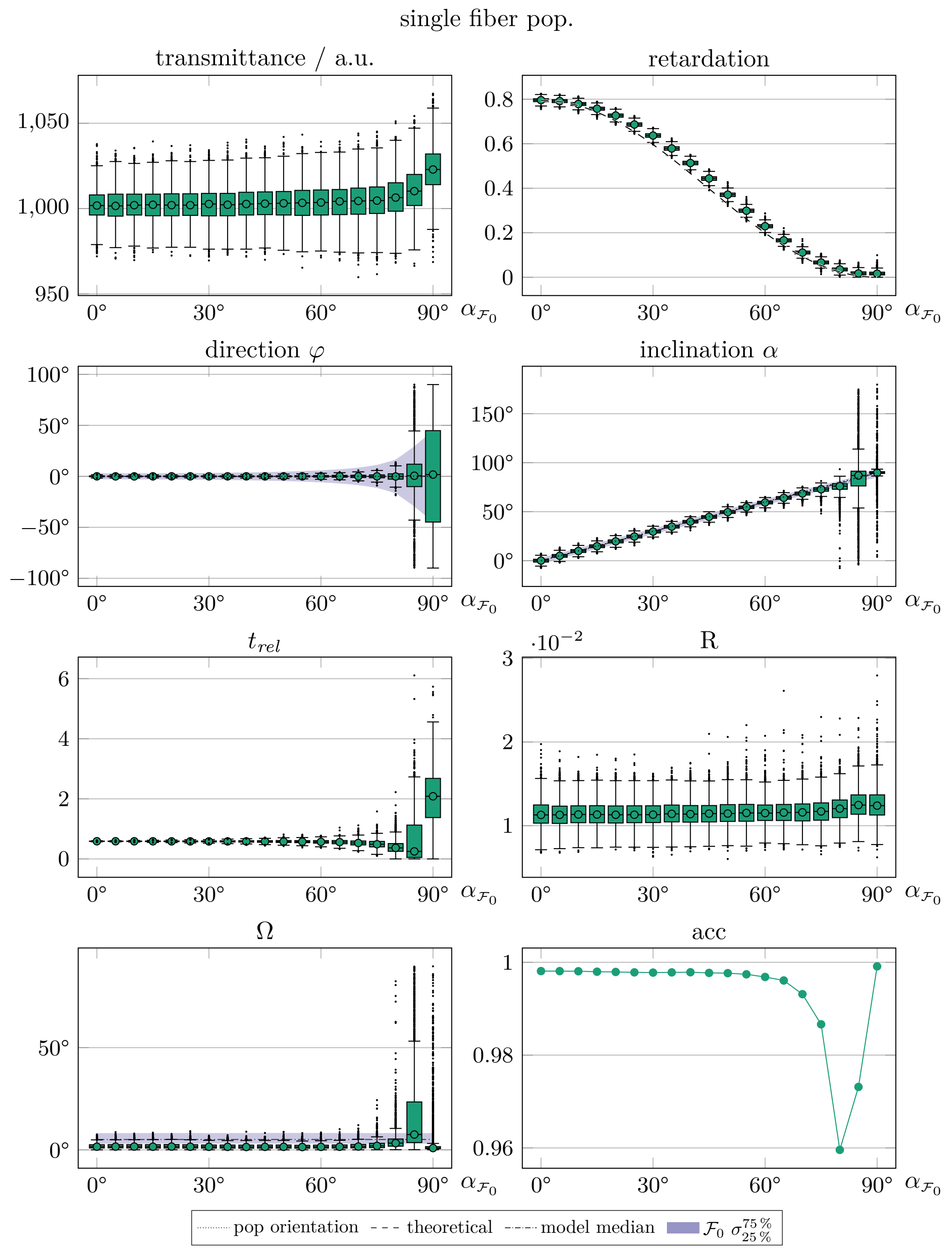


Fig. 9.6.: Single fiber population. Results of the simulation analysis with comparison of the fiber models orientations.

fiber models is plotted. The measured directions are in agreement with the models variance.

The inclination shows the results of the tilt analysis. The median follows the theoretical curve. For inclinations 80° and 85°, the variance increases. For $\alpha_{\mathcal{F}_0} = 90°$ the variance is similar to the flat results. The models $\sigma_{25\,\%}^{75\,\%}$ variance is constant and with an exception of $\alpha_{\mathcal{F}_0} = 75°$ higher than the tilt analysis results.

The mean value of the relative birefringence thickness t_{rel} is stable for values up to $\alpha_{\mathcal{F}_0} < 60°$. The variance increases slightly with increasing inclination angle $\alpha_{\mathcal{F}_0}$. From there on the t_{rel} value decreasing up to an inclination angle of $\alpha_{\mathcal{F}_0} = 85°$ and increases significant for $\alpha_{\mathcal{F}_0} = 90°$. The variance for $(85, 90)°$ is significantly increased and contains values as outliers $\gg 1$.

The R-value remains constant for inclination angles $\alpha_{\mathcal{F}_0} < 80°$. In the case of $\alpha_{\mathcal{F}_0} = (80 \text{ to } 90)°$ the values are slightly increased. Its variance remains stable for all inclinations.

The opening angle Ω is constant up to an inclination angle of $\alpha_{\mathcal{F}_0} \leq 70°$. For higher inclination angles, both the median and the variance increase significantly. In the case of $\alpha_{\mathcal{F}_0} = 90°$ the median as well as the variance is very low. In comparison, the fiber models opening angle Ω shows a significant higher median and higher $\sigma_{25\,\%}^{75\,\%}$ variance, except for $\alpha_{\mathcal{F}_0} = 85°$.

The acc-value remains close to 1 and starts to decrease significantly at an inclination angle of about 60°. It reaches a minimum at an inclination angle of 80°, from it increases again. At an inclination angle of 90° the maximum acc-value is reached.

Additional data is available in fig. B.4.

9.3.3 Flat crossing fiber populations

This simulation focuses on two flat crossing fiber populations, i. e., in the x-y-plane. Figures 9.7 and 9.8 shows the distribution of orientations for the model fiber segments and the resulting tilt analysis orientations for $\Psi_{\mathcal{F}_0} = 30\,\%$ and $\Psi_{\mathcal{F}_0} = 50\,\%$. The additional fiber population fractions are available in figs. B.6 to B.10.

The results show that the measured orientation follows the dominant fiber population. Depending on the fiber population fraction $\Psi_{\mathcal{F}_0}$, the resulting orientation is closer or further away from one of the fiber populations. In the case of $\Psi_{\mathcal{F}_0} = 50\,\%$, the measured orientation lies between the two fiber populations, except for a crossing angle

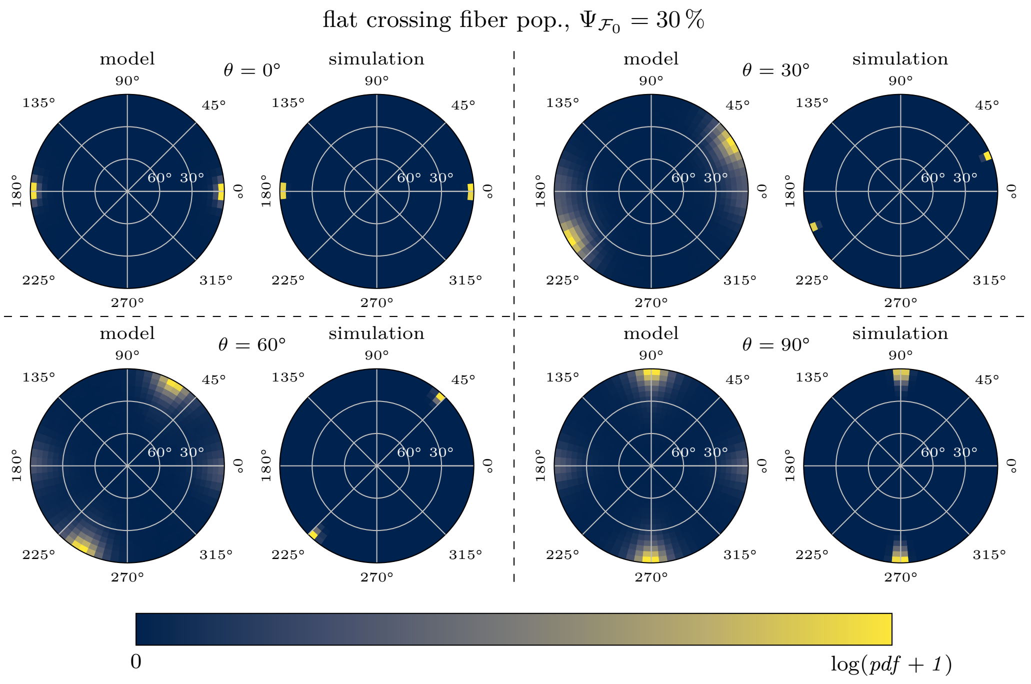


Fig. 9.7.: Flat crossing fiber population: $\Psi_{\mathcal{F}_0} = 30\,\%$.

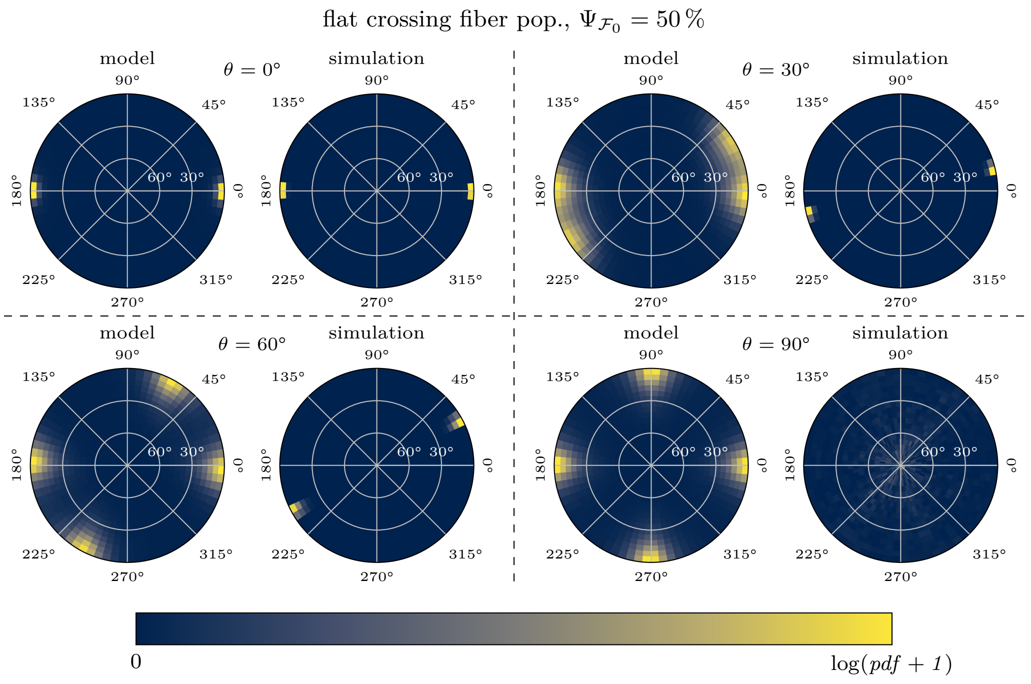


Fig. 9.8.: Flat crossing fiber population: $\Psi_{\mathcal{F}_0} = 50\,\%$.

of $\theta = 90°$, where the measured orientations are randomly distributed around steep orientations.

Figures 9.9 and 9.10 show the results of the 3D-PLI modalities, the tilt analysis, opening angle Ω and acc-value in detail.

The transmittance increases with increasing crossing angle θ. However, the rate of increase becomes flatter with increasing crossing angle. In the case of $\Psi_{\mathcal{F}_0} = 30\,\%$, the transmittance increases about $9\,\%$ whereas the increase for $\Psi_{\mathcal{F}_0} = 50\,\%$ is about $10\,\%$.

The retardation is negatively linearly correlated with increasing crossing angle. The retardation starts at 0.8 and drops in the case of For $\Psi_{\mathcal{F}_0} = 30\,\%$ to about 0.3. In the case of $\Psi_{\mathcal{F}_0} = 50\,\%$ it drops close to 0. The variance of the retardation is slightly increased for fiber population fractions close to $10\,\%$ and $90\,\%$ around $\theta = 45°$.

The direction analysis for all simulated fiber population fractions $\Psi_{\mathcal{F}_0}$ follows the theoretical *circmean* function. In the case for $\Psi_{\mathcal{F}_0} < 50\,\%$ the median is slightly higher than the theoretical prediction and in the case for $\Psi_{\mathcal{F}_0} > 50\,\%$ slightly lower. Depending on the fiber population fraction, the variance increases with the crossing angle. In the case of $\Psi_{\mathcal{F}_0} = 50\,\%$ and a crossing angle of $R_{\mathcal{F}_1} = 90°$, the directions are uniformly distributed $(-90$ to $90)°$. The distribution of the individual nerve fiber populations show the separation of the orientation with increasing crossing angle θ. The models $\sigma_{25\,\%}^{75\,\%}$ variance for both populations remains constant. The dominant fiber population has a slightly lower variance than its counterpart.

The inclination median is about 0° for all simulated parameters. The variance increases with increasing crossing angle θ and slightly for fiber population fractions close to $\Psi_{\mathcal{F}_0} = 50\,\%$. The models $\sigma_{25\,\%}^{75\,\%}$ variance of both fiber models increases with increasing crossing angle and is slightly larger for the dominant fiber population. Compared to the simulation results the $\sigma_{25\,\%}^{75\,\%}$ variance is significantly higher.

The relative birefringence thickness t_{rel} decreases with increasing crossing angle θ. The shape of the curve correlates negatively with an increasing crossing angle. The t_{rel} values start at about 0.6 and decrease to $t_{rel} = 0.2$ for $\Psi_{\mathcal{F}_0} = 30\,\%$ and to nearly $t_{rel} = 0$ for $\Psi_{\mathcal{F}_0} = 50\,\%$ depending on the fiber population fraction. For both fiber population fractions, the variance increases slightly. For $\Psi_{\mathcal{F}_0} = 50\,\%$ and $\theta = 90°$, the variance of t_{rel} increases significantly and outliers occur reaching t_{rel} values > 1.

The R-value is nearly constant for all crossing angles θ and fiber population fractions $\Psi_{\mathcal{F}_0}$. The variance shows no significant change.

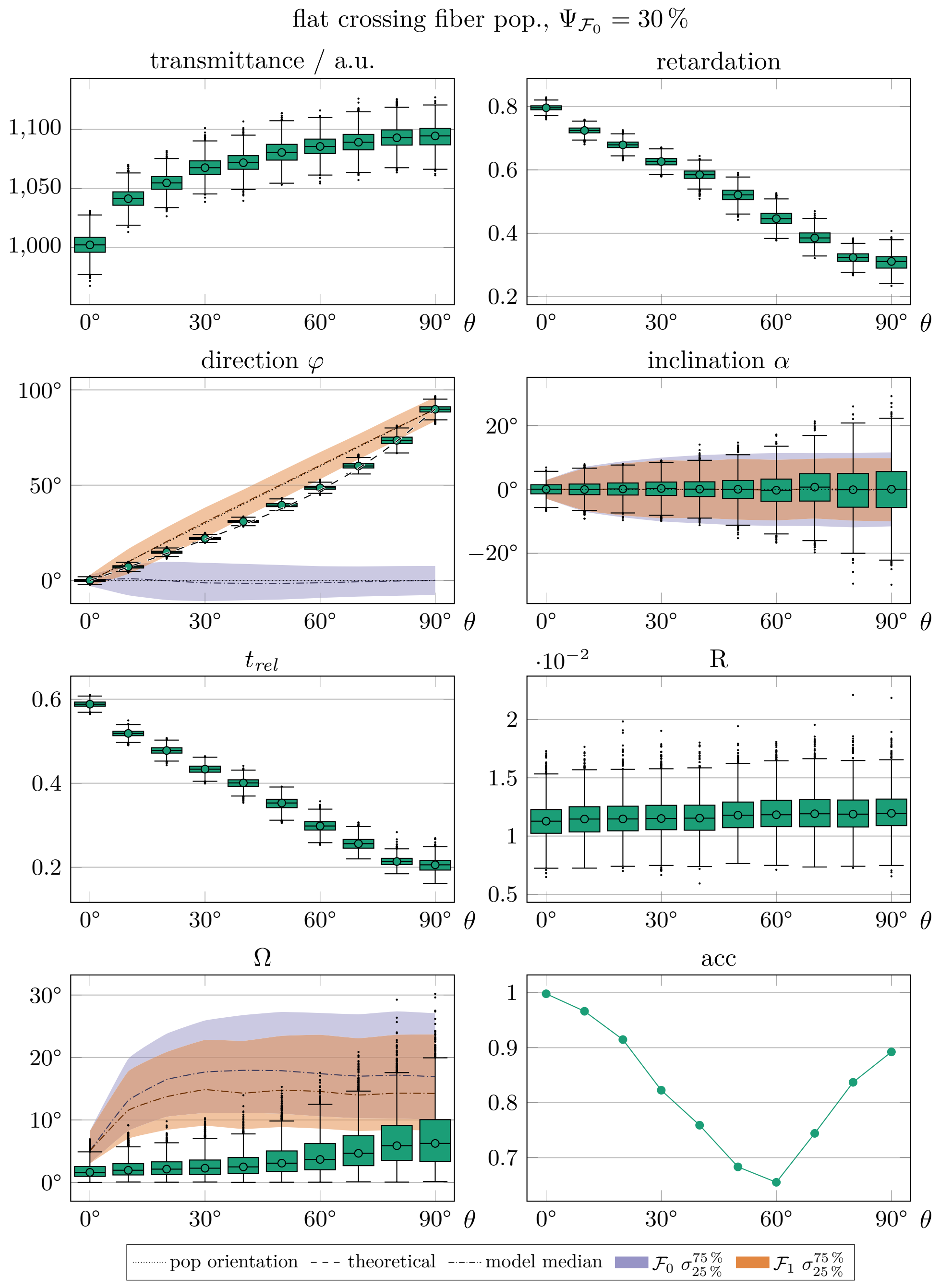


Fig. 9.9.: Flat crossing fiber population: $\Psi_{\mathcal{F}_0} = 30\,\%$. Results of the simulation analysis with comparison of the fiber models orientations.

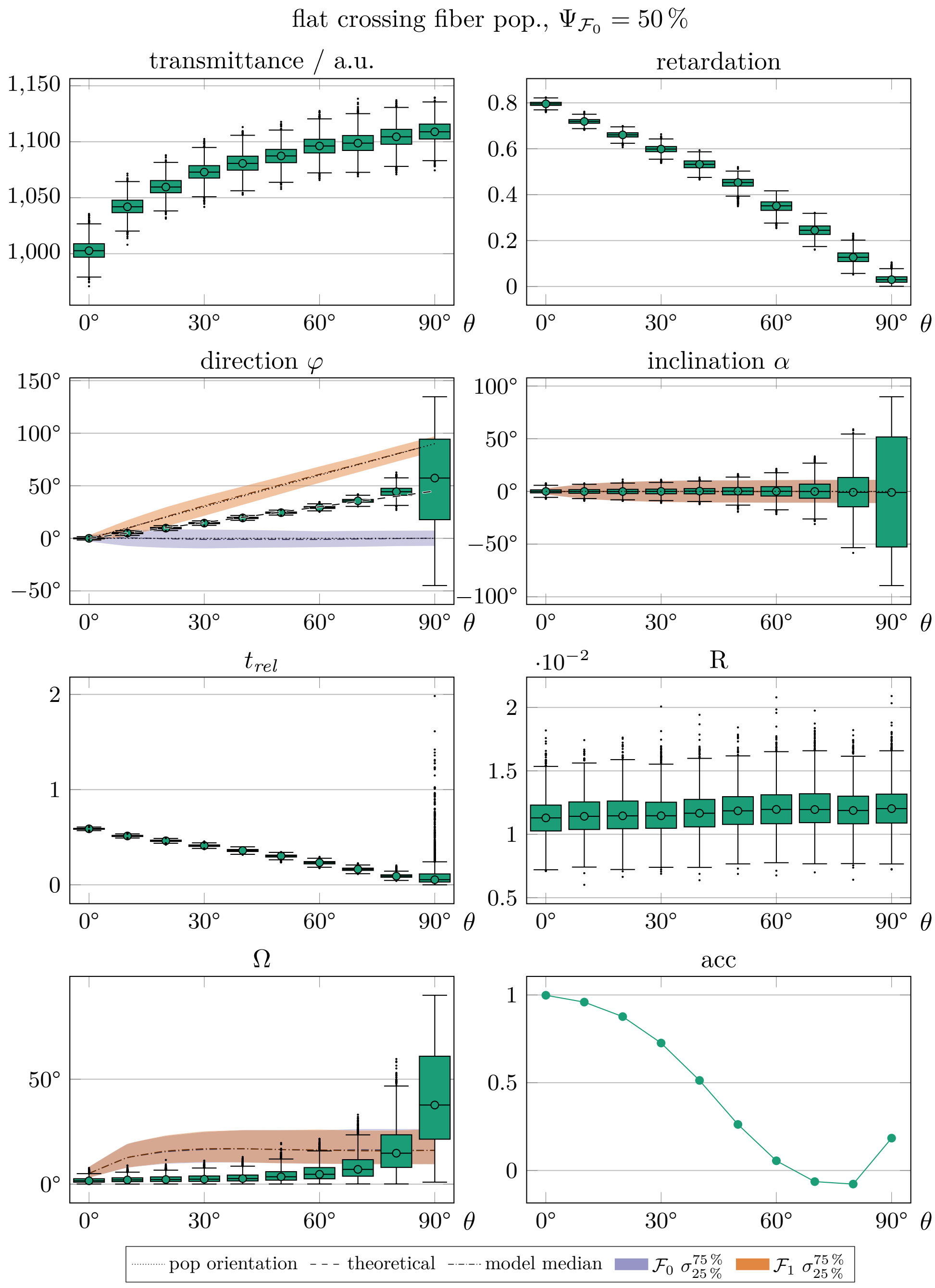


Fig. 9.10.: Flat crossing fiber population: $\Psi_{\mathcal{F}_0} = 50\,\%$. Results of the simulation analysis with comparison of the fiber models orientations.

The opening angle Ω is similar for all fiber population fractions up to a crossing angle of $\theta = 50°$. For higher crossing angles and fiber population fractions closer to $\Psi_{\mathcal{F}_0} = 50\,\%$, the median and variance of the opening angle increase significantly. The fiber models median Ω is significantly higher than the results from the tilting analysis. The dominant fiber population has a lower median and $\sigma_{25\,\%}^{75\,\%}$ variance than its counterpart.

The acc-value starts close to a value of 1 and decreases up to a crossing angle of about $\theta \approx 60°$ depending on the fiber population fraction $\Psi_{\mathcal{F}_0}$. After the minimum is reached it starts to increase significantly and with the exception for a fiber population fraction of $\Psi_{\mathcal{F}_0} = 50\,\%$ reaching a value closer to 1 again.

Additional data is available in figs. B.5 to B.10.

9.3.4 Inclined crossing fibers population

In this model configuration, the first fiber population $\mathcal{F}_0$ is placed along the x-axis and the second fiber population is inclined along the x-z axis. The second's fiber population inclination angle $\alpha_{\mathcal{F}_1}$ will be identical with the crossing angle θ between both fiber populations.

The orientation histograms of the inclined models and their simulation results are shown in figs. 9.11 and 9.12 and the remaining results are available in fig. B.11. The orientation of the tilt analysis of the simulation follows the inclination of the models, with the exception for an inclination angle $\alpha_{\mathcal{F}_1} = 90°$. The orientation is shifted towards the more dominant fiber population. The fiber models orientations show a higher variance as the tilt analysis results.

Figures 9.13 and 9.14 show the results of 3D-PLI modalities, the tilt analysis, the opening angle and the acc-value in detail for a fiber population of $(30, 50)\,\%$.The remaining fiber population fraction are listed in figs. B.12 to B.16.

The transmittance increases with increasing inclination angle $\alpha_{\mathcal{F}_1}$. Fiber population fraction close to $50\,\%$ tend to have a higher transmittance value. The increase of the transmittance slows down for higher inclination angles $\alpha_{\mathcal{F}_1}$.

The retardation is negatively linearly correlated with increasing inclination angle $\alpha_{\mathcal{F}_1}$. The slope of the negative correlation is higher with increasing fiber population fraction $\Psi_{\mathcal{F}_0}$. The variance increases as the inclination angle increases and increasing fiber population fraction.

The direction value is centered around 0°. The variance of the direction increases with increasing inclination angle $\alpha_{\mathcal{F}_1}$. The fiber population fraction has no significant effect

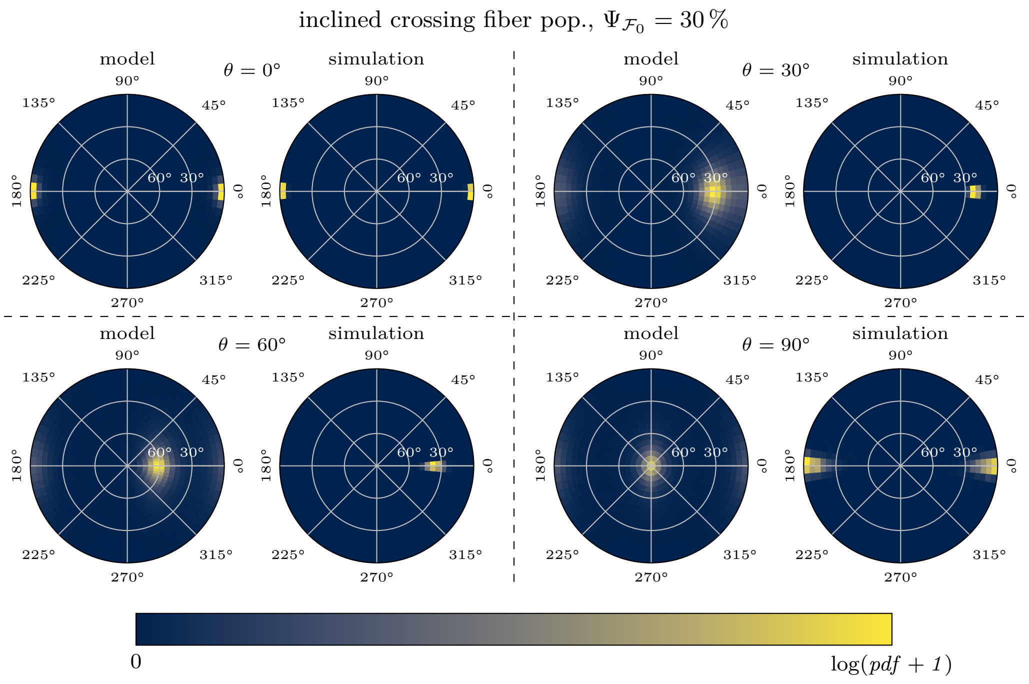


Fig. 9.11.: Inclined crossing fiber population: $\Psi_{\mathcal{F}_0} = 30\,\%$.

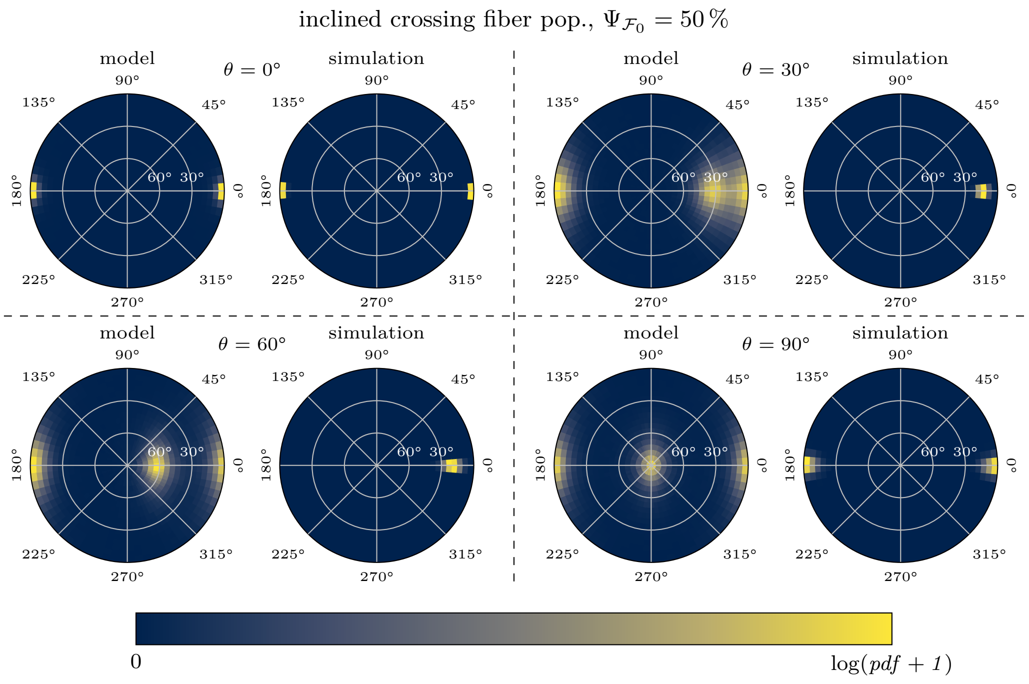


Fig. 9.12.: Inclined crossing fiber population: $\Psi_{\mathcal{F}_0} = 50\,\%$.

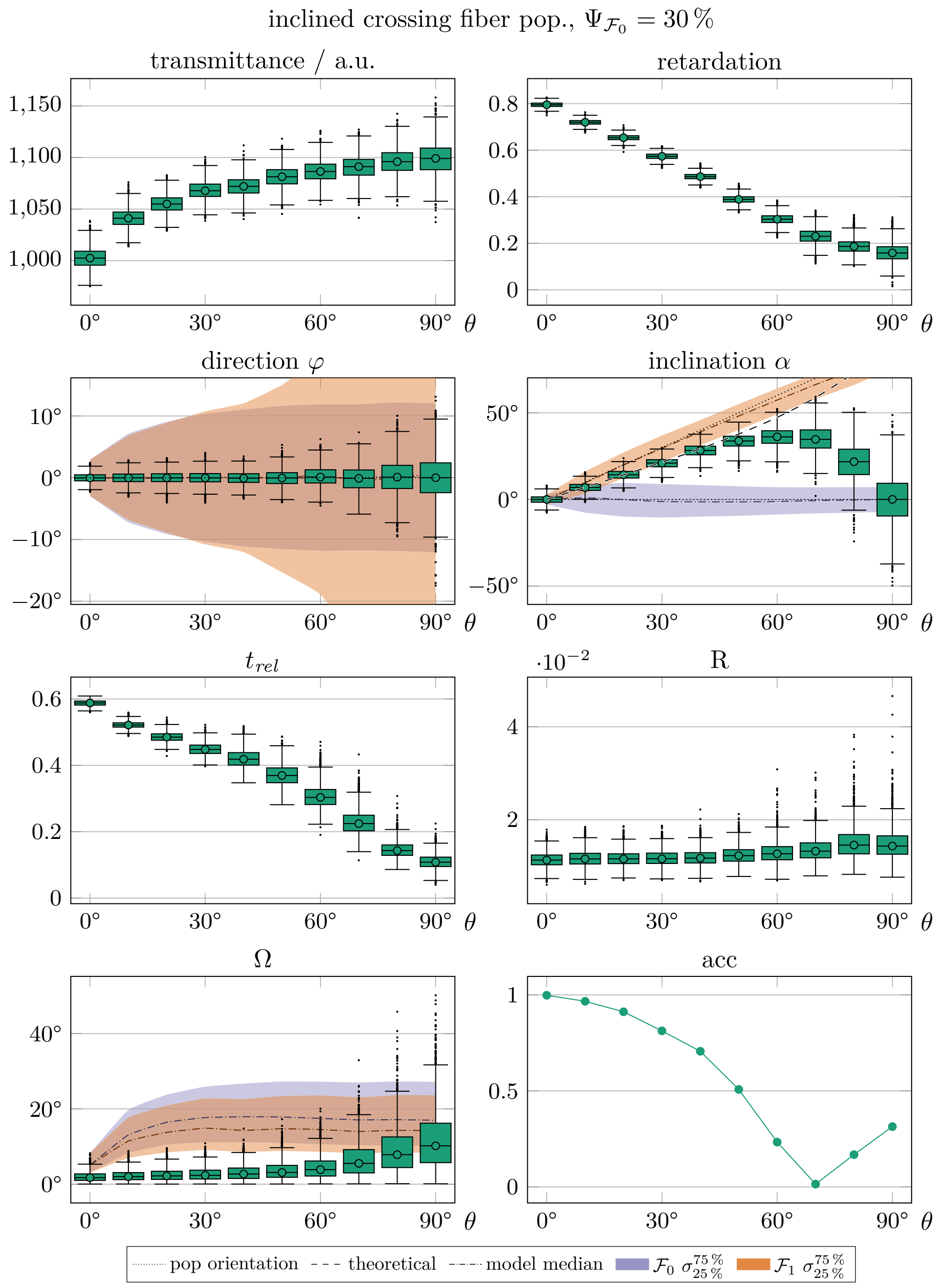


Fig. 9.13.: Population of inclined crossing fibers: $\Psi_{\mathcal{F}_0} = 30\,\%$. Results of the simulation analysis with comparison of the fiber models orientations.

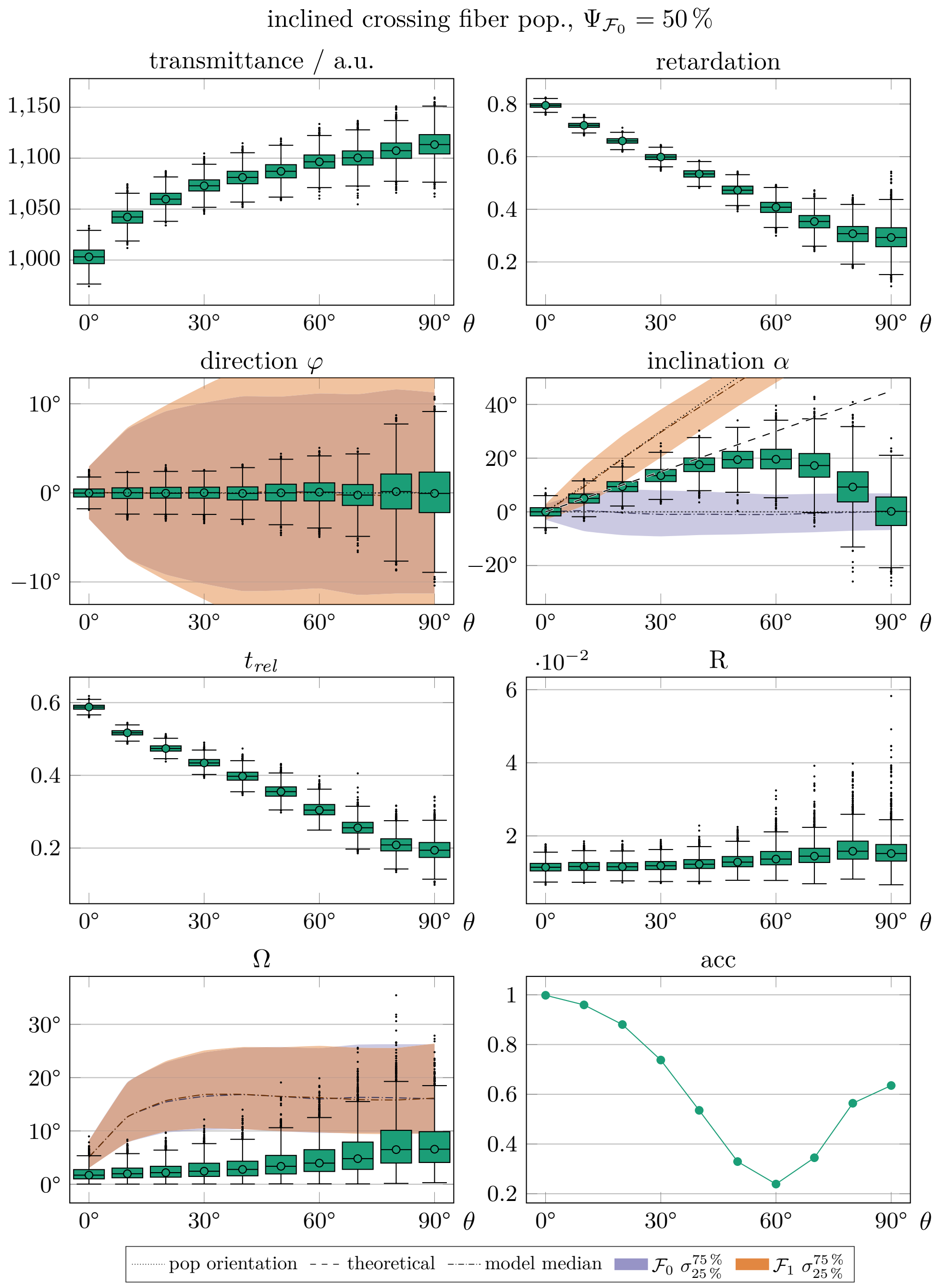


Fig. 9.14.: Population of inclined crossing fibers: $\Psi_{\mathcal{F}_0} = 50\,\%$. Results of the simulation analysis with comparison of the fiber models orientations.

except for a value of $\Psi_{\mathcal{F}_0} = 10\,\%$ and an inclination angle of $\alpha_{\mathcal{F}_1} = 90\,\%$. The models $\sigma_{25\,\%}^{75\,\%}$ variance of the first (flat) fiber population $\mathcal{F}_0$ is always smaller than the $\sigma_{25\,\%}^{75\,\%}$ variance of the second fiber population $\mathcal{F}_1$. The $\sigma_{25\,\%}^{75\,\%}$ variance for $\mathcal{F}_0$ increases up to an inclination angle of about $\alpha_{\mathcal{F}_1} = 50°$ and remains constant for higher inclination angles, whereas the $\sigma_{25\,\%}^{75\,\%}$ variance of the second fiber bundle increases further.

The inclination results have the shape of a convex curve, starting and ending at a value of $\alpha = 0°$. Depending on the fiber population fraction $\Psi_{\mathcal{F}_0}$ the maximum of the curve is shifted towards an inclination angle of 90° for a low $\Psi_{\mathcal{F}_0}$ and towards 45° for a high $\Psi_{\mathcal{F}_0}$. The $\sigma_{25\,\%}^{75\,\%}$ of the individual nerve fiber populations remain constant for inclination angles $\geq 10°$. For lower fiber population fractions $\Psi_{\mathcal{F}_0}$ and higher inclination angles $\alpha_{\mathcal{F}_1}$ the variance of the simulation results is higher than the $\sigma_{25\,\%}^{75\,\%}$ of the models. Otherwise, the simulation's inclination variance is significantly lower.

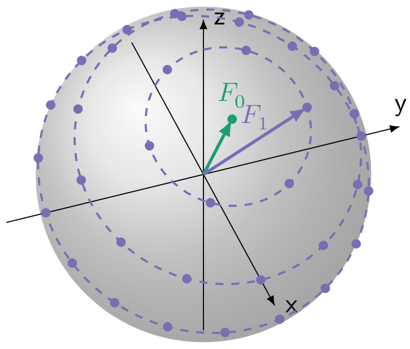
The relative thickness t_{rel} decreases linearly with increasing inclination angle $\alpha_{\mathcal{F}_1}$. The t_{rel} value start at about 0.6 and drops depending on the fiber population fraction $\Psi_{\mathcal{F}_0}$ to 0.45 for $\Psi_{\mathcal{F}_0} = 90\,\%$ to almost 0 for $\Psi_{\mathcal{F}_0} = 10\,\%$. The variance of the t_{rel} value is increased at about $\alpha_{\mathcal{F}_1} = 45°$ for lower fiber population fractions and towards $\alpha_{\mathcal{F}_1} = 90°$ for high fiber population fractions.

The R-value shows no significant effect for low fiber population fraction. Higher fiber population fractions have an increased R-value. With increasing inclination angle $\alpha_{\mathcal{F}_1}$ the R-value and its variance increases slightly for lower fiber population fractions and more strongly for higher fiber population fractions.

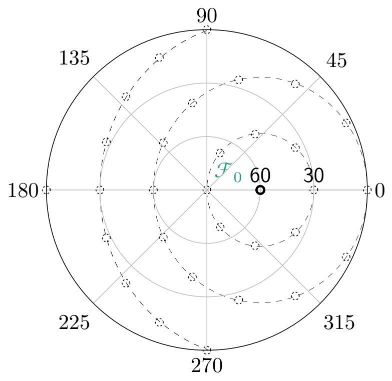
The measured mean value of the opening angle Ω and its variance increases slightly with increasing inclination angle. For lower fiber population fractions and a crossing angle $\alpha_{\mathcal{F}_1} \geq 80°$ the increase is significantly more. The fiber model opening angles are with the median as well as the $\sigma_{25\,\%}^{75\,\%}$ variance significantly higher than the simulation results for both fiber population fractions. The median and $\sigma_{25\,\%}^{75\,\%}$ variance of the opening angle of the dominant fiber population is higher than its counterpart.

The acc-value shows a decrease with increasing inclination angle to a minimum value and from there the acc-value raises again. The position of the minimum strongly depends on the fiber population fraction. For lower $\Psi_{\mathcal{F}_0}$ the minimum is close to higher inclination angles and for higher $\Psi_{\mathcal{F}_0}$ the minimum moves to an inclination angle of about $\alpha_{\mathcal{F}_1} \approx 45°$. The value of the acc-value is closer to zero for median fiber population fractions and closer to one for $\Psi_{\mathcal{F}_0}$ at $10\,\%$ and $90\,\%$.

Additional data is available in figs. B.11 to B.16.



(a) The spheres surface is sampled around the first fiber population axis $\mathcal{F}_0$ of inclination $\alpha_{\mathcal{F}_0}$ with a crossing angle θ and a rotation $R_{\mathcal{F}_1}$.



(b) A polar diagram is used to illustrate the scalar results of a fixed orientation of the first fiber population $\mathcal{F}_0$ and a variable orientation of the fiber population $\mathcal{F}_1$. The orientation $\mathcal{F}_0$ is marked with a thick black circle. The second orientation is indicated by dashed circles. For visualization, the scalar values are interpolated according to a spherical k-nearest neighbor interpolator algorithm.

Fig. 9.15.: Visualization of results for a fixed first fiber population $\mathcal{F}_0$.

9.3.5 Free crossing fiber populations

In this section, two nerve fiber populations $\mathcal{F}_0$ and $\mathcal{F}_1$ are simulated with sampled orientations on a sphere (see fig. 9.15a and table 9.1). A polar plot visualization is used to visualize the results from the 3D-PLI analysis pipeline (see fig. 9.15b). The spherical plots in figs. 9.16 to 9.21 and 9.23 are designed so that the thick black circle shows the orientation of the first fiber population $\mathcal{F}_0$ and the thin dashed circles show the orientation of the second fiber population $\mathcal{F}_1$. At the position of the second fiber population, the resulting average value is visualized. A spherical k-nearest neighbor interpolator is used to visualize values between the sampled orientations for nonangular data [Tre19].

Transmittance The average transmission value is visualized in fig. 9.16. The first fiber population fraction $\Psi_{\mathcal{F}_0} = 10\,\%$ is almost identical for all inclination angles $\alpha_{\mathcal{F}_0}$. The transmittance is lowest for models with a crossing angle of $\theta = 0°$ and for flat configurations.

For a fiber population fraction of $\Psi_{\mathcal{F}_0} = 30\,\%$, the transmittance increases. The same behavior holds for $\Psi_{\mathcal{F}_0} = (50, 70)\,\%$. $\Psi_{\mathcal{F}_0} = 50\,\%$ achieves the highest transmittance for fiber models with crossings.

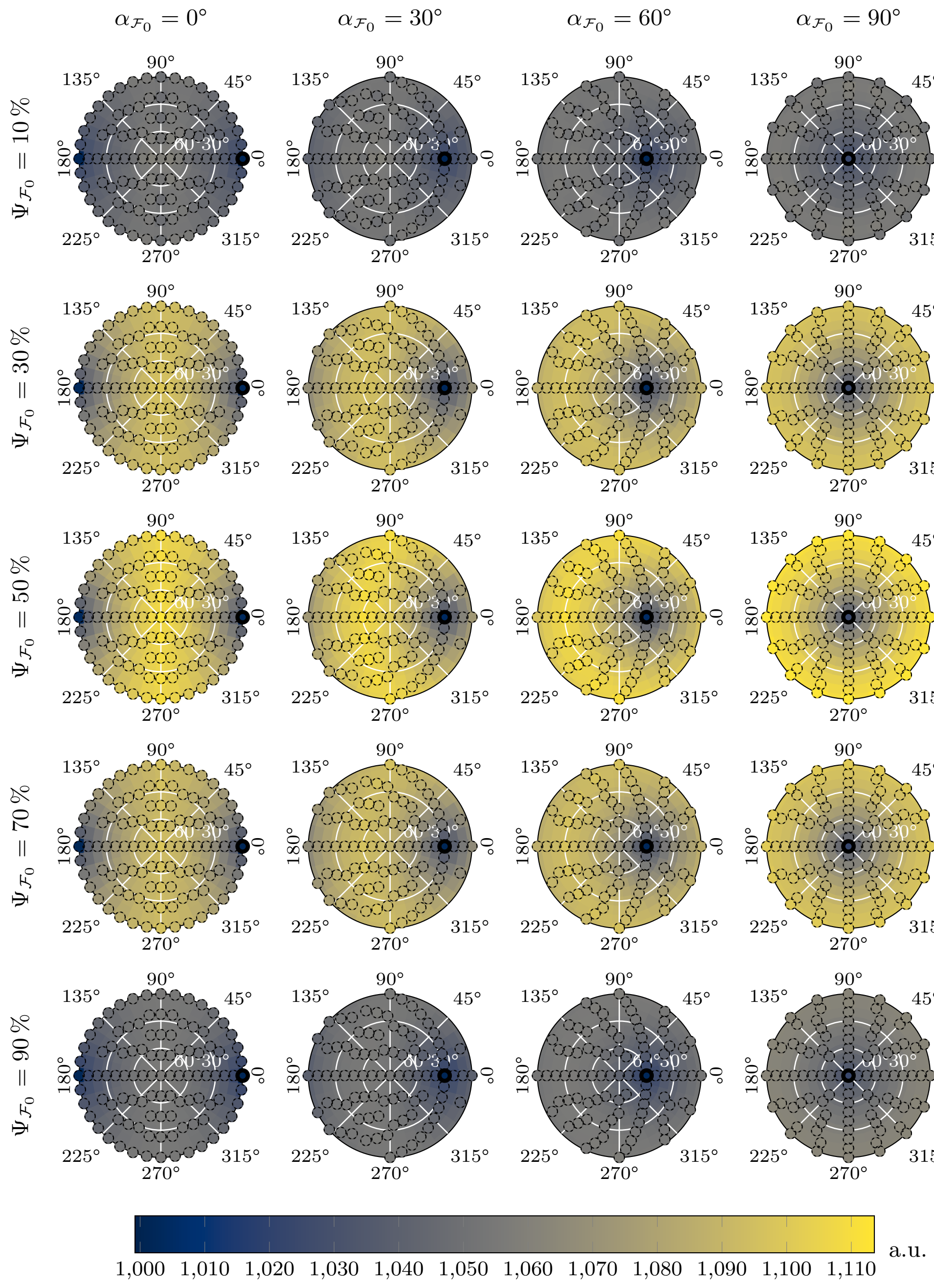


Fig. 9.16.: Mean transmittance value.

The average transmittance for the last fiber population fraction $\Psi_{\mathcal{F}_0} = 90\,\%$ is reduced similar to $\Psi_{\mathcal{F}_0} = 10\,\%$. Fiber configurations with no crossing have the lowest transmittance for all inclinations $\alpha_{\mathcal{F}_0}$.

Retardation Retardation results are shown in fig. 9.17. For a fiber population fraction of $\Psi_{\mathcal{F}_0} = 10\,\%$, the lowest retardation values are present for secondary fiber populations with an inclination of 90°. The retardation for configurations with a high crossing angle θ are slightly reduced. Flat fiber configurations retain a relative high retardation value.

The same behavior is also visible for fiber population fraction $\Psi_{\mathcal{F}_0} = (30,\ 50, 70)\,\%$, but the retardation values are significantly reduced, especially for high first fiber population inclinations $\alpha_{\mathcal{F}_0}$. The reduction in retardation at high crossing angles θ is also more pronounced, especially for $\Psi_{\mathcal{F}_0} = 50\,\%$. For fiber population fractions $\Psi_{\mathcal{F}_0} \geq 50\,\%$ and first fiber population inclinations $\alpha_{\mathcal{F}_0} \geq 60°$, the retardation is overall very low.

For $\Psi_{\mathcal{F}_0} = 90\,\%$, no reduction of retardation is visible for any secondary fiber orientation. The retardation decreases significantly with increasing inclination $\alpha_{\mathcal{F}_0}$ until it is close to 0 for $\alpha_{\mathcal{F}_0} = 90°$.

Direction φ and inclination α The results of the circular mean of direction are shown in fig. 9.18 and the circular mean of the inclination in fig. 9.19. An interpolation method is not used for the angular values because the required interpolation on a hemispherical manifold can not be easily accomplished. The k-nearest neighbor interpolator used above would lead to incorrect results.

Both angles show a value pattern following the dominant fiber population. For steep fibers with a $\alpha_{\mathcal{F}_0} = 90°$, the orientation results follow the second fiber population orientation, regardless of the fiber population fraction $\Psi_{\mathcal{F}_0}$. For fiber population fractions close to 50 %, the resulting orientations behave according to an averaging of the two individual orientations with respect to the fiber population fraction. From the direction it can be seen that for models with $\Psi_{\mathcal{F}_0} = 90°$ and $\alpha_{\mathcal{F}_0} \leq 60°$ as well as $\Psi_{\mathcal{F}_0} = 70°$ and $\alpha_{\mathcal{F}_0} \leq 30°$ the second fiber population has no or only a very small influence on the measured direction. The same is true for inclinations for a $\alpha_{\mathcal{F}_0} = 0°$ and $\Psi_{\mathcal{F}_0} \geq 50\,\%$.

Effective birefringence thickness t_{rel} The results of t_{rel} are shown in fig. 9.20. The t_{rel} value for the first fiber population fraction $\Psi_{\mathcal{F}_0} = 10\,\%$ is nearly constant for all fiber configurations. The exceptions are $\alpha_{\mathcal{F}_0} = (60, 90)°$ with secondary fiber population inclinations of $\alpha_1 = 90°$.

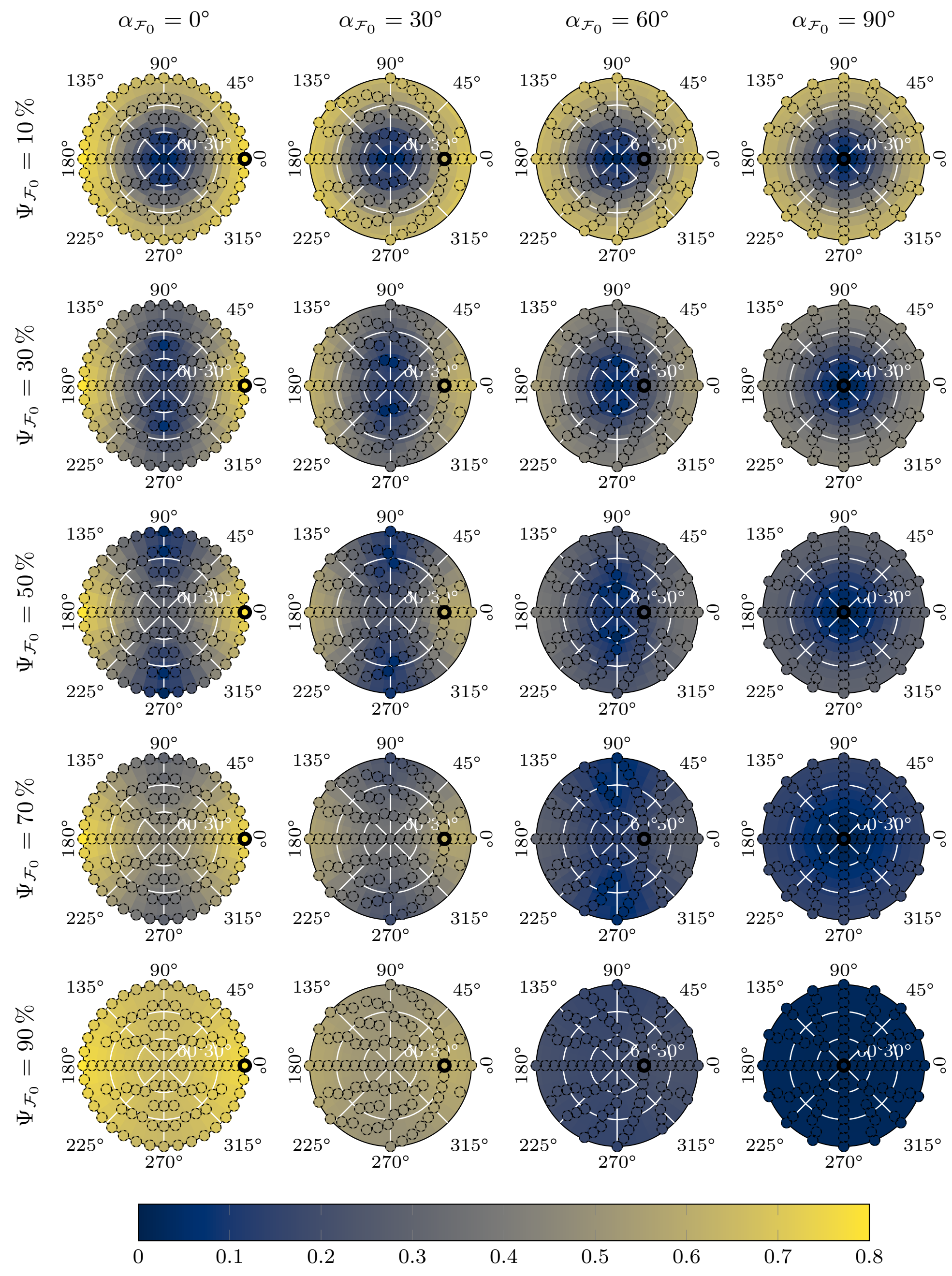


Fig. 9.17.: Mean retardation values.

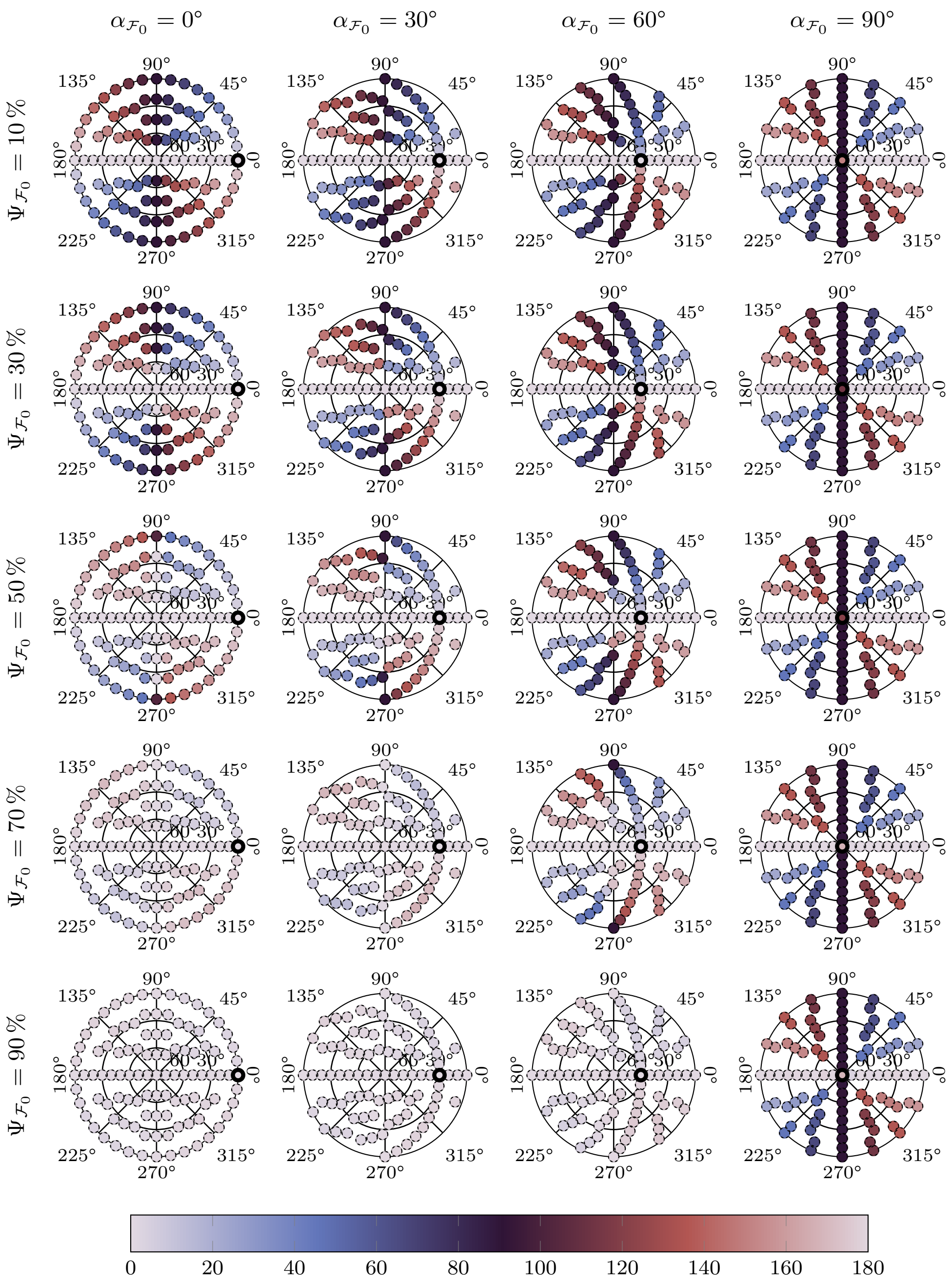


Fig. 9.18.: Circmean direction φ from tilt analysis.

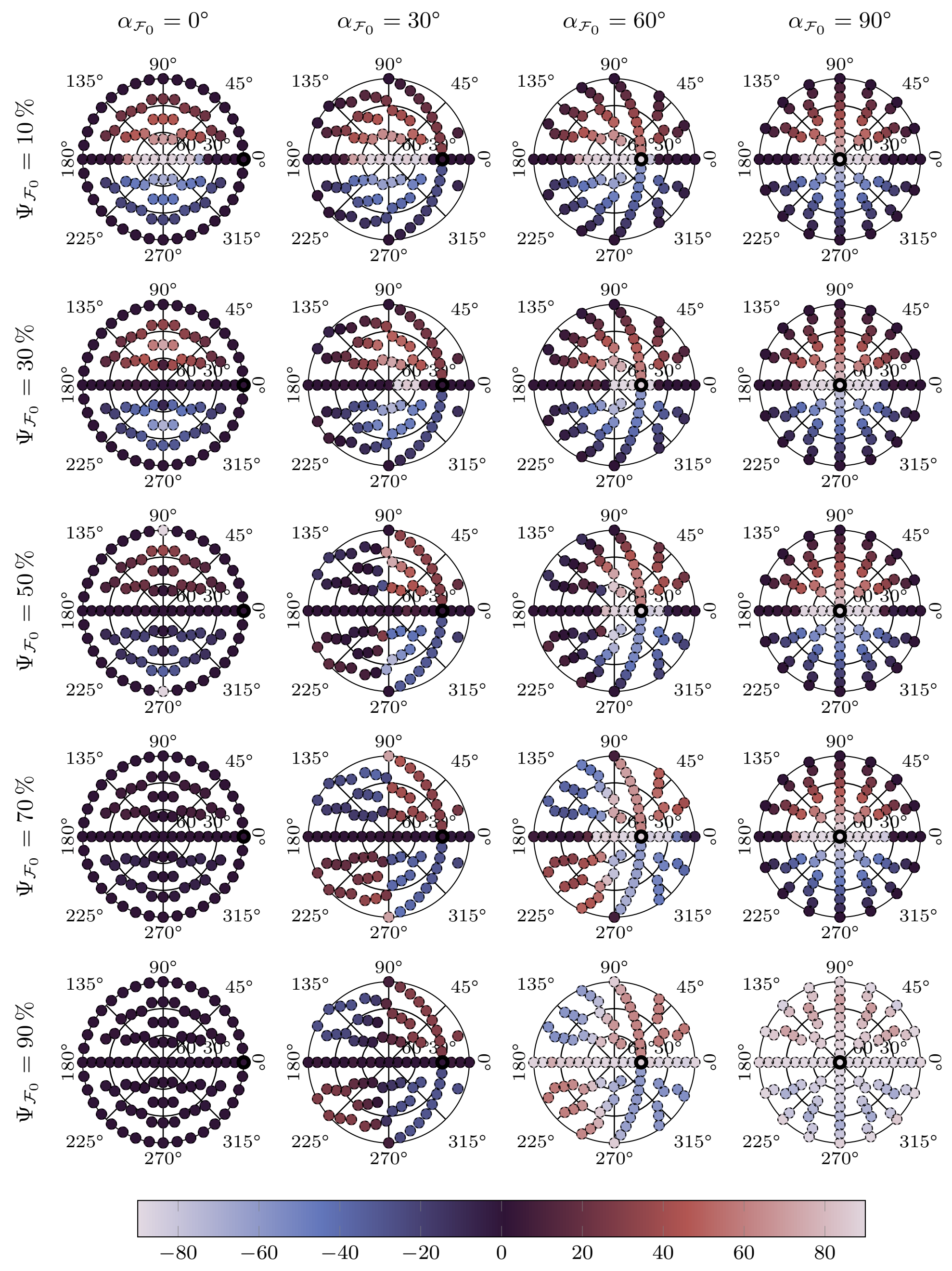


Fig. 9.19.: Circmean inclination α from tilt analysis. Because the direction is defined in the range of $[0°, 180°)$ and the inclination $[-90°, 90°)$ sign of the inclination value flips for the lower quadrants in the polar plots.

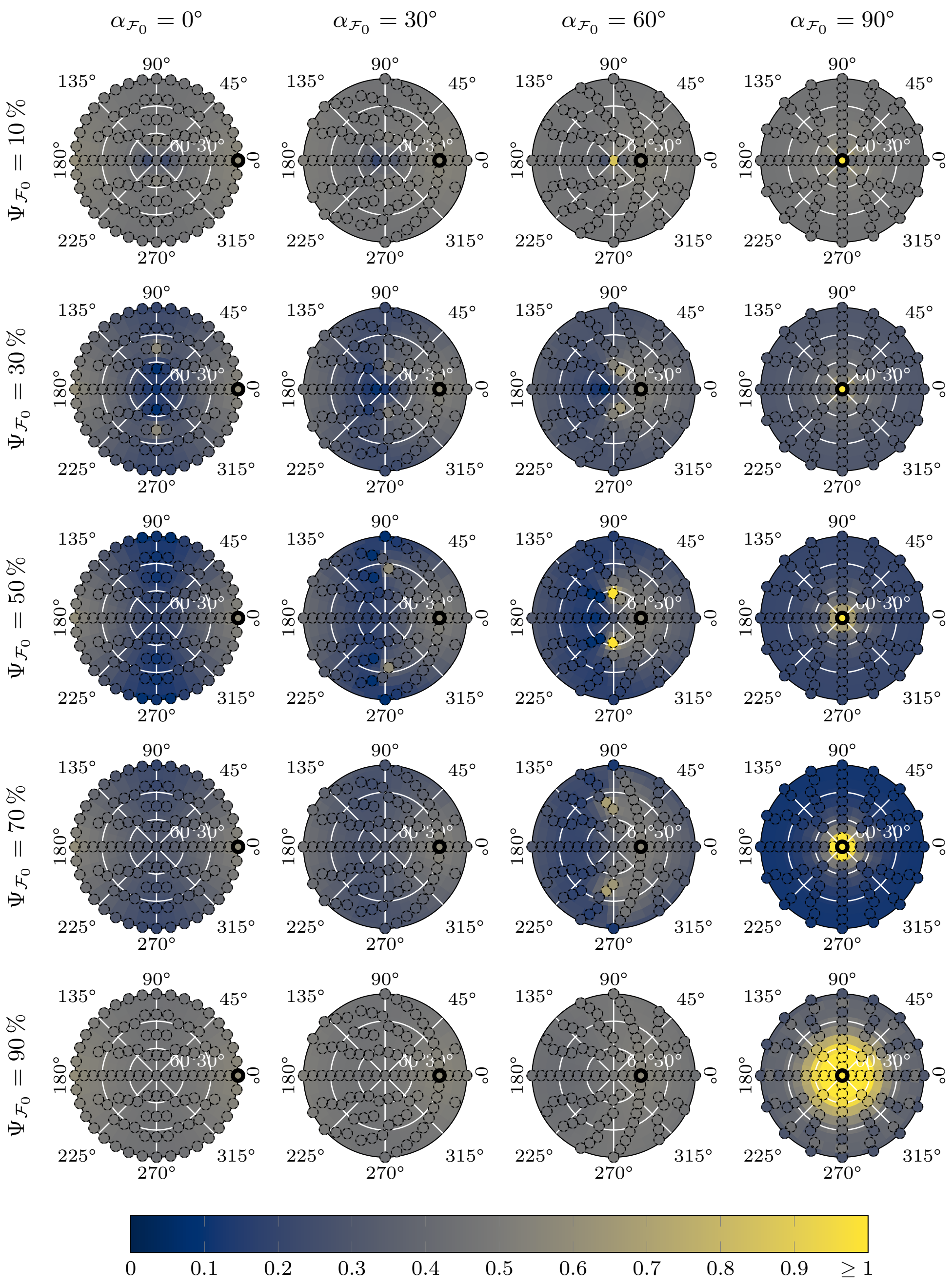


Fig. 9.20.: Mean t_{rel} from tilt analysis.

Areas of reduced t_{rel} become visible for $\Psi_{\mathcal{F}_0} = 30\,\%$. For $\alpha_{\mathcal{F}_0} = 0°$ this is located at a secondary fiber inclination of $\alpha_1 = 90°$. For $\alpha_{\mathcal{F}_0} = (30, 60)°$ the reduction of t_{rel} is slightly towards the negative x-axis next to the z-axis. For $\alpha_{\mathcal{F}_0} = 90°$ the t_{rel} value remains constant with the exception for both fiber populations orientated along the z-axis, where $t_{rel} \geq 1$.

For a fiber population fraction of $\Psi_{\mathcal{F}_0} = 50\,\%$ a band of reduction in the t_{rel} value is visible for a crossing angle of $\theta = 90°$. This "band" shaped area of reduced t_{rel} values is moved towards the negative x-axis for $\alpha_{\mathcal{F}_0} = (30, 60)°$. Additionally for $\alpha_{\mathcal{F}_0} = 60°$ two significantly increased t_{rel} values ≥ 1 are visible at $\varphi = (90, 270)°$ and $\alpha = 60°$. For $\alpha_{\mathcal{F}_0} = 90°$ the t_{rel} value remains constant, but more decreased than before, with the exception for both fiber populations orientated along the z-axis, where the t_{rel} value is ≥ 1.

$\Psi_{\mathcal{F}_0} = 70\,\%$ shows the same behavior as for the $\Psi_{\mathcal{F}_0} = 50\,\%$ case. However, the reduction is not as strong as in the previous case. The exception is for very steep primary fibers, where the t_{rel} values are significantly reduced. Again, the t_{rel} value along the z-axis, and its four closest neighbors are ≥ 1.

The last line shows the results for the fiber population fraction $\Psi_{\mathcal{F}_0} = 90\,\%$. All models and configurations remain at a stable t_{rel} value. The exception is the fiber configurations with $\alpha_{\mathcal{F}_0} = 90°$, where the t_{rel} values along the z-axis with inclinations $\alpha_1 \geq 45°$ are ≥ 1.

R-value The R-value of the tilt analysis fitting model is shown in fig. 9.21. It describes the mean absolute difference between the data and the fitted data from the tilt analysis (see eq. (9.3)).

For a fiber population fraction of $\Psi_{\mathcal{F}_0} = 10\,\%$ and a primary inclination angle $\alpha_{\mathcal{F}_0} \leq 30°$ the orientation of the second fiber population does not change the R-value of the tilt analysis. For $\alpha_{\mathcal{F}_0} = 60°$ the R-value starts increasing for secondary fiber orientations towards the xy-plane, i. e., secondary flat fiber orientations. The R-value increases even more significantly in the case of $\alpha_{\mathcal{F}_0} = 90°$ in this area.

For fiber population fractions $\Psi_{\mathcal{F}_0} = 30\,\%$ and $\alpha_{\mathcal{F}_0} = 0°$, the R-value remains low for flat crossing fiber configurations. Along the z-axis the R-value is slightly increased. For $\alpha_{\mathcal{F}_0} = 30°$ a slightly increased bend shape becomes visible close to the y-axis slightly towards the negative x-axis, i. e., $\theta \approx 90°$. Inclination angles of $(60, 90)°$ show the same behavior as for the $\Psi_{\mathcal{F}_0} = 10\,\%$ case, however the R-value does not increase as much.

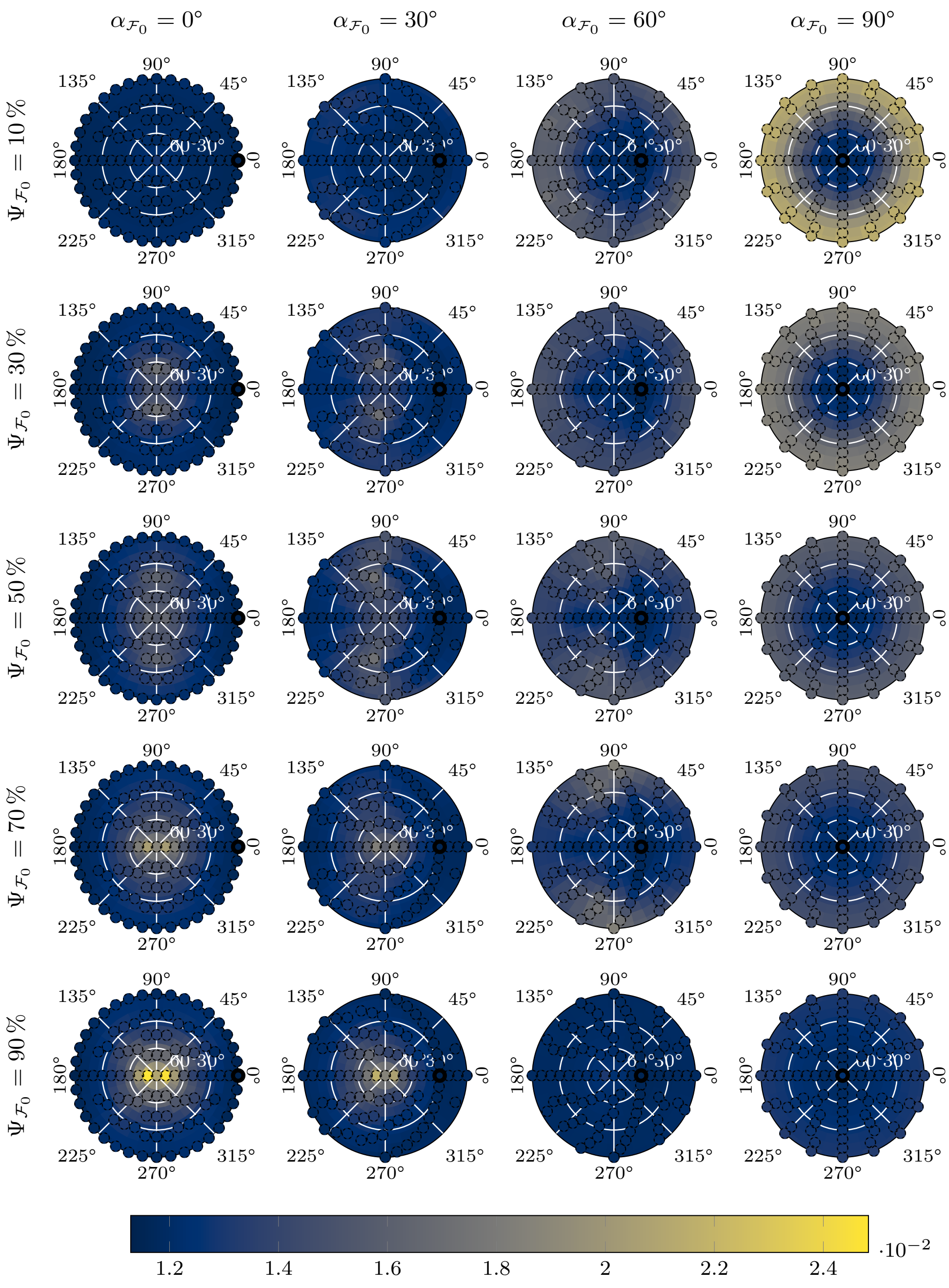


Fig. 9.21.: Mean R-value from tilt analysis.

Fiber population fractions $\Psi_{\mathcal{F}_0} = 50\,\%$ and inclinations of $\alpha_{\mathcal{F}_0} = 0°$, have an increased R-value for inclined secondary fiber configurations along the y-axis. For $\alpha_{\mathcal{F}_0} = 0°$, the increased area along the y-axis is curved in the direction of 180°. $\alpha_{\mathcal{F}_0} = 60°$ leaves a low R-value along the x-axis and an area around the first fiber population orientation. The other secondary orientations are slightly increased. The steep first fiber population inclinations of $\alpha_{\mathcal{F}_0} = 90°$ again has an increased R-value to the flat secondary orientations. The R-value relative to the lower first fiber population fractions is further reduced.

$\Psi_{\mathcal{F}_0} = 70\,\%$ and $\alpha_{\mathcal{F}_0} = 0°$ have only an increased R-value for the secondary fiber configuration along the z-axis. The points along the x-axis are slightly more increased than along the y-axis. For $\alpha_{\mathcal{F}_0} = 30°$, the curved shape from the previous result almost disappears. Only the secondary fiber configurations along the z-axis remain increased. $\alpha_{\mathcal{F}_0} = 60°$ remains a low R-value along the x-axis. Only the crossings for flat directions of $(110, 250)°$ are increased. The final inclination angle of $\alpha_{\mathcal{F}_0} = 90°$ is again similar to the previous fiber population fraction, but the R-value is again lower for the increased areas.

The final fiber population fraction $\Psi_{\mathcal{F}_0} = 90\,\%$ has an increased R-value on the z-axis for inclination angles of $\alpha_{\mathcal{F}_0} = (0, 30)°$. $\alpha_{\mathcal{F}_0} = (60, 90)°$ remains a low R-value for all configurations.

Opening angle Ω The opening angle Ω is shown in fig. 9.22.

Overall the opening angle Ω shows relatively low values for most fiber population fractions $\Psi_{\mathcal{F}_0}$ and inclination angles $\alpha_{\mathcal{F}_0}$. For lower fiber population fractions than $\Psi_{\mathcal{F}_0} < 50\,\%$ the opening angles are reduced when the secondary fiber population is orientated along the z-axis.

A fiber population fraction of $\Psi_{\mathcal{F}_0} = 50\,\%$ with an inclination of $\alpha_{\mathcal{F}_0} = 0°$ has significantly increased opening angles for flat crossings. For a primary inclination of $\alpha_{\mathcal{F}_0} = 30°$ an increased opening angle is visible for a crossing angle of $\theta = 90°$. $\alpha_{\mathcal{F}_0} = 60°$ shows an area of increased opening angles close to the z-axis towards the negative x-axis. $\alpha_{\mathcal{F}_0} = 90°$ shows no increased values. For a fiber population fraction of $\Psi_{\mathcal{F}_0} = 70\,\%$ the inclination angle of $\alpha_{\mathcal{F}_0} = 60°$ shows an increase in the opening angle along flat crossings along the y-axis. $\alpha_{\mathcal{F}_0} = 90°$ shows an overall slightly increased value of the opening angle Ω.

For the last shown fiber population fraction $\Psi_{\mathcal{F}_0} = 90°$, the opening angle values are very small with the exception for the inclination angle of $\alpha_{\mathcal{F}_0} = 90°$ where for secondary fiber orientations towards the x-y-plane the opening angle significantly increases.

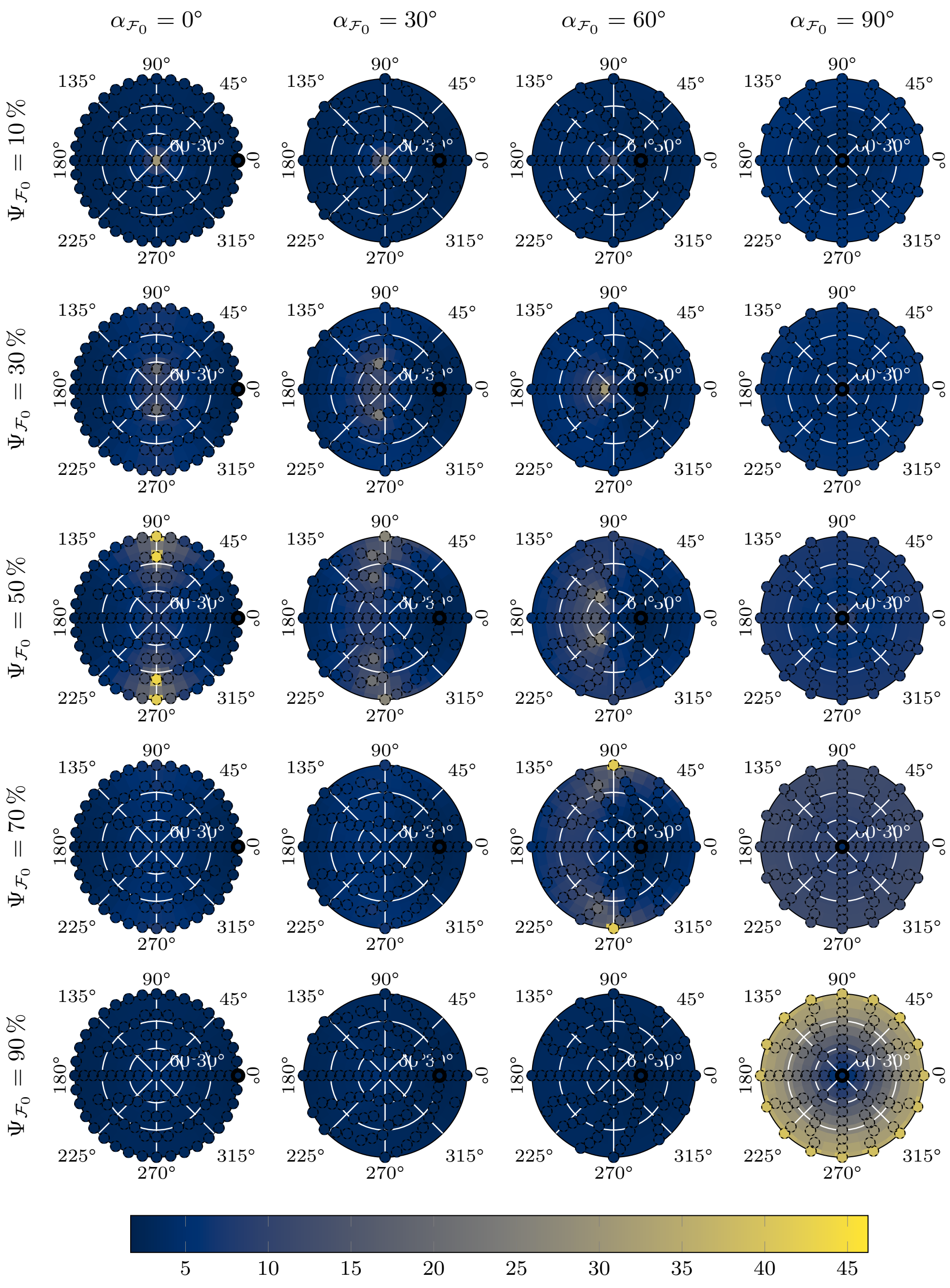


Fig. 9.22.: Mean opening angle Ω from tilt analysis.

ACC value Figure 9.23 shows the angular correlation coefficient (ACC), i. e., how well the coefficients of the ODF basis function are matched [Sch+18a].

The fiber population fraction $\Psi_{\mathcal{F}_0} = 10\,\%$ shows no decrease in the acc-value for the first fiber model inclination $\alpha_{\mathcal{F}_0}$. For $\Psi_{\mathcal{F}_0} = 30\,\%$, a reduction of the acc-value for the lower inclined models of the first fiber population is visible. The reduction is found in the region of the steep secondary fiber populations. For $\alpha_{\mathcal{F}_0} = (30, 60)°$, the reduction shifts in a direction along 180°. The effective area of the reduction decreases with increasing $\alpha_{\mathcal{F}_0}$ until no area effect is visible for an inclination of $\alpha_{\mathcal{F}_0} = 90°$. However, the acc-value for this last plot is still reduced compared to $\Psi_{\mathcal{F}_0} = 10\,\%$. For $\alpha_{\mathcal{F}_0} < 90°$, a slight increase in the acc-value is visible for flat fiber crossings with a crossing angle of $\theta = 90°$.

For the equally proportional fiber population $\Psi_{\mathcal{F}_0} = 50°$, the acc-value reaches values close to 0 for $\alpha_{\mathcal{F}_0} = (0, 30)°$ and a crossing angle of $\theta = 90°$. Inclined secondary fiber populations achieve a higher acc-value. The $\alpha_{\mathcal{F}_0} = 90°$ configuration reaches its maximum acc-value at a crossing angle of $\theta = 0°$ and lower stable values for all other configurations.

The first two fiber inclinations $\alpha_{\mathcal{F}_0} = (0, 30)°$ for $\Psi_{\mathcal{F}_0} = 70\,\%$ show a similar distribution of the acc-value. Small reductions are visible. For $\alpha_{\mathcal{F}_0} = 60°$, the acc-value is highest at low crossing angles θ, i. e., near the first fiber population. The acc-value decreases significantly at higher crossing angles. In the case of $\alpha_{\mathcal{F}_0} = 90°$ this is also visible, but the minimum for the acc-value is higher than in the previous case $\alpha_{\mathcal{F}_0} = 60°$.

The last case $\Psi_{\mathcal{F}_0} = 90\,\%$ shows high values for the acc-value parameter for the first three inclinations $\alpha_{\mathcal{F}_0} = (0,\ 30, 60)\,\%$. The last inclination $\alpha_{\mathcal{F}_0} = 90°$ shows a slight decrease of acc-value for higher crossing angles θ.

9.4 Discussion

Single fiber population Section 9.3.2 shows the results for the case of a single fiber population with inclined configurations. The transmittance values show an increase for the last free inclination values. This is to be expected since the 3D model configurations are aligned in parallel along an axis with some randomness. When this axis is aligned along the z-axis, i. e., $\alpha_{\mathcal{F}_0} = 90°$, the light rays, which also travel along the z-axis, statistically strike less tissue, so the transmittance must increase. Since the density of the tissue is quite high, the effect is only a few percent. In reality, however, one must take into account that additional effects such as light scattering also have a significant

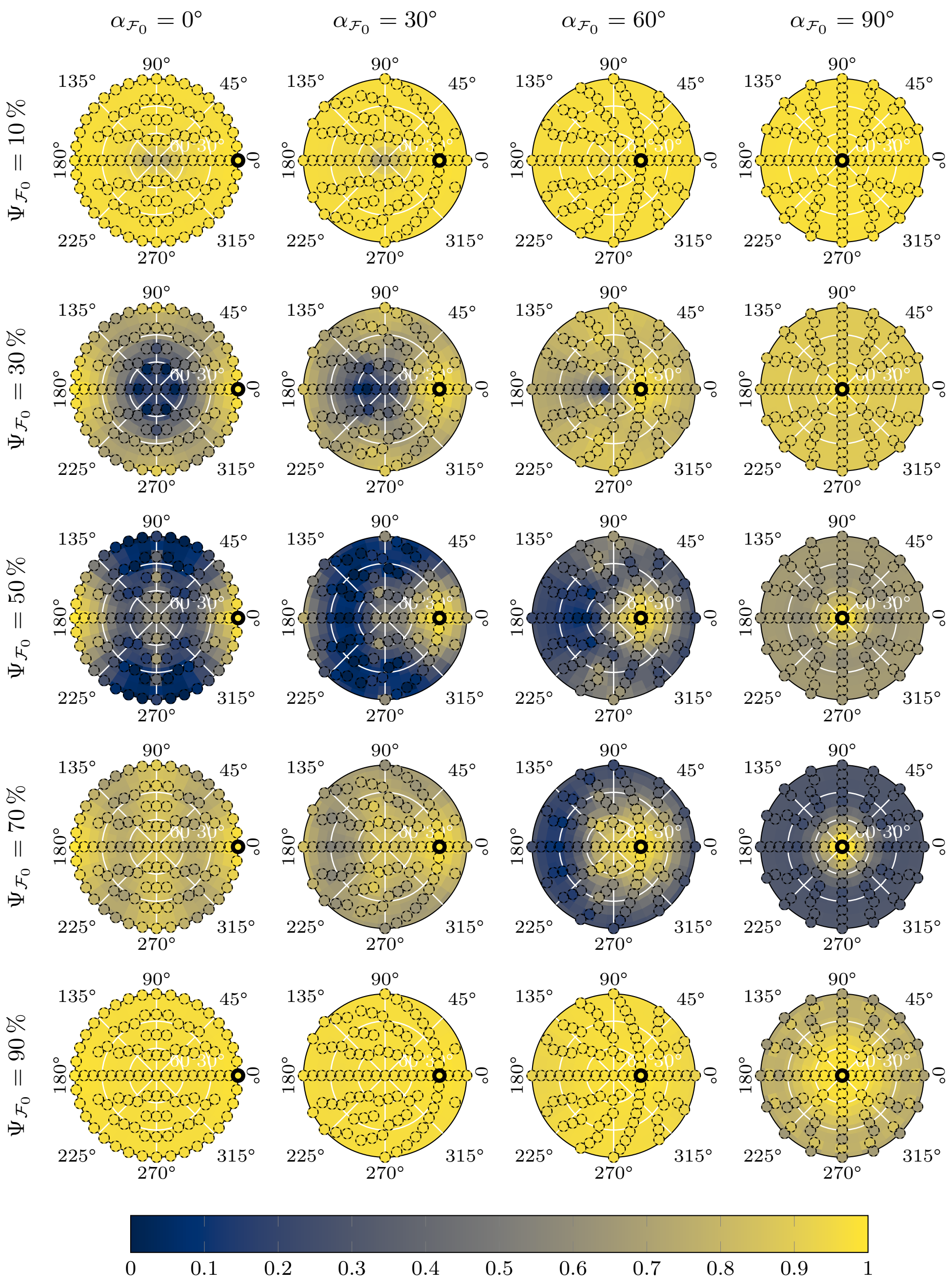


Fig. 9.23.: Mean acc-value between model and tilt analysis orientations.

effect on the transmittance [Men+21]. The variance of the transmittance corresponds to the variance of the noise model.

The retardation behaves similarly to the theoretical curve for a single retardation signal (see eqs. 4.1). However, the analyzed retardation is slightly higher. The theoretical signal is calculated by normalizing $\cos(\alpha_{\mathcal{F}_0})^2$ with the analyzed retardation for the flat case, i. e., $\alpha_{\mathcal{F}_0} = 0°$. This could explain the reduced values of the theoretical line compared to the simulation results for the mid-range inclinations.

The direction values have a very small variance, which increases strongly for high inclination values. Since the direction value is coupled to the inclination angle, this is to be expected: for an 90° inclined orientation, there is no direction or retardation. Therefore, the noise affects the signal, and generates a random phase of the sine. The variance of the tilting analysis is expected to be smaller than the actual distribution of the individual nerve fiber segments because the signal results from the overall interaction from the light with all segments on its path.

The same behavior can be observed for the inclination curve. However, here the variance for the last angle is lower. It is possible that this is an effect that for exactly $\alpha_{\mathcal{F}_0} = 90°$ the change of the tilted light beam statistically ends in four opposite phases in the sinusoidal signal. This might be more detectable for the tilt analysis than a slightly less inclined fiber configuration where only the amplitude changes. It is interesting to note that the variance is quite small and constant up to a range of about (70 to 75)°. This demonstrates that the slope analysis for a single fiber population is relatively accurate with respect to the dimension used here.

The relative thickness t_{rel} should theoretically be constant, since the volume fraction remains the same for different inclination angles. Due to the reduction in retardation, a higher variance is expected with increasing inclination angle. However, at an inclination angle of about $\alpha_{\mathcal{F}_0} = 65°$, the mean value begins to decrease. For $\alpha_{\mathcal{F}_0} = (85, 90)°$, the variance then increases significantly and for $\alpha_{\mathcal{F}_0} = 90°$, the mean is > 1. This indicates that the tilt analysis is unable to find a local minimum for t_{rel} values < 1. This may be one of the following three reasons. First, there is no local minimum with $t_{rel} < 1$. Second, there is a local minimum, but the optimizer is not able to find it with the initial values. Finally, there is a local minimum, but the solution found with a t_{rel} value > 1 is a better choice, i. e., has a lower R^2 value. Which of these is true cannot be said from this data, and remains an interesting investigation for the future. However, up to this point, the t_{rel} value can be used to indicate that these values cannot be trusted, i. e., the resulting orientation is not trustworthy.

The R-value reflects the noise on the signal for all inclination angles and stays constant.

The opening angle shows the combined information from the direction results and the inclination results. This value can possibly serve as additional uncertainty for further calculations like in a tractography. The median value is expected to be lower than the actual fiber models opening angle for the same reason as for a lower $\sigma_{25\,\%}^{75\,\%}$ variance in the direction and inclination angle.

The acc-value for fiber inclinations $\alpha_{\mathcal{F}_0} \leq 75°$ is very high. For an inclination angle of about $\alpha_{\mathcal{F}_0} = 80°$, a minimum is visible because for higher inclination angles the tilting analysis accuracy increases. Therefore, the ODF similarity has to also increase again.

The information about orientation, i. e., direction and inclination, is visible in the histogram (see fig. 9.5). However, the statistical limits are not quite as easy to see, e. g. median or quantiles. Nevertheless, it is a good and fast way to give visual feedback to the user, especially because the information about direction and inclination is not decoupled.

In summary, for the case of a single fiber population, the resulting tilt analysis shows good agreement with the individual orientations of the model, except for very steeply inclined configurations ($\gtrsim 70°$).

Flat crossing fiber population Section 9.3.3 describes the results of the flat crossing configurations. The transmittance value changes significantly with increasing θ, which is to be expected since non overlapping crossing models require more space. Therefore, fewer fibers can absorb light and the transmittance value must increase. This effect is also responsible for the transmittance to reach a maximum at a fiber population fraction of $\Psi_{\mathcal{F}_0} = 50\,\%$, since it is more complicated to find a collision-free state.

The retardation decreases almost linearly with increasing crossing angle. Theoretically, with a crossing angle of $\theta = 90°$ and a fiber population fraction of $\Psi_{\mathcal{F}_0} = 30\,\%$, only the remaining $40\,\%$ fibers affect retardation. However, the measured retardation of about 0.36 is larger than the expected $0.8 \cdot 0.4 = 0.32$. An explanation of this could be the noise on the data. It effects the distribution of the retardation for low amplitudes towards higher values. The reason for this is, that the retardation and the individual light intensities cannot be lower than 0.

The direction for the flat fiber crossings follows the expected theoretical curve of the circular mean. For the case $\Psi_{\mathcal{F}_0} = 50\,\%$ and $\theta = 90°$, the direction values are uniformly distributed since there is no unique direction in the sinusoidal. This is the effect of retardation cancellation and signal noise. For the here investigated distribution of fiber radii at microscopes resolution and pixel size individual fiber population for an interwoven dense fiber bundle cannot be distinguished. However, the tilt analysis used here uses only a single fiber orientation, or more generally, a single optic axis. In the future, more comprehensive models may be able to reveal the difference.

The variance of the inclination angle increases with increasing θ, which is to be expected as the retardation decreases, making the model more uncertain. In the case of a uniformly distributed fiber crossing, the inclination can reach any value as in the case for the direction value.

The same effect as for retardation and inclination is also observed for the decreasing value of t_{rel}. Since there is only one optic axis in the tilt analysis model, the t_{rel} value must follow the retardation for flat crossing fiber models. In the case of $\Psi_{\mathcal{F}_0} = 90°$, the tilt analysis also reaches values of $t_{rel} > 1$. As mentioned for the single fiber population, this is an effect of the solution algorithm of the tilt analysis. It can be used as an indication that the resulting values are not to be trusted and the current model does not fit the data.

The R-value is constant overall for all crossing angles. This agrees with theory as long as the tilt analysis is able to find the minimum of the cost function, the R-value^2, because then the remaining difference comes only from the noise.

The measured opening angle Ω, does not agree with the underlying distributions of the orientation of the fiber segments. However, this is to be expected since in a single voxel, i. e., has a volume of $1.3\,\mu m \times 1.3\,\mu m \times 60\,\mu m$, there are many fiber segments with which the light, i. e., all light rays, incident on the same CCD pixel can interact. This means that an average signal is measured and analyzed. Average value for an aperture angle means, of course, a decrease of its value. The fact that the measured aperture angles increase with increasing crossing angle is also to be expected. The main effect of an increasing crossing angle is that the amplitude of the sinusoidal signal, i.e. the retardation, decreases. Therefore, the estimation from the tilt analysis is more difficult and the result error increases.

The acc-value value has a local minimum that depends on the crossing angle θ. The minimum is higher for $\Psi_{\mathcal{F}_0}$ near 0 or 1. The curve is explained by the fact that with the currently used tilt analysis only the main fiber orientation, more precisely the circular mean, can be determined from the measured signal. Therefore, the acc-value must shrink. However, beyond a certain point, the delay is reduced to such an extent that the uncertainty increases to such an extent that the measured orientations become more and more random. Therefore, the ODF of the measured signal becomes more and more spherical, which after a certain point, i. e., the minima, is closer to the ODF of the model than an ODF that predicts only one main orientation, i. e., a cigar shape. This also explains why the position of the minimum is closer to 90° for fiber population proportions closer to 50 %, because the intersection of the ODF of fiber segments is more pronounced.

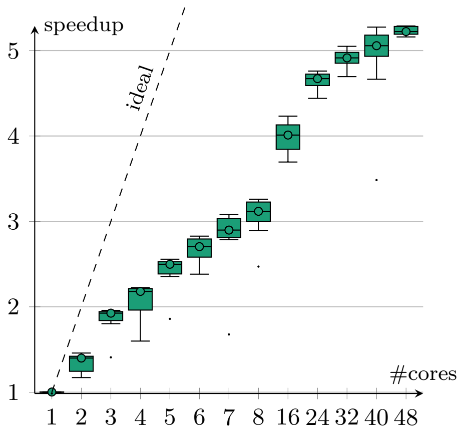
Inclined crossing The eight investigated parameters can be explained by the same arguments as for the previous flat fiber crossing, except for the inclination parameter. Here, the inclination follows the circle mean only for smaller crossing angles and reaches a maximum value as a function of the fiber population fraction $\Psi_{\mathcal{F}_0}$ until it decreases to 0° for $\theta = 90°$. This behavior can be explained by the fact that as the inclination increases, the light rays are no longer affected by the birefringence of the optic axis of the fibers. Therefore, the only remaining effect comes from the flat fiber population.

Free crossing fiber population The free crossing fiber population results can be interpreted as a summary of the previous results.

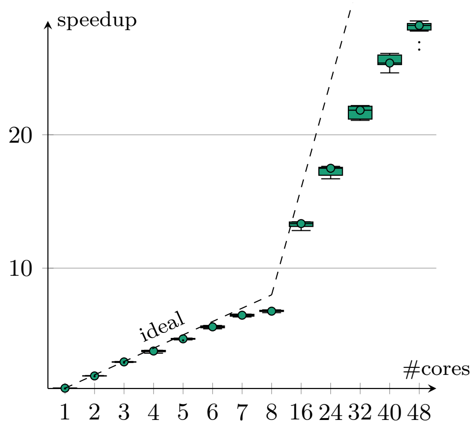
The transmittance shows the expected behavior, that the lowest values are reached for non-crossing fiber population. Here, the fiber density is the highest and therefore as well the absorption of the light intensity. For orientations with increasing crossing angles, transmittance increases as the fibers require more space to be collision free. This effect is further increased for fiber population fractions closer to 50 %.

The highest retardation values are reached for flat and non-crossing fiber populations. Here no extinction effect of the 3D-PLI signal is present. The retardation value shows a map of which orientations from two fiber population fractions the tilting analysis will lead to a result with high certainty. This, however, does not mean that the resulting orientation is the orientation of the underlying fibers, as it can be seen in the results of the direction and inclination. As in the previous results of the single flat crossing and oblique crossing fiber population(s), the analyzed angle corresponds to the circular value for flat fibers and the inclination is strongly influenced by the inclination angle.

From the results of t_{rel}, the statement that t_{rel} values greater than > 1 indicates an untrustworthy result is still true. However, the number of fiber configurations where this is the case is relatively small. In combination with retardation, though, both low t_{rel} values and low retardation values indicate considerably more untrustworthy results. This results in the distribution of the acc-value. The reduced acc-value are the fiber configurations one has to focus on improving the tilt analysis in the future. The acc-value can potentially be a good tool to use as a cost function in machine learning techniques, for example.



(a) OpenMP speedup discrete tissue generation.



(b) *MPI* speedup discrete tissue generation.

Fig. 9.24.: Discrete tissue generation speedup.

9.5 Speedup

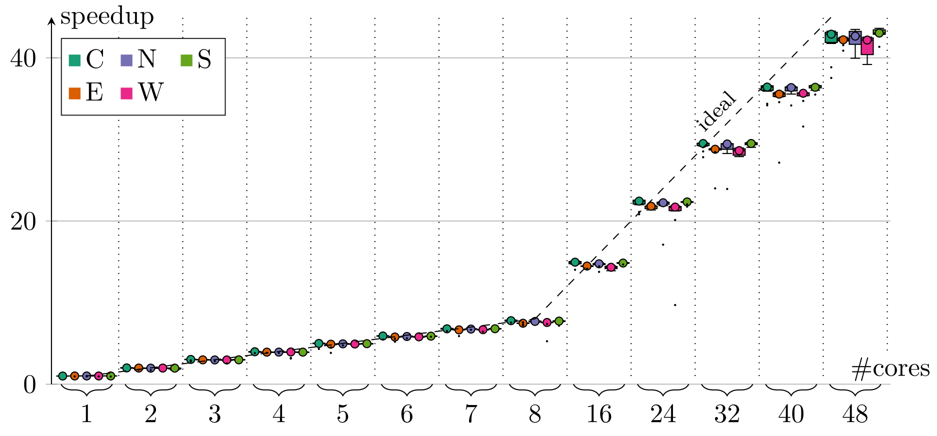
This section shows the results of speedup measurements for discretized tissue volume generation (see section 6.1) and light matter interaction simulation (see section 6.2) for the implemented parallelization with OpenMP and *Message Passing Interface* (*MPI*). Simulations were performed using a single compute node with CPU architecture `2x Intel(R) Xeon(R) CPU E5-4657L v2`. To measure the speedup, each algorithm is executed $N = 10$ times. To calculate the speedup value, the average measured time for $n_{cpu} = 1$ was then divided by the measured time for each value n_{cpu}. The volume used is the $\Psi_{\mathcal{F}_0} = 0\,\%/\alpha_{\mathcal{F}_0} = 0°$ from the section 9.3.1 parameterization.

9.5.1 Results

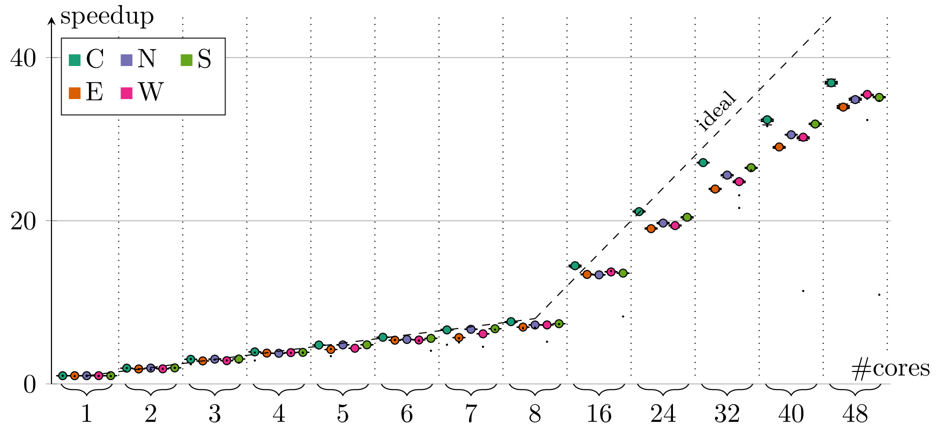
Figure 9.24a shows the speedup results for the parallelization of OpenMP of the discrete tissue generation algorithm. The speedup increases linearly from 1 core to 8 cores up to a speedup of about 3. For $(16,\ 24,\ 32,\ 40, 48)$ cores the speedup is increased further, but is still in the range of $(4$ to $5)$.

Figure 9.24b shows the speedup for the *MPI* implementation. Here, the speedup up to 8 cores is again linear and reaches about 6.8. The speedup from 16 cores to 48 cores is also linear, but the inclination decreases. For 48 cores, a speedup of about 28 is obtained. The variance is significantly lower compared to the OpenMP implementation.

The results of the speedup measurements for the light matter interaction are then presented. All tilt directions (flat, east, north, west, and south) are simulated. Figure 9.25a



(a) OpenMP speedup simulation for 5 tilt direction.



(b) *MPI* speedup simulation including all five tilt direction.

Fig. 9.25.: 3D-PLI simulation speedup measurements.

shows the speedup for the OpenMP implementation. The results show ideal speedup up to 8 cores. As the CPU count is further increased up to 48 cores, the speedup is slightly lower than the ideal line with a speedup of 42 for 48 cores. No significant difference in acceleration for different tilt directions is apparent.

Figure 9.25b shows the speedup for the *MPI* implementation. Again, the speedup is ideal up to 8 cores. For CPU numbers in the range of (16 to 48) cores, the speedup is again lower than the ideal line with a speedup of about 35 for 48 cores. Compared to the previous OpenMP implementation, the speedup is further reduced. A significant difference in the measurement is visible for different tilt directions. The flat measurement has the highest speedup.

9.5.2 Discussion

The speedup for the discrete tissue generator in the case of OpenMP is appropriate up to 4 cores with a speedup of slightly above 2. Above 4 cores, especially $\geq$ 16 cores is not recommended to use. This behavior is most likely due to the fact that the CPUs read from the same memory address. However, since the different writing instructions do not use the same memory address and there are no race conditions, a higher speedup was expected.

The speedup for the *MPI* implementation, on the other hand, is almost optimal up to 8 cores. Above that, the speedup is reduced compared to the ideal case, but still quite good. Since the CPUs (cores) run independently in this case, no communication is required. However, on a single-node system, as in this case, the allocation of memory is a race condition and slows down the algorithm. This is not expected on a multi-node system. An additional reduction in computation time results from the fact that all cores must traverse all fiber coordinates. Pre-filtering could shorten this process. For example, a global axis aligned bounding box (AABB) can be computed for each fiber.

The speedups for the 3D-PLI simulation are quite similar and very high for both implementations. The OpenMP implementation is slightly better, which is to be expected since the parallelization is done along a `std::vector` of Stokes vectors. The *MPI* implementation on the other hand still needs to communicate with each other in case of transferring the Stokes vectors. Also, the discretized volumes are already present in memory, so memory allocation does not slow down the runtime compared to the volume discretizer. For the *MPI* implementation, the communication effect is also visible for flat and tilted simulations. In the case of a flat simulation, no communication and thus no barriers are required, so the speedup is higher. A difference in North-South and East-West tilt could be explained for different number of cores by the fact that the volume has to be divided into subvolumes. This is done within the algorithm so that the surface area is minimized. The splitting process produces subvolumes with a tendency to have longitudinal quartiles along the x-axis. This may explain why the North-South tilt has a slightly higher speedup.

In summary, the *MPI* implementation should be used to speed up the simulation.

Part IV

Closing Remarks

10 What is Next?

Nerve fiber modeling software The collision-free nerve fiber modeling algorithm can generate densely packed fiber models, which are suitable for 3D-Polarized Light Imaging (3D-PLI) simulations. However, for larger volumes, the algorithm is limited in terms of computation time or, more precisely, the number of objects in the volume to be solved.

However, there are optimization options available to improve performance. First, the models created by the sandbox can be improved so that the initial fiber configurations have less overlap. This means that for a given volume, the number of objects and thus the runtime are reduced. Additionally, it was discovered that most collisions can be solved in the first $\approx 10\,\%$ of the runtime, and the remaining time is required for the remaining minimal overlaps (see section 8.3.1).

A second strategy is to design the algorithm in such a way that the number of calculation steps is reduced. For example, one could increase the motion that a fiber segment is allowed to perform. This strategy would probably lead to less densely packed models, since without an external force or attraction between the nerve fibers, they can only move apart. [1]

Another possibility is to speed up the runtime per step. This can be done, for example, by choosing a simpler calculation. In the *Microstructure Environment Designer with Unified Sphere Atoms* (*MEDUSA*) algorithm, this is already been realized by using spheres instead of fiber segments. The collision of spheres is only be calculated by the Euclidean distance with respect to the sum of the spheres radii. However, the use of spheres is associated with a disadvantage. The number of objects increases dramatically, as many spheres are needed to approximate the surface of a single conical fiber segment.

A current limitation of the algorithm is the parallelization on a multicore system. For shared memory parallelization, atomic operations must be introduced, which takes a lot of time. For an algorithm without shared memory, the required data must be exchanged between the individual CPUs, which takes also quite some time. Therefore, one needs to redesign the algorithm to limit the necessary locks or data exchange. Common solutions

[1] With an external or attracting force, one would introduce oscillations.

are to divide the volume into subvolumes (as in the case of 3D-PLI simulation), where each volume can be computed separately. The boundaries of such subvolumes contain then the objects from neighboring volumes (halo). Therefore, only the information of the halo needs to be exchanged.

At this point, the most promising optimization is to use the architecture of the graphics processing unit (GPU) as e. g. in the *MEDUSA* algorithm. An algorithm exists, which operates on axis aligned bounding box (AABB) of the objects and uses a z-ordered tree instead of an octree [Kar12]. This algorithm also has the advantage that not only the collision checking computation is executed in parallel on the GPU, but also the generation of the z-ordered tree. The entire algorithm runs on the GPU and therefore does not need to communicate with the central processing unit (CPU) or random access memory (RAM) as long as the complete information fits on the GPU memory. This will drastically increase the speed of the collision checking. The remaining steps of the modeling algorithm, such as the movement of the fiber segments, can also be easily performed in parallel on the GPU.

3D-PLI software As part of this work, two proof-of-concept projects were conducted to develop a parallel GPU implementation for 3D-PLI simulations. The first was a seminar project to implement a parallel discrete tissue volume computation algorithm for the 3D-PLI simulation on the GPU architecture [Kob20a]. It was shown that the speedup of the discrete volume computation was very high, but the large memory requirements of the discretized volume negated this speedup overall, as GPUs are relatively limited here. In addition, the need to transfer the data back to RAM was too much of an overhead. Therefore, the second project was to implement a ray tracing algorithm that computes a light matter collision and matrix computation without pre-computing the discretized volume [Kob20b; @Ale20]. A light particle only computes the Müller-Stokes calculus only if it collides with a fiber object. This means, that a collision algorithm needs to be used. As a first implementation of a collision detection algorithm on the GPU, an uniaxial aligned collision search algorithm [Kar12] was implemented. The results showed that the acceleration possibilities on the GPU are enormous. However, due to the rather simple collision checking algorithm, the runtime was slightly longer, but still in the same order of magnitude as the CPU version. As with fiber generation, implementing the z-order algorithm mentioned above [Kar12] would be a vast improvement and solve the runtime issue.

This second work also implemented the algorithm is such a way, that the light rays can have any direction in space. This allows for the future an implementation of directional change such as it would be necessary for light scattering simulations.

Nerve fiber modeling and 3D-PLI simulation As a first achievement, the number of models was limited to a reasonable number. The next study can increase the number of nerve fiber models and simulations should be increased. Since two models are independent, all models can be generated in parallel using the entire computer architecture. In addition, the essential parameters for the models and simulations should be further investigated so that the number of required models can be reduced. Modern machine learning algorithms are a suitable tool for this purpose.

Another important task is the study of larger nerve fiber radii. This would include the larger nerve fiber radii that are anatomically present in the brain. In addition, larger nerve fiber radii could also be an approximation for a nerve fiber bundle consisting of multiple nerve fibers in the 3D-PLI simulation. They should then be simulated with a macroscopic birefringence model [Men+15]. If this is possible, one could significantly reduce the number of objects for model generation, which means that one could have either very small runtimes or larger volumes.

A further study should investigate if nerve fiber models can be used for 3D-PLI simulation, which are not fully collision free. Since about 90 % of the runtime is used on the last small remaining colliding objects, this would reduce the runtime by an order of magnitude.

Another interesting study is to investigate more complex models, e. g. three fiber populations or boundary regions consisting of neighboring nerve fiber tracts. This can be very interesting since this could improve the understanding of how the signal changes when the light beam is tilted and therefore improve the tilt analysis further. For more complex models, machine learning can possibly be used to identify the underlying fiber structure within a voxel using the 3D-PLI signals. In diffusion Magnetic Resonance Imaging (dMRI), nerve fiber models and their simulations, as well as the use of deep learning, have already shown that the underlying fiber structure can be identified from the original signals [Gin19]. This can potentially be applied to 3D-PLI similarly.

By publishing the code as open access software, other researchers could already use the fiber modeling software to generate non colliding fiber models and use them as skeletal muscle fiber models [Ji+21].

11 Conclusion

In this work, two algorithms were presented within the software package *fiber architecture simulation toolbox for 3D-PLI* (*fastPLI*). The first algorithm is capable of designing non-colliding nerve fiber models in a 3D volume. These models can then be used in the second algorithm, which simulates the interaction of polarized light with the modeled nerve fibers in a virtual 3D-PLI microscopic setup.

Nerve fiber modeling software The algorithm for developing non-colliding nerve fiber models takes as input a list of 4D points for each nerve fiber, where the first three values are the x, y, z coordinate in space and the fourth value is the radius of the nerve fiber at that point. Therefore, a nerve fiber is represented as a cylindrical conical segments that can change its radius along the path. For each fiber segment containing two adjacent points, a test is performed to determine if there is a collision with another fiber segment. If this is the case, both fiber segments are moved slightly away from each other. This is done for all fiber segments in the volume as long as a collision is detected. Both the length of a fiber segment and the bending radius of a nerve fiber are controlled by the user-defined software parameters. These parameters change the discretization and stiffness of the fiber models.

To speed up model generation, an octree is used in the algorithm to divide the volume into subvolumes. This octree can be executed in parallel on multiple CPUs. A visualization is provided to display the volume, allowing the user to interact with the algorithm after each computation step.

An additional project called *MEDUSA* was developed in collaboration with Neurospin at Alternative Energies and Atomic Energy Commission (CEA), which allows designing non-colliding nerve fiber models with cells such as astrocytes or oligodendrocytes that connect to myelin-enveloped axons.

Simulation software The simulation software for 3D-PLI takes a configuration of nerve fibers as input and simulates the light matter interaction within a 3D-PLI setup. The simulation is divided into two consistent parts. The first part generates a discretized

3D volume. This volume is used by the second part, which calculates the resulting light intensity using the Müller-calculus calculation and the simulation of several light vectors through the volume. The final light intensity is stored in a 2D array, which can be modeled as a charge-coupled device (CCD) array with a specified resolution and noise model. The simulation is capable of simulating multiple tilted light beams, allowing multiple views of the same subvolume that the light is traversing. In addition, the analysis algorithms are implemented into the software package, allowing the nerve fiber orientation to be calculated using the tilt analysis *robust orientation fitting via least squares* (*ROFL*).

The simulation is capable of using multiple cores as well as a system with multiple nodes communicating via *Message Passing Interface* (*MPI*). This allows the simulation of a large volume of nerve fibers, which takes up a large amount of memory due to the discretized volume required.

Software package *fastPLI* All algorithms are written as modules in *Python3* within the software package *fastPLI*. The software package is published as an open source software package so that users can share, ask, and further develop the software to ensure high interchangeability. The software has been tested by several users and published in Journal of Open Source Software (JOSS).

Nerve fiber modeling results The nerve fiber modeling algorithm controls the necessary movement of the fibers to solve the collisions via the length of the fiber segments and the bending radius of the fibers. Both parameters were characterized in terms of the resulting orientations and computational speed. To reduce the dimensionality of the possible configurations, a set of parameters was designed to describe the model. This set consists of a relative angle between the two fiber populations, an inclination angle of the first fiber population, a rotation angle of the second fiber population around the first one and a population fraction parameter. The set allows nerve fiber models with up to two nerve fiber populations with arbitrary crossing angles to be examined in 3D without describing each fiber separately.

Based on these parameters, the properties of the modeling parameters were first characterized. A set of values suitable to generate non-colliding nerve fiber models without introducing significant distortion to the initial configuration was identified. In addition, these parameter values are suitable to reduce the runtime to a reasonable extent without losing configuration characteristics or distorting the resulting models. To reduce the runtime even further, it was found that about an order of magnitude of computation time can be reduced by not solving the models completely. However, the impact of a non-collision-free model remains to be investigated in an additional study.

With the model software parameters found, a library of up to two nerve fiber models described by the four model parameters was created for the 3D-PLI simulation. The orientation distribution was analyzed to be used as a comparison for the orientation analysis of the simulation.

3D-PLI simulation results Analogous to the nerve fiber modeling software, the parameters of the simulation software were first characterized and measured. The optical resolution of the microscope used was reproduced with previous results, as was the optical noise of the system. Tissue properties were derived from tissue samples measured in 3D-PLI so that the simulation could reproduce the results with its limitations. The most important characteristic of the simulation, the accuracy and runtime, was characterized by studying multiple `voxel_size`. It was possible to identify a lower bound that must be used for a given nerve fiber radius.

Using the identified software parameters and the prepared nerve fiber models, simulations were performed and analyzed with the tilt analysis. The models were divided into four groups: a single, a flat crossing, an oblique crossing, and a free crossing nerve fiber population. This allowed to focus on a specific behavior.

In the case of a single nerve fiber population, only the inclination of the models was a variable. With the simulation results, it was shown that in the case of a single fiber population, the orientation can be correctly identified, with an increased uncertainty for very steep fibers. Statistically, the mean can be measured correctly when measuring a homogenous volume with multiple image pixel. For very steep nerve fibers, the relative effective tissue thickness becomes unstable in the tilt analysis. This behavior can be used as an indication of how uncertain the results are.

The flat crossing nerve fiber population results in a single value for the tilt analysis that appears to follow the circular mean value of the individual orientations. With this behavior, the predominant nerve fiber population is mostly visible, with a slight systematic error due to the orientation to the second nerve fiber direction. With the nerve fiber radius distribution and image resolution chosen here, the underlying orientation of the individual nerve fiber population could not be resolved.

The population of inclined crossing nerve fibers has similar characteristics to the population of flat crossing nerve fibers. The inclination also appears to follow the circular mean value of each population, but due to the fact that inclined nerve fibers result in less change in the polarization of the light, the resulting orientation is biased toward the less inclined fiber population. At this resolution, the current tilt analysis cannot identify the individual nerve fiber populations.

The last models examined were unrestricted nerve fiber populations, so any configuration describable by the four model parameters was possible. As a comparison an orientation distribution function (ODF) metric was used to determine the orientations of the models for which the tilt analysis of the 3D-PLI signal was error-prone and which were highly reliable. In addition, the individual parameters were discussed and the underlying behavior leading to the erroneous orientations was determined.

An analysis of the simulation speed has shown that the simulation software can be parallelized very well with the capabilities of *MPI* and can use a multi-node system that allows large volumes to be simulated without loss of computation time. However, the tissue generation process is slightly less efficient. Nevertheless, since multiple 3D-PLI simulations with tilted light beams are run on the same discretized tissue, this is not a critical issue.

Many possible further optimizations of the algorithm could be identified. The general advice is to increase the computational power by using parallel computing architectures like a GPU. Existing algorithms suitable for this type of computation for both the nerve fiber modeling and 3D-PLI simulations could be identified and should be implemented in the future.

Overall, this thesis presents a software package able to generate complex non colliding nerve fiber models which can be simulated inside a virtual 3D-PLI microscope setup. The algorithms are highly parallelized and able to run on current supercomputing facilities. Additionally, the software package was designed with ease of use in mind. For the future, possible improvements were presented that further reduce the runtime. During the development phase of this software package, several publications with a release of the final software code in the JOSS were created. Simulation showed the behavior of the 3D-PLI signal for multiple fiber populations for the first time.

Bibliography

Literature

[Ali+20] Abib Alimi, Samuel Deslauriers-Gauthier, Felix Matuschke, et al. “Analytical and fast Fiber Orientation Distribution reconstruction in 3D-Polarized Light Imaging”. In: *Medical Image Analysis* 65 (Oct. 2020), p. 101760. URL: https://doi.org/10.1016/j.media.2020.101760 (cit. on p. 112).

[Amu+13] K. Amunts, C. Lepage, L. Borgeat, et al. “BigBrain: An Ultrahigh-Resolution 3D Human Brain Model”. In: *Science* 340.6139 (June 2013), pp. 1472–1475. URL: https://doi.org/10.1126/science.1235381 (cit. on pp. 5, 12).

[Amu+16] Katrin Amunts, Christoph Ebell, Jeff Muller, et al. “The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain”. In: *Neuron* 92.3 (Nov. 2016), pp. 574–581. URL: https://doi.org/10.1016/j.neuron.2016.10.046 (cit. on p. 5).

[Axe+11a] Markus Axer, Katrin Amunts, David Grässel, et al. “A novel approach to the human connectome: Ultra-high resolution mapping of fiber tracts in the brain”. In: *NeuroImage* 54.2 (Jan. 2011), pp. 1091–1101. URL: https://doi.org/10.1016/j.neuroimage.2010.08.075 (cit. on p. 6).

[Axe+11b] Markus Axer, David Grässel, Melanie Kleiner, et al. “High-Resolution Fiber Tract Reconstruction in the Human Brain by Means of Three-Dimensional Polarized Light Imaging”. In: *Frontiers in Neuroinformatics* 5 (2011). URL: https://doi.org/10.3389/fninf.2011.00034 (cit. on pp. 6, 25, 28).

[Axe+16] Markus Axer, Sven Strohmer, David Gräßel, et al. “Estimating Fiber Orientation Distribution Functions in 3D-Polarized Light Imaging”. In: *Frontiers in Neuroanatomy* 10 (Apr. 2016). URL: https://doi.org/10.3389/fnana.2016.00040 (cit. on p. 6).

[Bea+19] Justine Beaujoin, Alexandros Popov, Raïssa Yebga Hot, et al. “CHENONCEAU: towards a novel mesoscopic (100/200μm) post-mortem human brain MRI atlas at 11.7T”. In: *OHBM (Organization for Human Brain Mapping)*. Rome, Italy, 2019. URL: https://hal.archives-ouvertes.fr/hal-02876136 (cit. on p. 5).

[Ber+18] Shai Berman, Kathryn L. West, Mark D. Does, Jason D. Yeatman, and Aviv A. Mezer. “Evaluating g-ratio weighted changes in the corpus callosum as a function of age and sex”. In: *NeuroImage* 182 (Nov. 2018), pp. 304–313. URL: https://doi.org/10.1016/j.neuroimage.2017.06.076 (cit. on p. 14).

[Cal+19] Ross Callaghan, Daniel C. Alexander, Hui Zhang, and Marco Palombo. “Contextual Fibre Growth to Generate Realistic Axonal Packing for Diffusion MRI Simulation”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2019, pp. 429–440. URL: https://doi.org/10.1007/978-3-030-20351-1_33 (cit. on p. 6).

[Cer+17] Mara Cercignani, Giovanni Giulietti, Nick G. Dowell, et al. “Characterizing axonal myelination within the healthy population: a tract-by-tract mapping of effects of age and gender on the fiber g-ratio”. In: *Neurobiology of Aging* 49 (Jan. 2017), pp. 109–118. URL: https://doi.org/10.1016/j.neurobiolaging.2016.09.016 (cit. on p. 14).

[Cos+21] Irene Costantini, Enrico Baria, Michele Sorelli, et al. “Autofluorescence enhancement for label-free imaging of myelinated fibers in mammalian brains”. In: *Scientific Reports* 11.1 (Apr. 2021). URL: https://doi.org/10.1038/s41598-021-86092-7 (cit. on p. 6).

[Cos+20] Irene Costantini, Enrico Baria, Michele Sorelli, et al. “MAGIC: A label-free fluorescence method for 3D high-resolution reconstruction of myelinated fibers in large volumes”. In: (July 2020). URL: https://doi.org/10.1101/2020.07.28.225011 (cit. on p. 6).

[Dea+16] Douglas C. Dean, Jonathan O’Muircheartaigh, Holly Dirks, et al. “Mapping an index of the myelin g-ratio in infants using magnetic resonance imaging”. In: *NeuroImage* 132 (May 2016), pp. 225–237. URL: https://doi.org/10.1016/j.neuroimage.2016.02.040 (cit. on p. 14).

[Dem06] Wolfgang Demtröder. *Experimentalphysik*. Berlin u.a: Springer, 2006 (cit. on p. 15).

[Doh+15] Melanie Dohmen, Miriam Menzel, Hendrik Wiese, et al. “Understanding fiber mixture by simulation in 3D Polarized Light Imaging”. In: *NeuroImage* 111 (May 2015), pp. 464–475. URL: https://doi.org/10.1016/j.neuroimage.2015.02.020 (cit. on pp. 59, 77).

[Fli12] Torsten Fließbach. *Elektrodynamik*. Spektrum Akademischer Verlag, 2012. URL: https://doi.org/10.1007/978-3-8274-3036-6 (cit. on p. 15).

[Gin19] Kévin Ginsburger. *Modeling and simulation of the diffusion MRI signal from human brain white matter to decode its microstructure and produce an anatomic atlas at high fields (3T)*. Aug. 2019. URL: https://www.theses.fr/2019SACLS158 (cit. on pp. 6, 37, 54, 71, 88, 151).

[Gin+19] Kévin Ginsburger, Felix Matuschke, Fabrice Poupon, et al. “MEDUSA: A GPU-based tool to create realistic phantoms of the brain microstructure using tiny spheres”. In: *NeuroImage* 193 (June 2019), pp. 10–24. URL: https://doi.org/10.1016/j.neuroimage.2019.02.055 (cit. on pp. 37, 54, 56, 58, 71, 88).

[Gin+18] Kévin Ginsburger, Fabrice Poupon, Justine Beaujoin, et al. “Improving the Realism of White Matter Numerical Phantoms: A Step toward a Better Understanding of the Influence of Structural Disorders in Diffusion MRI”. In: *Frontiers in Physics* 6 (Feb. 2018). URL: https://doi.org/10.3389/fphy.2018.00012 (cit. on pp. 6, 88).

[Goo+16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on p. 5).

[Ji+21] Fengting Ji, Manik Bansal, Bingrui Wang, Yi Hua, and Ian A Sigal. "Mechanical properties of scleral collagen fibers obtained using a new fiber-based specimen-specific model of sclera microstructure." In: *Investigative Ophthalmology & Visual Science* 62 (8 June 2021). URL: https://iovs.arvojournals.org/article.aspx?articleid=2773519 (cit. on p. 151).

[Jon41] R. Clark Jones. "A New Calculus for the Treatment of Optical SystemsI Description and Discussion of the Calculus". In: *Journal of the Optical Society of America* 31.7 (July 1941), p. 488. URL: https://doi.org/10.1364/josa.31.000488 (cit. on p. 21).

[Jun+18] Woojin Jung, Jingu Lee, Hyeong-Geol Shin, et al. "Whole brain g-ratio mapping using myelin water imaging (MWI) and neurite orientation dispersion and density imaging (NODDI)". In: *NeuroImage* 182 (Nov. 2018), pp. 379–388. URL: https://doi.org/10.1016/j.neuroimage.2017.09.053 (cit. on p. 14).

[Kar12] Tero Karras. *Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees*. eng. 2012. URL: http://diglib.eg.org/handle/10.2312/EGGH.HPG12.033-037 (cit. on pp. 55, 56, 104, 150).

[Kob20a] Alexander Kobusch. "Parallele Implementierung einer Polarisationsmikroskopie-Simulation zur Licht-Hirngewebe-Wechselwirkung auf der GPU". Seminararbeit, FH Aachen, 2020. Seminararbeit. FH Aachen, 2020, p. 23 (cit. on p. 150).

[Kob20b] Alexander Kobusch. "Simulation von 3D-Polarized Light Imaging mit GPU basiertem Ray Tracing". Bachelorarbeit, FH Aachen, 2020. Bachelorarbeit. FH Aachen, 2020, p. 46. URL: https://juser.fz-juelich.de/record/887783 (cit. on p. 150).

[Lie+14] Daniel Liewald, Robert Miller, Nikos Logothetis, Hans-Joachim Wagner, and Almut Schüz. "Distribution of axon diameters in cortical white matter: an electron-microscopic study on three human brains and a macaque". In: *Biological Cybernetics* 108.5 (Aug. 2014), pp. 541–557. URL: https://doi.org/10.1007/s00422-014-0626-2 (cit. on p. 14).

[Luc16] Sonja Lucksch. *Optimierung und Parallelisierung der Software simPLI zur Simulation von 3D-Polarized-Light-Imaging Messungen*. 2016. URL: http://juser.fz-juelich.de/record/819319 (cit. on pp. 59, 77).

[Mai+17] Klaus H. Maier-Hein, Peter F. Neher, Jean-Christophe Houde, et al. "The challenge of mapping the human connectome based on diffusion tractography". In: *Nature Communications* 8.1 (Nov. 2017). URL: https://doi.org/10.1038/s41467-017-01285-x (cit. on p. 5).

[Mar06] Henry Markram. "The Blue Brain Project". In: *Nature Reviews Neuroscience* 7.2 (Feb. 2006), pp. 153–160. URL: https://doi.org/10.1038/nrn1848 (cit. on p. 5).

[Mat+21] Felix Matuschke, Katrin Amunts, and Markus Axer. "fastPLI: A Fiber Architecture Simulation Toolbox for 3D-PLI". In: *Journal of Open Source Software* 6.61 (May 2021), p. 3042. URL: `https://doi.org/10.21105/joss.03042` (cit. on pp. 7, 38, 44, 71, 79).

[Mat+19] Felix Matuschke, Kévin Ginsburger, Cyril Poupon, Katrin Amunts, and Markus Axer. "Dense Fiber Modeling for 3D-Polarized Light Imaging Simulations". In: *Advances in Parallel Computing* 34.Future Trends of HPC in a Disruptive Scenario (2019), pp. 240–253. URL: `https://doi.org/10.3233/APC190017` (cit. on pp. 7, 38, 44).

[Men+15] M. Menzel, K. Michielsen, H. De Raedt, et al. "A Jones matrix formalism for simulating three-dimensional polarized light imaging of brain tissue". In: *Journal of The Royal Society Interface* 12.111 (Oct. 2015), p. 20150734. URL: `https://doi.org/10.1098/rsif.2015.0734` (cit. on pp. 59, 151).

[Men18] Miriam Menzel. *Finite-Differenzen-Simulationen im Zeitbereich zur verbesserten Rekonstruktion der Nervenfaserarchitektur des Gehirns durch 3D-Bildgebung mit polarisiertem Licht*. en. Vol. RWTH Aachen University. RWTH Aachen University, 2018, p. 2018. URL: `http://publications.rwth-aachen.de/record/750948` (cit. on pp. 27, 30, 37, 40, 59, 60).

[Men14] Miriam Menzel. *Simulation and Modeling for the Reconstruction of Nerve Fibers in the Brain by 3D Polarized Light Imaging*. RWTH Aachen, Masterarbeit, 2014. 2014, pp. v, 165. URL: `https://juser.fz-juelich.de/record/155964` (cit. on pp. 27, 59, 107).

[Men+16] Miriam Menzel, Markus Axer, Hans De Raedt, and Kristel Michielsen. "Finite-Difference Time-Domain Simulation for Three-Dimensional Polarized Light Imaging". In: *Brain-Inspired Computing*. Ed. by Katrin Amunts, Lucio Grandinetti, Thomas Lippert, and Nicolai Petkov. Cham: Springer International Publishing, 2016, pp. 73–85 (cit. on p. 59).

[Men+20] Miriam Menzel, Markus Axer, Hans De Raedt, et al. "Toward a High-Resolution Reconstruction of 3D Nerve Fiber Architectures and Crossings in the Brain Using Light Scattering Measurements and Finite-Difference Time-Domain Simulations". In: *Physical Review X* 10.2 (Apr. 2020). URL: `https://doi.org/10.1103/physrevx.10.021002` (cit. on pp. 6, 59).

[Men+21] Miriam Menzel, Jan André Reuter, David Gräßel, et al. "Scattered Light Imaging: Resolving the substructure of nerve fiber crossings in whole brain sections with micrometer resolution". In: *NeuroImage* 233 (June 2021), p. 117952. URL: `https://doi.org/10.1016/j.neuroimage.2021.117952` (cit. on pp. 59, 138).

[Moh+15] Siawoosh Mohammadi, Daniel Carey, Fred Dick, et al. "Whole-Brain In-vivo Measurements of the Axonal G-Ratio in a Group of 37 Healthy Volunteers". In: *Frontiers in Neuroscience* 9 (Nov. 2015). URL: `https://doi.org/10.3389/fnins.2015.00441` (cit. on p. 14).

[Mue43] Hans Mueller. "Memorandum on the polarization optics of the photoelastic shutter". In: *Report of the OSRD project OEMsr-576* 2 (1943) (cit. on p. 21).

[Mur13] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, 2013. URL: https://mitpress.mit.edu/books/machine-learning-1 (cit. on p. 5).

[Reu+19] Jan André Reuter, Felix Matuschke, Miriam Menzel, et al. “FAConstructor: an interactive tool for geometric modeling of nerve fiber architectures in the brain”. In: *International Journal of Computer Assisted Radiology and Surgery* 14.11 (Aug. 2019), pp. 1881–1889. URL: https://doi.org/10.1007/s11548-019-02053-6 (cit. on pp. 7, 53).

[Sch+18a] Kurt G. Schilling, Vaibhav Janve, Yurui Gao, et al. “Histological validation of diffusion MRI fiber orientation distributions and dispersion”. In: *NeuroImage* 165 (Jan. 2018), pp. 200–221. URL: https://doi.org/10.1016/j.neuroimage.2017.10.046 (cit. on pp. 112, 136).

[Sch+21] Kurt G. Schilling, François Rheault, Laurent Petit, et al. “Tractography dissection variability: What happens when 42 groups dissect 14 white matter bundles on the same dataset?” In: *NeuroImage* 243 (Nov. 2021), p. 118502. URL: https://doi.org/10.1016/j.neuroimage.2021.118502 (cit. on p. 5).

[Sch+18b] Daniel Schmitz, Sascha E. A. Muenzing, Martin Schober, et al. “Derivation of Fiber Orientations From Oblique Views Through Human Brain Sections in 3D-Polarized Light Imaging”. In: *Frontiers in Neuroanatomy* 12 (Sept. 2018). URL: https://doi.org/10.3389/fnana.2018.00075 (cit. on pp. 29, 74).

[She+12] Elaine H. Shen, Caroline C. Overly, and Allan R. Jones. “The Allen Human Brain Atlas”. In: *Trends in Neurosciences* 35.12 (Dec. 2012), pp. 711–714. URL: https://doi.org/10.1016/j.tins.2012.09.005 (cit. on p. 5).

[Sti+15] Nikola Stikov, Jennifer S.W. Campbell, Thomas Stroh, et al. “In vivo histology of the myelin g-ratio with magnetic resonance imaging”. In: *NeuroImage* 118 (Sept. 2015), pp. 397–405. URL: https://doi.org/10.1016/j.neuroimage.2015.05.023 (cit. on p. 14).

[Sto52] G.G. Stokes. *On the Composition and Resolution of Streams of Polarized Light from Different Sources*. Proceedings of the Cambridge Philosophical Society : Mathematical and physical sciences. Printed at the Pitt Press by John W. Parker, 1852. URL: https://books.google.de/books?id=41VFGwAACAAJ (cit. on p. 21).

[Tre19] Philippe Trempe. “Spherical k-Nearest Neighbors Interpolation”. In: *CoRR* abs/1910.00704 (2019). arXiv: 1910.00704. URL: http://arxiv.org/abs/1910.00704 (cit. on p. 125).

[Wal+14] K.B. Walhovd, H. Johansen-Berg, and R.T. Káradóttir. “Unraveling the secrets of white matter – Bridging the gap between cellular, animal and human imaging studies”. In: *Neuroscience* 276 (2014). Secrets of the CNS White Matter, pp. 2–13. URL: https://www.sciencedirect.com/science/article/pii/S0306452214005430 (cit. on p. 14).

[Wie16] Hendrik Wiese. “Enhancing the signal interpretation and microscopical hardware concept of 3D Polarized Light Imaging”. Dissertation, Bergische Universität Wuppertal, 2016. Dissertation. Bergische Universität Wuppertal, 2016, 145 p. URL: https://juser.fz-juelich.de/record/887678 (cit. on pp. 26, 28, 29, 31, 64, 107).

[Yen+21] Anastasia Yendiki, Manisha Aggarwal, Markus Axer, et al. “Post mortem mapping of connectional anatomy for the validation of diffusion MRI”. In: (Apr. 2021). URL: https://doi.org/10.1101/2021.04.16.440223 (cit. on p. 5).

Software

[@Ale20] Felix Matuschke Alexander Kobusch. *fastPLI – Fiber Architecture Simulation Toolbox for PLI with GPU acceleration*. Version 1.0. https://jugit.fz-juelich.de/a.kobusch/fastpli. 2019-2020 (cit. on p. 150).

[@Col13] Andrew Collette. *Python and HDF5*. O’Reilly, 2013 (cit. on p. 72).

[@Dag+98] Leonardo Dagum and Ramesh Menon. “OpenMP: an industry standard API for shared-memory programming”. In: *Computational Science & Engineering, IEEE* 5.1 (1998), pp. 46–55 (cit. on p. 73).

[@Dal+11] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. “Parallel distributed computing using Python”. In: *Advances in Water Resources* 34.9 (Sept. 2011), pp. 1124–1139. URL: https://doi.org/10.1016/j.advwatres.2011.04.013 (cit. on p. 72).

[@Dal+05] Lisandro Dalcín, Rodrigo Paz, and Mario Storti. “MPI for Python”. In: *Journal of Parallel and Distributed Computing* 65.9 (Sept. 2005), pp. 1108–1115. URL: https://doi.org/10.1016/j.jpdc.2005.03.010 (cit. on p. 72).

[@Dal+08] Lisandro Dalcín, Rodrigo Paz, Mario Storti, and Jorge D’Elía. “MPI for Python: Performance improvements and MPI-2 extensions”. In: *Journal of Parallel and Distributed Computing* 68.5 (May 2008), pp. 655–662. URL: https://doi.org/10.1016/j.jpdc.2007.09.005 (cit. on p. 72).

[@For15] Message Passing Interface Forum. *MPI: A Message-passing Interface Standard, Version 3.1 ; June 4, 2015*. High-Performance Computing Center Stuttgart, University of Stuttgart, 2015. URL: https://books.google.de/books?id=uv1ajwEACAAJ (cit. on p. 73).

[@Fou20] Standard C++ Foundation. *Standard C++ on the web — news, status and discussion about the C++ standard on all compilers and platforms.* 2020. URL: https://isocpp.org/ (cit. on p. 53).

[@Jak+17] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. *pybind11 – Seamless operability between C++11 and Python*. https://github.com/pybind/pybind11. 2017 (cit. on p. 73).

[@Lam+15] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. “Numba”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM ’15*. ACM Press, 2015. URL: https://doi.org/10.1145/2833157.2833162 (cit. on p. 72).

[@Mat21] Felix Matuschke. *fastPLI – Fiber Architecture Simulation Toolbox for PLI*. Version 1.1. https://github.com/3d-pli/fastpli. 2016-2021 (cit. on p. 71).

[@The97] The HDF Group. *Hierarchical Data Format, version 5*. http://www.hdfgroup.org/HDF5/. 1997 (cit. on pp. 72, 75).

[@Vir+19] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. “SciPy 1.0 – Fundamental Algorithms for Scientific Computing in Python”. In: *arXiv e-prints* (July 2019). arXiv: 1907.10121 [cs.MS] (cit. on p. 72).

[@Wik18] OpenGL Wiki. *Main Page — OpenGL Wiki*. [Online; accessed 8-January-2020]. 2018. URL: http://www.khronos.org/opengl/wiki_opengl/index.php?title=Main_Page&oldid=14430 (cit. on pp. 53, 73).

Abbreviations

3D-PLI 3D-Polarized Light Imaging5 f., 15, 25, 28, 30, 34, 38, 41, 59 f., 62, 64, 66, 68, 71, 74, 79, 84, 87, 99, 101, 105 f., 108, 110 ff., 114, 116 ff., 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140 ff., 144, 149 ff., 153, 155 f.

AABB axis aligned bounding box 45, 47, 63, 144, 150

ACC angular correlation coefficient 134

API application programming interface 71, 79

CC conical capsule 39, 45, 53

CCD charge-coupled device . . . . 25 f., 30, 65 ff., 107, 140, 153

CEA Alternative Energies and Atomic Energy Commission 37, 54, 153

CPU central processing unit . . . . 48, 52, 59, 63, 67 ff., 77, 91, 141–144, 150, 153

dMRI diffusion Magnetic Resonance Imaging . . 5, 11, 13, 37, 53, 71, 87, 101, 151

FAConstructor Fiber Architecture Constructor 53

fastPLI fiber architecture simulation toolbox for 3D-PLI 6, 35, 38, 44, 53, 71–74, 76, 78, 80, 105, 110 f., 153 f.

FOM fiber orientation map 74

GM gray matter . . . . 11 f., 29

GPU graphics processing unit 55, 104, 150, 156

HDF5 Hierarchical Data Format v5 . . . . 74 f., 77

INM-1 Institute of Neuroscience and Medicine for structural and functional organisation of the brain 11, 25

JOSS Journal of Open Source Software 71, 79, 154, 156

LAP large-area polarimeter 26, 64

LMP large metripol 26, 107

LMP3D large metripol 3D 26, 29, 59, 64

MEDUSA Microstructure Environment Designer with Unified Sphere Atoms54, 149 f., 153

MOS metal oxide semiconductor 26

MPI Message Passing Interface . . . . 59, 63, 67 ff., 141–144, 154, 156

ODF orientation distribution function . . . . 112, 134, 139 f., 156

OpenGL Open Graphics Library 52

OpenMP Open Multi-Processing . . . . 52, 63, 68 f., 103, 141–144

RAM random access memory 69, 77, 150

ROFL robust orientation fitting via least squares 29, 68, 74, 110, 153

ROI region of interest 105

VOI volume of interest 60, 63

WM white matter . . . . 12, 29, 37 ff., 53 f., 71, 88

Nomenclature

μ	absorption coefficient
acc-value	angular correlation coefficient value
Δn	birefringence strength
$\mathcal{F}_0$	first fiber population
$\mathcal{F}_1$	second fiber population
$(\times)$	crossing fiber populations
$(\|\|)$	parallel fiber populations
λ	wavelength of the light
I	light intensity
θ	crossing angle between the two nerve fiber populations $\measuredangle \mathcal{F}_0, \mathcal{F}_1$
$\alpha_{\mathcal{F}_0}$	inclination of first nerve fiber population
$\Psi_{\mathcal{F}_0}$	model proportion fraction between $\mathcal{F}_0$ and $\mathcal{F}_0 + \mathcal{F}_1$
$R_{\mathcal{F}_1}$	rotation of second nerve fiber population around the first nerve fiber population
Ω	opening angle of angular distributions
g	gain factor of CCD-sensor
σ_{optic}	mathematical convolution parameter for applying camera resolution to the simulated image
p_s	size of a pixel in µm
r_f	Nerve fiber radius
μ_{r_f}	μ-value inside gaussian distribution to generate random nerve fiber radii
$\bar{r}_f$	mean nerve fiber radius along the fiber

σ_{r_f}	σ-value inside gaussian distribution to generate random nerve fiber radii
φ	direction
α	inclination
R-value	R-value of the ROFL fit result
seg_L	mean segment length for nerve fibers in modeling process
ν_l	relative segment length factor
seg_R	minimal segment bending radius for nerve fibers in modeling process
ν_r	relative minimal segment bending radius factor
s_s	length of light step in vx
t_{rel}	effecive birefringence thickness
v_s	size of voxel in µm

Part V

Appendices

A

Modeling

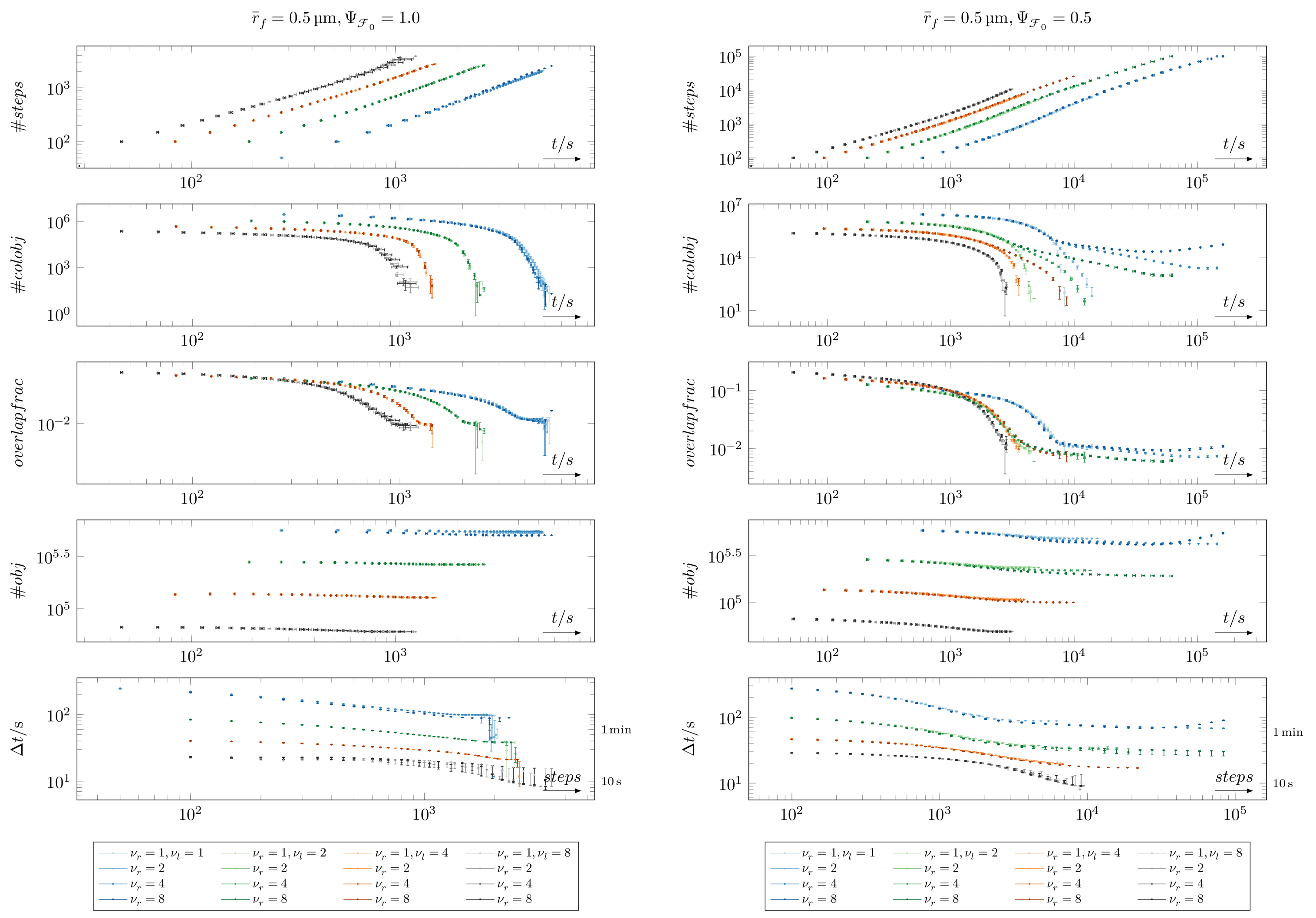


Fig. A.1.: Time evolution of the model building process of parallel and crossing fiber populations with $\bar{r}_f = 0.5\,\mu\text{m}$. Error bars indicate 25 % and 75 % quantiles (see section 8.3.1).

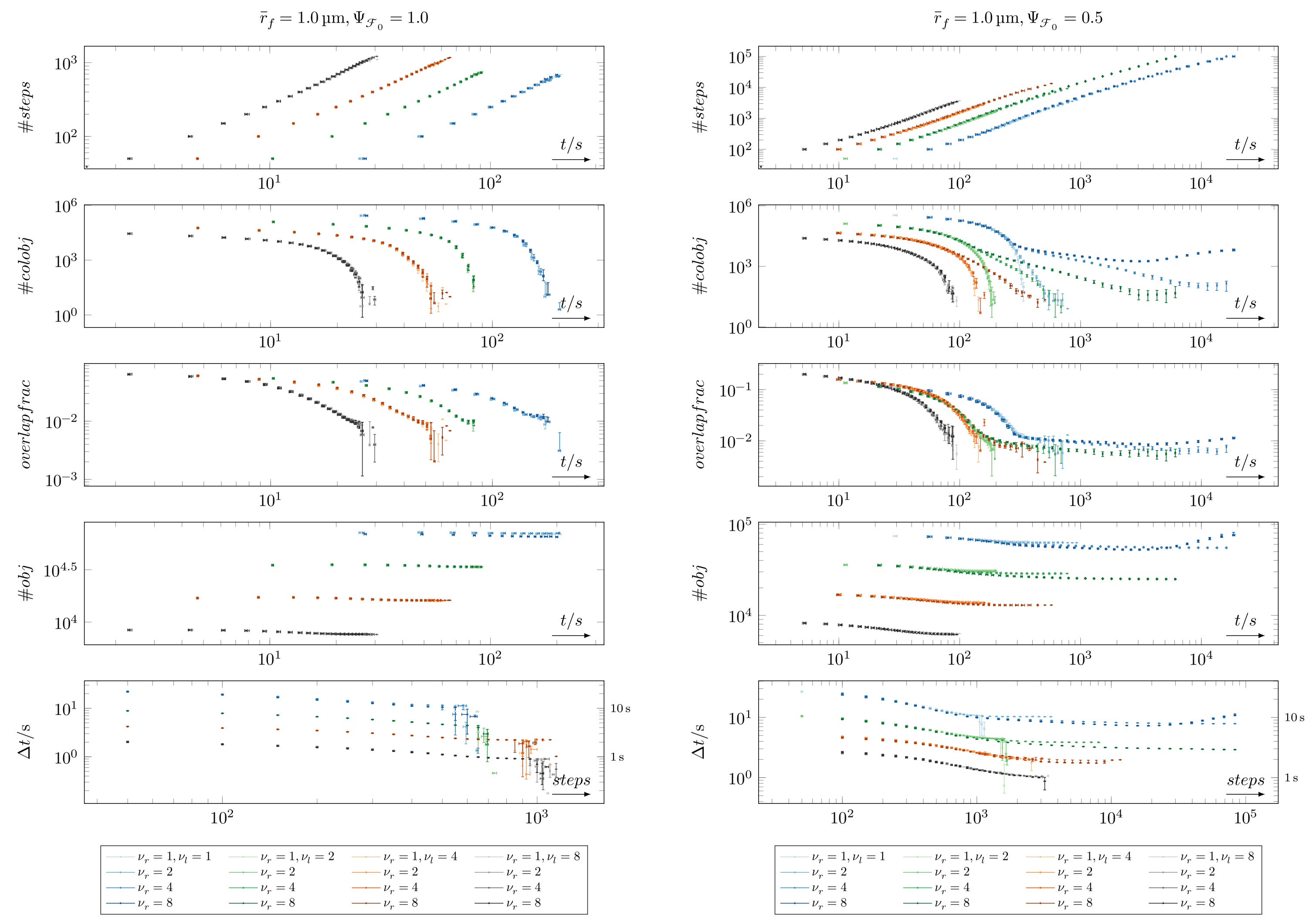


Fig. A.2.: Time evolution of the model building process of parallel and crossing fiber populations with $\bar{r}_f = 1.0\,\mu\text{m}$. Error bars indicate 25 % and 75 % quantiles (see section 8.3.1).

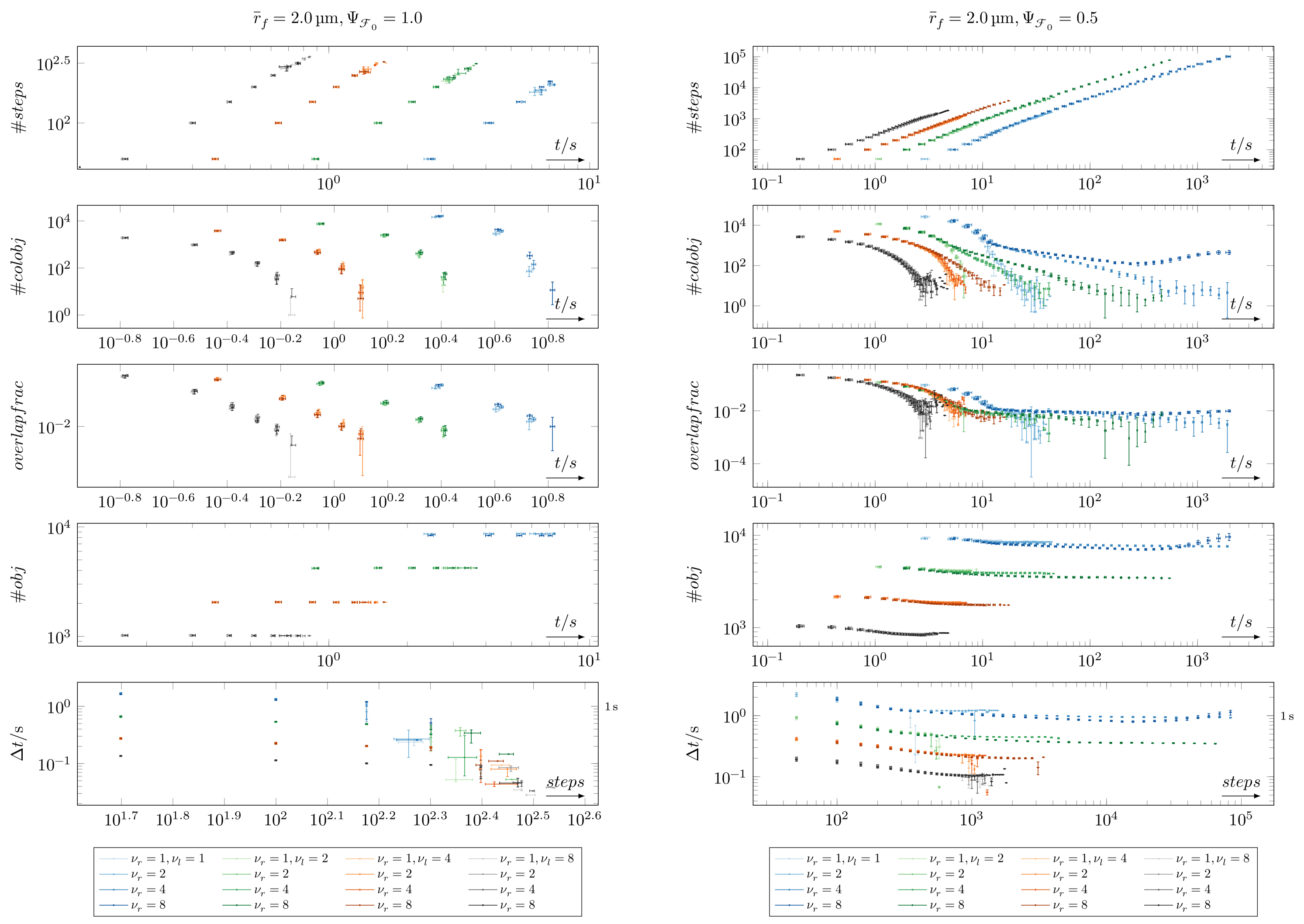


Fig. A.3.: Time evolution of the model building process of parallel and crossing fiber populations with $\bar{r}_f = 2.0\,\mu\mathrm{m}$. Error bars indicate 25 % and 75 % quantiles (see section 8.3.1).

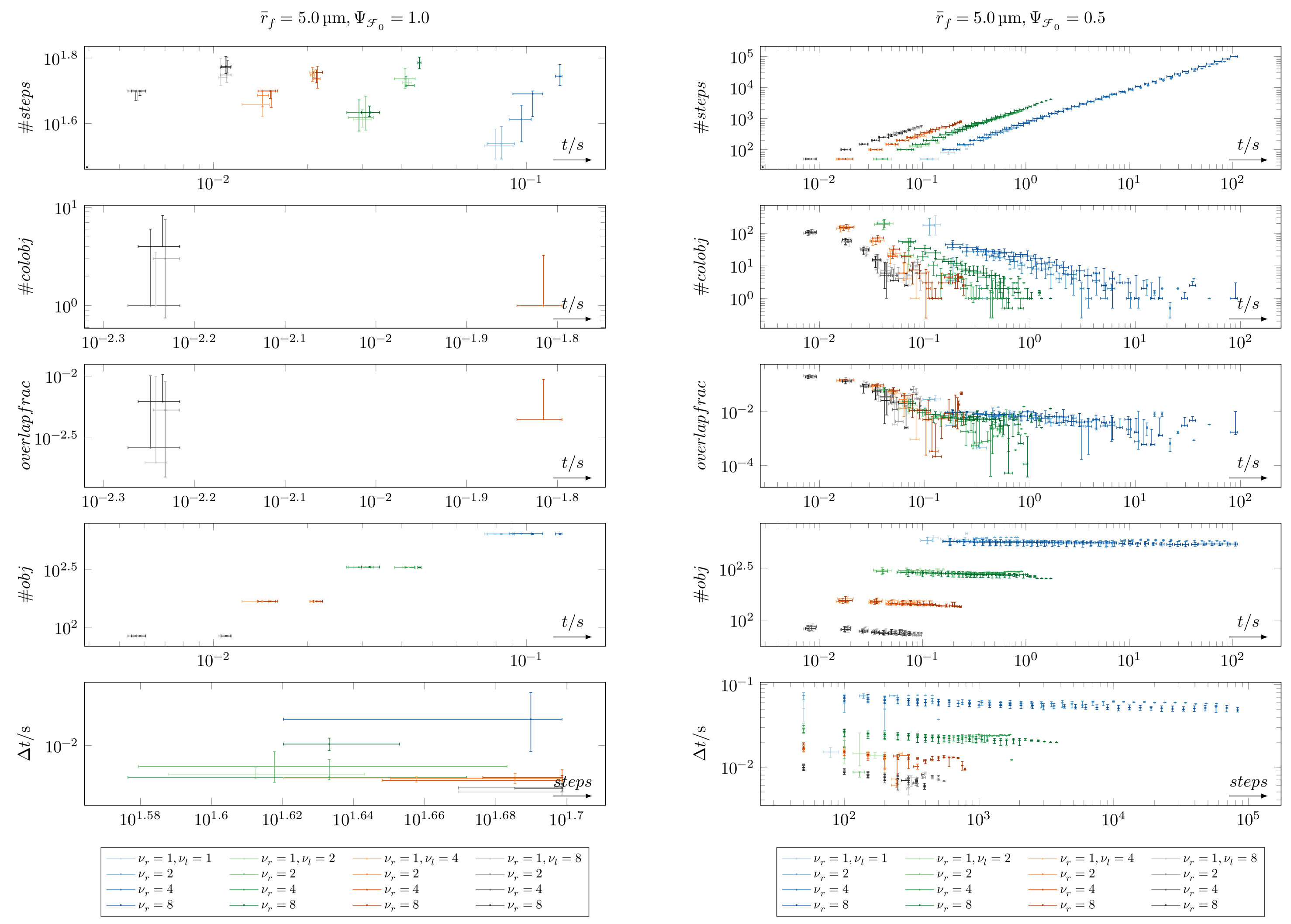


Fig. A.4.: Time evolution of the model building process of parallel and crossing fiber populations with $\bar{r}_f = 5.0\,\mu\text{m}$. Error bars indicate 25 % and 75 % quantiles (see section 8.3.1).

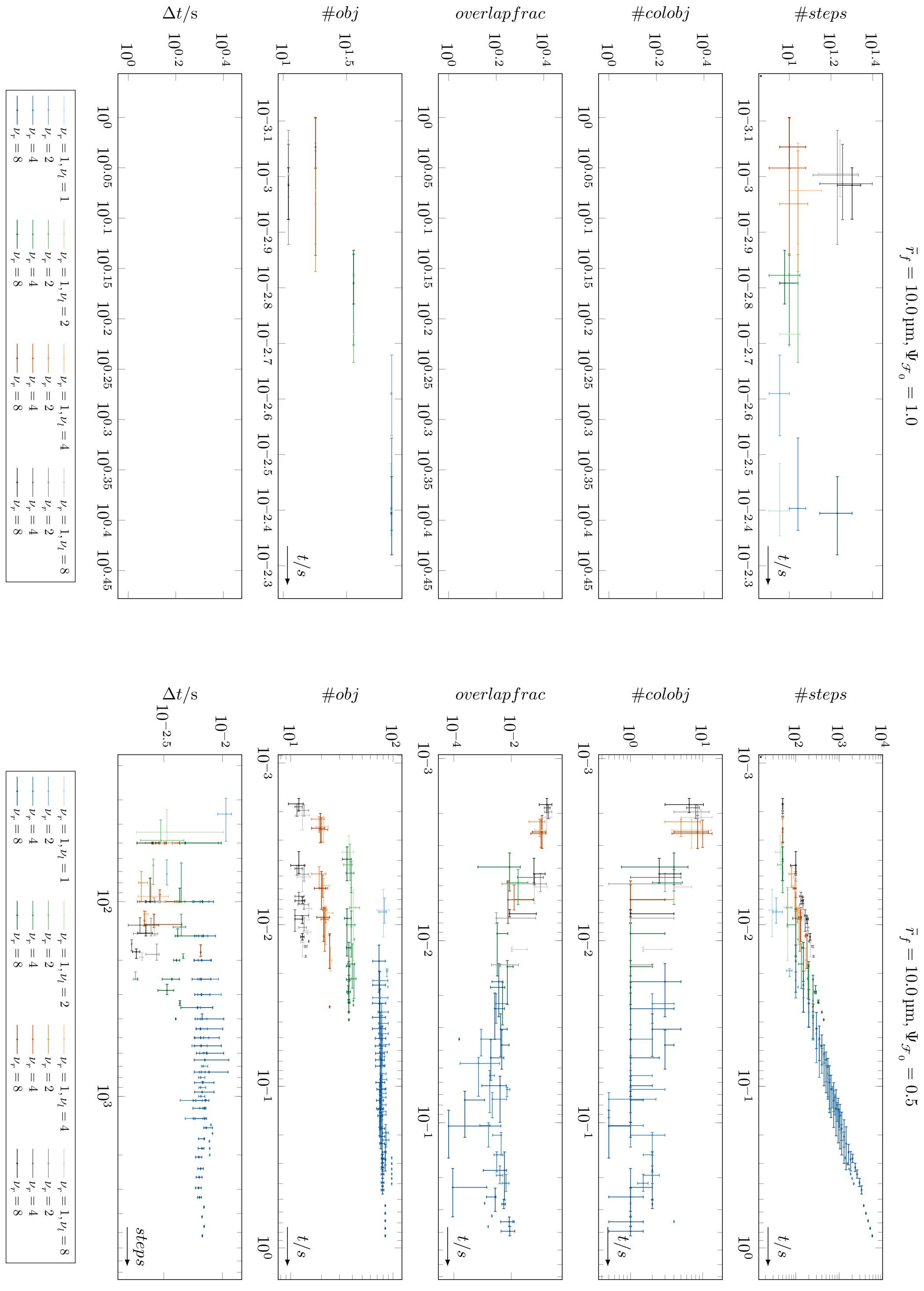


Fig. A.5.: Time evolution of the model building process of parallel and crossing fiber populations with $\bar{r}_f = 10.0\,\mu m$. Error bars indicate 25 % and 75 % quantiles (see section 8.3.1).

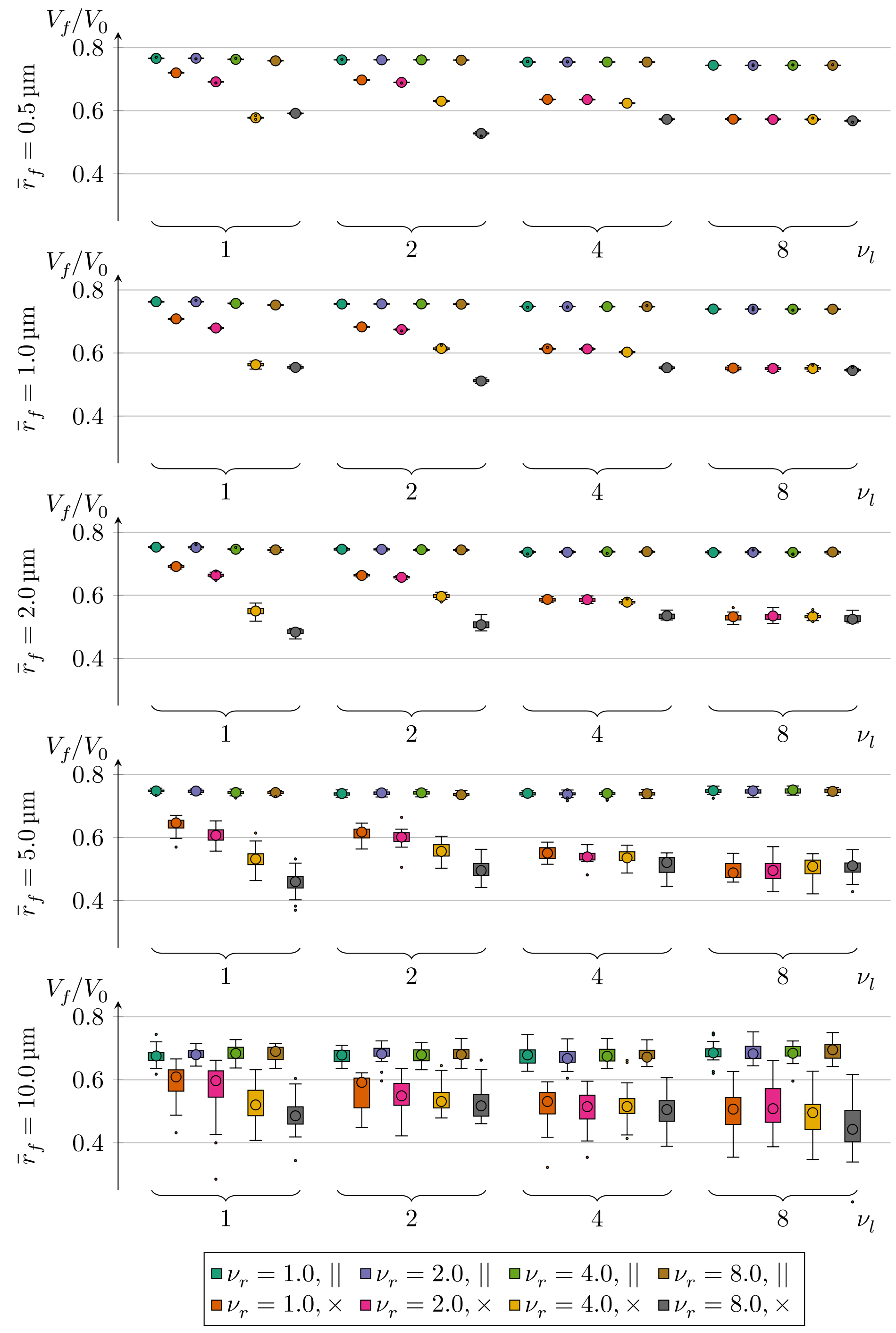


Fig. A.6.: Volume fractions V_f/V_0 for parallel ($||$) and crossing ($\times$) fiber populations of different relative fiber segment lengths ν_l and relative fiber bending radii ν_r and multiple mean fiber radii $\bar{r}_f$. (see section 8.3.1).

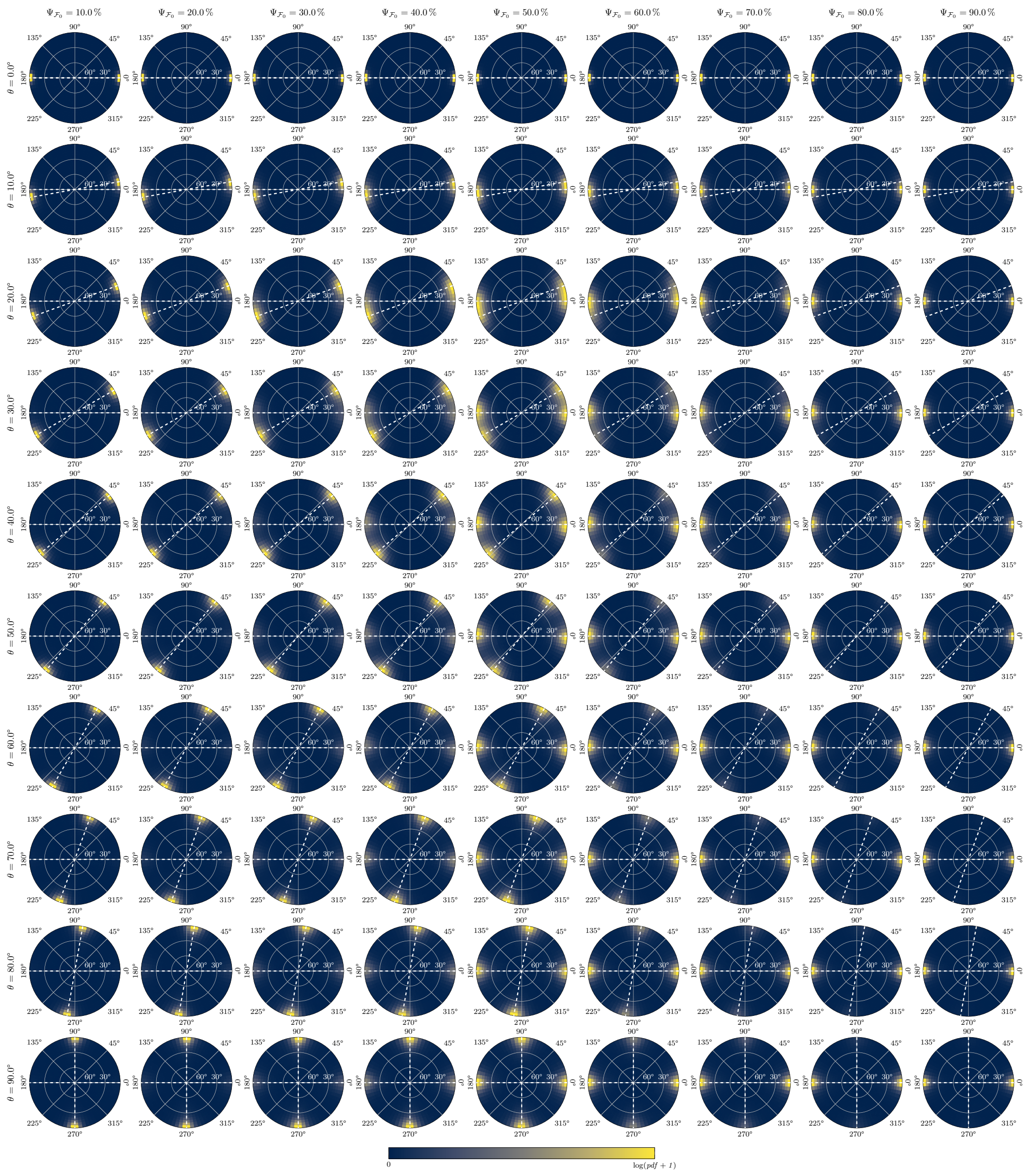


Fig. A.7.: Density distribution of fiber segment orientation in the simulation models. The value of the segments is weighted according to the area on a spherical surface and normalized so that the integral over one hemisphere is 1. The dashed white line indicates the orientation of the two fiber populations. (see section 8.4.1).

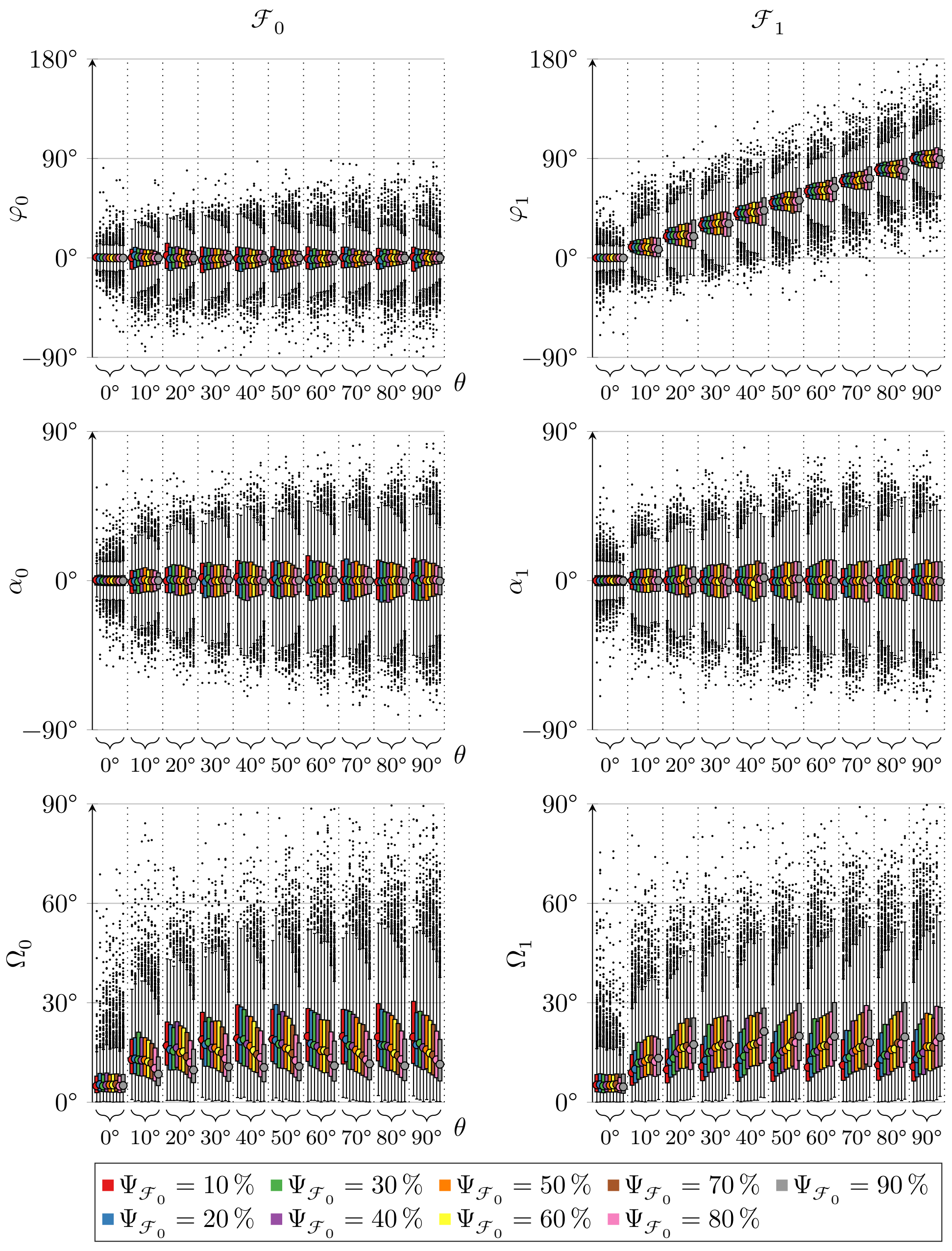


Fig. A.8.: Direction φ, inclination α and opening angle Ω distribution of the model library (see section 8.4.1).

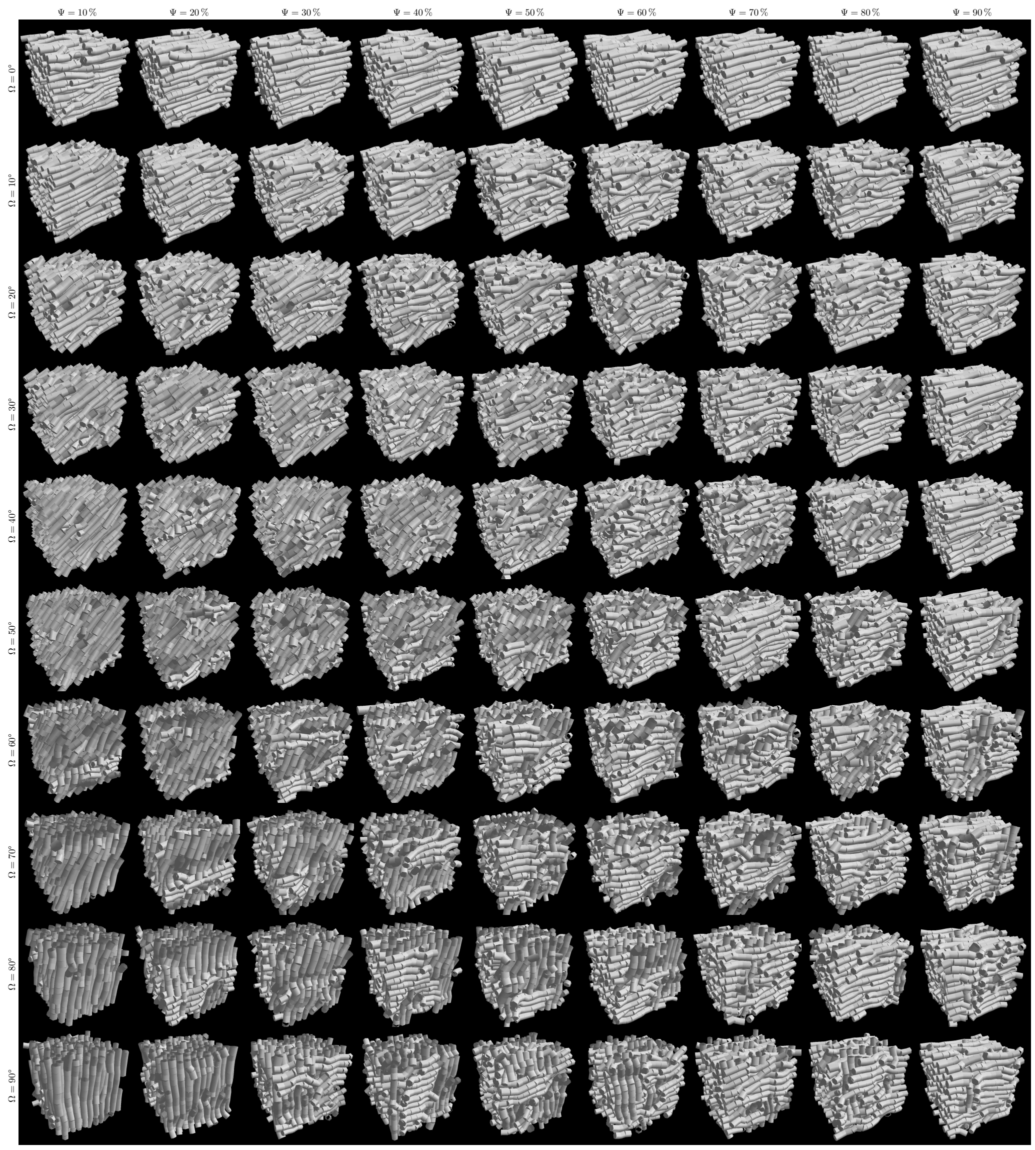
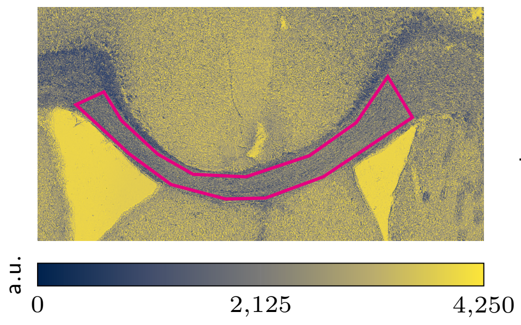


Fig. A.9.: D: Simulation model library. The inner 10 µm × 10 µm × 10 µm of the volume is shown (see section 8.4.1).

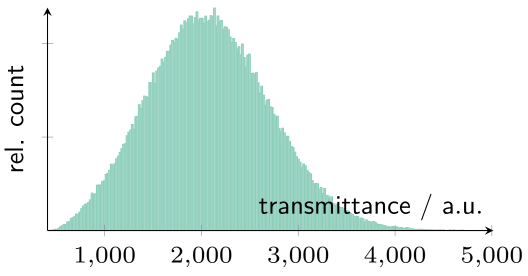
Tab. A.1.: Orientation statistic of simulation model library (see section 8.4.1).

Ω	Ψ	pop.	$<\alpha>$	$\sigma(\alpha)$	$25(\alpha)$	$50(\alpha)$	$75(\alpha)$	$<\varphi>$	$\sigma(\varphi)$	$25(\varphi)$	$50(\varphi)$	$75(\varphi)$	$<\Omega>$	$\sigma(\Omega)$	$25(\Omega)$	$50(\Omega)$	$75(\Omega)$
$^\circ$	%		$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$	$^\circ$
30	30	0	0	16	−10	0	10	−1	16	−11	−1	9	20	11	11	18	26
30	30	1	0	15	−9	0	9	30	13	22	31	38	17	11	9	15	23
30	60	0	0	16	−10	0	10	0	14	−9	−1	8	18	11	10	16	24
30	60	1	0	16	−10	0	10	31	16	21	31	40	19	11	11	17	26
30	90	0	0	12	−6	0	6	0	11	−6	0	6	13	9	6	10	17
30	90	1	0	15	−9	0	9	32	18	20	31	42	20	12	11	18	27
60	30	0	0	18	−11	0	11	−1	16	−9	−1	7	20	13	11	17	27
60	30	1	0	17	−10	0	9	60	13	53	60	67	18	13	9	15	24
60	60	0	0	18	−10	0	10	0	14	−8	−1	7	18	13	9	15	24
60	60	1	0	19	−11	0	11	60	16	53	61	69	20	13	10	17	27
60	90	0	0	14	−7	0	7	0	11	−6	0	5	14	11	7	11	18
60	90	1	0	18	−12	0	12	61	18	52	62	72	21	13	12	19	29
90	30	0	0	19	−12	0	12	0	16	−8	0	8	20	14	10	17	27
90	30	1	0	18	−10	0	10	90	13	84	90	96	18	13	8	14	24
90	60	0	0	19	−10	0	10	0	14	−7	0	7	18	14	9	15	25
90	60	1	0	19	−11	0	11	90	16	82	90	98	20	14	10	17	27
90	90	0	0	16	−8	0	8	0	10	−5	0	5	14	11	6	11	19
90	90	1	0	19	−12	0	12	90	19	81	90	100	22	14	12	19	29

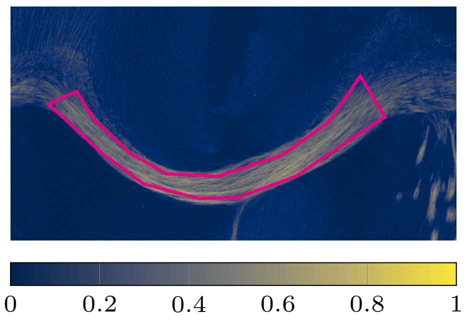
B Simulation



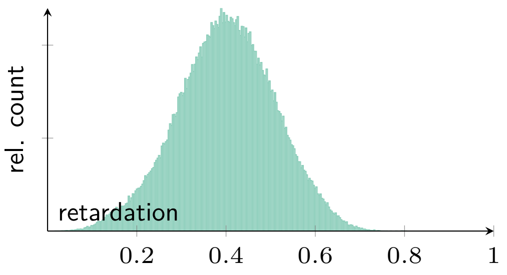
(a) Transmittance.



(b) Transmittance histogram.

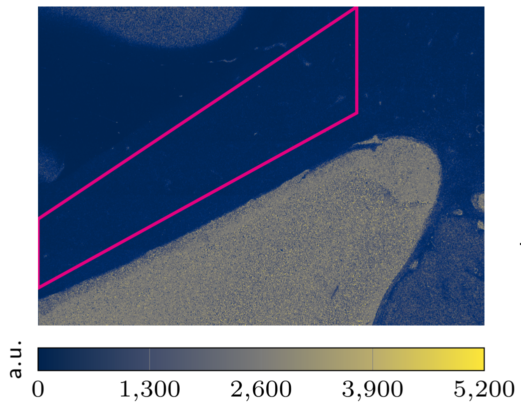


(c) Retardation.

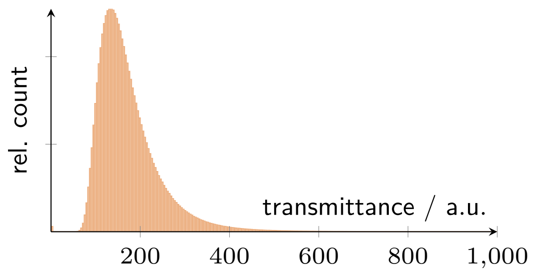


(d) Retardation histogram.

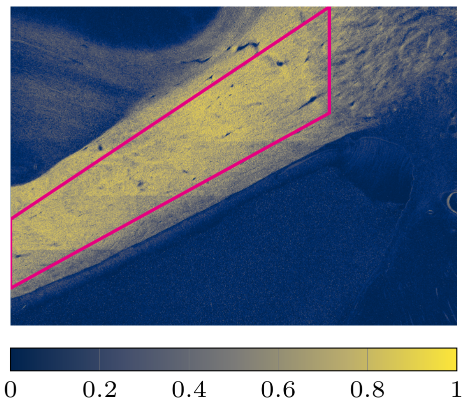
Fig. B.1.: Transmittance and retardation map of coronal section of a mouse (Brain id: PE-2020-00660-M, section: 127). The absorption coefficient and birefringence is estimated from the measurements in the corpus callosum (see section 9.2.1).



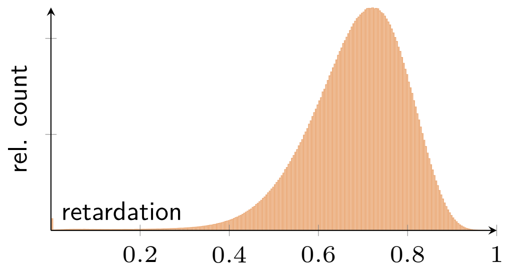
(a) Transmittance.



(b) Transmittance histogram.



(c) Retardation.



(d) Retardation histogram.

Fig. B.2.: Transmittance and retardation map of coronal section of the right hemisphere human coronal brain section (Brain id: PE-2011-00181-H, section 1006). The absorption coefficient and birefringence is estimated from the measurements in the corpus callosum (see section 9.2.1).

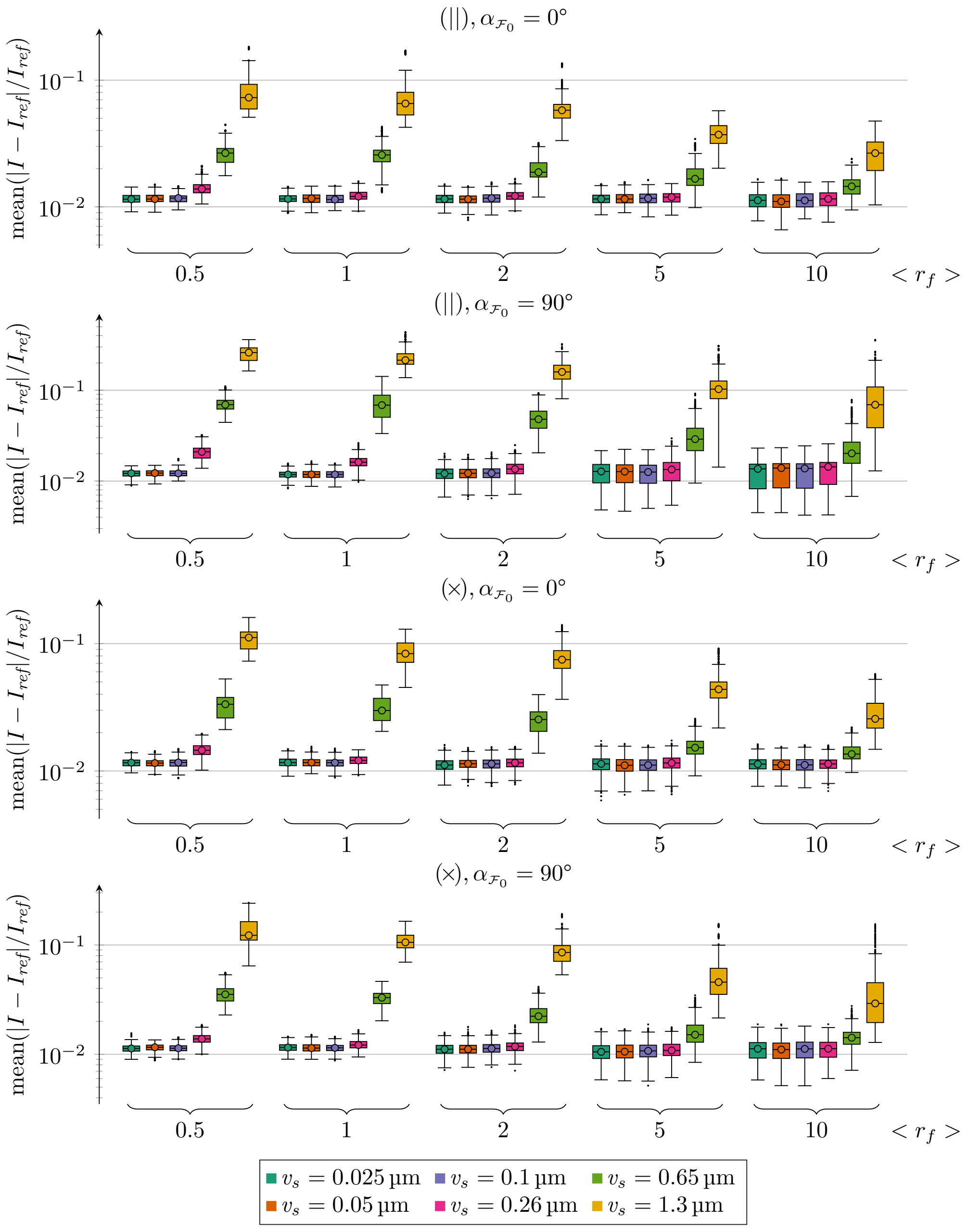


Fig. B.3.: Comparison of the simulation results with different mean fiber radii $\bar{r}_f$, voxel sizes v_s and fiber configurations. (see section 9.2.4).

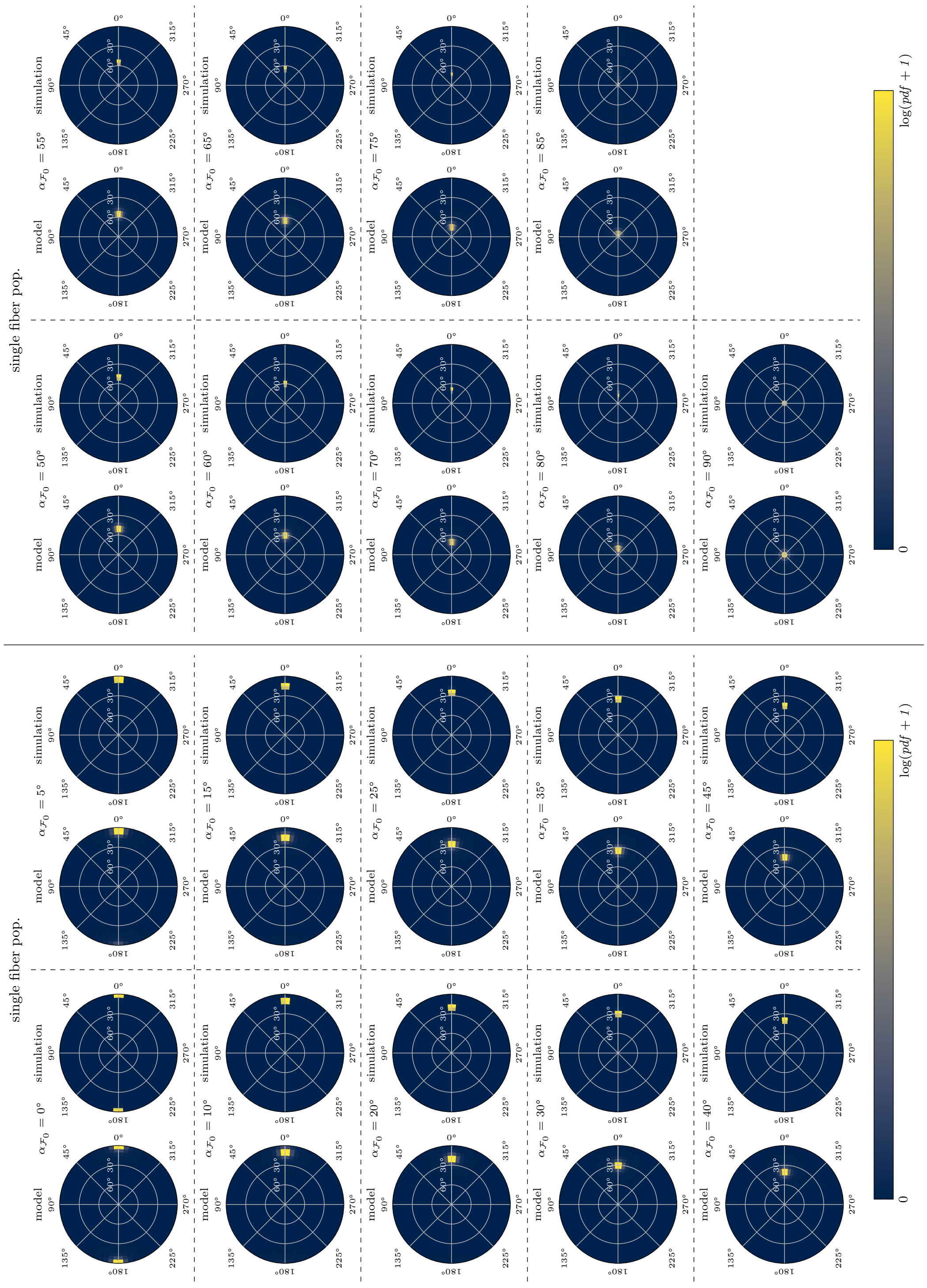


Fig. B.4.: Single fiber population orientation histograms of the initial fiber model and the simulation results (see section 9.3.2).

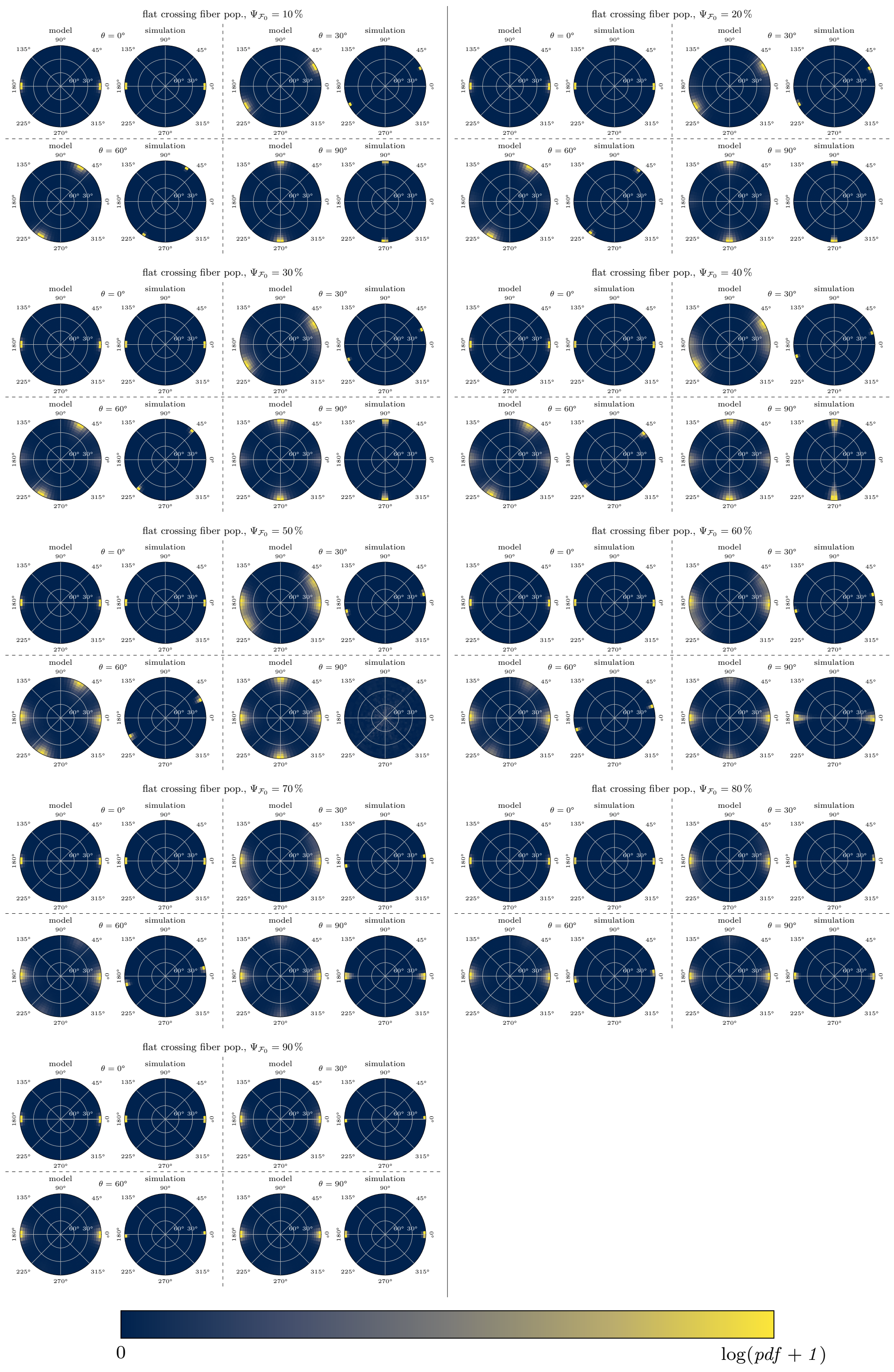


Fig. B.5.: Flat crossing fiber population (see section 9.3.3).

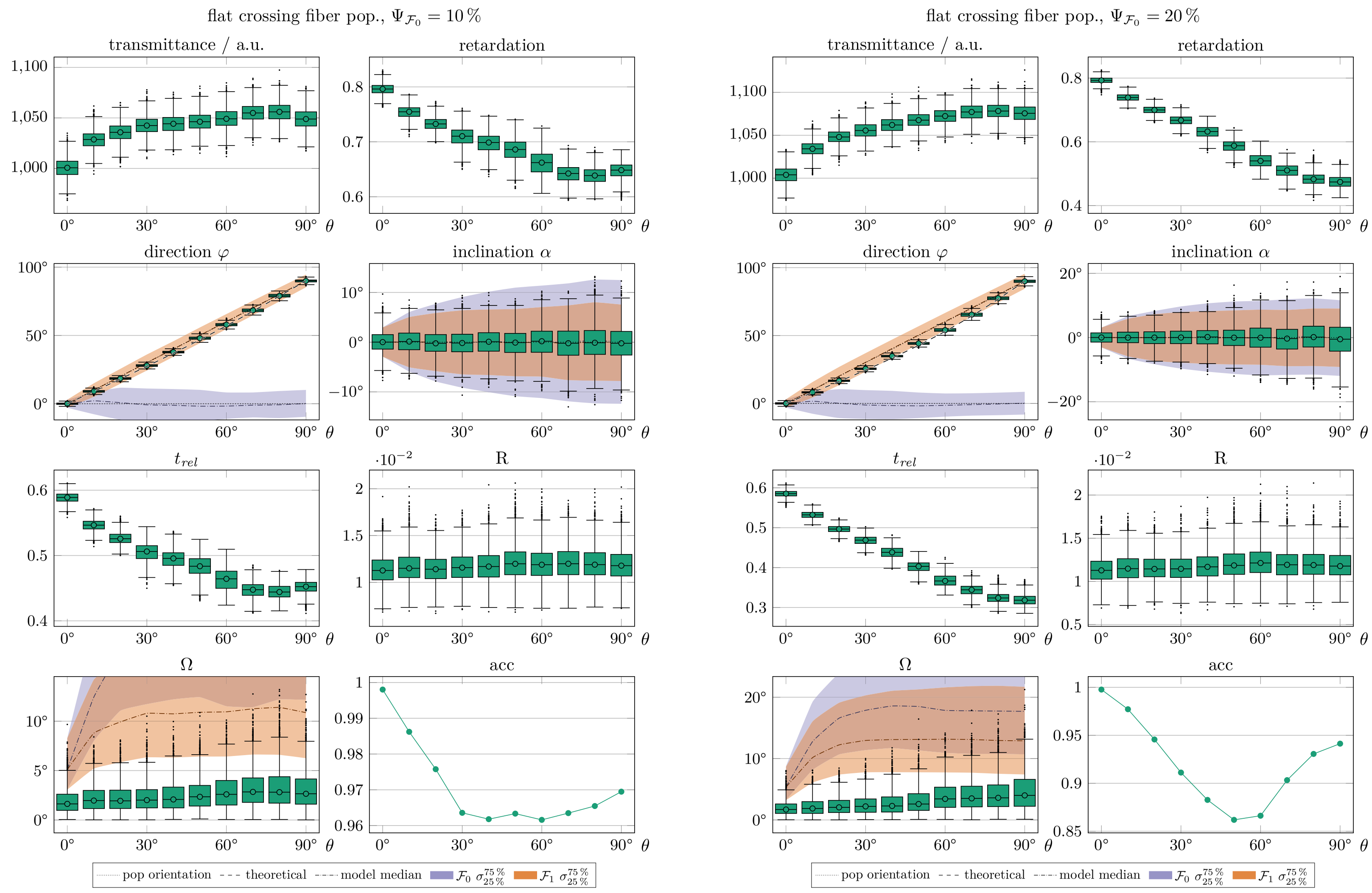


Fig. B.6.: Flat crossing fiber population. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.3).

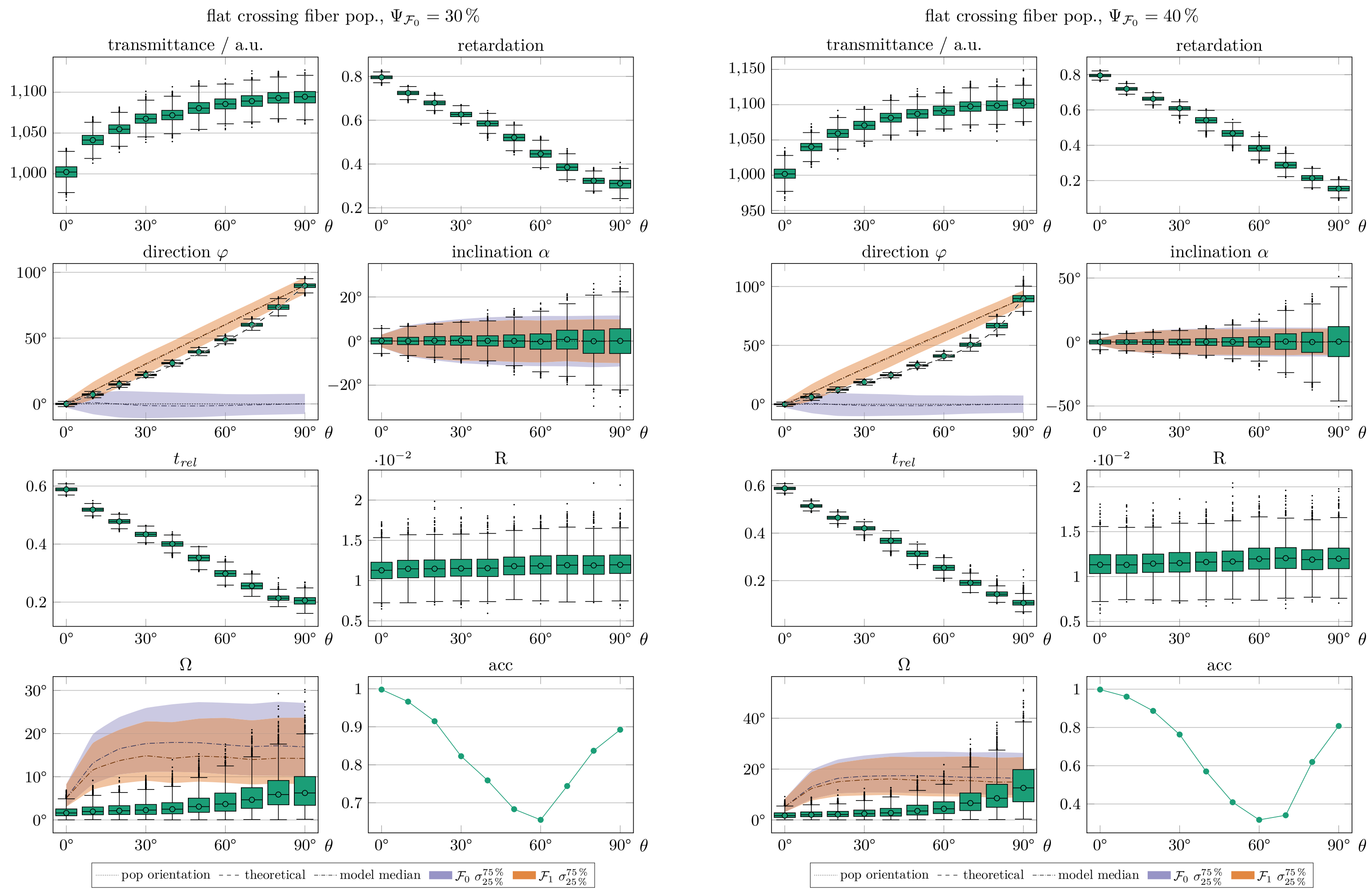


Fig. B.7.: Flat crossing fiber population. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.3).

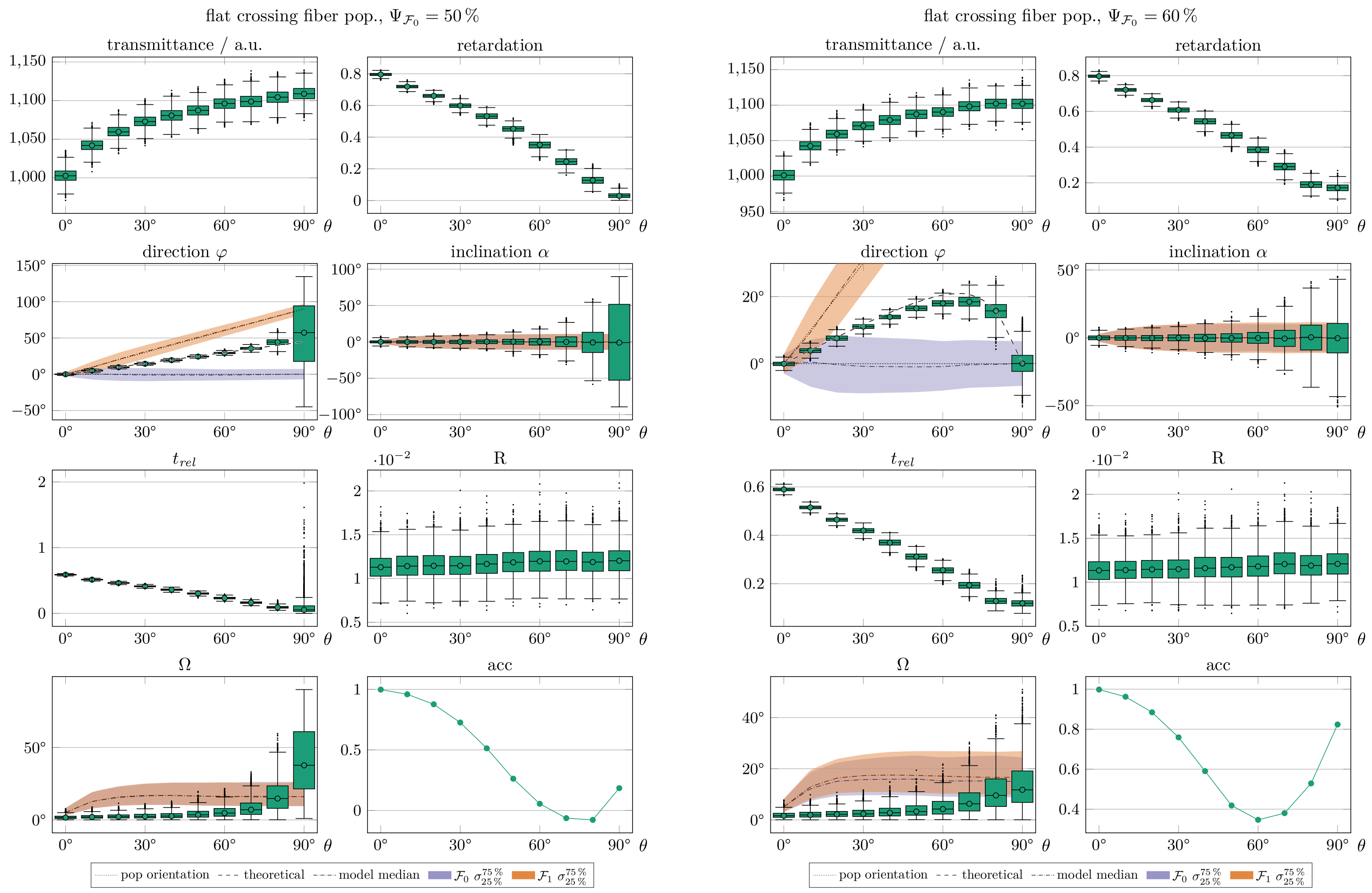


Fig. B.8.: Flat crossing fiber population. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.3).

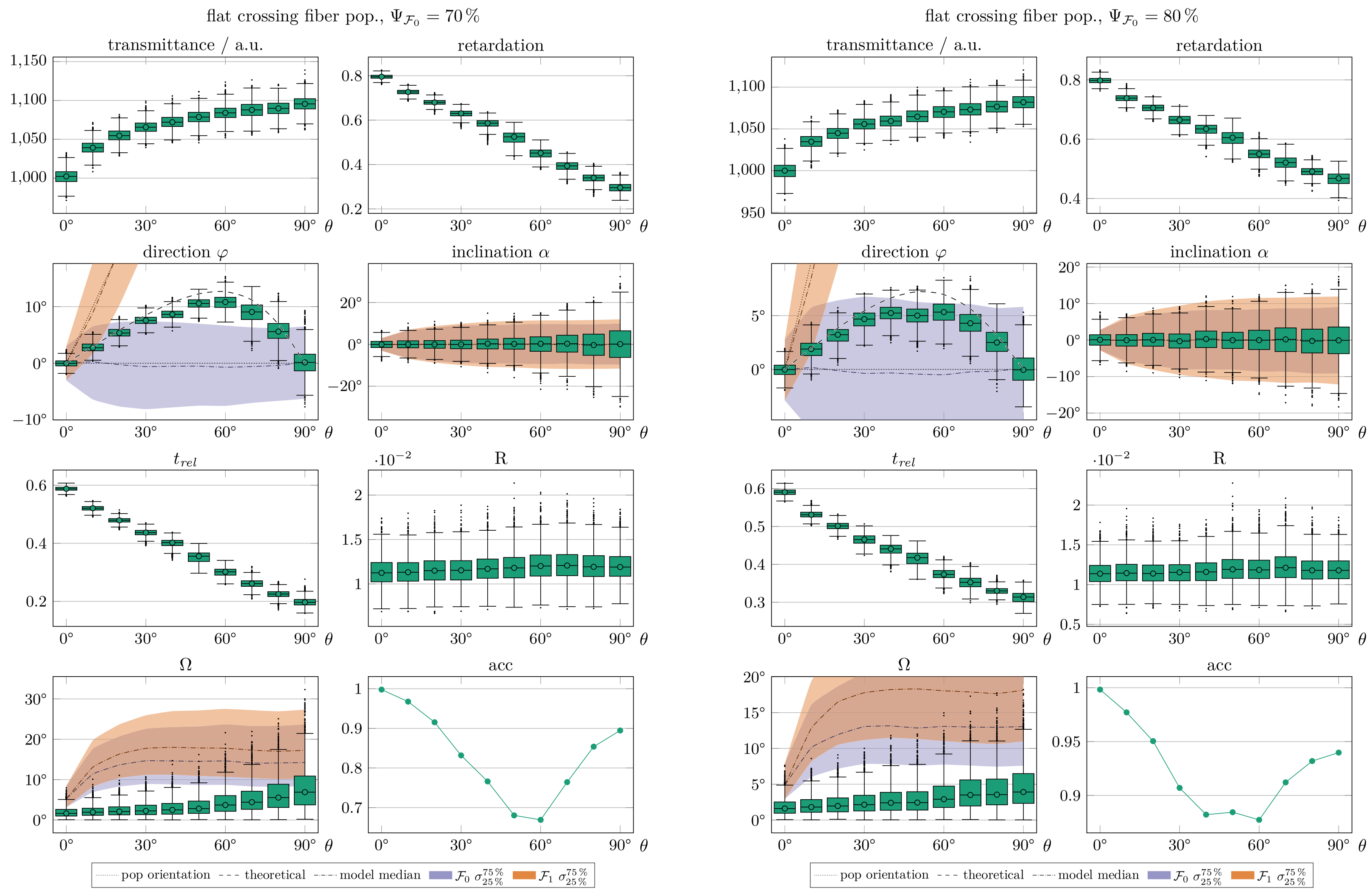


Fig. B.9.: Flat crossing fiber population. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.3).

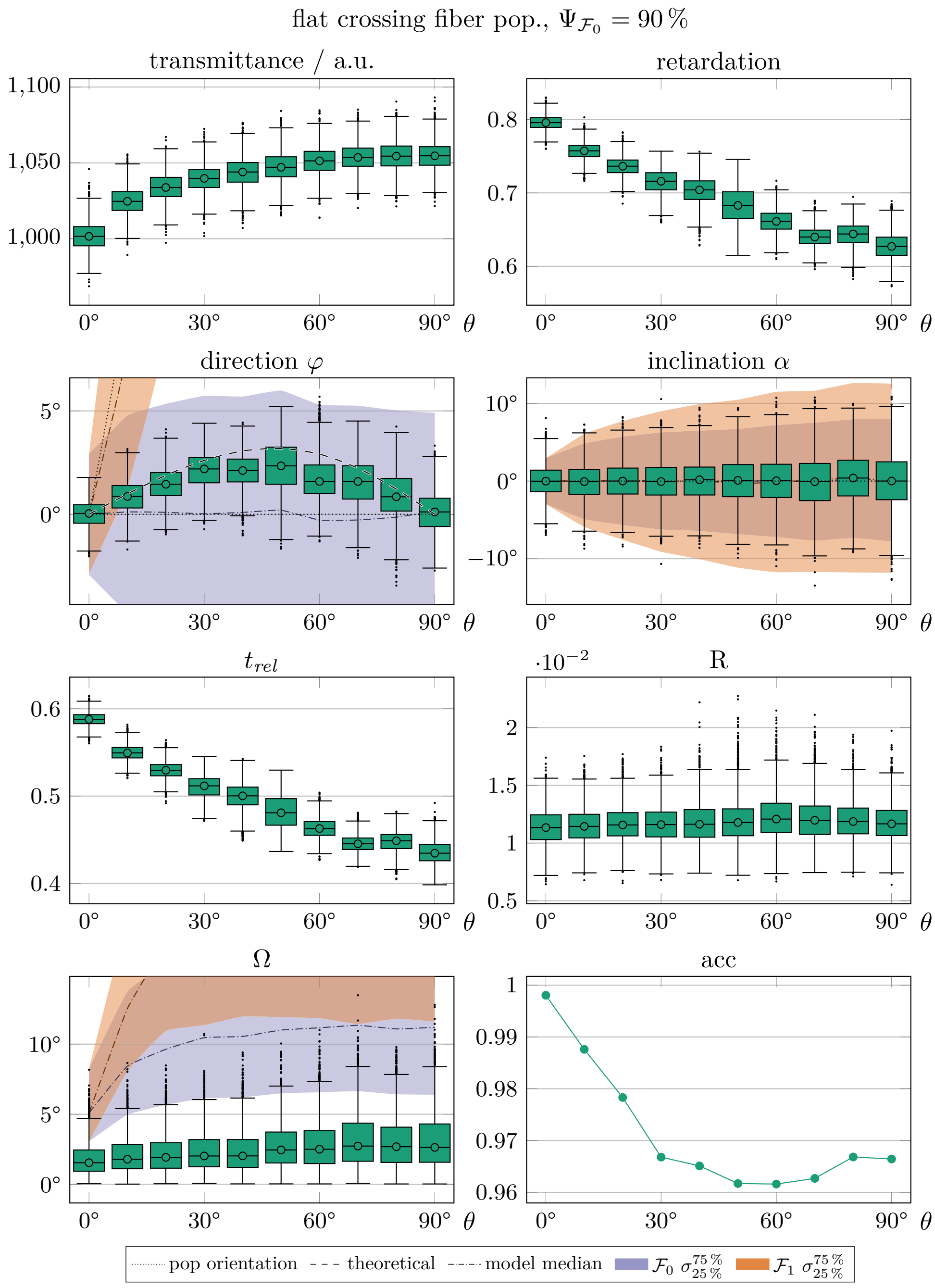


Fig. B.10.: Flat crossing fiber population. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.3).

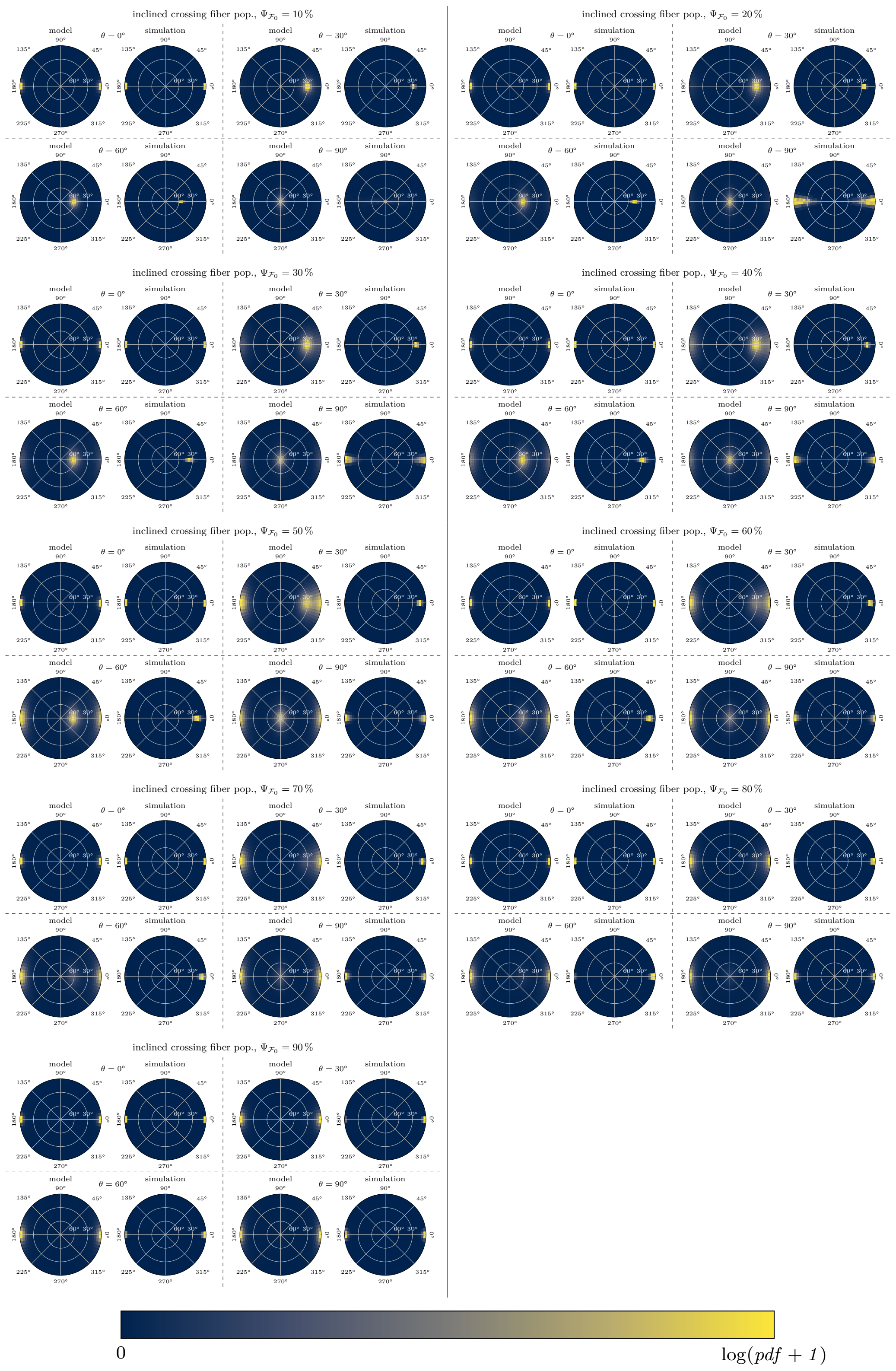


Fig. B.11.: Inclined crossing fiber population (see section 9.3.4).

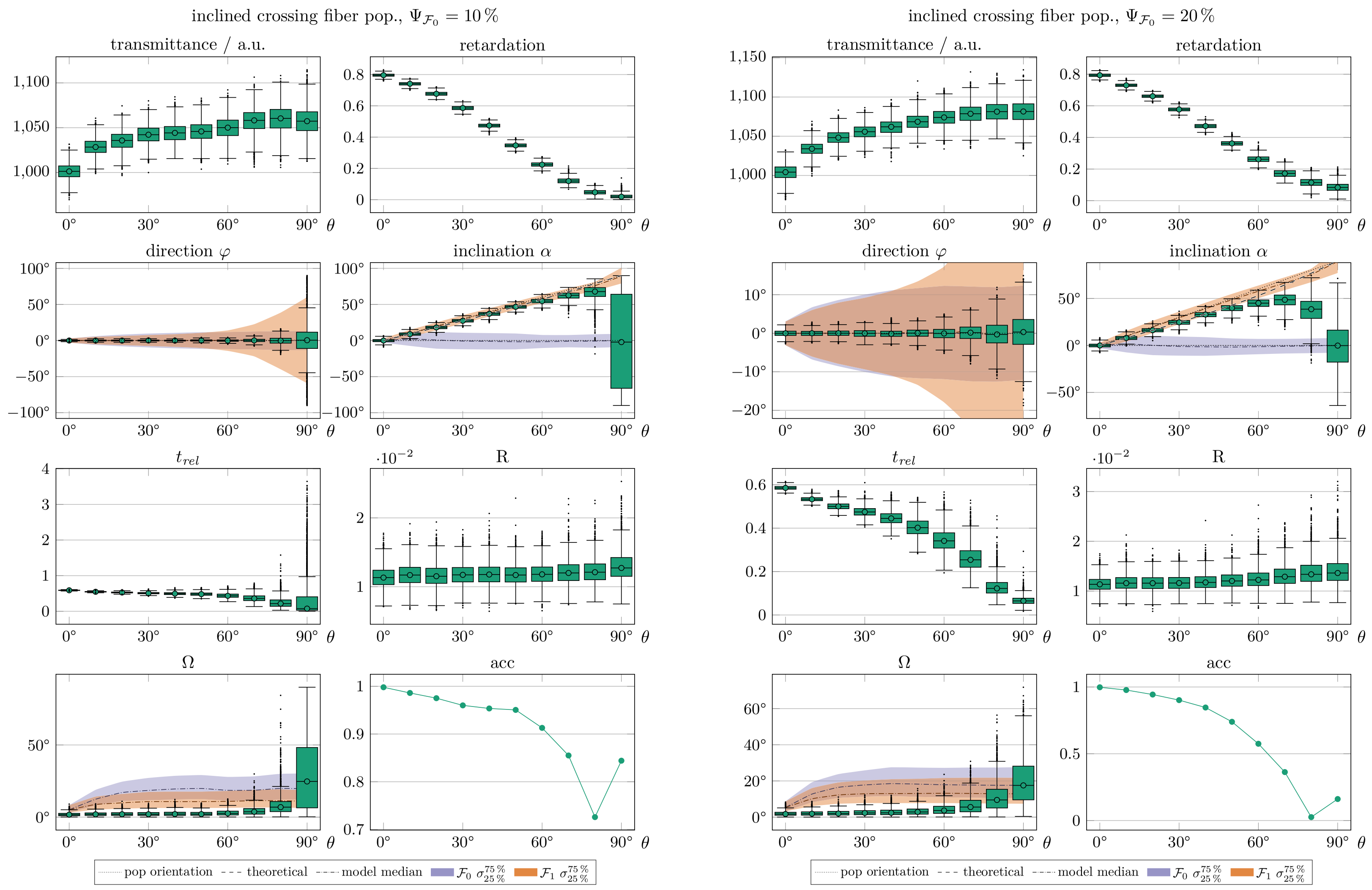


Fig. B.12.: Population of inclined crossing fibers. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.4).

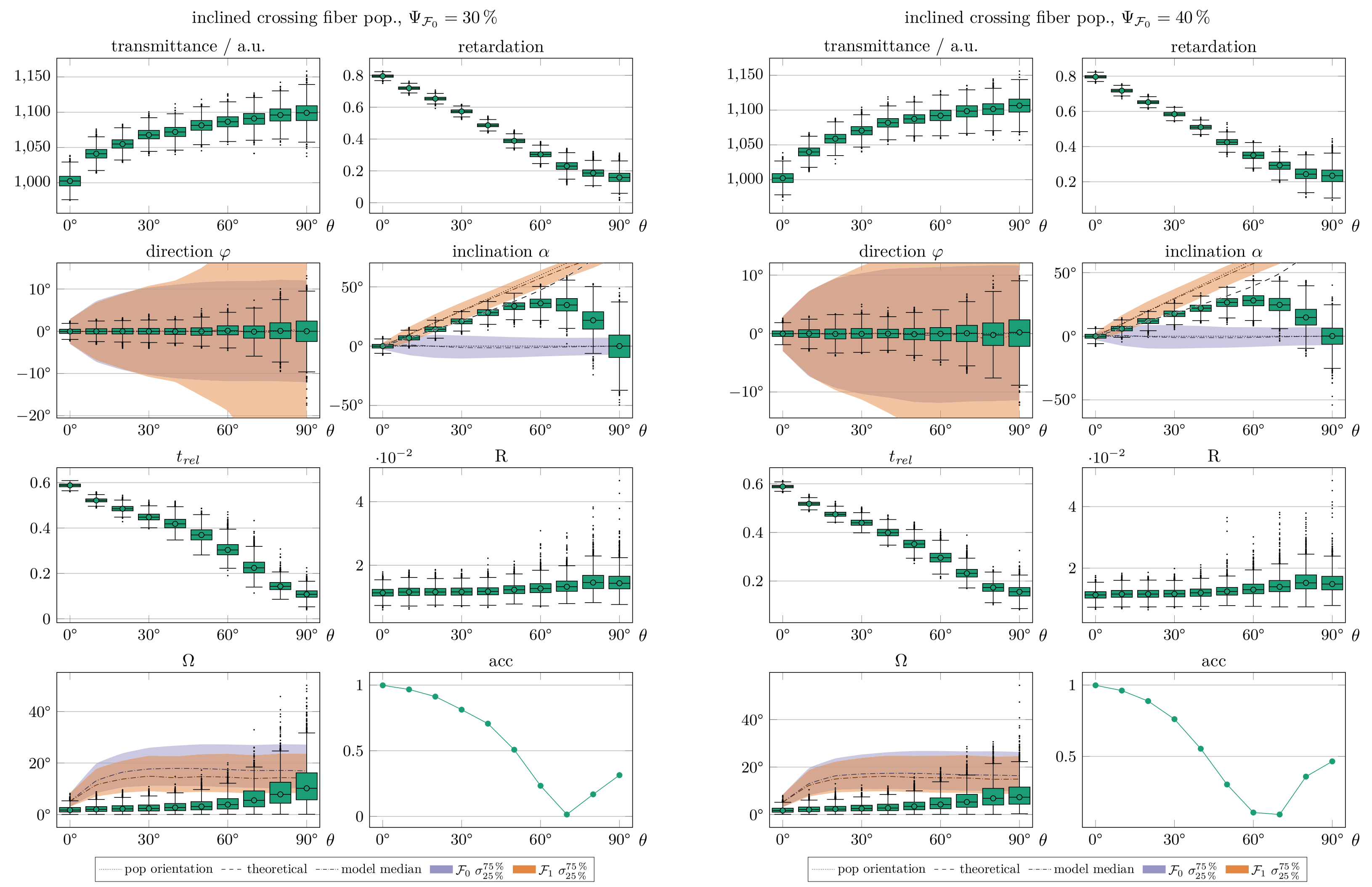


Fig. B.13.: Population of inclined crossing fibers. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.4).

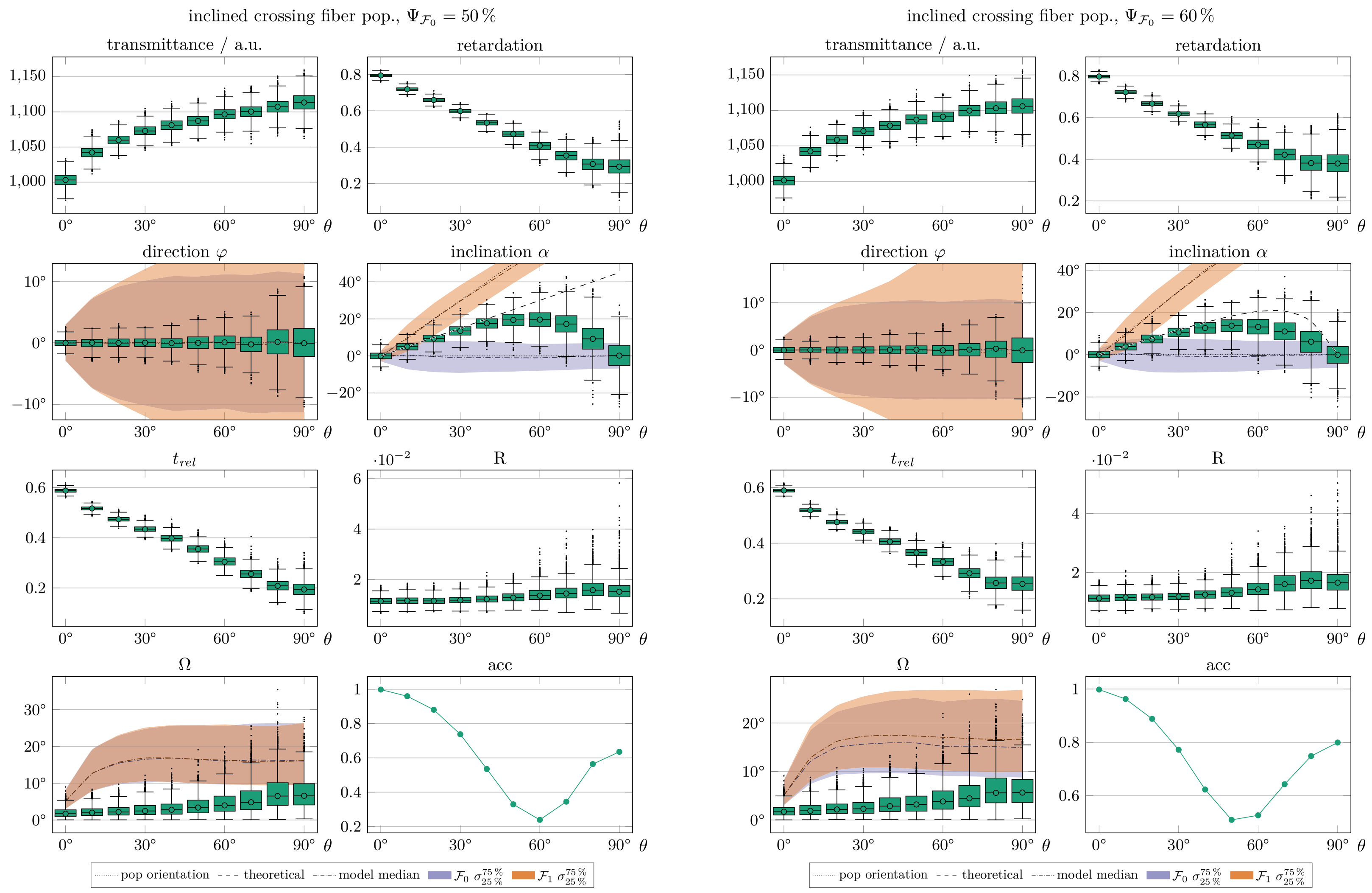


Fig. B.14.: Population of inclined crossing fibers. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.4).

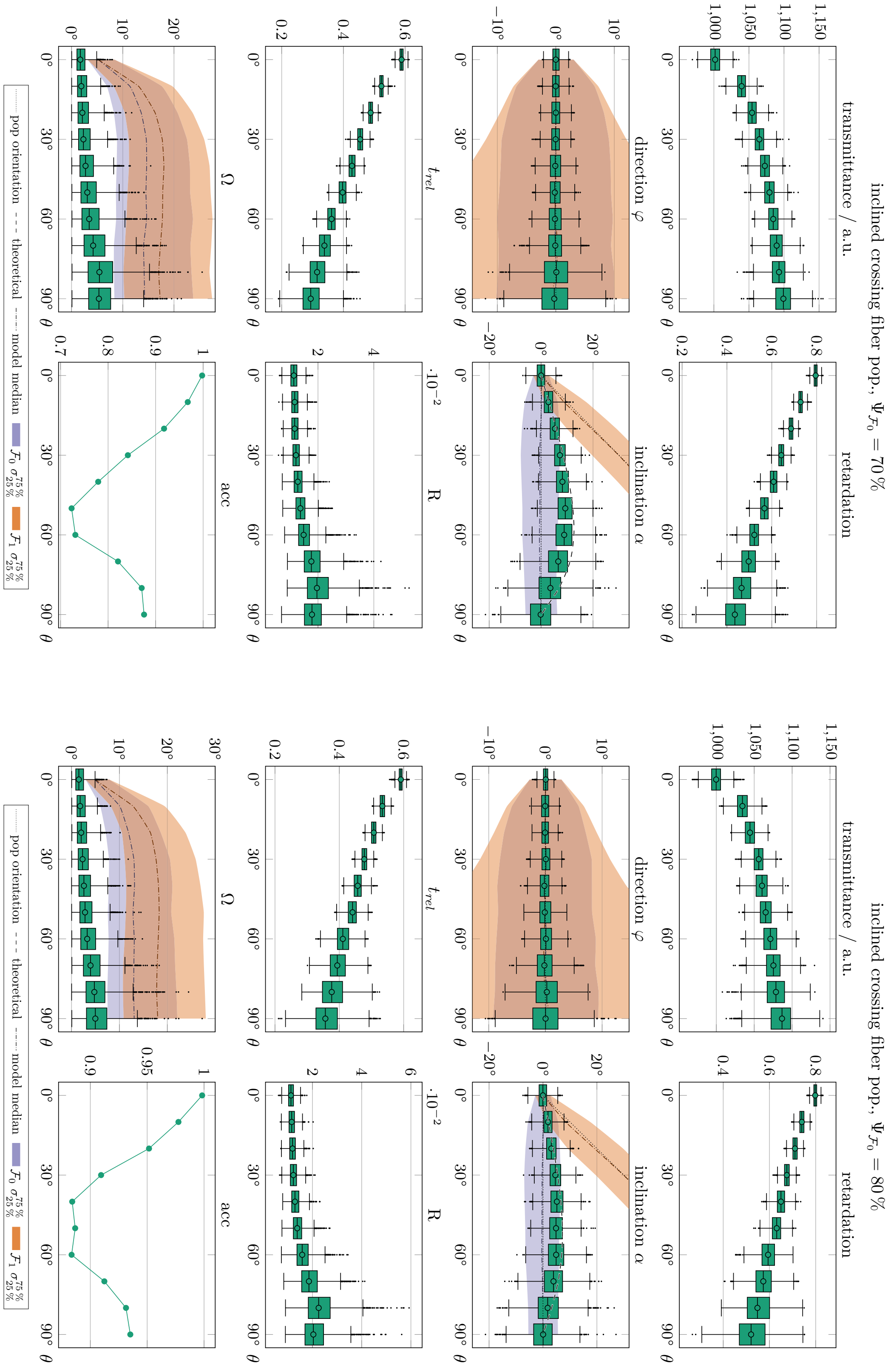


Fig. B.15.: Population of inclined crossing fibers. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.4).

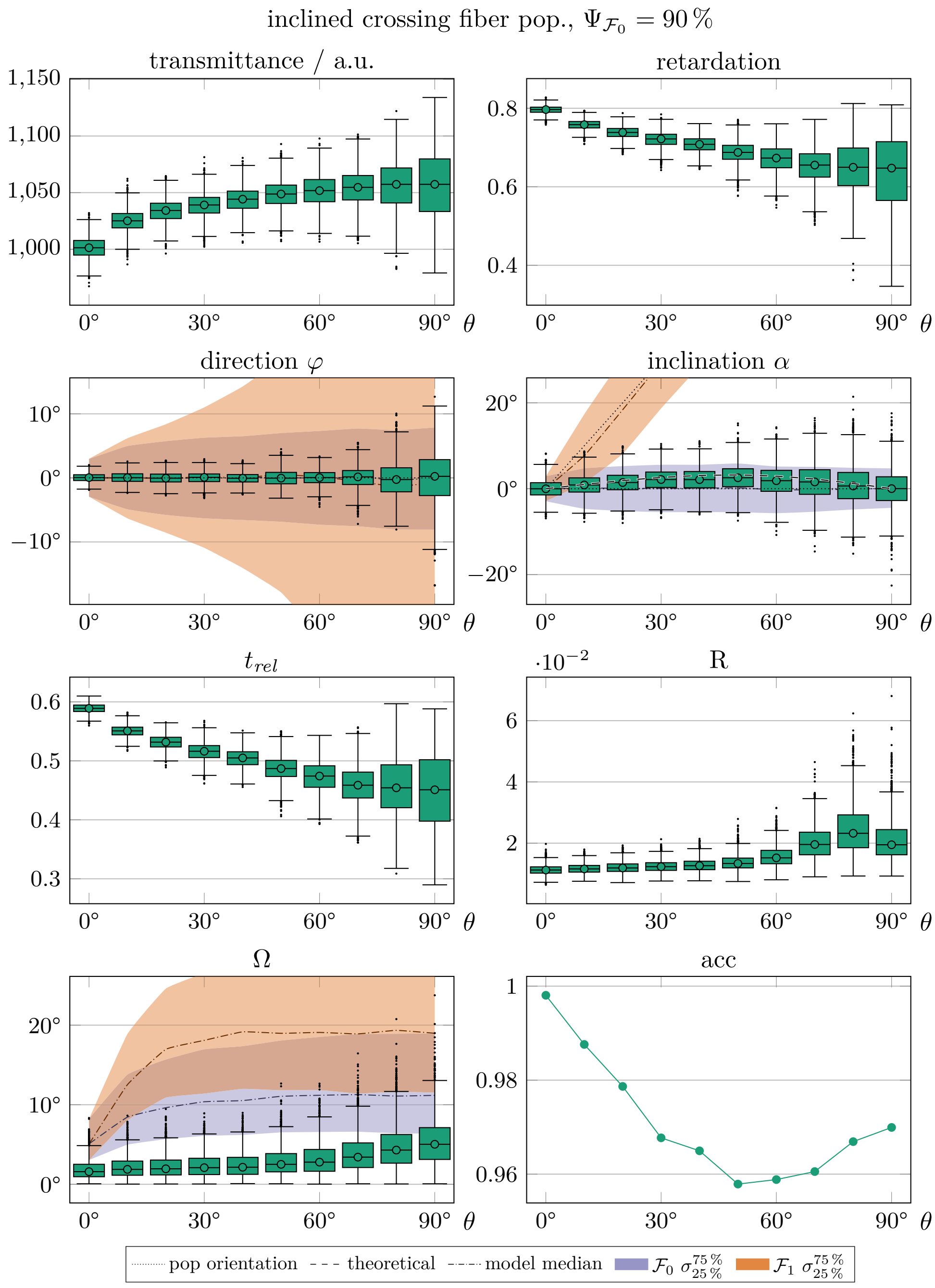


Fig. B.16.: Population of inclined crossing fibers. Results of the simulation analysis with comparison of the fiber models orientations (see section 9.3.4).

Declaration

Ich versichere an Eides Statt, dass die Dissertation von mir selbständig und ohne unzulässige fremde Hilfe unter Beachtung der „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf“ erstellt worden ist.

Ort, Datum	Felix Matuschke