



# Bachelor-Thesis

**Konstruktion von asymmetrischen Orientation Distribution Functions  
(ODFs) auf Basis von symmetrischen ODFs gewonnen aus der  
Polarisationsmikroskopie**

Tim Lockstedt  
1911311

Bachelor of Science Physik

Wuppertal, den 25. Januar 2024

Erstgutachter Prof. Dr.-Ing. Markus Axer  
Zweitgutachter Dr. Felix Matuschke



---

## Kurzfassung

Auf der Basis von Orientation Distribution Functions (ODFs) werden durch Sampling der Umgebung Informationen zur Konstruktion von asymmetrischen ODFs gesammelt. Das Samplen verwendet dabei abgerundete rotierte Kegel, zusammen mit einer Gaußschen Funktion zur positionsabhängigen Gewichtung.

Die Realisierung wird mithilfe von Python und diverser Bibliotheken ermöglicht. Dabei werden Code-Beispiele und verschiedene Methoden vorgestellt. Anhand von künstlichen ODF-Verteilungen wird die Konsistenz der Methoden gezeigt und die Wirkung der Parameter des Skripts erläutert. Anschließend werden am Beispiel von Daten aus dem Polarised Light Imaging (PLI) die 3D-Verformungen der ODFs zu asymmetrischen ODFs gezeigt.

## Abstract

Based on Orientation Distribution Functions (ODFs), neighborhood sampling is used to collect information for the construction of asymmetric ODFs. The sampling uses rounded rotated cones, together with a Gaussian function for position-dependent weighting.

The realization is made possible with the help of Python and various libraries. Code examples are provided and various methods are presented. Using artificial ODF distributions, the consistency of the methods is shown and the effect of the script parameters is explained. The 3D deformations of the ODFs to AODFs are then shown using the example of Polarised Light Imaging (PLI) data.

---

# Inhaltsverzeichnis

|  |           |
|--|-----------|
| <b>1 Einleitung</b>  | <b>1</b>  |
| 1.1 Motivation . . . . .   | 1         |
| 1.2 Problemstellung & Ziele . . . . .                                      | 1         |
| 1.3 Aufbau der Thesis . . . . .  | 1         |
| <b>2 Theorie</b>   | <b>3</b>  |
| 2.1 Kugelflächenfunktionen . . . . .                                       | 3         |
| 2.2 Orientation Distribution Functions . . . . .                           | 3         |
| 2.3 Asymmetric Orientation Distribution Functions . . . . .                | 5         |
| 2.4 Definition des mathematischen und abgerundeten Kegels . . . . .        | 5         |
| <b>3 Entwurf</b>   | <b>6</b>  |
| 3.1 Methoden zur Bestimmung der Punkte innerhalb rotierter Kegel . . . . . | 7         |
| 3.1.1 Rotation des Kegels . . . . .  | 7         |
| 3.1.2 Rotation des Koordinatensystems . . . . .                            | 7         |
| 3.1.3 Vergleich der Rotationsmethoden . . . . .                            | 8         |
| 3.2 Konstruktion der AODF Oberfläche . . . . .                             | 8         |
| 3.3 Gewichte . . . . .   | 9         |
| <b>4 Realisierung</b>  | <b>10</b> |
| 4.1 Erstellung der Kegel . . . . .   | 11        |
| 4.1.1 Cache . . . . .  | 12        |
| 4.1.2 Validierung . . . . .  | 13        |
| 4.1.3 Zufallswinkel . . . . .  | 16        |
| 4.1.4 Homogen verteilte Punkte auf einer Kugeloberfläche . . . . .         | 16        |
| 4.2 Gewichtung der ODFs in den Kegeln . . . . .                            | 18        |
| 4.3 Zusammenfassen der Kegel zu Amplituden . . . . .                       | 18        |
| 4.4 Generieren der AODFs . . . . .   | 20        |
| <b>5 Analyse</b>   | <b>22</b> |
| 5.1 Resultate . . . . .  | 22        |
| 5.2 Diskussion . . . . .   | 26        |
| 5.2.1 Zufällig und homogen verteilte Winkel . . . . .                      | 26        |
| 5.2.2 Große Stern-Verteilung . . . . .                                     | 27        |
| 5.2.3 Große zirkuläre Verteilung . . . . .                                 | 28        |
| 5.2.4 Linienförmige AODFs . . . . .  | 29        |
| 5.2.5 PLI-Daten . . . . .  | 30        |
| <b>6 Schlussbetrachtungen</b>  | <b>32</b> |
| 6.1 Fazit . . . . .  | 32        |
| 6.2 Ausblick . . . . .   | 32        |

---

---

**Literatur** **35**

**A Anhang** **36**



---

# 1 Einleitung

## 1.1 Motivation

In der diffusionsgewichteten MRT-Bildgebung werden Traktographie-Algorithmen verwendet, um die Verbindung zwischen Gehirnregionen zu motivieren. Diese Algorithmen basieren auf sogenannten Orientation Distribution Functions, kurz ODFs, welche die im MRT gemessenen Diffusionswahrscheinlichkeiten übersichtlich darstellen (mehr dazu in [Mar22]).

Zur Anwendung von ODF-basierten Traktographie-Algorithmen im Bereich des Polarised Light Imaging (PLI) müssen die dort gemessenen Daten entsprechend in ODFs umgerechnet werden (mehr dazu in [Mar11]). Da die gemessenen Daten symmetrisch sind, folgen daraus symmetrische ODFs. Der Verlauf der Nervenfasern im Gehirn ist im Allgemeinen von der Form nicht symmetrisch und kann somit durch symmetrische ODFs nur begrenzt angenähert werden. Die Idee der asymmetrischen ODFs (AODFs) kommt aus dieser Überlegung. AODFs beschreiben den statistischen Verlauf der Nervenfasern besser als symmetrische ODFs und führen zu potenziell besseren Ergebnissen der Traktographie.

## 1.2 Problemstellung & Ziele

Die Konstruktion der AODFs ist bisher nicht weit etabliert. Da die in Abschnitt 1.1 genannten gewonnenen ODFs im PLI symmetrisch sind, können sie nicht direkt in asymmetrische ODFs umgerechnet werden. Für die Konstruktion der AODFs sind ebenfalls asymmetrische Daten erforderlich, die zunächst ermittelt werden müssen. Sie können aus dem Umfeld der einzelnen ODFs berechnet werden.

Dabei wird sich an dem Paper „Asymmetric Orientation Distribution Functions (AODFs) revealing intravoxel geometry in diffusion MRI“ ([Suh18]) orientiert. Die darin beschriebene Kegel-Methode soll für das Sammeln von Daten rekonstruiert und auf PLI-Messdaten angewandt werden. Die auf der Basis dieser Daten konstruierten AODFs sollen die Umgebung der ODFs und den Verlauf der zugrundeliegenden Nervenfasern, durch die freie Asymmetrische Form, genauer beschreiben.

## 1.3 Aufbau der Thesis

Basierend auf dem zuvor genannten Paper ([Suh18]) soll die dort verwendete Kegel-Methode zum Sammeln von Daten für die Konstruktion von AODFs genutzt werden. Dabei wird die dort beschriebene Methode explizit für die Anwendung auf PLI-Messdaten rekonstruiert und anhand von Code-Beispielen erklärt. Der vollständige Code ist auf <https://github.com/TimLockstedt/BA> zu finden. Anschließend wird die Funktionsweise analysiert und diskutiert.

Zu Beginn (Kapitel 2) werden die Grundlagen von Kugelflächenfunktionen eingeführt. Diese bilden das Fundament für die Konstruktion von symmetrischen ODFs. Der Aufbau jener wird anschließend, zusammen mit dem Unterschied zu den asymmetrischen ODFs, besprochen.

Für die später verwendete Kegel-Methode werden die Definitionen für den mathematischen und den dort eingeführten abgerundeten Kegel aufgeführt.

Um den Prozess der Konstruktion der AODFs besser zu verstehen wird dieser Schritt für Schritt geplant und durchgegangen: Von der Bestimmung der Punkte innerhalb eines Kegels zu der Rotation bis hin zur Umrechnung zu nutzbaren Daten für die Konstruktion der AODFs (Kapitel 3).

In Kapitel 4 wird daraufhin die praktische Umsetzung mithilfe der Hochsprache Python und diverser Bibliotheken gezeigt. Insbesondere werden Code-Beispiele für die konkrete Umsetzung, hilfreiche Methoden zur Validierung der Ergebnisse und alternative Herangehensweisen vorgestellt.

Das daraus entstehende vollständige Skript wird verwendet, um AODFs für 3 künstlich konstruierte ODF-Verteilungen und einen Ausschnitt reeller PLI-Daten zu berechnen (Kapitel 5). Die Ergebnisse werden anschließend zusammen mit den relevanten Parametern des Skriptes diskutiert.

Abschließend wird in Kapitel 6 ein Fazit gezogen und weitere Verbesserungsmöglichkeiten des Skriptes angesprochen.

---

## 2 Theorie

### 2.1 Kugelflächenfunktionen

Kugelflächenfunktionen werden für die Konstruktion von sogenannten ODFs verwendet. Diese werden in Abschnitt 2.2 genauer erläutert. Kugelflächenfunktionen sind Teil der Lösung der Laplace-Gleichung

$$\Delta f(r, \theta, \phi) = 0 \quad (2.1)$$

in Kugelkoordinaten. Dort stellen sie die Lösung für den Polar- und Azimutwinkel  $\theta$  und  $\phi$  dar. Da keine Informationen über die Phase aus den Messungen in der PLI-Messung gewonnen wird, ist der Imaginärteil im Kontext der ODFs nicht relevant. Der Realteil kann dreidimensional visualisiert werden.

**Tabelle 2.1** Reelle Anteile der Kugelflächenfunktionen ( $\phi$  und  $\theta$  aus Kugelkoordinaten) [Ini]

| $Y_{lm}(\theta, \phi)$ | $l = 0$                 | $l = 1$  | $l = 2$   |
|------------------------|-------------------------|--|---|
| $m = -2$               |                         |  | $\sqrt{\frac{15}{32\pi}} \sin \theta^2 \sin 2\phi$          |
| $m = -1$               |                         | $\sqrt{\frac{3}{8\pi}} \sin \theta \sin \phi$  | $\sqrt{\frac{15}{8\pi}} \sin \theta \cos \theta \sin \phi$  |
| $m = 0$                | $\sqrt{\frac{1}{4\pi}}$ | $\sqrt{\frac{3}{4\pi}} \cos \theta$            | $\sqrt{\frac{5}{16\pi}} (3 \cos \theta^2 - 1)$              |
| $m = 1$                |                         | $-\sqrt{\frac{3}{8\pi}} \sin \theta \cos \phi$ | $-\sqrt{\frac{15}{8\pi}} \sin \theta \cos \theta \cos \phi$ |
| $m = 2$                |                         |  | $\sqrt{\frac{15}{32\pi}} \sin \theta^2 \cos 2\phi$          |

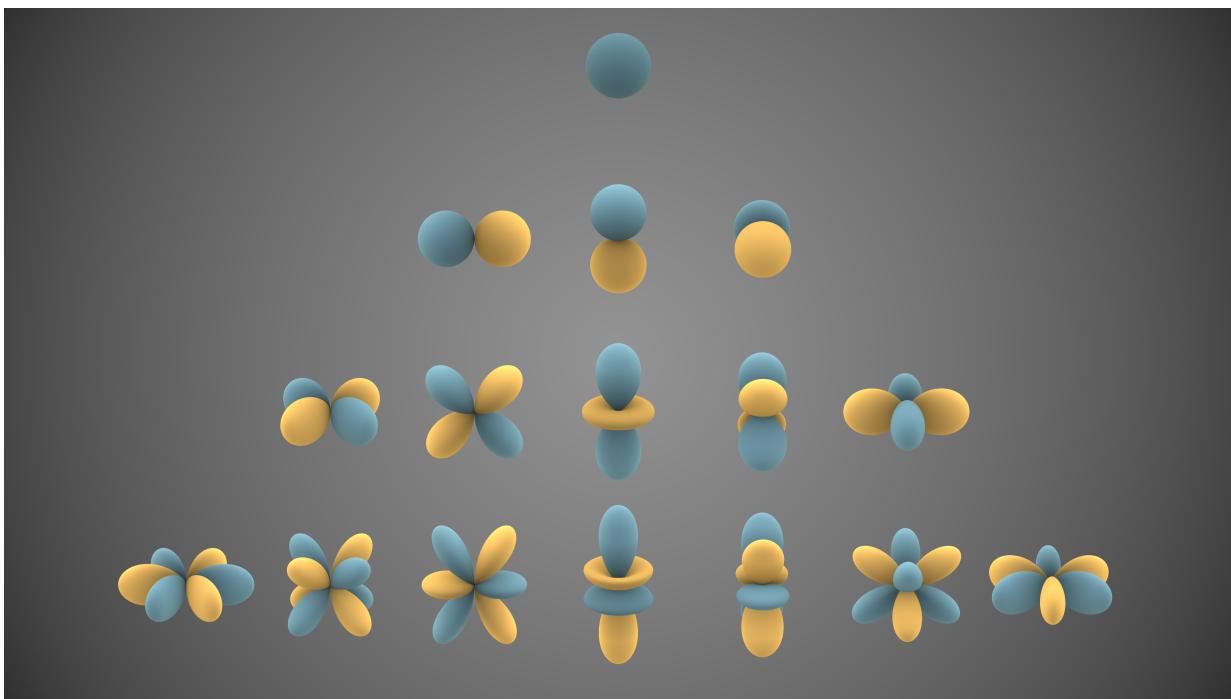
In dieser Form können die Kugelflächenfunktionen auch dazu benutzt werden um komplexe Verteilungen relativ kompakt zu definieren. So können beispielsweise komplexe Wahrscheinlichkeitsverteilungen als Summe von Kugelflächenfunktionen dargestellt werden. Jedes  $l \in \mathbb{N}$  wird dabei als ein Band bezeichnet und  $m$  durch  $l$  als das Intervall der ganzen Zahlen von  $[-l, l]$  definiert. Dabei fällt auf, dass die ungeraden Bänder asymmetrische und die geraden symmetrische Verteilungen sind.

### 2.2 Orientation Distribution Functions

Orientation Distribution Functions, kurz ODFs, beziehen sich auf eine Summe von Kugelflächenfunktionen. In der Praxis sind diese durch die Angabe der einzelnen Koeffizienten der zugehörigen Bänder der Kugelflächenfunktionen eindeutig definiert.

ODFs werden zumeist im „diffusion magnetic resonance imaging“ (DMRI) verwendet um die Diffusion, oder auch den Fluss, des Wassers durch das Gewebe des Körpers (zumeist Gehirnregionen) zu beschreiben.

Bei der Konstruktion von ODFs in der Polarisationsmikroskopie (oder auch Polarized Light Imaging, PLI) werden diese auf unterschiedliche Art und Weise berechnet. Entweder werden



**Abbildung 2.1** Reelle Anteile der Kugelflächenfunktionen [ $I_{nlm}$ ], blaue Teile zeigen hier die positiven Anteile der Funktion, gelbe die negativen. Von oben nach unten sind hier  $l = 0$  bis  $l = 3$  und von links nach rechts  $m = -3$  bis  $m = 3$  abgebildet.

mehrere Messwerte zusammengenommen, um diese zu einem einzelnen ODF zusammenzufassen. Dies reduziert die Auflösung der Daten und führt zum Verlust von Informationen. Oder es werden ODFs auf Basis jedes einzelnen Messwerts berechnet werden. Die so ermittelten ODFs enthalten jedoch wiederum keine Informationen über die Umgebung, sondern nur über die dominante Ausrichtung der im ausgemessenen Volumen enthaltenen Faser. Die Messung durch PLI liefert symmetrische Messwerte. Daher sind die geraden Bänder der Kugelflächenfunktionen ausreichend, um die ODFs dieser abzubilden. Es resultieren somit auch symmetrische ODFs.

Verwendet man beispielsweise ungerade Bänder, so ist das resultierende ODF nicht symmetrisch und führt zu einer falschen Abbildung der Messwerte, da nicht zutreffende Informationen angenommen werden (Asymmetrie). Die Wahrscheinlichkeit für eine bestimmte Richtung im ODF wird durch Abstand zum Ursprung angegeben. Dieser Abstand wird fortlaufend als Amplitude bezeichnet. Für weitere Infos über die Konstruktion von ODFs auf Basis von PLI-Daten siehe [Mar16].

## 2.3 Asymmetric Orientation Distribution Functions

Asymmetrische ODFs, kurz AODFs werden, im Gegensatz zu den ODFs, bei denen ausschließlich gerade Ordnungen berücksichtigt werden, aus allen Ordnungen der Kugelflächenfunktionen berechnet.

Dies erfordert jedoch asymmetrische Messwerte.

Somit ist zunächst die Aufgabe, diese Informationen aus den ODFs abzuleiten. Für dies wird die Umgebung, mithilfe einer auf Kegel basierten Methode, zusammengefasst.

## 2.4 Definition des mathematischen und abgerundeten Kegels

Für die Konstruktion der Kegel sind die Definitionen für den einfachen mathematischen und des abgerundeten Kegels relevant.

Die Bedingung für  $\vec{x} \in \mathbb{R}^3$  die innerhalb eines Kegel sind, mit der x-Achse als Symmetriearchse, ist gegeben durch:

$$x_2^2 + x_3^2 < x_1^2 \quad (2.2)$$

$$x_1 \geq 0 \quad (2.3)$$

$$x_1 < l \quad (2.4)$$

Gleichung 2.4 kann angepasst werden, sodass der Kegel durch eine Kugeloberfläche mit Radius  $l$  abschließt. Diese Art von Kegel wird nachfolgend als abgerundeter Kegel bezeichnet. Hierzu mehr in Unterunterabschnitt 4.1.2.2.

$$x_1^2 + x_2^2 + x_3^2 < l^2 \quad (2.5)$$

Für die Rotation wird der Kegel um die x- und y-Achse gedreht:

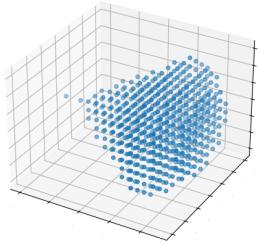
$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}, \quad R_y(\beta) = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \quad (2.6)$$

Fortfolgend wird der Winkel  $\alpha$  die Rotation um die x-Achse und der Winkel  $\beta$  die Rotation um die y-Achse beschreiben. Die Richtung des Einheitsvektors in x-Richtung nach der Rotation um zuerst die y- und anschließend die x-Achse wird im folgenden Verlauf als „Blickwinkel“ bezeichnet.

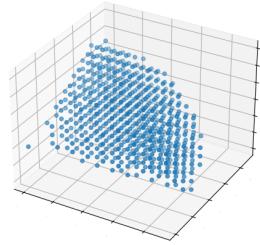
# 3 Entwurf

**Tabelle 3.1** Visualisierung der einzelnen Schritte, des Sammelns von Informationen für die Konstruktion von AODFs

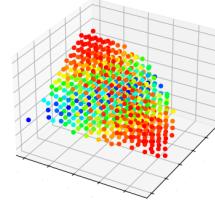
a Kegel erstellen



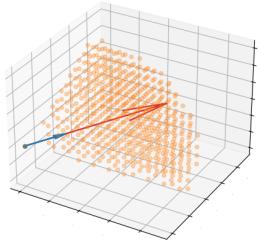
b Kegel rotieren



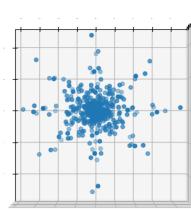
c Punkte im Kegel gewichten



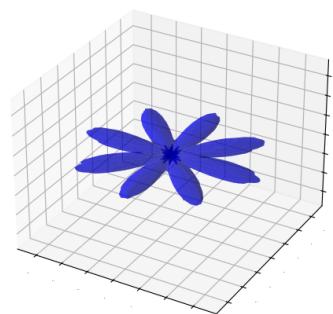
d ODFs auswerten und aufsummieren



e Prozess wiederholen für verschiedene Richtungen



f AODF erstellen



Mithilfe von rotierten Kegeln soll die Umgebung der einzelnen symmetrischen ODFs untersucht werden. Dabei entscheiden die ODFs innerhalb des Kegels über die Amplitude des entstehenden AODFs. Für jedes ODF werden  $n$  zufällige Winkel,  $\alpha$  und  $\beta$ , gewählt. Diese Winkel ergeben Blickrichtungen ausgehend vom ODF im Ursprung.

Für jeden dieser Blickwinkel wird ein Kegel mit einer Länge von  $l$  konstruiert. Alle ODFs innerhalb des Kegels werden anschließend in der jeweiligen Blickrichtung ausgewertet und abhängig von ihrer Position im Kegel gewichtet. Diese Gewichtung kann durch eine beliebige Funktion erfolgen, in dieser Thesis ist eine Gaußverteilung gewählt worden.

Nach der Gewichtung werden alle Punkte im Kegel aufsummiert. Diese Summe ergibt einen Punkt auf der Oberfläche des neuen asymmetrischen ODFs. Dieser Prozess wird hier als samplen bezeichnet. Um das AODF möglichst genau abzubilden, müssen viele Punkte auf diese Art und Weise gesampled werden.

## 3.1 Methoden zur Bestimmung der Punkte innerhalb rotierter Kegel

Zur Bestimmung der Punkte innerhalb eines rotierten Kegels gibt es 2 Möglichkeiten: Die Rotation des Kegels oder die Rotation des Koordinatensystems um den Kegel.

### 3.1.1 Rotation des Kegels

Für diese Methode werden zunächst die Punkte innerhalb eines Kegels im Ursprung, parallel zur x-Achse, bestimmt. Anschließend wird dieser um die zufällig generierten Winkel gedreht.

$$\vec{x} \in \mathbb{N}^3 \xrightarrow{\text{Rotation}} \tilde{\vec{x}} \in \mathbb{R}^3 : \tilde{\vec{x}} = R_x(\alpha)R_y(\beta)\vec{x} \quad (3.1)$$

Die einzelnen Punkte im Kegel müssen nicht für jeden Blickwinkel neu berechnet werden. Damit ist die einzige Rechnung pro Blickwinkel die Rotation eines einmalig im voraus berechneten Kegels. Jedoch liefert diese Methode  $\tilde{\vec{x}} \in \mathbb{R}^3$ . Jedes  $\tilde{\vec{x}}$  soll ein ODF an den Koordinaten von  $\vec{x}$  beschreiben. Da die ODFs ausschließlich an Koordinaten mit natürlichen Zahlen zu finden sind, muss  $\tilde{\vec{x}} \in \mathbb{N}^3$  gelten. Somit werden hier die Punkte gerundet.

### 3.1.2 Rotation des Koordinatensystems

Im Vergleich zu Gleichung 3.1 werden nun direkt die Punkte innerhalb des rotierten Kegels bestimmt. Um dies zu erreichen wird für jeden Winkel das Koordinatensystem und somit auch die Kegelbedingungen rotiert. Dafür werden die Bedingungen für den Kegel aus Gleichung 2.2 - 2.4 mit den rotierten Koordinaten  $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$  umgeschrieben.

Die Rotation erfolgt hier in umgekehrter Reihenfolge und mit umgedrehten Vorzeichen der Winkel, da die Koordinaten und nicht die Kegel rotiert werden.

$$\begin{aligned} \tilde{\vec{x}}(\alpha, \beta) &= \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{pmatrix} = R_y(-\beta)R_x(-\alpha)\vec{x} = \\ \alpha \rightarrow -\alpha, \quad \beta \rightarrow -\beta & \begin{pmatrix} x_3 \cos -\alpha \sin -\beta + x_2 \sin -\alpha \sin -\beta + x_1 \cos -\beta \\ x_2 \cos -\alpha - x_3 \sin -\alpha \\ x_3 \cos -\alpha \cos -\beta - x_1 \sin -\beta + x_2 \sin -\alpha \cos -\beta \end{pmatrix} \end{aligned} \quad (3.2)$$

Daraus ergeben sich die neuen Gleichungen für einen gedrehten Kegel aus Gleichung 2.2-2.4:

$$\begin{aligned} \vec{x} &\xrightarrow{\text{Rotation}} \tilde{\vec{x}} \\ &(x_2 \cos -\alpha - x_3 \sin -\alpha)^2 + \\ &(x_3 \cos -\alpha \cos -\beta - x_1 \sin -\beta + x_2 \sin -\alpha \cos -\beta)^2 \\ &< (x_3 \cos -\alpha \sin -\beta + x_2 \sin -\alpha \sin -\beta + x_1 \cos -\beta)^2 \end{aligned} \quad (3.3)$$

$$x_3 \cos -\alpha \sin -\beta + x_2 \sin -\alpha \sin -\beta + x_1 \cos -\beta \geq 0 \quad (3.4)$$

$$x_3 \cos -\alpha \sin -\beta + x_2 \sin -\alpha \sin -\beta + x_1 \cos -\beta < l . \quad (3.5)$$

Für alle  $x \in \mathbb{N}$ , die Gleichung 3.3-3.5 erfüllen, sind innerhalb des gedrehten Kegels. Da die Gleichungen abhängig von den Winkeln  $\alpha$  und  $\beta$  sind, ergeben sich unterschiedliche Bedingungen für jeden Kegel. Dies resultiert in einer längeren Bearbeitungszeit für die Berechnung der einzelnen Kegel. Die Ergebnisse sind hier  $\vec{x} \in \mathbb{N}^3$ . Es sind keine weiteren Anpassungen notwendig.

### 3.1.3 Vergleich der Rotationsmethoden

Die erste Methode (Gleichung 3.1) hat einen geringeren Rechenaufwand, jedoch müssen die Punkte gerundet werden, welches zu weiteren Herausforderungen führt. Das Runden führt zu Duplikaten in der Ergebnismenge und somit auch zu unvollständig besetzten Kegeln. Besonders für kleine  $l$  führt dies zu Verteilungen, die man nicht als Kegel identifizieren kann. Weiterhin führt dies zu unübersichtlichen und nur schwer nachvollziehbaren Ergebnissen, da nicht unbedingt alle ODFs einer Blickrichtung auch in die Berechnung der AODF eingehen. Die zweite Methode (Unterabschnitt 3.1.2) ist deutlich langsamer, liefert jedoch genau die Punkte innerhalb des definierten Kegels, ohne das Problem der doppelten Punkte oder Löcher im Kegel.

Da die relativen Positionen der ODF, unabhängig von dem Ort des ODFs im Ursprung, im Kegel identisch sind, kann ein Cache mit Kegeln in verschiedenen Blickrichtungen erstellt werden. Aus diesem Cache können anschließend die Kegel ausgelesen werden, wodurch das Skript beschleunigt wird.

## 3.2 Konstruktion der AODF Oberfläche

Wie zuvor beschrieben, können AODFs nur mit asymmetrischen Messwerten konstruiert werden. Das Kegelmodell ermöglicht, diese Informationen auf Basis der Umgebung der ODFs abzuleiten. So wird für jeden Kegel die Amplitude der enthaltenen ODFs in Richtung der Symmetrieachse ausgewertet. Da die Berechnung wie bei der Konstruktion der Kegel zeitaufwändig ist, wird auch hier ein Cache auf gleiche Art und Weise wie in Unterabschnitt 4.1.1 verwendet.

Diese Auswertung liefert für jeden Kegel eine Summe an Amplituden. Diese kann abhängig von der Position der ODFs im Kegel gewichtet werden. Die Funktion, mit der die Gewichtung bestimmt wird, kann frei gewählt werden und hat eine große Auswirkung auf die resultierenden AODFs. Dazu mehr in Kapitel 4.

Mithilfe der gewichteten Summe kann somit eine Aussage über die Amplitude des AODFs in Richtung der Kegel getroffen werden. Eine große Summe sorgt dafür, dass die dazugehörige Blickrichtung stärker in der Berechnung des AODFs eingeht.

## 3.3 Gewichte

Die Gewichte beziehen sich auf die gewichtete Summe der ODFs im Kegel. Diese Gewichtung wird durch eine frei wählbare Funktion, abhängig von den kartesischen Koordinaten im nicht rotierten und nicht translatierten Raum, gegeben. Das Ergebnis dieser Summe liefert die Amplitude des AODFs in der jeweiligen Blickrichtung.

---

## 4 Realisierung

Das Skript ist in Python geschrieben, da durch die weite Verbreitung in der Wissenschaft viele nützliche Open-Source-Ressourcen zur Verfügung stehen.  
Dazu werden die folgenden Bibliotheken verwendet:

- NumPy (Version 1.26.0)
  - Liefert vielseitige und gut optimierte mehrdimensionale Array-Operationen
- matplotlib (Version 3.8.0)
  - Bietet viele Möglichkeiten zur graphischen Darstellung von mathematischen Funktionen und Verteilungen
- fastpli [Fel]
  - Stellt die Grundlage der Berechnung der ODFs und AODFs auf Basis von PLI-Daten dar, darüber hinaus ermöglicht es die Simulation von künstlichen Nervenfasern. Die Generierten Bilder der ODFs und AODFs werden ebenfalls mithilfe dieser Bibliothek erstellt.

Die Konstruktion der AODFs ist wie in Kapitel 3 erläutert in 5 Teile unterteilt.

1. Erstellung der Kegel
2. Rotation der Kegel
3. Gewichtung der ODFs in den Kegeln
4. Zusammenfassen der Kegel zu Amplituden
5. Konstruktion der AODFs

## 4.1 Erstellung der Kegel

Für die Konstruktion der Kegel wird die Bibliothek NumPy verwendet (kurz np). Diese bietet sehr gut optimierte Array-basierte Operationen. Fast alle Daten für das gesamte Skript werden in den sogenannten NumPy-Arrays gespeichert.

Das Generieren der Punkte innerhalb eines Kegels kann mithilfe von `np.meshgrid()` und der Verwendung von Masken realisiert werden. Mit `X = np.array([1,2,3])` erzeugt `np.meshgrid(X,X,X)` einen 3x3x3-Würfel mit allen Kombinationen von  $x$ -,  $y$ - und  $z$ -Werten. Mithilfe von NumPy kann eine Maske diesen Würfel auf einen Kegel „zuschneiden“. Dafür werden an die  $x$ -,  $y$ - und  $z$ -Koordinaten des Würfels die Bedingungen für einen Kegel gestellt (siehe Gleichung 2.2-2.4). Alle Koordinaten, die nicht diesen Bedingungen entsprechen, werden durch die Maske aus der Liste genommen.

```
x,y,z = np.meshgrid(.)
maske = (
    (y**2+z**2 < x**2) &
    (x >= 0) &
    (x < 1)
)
Kegel_Punkte = np.array([x[maske], y[maske], z[maske]])
```

Das Resultat sind Punkte innerhalb eines nicht rotierten Kegels im Ursprung des Koordinatensystems.

Die Translation der Punkte im Kegels um  $\vec{x}_0$  ist durch eine einfache Transformation der Koordinaten gegeben:

$$\vec{x} \xrightarrow{\text{Translation}} \begin{pmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \end{pmatrix} \quad (4.1)$$

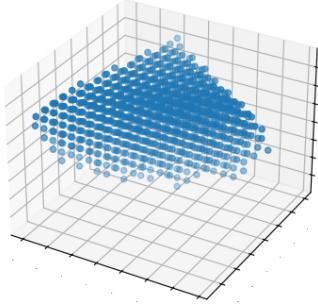
Beim Programmieren kann diese Operation mit

```
Kegel[0,:] += x_0
Kegel[1,:] += y_0
Kegel[2,:] += z_0
```

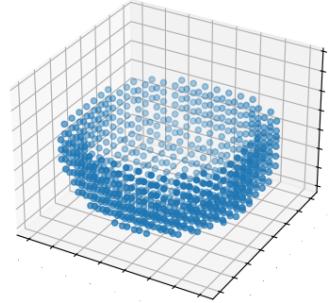
realisiert werden. Dabei ist `Kegel` ein zweidimensionales Array, wobei `Kegel[0,:]` die  $x$ -, `Kegel[1,:]` die  $y$ - und `Kegel[2,:]` die  $z$ -Koordinaten der Punkte im Kegel bezeichnet. Die Rotation des Kegels ist etwas komplizierter. Dabei wird aus den schon genannten Gründen (Unterabschnitt 3.1.3) das Koordinatensystem anstatt des Kegels gedreht. Dies gelingt, indem die Bedingungen der Maske rotiert werden, siehe Gleichung 3.3-3.5.

```
mask = (
    (y_rotiert**2+z_rotiert**2 < x_rotiert**2) &
    (x_rotiert >= 0) &
    (x_rotiert < 1)
)
```

Abbildung 4.1 ist ein Beispiel für einen rotierten Kegel. Dabei ist es wichtig, dass die Rotation vor der Translation stattfindet. Wird dies nicht berücksichtigt, wird der rotierte



**Abbildung 4.1** Rotierter Kegel mit  $l = 10$ , rotiert um  $\alpha \simeq 300, \beta \simeq 70$ , für alle sichtbaren Punkte  $\vec{x}$  gilt  $\vec{x} \in \mathbb{Z}^3$



**Abbildung 4.2** Halbierte Kugelschale mit dem Radius von  $F = 7$ , zeigt die Hälfte der Punkte die für den Aufbau des Caches mit dem zugehörigen  $F$  verwendet werden. Für alle sichtbaren Punkte  $\vec{x}$  gilt  $\vec{x} \in \mathbb{Z}^3$

Kegel, da um den Ursprung des Koordinatensystems rotiert wird, ungewollt verschoben.

### 4.1.1 Cache

Ein Cache bezeichnet einen Zwischenspeicher von Daten. Hier kann dieser dafür genutzt werden, dass Daten nicht mehrfach berechnet werden müssen. Da der Prozess der Berechnung des rotierten Kegels zeitlich intensiv ist, wird ein Cache mit rotierten Kegeln im Ursprung erstellt. Diese müssen dann nur aus den Cache herausgelesen und an die korrekten Koordinaten translatiert werden. Um einen endlich großen Cache zu realisieren, muss hinsichtlich der Genauigkeit ein Kompromiss eingegangen werden.

Der Cache wird konstruiert, indem zunächst die Punkte auf einer Kugelschale mit dem Radius  $F$  ermittelt werden. Größere  $F$  führen zu höherer Genauigkeit, erhöhen jedoch exponentiell die Anzahl der Punkte im Cache.

In der Kugelschale sind nur die  $\vec{x} \in \mathbb{Z}^3$  relevant. Es wird dazu wie zuvor `np.meshgrid()` verwendet und mithilfe einer Maske eine Kugelschale mit einer Schalendicke von 2 zu bestimmen.

```
x,y,z = np.meshgrid(·)
mask = (
    (x**2+y**2+z**2 < (F+1)**2)&
    (x**2+y**2+z**2 > (F-1)**2)&
)
```

Die dadurch erhaltenen Punkte (Abbildung 4.2) werden in die Rotationswinkel  $\alpha$  und  $\beta$

umgerechnet.

```
alpha = np.arctan2(z,y)+np.pi/2
beta = np.arccos(x/F)
```

Die Winkel der Kugelkoordinaten werden verwendet, um den Punkten entsprechend rotierte Kegel zuzuweisen.

#### 4.1.1.1 Auslesen aus dem Cache

Um den Kegel an einem beliebigen Punkt  $\vec{r}$  in eine Blickrichtung mit den Winkeln  $\beta_0$  und  $\alpha_0$  zu erhalten, wird der aus den Winkeln berechnete Einheitsvektor in Richtung der Symmetrieachse des Kegels berechnet. Da der Kegel im Ursprung (nicht gedreht) parallel zur x-Achse angenommen wird, ergibt sich der Einheitsvektor aus

$$\vec{e}_{\beta_0, \alpha_0} = |R_x(\beta_0)R_y(\alpha_0)|_{x=1, y=z=0} = \begin{pmatrix} \cos \alpha_0 \\ \sin \beta_0 \sin \alpha_0 \\ -\cos \beta_0 \sin \alpha_0 \end{pmatrix}. \quad (4.2)$$

Multipliziert mit dem Faktor  $F$ , ergibt sich der Punkt  $\vec{e}_{\beta_0, \alpha_0}F = \vec{C}_{\beta_0, \alpha_0, F}$ . Nach dem Runden der kartesischen Koordinaten können diese verwendet werden, um den zugehörigen Kegel im Cache auszulesen.

Der erhaltene Kegel hat seinen Ursprung bei  $(0, 0, 0)$  des Koordinatensystems. Um den Kegel an den Ort  $\vec{r}$  zu verschieben, muss dieser im Gesamten um  $\vec{r}$  verschoben werden.

$$\forall \vec{x}_0 \in \text{Kegel}_{\text{Ursprung}} \text{ gilt, } \vec{x}_0 + \vec{r} = \vec{x} \in \text{Kegel}_{\vec{r}}$$

Bei Python-Dictionaries kann mit `Dict[key]` der Eintrag des Dictionarys mit `key` als Schlüssel ausgegeben werden. Damit lässt sich das Auslesen der benötigten Kegel einfach realisieren:

```
x = np.cos(beta)
y = np.sin(alpha)*np.sin(beta)
z = -np.cos(alpha)*np.sin(beta)
koords = np.concatenate((x,y,z), axis=1)
koords_rounded_int = np.array(np.round(koords*factor, 0), dtype=int)
Kegel = [Kegel_cache[tuple(key)] for key in koords_rounded_int]
```

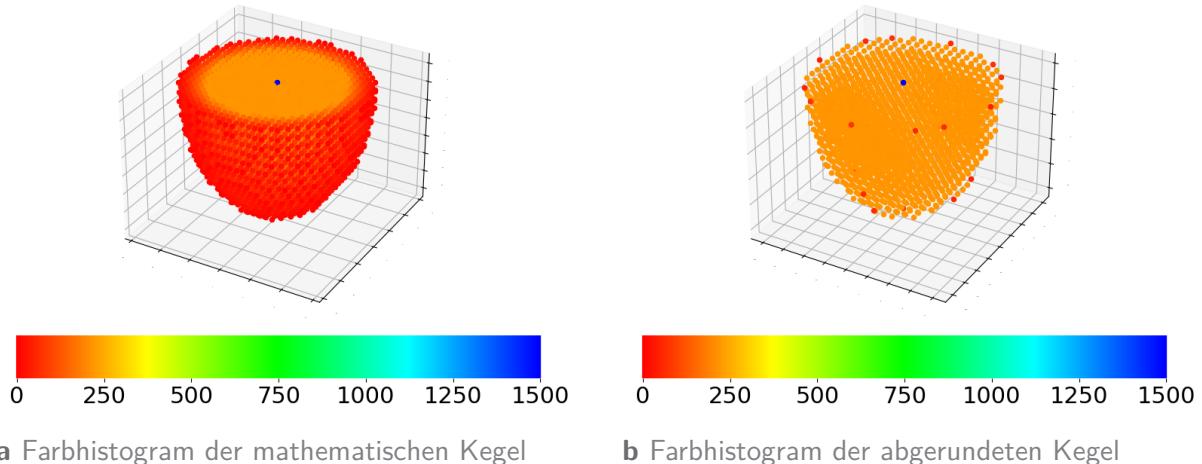
#### 4.1.2 Validierung

Es ist sehr wichtig, dass bei der Generierung der Kegel keine Fehler unterlaufen. Diese äußern sich in verformten AODFs und sind schwer auf die tatsächliche Ursache zurückzuführen. Daher ist es hilfreich, Kegel mit zufälligen Winkeln zu generieren und zu überprüfen, ob die Orientierung korrekt ist.

Wenn dies gelingt werden als nächstes für eine große Anzahl von Winkeln Kegel generiert. Anschließend wird gezählt wie oft die Punkte in den Kegeln enthalten sind, die Häufigkeit

wird mithilfe eines Farbgradienten dargestellt.

Dabei fallen so Punkte auf, die uncharakteristisch oft getroffen werden. Dies erleichtert die



**Abbildung 4.3** Durch Überlagerung von 1500 Kegeln in homogen verteilten Blickrichtungen entstehen kugelförmige Verteilungen. Die Farbe der Punkte stellt die Häufigkeit der Punkte in den Kegeln dar. Für eine bessere Darstellung wird die entstehende Kugel halbiert dargestellt.

Fehlersuche.

Wenn beispielsweise die Bedingungen der Kegel falsch gewählt werden, ist entweder die gesamte Verteilung nicht kugelsymmetrisch oder einzelne Zonen werden öfter als andere getroffen.

In Abbildung 4.3 fällt der zentrale Punkt der Kugel besonders auf. Dies bedeutet, dass das ODF im Zentrum zu jeder Blickrichtung beträgt. Dementsprechend sollte dies im Idealfall der einzige Punkt sein, der öfter getroffen wird als die anderen. Durch Gleichung 2.3/3.4 soll der Punkt im Ursprung durch  $\leq$  in jedem Kegel einbezogen werden. Da die Genauigkeit der Rechenoperationen endlich ist, reicht diese Bedingung nicht unbedingt aus. Wenn in den erzeugten Histogrammen auffällt, dass der zentrale Punkt nicht in allen Kegeln enthalten ist, sollte die Bedingung zu streng kleiner oder größer geändert und der zentrale Punkt separat in jedem Kegel eingefügt werden.

```
x,y,z = np.meshgrid(·)
mask = (
    (y**2+z**2 < x**2) &
    (x > 0) &
    (x < 1)
)
x_masked = np.append(x[mask], [0])
y_masked = np.append(y[mask], [0])
z_masked = np.append(z[mask], [0])
Kegel = np.array([x_masked, y_masked, z_masked])
```

Der  $(0,0,0)$ -Punkt bezieht sich auf das ODF, an dessen Position das AODF bestimmt

werden soll. Die Argumentation für das Hinzufügen des  $(0, 0, 0)$ -Punktes beruht auf der grundlegenden Aufgabenstellung. Demnach sollen die ODFs unter Berücksichtigung ihrer Umgebung zu AODFs angepasst werden. Durch das Hinzufügen geht die Form des zentralen ODFs in jeder gesampleten Blickrichtung ein und wird in der Berechnung des AODFs berücksichtigt. Somit entsteht ein AODF, das ein Abbild von dem originalen ODF und auch der Umgebung ist. Ohne den zentralen Punkt stellt die AODF nur die Umgebung dar. Falls dies der gewollte Effekt ist, sollte hier einfach das Hinzufügen des  $(0, 0, 0)$ -Punktes weggelassen werden.

### 4.1.2.1 Mathematischer Kegel

Bei Verwendung der Gleichung 3.3-3.5 für die Definition der Kegel, folgt eine Verteilung wie in Abbildung 4.3a. Dabei fällt auf, dass die äußeren Punkte der Kugel weniger getroffen werden, als die inneren. Dies folgt aus der Tatsache, dass die Punkte innerhalb des Kegels mit maximalen Abstand zum Ursprung einen Betrag von  $\frac{\sqrt{2}}{2}l$  haben. Diese können nur innerhalb des Kegels liegen, wenn dessen Symmetriearchse in einem  $45^\circ$ -Winkel zu diesen Punkten steht. Dies führt dazu, dass nicht alle Punkte die gleiche Wahrscheinlichkeit haben getroffen zu werden. Generiert man mithilfe der in Unterabschnitt 4.1.2 beschriebenen Methode ein Farbhistogramm, so ergibt sich die Abbildung 4.3a. Der klar sichtbare rote Rand stellt die geringere Wahrscheinlichkeit dar, mit der diese Punkte getroffen werden. In den Kegeln sind somit auch Punkte mit einem Radius zwischen  $l$  und  $\frac{\sqrt{2}}{2}l$  enthalten. Darüber hinaus sind diese Punkte ausschließlich abseits der Symmetriearchse. Dies führt zu einer zunehmenden Überrepräsentation von Punkten, je weiter abseits sich diese von Symmetriearchse befinden und somit zu einem Bias für diese Blickrichtungen.

### 4.1.2.2 Abgerundeter Kegel

Eine Möglichkeit die Verteilung homogener zu machen ist, die Definition der Kegel anzupassen. Dafür wird Gleichung 2.4 mit Gleichung 2.5 ersetzt. Das bedeutet, dass die Begrenzung des Kegels in die positive Richtung der x-Achse durch eine Kugeloberfläche mit dem Radius  $l$  gegeben ist. Dies führt dazu, dass alle Punkte innerhalb dieser Kugel mit Radius  $l$  mit gleicher Wahrscheinlichkeit getroffen werden können. Dies wird durch die homogene Farbe des Histogramms in Abbildung 4.3b sichtbar. Dabei fallen jedoch vereinzelte Punkte auf, die weniger oft in den Kegeln auftauchen (hier Rot). Diese Punkte befinden sich ausschließlich an der Oberfläche der Kugel.

Bei der Berechnung von beliebigen mathematischen Operationen, besonders der Multiplikation, treten durch endliche Nachkommastellen kleinste Fehler auf. Diese führen hier zu den Anomalien. Da die Anzahl dieser Anomalien im Vergleich mit der Gesamtzahl der Punkte verschwindend gering ist, führen sie zu keinen großen Fehlern. Diese Ungenauigkeiten sollten jedoch nicht gänzlich unbeachtet bleiben.

Aufgrund der gleichmäßigeren Abdeckung des Kugelvolumens durch den abgerundeten Kegel (Abbildung 4.3) ist das Sampeln eindeutiger. Da der mathematische Kegel Punkte

enthält, die einen größeren Abstand zum Ursprung des Kegels haben als die Länge des Kegels entlang der Symmetriearchse. Damit werden innerhalb eines Kegels unterschiedliche maximale Abstände zugelassen. Die daraus resultierenden AODFs gewichten ODFs abseits der Symmetriearchse höher da in diesen Blickrichtungen mehr ODFs innerhalb des Kegels liegen.

Aus diesem Grund wird für die folgenden Bilder von AODFs der abgerundete Kegel verwendet.

Darüber hinaus wird das ODF im Ursprung des Kegels wie in Unterabschnitt 4.1.2 erläutert in jeden Kegel aufgenommen.

### 4.1.3 Zufallswinkel

Um die Zufallswinkel für die Kegel zu generieren, muss darauf geachtet werden, dass die Pole der Kugel nicht unverhältnismäßig oft werden. Aus [Cor] folgt, dass die Anpassung:

$$\beta = \arccos(1 - 2Z) \quad \text{und} \tag{4.3}$$

$$\alpha = 2\pi Z \tag{4.4}$$

für die Generation der Zufallswinkel notwendig ist, um die korrekte homogene Gleichverteilung der Punkte auf der Kugeloberfläche zu erreichen. Hier ist  $Z$  eine Zufallszahl mit  $Z \in [0, 1)$ .

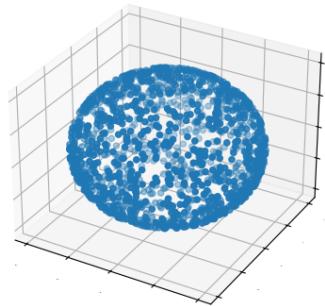
Die Zufallszahlen werden mithilfe von dem `numpy.random` Modul erzeugt.

```
rng = np.random.default_rng(random.randint(100000,1000000000))
beta = np.arccos(1-2*(rng.random(n).reshape(n,1)))
alpha = rng.random(n).reshape(n,1)*math.pi*2
```

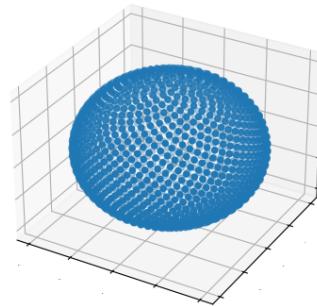
Dabei sind  $n$  die Anzahl der zufälligen Winkel. Diese sind für  $n \rightarrow \infty$  homogen auf der Einheitskugel verteilt. Daraus folgt, dass  $n$  ausreichend groß gewählt werden muss, damit die Verteilung homogen und somit die Ergebnisse reproduzierbar werden. Für kleine  $n$ , beispielsweise  $n = 1500$  (siehe Abbildung 4.4), sind die inhomogenen Bereiche der Kugel sehr offensichtlich.

### 4.1.4 Homogen verteilte Punkte auf einer Kugeloberfläche

Um die homogene Verteilung mit kleinen  $n$  zu realisieren, kann die Aufgabe rückwärts betrachtet werden. So wird versucht, direkt eine annähernd homogene Verteilung zu generieren, ohne die Notwendigkeit von großen  $n$ . Dafür wird nun auf eine Methode eingegangen, die den Goldenen Schnitt verwendet.



a Zufallswinkel-Methode mit  $n = 1500$



b Goldener-Schnitt-Methode mit  $n = 1500$

**Abbildung 4.4** Vergleich der generierten Punkte auf einer Einheitskugel durch Verwendung der Zufalls- und Goldener-Schnitt-Methode. Ziel ist eine möglichst homogene Verteilung der Punkte auf der Kugeloberfläche.

#### 4.1.4.1 Goldener Schnitt

Der Goldene Schnitt bezeichnet ein bestimmtes Teilungsverhältnis einer Strecke in zwei kleinere.

$$\text{Goldener Schnitt} = \Phi = \frac{1 + \sqrt{5}}{2} \quad (4.5)$$

Dieser kann auch mithilfe der Verhältnisse von aufeinander folgenden Fibonacci-Zahlen bestimmt werden. Auf diese Weise können gut genähert homogene Verteilungen von Punkten generiert und folgendermaßen auf die Oberfläche einer Kugel übertragen werden:

Der zu  $\beta = \arccos(1 - 2x) \mid x \in [0, 1]$  passende Winkel  $\alpha$  ist damit als

$$\alpha = 2\pi\Phi x = \pi(1 + \sqrt{5})x \quad (4.6)$$

gegeben. Eine mögliche Umsetzung in Python ist:

```
def Goldener_Schnitt_punkte(num_pts):
    indices = np.arange(0, num_pts, dtype=float) + 0.5
    beta = np.arccos(1 - 2*indices/num_pts)
    alpha = np.pi * (1 + 5**0.5) * indices
    return (alpha, beta)
```

Dieser Code ist einer Frage auf Stackoverflow [CR ] entnommen worden.

## 4.2 Gewichtung der ODFs in den Kegeln

Um zu vermeiden, dass die ODFs nur geglättet werden und die Details verschwinden, muss der Kegel durch die Gewichtung der enthaltenen Punkte „schmäler gemacht“ werden. Dafür wird eine einfache zweidimensionale Gaußverteilung verwendet.

```
def gauss_2d(y,z,mu_y=0,mu_z=0,sigma=2):
    return (1/(np.sqrt(math.pi*2)*sigma)
            *np.exp(-1/2*((y-mu_y)/sigma)**2+((z-mu_z)/sigma)**2))
    )
weights = gauss_2d(y,z,y_0,z_,sigma)
```

Diese soll den von der Symmetriearchse entfernten Punkten im Kegel geringere Gewichte zuordnen (für nicht rotierte Kegel im Ursprung ist die Symmetriearchse die x-Achse).

Um die Gaußfunktion verwenden zu können, müssen die Punkte des Kegels zurück rotiert werden. Dies erfordert zunächst die Rücktranslation an den Ursprung. Da die Gaußfunktion kontinuierlich ist, können die Punkte des rotierten Kegels durch simple Matrizenmultiplikation (4.1) rücktransformiert werden. Hierbei ist jedoch die Reihenfolge der Rotation und das Vorzeichen der Winkel umgedreht.

$$\tilde{\vec{x}} \in \mathbb{N}^3 \xrightarrow{\text{Rücktransformation}} \vec{x} \in \mathbb{R}^3 : \vec{x} = R_y(-\alpha)R_x(-\beta)\tilde{\vec{x}} \quad (4.7)$$

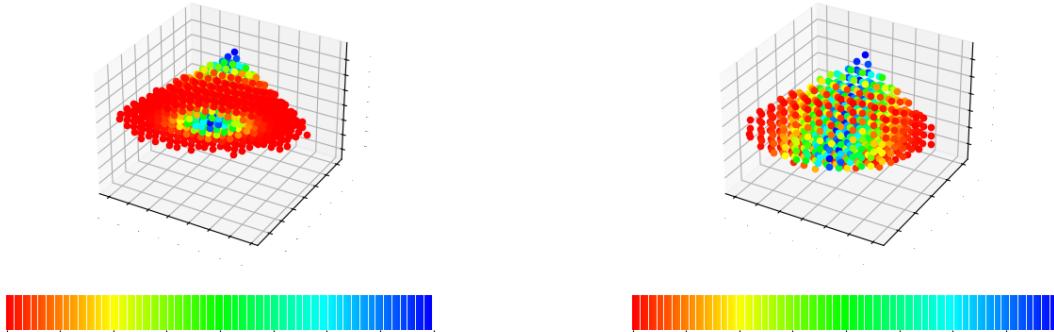
Die gewonnenen  $\vec{x}$  liefern eingesetzt in die Gaußfunktion die Gewichte der einzelnen Punkte. Als Skript lässt sich dies folgendermaßen realisieren:

```
Kegel[0,:] -= x_0
Kegel[1,:] -= y_0
Kegel[2,:] -= z_0
Kegel = np.matmul(rotation_matrix, Kegel)
```

`np.matmul(·)` ist hier die Matrixmultiplikation. Auch ist es sehr hilfreich bei einzelnen Kegeln die Gewichte durch einen Farbgradienten der Punkte zu visualisieren, um Fehler auszuschließen.

## 4.3 Zusammenfassen der Kegel zu Amplituden

Nachdem die Gewichte der einzelnen Punkte festgelegt sind müssen die Amplituden der ODFs an diesen Punkten in die Richtung der Symmetriearchse des jeweiligen Kegels ermittelt werden. Dazu wird die Funktion `odf3._analytic_single_odf(theta, phi, N_Bänder)` verwendet. Diese Funktion berechnet für die angegebene Richtung, abhängig von den Polar- und Azimutwinkeln  $\theta, \phi$  und die Anzahl der Bänder  $N_{Bänder}$  ein ODF mit diesen Ausprägungen.  $\theta$  und  $\phi$  sind dabei die Winkel der Kugelkoordinaten. Aus den Rotationswinkeln  $\alpha$  und  $\beta$  werden über die kartesischen Koordinaten die Winkel der Kugelkoordinaten bestimmt. Vergleicht man nun ein beliebiges ODF mit dem neu berechneten ODF, so kann der Anteil dieser Richtung an letzterem abgeleitet werden. In der Praxis ist dies eine Multiplikation der beiden Arrays:



**a** Rotierter mathematischer Kegel. Farblicher Verlauf der Gewichtung der Punkte, rot = geringes Gewicht, blau = hohes Gewicht

**b** Rotierter abgerundeter Kegel. Farblicher Verlauf der Gewichtung der Punkte, rot = geringes Gewicht, blau = hohes Gewicht

**Abbildung 4.5** Visualisierung der Gewichtung rotierter Kegel auf Basis der zweidimensionalen Gaußfunktion ( $\sigma = 2$ ), Vergleich mathematischer und abgerundeter Kegel

$$\frac{1}{N} \sum_i^N (\text{Ausprägungen}(\beta, \alpha, N_{\text{Bänder}}) \cdot \text{ODF} \cdot \text{Gewichte})_i = \frac{\text{Amplitude}(\beta, \alpha, N_{\text{Bänder}}) \text{ODF\_Array}}{N}$$

Hier stellt die Multiplikation ( $\cdot$ ) die komponentenweise Multiplikation der Arrays dar. Summiert man die Beiträge aller ODFs im Kegel (Anzahl  $N$ ) auf, liefert dies die Amplitude für die Richtung der Symmetriearchse des dazugehörigen Kegels. Das Teilen durch  $N$  ergibt sich daher, da die Anzahl der ODFs in jedem Kegel nicht identisch ist. Damit ist die Gleichgewichtung gewährleistet.

Diese Berechnung wird für die Anzahl der generierten Kegel wiederholt. Die Berechnung der Ausprägungen der ODFs ist wie ein Einheitsvektor im Kontext der ODFs zu verstehen. Daher muss dieser nicht für jedes ODF neu berechnet werden. Diese können somit im Voraus ausgerechnet und in einem weiteren Cache gespeichert werden. Wenn benötigt können diese schnell aus dem Cache ausgelesen werden. In der Umsetzung wird der Cache der Kegel dupliziert und die Ausprägungen der ODFs anstatt der Kegel pro kartesischer Koordinate abgespeichert (vgl. Unterabschnitt 4.1.1).

Mithilfe der Amplituden kann nun die AODF generiert werden. Der zugrunde liegende Code, der für die Berechnung der AODFs benutzt wird, verwendet alle angegebenen Blickrichtungen ungeachtet der Amplitude gleich. Somit müssen Blickrichtungen mit hoher Amplitude öfter in die Berechnung des AODFs eingehen als solche mit kleinerer Amplitude. Dies wird wieder mit der Skalierung der Amplitude realisiert.

Die Amplitude wird mit dem Faktor  $F_{AODF}$  multipliziert und auf eine ganze Zahl  $N$  gerundet.  $N$  gibt an, wie oft die dazugehörige Blickrichtung in das AODF eingeht. Dies realisiert eine Gewichtung für besonders ausgeprägte Amplituden und ignoriert sehr kleine, abhängig von dem gewählten Faktor. Durch das Runden auf  $N$  enthält das AODF keine Blickrichtungen

mit

$$\text{Amplitude} < \frac{1}{F_{AODF}} .$$

In Python wird die einzelne Amplitude eines Kegels durch

```
Amplitude_einzelter_Kegel = np.dot(basis, np.dot(
    weights, ODFs_im_Kegel
)
) / Anzahl_ODFs_im_Kegel
```

berechnet.

Dabei werden die ODFs innerhalb des Kegels durch eine Maske isoliert. In NumPy können Masken 2 verschiedene Typen annehmen. Entweder ist die Maske von der Dimension identisch mit dem zu maskierenden Array und ist gefüllt mit 0 und 1, je nachdem ob diese Positionen herausgefiltert werden sollen oder nicht. Oder die Maske hat eine kleinere Dimension, dann sind die Inhalte der Maske die Indizes der zu maskierenden Positionen im Array.

Letztere Methode wird hier verwendet. Die generierten Kegel beinhalten die kartesischen Koordinaten der ODFs, die in diesem enthalten sind. Diese sind identisch zu dem jeweiligen Index der ODFs. Somit kann ein Satz von Koordinaten innerhalb eines Kegels als Maske für die ODFs verwendet werden, um nur die ODFs innerhalb des Kegels zu erhalten.

Dabei muss auf die korrekte Form der Arrays geachtet werden. Diese kann mithilfe von `np.shape()` überprüft und mit `np.reshape()` oder `np.transpose()` angepasst werden. Dabei sollte stets geprüft werden ob die Transformation die ursprüngliche Sortierung der Daten beibehalten hat.

Die Liste der Winkel, abhängig von der Amplitude, wird durch

```
multiple_dir = np.repeat(phi_Amp_gößer_1,
    np.round((zugehörige_Amp) * F_AODF).astype(int))
multiple_inc = np.pi/2 - np.repeat(phi_Amp_gößer_1,
    np.round((zugehörige_Amp) * F_AODF).astype(int))
```

generiert.

## 4.4 Generieren der AODFs

Die mithilfe der Amplituden generierte Liste an Blickrichtungen wird an die Funktion `odf.compute()` gegeben. Die Funktion setzt jede angegebene  $\theta$  und  $\phi$  in die Bänder der Kugelflächenfunktionen ein. In den einzelnen Bändern werden die Ergebnisse unterschiedlicher Richtungen summiert. Die Summe liefert für das einzelne Band den Koeffizienten des resultierenden nicht normierten AODFs. Daher wird dies abschließend noch normiert. Das Ergebnis ist eine von der Anzahl der Samplingpunkte abhängige Repräsentation der Umgebung der ODFs. Dies entspricht dem gewollten Ergebnis.

Die alternative Methode ist, dass an die gesampelten Punkte eine AODF gefittet wird. Dies ist jedoch eine sehr zeitaufwendige Methode und das produzierte Ergebnis ist nicht nennenswert besser.

Für die Konstruktion der ODFs wird der selbige Prozess verwendet, wobei die Summe aus nur einer Richtung besteht. Verwendet man dabei unendlich Bänder entsteht ein unendlich dünnes ODF in der angegebenen Richtung. Da die Bänder jedoch endlich sind, ist das Resultat eine ausgedehnte Verteilung. Die Amplitude fällt somit bei einer Abweichung von der Symmetriearchse langsamer ab. Die Ausdehnung des ODFs ist nicht - wie bei dem Fit der ODFs an mehrere Richtungen - repräsentativ von der tatsächlichen Ungenauigkeit des ODFs, sondern ein pur mathematisches Phänomen der verwendeten Methode.

---

# 5 Analyse

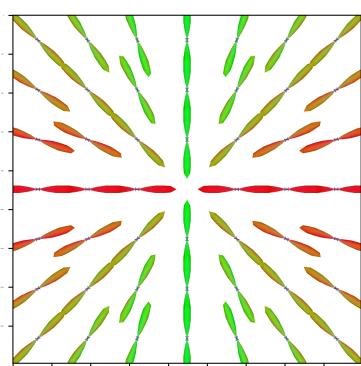
Nach der erfolgreichen Umsetzung, der Konstruktion von AODFs in Python, soll nun die Funktionsweise überprüft werden. Dafür werden zunächst die ausgewählten Methoden erläutert und anschließend die Ergebnisse des Skripts auf Basis von verschiedenen künstlichen ODF-Verteilungen gezeigt und diskutiert. Abschließend wird ein kleiner Teil einer PLI-Messung mit ODFs visualisiert und die berechneten AODFs ausgewertet.

Die Visualisierung basiert auf dem Samplen der Oberfläche der einzelnen AODFs. Die somit abgetastete Oberfläche wird in der x-y-Ebene liegend von oben betrachtet. Die Farben der AODFs geben dabei die Richtung dieser an. Daher sind dreidimensionale Formen durch die zweidimensionale Projektion nur durch den Farbverlauf erkennbar. Um Schwierigkeiten bei der Beurteilung der AODFs zu vermeiden, wird die Steigung (Auslenkung aus der xy-Ebene heraus) der ODFs für die künstlich generierten Verteilungen auf 0 gesetzt.

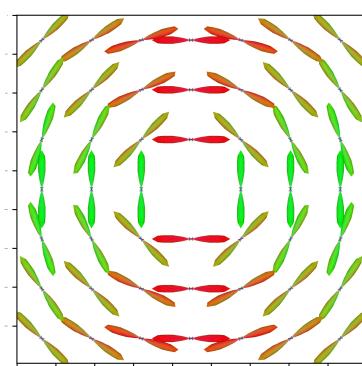
Darüber hinaus ist die Größe der AODFs nicht repräsentativ für die tatsächliche Dimension. Die Visualisierung soll lediglich dazu dienen die Form und Proportionen der AODFs zu beurteilen.

## 5.1 Resultate

Hier werden zunächst zwei ODF-Verteilungen konstruiert. Bei der ersten Verteilung werden die ODFs so gewählt, dass diese von einem zentralen Punkt weg zeigen, siehe Abbildung 5.1a. Die zweite Verteilung rotiert diese ODFs um 90°, sodass ein zirkuläres Muster (Abbildung 5.1b) entsteht. Die Idee hierbei ist, dass eventuell fehlerhafte Gewichtungen von bestimmten Blickrichtungen auffallen. Da die Komplexität dieser ODF-Verteilungen sehr überschaubar ist, kann das Ergebnis ebenfalls leichter auf Fehler untersucht werden.

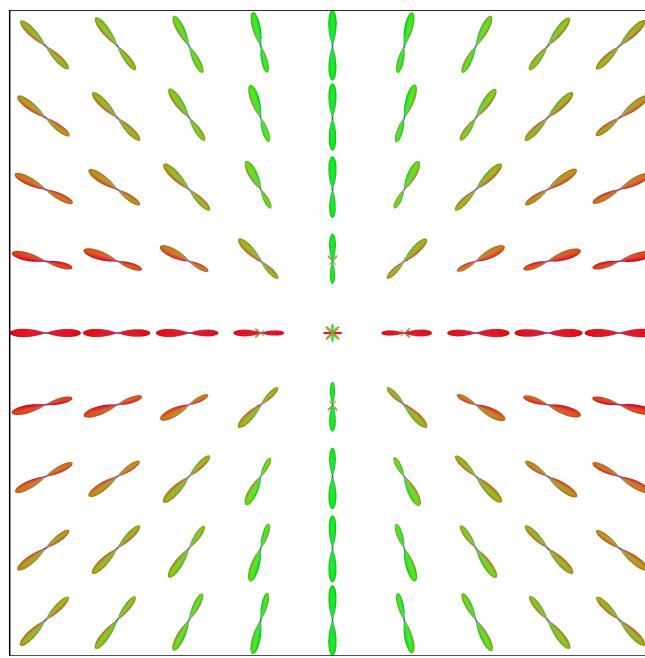


a ODFs der Stern-Verteilungen

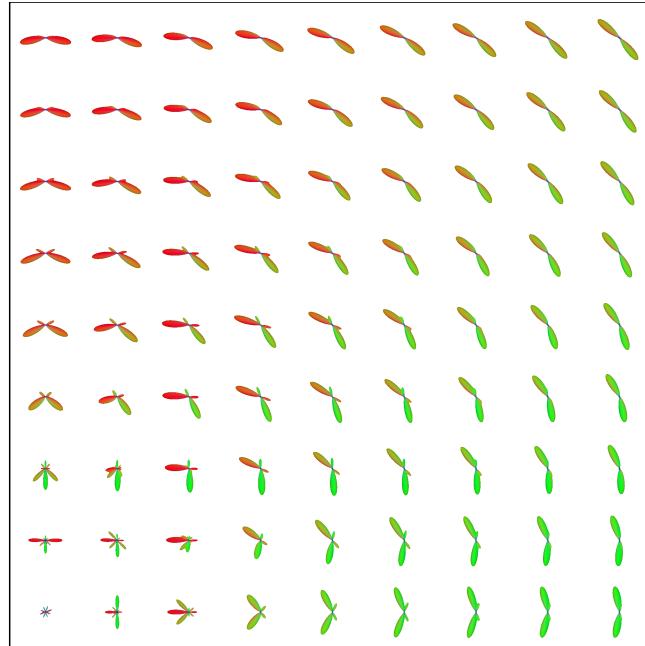


b ODFs der zirkulären Verteilungen

**Abbildung 5.1** ODFs der Stern- und zirkulären Verteilungen



**Abbildung 5.2** Große Stern-Verteilung, AODFs mit  $n = 1500$ ,  $l = 3$ ,  $\sigma = 0.3$ ,  $F_{AODF} = 100$

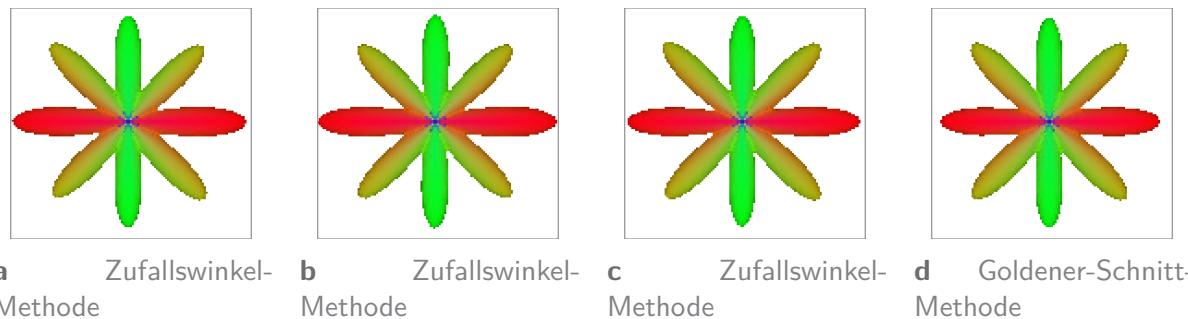


**Abbildung 5.3** Große zirkuläre Verteilung mit Zentrum unten links, AODFs mit  $n = 1500$ ,  $l = 4$ ,  $\sigma = 2$ ,  $F_{AODF} = 100$ , siehe Abbildung A.2, ODF-Basis

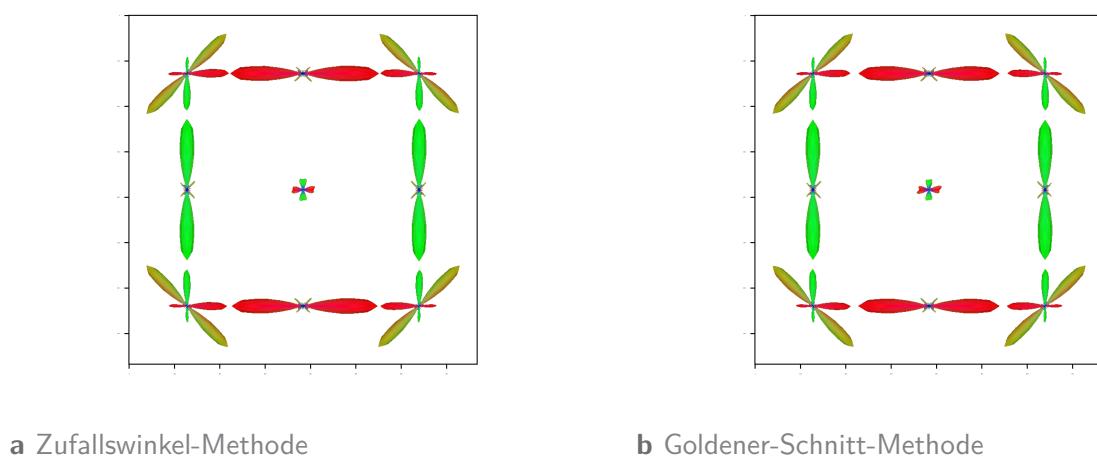
Da die ODF-Verteilung Abbildung 5.1b rotationssymmetrisch ist, wird in Abbildung A.3 nur ein Viertel der AODFs gezeigt. Die einzelnen AODFs können so bei einer größeren Distanz vom Ursprung in ausreichender Größe abgebildet werden.

Die AODFs in Abbildung 5.9 sind dieser AODF-Verteilung mit unterschiedlichen Parametern entnommen.

Abbildung 5.3 unterscheidet sich von dieser im Wesentlichen in der Größe von  $\sigma$ . Hier wird  $\sigma = 2$  gewählt.

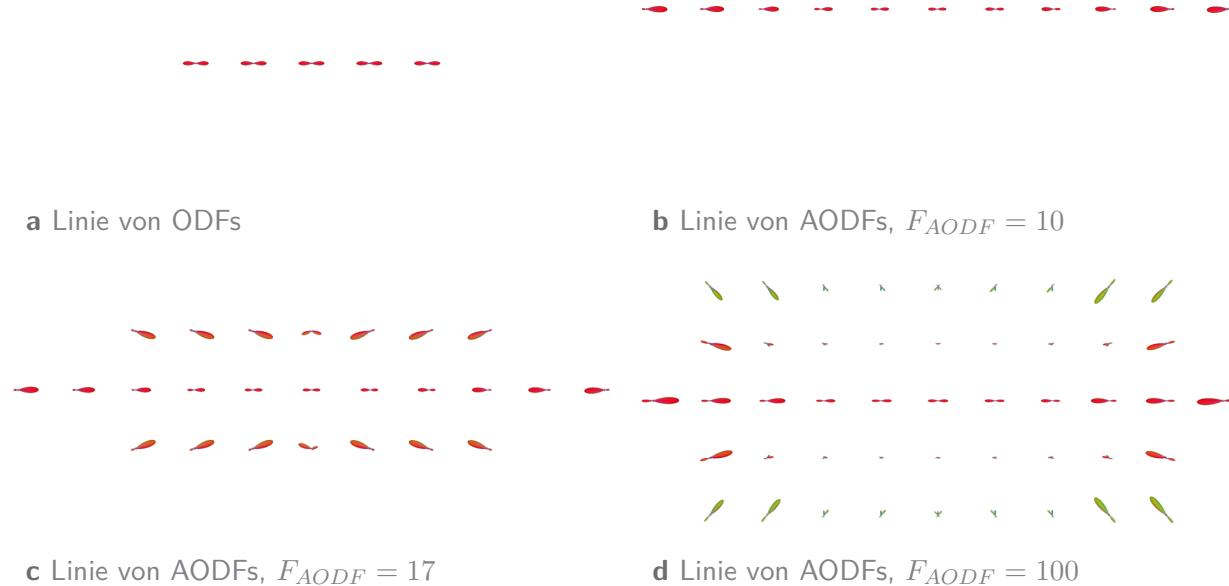


**Abbildung 5.4** Vergleich einzelner AODFs der Stern-Verteilung, generiert mit unterschiedlichen Sampling-Methoden, für die Zufallswinkel-Methode ist  $n = 200000$  und für die Goldener-Schnitt-Methode  $n = 1500$  gewählt



**Abbildung 5.5** Vergleich mehrerer AODFs der zirkulären Verteilung, generiert mit unterschiedlichen Sampling-Methoden, für die Zufallswinkel-Methode ist  $n = 200000$  und für die Goldener-Schnitt-Methode  $n = 1500$  gewählt

Die Bilder in den Abbildungen 5.4 und 5.5 stellen die Zufallswinkel- und Goldener-Schnitt-Methode gegenüber. Die verwendeten Parameter, außer  $n$ , sind dabei bei allen Bildern identisch.



**c** Linie von AODFs,  $F_{AODF} = 17$

**b** Linie von AODFs,  $F_{AODF} = 10$

**d** Linie von AODFs,  $F_{AODF} = 100$

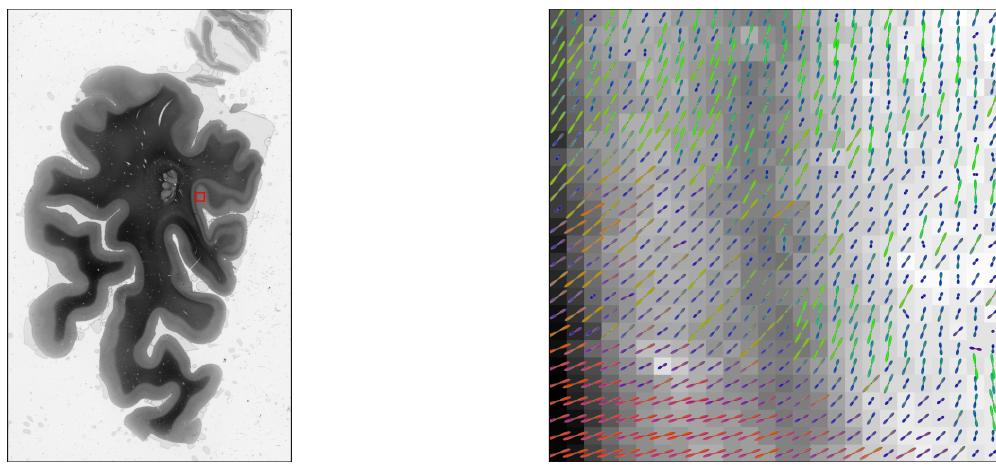
**Abbildung 5.6** Linien-Verteilung von ODFs und AODFs mit 7 Bändern,  $n = 1500, l = 3, \sigma = 0.3$  und verschiedenem  $F_{AODF}$

Anhand einer kurzen Kette von ODFs kann die Wirkung des Parameters  $F_{AODF}$  gut gezeigt werden. Durch auftretende Artefakte im Visualisierungs-Skript bei höherer Bandanzahl wurden die AODFs hier mit nur den ersten 7 Bändern der Kugelflächenfunktionen berechnet.

Die gewählten PLI-Daten sind die Schichten 300-309 der Abbildung A.4-A.12. Die gewählte Position ist arbiträr und hat keinen besonderen Grund. Die Anwendung des Skripts auf die PLI-Daten soll die Funktionalität über mehrere Ebenen zeigen. Dabei ist zu beachten, dass in den PLI-Daten die Ebenen eine gewisse Tiefe aufweisen. Diese Tiefe entspricht nicht der Voxel-Breite oder Tiefe. Da identische Einheiten für alle Koordinaten angenommen werden, sind die folgenden AODFs nicht korrekt.

Die Bilder können trotzdem zeigen, ob die AODFs die Informationen von darüber liegenden Ebenen einbinden.

Konkret soll hier die Wirkung der Veränderung von den verschiedenen Parametern  $\sigma$ ,  $n$ ,  $l$  und  $F_{AODF}$  auf die AODF gezeigt werden. Damit soll ein Einblick verschafft werden, wie genau die einzelnen Parameter für reelle Daten gewählt werden müssen.



**a** Transmittanz Gesamt

**b** Transmittanz Ausschnitt, mit ODFs im Vordergrund vgl. A.8

**Abbildung 5.7** Transmittanz der PLI-Daten, Ebene 304 und gewählter Ausschnitt

## 5.2 Diskussion

### 5.2.1 Zufällig und homogen verteilte Winkel

In Abbildung 4.4 ist gut zu erkennen, dass die Verteilung bei kleinen  $n$  deutlich homogener bei der Anwendung der Goldenen-Schnitt-Methode ist. Als Test, wie homogen die Zufallswinkel sind, werden mehrere AODFs mit gleichen  $n$  generiert. Unterschiede zwischen diesen weisen auf inhomogene Verteilung hin, unabhängig von den ODFs als Grundlage.

In Tabelle A.1 fällt auf, dass eine hohe Anzahl von Samples notwendig ist, um die Oberfläche des AODFs ausreichend homogen und genau zu beschreiben. Die notwendige Anzahl der Zufallswinkel beläuft sich auf ca. 100.000. Da diese für jedes AODF neu gesampled werden müssen, ergibt dies einen sehr hohen Zeitaufwand für die Berechnung. Im Gegenzug dazu liefert diese Methode jedoch AODFs ohne systematischen Bias.

Die alternative Goldene-Schnitt-Methode ist deutlich schneller, verwendet jedoch für alle AODFs die gleiche Verteilung der Winkel. Im Allgemeinen ist dies kein optimaler Ansatz, da die festgelegten Winkel einen Bias einbringen. Dieser führt zu systematischen Fehlern, die sich in der Übergewichtung dieser Winkel und einer geringeren Sensibilität für die nicht enthaltenen Blickrichtungen äußern. Jedoch wird dieser Fehler einerseits kleiner, je mehr Winkel aufgenommen werden. Anderseits ist durch die kontinuierliche Form der ODFs garantiert, dass auch wenn nicht exakt die richtige Blickrichtung gesampled wird, die Amplitude des ODFs trotzdem noch nicht vollständig abgefallen ist und somit ein Anteil noch erfasst werden kann.

Basierend auf diesen einander begünstigenden Überlegungen, wird die Methode nun daraufhin geprüft, ab welcher Anzahl der Winkel die AODFs hinreichend genau angenähert werden.

Um dies zu überprüfen werden zunächst von einem kleinen Datensatz von ODFs AODFs mithilfe einer großen Anzahl (hier  $n = 200.000$ ) von Zufallswinkeln generiert. Diese sollen als Referenz dienen.

Um die Varianz der Methode beurteilen zu können, werden die AODFs insgesamt dreimal mit gleichen Bedingungen erzeugt. Mit diesen AODFs kann nun das AODF der Goldenen-Schnitt-Methode verglichen werden.

Die zugrundeliegenden ODFs, auf deren Basis die AODFs generiert werden, sollten hierbei bedacht gewählt werden. Hier wird aufgrund der Überschaubarkeit, die Verteilung in Abbildung 5.1a gewählt. Aus dem Vergleich der AODFs in Abbildung 5.4 geht hervor, dass der Unterschied zwischen den 4 AODFs sehr gering ist. Die Goldene-Schnitt-Methode sticht im Vergleich nicht besonders heraus. Betrachtet man die zirkuläre Verteilung, so scheint dies dort ebenfalls bestätigt zu werden.

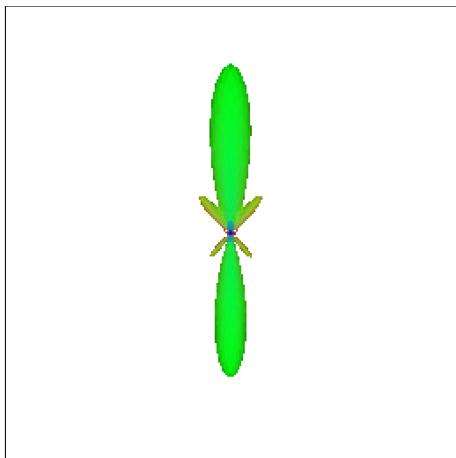
Basierend auf diesem Vergleich wird für die Berechnung der nachfolgenden AODFs die Goldene-Schnitt-Methode verwendet. Allgemein ist für die Berechnung einzelner AODFs oder bei kleinen Datenmengen die Zufallsmethode mit einer großen Anzahl von Winkeln eine bessere Wahl.

Nach der Klarstellung der verwendeten Elemente der vorgestellten Methoden werden AODFs von größeren Verteilungen erstellt, um diese einerseits auf Korrektheit zu prüfen. Aber auch um festzustellen, ob die AODFs die zugrundeliegenden Verläufe besser abbilden. Darüber hinaus wird die Wirkung der allgemeinen Parametern des Skripts diskutiert und visualisiert.

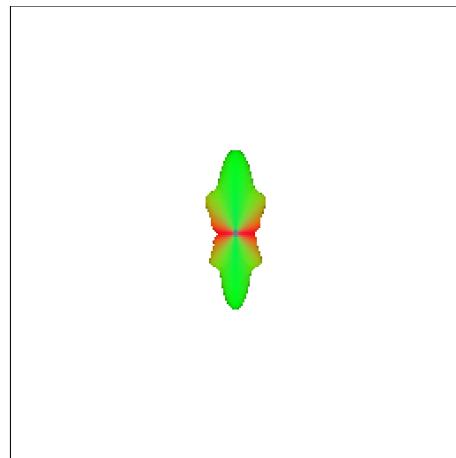
## 5.2.2 Große Stern-Verteilung

Bei der großen Stern-Verteilung (Abbildung 5.2) ist gut zu erkennen, dass die inneren AODFs sich stark von den ODFs unterscheiden. Die Form entspricht den Erwartungen, da im Inneren der Verteilung die größten Richtungsunterschiede auftreten. Dementsprechend ist die Form von weiter entfernten AODFs in Bereichen mit fast identischer Richtung den ODFs sehr ähnlich.

In dieser Verteilung ist die Auswirkung von  $l$  besonders gut erkennbar, in Abbildung 5.8



**a**  $l = 3$

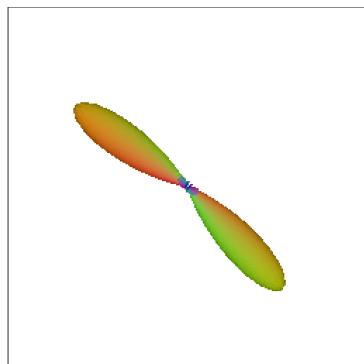


**b**  $l = 8$ , siehe Abbildung A.1, großes AODF Bild

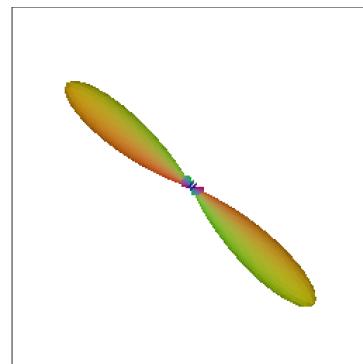
**Abbildung 5.8** Vergleich einzelner AODFs aus der großen Stern-Verteilung, die identischen Parameter sind:  $n = 1500$ ,  $\sigma = 0.3$ ,  $F_{AODF} = 100$

ist ein direkter Vergleich einzelner AODFs unter verändertem Parameter  $l$ . So ist in Abbildung 5.9b klar zu erkennen, dass die größere Kegellänge  $l = 8$  die Proportionen des AODFs stark verändert hat. Dies liegt hier daran, dass mehr ODFs verschiedener Richtungen in die AODF-Berechnung eingehen. Da diese weiter vom Ursprung entfernt sind, gehen diese bei einem kleineren  $l$  nicht ein und polarisieren somit die AODFs. Diese haben also eine sehr klar in einzelne große Amplituden unterteilte Struktur. Für große  $l$  führt dies zu einer geglätteten Oberfläche.

### 5.2.3 Große zirkuläre Verteilung



**a**  $\sigma = 2$



**b**  $\sigma = 0.3$ , siehe Abbildung A.3, großes AODF-Bild

**Abbildung 5.9** Vergleich einzelner AODFs aus der großen zirkulären Verteilung, die identischen Parameter sind:  $n = 1500$ ,  $l = 3$ ,  $F_{AODF} = 100$

Bei der Betrachtung der AODFs der Großen zirkulären Verteilung (Abbildung A.3) fällt auf, dass die AODFs mit größerem Abstand zum Mittelpunkt der Rotation eine klar sichtbare Krümmung aufweisen (Abbildung 5.9). Vergleicht man diese mit den AODFs mit  $\sigma = 0.3$  so ist nahezu keine Asymmetrie oder Krümmung zu erkennen, siehe Abbildung 5.9. Dies lässt darauf schließen, dass größere  $\sigma$  die Sensibilität gegenüber Kurven steigert. Da  $\sigma$  die Gewichtung der ODFs im Bezug zu ihrem Abstand zur Symmetriechse des Kegels beschreibt, ist die zuvor beschriebene Wirkung wie erwartet. Genauer, da die ODFs, die Teil einer Kurve sind, zwangsläufig abseits der Symmetriechse der gesammelten Blickrichtung sein müssen, kann  $\sigma$  somit als ein Maß der Kurven-Empfindlichkeit gesehen werden.

## 5.2.4 Linienförmige AODFs

Um den Parameter  $F_{AODF}$  besser zu verstehen, ist das Beispiel einer einzelnen ODF-Linie gut zur Illustration geeignet (Abbildung 5.6a). Dabei fällt auf, dass für größere Faktoren von  $F_{AODF}$  neue AODFs am Rand der Linie entstehen, da selbst kleine gesamme Amplituden der ODFs ausreichen, um AODFs zu erzeugen. Dies resultiert in der Generation von neuen AODFs in Positionen außerhalb der tatsächlichen ODF-Verteilung (Abbildung 5.6c und 5.6d).

Die Wirkung von  $F_{AODF}$  ist explizit eine Schranke für gesamme Amplituden. Nur ausreichend große Amplituden gehen so in die AODF-Berechnung ein.

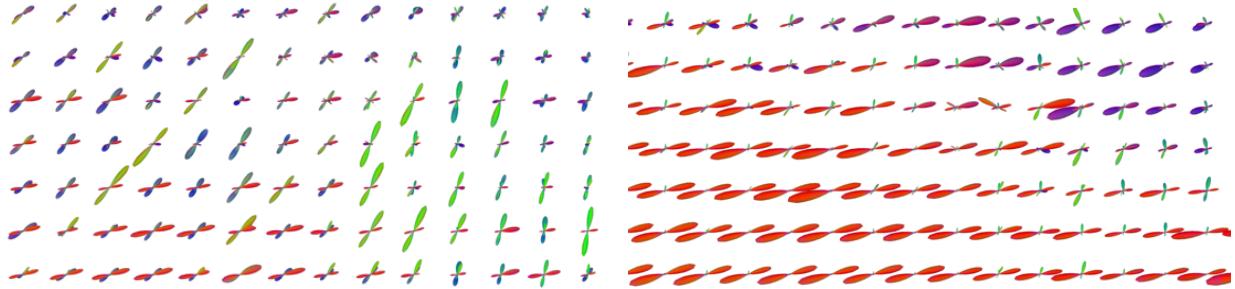
Im Zentrum der zirkulären Verteilung sollte theoretisch kein AODF generiert werden, da alle ODFs senkrecht zu diesem stehen. Es bildet sich jedoch trotzdem ein AODF. Dies liegt daran, dass die kontinuierliche Oberfläche der ODFs in alle Richtungen eine minimale Amplitude hat. Wird  $F_{AODF}$  zu groß gewählt, führt dies dazu, dass diese kleinen Amplituden alleinig ein neues AODF erzeugen. Diese Amplituden basieren nicht auf tatsächlich existierenden Messwerten, sondern sind ein Artefakt der ODFs.

Zusammenfassend sollte  $F_{AODF}$  nicht zu groß gewählt werden, da dies die Daten verfälschen kann. Diese Artefakte treten dazu, außerhalb von besonderen Verteilungen, nur in Randbereichen auf, da in anderen Fällen einzelne große Amplituden die vielen kleinen dominieren und das AODF insgesamt nur minimal von diesen beeinflusst wird.

Ergänzend dazu führen zu kleine  $F_{AODF}$  zu unvollständigen und fehlerhaften Abbildungen. Daher ist es wichtig die Randbereiche der AODFs auf diese Artefakte zu untersuchen, um die richtige Wahl für  $F_{AODF}$  zu bestätigen.

Unabhängig von  $F_{AODF}$  ist die Fortsetzung von Mustern (Abbildung 5.6b) oder auch Füllen von Lücken bei fehlenden ODFs immer ein Ergebnis dieser Methode (unabhängig von  $F_{AODF}$ ). Dies ist ein Resultat der normalen Funktionsweise des Skriptes und kann durch das Anwenden einer Maske, basierend auf den zugrundeliegenden ODFs, korrigiert werden. Solange eine Maske verwendet wird, um die Artefakte herauszufiltern, ist auch ein hohes  $F_{AODF}$  kein Problem. Dies erhöht dann jedoch den Aufwand bei der Berechnung der AODFs.

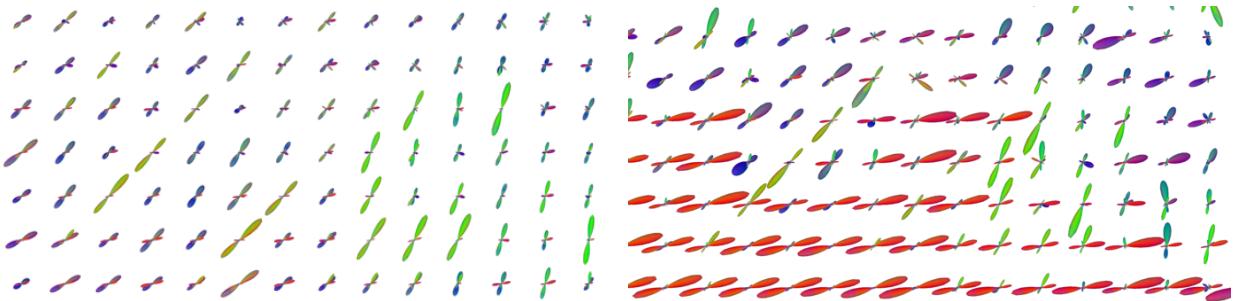
## 5.2.5 PLI-Daten



**a**  $\sigma = 0.5$ ,  $l = 3$ ,  $F_{AODF} = 10$ , siehe Abbildung A.13

**b**  $\sigma = 2$ ,  $l = 3$ ,  $F_{AODF} = 10$ , siehe Abbildung A.14

**Abbildung 5.10** Ausschnitte der AODFs der PLI-Daten, Ebene 304, mit verschiedenem  $\sigma$



**a**  $\sigma = 0.5$ ,  $l = 2$ ,  $F_{AODF} = 10$ , siehe Abbildung A.16

**b**  $\sigma = 0.5$ ,  $l = 4$ ,  $F_{AODF} = 10$ , siehe Abbildung A.15

**Abbildung 5.11** Ausschnitte von den AODFs der PLI-Daten, Ebene 304, mit verschiedenem  $l$

Vergleicht man Abbildung A.13 und A.14 speziell in dem Bereich aus 5.10, so fällt auf, dass die dünneren Stränge durch die hier orange Orientierung dominiert und überlagert wird. Dies sorgt für eine Art Glättung der Daten. Der Prozess kann mit der Wirkung eines Gauß-Kernels bei der Bildbearbeitung verglichen werden.

Somit sollte  $\sigma$  einerseits groß genug gewählt werden, dass Kurven erkannt werden, andererseits auch klein genug, damit die Details der Messwerte erhalten bleiben.

Ähnlich wie die Wirkung von großen  $\sigma$  führen große  $l$  zu einer Verringerung der Details. Die verwendete Gewichtungsfunktion betrachtet nur den Abstand zur Symmetrieachse. Somit ist die absolute Distanz zu den ODFs nicht relevant. Daraus folgt, dass feine Fasern in der Nähe durch breitere Fasern, trotz weiterer Entfernung, dominiert werden können. Dies verbessert potentiell das Zusammenführen von großen Fasern, führt jedoch auch dazu, dass kleinere Fasern stärker unterdrückt werden.

Die Verwendung von einer dreidimensionalen Gaußverteilung oder allgemein einer Gewichtungsfunktion, die auch den absoluten Abstand zum Ursprung des Kegels berücksichtigt, sollte dieses Problem umgehen können. So geht beispielsweise eine breite Faser mit größerem

Abstand im Idealfall mit geringerer Wertung in das AODF ein, sodass die Richtung der Faser nicht das AODF dominiert und die Information der kleinen Faser in direkter Nähe erhält.

Dies müsste jedoch zunächst genauer getestet werden.

Die Ausschnitte der PLI-Daten (Abbildung A.15-A.16) zeigen durch Vergleich mit den zugrundeliegenden ODFs (Abbildung A.4-A.12), dass die AODFs die verschiedenen Schichten erfolgreich einbinden, auch wenn dabei, wie zuvor erwähnt, die Dicke der Schichten nicht korrekt berücksichtigt wird.

---

# 6 Schlussbetrachtungen

## 6.1 Fazit

Es konnten auf Basis von ODFs erfolgreich AODFs konstruiert werden. Der zunächst postulierte Entwurf wurde dafür an mehreren Stellen angepasst. Die mathematischen Kegel wurden durch abgerundete Kegel ersetzt. Dies erzielt eine homogenere Gewichtung der ODFs und verringert den systematischen Bias.

Darüber hinaus wurden die Zufallswinkel durch homogen verteilte Winkel ersetzt. Die Zufallswinkel-Methode ist weiterhin für kleinere ODF-Verteilungen die bessere Methode, da wie zuvor hier ebenfalls ein systematischer Bias ausgeschlossen wird. Jedoch ist der Aufwand dieser Methode für größere Datensätze zu groß. Die Goldener-Schnitt-Methode liefert hier vergleichbare Ergebnisse für einen Bruchteil des Rechenaufwands.

Die Parameter des Skripts beeinflussen die berechneten AODFs stark.  $\sigma$  und  $l$  haben eine sehr ähnliche Wirkung auf die AODFs. Beide sorgen dafür, dass die AODFs homogener aussehen und Details verschwinden. Bei  $\sigma$  werden einzelne Fasern breiter,  $l$  sorgt hingegen dafür, dass diese verlängert werden. Beides kann andere kleinere Fasern überlagern und unkenntlich machen.  $F_{AODF}$  sorgt dafür, dass AODFs mit Anteilen von Richtungen berechnet werden, die unter Umständen nicht existieren sollten.

Insgesamt ist die Funktion des Skripts sehr zufriedenstellend, die Geschwindigkeit ist allerdings ein Schwachpunkt. Die gezeigten kleinen Umgebungen von AODFs werden problemlos innerhalb von Sekunden berechnet. Der Versuch, 6 vollständige Ebenen der PLI-Daten zu AODFs umzurechnen, hätte voraussichtlich 10 Stunden gedauert.

## 6.2 Ausblick

Es gibt trotz der bestätigten Funktion einige übrige Aufgaben. Die Geschwindigkeit des Skripts bei der Berechnung der AODFs ist für große Datenmengen nicht ausreichend. Da die Berechnung von AODFs unabhängig von den anderen ist, sollte der Prozess gut für Parallelisierung geeignet sein.

Darüber hinaus könnte über eine Version auf Basis von mit Numba kompatiblen Funktionen die allgemeine Laufzeit des Programms minimiert werden. Numba ist eine Python-Bibliothek, die einen JIT-Compiler verwendet, um den Code zu Maschinencode zu übersetzen. Da jedoch die verwendeten Numpy-Funktionen ebenfalls gut optimiert sind ist nicht klar ob dies zu einer tatsächlichen Verbesserung führt.

Wie in Unterabschnitt 5.2.5 erwähnt werden die Abstände der ODFs in alle Richtungen zu ihren Nachbarn als gleich groß angenommen. Bei der Erstellung der Maske für den Kegel könnten die verschiedenen Achsen mit einem Faktor multipliziert werden, der relativ zu den anderen den Abstand definiert.

An den Rändern der ODF-Verteilung kann es passieren, dass die Punkte innerhalb des Kegels außerhalb der Liste der ODFs liegen. Dies führt zu einem Absturz des Skripts. Um dies zu

verhindern können die Ränder der ODFs mit Null-ODFs um die Länge der Kegel  $l$  erweitert werden. Das Skript wird weiterhin nur auf die originalen ODFs angewandt. Alternativ wird das Skript nur auf die ODFs mit einem ausreichenden Abstand zum Rand angewendet. In der Praxis sollte dies kein großes Problem darstellen, da die meisten Datensätze Null-ODFs an den Rändern der ODF Verteilungen haben, die von der AODF-Berechnung ausgeschlossen werden können.

Um die Aussagekraft der AODFs für die Traktographie zu überprüfen, kann des Weiteren das FiberCup-Beispiel verwendet werden. Das FiberCup-Beispiel ist eine Zusammenstellung von verschiedenen anatomisch relevanten Faserverläufen.

Bei der Visualisierung der AODFs treten nicht eindeutig reproduzierbare Artefakte auf. Diese stammen sehr wahrscheinlich aus der Visualisierung. Die konnte jedoch nicht mit Gewissheit bestätigt werden. Eine mögliche Methode dies zu überprüfen ist die Visualisierung der AODFs durch alternative Programme.



---

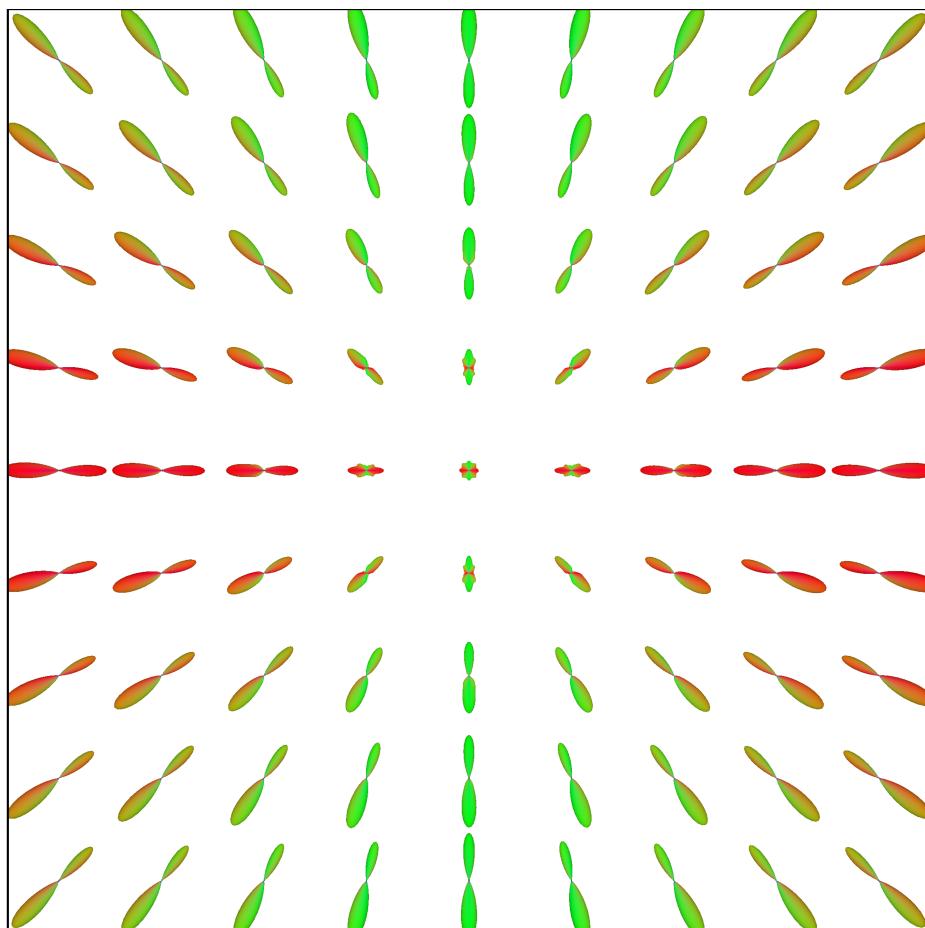
# Literatur

- [CR] CR Drost. *Evenly distributing n points on a sphere*. Webseite. Stackoverflow Frage. Abgerufen am 19.01.2024 um 11:26 Uhr. URL: <https://stackoverflow.com/questions/9600801/evenly-distributing-n-points-on-a-sphere>.
- [Cor] Cory, Simon. *Generating uniformly distributed numbers on a sphere*. Webseite. Private Webseite. Abgerufen am 42.03.1337 um 23:59 Uhr. URL: <https://corysimon.github.io/articles/uniformdistn-on-sphere/>.
- [Fel] Felix, Matuschke. *Fiber Architecture Simulation Toolbox for 3D-PLI*. Webseite. Github Repository. Abgerufen am 19.12.2023 um 12:18 Uhr. URL: <https://github.com/3d-pli/fastpli>.
- [Ini] Inigo.quilez CC BY-SA 3.0. *Spherical harmonics*. Webseite. Bild. Abgerufen am 11.12.2023 um 11:50 Uhr. URL: <https://commons.wikimedia.org/w/index.php?curid=32782753>.
- [Loh21] Lohmann, Simon. *Thesisvorlage der Fakultät für Elektrotechnik, Informationstechnik und Medientechnik*. Nov. 2021.
- [Mar22] Markus Aixer and Katrin Amunts. *Scale matters: The nested human connectome*. Webseite. Paper. abgerufen am 07.01.2024 um 22:03. Nov. 2022. URL: <https://www.science.org/doi/10.1126/science.abq2599>.
- [Mar11] Markus Aixer. *High-resolution fiber tract reconstruction in the human brain by means of three-dimensional polarized light imaging*. Webseite. Paper. abgerufen am 07.01.2024 um 22:03. Dez. 2011. URL: <https://www.frontiersin.org/articles/10.3389/fninf.2011.00034/full>.
- [Mar16] Markus Aixer. *Estimating Fiber Orientation Distribution Functions in 3D-Polarized Light Imaging*. Webseite. Paper. abgerufen am 08.01.2024 um 15:15. Apr. 2016. URL: <https://doi.org/10.3389/fnana.2016.00040>.
- [Suh18] Suheyla Cetin Karayumak. *Asymmetric Orientation Distribution Functions (AODFs) revealing intravoxel geometry in diffusion MRI*. ScienceDirect. Paper. Abgerufen am 10.10.2023 um 15:15 Uhr. 2018. URL: <https://sci-hub.hkvisa.net/10.1016/j.mri.2018.03.006>.

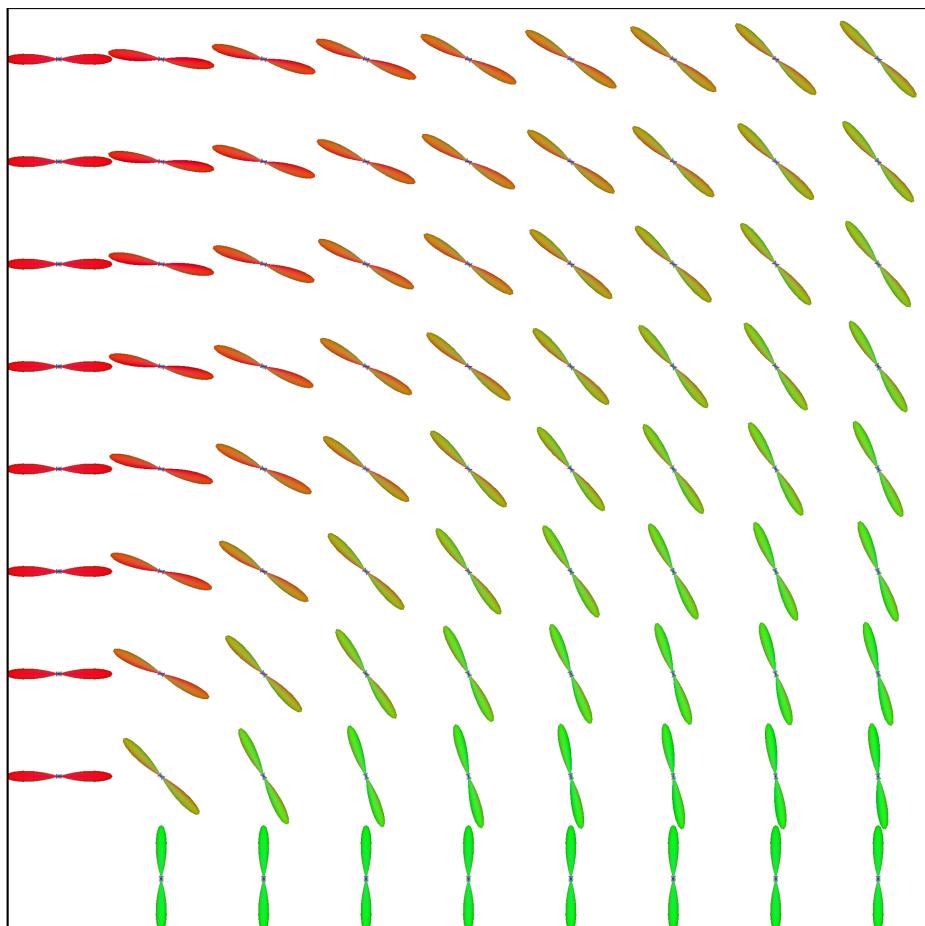
---

# A Anhang

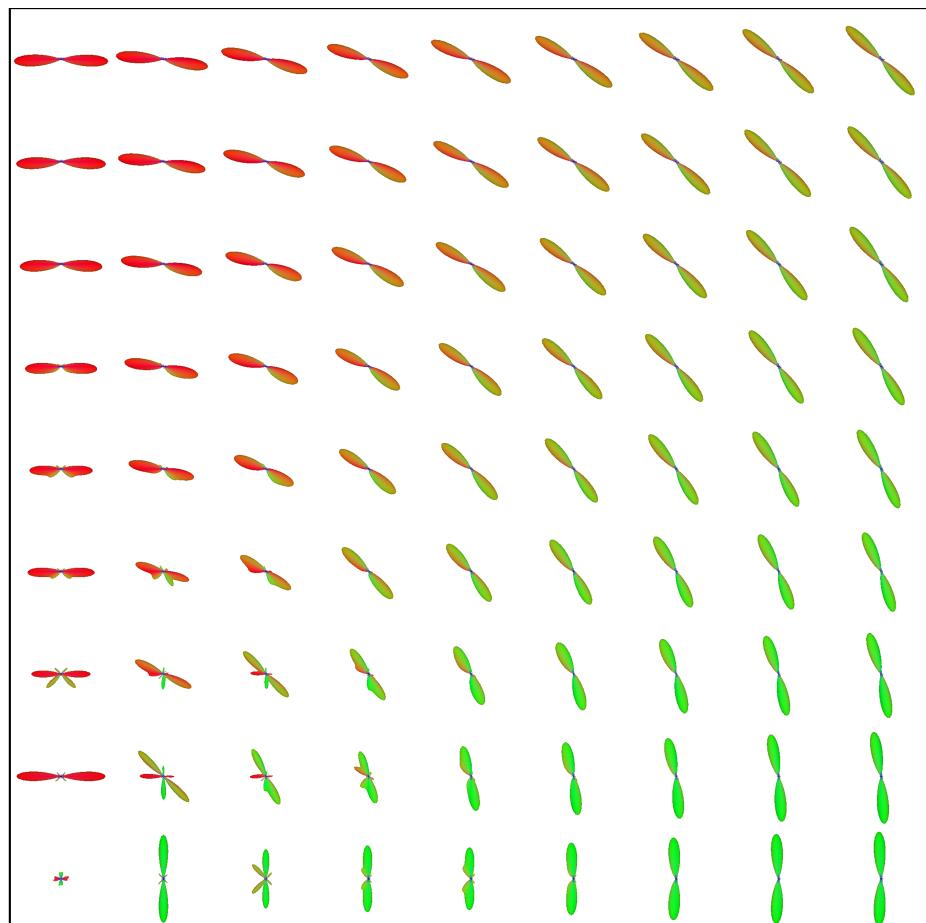
Der gesamte Code und weitere Bilder sind auf <https://github.com/TimLockstedt/BA> zu finden.



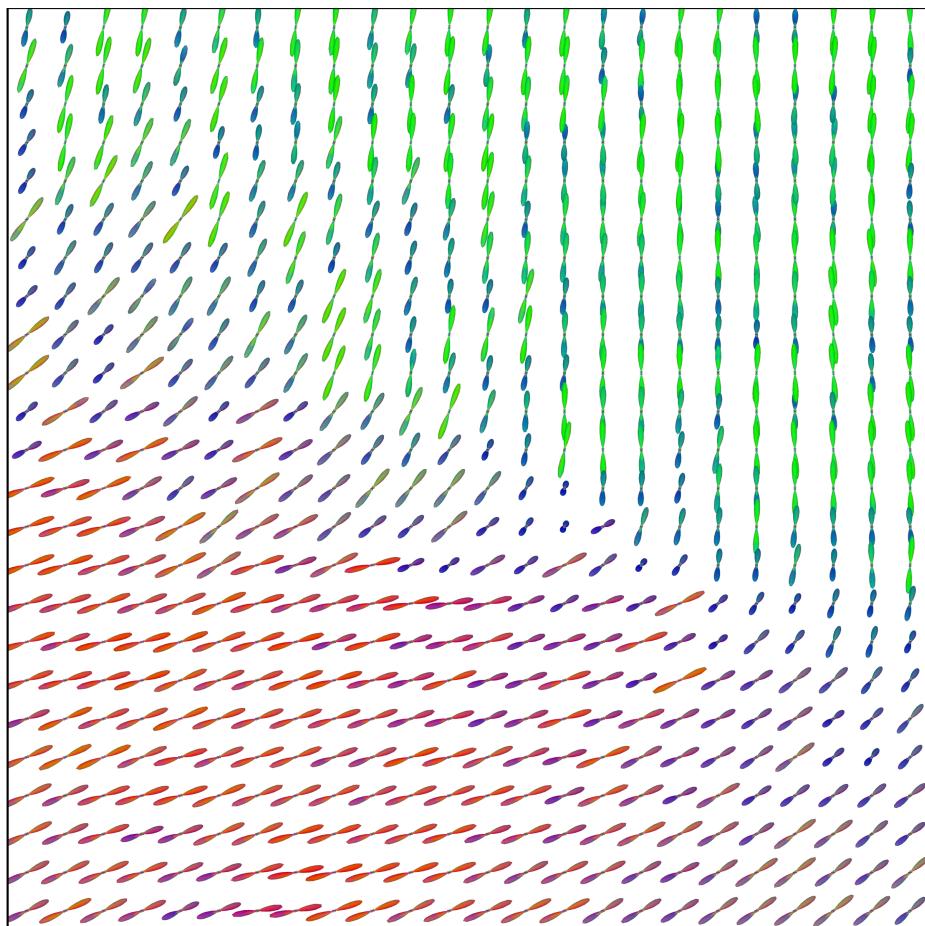
**Abbildung A.1** Große Stern Verteilung mit Zentrum unten Links, ODFs



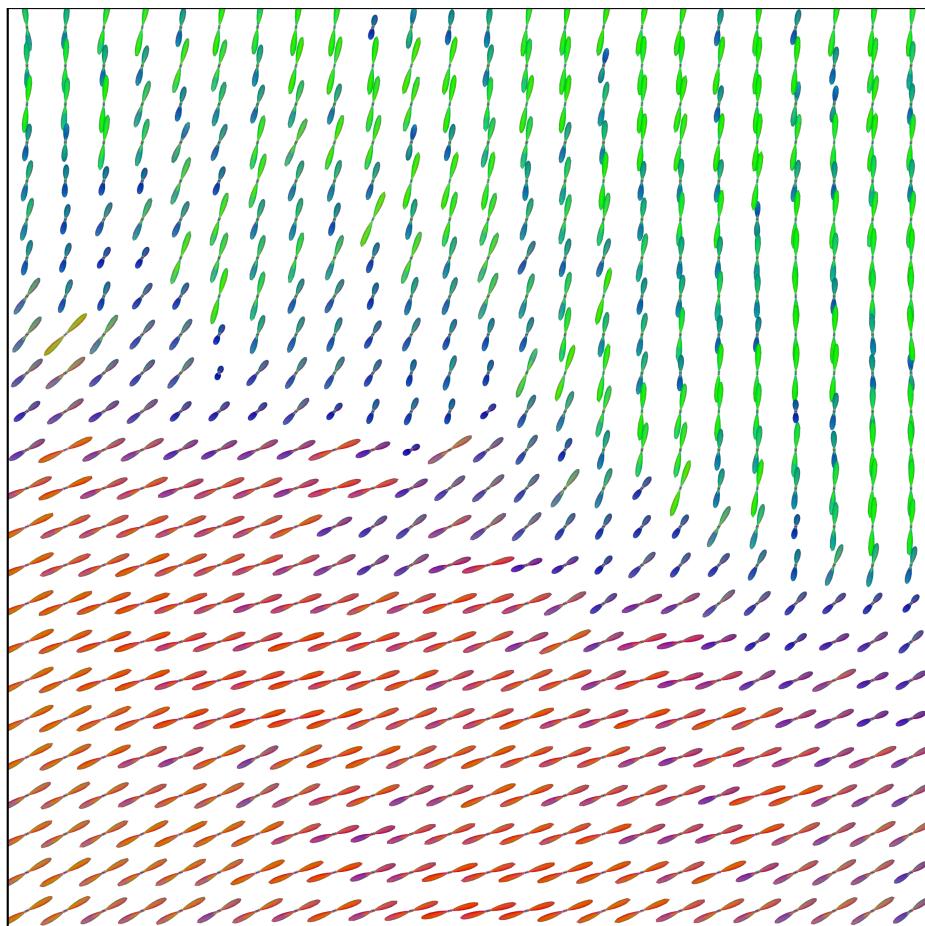
**Abbildung A.2** Große zirkuläre Verteilung mit Zentrum unten Links, ODFs



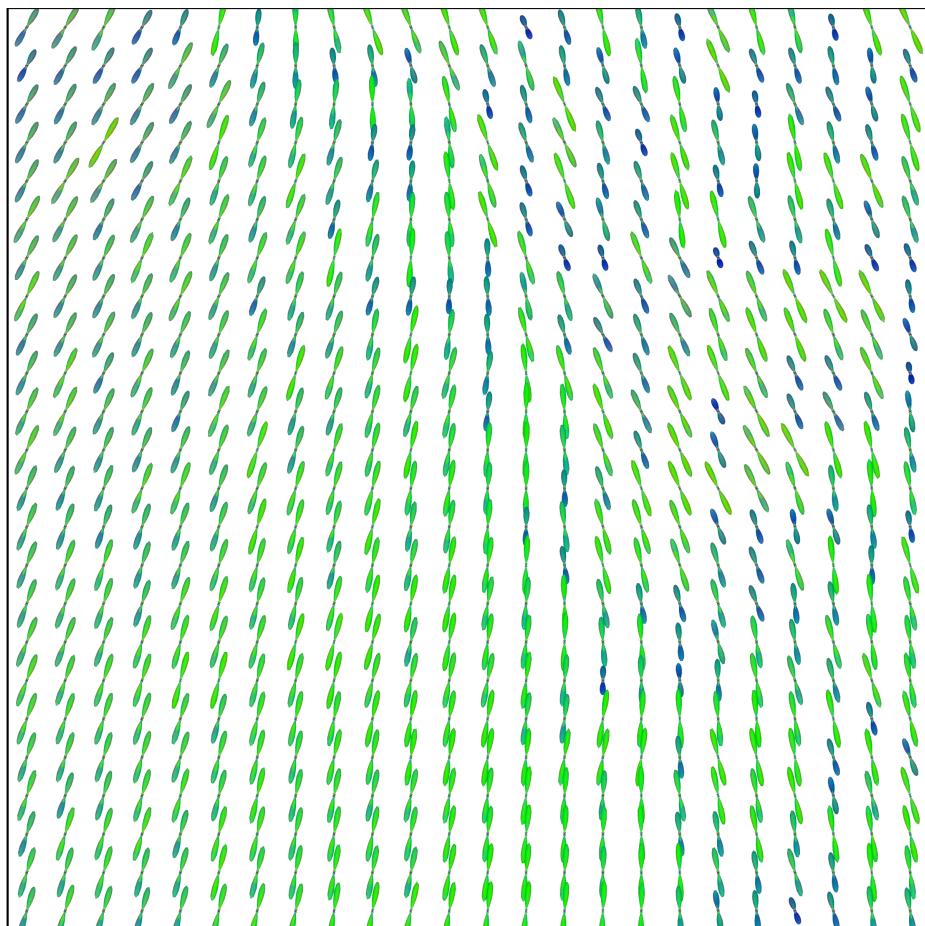
**Abbildung A.3** Große zirkuläre Verteilung mit Zentrum unten Links, AODFs mit  $n = 1500, l = 3, \sigma = 0.3, F_{AODF} = 100$



**Abbildung A.4** ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 300



**Abbildung A.5** ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 301



**Abbildung A.6** ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 302

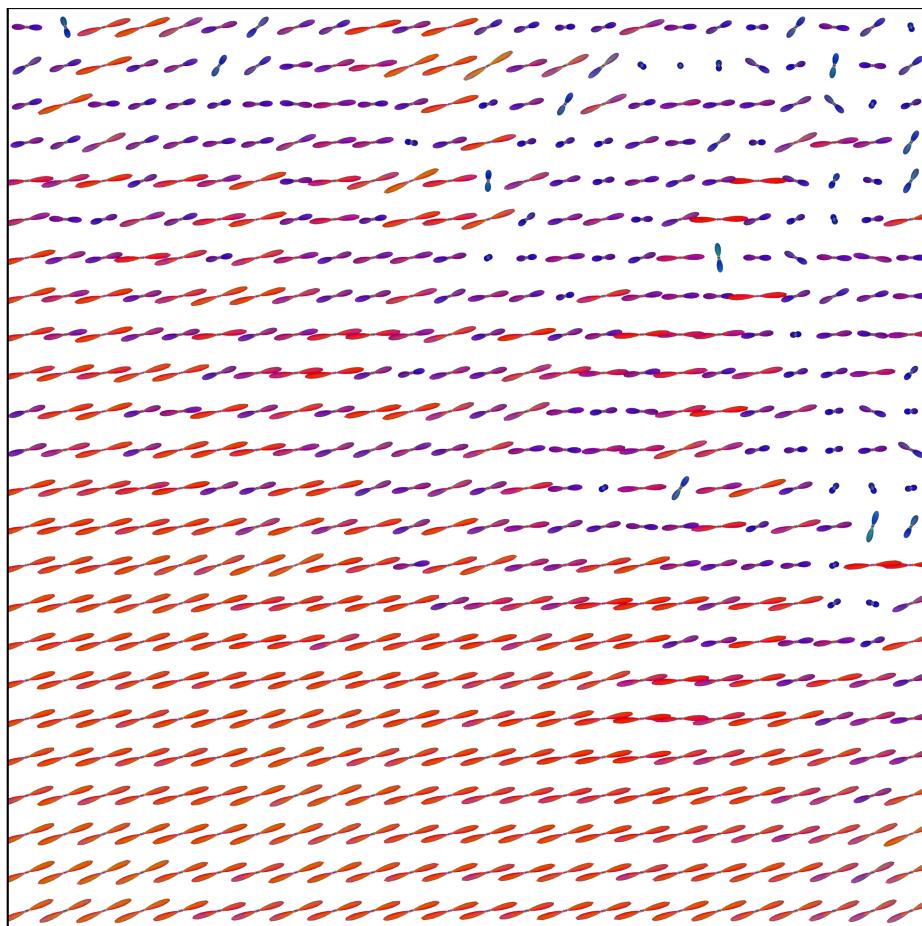
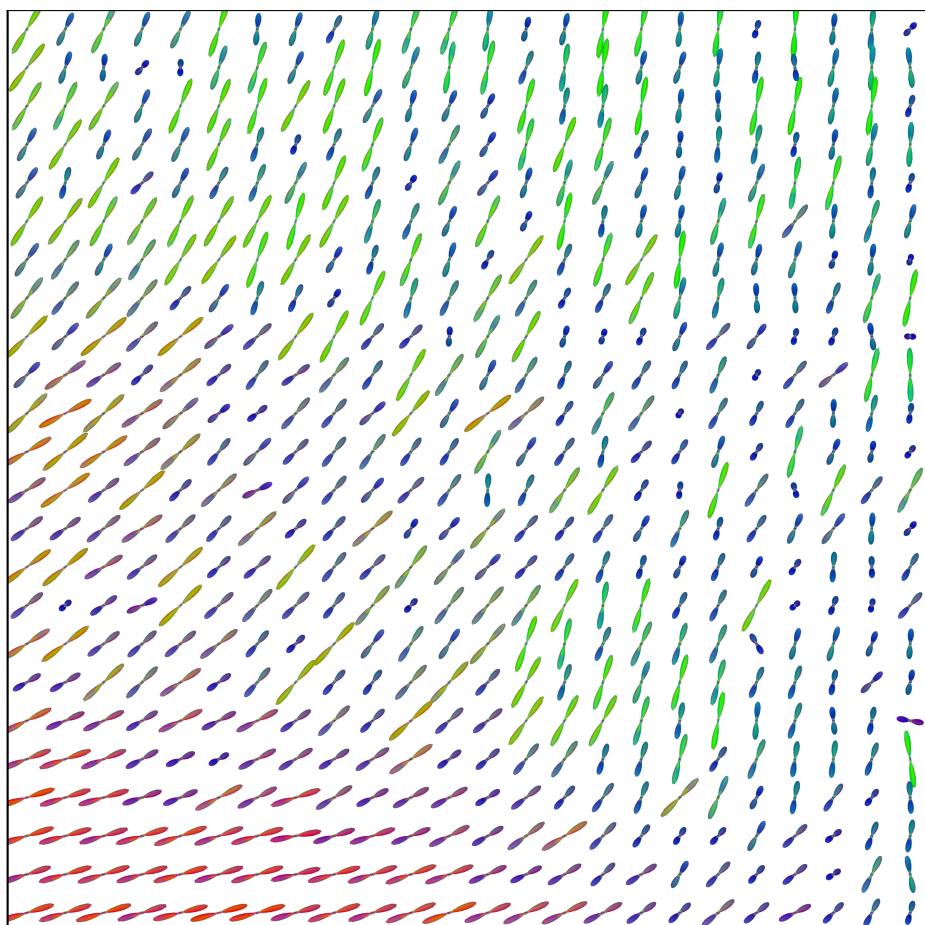
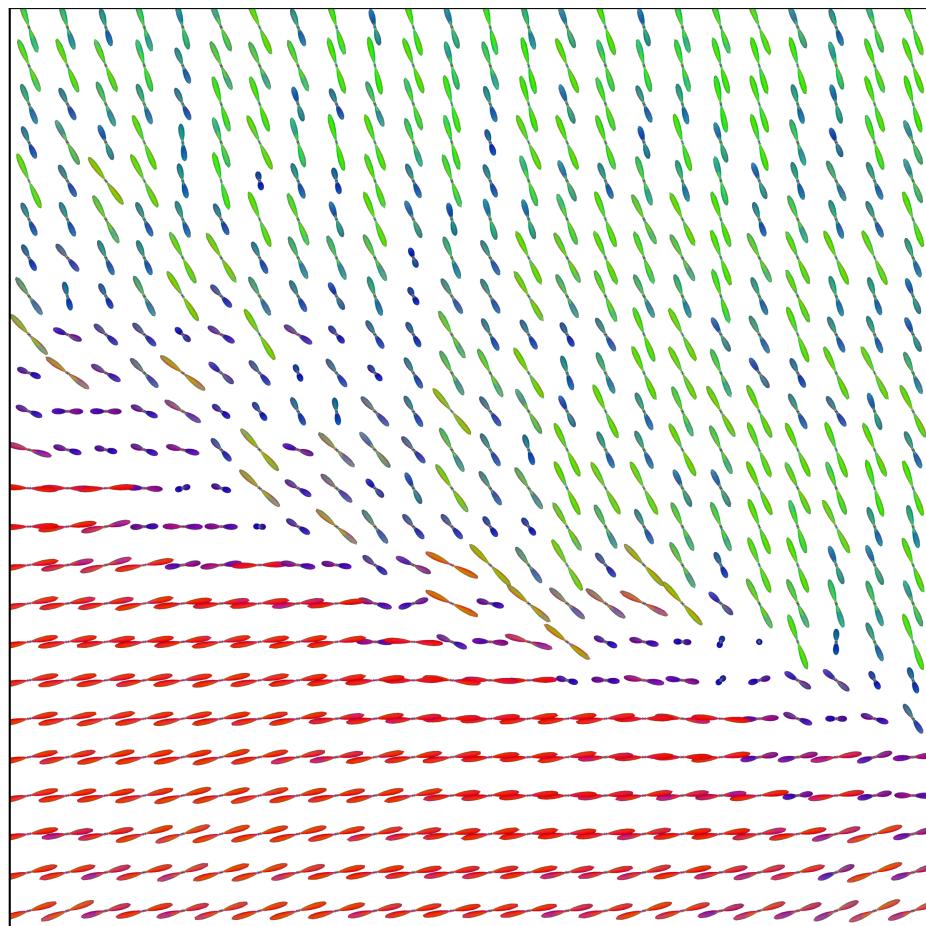


Abbildung A.7 ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 303



**Abbildung A.8** ODFs der PLI Daten, ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 304



**Abbildung A.9** ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 305

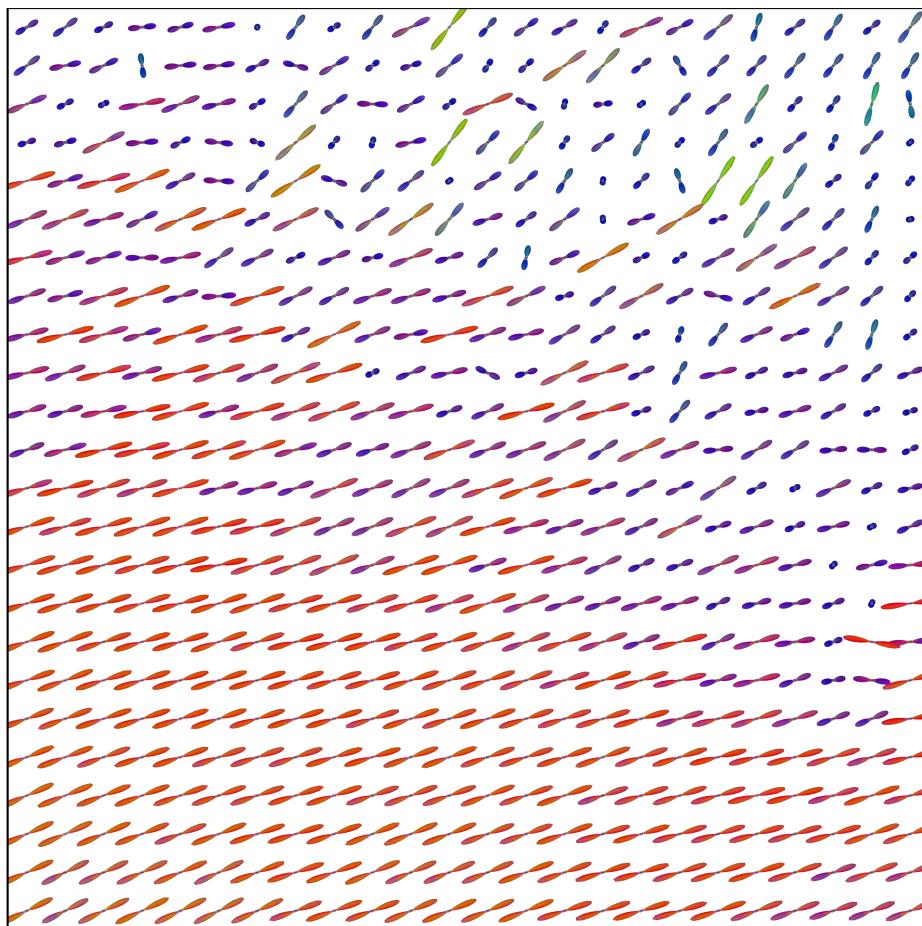


Abbildung A.10 ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 306

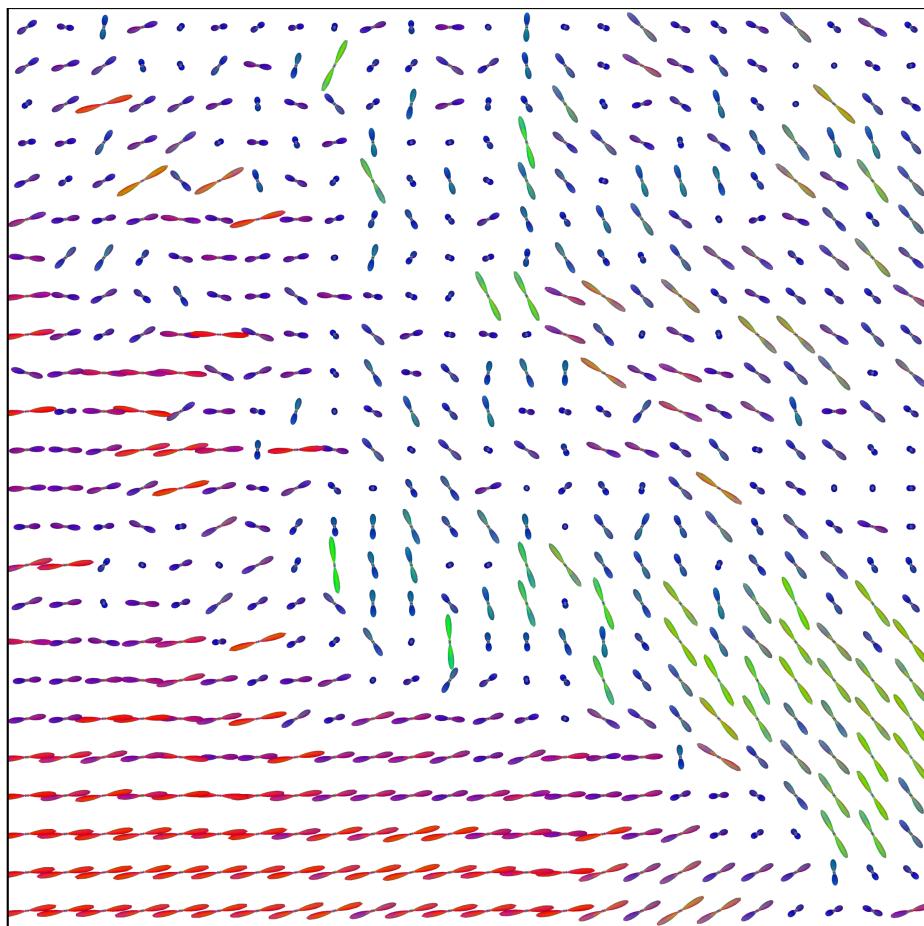
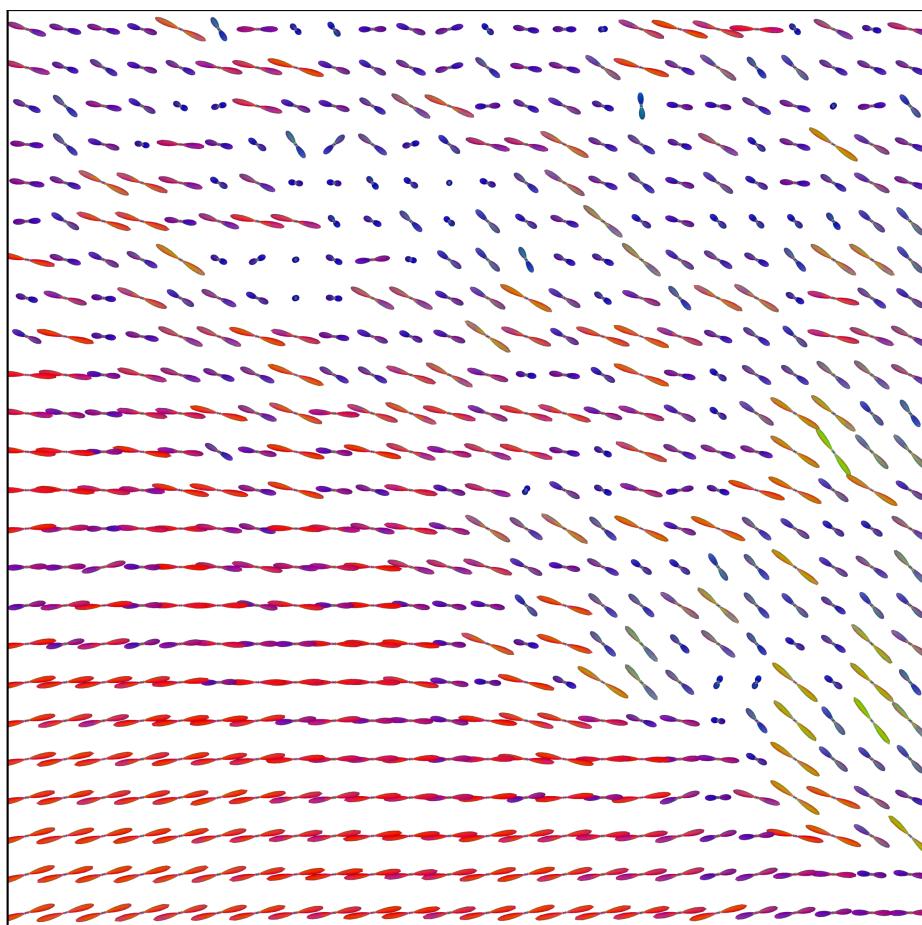
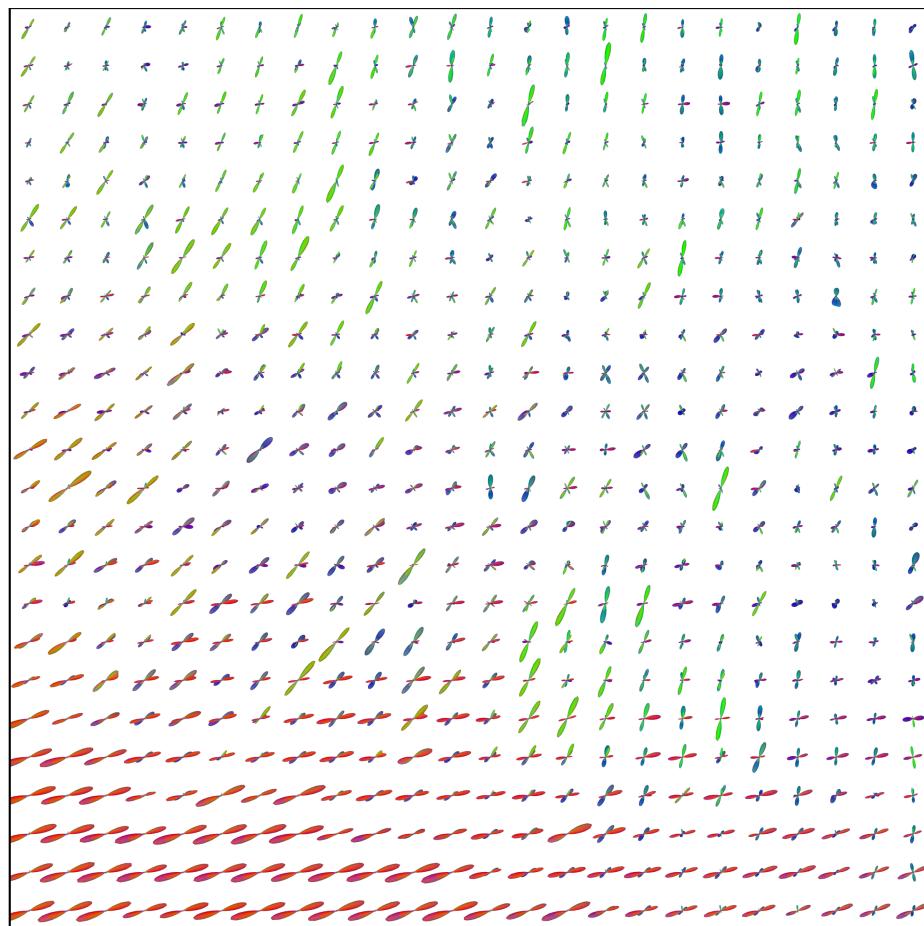


Abbildung A.11 ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 307



**Abbildung A.12** ODFs des Ausschnitts der PLI Daten 5.7a, Ebene 308



**Abbildung A.13** AODFs des Ausschnitts der PLI Daten, Ebene 304,  $\sigma = 0.5, l = 3, F_{AODF} = 10$

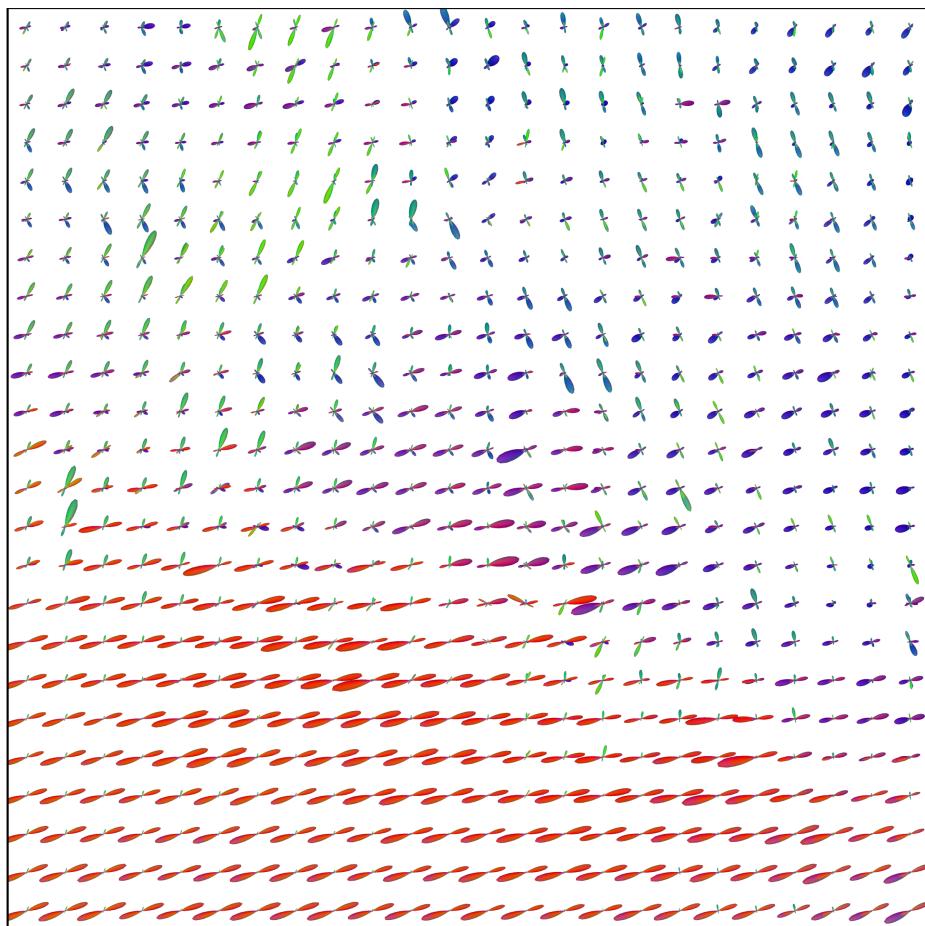
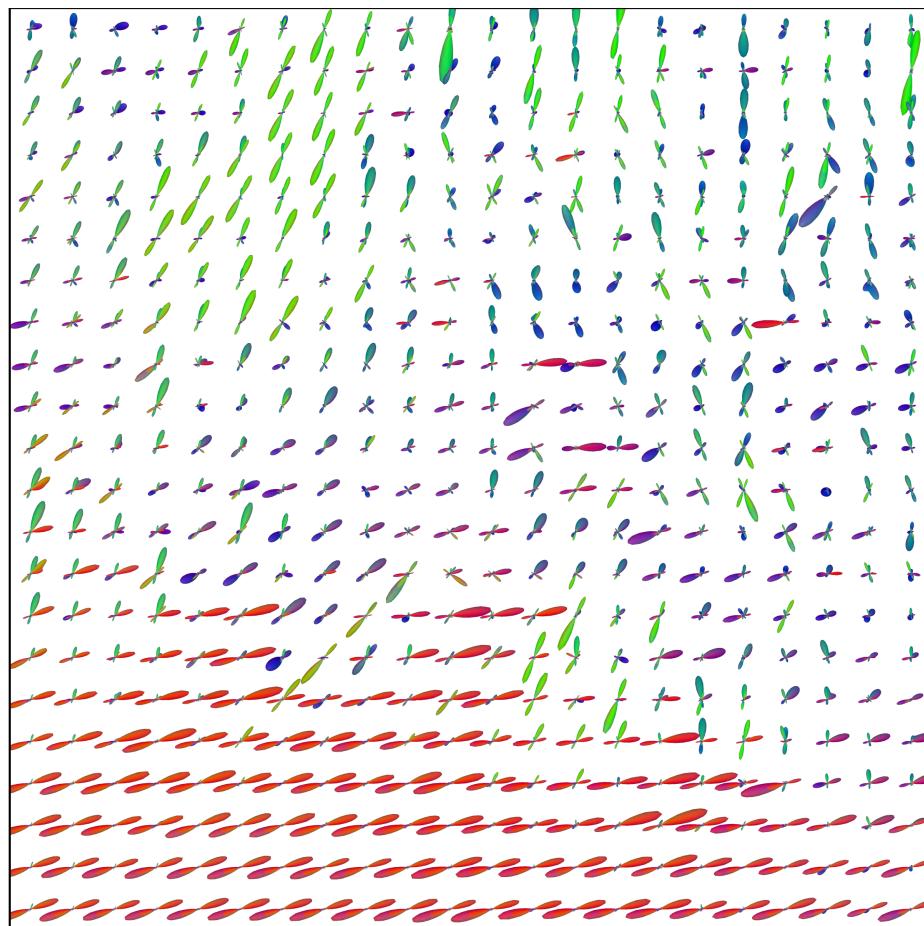
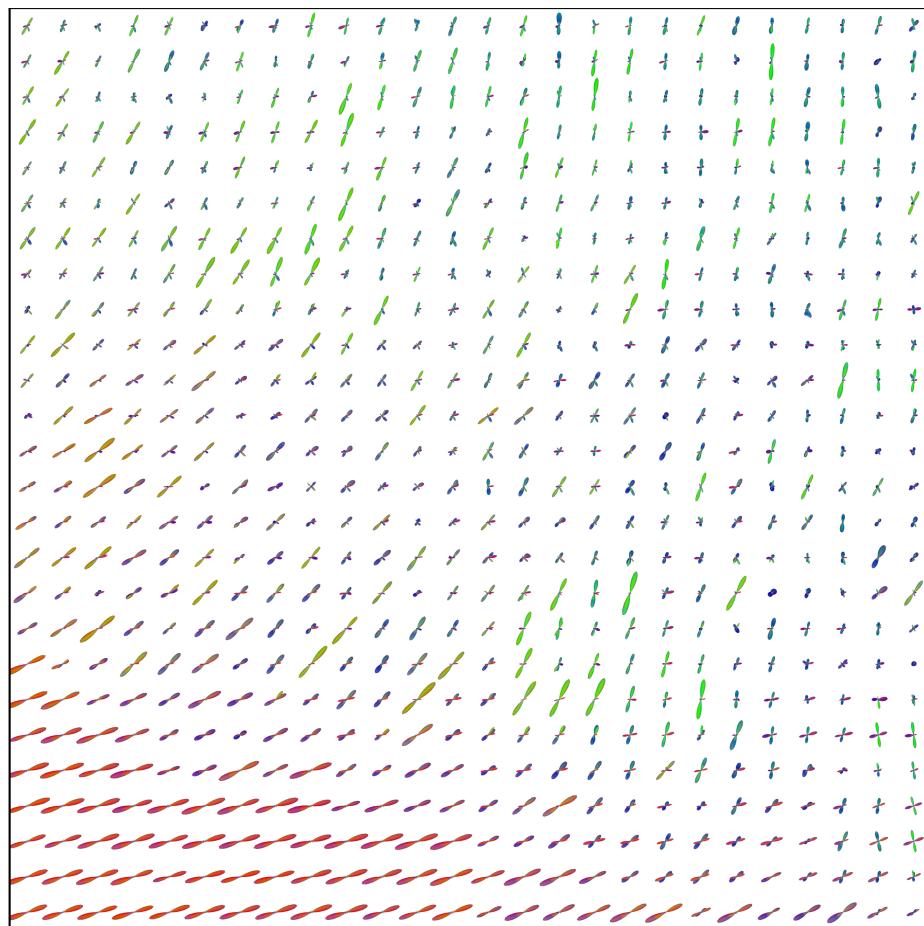


Abbildung A.14 AODFs des Ausschnitts der PLI Daten, Ebene 304,  $\sigma = 2, l = 3, F_{AODF} = 10$

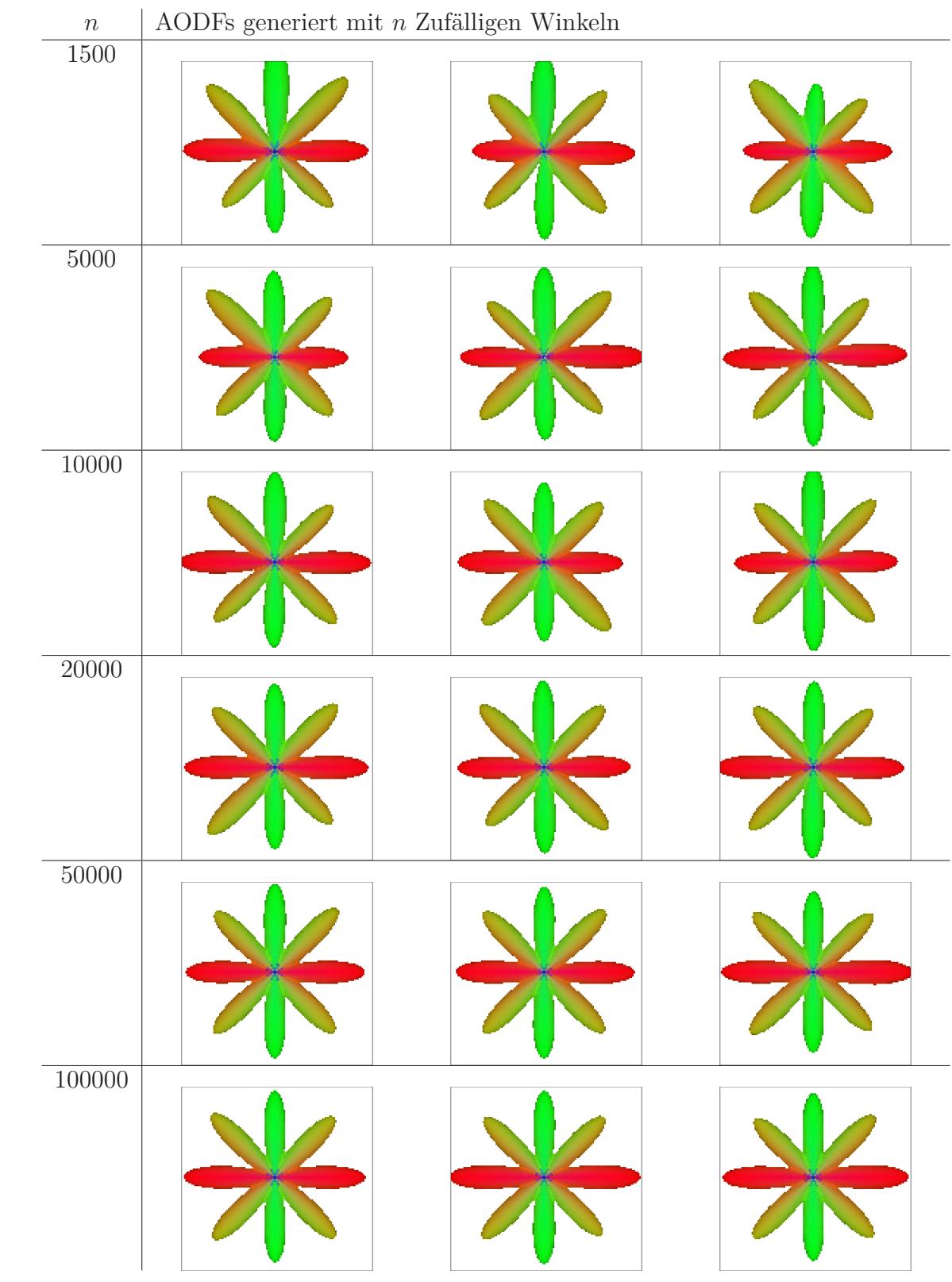


**Abbildung A.15** AODFs des Ausschnitts der PLI Daten, Ebene 304,  $\sigma = 0.5, l = 4, F_{AODF} = 10$



**Abbildung A.16** AODFs des Ausschnitts der PLI Daten, Ebene 304,  $\sigma = 0.5$ ,  $l = 2$ ,  $F_{AODF} = 10$

**Tabelle A.1** Vergleich der AODFs Basierend auf der Stern ODF Verteilung, generiert mithilfe von  $n$  Zufälligen Sampling Punkten



Name, Vorname: \_\_\_\_\_

## Erklärung

gem. §15 Abs. 6 der Prüfungsordnung vom 25.11.2019

Hiermit erkläre ich, dass ich die Bachelor-Thesis selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

---

(Datum)

---

(Unterschrift)

## Erklärung

Hiermit erkläre ich mich damit einverstanden, dass meine Abschlussarbeit (Bachelor-Thesis) wissenschaftlich interessierten Personen oder Institutionen und im Rahmen von externen Qualitätssicherungsmaßnahmen des Studienganges zur Einsichtnahme zur Verfügung gestellt werden kann. Korrektur- oder Bewertungshinweise in meiner Arbeit dürfen nicht zitiert werden.

---

(Datum)

---

(Unterschrift)