Fachhochschule Aachen
Campus Jülich

Fachbereich: Medizintechnik und Technomathematik
Studiengang: Technomathematik
Matrikelnummer: 869601

Implementierung und Parallelisierung einer 3D-Strukturtensoranalyse auf PLI-Bildern

Masterarbeit von

Andreas Müller

22. August 2017

Eigenständigkeitserklärung

Diese Arbeit ist von mir selbstständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Ort und Datum

Andreas Müller

Diese Arbeit wurde betreut von:

Erstprüfer: Prof. Dr. Johannes Grotendorst (JSC)
Zweitprüfer: Dr. Markus Axer (INM-1)

Diese Arbeit wurde erstellt im Forschungszentrum Jülich am Jülich Supercomputing Centre (JSC) in Kooperation mit dem Institut für Neurowissenschaften und Medizin (INM-1)

Abstract

Um den Verlauf von Nervenfasern im menschlichen Gehirn nachvollziehen zu können, wurde am Institut für Neurowissenschaften und Medizin des Forschungszentrum Jülich die Methode *Polarized Light Imaging*, kurz PLI, entwickelt. Dabei wird ein Gehirn post mortem in 60 µm dünne Scheiben (ca. 2000 pro Gehirn) geschnitten und durch ein Mikroskop aufgenommen.

Um die zur Faserdarstellung benötigte Auflösung zu erreichen, werden die Schnitte jeweils kachelweise aufgenommen. Die so erzeugten Daten werden vom PLI-Rekonstruktionsworkflow dreidimensional rekonstruiert. Hierbei werden unter anderem die beiden Raumwinkel der Fasern bestimmt. In weiteren Schritten werden eine Maske zur Trennung von Hirn und Hintergrund berechnet und die Kacheln zu einem Gesamtschnitt zusammengefügt. Danach werden die einzelnen Schnitte registriert, d.h. strukturell passend übereinandergelegt.

Die Raumwinkel dieser 3D-Karte, dem Volumen, konnten bisher nicht verifiziert werden. Dazu wird in dieser Arbeit eine 3D-Strukturtensoranalyse entwickelt und auf die Daten angewandt. Ziel der Arbeit ist ein Vergleich der Raumwinkel des PLI-Rekonstruktionsworkflows mit den Ergebnissen der 3D-Strukturtensoranalyse.

Darüber hinaus wurde das zur Berechnung implementierte Programm parallelisiert, um den Zeitaufwand der Berechnungen signifikant zu senken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Eines der sicherlich anspruchsvollsten und komplexesten Vorhaben der heutigen Zeit ist der Versuch, das menschliche Gehirn in seiner Struktur und Funktionsweise zu verstehen und zu simulieren. Als eines von zwei langzeitgeförderten Flagship-Projekten unterstützt die Europäische Union das *Human Brain Project* [BMBF], unter dessen Dach Wissenschaftler aus vielen verschiedenen Ländern und Bereichen, wie z.B. Neurowissenschaften, Informatik, Physik und Mathematik, zusammenarbeiten. Ziel des Projektes ist, zukünftig das Gehirn in seiner Gesamtheit simulieren zu können.

Das Forschungszentrum Jülich ist mit verschiedenen Instituten daran beteiligt. Dazu zählen das Institut für Neurowissenschaften und Medizin (INM) und das Jülich Supercomputing Centre (JSC).

Um das menschliche Gehirn simulieren zu können, muss zunächst herausgefunden werden, wie die etwa 86 Mrd. Nervenzellen [AC+09] miteinander verbunden sind. Um Faserdarstellung zu ermöglichen, wurde am INM-1 ein Bildgebungsverfahren entwickelt, dessen Auflösung in den Bereich der Größe von Nervenfasern vordringt: 3D Polarized Light Imaging (3D-PLI) [AA+11].

Dafür wird ein Gehirn post mortem in 60 µm dünne Scheiben geschnitten und mit polarisiertem Licht durchleuchtet. Die transmittierte Lichtintensität wird aufgenommen und mit Hilfe des PLI-Rekonstruktionsworkflows der Verlauf der Faser in einem Bildpunkt bestimmt. Die kleinste darstellbare Strukturgröße liegt bei 1.3 µm [AS+16]. Diese Auflösung reicht grundsätzlich zur Faserdarstellung aus.

Um Korrektheitsausssagen über die durch PLI gewonnenen Richtungsinformationen von Fasern treffen zu können, soll in dieser Arbeit eine 3D-Strukturtensoranalyse durchgeführt werden. Diese beruht auf keinerlei PLI-spezifischen Annahmen und kann auf jedes 3D-Grauwertbild angewendet werden. Als Ergebnis liefert die 3D-Strukturtensoranalyse ebenfalls Richtungsinformationen, welche mit Ergebnissen des PLI-Rekonstruktionsworkflows verglichen werden können.

2. PLI und Rekonstruktionsworkflow

Um aus den Bildinformationen der einzelnen Schnitte des Gehirns ein räumliches Fasermodell zu erzeugen, müssen die Richtungsinformationen der Fasern berechnet werden. Da diese nicht direkt aufgenommen werden können, müssen sie nach Aufnahme der Schnitte in einem anschließenden Workflow bestimmt werden.

2.1. Polarized Light Imaging

Die meisten Nervenfasern im Gehirn sind von einer dünnen Schicht aus Myelin umgeben (Abb. 2.1), die unter anderem der elektrischen Isolation dient. PLI basiert auf der doppelbrechenden Eigenschaft des Myelins mit optischer Achse in Faserrichtung. Die Bestimmung der Faserrichtung kann also indirekt über die Bestimmung der Richtung der optischen Achse der Myelinscheide erfolgen.

Da die dünnsten myelinisierten Fasern einen Durchmesser von etwa 3 µm haben [SHF11], reicht PLI zur Faserdarstellung aus.

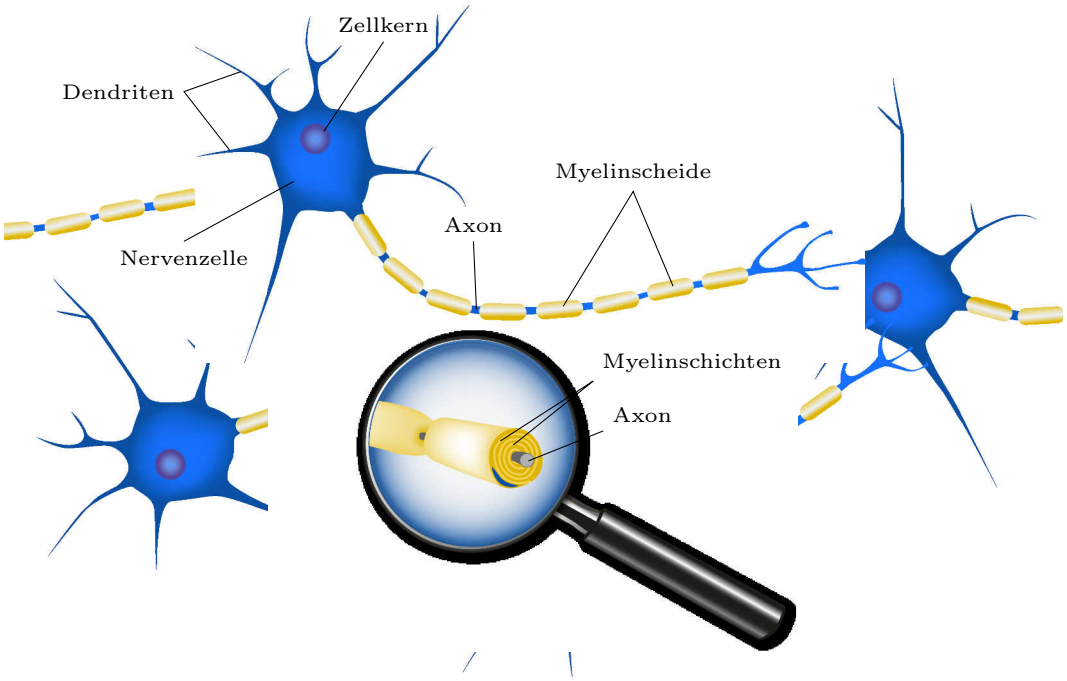
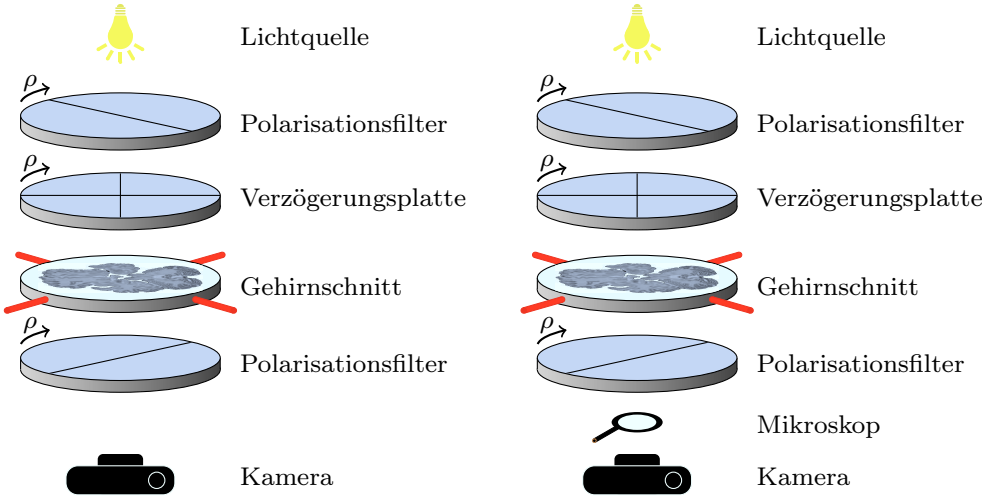


Abbildung 2.1.: Aufbau einer myelinisierten Nervenfaser

Jeder der etwa 2000 Gehirnschnitte wird in dem dafür entwickelten Versuchsaufbau mit zirkular polarisiertem Licht einer bestimmten Wellenlänge λ durchleuchtet. An den myelinisierten Fasern des Schnittes erfährt es Doppelbrechung und elliptische Polarisation. Um nur den doppelt gebrochenen Anteil des Lichtes zu vermessen, wird ein weiterer Polarisationsfilter eingesetzt, dessen Transmissionsachse senkrecht zu der des ersten Polarisationsfilters steht (Abb. 2.2a). Diesen Aufbau nennt man Large Area Polarimeter (LAP), da er den gesamten Schnitt aufnehmen kann.

Weil die eingesetzte Kamera eine begrenzte Auflösung hat, ist es beim LAP nicht möglich, einzelne Fasern darzustellen. Um dieses Problem zu lösen, werden beim Aufbau mit dem Namen Polarisationsmikroskop (PM) die Schnitte mit Hilfe eines Mikroskopes aufgenommen (Abb. 2.2b).

Durch das Mikroskop wird die Auflösungskraft des Verfahrens erhöht. Gleichzeitig führt dies dazu, dass der Schnitt nicht mehr in seiner Gesamtheit aufgenommen werden kann. Deshalb muss der Gesamtschnitt in Teilkacheln unterteilt werden. Um bei der Aufnahme der 2048×2048 Pixel großen Kacheln keine Informationen zu verlieren, werden diese überlappend aufgenommen.



(a) Large Area Polarimeter (b) Polarisationsmikroskop

Abbildung 2.2.: Schematischer Aufbau von Large Area Polarimeter (LAP) und Polarisationsmikroskop (PM). Beim LAP wird ein Schnitt in seiner Gesamtheit aufgenommen, beim PM kachelweise, aber in maximaler Auflösung.

Informationen über den Verlauf der Brechungsachse können nicht aus einem einzelnen Bild extrahiert werden. Dafür sind Informationen über die Stärke der Doppelbrechung aus unterschiedlichen Richtungen nötig. Deshalb werden pro Kachel 18 Bilder aufgenommen, jeweils mit um 10° vergrößertem Winkel ρ der Polarisationsfilter bzw. Verzögerungsplatte. Da der Winkel, in dem das Licht auf eine einzelne Faser trifft, sich abhängig von der Position des Polarisationsfilters ändert, ändert sich auch die Stärke der Doppelbrechung und somit die Intensität des aufgenommenen Lichtes. So ergibt sich für jedes Pixel über diese 18 Bilder gesehen ein individueller Helligkeitsverlauf.

2.2. PLI-Rekonstruktionsworkflow

Um aus einzelnen Kacheln mit je 18 Bildern ein 3D-Gesamtbild zu erzeugen, durchlaufen die Daten den PLI-Rekonstruktionsworkflow. Dieser berechnet in mehreren Schritten die räumliche Orientierung, fügt die Kacheln wieder zu Schnitten zusammen und diese Schnitte zu einem Gesamtvolumen.

2.2.1. Kalibrierung

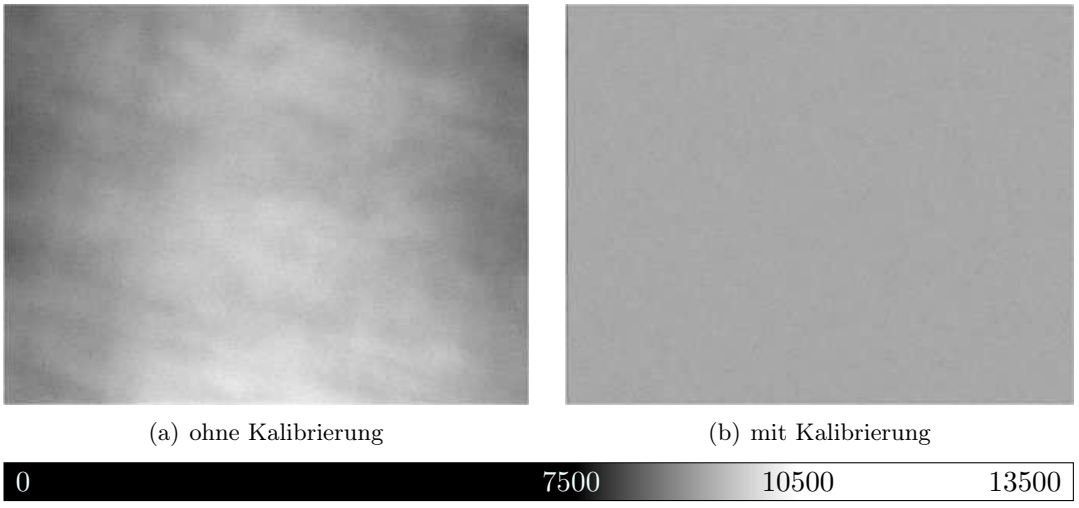
In einem ersten Schritt werden dabei die Helligkeitswerte der einzelnen Pixel angeglichen. Dies ist nötig, da trotz festem Aufbau keine hundertprozentig homogene Ausleuchtung möglich ist. Gründe dafür sind beispielsweise signalbeeinflussende elektrische Ströme innerhalb der Kamera und minimale Inhomogenitäten der Polarisationsfilter. Um diese Effekte auszugleichen, ist eine Kalibrierung der Aufnahmen nötig [DA+10].

Dafür werden für jede der 18 Stellungen der Filter 100 Leerbilder aufgenommen, d.h. ohne Objekt. Über diese 100 Bilder wird nun der Durchschnitt berechnet, um ein möglichst genaues Null-Bild $Im_{Null}(\rho)$ zu erhalten. Die Referenzintensität I_{ref}, auf die das Nullbild kalibriert werden soll, ist das Maximum der Dichtefunktion über alle 18 Bilder hinweg. Die Faktoren, mit denen die Helligkeitsangleichung erfolgt, heißen Gainfaktoren und werden berechnet durch Formel 2.1.

$$g(x, y, \rho) = \frac{I_{ref}}{Im_{Null}(x, y, \rho)} \tag{2.1}$$

Die Kalibrierung (Abb. 2.3) erfolgt dann anhand von Formel 2.2, wobei $Im(x, y, \rho)$ das kalibrierte und $Im'(x, y, \rho)$ das unkalibrierte Bild ist.

$$Im(x, y, \rho) = g(x, y, \rho) \cdot Im'(x, y, \rho) \tag{2.2}$$



(a) ohne Kalibrierung

(b) mit Kalibrierung

(c) Skala der Grau-/Helligkeitswerte

Abbildung 2.3.: Leerbild vor und nach der Kalibrierung. Die Grauwerte decken zur Verdeutlichung nur einen Teil der Helligkeitsskala ab.

2.2.2. Direktion, (normierte) Transmittanz und Retardierung

Trägt man die Helligkeitswerte eines Pixels über den Drehwinkel der Polarisationsfilter bzw. der Verzögerungsplatte auf, so lässt sich der Helligkeitsverlauf der 18 Bilder durch eine Sinuskurve approximieren (Abb. 2.4). Der Verlauf einer solchen Sinuskurve wird durch drei Werte charakterisiert: Die Verschiebungen an x- und y-Achse sowie die Amplitude der Kurve.

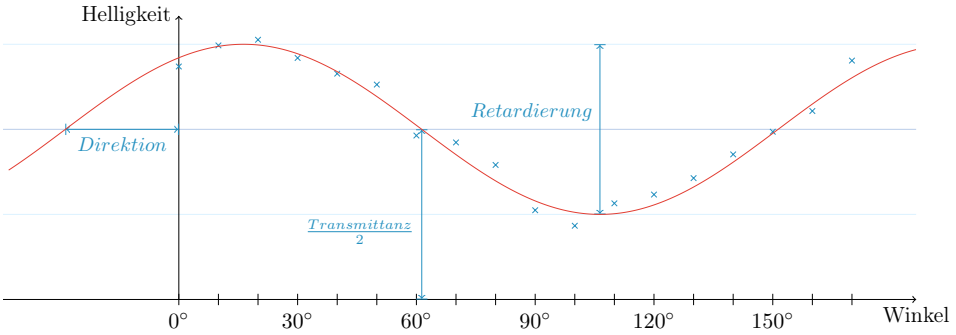


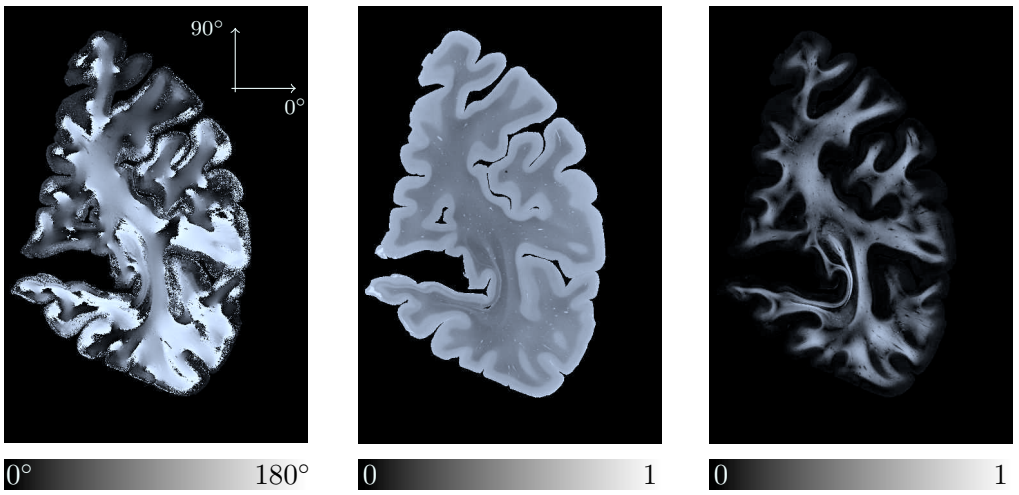
Abbildung 2.4.: Bestimmung von Direktion, Transmittanz und Retardierung aus dem Helligkeitsverlauf eines Pixels der 18 Bilder einer Kachel

Die Verschiebung entlang der x-Achse wird **Direktion** genannt. Diese entspricht dem Winkel der Projektion der Faser innerhalb der Schnittebene zur Null-Grad-Richtung (Abb. 2.5).

Die Verschiebung der Sinuskurve in y-Richtung, d.h. die Höhe der Ruhelage der Kurve, ist die Hälfte der **Transmittanz**. Sie ist ein Maß für die Lichtdurchlässigkeit des Schnittes an dieser Stelle. Um sie für weitere Berechnungen zu vereinheitlichen und eventuelle Unregelmäßigkeiten, die in der Belichtungszeit oder der Lagerung des Schnittes begründet liegen, zu entfernen, wird die **Transmittanz normiert**. Dazu dividiert man die Werte der Transmittanz pixelweise durch die Transmittanzwerte eines Leerbildes, also einer Aufnahme ohne Gehirnschnitt.

Die Amplitude der Sinuskurve, die Differenz zwischen Maximum und Minimum, ist der **Retardierungswert**. Er ist ein Maß für die Doppelbrechung in diesem Punkt, da die Amplitude der Sinuskurve abhängig ist vom Anteil des doppelt gebrochenen Lichtes. Dieser ist am größten, wenn die Faser in der Ebene verläuft, und am kleinsten, wenn sie senkrecht zur Ebene steht, da entlang der optischen Achse keine Doppelbrechung stattfindet.

Für jedes Bild werden pro Pixel nicht die 18 Werte, sondern die Modalitäten (Direktion, Transmittanz und Retardierung) gespeichert. Dieser Vorgang ist verlustfrei, da man davon ausgehen kann, dass die Abweichungen zur Sinuskurve durch Aufnahmefehler und -ungenauigkeiten begründet sind. Die Speicherung der Charakteristika der Sinuskurve hat den positiven Nebeneffekt, dass dadurch die zu verarbeitenden Daten auf ein Sechstel ihrer Größe reduziert werden können. So sinkt beispielsweise der Speicherplatzbedarf eines Schnittes, der in 70×50 überlappende Kacheln aufgeteilt wurde, von 70×50 (Kacheln) $\cdot 2048 \times 2048$ (Pixel/Bild) $\cdot 18$ (Bilder/Kachel) $\cdot 4$ (Bytes/Pixel) $= 984.375$ GiB um den Faktor 6 auf etwa 164 GiB.



(a) Direktion (b) normierte Transmittanz (c) Retardierung

Abbildung 2.5.: Beispielbilder für Direktion, normierte Transmittanz und Retardierung. Die 0-Grad-Richtung im Direktionsbild verläuft waagrecht von links nach rechts. Alle Winkel werden gegen den Uhrzeigersinn gemessen. Der in diesem Kapitel für Beispielbilder verwendete Datensatz wurde aus der linken Hemisphäre des Hirns eines Patienten mit Multisystematrophie gewonnen. Die Schnittebene ist frontal.

2.2.3. Stitching

Unter Zuhilfenahme der normierten Transmittanz werden die einzelnen Kacheln wieder zu einem Gesamtschnitt zusammengefügt. Beim Stitching werden dafür in benachbarten Kacheln markante Punkte gesucht, die dann, durch Verschiebung der Kacheln zueinander, übereinandergelegt werden (Abb. 2.6). Dadurch wird es möglich, einzelne Schnitte in ihrer Gesamtheit zu verarbeiten. Auch würde eine kachelweise Verarbeitung der Daten nicht garantieren können, dass jeder reale Punkt des Gehirns nur in einem Pixel enthalten ist.

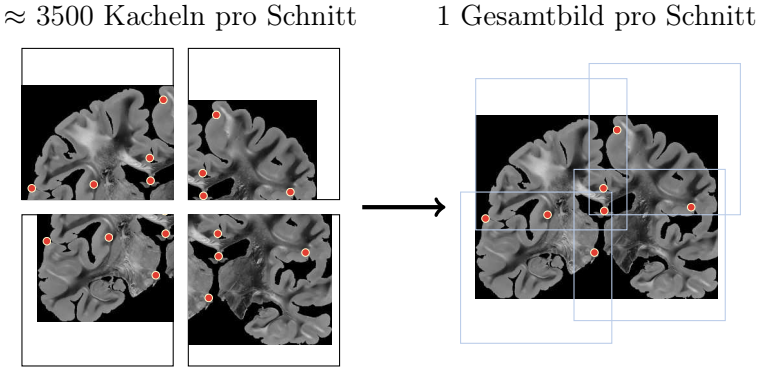


Abbildung 2.6.: Schematische Darstellung des Stitchings

Ebenso wird so die Datenmenge um knapp die Hälfte reduziert, da die einzelnen Kacheln in x- und y-Richtung zu etwa 30% überlappen (Abschnitt 2.1).

2.2.4. Segmentierung

Da das Transmittanzbild die klarste Abgrenzung zwischen Gehirn und Hintergrund, aber auch zwischen grauer und weißer Substanz besitzt, wird auf dieser Basis die Segmentierung durchgeführt [W13]. Dabei wird mit einem Region-Growing-Algorithmus eine Maske berechnet, die die Bildpunkte bestimmten Gehirnregionen zuordnet (Abb. 2.7). Dazu gehören Hintergrund, graue Substanz und weiße Substanz. Die weiße Substanz ist dabei der Bereich des Hirns, in dem hauptsächlich stark myelinisierte Nervenfasern verlaufen und weniger Nervenzellen liegen. Umgekehrt liegen die Nervenfasern in der grauen Substanz nicht so dicht beeinander.

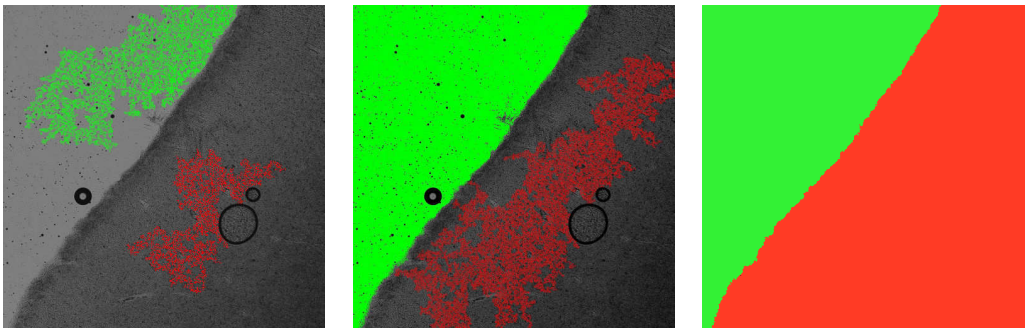


Abbildung 2.7.: Beispielverlauf des Region-Growings

Durch den Einsatz der Maske kann die Menge der zu verarbeitenden Daten weiter reduziert werden, da etwa ein Sechstel aller Daten zum Hintergrund gehört und bei den weiteren Berechnungen nicht mehr beachtet werden muss. Dafür benötigt man etwa 1% mehr Speicherplatz, da man die Daten der Maske im kleinsten Datentyp (1 Byte pro Pixel) abspeichern kann, weil man nur drei Werte unterscheiden muss. Diese sind die Indices für Hintergrund, graue Substanz und weiße Substanz.

2.2.5. Inklination

Nach dem Stitching der einzelnen Kacheln zu einem Gesamtbild für jeden Schnitt kann in jedem Pixel der Inklinationswinkel α zwischen Faser und Schnittebene berechnet werden (Abb. 2.8). Dabei wird, mit Hilfe von Formel 2.3, abhängig von der normierten Transmittanz $Trans$, jedem Retardierungswert Ret ein Inklinationswinkel α zugeordnet [M14].

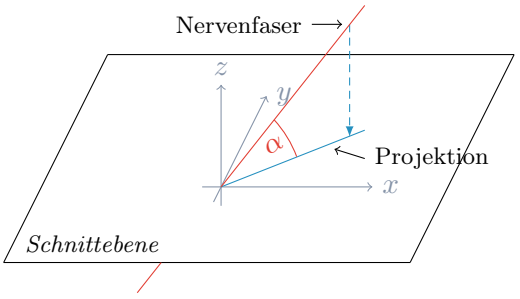


Abbildung 2.8.: Lage des Inklinationswinkels

$$\alpha \approx \arccos\left(\sqrt{\frac{\arcsin\left(Ret\right)}{\arcsin\left(x_{Fit}^{dyn}\right)} \cdot \frac{\log(Trans_{min})}{\log\left(\frac{Trans}{Trans_{max}}\right)}}\right) \tag{2.3}$$

$Trans_{min}$ ist der kleinste vorkommende Transmittanzwert im aktuell zu verarbeitenden Schnitt und $Trans_{max}$ entsprechend der größte. Wichtig dabei ist, dass der Wertebereich der Transmittanz schnittweise betrachtet wird, da diese stark von der Präparation und Lagerdauer der einzelnen Schnitte abhängt und deshalb von Schnitt zu Schnitt schwanken kann.

Weil das normierte Transmittanzbild stark verrauscht ist, muss es vorher geglättet werden. Dafür wird ein Nachbarschaftsmedianfilter angewendet. Das bedeutet, dass jedes Pixel so verändert wird, dass dessen Wert dem Median aller ungefilterten Werte in einem runden Bereich um das ursprüngliche Pixel entspricht.

Für die Berechnung des noch fehlenden Wertes x_{Fit}^{dyn} braucht man den Retardierungswert, dem ein Winkel von 0° zugeordnet wird, jeweils für graue und weiße Substanz. Dieser entspricht grundsätzlich dem größten Retardierungswert (Abschnitt 2.2.2). Dazu bildet man anhand der Maske (Abschnitt 2.2.4) über alle Schnitte hinweg je ein Histogramm für die Retardierungswerte der grauen und weißen Substanz. Man sucht nun am Ende des Histogrammes nach der Stelle mit der größten Krümmung. Dieser Wert entspricht dem Retardierungswert einer Faser, die in der Ebene verläuft. Alle größeren Retardierungswerte beinhalten nur Rauschen.

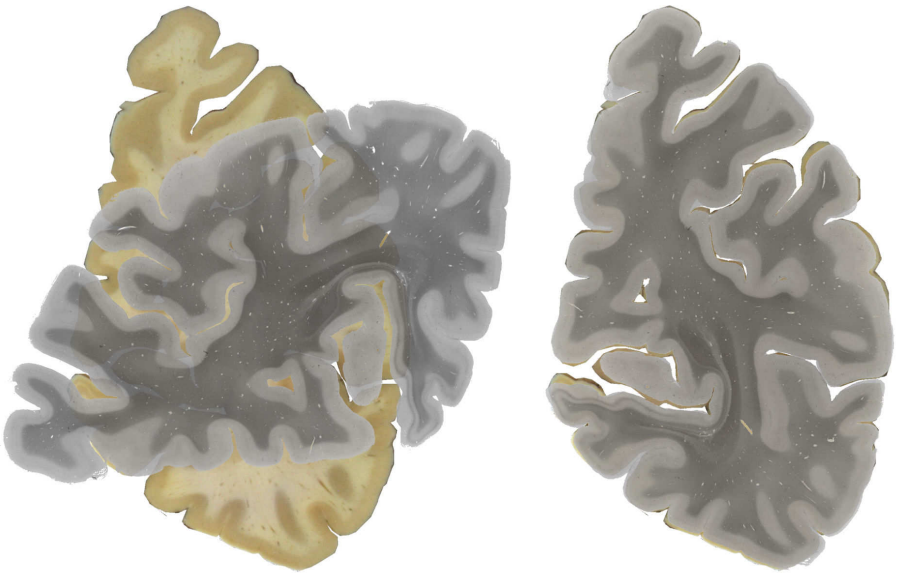
Interpoliert man nun in jedem Pixel anhand des Transmittanzwertes linear zwischen diesen beides Fitwerten, so erhält man den Wert x_{Fit}^{dyn}. Diese lineare Interpolation ist nötig, da im Gehirn die Übergänge zwischen grauer und weißer Substanz fließend sind und keine klaren Grenzen gezogen werden können.

2.2.6. Registrierung der einzelnen Schnitte

In einem letzten Schritt werden die Einzelschnitte zu einem 3D-Bild, dem PLI-Volumenbild, zusammengefügt [S+15]. Ein simples Übereinanderlegen reicht dabei nicht. So werden die Schnitte beispielsweise unterschiedlich auf den Objektträgern platziert, d.h. verdreht aufgenommen. Andererseits erfahren sie beim Schneidevorgang eine gewisse Verformung, reißen zum Teil sogar ein. Nach dem Schneiden ist es demnach nicht möglich, durch Zusammenlegen wieder die ursprüngliche Form zu erhalten.

Um für das spätere Zusammenfügen der Schnitte Informationen über die absolute Position markanter Punkte des Schnittes zu erhalten, wird vor dem Abschneiden eines Schnittes ein Bild des noch nicht geschnittenen Gehirns aufgenommen, das sogenannte Blockface-Bild. Auf dessen Basis ist es möglich, beim Schneiden und Aufnehmen auftretende Veränderungen des Schnittes wieder rückgängig zu machen (Abb. 2.9).

So haben zum Beispiel graue und weiße Substanz unterschiedliche Eigenschaften beim Schneiden. Unter anderem wegen ihrer hohen Myelindichte verformt sich weiße Substanz beim Schneiden anders als graue, weshalb das Bild unter Zuhilfenahme von Drehung, Streckung, Scherung und anderer Transformationen modifiziert werden muss, bis der transformierte Schnitt in markanten Punkten mit dem Blockface-Bild übereinstimmt.



(a) vor der Registrierung (b) nach der Registrierung

Abbildung 2.9.: Registrierung eines Schnittes auf dem Blockfacebild [DS15]. Grau: Transmittanzbild, braun: Entsprechendes Blockfacebild.

3. Strukturtensoranalyse

Um Richtungsinformationen aus Grauwertbildern zu extrahieren, kann man ebenso die Strukturtensoranalyse (STA) anwenden [BG87]. Sie dient zur Findung der Hauptrichtung von Gradienten in einem Bild. In diesem Fall wird sie genutzt, um die Richtung zu finden, in der die Gradienten minimal sind. Diese Richtung zeigt entlang einer Kante im Bild und entspricht dadurch der Richtung einer Nervenfaser. Man kann sie sowohl auf 2D-, als auch auf 3D-Bilder anwenden.

3.1. 2D-STA

Im zweidimensionalen Fall werden die Daten schnittweise unabhängig verarbeitet. Die 2D-STA dient der Kantenfindung innerhalb des Schnittes.

3.1.1. Glättung

Vor der Anwendung der Strukturtensoranalyse ist es üblich, das Bild zu glätten, um Rauschen zu entfernen und dadurch zufällige Gradienten zu minimieren (Abb. 3.1). Diese Glättung kann auf verschiedene Arten erfolgen. Eine sehr häufig genutzte Variante ist die diskrete Faltung des Bildes

$$Im_{glatt}(x,y) = Im_{orig}(x,y) * G(x,y) \tag{3.1}$$

mit '*' als Faltungsoperator und Gaußschem Faltungskern

$$G(x,y) = \frac{1}{2\pi\sigma^2} \cdot \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.2}$$

Das Bild Im_{orig} ist eine Matrix der Helligkeitswerte. Ebenso muss der Gaußsche Faltungskern diskret betrachtet werden. Dazu wertet man die Funktion $G(x,y)$ um den Mittelpunkt der Matrix herum aus.

Man wählt σ hierbei so, dass die Matrix 95% der Gaußglocke abdeckt, um ihre Gestalt möglichst gut wiederzugeben. Um 95% der Werte abzudecken, muss sie das entsprechende Konfidenzintervall $[-2\sigma; 2\sigma]$ enthalten. Dies erreicht man, indem man die Anzahl der Spalten bzw. Zeilen, abzüglich der mittleren, mit 4σ gleichsetzt.

Beispielhaft wird im Folgenden der 3×3-Filter berechnet. Das zur Berechnung nötige σ kann bestimmt werden durch $3 - 1 = 4\sigma \Leftrightarrow 0.5 = \sigma$.

$$\widehat{G}_{3\times3}(x,y) = \begin{pmatrix} G(-1,\ 1) & G(0,\ 1) & G(1,\ 1) \\ G(-1,\ 0) & G(0,\ 0) & G(1,\ 0) \\ G(-1,-1) & G(0,-1) & G(1,-1) \end{pmatrix} \approx \begin{pmatrix} 0.012 & 0.086 & 0.012 \\ 0.086 & 0.637 & 0.086 \\ 0.012 & 0.086 & 0.012 \end{pmatrix} \tag{3.3}$$

Die Anwendung des Gaußfilters auf ein perfekt glattes Bild, d.h. mit identischen Helligkeitswerten in allen Pixeln, muss wieder dasselbe Bild liefern, darf es also nicht verändern. Daher muss die Summe der Elemente des Faltungskerns 1 ergeben. Im berechneten Kern muss somit noch jeder Matrixeintrag durch die Summe aller Einträge dividiert werden.

Dadurch ergibt sich der endgültige 3×3-Gaußfilter

$$G_{3\times3}(x,y) \approx \begin{pmatrix} 0.011 & 0.084 & 0.011 \\ 0.084 & 0.619 & 0.084 \\ 0.011 & 0.084 & 0.011 \end{pmatrix}. \tag{3.4}$$

Eine zusätzliche Anforderung an die Faltung ist die gleichbleibende Größe des Bildes. Dies wird erreicht, indem die Faltung so berechnet wird, dass der mittlere Eintrag des Kerns immer mit einem Bildpunkt multipliziert werden muss. Zur Vereinfachung der Berechnung werden deshalb nur Faltungskerne mit ungerader Dimensionsgröße verwendet.

(a) Originalbild (b) Geglättetes Bild

Abbildung 3.1.: Beispielbild (1920x1280 Pixel) [CAT] vor und nach der Glättung mit einem Gauß-Filter der Größe 21×21

3.1.2. Partielle Ableitungen

Zur Bestimmung des Strukturtensors wird der Gradient des Bildes benötigt. Dessen Komponenten sind die partiellen Ableitungen des Bildes.

$$\nabla Im = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial Im}{\partial x} \\ \frac{\partial Im}{\partial y} \end{bmatrix} \tag{3.5}$$

Die partiellen Ableitungen des Bildes können dabei, wie auch die Glättung, durch Faltung berechnet werden. Dabei gibt es verschiedene Ansätze für die Wahl des Faltungskerns.

Intuitiver Operator

Ein einfacher und intuitiver Ansatz ist die Bestimmung der partiellen Ableitungen über finite Differenzen. So kann beispielsweise die Ableitung des Bildes in x-Richtung durch Faltung des Bildes mit dem Kern $\begin{bmatrix}-1 & 1\end{bmatrix}$ bestimmt werden. Es gilt also

$$\frac{\partial Im}{\partial x} = \begin{bmatrix}-1 & 1\end{bmatrix} * Im \tag{3.6}$$

und analog

$$\frac{\partial Im}{\partial y} = \begin{bmatrix}1 \\ -1\end{bmatrix} * Im. \tag{3.7}$$

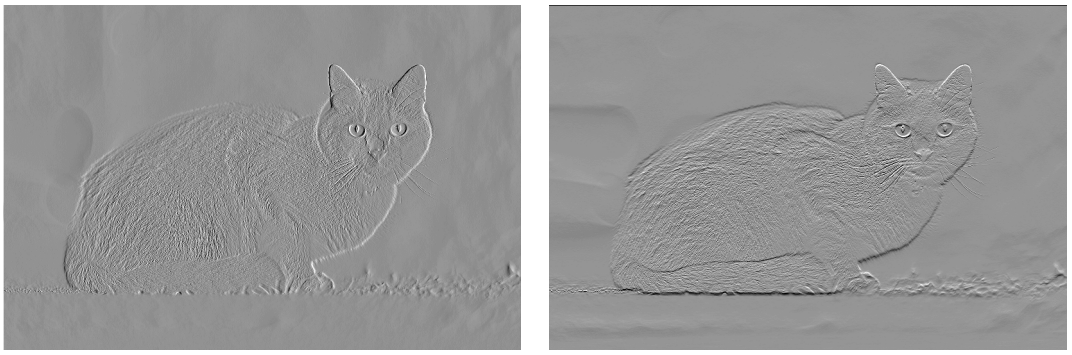
Bei Faltung mit diesen Kernen würde die Größe des Bildes um ein Pixel in jeder Dimension wachsen, d.h. das Bild würde um ein halbes Pixel verschoben. Um das zu vermeiden, faltet man das Bild mit den leicht modifizierten Kernen ungerader Größe (Abb. 3.2):

$$\frac{\partial Im}{\partial x} = \begin{bmatrix}-1 & 0 & 1\end{bmatrix} * Im \tag{3.8}$$

bzw.

$$\frac{\partial Im}{\partial y} = \begin{bmatrix}1 \\ 0 \\ -1\end{bmatrix} * Im \tag{3.9}$$

Die Koordinatenachsen verlaufen von links nach rechts (x-Koordinate) und von unten nach oben (y-Koordinate). Auch hier wird gefordert, dass das Ergebnisbild die gleiche Größe wie das Ursprungsbild hat (Abschnitt 3.1.1).



(a) Gradienten in x-Richtung (b) Gradienten in y-Richtung

Abbildung 3.2.: Gradienten des Beispielbildes (Abb. 3.1a), berechnet mit dem intuitiven Operator

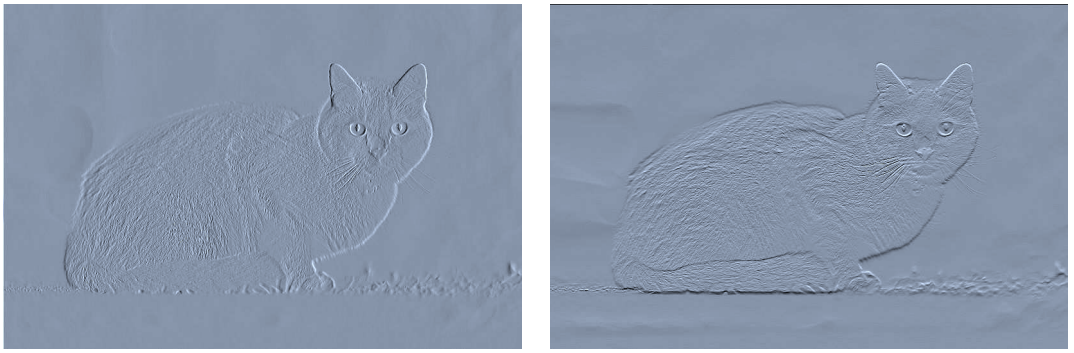
Prewitt-Operator

Als eine erste Erweiterung dieses sehr einfachen Operators wurde von Judith M. S. Prewitt der Prewitt-Operator entwickelt [JP70]. Er kombiniert die Gradientenbildung mit einer zusätzlichen Glättung über mehrere Pixel einer Nachbarschaft der jeweils anderen Dimension. Diese Glättung sieht man sehr deutlich, wenn man die Kerne aus einzelnen Komponenten konstruiert:

$$\frac{\partial Im}{\partial x} = \begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}-1 & 0 & 1\end{bmatrix} * Im = \begin{bmatrix}-1 & 0 & 1\\-1 & 0 & 1\\-1 & 0 & 1\end{bmatrix} * Im \tag{3.10}$$

bzw.

$$\frac{\partial Im}{\partial y} = \begin{bmatrix}1\\0\\-1\end{bmatrix}\begin{bmatrix}1 & 1 & 1\end{bmatrix} * Im = \begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix} * Im \tag{3.11}$$



(a) Gradienten in x-Richtung (b) Gradienten in y-Richtung

Abbildung 3.3.: Gradienten des Beispielbildes (Abb. 3.1a), berechnet mit dem Prewitt-Operator

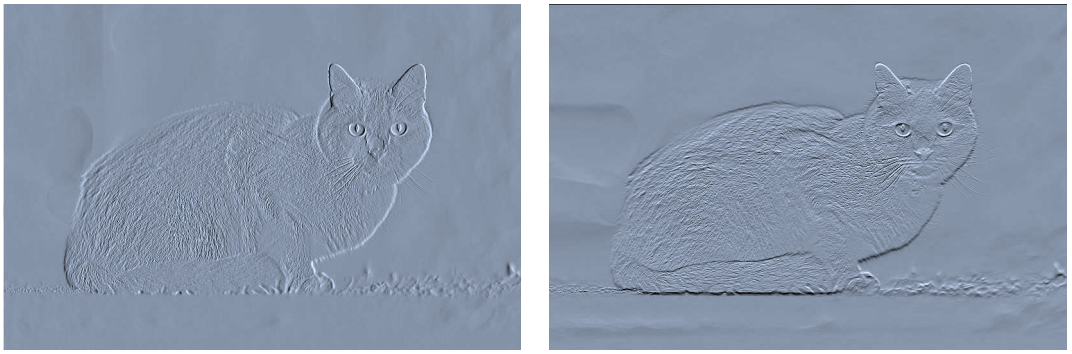
Sobel-Operator

Eine weiterer Kern zur Berechnung von Bildgradienten ist der Sobel(-Feldman)-Operator [SF68]. Benannt nach Irwin Sobel und Gary Feldman, gewichtet dieser Operator im Vergleich mit dem Prewitt-Operator die Punkte zusätzlich anhand ihres Abstandes zur Kernmitte. Auch hier wird dies am deutlichsten, wenn man die Kerne aus einzelnen Komponenten konstruiert:

$$\frac{\partial Im}{\partial x} = \begin{bmatrix}1\\2\\1\end{bmatrix}\begin{bmatrix}-1 & 0 & 1\end{bmatrix} * Im = \begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix} * Im \tag{3.12}$$

bzw.

$$\frac{\partial Im}{\partial y} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * Im = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * Im \tag{3.13}$$



(a) Gradienten in x-Richtung (b) Gradienten in y-Richtung

Abbildung 3.4.: Gradienten des Beispielbildes (Abb. 3.1a), berechnet mit dem Sobel-Operator

Vergleich

Vergleicht man die Gradientenbilder (Abb. 3.2, 3.3 und 3.4) der jeweiligen Richtungen miteinander, so sieht man mit dem bloßen Auge keinen Unterschied. Eine Betrachtung der Differenzbilder zeigt jedoch, dass der intuitive Operator im Vergleich mit beiden anderen Operatoren an sehr filigranen Stellen, wie etwa den Schnurrhaaren, leicht abweichende Gradienten liefert. Sobel- und Prewitt-Operator liefern nahezu das gleiche Ergebnis, das Differenzbild ist annähernd ein Nullbild.

Im weiteren Verlauf des Kapitels werden die Gradienten, die der Sobel-Operator liefert, für weitere Berechnungen verwendet. Dieser bietet im Hinblick auf später vergrößerte Faltungskerne das größte Potential durch seine Gewichtung des Abstandes zur Kernmitte.

3.1.3. Strukturtensor

Unter Zuhilfename der Vorüberlegungen zu Glättung und Gradientenberechnung ist es nun möglich, in jedem Bildpunkt einen Strukturtensor $S_\omega[x, y]$ zu berechnen. Dieser ist die gewichtete Summe aller dyadischen Produkte von Bildgradienten mit sich selbst in einer lokalen Nachbarschaft. Es gilt

$$S_\omega[x, y] = \sum_{i=-m}^{m} \sum_{j=-m}^{m} w(i, j) S_0[x - i, y - j], \tag{3.14}$$

wobei w eine Gewichtsfunktion und $S_0[x, y]$ das dyadische Produkt des Gradienten in einem Pixel mit sich selbst ist, d.h.

$$S_0[x,y] = \begin{bmatrix} \left(\frac{\partial Im}{\partial x}[x,y]\right)^2 & \frac{\partial Im}{\partial x}[x,y]\frac{\partial Im}{\partial y}[x,y] \\ \frac{\partial Im}{\partial x}[x,y]\frac{\partial Im}{\partial y}[x,y] & \left(\frac{\partial Im}{\partial y}[x,y]\right)^2 \end{bmatrix}. \tag{3.15}$$

Als Gewichtsfunktion wird auch hier der normierte Gaußkern genutzt (Abschnitt 3.1.1). Daraus ergibt sich sofort, dass die Summationsindices i und j nur den Bereich der Gewichtsfunktion abdecken müssen. Entsprechend der Umrechnung bei der Glättung gilt auch hier $2m + 1 = 4\sigma$.

Hat man den Strukturtensor erfolgreich bestimmt, lassen sich an ihm strukturelle Eigenschaften der lokalen Umgebung des Punktes (x, y) ablesen. Die Eigenwerte λ_1, λ_2 (mit $\lambda_1 \geq \lambda_2 \geq 0$) und zugehörigen Eigenvektoren v_1, v_2 des Strukturtensors stehen in direktem Bezug zur Verteilung der Gradienten. Gilt beispielsweise $\lambda_1 >> \lambda_2$, dann ist v_1 die Hauptrichtung der Gradienten innerhalb der lokalen Nachbarschaft. Gilt sogar $\lambda_2 = 0$, sind alle Gradienten Vielfache von v_1.

Andererseits existiert keine Gradientenhauptrichtung, wenn $\lambda_1 = \lambda_2$. Das ist beispielsweise bei komplett zufälliger Verteilung der Gradientenrichtungen der Fall. Ein Sonderfall ist dabei $\lambda_1 = \lambda_2 = 0$. Dieser tritt ausschließlich dann auf, wenn alle Gradienten Null sind, was bei konstanten Bildwerten innerhalb der lokalen Nachbarschaft auftritt.

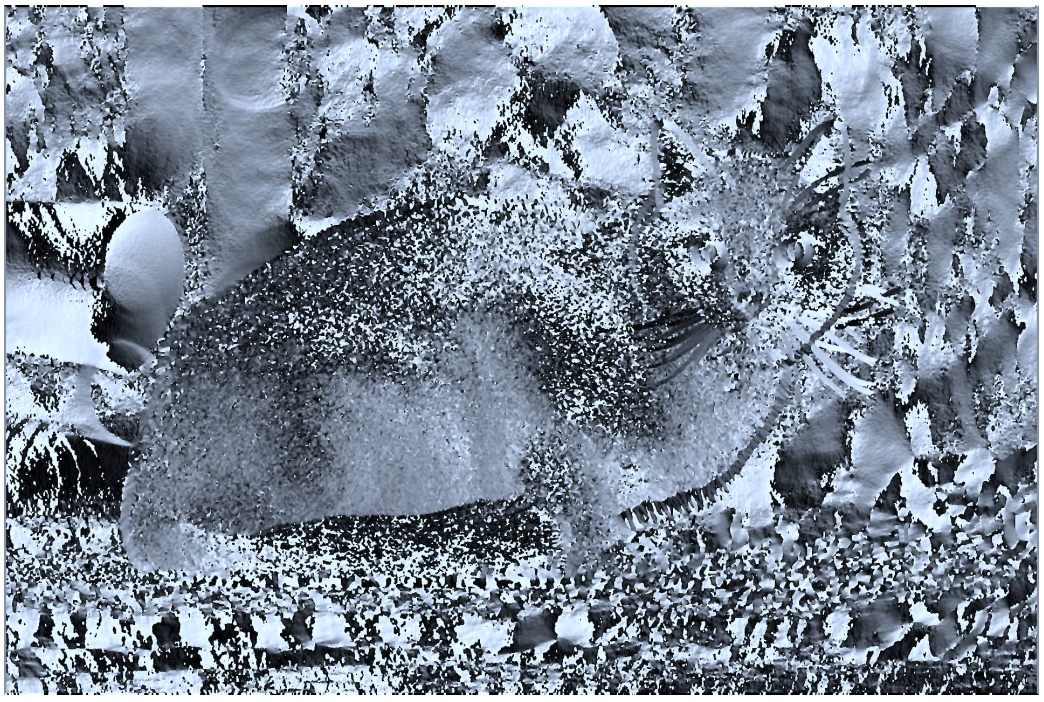


Abbildung 3.5.: Aus dem Strukturtensor extrahierte Richtungen der minimalen Gradienten. Schwarz entspricht einem Winkel von 0°, weiß entspricht einem Winkel von 180°. 0° bedeutet, dass der Gradient waagrecht verläuft. Bei 90° verläuft er senkrecht.

Zu der Richtung des kleinsten Gradienten im Strukturtensor (v_2) lässt sich ein entsprechender Winkel berechnen (Abb. 3.5). Dieser entspricht der Faserrichtung, da der minimale Gradient immer entlang einer Kante zeigt. Zusätzlich muss der Wertebereich auf $[0; 180]$ eingeschränkt werden (Formel 3.16), da die Gradienten sowohl in v_2, als auch in $-v_2$ minimal sind und beide Richtungen im PLI-Sinne identisch sind.

$$\theta = \arctan\left(\frac{g_y}{g_x}\right) \bmod 180 \tag{3.16}$$

Allerdings können bei dieser einfachen Formel Probleme auftreten [ATAN]. Der Quadrant, in dem der Winkel liegt, ist mit nur einem Argument nicht immer zweifelsfrei bestimmbar. Ein Beispiel hierfür sind die Fälle $g_x = 1, g_y = 1$ und $g_x = -1, g_y = -1$. Der Quotient ist in beiden Fällen 1, dementsprechend liefert arctan auch in beiden Fällen identische Ergebnisse. In Wahrheit liegt jedoch die erste Lösung im ersten Quadranten, die zweite Lösung im dritten Quadranten. Um diesem Problem entgegenzutreten, nutzt man die Funktion *arctan2* [ATAN2]. Diese bestimmt den Quadranten der Lösung über die Vorzeichen der beiden Eingabeparameter und beinhaltet dadurch eine zusätzliche Logik zur einfachen Umkehrfunktion des Tangens (Formel 3.17).

$$\theta = \arctan2\left(g_y, g_x\right) \bmod 180 \tag{3.17}$$

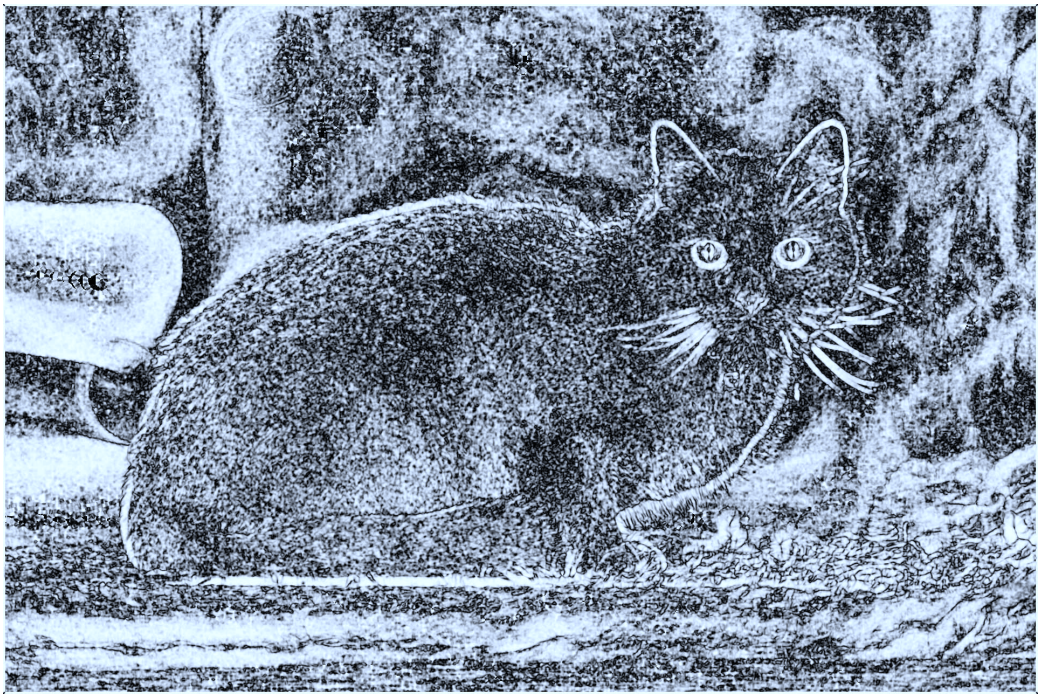


Abbildung 3.6.: Kohärenz der Strukturtensoren. Schwarz entspricht einem Kohärenzwert von 0, d.h. es gibt keine vorherrschende Gradientenrichtung. Weiß entspricht einem Wert von 1., d.h. die Gradienten sind alle gleich.

Eine weitere wichtige Eigenschaft des Strukturtensors ist, dass der Unterschied zwischen den beiden Eigenwerten als ein Maß für die Dominanz einer Richtung interpretiert werden kann (Abb. 3.6).

Dieser Kohärenz genannte Wert lässt sich berechnen als

$$c_{(x,y)} = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2 . \tag{3.18}$$

Er nimmt den Wert 1 an, wenn alle Gradienten in der Umgebung von (x, y) Vielfache des größten Eigenvektors sind. Gibt es keine vorherrschende Richtung, so ist die Kohärenz 0. Für den Fall eines konstanten Bildes (alle Gradienten gleich Null) ist der Wert nicht definiert.

Aus den vorhandenen Informationen lässt sich ein 2D-Farbbild erzeugen. Man wählt dazu den HSV-Farbraum [AS78], bei dem jedes Pixel die Eigenschaften *Hue*, *Saturation* und *Value* hat. *Hue* ist dabei ein Farbwert in Abhängigkeit vom Winkel, *Saturation* entspricht der Farbsättigung und *Value* stellt die Helligkeit des Pixels dar (Abb. 3.7).

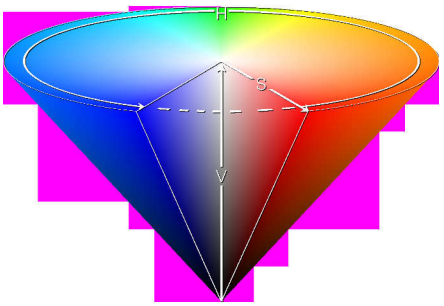


Abbildung 3.7.: HSV-Farbraum [HSV]

Wählt man nun den Winkel als *Hue*, die Kohärenz als *Saturation* und den Grauwert des Originalbildes als *Value*, so erhält man ein Bild mit farbcodierten Richtungen.

Um die auf diesem Bild vorhandenen Informationen zu verstärken, wird aus Sättigung und Helligkeit jeweils die Wurzel gezogen. Da der Wertebereich beider Werte das Intervall $[0; 1]$ ist, werden diese dadurch leicht erhöht.

(a) normale Codierung (b) angepasste Codierung

Abbildung 3.8.: Codierung der Winkel als Farben im HSV-Farbraum

In der üblichen Definition des HSV-Farbraums werden die Winkel von 0° bis 360° in Farben übertragen. Da für den gegebenen Anwendungsfall der Wertebereich jedoch auf $[0; 180]$ eingeschränkt ist, wäre eine Kodierung nach diesem Schema ungünstig. Der Übergang von 180° auf 0° wäre farblich sehr abrupt (türkis zu rot) und der eigentlich fließende Übergang der Richtungen würde farblich falsch wahrgenommen. Daher ist es sinnvoll, das gesamte Farbspektrum nur für die Abbildung des Intervalls $[0; 180]$ zu nutzen (Abb. 3.8).

Erzeugt man nun aus Direktion, Kohärenz und Originalbild das HSV-Farbbild, lassen sich die Richtungen der einzelnen Strukturen sehr schön ablesen (Abb. 3.9). Vor allem die Schnurrhaare der Katze machen das sehr deutlich.

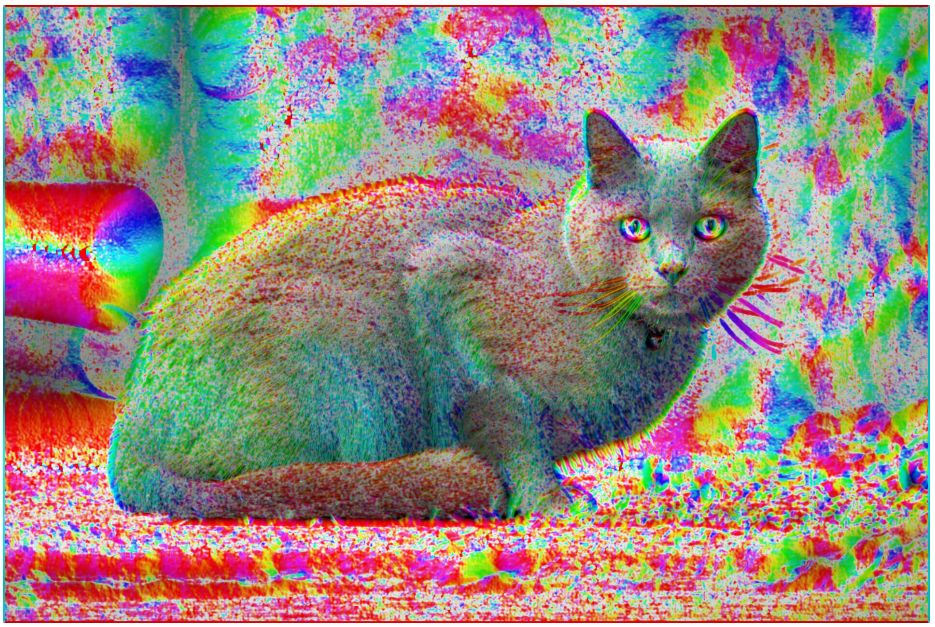


Abbildung 3.9.: Farbcodiertes Bild der Katze. Insbesondere an den Schnurrhaaren sind die Orientierungen sehr schön zu sehen.

3.2. 3D-STA

Um zusätzlich zu den Richtungen in einer Ebene auch räumliche Informationen aus Bildern extrahieren zu können, ist es nötig, die Strukturtensoranalyse auf drei Dimensionen zu erweitern.

3.2.1. Glättung

Im Dreidimensionalen soll die Glättung durch Faltung mit einem Gaußkern mit der Faltungsvorschrift

$$Im_{glatt}(x, y, z) = Im_{orig}(x, y, z) * G(x, y, z) \tag{3.19}$$

erfolgen. Als Faltungskern nutzt man in diesem Fall den 3D-Gaußkern

$$G(x, y, z) = \frac{1}{\sqrt{(2\pi)^3\sigma^3}} \cdot \mathrm{e}^{-\frac{x^2+y^2+z^2}{2\sigma^2}}. \tag{3.20}$$

Die Berechnung des diskreten Faltungskerns erfolgt analog zum 2D-Fall. Man wertet die Funktion $G(x, y, z)$ um den Mittelpunkt der Matrix herum aus. Beispielhaft wird der 3×3×3-Filter (d.h. $\sigma = 0.5$) berechnet.

$$\widehat{G}_{3\times3\times3}(x,y,z) = \begin{array}{c} \begin{bmatrix} G(-1,1,-1) & G(0,1,-1) & G(1,1,-1) \\ G(-1,0,-1) & G(0,0,-1) & G(1,0,-1) \end{bmatrix} \\ \begin{bmatrix} G(-1,1,0) & G(0,1,0) & G(1,1,0) \\ G(-1,0,0) & G(0,0,0) & G(1,0,0) \end{bmatrix}, -1) \\ \begin{bmatrix} G(-1,1,1) & G(0,1,1) & G(1,1,1) \\ G(-1,0,1) & G(0,0,1) & G(1,0,1) \\ G(-1,-1,1) & G(0,-1,1) & G(1,-1,1) \end{bmatrix}, 0) \end{array}$$

$$\approx \begin{array}{c} \begin{bmatrix} 0.001 & 0.009 & 0.001 \\ 0.009 & 0.069 & 0.009 \end{bmatrix} \\ \begin{bmatrix} 0.009 & 0.069 & 0.009 \\ 0.069 & 0.508 & 0.069 \end{bmatrix}, 0.001 \\ \begin{bmatrix} 0.001 & 0.009 & 0.001 \\ 0.009 & 0.069 & 0.009 \\ 0.001 & 0.009 & 0.001 \end{bmatrix}, 0.009 \end{array} \tag{3.21}$$

Auch der 3D-Kern darf ein perfekt glattes Bild nicht verändern. Daher muss auch hier die Summe der Einträge 1 ergeben. Normierung ergibt den endgültigen Kern

$$G_{3\times3\times3}(x,y,z) \approx \begin{array}{c} \begin{bmatrix} 0.001 & 0.009 & 0.001 \\ 0.009 & 0.066 & 0.009 \end{bmatrix} \\ \begin{bmatrix} 0.009 & 0.066 & 0.009 \\ 0.066 & 0.489 & 0.066 \end{bmatrix}, 0.001 \\ \begin{bmatrix} 0.001 & 0.009 & 0.001 \\ 0.009 & 0.066 & 0.009 \\ 0.001 & 0.009 & 0.001 \end{bmatrix}, 0.009 \end{array} \tag{3.22}$$

3.2.2. Partielle Ableitungen

Der für die Berechnung des Strukturtensors benötigte Gradient des Bildes erhält im dreidimensionalen Fall eine dritte Komponente, die partielle Ableitung in z-Richtung.

$$\nabla Im = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} \frac{\partial Im}{\partial x} \\ \frac{\partial Im}{\partial y} \\ \frac{\partial Im}{\partial z} \end{bmatrix} \tag{3.23}$$

Die einzelnen Komponenten werden durch Faltung berechnet. Die Kerne selbst müssen zum Teil angepasst werden.

Intuitiver Operator

Die auf finiten Differenzen basierenden Kerne des intuitiven Operators für die Ableitungen in x- bzw. y-Richtung ändern sich nicht. Man wählt auch hier

$$\frac{\partial Im}{\partial x} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * Im \tag{3.24}$$

bzw.

$$\frac{\partial Im}{\partial y} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * Im. \tag{3.25}$$

Analog erzeugt man einen Faltungskern, der dieselben finiten Differenzen in z-Richtung abbildet. Um Verwechslungen mit den anderen Kernen zu vermeiden, wird er hier diagonal von links unten nach rechts oben dargestellt.

$$\frac{\partial Im}{\partial z} = \begin{bmatrix} & & -1 \\ & 0 & \\ 1 & & \end{bmatrix} * Im \tag{3.26}$$

Prewitt-Operator

Bei der Erweiterung des Prewitt-Operators auf drei Dimensionen müssen die bereits vorhandenen Kerne für x- und y-Richtung angepasst werden. Diese Anpassung erfolgt durch die Vergrößerung der Nachbarschaft, über die gemittelt wird, in der dritten Dimension. Dann gilt:

$$\frac{\partial Im}{\partial x} = \begin{bmatrix} & & \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \\ & \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} & \begin{matrix} 0 & 1 \end{matrix} & \end{bmatrix} * Im \tag{3.27}$$

bzw.

$$\frac{\partial Im}{\partial y} = \begin{bmatrix} & & \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ & \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} & \begin{matrix} -1 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} & \begin{matrix} -1 \end{matrix} & \end{bmatrix} * Im \tag{3.28}$$

Analog lässt sich nun auch der Operator zur Bestimmung der partiellen Ableitung in z-Richtung bilden. Dieser lautet:

$$\frac{\partial Im}{\partial z} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ & & \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ & & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} *Im \tag{3.29}$$

Im Dreidimensionalen lässt sich der Kern ebenfalls als Kombination von Glättung und dem intuitiven Operator darstellen. So gilt etwa für die Ableitung in z-Richtung:

$$\frac{\partial Im}{\partial z} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} *Im = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} *Im \tag{3.30}$$

Sobel-Operator

Um die Gestalt des Sobel-Operators im Dreidimensionalen zu verstehen, schaut man sich hier zuerst die Zerlegung in die einzelnen Komponenten an. Diese ist ähnlich zum Prewitt-Operator, allerdings mit der Gewichtung der Punkte bei der Glättung.

$$\frac{\partial Im}{\partial x} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} *Im = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} *Im \tag{3.31}$$

$$\frac{\partial Im}{\partial y} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} *Im = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} *Im \tag{3.32}$$

$$\frac{\partial Im}{\partial z} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} *Im = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} *Im \tag{3.33}$$

Die Kerne selbst haben dann folgende Gestalt:

Gradient in x-Richtung

$$\frac{\partial Im}{\partial x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2 & 0 & 2 \\ -4 & 0 & 4 \\ -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * Im \tag{3.34}$$

Gradient in y-Richtung

$$\frac{\partial Im}{\partial y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 0 & 0 \\ -2 & -4 & -2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * Im \tag{3.35}$$

Gradient in z-Richtung

$$\frac{\partial Im}{\partial z} = \begin{bmatrix} -1 & -2 & -1 \\ -2 & -4 & -2 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * Im \tag{3.36}$$

3.2.3. Strukturtensor

Mit den notwendigen Überlegungen zur Übertragung von Glättung und Gradientenberechnung ins Dreidimensionale ist es möglich, in jedem Bildpunkt einen 3D-Strukturtensor zu berechnen. Dieser ist wie auch im zweidimensionalen Fall die gewichtete Summe aller dyadischen Produkte von Bildgradienten mit sich selbst in einer lokalen Nachbarschaft. Es gilt

$$S_\omega[x, y, z] = \sum_{i=-m}^{m} \sum_{j=-m}^{m} \sum_{k=-m}^{m} w(i, j, k) S_0[x - i, y - j, z - k], \tag{3.37}$$

wobei w eine Gewichtsfunktion ist und $S_0[x, y, z]$ das äußere Produkt des Gradienten in einem Pixel, d.h.

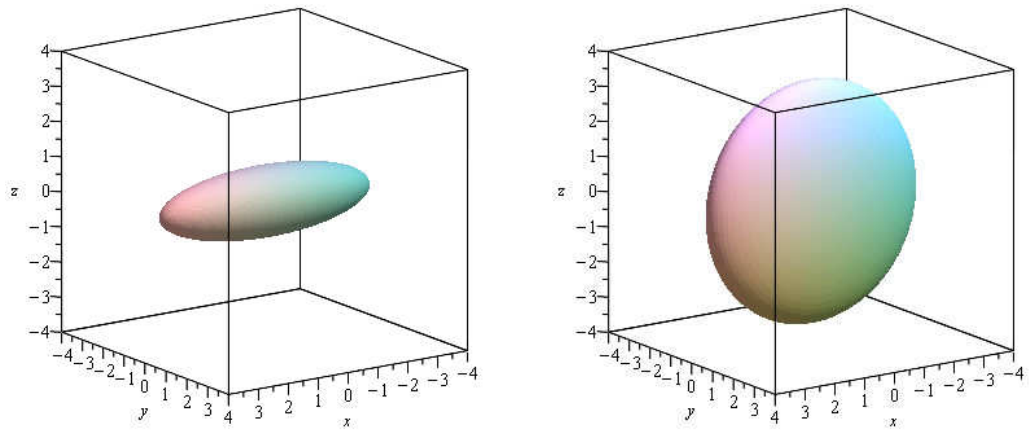
$$S_0[x,y,z] = \begin{bmatrix} \left(\frac{\partial Im}{\partial x}[x,y,z]\right)^2 & \frac{\partial Im}{\partial x}[x,y,z]\frac{\partial Im}{\partial y}[x,y,z] & \frac{\partial Im}{\partial x}[x,y,z]\frac{\partial Im}{\partial z}[x,y,z] \\ \frac{\partial Im}{\partial x}[x,y,z]\frac{\partial Im}{\partial y}[x,y,z] & \left(\frac{\partial Im}{\partial y}[x,y,z]\right)^2 & \frac{\partial Im}{\partial y}[x,y,z]\frac{\partial Im}{\partial z}[x,y,z] \\ \frac{\partial Im}{\partial x}[x,y,z]\frac{\partial Im}{\partial z}[x,y,z] & \frac{\partial Im}{\partial y}[x,y,z]\frac{\partial Im}{\partial z}[x,y,z] & \left(\frac{\partial Im}{\partial z}[x,y,z]\right)^2 \end{bmatrix} \tag{3.38}$$

Als Gewichtsfunktion wird wieder der normierte Gaußkern genutzt. Die Summationsindices i, j und k müssen dabei wie im Zweidimensionalen ausschließlich den Bereich der Gewichtsfunktion abdecken. Daher gilt auch im dreidimensionalen Fall die Formel $2m + 1 = 4\sigma$.

Aufgrund der zusätzlichen Dimension gibt es bei der Interpretation des Strukturtensors mehr Möglichkeiten, da die Matrix nun drei Eigenwerte ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) und die dazugehörigen Eigenvektoren (v_1, v_2, v_3) hat. Aus ihren Größenverhältnissen lassen sich, wie im Zweidimensionalen, Informationen über die Verteilung der Gradienten ableiten.

Gilt etwa $\lambda_1 >> \lambda_2 \approx \lambda_3$, dann ist v_1 die Gradientenhauptrichtung. Der Tensor bzw. das von den um die Eigenwerte skalierten Eigenvektoren aufgespannte Ellipsoid ist in diesem Fall zigarrenförmig (Abb. 3.10a). Für $\lambda_2 = \lambda_3 = 0$ sind sogar alle Gradienten Vielfache von v_1.

Für $\lambda_1 \approx \lambda_2 >> \lambda_3$ ist v_3 die Richtung mit dem kleinsten Gradienten innerhalb der lokalen Nachbarschaft. Im Raum, der von v_1 und v_2 aufgespannt wird, sind die Gradienten wiederum am größten. Der Tensor selbst ist in diesem Fall linsenförmig (Abb. 3.10b).



(a) Zigarrenförmiger Tensor (b) Linsenförmiger Tensor

Abbildung 3.10.: Zigarrenförmiger Tensor und linsenförmiger Tensor

Andererseits existiert keine Gradientenhauptrichtung, wenn alle Eigenwerte annähernd gleich sind, d.h. $\lambda_1 \approx \lambda_2 \approx \lambda_3$. Das ist bei komplett zufälliger Verteilung der Gradientenrichtungen in der betrachteten lokalen Nachbarschaft der Fall. Der Tensor selbst hat in diesem Fall die Form einer Kugel (Abb. 3.11). Ein wichtiger Sonderfall ist dabei $\lambda_1 = \lambda_2 = \lambda_3 = 0$. Dieser tritt ausschließlich dann auf, wenn die Bildwerte der lokalen Nachbarschaft konstant und somit alle Gradienten Null sind. Die Kugel hat dann den Radius Null.

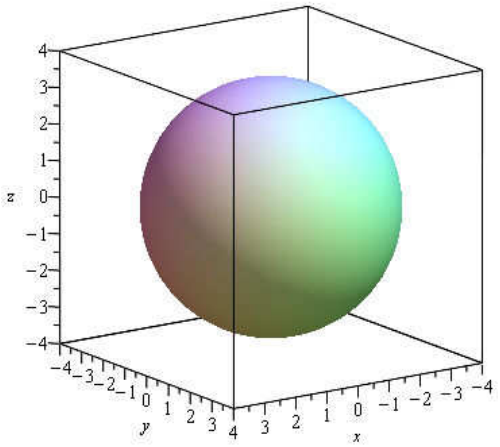


Abbildung 3.11.: Kugelförmiger Tensor

Hat man den Vektor in Richtung des kleinsten Gradienten bestimmt, lässt sich dessen Richtung in die sphärische Darstellung des Vektors umrechnen.

Für den Winkel in der Ebene, die Direktion θ, gilt wie im 2D-Fall die Formel

$$\theta = \arctan2\,(g_y, g_x) \bmod 180. \tag{3.39}$$

Der Winkel des Vektors zur Ebene, die Inklination α, lässt sich durch Formel 3.40 berechnen. Dabei ist es wichtig, dass im Vergleich zur Umrechnung von sphärischen in kartesische Koordinaten der Arkussinus benutzt wird, da bei PLI 0° Inklination eine Faser in der Ebene repräsentiert, während 0° in sphärischen Koordinaten eine senkrecht zur Ebene laufende Faser darstellt.

$$\alpha = \arcsin\,(g_z) \tag{3.40}$$

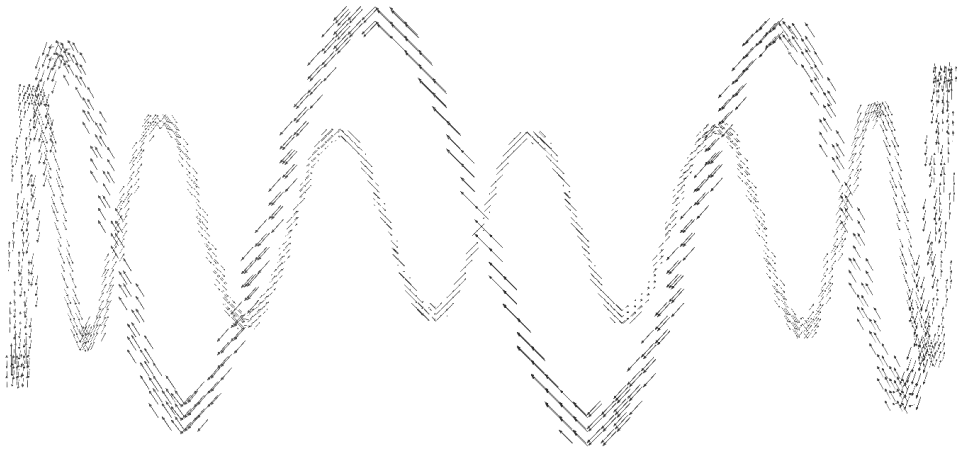


Abbildung 3.12.: Auf einer Kreisbahn verlaufende Sinuskurve (Seitenansicht)

Das Testbild für den dreidimensionalen Fall zeigt Wellen, die auf Kreisbahnen sinusförmig durch die z-Ebenen laufen (Abb. 3.12). Für viele, auf konzentrischen Kreisen laufende Kurven lässt sich ein 3D-Bild erzeugen. Unterteilt man dieses Bild künstlich in Schnitte und färbt ausschließlich die Pixel weiß, die eine Welle enthalten, erhält man den Testdatensatz für die 3D-STA (Abb. 3.13). Gezeigt wird der Schnitt, der die Ruhelage der Sinuskurven enthält, weshalb die Abstände der weißen Pixel innerhalb der Kreisbahnen äquidistant sind.

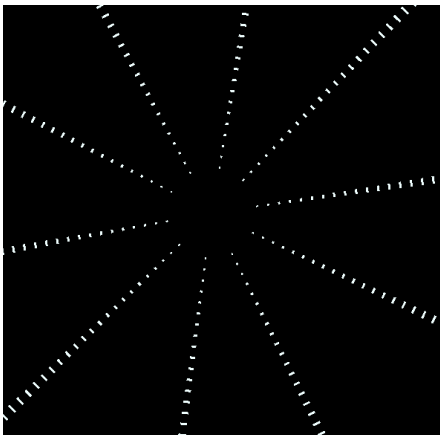
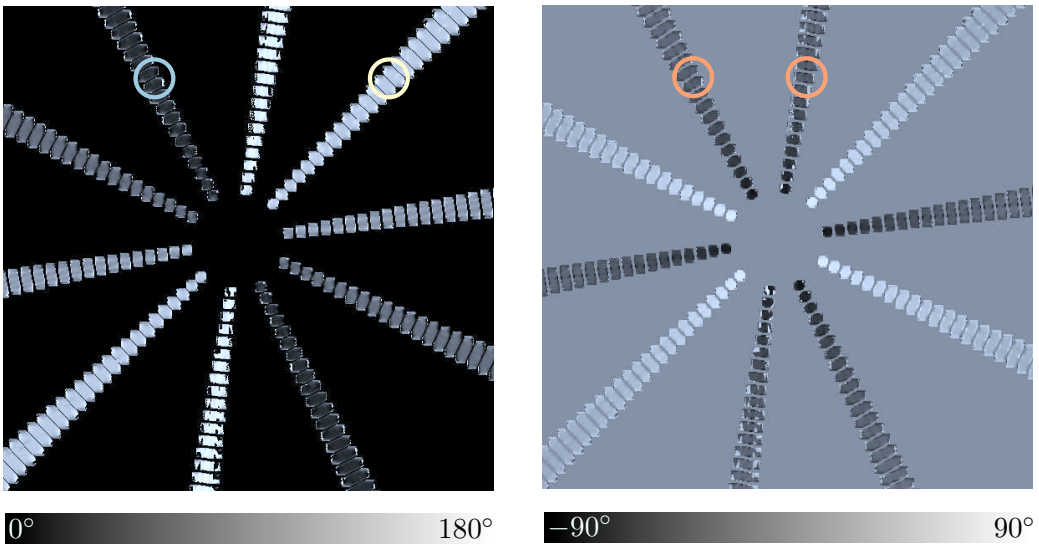


Abbildung 3.13.: 3D-Testbild

Die für das Testbild berechneten Winkel decken sich sehr gut mit den zu erwartenden Richtungen. Betrachtet man das Direktionsbild, so ist der Kreis als Träger klar erkennbar. Die Winkel liegen immer in den erwarteten Bereichen, wie etwa bei 25° (Abb. 3.14a, blauer Kreis) oder 135° (Abb. 3.14a, gelber Kreis).

Auch das Inklinationsbild (Abb. 3.14b) passt sehr gut. Der Grauwert des Hintergrundes bedeutet eine Neigung von 0°. Dunklere Pixel stehen für Fasern, die in die Ebene hineinzeigen, hellere für aus der Ebene herauszeigende Fasern. Ebenso werden von innen nach außen die Winkel flacher, da die Sinuskurven die gleiche Periodenzahl auf einer längeren Strecke zurücklegen. Erkennbar ist dies an einer sich grau annähernden Färbung.



(a) Direktion des Testbildes

(b) Inklination des Testbildes

Abbildung 3.14.: Berechnete Direktion und Inklination des Testbildes

Die auf den ersten Blick unlogische Abfolge von Inklinationswinkeln im Verlauf einer Welle, bei der manchmal zwei negative oder zwei positive Winkel in Folge auftreten (Abb. 3.14b, orange Kreise), kann durch die Einschränkung des Direktionswertebereichs auf $[0; 180]$ erklärt werden. Liegt ein Vektor außerhalb des Wertebereiches, so kann man nicht mehr wie im 2D-Fall einfach modulo 180 rechnen. Im 3D-Fall würde dies nicht einer Invertierung seiner Richtung entsprechen. Zusätzlich muss dafür auch das Vorzeichen der Inklination umgeändert werden.

Die Kohärenz könnte man wie im 2D-Fall berechnen, wenn man argumentiert, einzig und allein der relative Unterschied zwischen den beiden kleinsten Eigenwerten spielt eine Rolle. Allerdings würden die Fälle $\lambda_1 \approx \lambda_2 >> \lambda_3$ und $\lambda_1 >> \lambda_2 >> \lambda_3$ dabei gleich bewertet. Der erste Fall, der linsenförmige Tensor, ist der Optimalfall. Dort ist die Richtung des kleinsten Gradienten eindeutig. Der zweite Fall sollte nicht dieselbe Bewertung erhalten, da dort die Richtung nicht so klar bestimmt ist.

Deshalb wird die Kohärenzformel

$$c_{(x,y)} = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 \tag{3.41}$$

abgeändert zu

$$c_{(x,y)} = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 - \frac{1}{2} \cdot \left(\frac{\lambda_2 - \lambda_3}{\lambda_2 + \lambda_3}\right)^2 . \tag{3.42}$$

Die Änderung der Formel bewirkt, dass der Kohärenzwert kleiner wird, wenn sich die beiden größten Eigenwerte zu stark unterscheiden. Allerdings ist für den Fall $\lambda_2 >> \lambda_3$ garantiert, dass der Kohärenzwert etwa 0.5 nicht unterschreiten kann.

Auch aus dreidimensionalen Richtungsinformationen lässt sich ein Farbbild erzeugen. Hierbei ist es nicht mehr möglich, die in Abschnitt 3.1.3 beschriebene Umwandlung in HSV-Werte vorzunehmen, da hierbei keine Möglichkeit besteht, die Inklination α mit abzubilden.

Um aus dem Farbbild Informationen über die Inklination α ablesen zu können, lässt man diese in die Berechnung von Sättigung und Helligkeitswert eines Pixels mit einfließen. Dabei muss man folgende Abhängigkeiten berücksichtigen:

Je steiler eine Faser verläuft, desto geringer soll die Farbsättigung sein. Je flacher eine Faser verläuft, desto geringer soll der Helligkeitswert sein. Die Abhängigkeiten hierbei sind quadratisch, um die Häufigkeit des Vorkommens sehr dunkler bzw. farbarmer Fasern zu minimieren.

Die zusätzlich zur Farbtonberechnung aus der Direktion (Abb. 3.8b) aus diesen Anforderungen resultierenden Formeln zur Berechnung lauten:

$$Sat = 1 - \left(\frac{\alpha}{90}\right)^2 \tag{3.43}$$

$$Val = 1 - \cos(\alpha)^2 \tag{3.44}$$

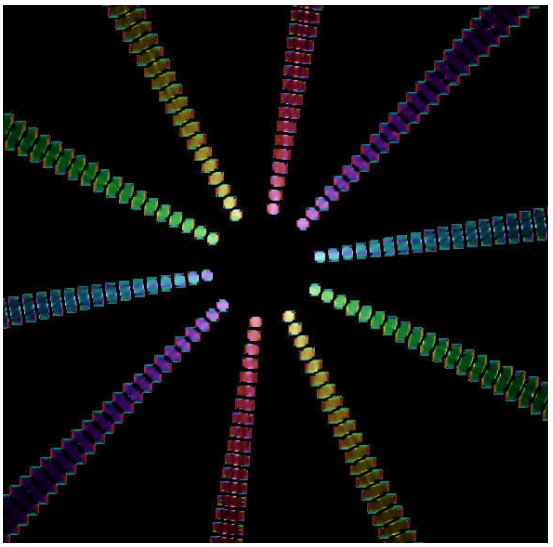


Abbildung 3.15.: Farbbild im HSV-Farbraum mit Anpassung von Sättigung und Helligkeit an die Inklination

Erzeugt man nach diesen Vorschriften ein Farbbild (Abb. 3.15), lassen sich die geforderten Eigenschaften von Helligkeit und Farbsättigung der Pixel im Hinblick auf das Inklinationsbild (Abb. 3.14b) gut erkennen.

4. Anwendung der STA auf PLI-Bilder

PLI-Bilder weisen im Gegensatz zu *normalen* Grauwertbildern einige Besonderheiten auf, die bei der Anwendung der STA beachtet werden müssen.

Der erste relevante Punkt dabei ist die Wahl des Bildes, das verarbeitet werden soll. Die PLI-Rohdaten bestehen aus 18 Bildern pro Kachel, noch nicht zu Schnitten zusammengefügt. Auf diese ist die STA nicht anwendbar, da nicht klar ist, welches der 18 Bilder die beste Grundlage bietet und wie die Bilder nachher zusammengefügt werden. Daher wählt man als Basisbild das gestitchte Retardierungsbild. Dieses hat die Eigenschaft, bereits sehr glatt zu sein.

Die im weiteren Verlauf dieser Arbeit genutzten Bilder stammen alle vom Datensatz Vervet1818, PLI-Aufnahmen des Hirns einer 2.4 Jahre alten Südlichen Grünmeerkatze, einer Primatenart. Die Schnittebene ist frontal. Die Schnittdicke beträgt 60 µm.

4.1. Glättung

Da PLI-Bilder nicht in jede Richtung äquidistante Auflösungen gewährleisten und die Registrierung der Schnitte nicht immer zufriedenstellend ist, muss eine Glättung durchgeführt werden, die genau diese Tatsachen berücksichtigt, d.h. in manche Richtungen stärker glättet. Sinnvoll ist hier die Abwandlung des Gaußkerns zu einem elliptischen Gaußkern durch unterschiedliche Skalierung in den einzelnen Dimensionen. Die Formeln lauten dann

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \cdot \mathrm{e}^{-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}}, \tag{4.1}$$

bzw. im Dreidimensionalen

$$G(x,y,z) = \frac{1}{\sqrt{(2\pi)^3}\sigma_x\sigma_y\sigma_z} \cdot \mathrm{e}^{-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}-\frac{z^2}{2\sigma_z^2}}. \tag{4.2}$$

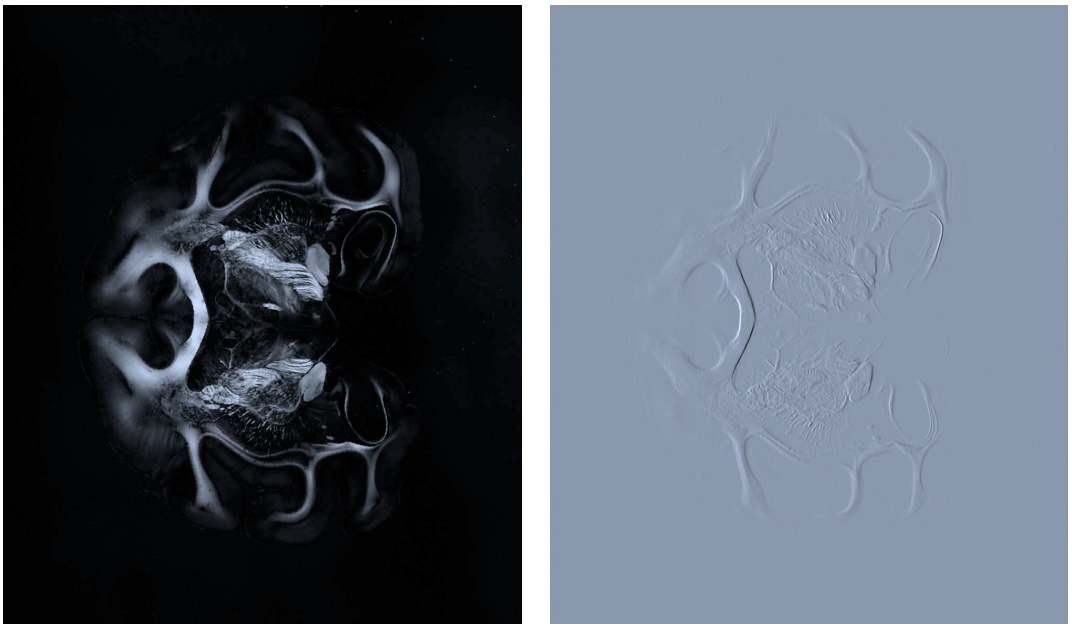
Um die Berechnungen von Formel 4.2 weiter zu vereinfachen, kann der konstante Vorfaktor der Exponentialfunktion weggelassen werden. Die dadurch entstehende falsche Skalierung wird bei der anschließenden Normierung des Faltungskerns wieder ausgeglichen.

Wertet man nun $G(x,y) = \mathrm{e}^{-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}}$ aus, so erhält man mit $\sigma_x = 1$ und $\sigma_y = 0.5$ den 2D-Gaußkern der Größe 3×5.

$$
\begin{aligned}
G_{3\times5}(x,y) &= \begin{pmatrix} G(-2,\ 1) & G(-1,\ 1) & G(0,\ 1) & G(1,\ 1) & G(2,\ 1) \\ G(-2,\ 0) & G(-1,\ 0) & G(0,\ 0) & G(1,\ 0) & G(1,\ 0) \\ G(-2,-1) & G(-1,-1) & G(0,-1) & G(1,-1) & G(2,-1) \end{pmatrix} \cdot \frac{1}{||\dots||} \\
&\approx \begin{pmatrix} 0.018 & 0.082 & 0.135 & 0.082 & 0.018 \\ 0.135 & 0.607 & 1.000 & 0.607 & 0.135 \\ 0.018 & 0.082 & 0.135 & 0.082 & 0.018 \end{pmatrix} \cdot \frac{1}{||\dots||} \\
&\approx \begin{pmatrix} 0.006 & 0.026 & 0.043 & 0.026 & 0.006 \\ 0.043 & 0.192 & 0.317 & 0.192 & 0.043 \\ 0.006 & 0.026 & 0.043 & 0.026 & 0.006 \end{pmatrix}
\end{aligned}
\tag{4.3}
$$

4.2. Gradienten

Die in dieser Arbeit vorgestellten Faltungskerne zur Gradientenbestimmung sind in ihrer Größe beschränkt. So hat etwa der intuitive Operator immer die Größe 3×1 und der Sobel- bzw. Prewitt-Operator sind auf eine Größe von 3×3×3 limitiert. Die durch diese Faltungskerne gefundenen Strukturen sind teils sehr schwach ausgeprägt.



(a) Retardierungsbild (b) Gradient in x-Richtung

Abbildung 4.1.: Retardierungsbild und Gradient in x-Richtung des LAP-Bildes. Es wurde der Prewitt-Operator der Größe 3×3×3 benutzt.

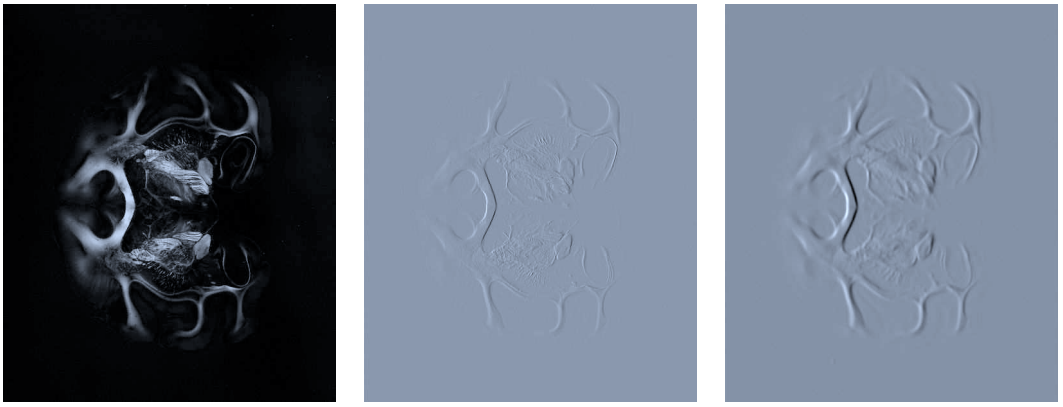
Um dieses Problem zu lösen, wurden Erweiterungen von Sobel- und Prewitt-Operator entwickelt, deren Dimensionsgrößen grundsätzlich nicht beschränkt sind.

Prewitt-Operator

Wie in Abschnitt 3.1.2 beschrieben, kombiniert der Prewitt-Operator den intuitiven Operator mit einer Glättung über eine Nachbarschaft. Diese ursprünglich ausschließlich auf die direkten Nachbarn des Pixels beschränkte Umgebung lässt sich beliebig vergrößern [SU05]. So lautet beispielsweise der 5×5-Faltungskern für Gradientenbestimmung in x-Richtung:

$$\frac{\partial Im}{\partial x} = \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix} * Im \tag{4.4}$$

Der dreidimensionale 5×5×5-Faltungskern für Gradientenbestimmung in x-Richtung entspricht dem oben stehenden 2D-Kern, fünfmal hintereinander gelegt. Alle anderen Richtungen erhält man analog durch Auffüllen der dritten Dimension mit dem bereits vorhandenen Faltungskern.



(a) Retardierungsbild (b) 3×3×3 (c) 11×11×11

Abbildung 4.2.: Retardierungsbild und Gradient in x-Richtung des LAP-Bildes. Gradientenbestimmung mit Prewitt-Operatoren der Größen 3×3×3 und 11×11×11.

Man erkennt deutlich (Abb. 4.2), dass vorhandene Strukturen stärker hervorgehoben werden. Andererseits werden die Strukturen aufgeweicht und Feinheiten gehen verloren.

Sobel-Operator

Auch der Sobel-Operator lässt sich auf verschiedene Dimensionsgrößen erweitern. Da er im Vergleich zum Prewitt-Operator die Entfernung eines Punktes zur Kernmitte in die Gewichtung einbezieht, ist die Vergrößerung aufwendiger.

Im Zweidimensionalen Fall lassen sich alle Operatoren aus dem 3×3-Faltungskern berechnen. Der 5×5-Faltungskern wird dabei als Faltung des 3×3-Kerns mit einem 3×3-Gewichtungskern bestimmt. Dieser Gewichtungskern wiederum ist das äußere Produkt des Glättungsteils des Sobeloperators mit sich selbst.

$$\begin{aligned}\frac{\partial Im}{\partial x} &= \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} \begin{bmatrix}1 & 2 & 1\end{bmatrix} * Im \\ &= \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} * Im \\ &= \begin{bmatrix}-1 & -2 & 0 & 2 & 1\\ -4 & -8 & 0 & 8 & 4\\ -6 & -12 & 0 & 12 & 6\\ -4 & -8 & 0 & 8 & 4\\ -1 & -2 & 0 & 2 & 1\end{bmatrix} * Im\end{aligned} \tag{4.5}$$

Dieser Kern entspricht dabei dem von Intel in seinen Bibliotheken zur Verfügung gestellten Sobel-Operator der Größe 5×5 [S5x5]. Größere Kerne wie beispielsweise 7×7 oder 9×9 erhält man durch die wiederholte Faltung des nächstkleineren Faltungskerns mit

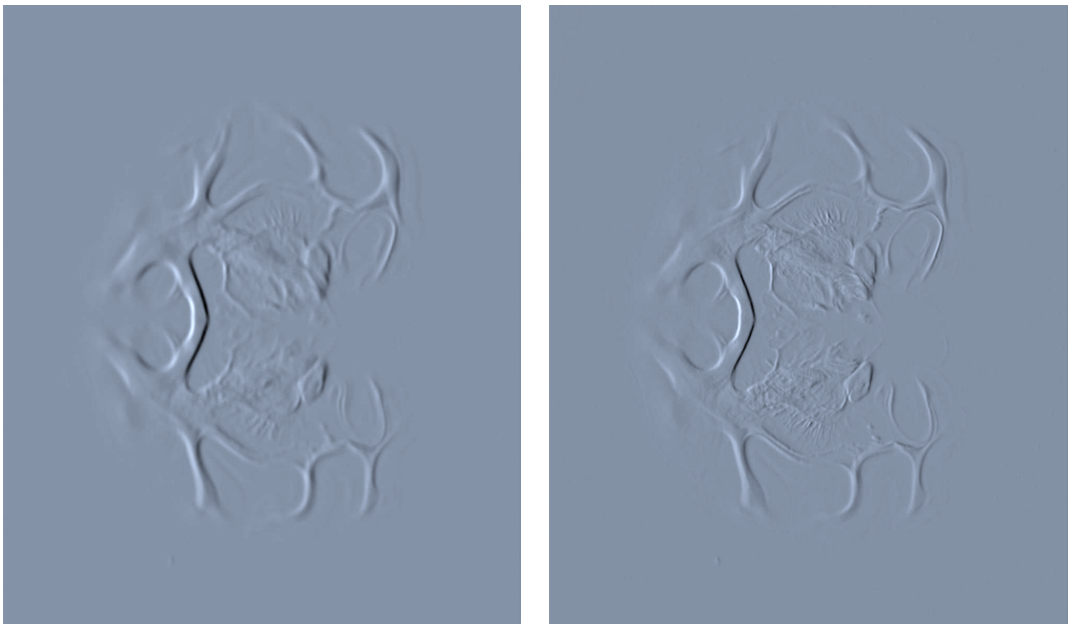
$$\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \tag{4.6}$$

Beliebig große dreidimensionale Faltungskerne lassen sich auf gleiche Art und Weise erzeugen. Basis ist in diesem Fall der 3×3×3-Faltungskern, gefaltet wird mit dem dreifachen äußeren Produkt des Glättungsteils mit sich selbst.

$$\frac{\partial Im}{\partial x} = \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} \begin{bmatrix}1 & 2 & 1\end{bmatrix} \begin{bmatrix}1\\2\\1\end{bmatrix} * Im$$

$$= \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} * Im \tag{4.7}$$

$$
= \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}
\begin{bmatrix} -4 & -8 & 0 & 8 & 4 \\ -16 & -32 & 0 & 32 & 16 \\ -24 & -48 & 0 & 48 & 24 \end{bmatrix}
\begin{bmatrix} -6 & -12 & 0 & 12 & 6 \\ -24 & -48 & 0 & 48 & 24 \\ -36 & -72 & 0 & 72 & 36 \end{bmatrix}
\begin{bmatrix} -4 & -8 & 0 & 8 & 4 \\ -16 & -32 & 0 & 32 & 16 \\ -24 & -48 & 0 & 48 & 24 \end{bmatrix}
\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}
* Im
$$



(a) Prewitt 11×11×11 (b) Sobel 11×11×11

Abbildung 4.3.: Gradientenbilder des LAP-Bildes in x-Richtung. Berechnung mit Prewitt- und Sobel-Operatoren der Größe 11×11×11

Während Sobel- und Prewitt-Operator bei einer Größe von 3×3 nahezu identische Ergebnisse geliefert haben, sieht man bei Vergrößerung der Faltungskerne deutliche Unterschiede (Abb. 4.3). Durch die vom Abstand zur Kernmitte abhängige Gewichtung gehen beim Sobel-Operator weniger Details verloren als beim Prewitt-Operator glei-

cher Größe. Vorhandene Strukturen werden durch den Sobel-Operator jedoch allgemein nicht ganz so stark hervorgehoben.

Als sehr guten Kompromiss aus dem allgemein stärkeren Hervorheben von Strukturen und dem gleichzeitigen Erhalt feiner Strukturen wählt man bei größeren Faltungskernen daher den Sobel-Operator. Insbesondere der Verlust von Strukturinformationen ist im Hinblick auf die Weiterverarbeitung nicht vertretbar.

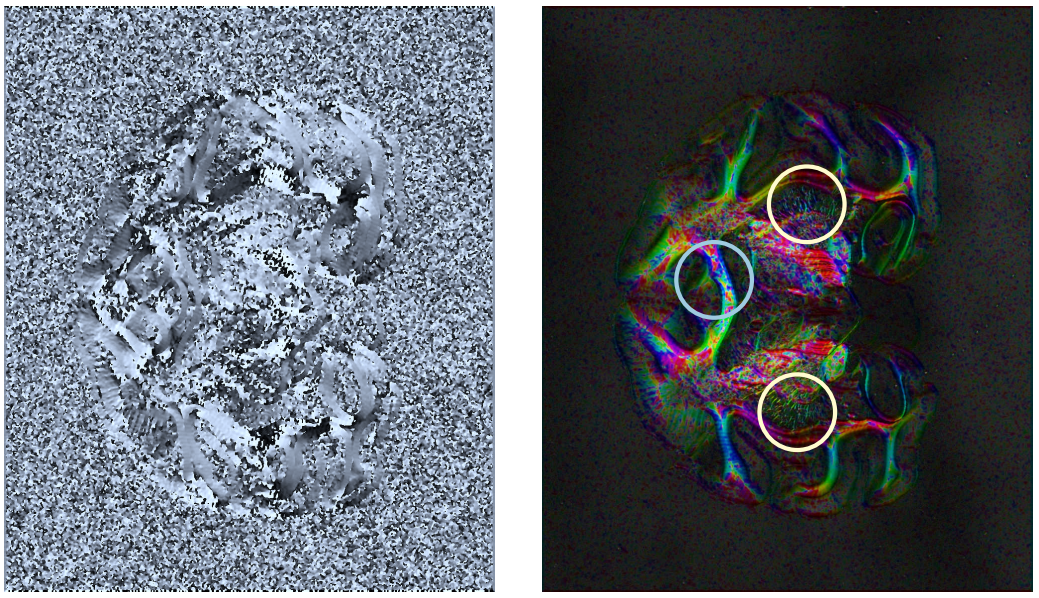
4.3. Bilder

In einem letzten Schritt kann nun die vollständige Strukturtensoranalyse auf PLI-Bilder angewendet werden. Um Fehlerquellen und Probleme frühzeitig identifizieren zu können, wird immer erst eine 2D-STA durchgeführt und im Anschluss die 3D-STA. Ebenso wird die Komplexität der Bilder Stück für Stück gesteigert.

4.3.1. LAP

2D

Erstes Anwendungsbeispiel ist die 2D-STA des bereits in Abschnitt 4.2 genutzten LAP-Bildes mit 3×3-Glättungskern, 5×5-Gradientenkern und 7×7-Tensorkern (Abb. 4.4).

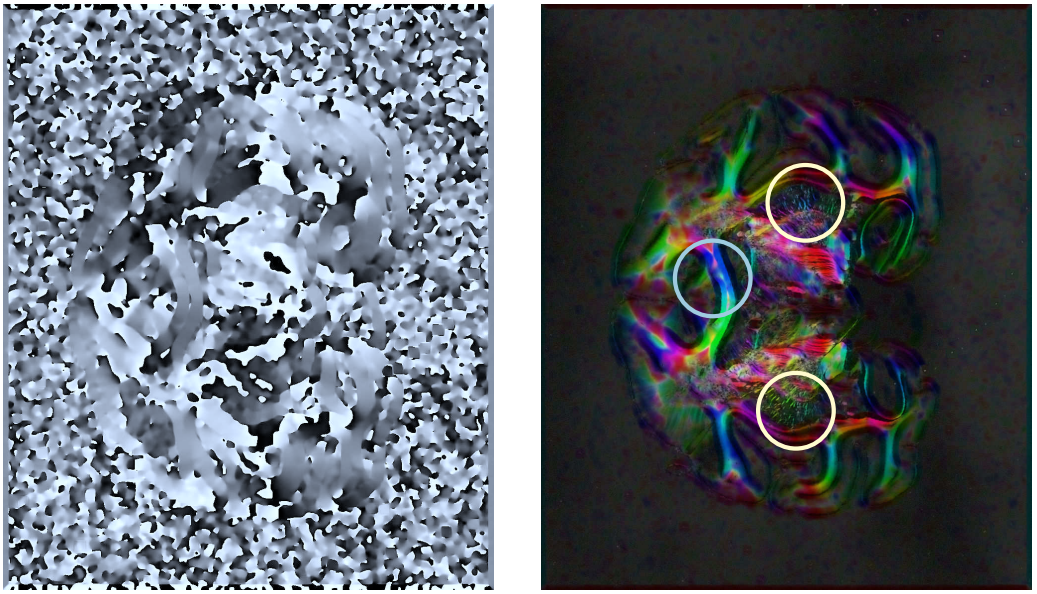


(a) Direktionsbild (b) RGB-Bild

Abbildung 4.4.: Direktions- und RGB-Bild nach Anwendung der 2D-Strukturtensoranalyse auf das LAP-Bild. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 7×7-Tensorkern. Gute Erkennung feiner Strukturen (gelbe Kreise), Probleme bei groben Strukturen (blauer Kreis).

Man sieht dabei insbesondere im RGB-Bild deutlich, dass sehr feine Strukturen bei dieser Wahl der Parameter gut erkannt werden (gelbe Kreise), grobe Strukturen jedoch vor allem in der Mitte der Struktur uneinheitliche Richtungsinformationen aufweisen (blauer Kreis). Grund hierfür ist das Verhältnis aus Größe des Tensorkerns und der Größe der zu erkennenden Strukturen.

Da in der Mitte einer (annähernd homogenen) Struktur kaum Gradienten vorliegen, muss der Tensorkern so groß sein, dass er bei Platzierung auf der Mitte der Struktur ihre Ränder noch enthält. Andernfalls werden aus den minimalen Gradienten in der Mitte der Struktur nahezu zufällige Richtungen berechnet. Diese eingestreuten falschen Richtungen sind auch im Direktionsbild sehr gut zu sehen. Es wirkt zum Teil sehr verrauscht.

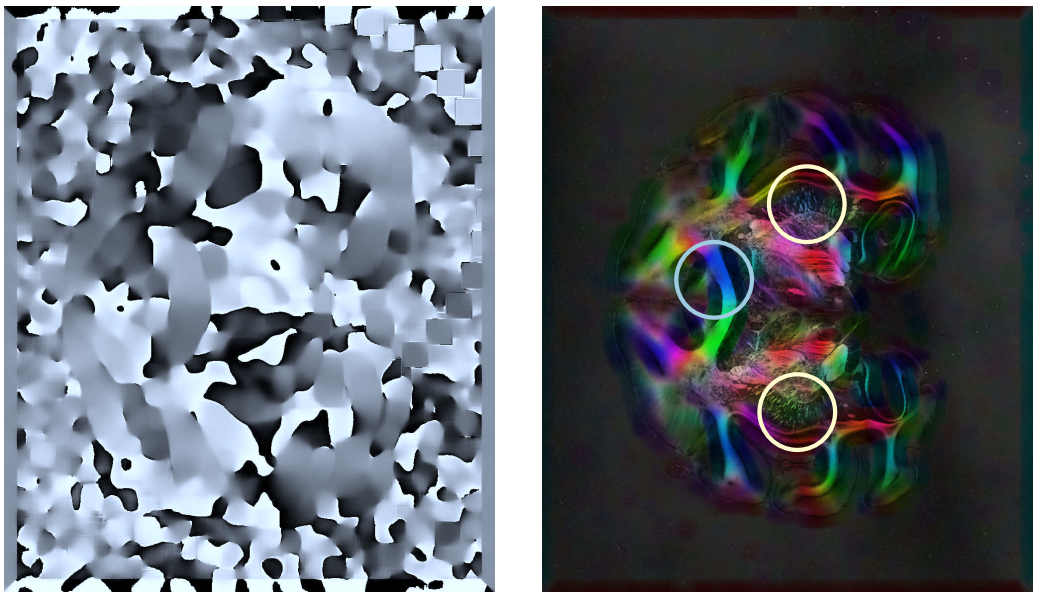


(a) Direktionsbild (b) RGB-Bild

Abbildung 4.5.: Direktions- und RGB-Bild nach Anwendung der 2D-Strukturtensoranalyse auf das LAP-Bild. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 21×21-Tensorkern. Immer noch gute Erkennung feiner Strukturen (gelbe Kreise), Probleme bei groben Strukturen nur noch in einem schmalen Streifen in der Mitte der Struktur (blauer Kreis).

Um dieses Problem zu lösen, erhöht man die Größe des Tensorkerns auf 21×21 (Abb. 4.5). Man erkennt gut, wie das Rauschen im Direktionsbild stark zurückgeht, während das Bild selbst allerdings viel gröber wird.

Betrachtet man das RGB-Bild, sieht man, dass die Erkennung der feinen Strukturen auch bei diesem Tensorkern noch sehr gut funktioniert (gelbe Kreise). Die groben Strukturen (blauer Kreis) werden nur in einem schmalen Streifen in der Mitte noch nicht vollständig erkannt.



(a) Direktionsbild (b) RGB-Bild

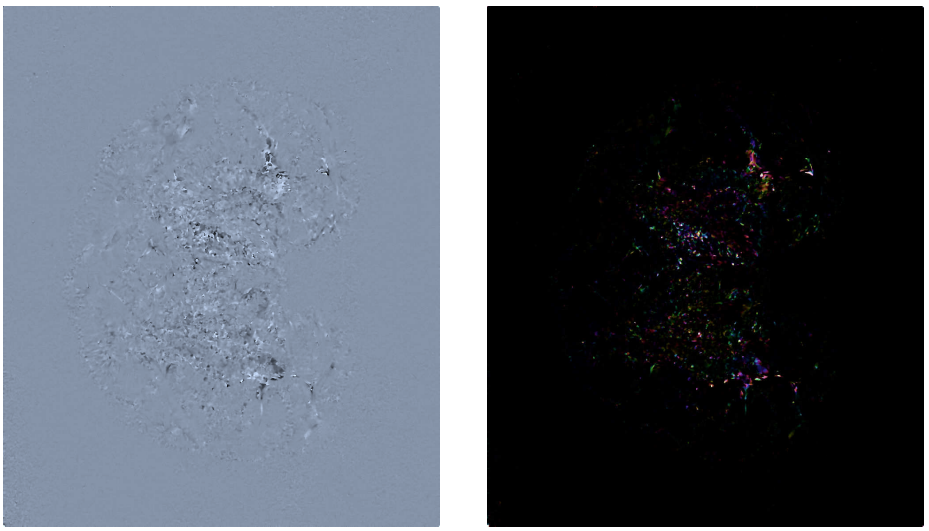
Abbildung 4.6.: Direktions- und RGB-Bild nach Anwendung der 2D-Strukturtensoranalyse auf das LAP-Bild. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 51×51-Tensorkern. Sehr verwaschene Richtungen bei feinen Strukturen (gelbe Kreise), gute Erkennung grober Strukturen (blauer Kreis).

Erhöht man die Größe des Tensorkerns auf 51×51, werden auch die größten Strukturen zuverlässig erkannt (Abb. 4.6, blauer Kreis). Ein großes Problem dabei ist, dass zwischen einzelnen feinen Strukturen nicht mehr unterschieden wird. So haben alle Strukturen innerhalb der gelben Kreise annähernd gleiche Richtungen zugeordnet bekommen, was für kleinere Faltungskerne nicht der Fall war. Gut sichtbar ist dieses Phänomen im sehr groben Direktionsbild, insbesondere im direkten Vergleich mit dem Direktionsbild aus Abbildung 4.5.

Die Größe des Tensorkerns sollte daher von den sichtbaren Strukturen im Bild abhängen. Je gröber die Strukturen, desto größer sollte der Kern sein. Allerdings gilt es darauf zu achten, den Kern nicht zu groß zu wählen, da sonst weichzeichnende Effekte zu groß werden.

3D

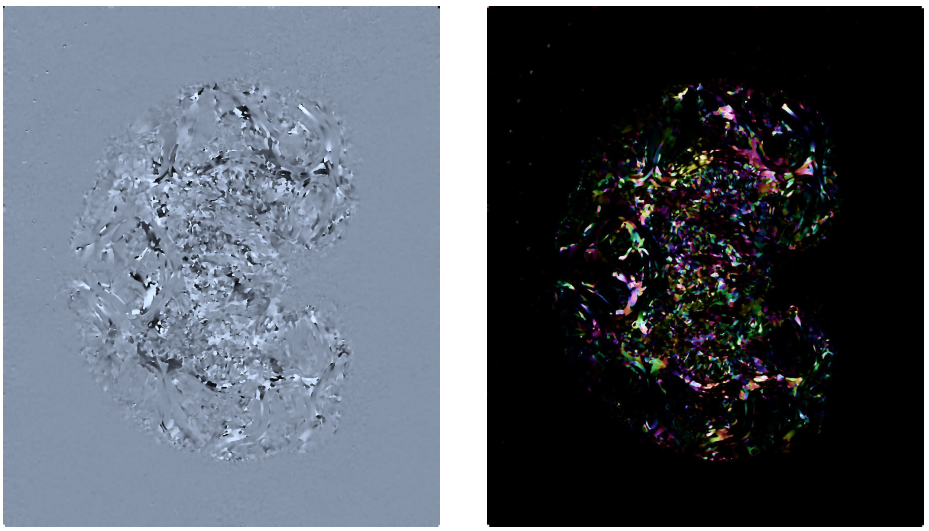
Wendet man die 3D-STA auf den gesamten Stack der LAP-Bilder an, sind die Ergebnisse sehr schlecht (Abb. 4.7). Das Inklinationsbild ist nahezu einheitlich grau, d.h. die Inklinationswinkel sind fast überall etwa Null. Aus diesem Grund ist das dazugehörige RGB-Bild nahezu komplett schwarz. Darüber hinaus lassen die Stellen, an denen die Inklinationswinkel nicht Null sind, keine klare Struktur, sondern nur ein Rauschen erkennen.



(a) Inklinationsbild (b) RGB-Bild

Abbildung 4.7.: Direktions- und RGB-Bild nach Anwendung der 3D-Strukturtensoranalyse auf den LAP-Stack. Genutzt wurden ein 3×3×3-Glättungskern, ein 5×5×5-Gradientenkern und ein 7×7×7-Tensorkern.

Lösung dieses Problems könnte eine stärkere Glättung sein. Ist die Inklination immer etwa Null, liegen in z-Richtung zu große Gradienten vor. Dann liegt die Richtung des kleinsten Gradienten fast immer in der Schnittebene.



(a) Inklinationsbild (b) RGB-Bild

Abbildung 4.8.: Direktions- und RGB-Bild nach Anwendung der 3D-Strukturtensoranalyse auf den LAP-Stack. Genutzt wurden ein 11×11×11-Glättungskern, ein 5×5×5-Gradientenkern und ein 7×7×7-Tensorkern.

Auch eine Verstärkung der Glättung führt zu keiner qualitativen Verbesserung der Ergebnisse (Abb. 4.8). Das Inklinationsbild enthält größtenteils nur Nullwinkel. Alle anderen Winkel haben zufälligen Charakter, sodass auch das RGB-Bild unbrauchbar ist.

In dieser Situation ist es sinnvoll, sich einen Querschnitt des LAP-Stacks anzuschauen. Dabei müssten eventuelle Probleme mit den Gradienten in z-Richtung sichtbar werden.

(a) ungeglättet

(b) 3×3×3-Glättungskern

(c) 11×11×11-Glättungskern

Abbildung 4.9.: Querschnitte durch den (geglätteten) LAP-Stack

Bei der Betrachtung der Querschnitte (Abb. 4.9) fällt auf, dass sowohl im schwach, als auch im stark geglätteten Bild sehr unregelmäßige Kanten zu sehen sind. Am deutlichsten sind diese Zickzacklinien in der Mitte des Bildes. Diese unregelmäßigen Kanten sind der Grund für die starken Gradienten in z-Richtung, die eine Berechnung des Inklinationswinkels verhindern.

Dieses Problem ist an dieser Stelle leider nicht lösbar, da hierfür eine Verbesserung der Registrierung nötig wäre. Damit ist eine 3D-Strukturtensoranalyse der LAP-Bilder zum jetzigen Zeitpunkt nicht möglich.

4.3.2. Mikroskop

Beim LAP-Bild beträgt die Auflösung in alle Richtungen etwa 60 µm. Bei Mikroskopdaten beträgt die Auflösung der z-Achse weiterhin 60 µm, während in x- und y-Richtung eine Auflösung von 1.3 µm erreicht wird. Diese ungleiche räumliche Auflösung muss bei der STA berücksichtigt werden.

Dazu skaliert man die z-Komponenten der Eigenvektoren des Strukturtensors um einen Faktor, der dem Auflösungsunterschied der Dimensionen entspricht. In diesem Fall wäre der Faktor etwa $\frac{60}{1.3} \approx 46$. Diese Skalierung korrigiert die Orientierung der Eigenvektoren. Ein Gradient, der im Bild der Raumdiagonalen entspricht, steht in Wirklichkeit nahezu senkrecht, da ein Pixel in z-Richtung etwa 46 Pixeln in x- bzw. y-Richtung entspricht.

Anstelle der vollständigen Mikroskopdaten werden auf Grund der enormen Auflösung und den daraus resultierenden Darstellungsproblemen im Folgenden zwei ausgesuchte Bereiche betrachtet (Abb. 4.10).

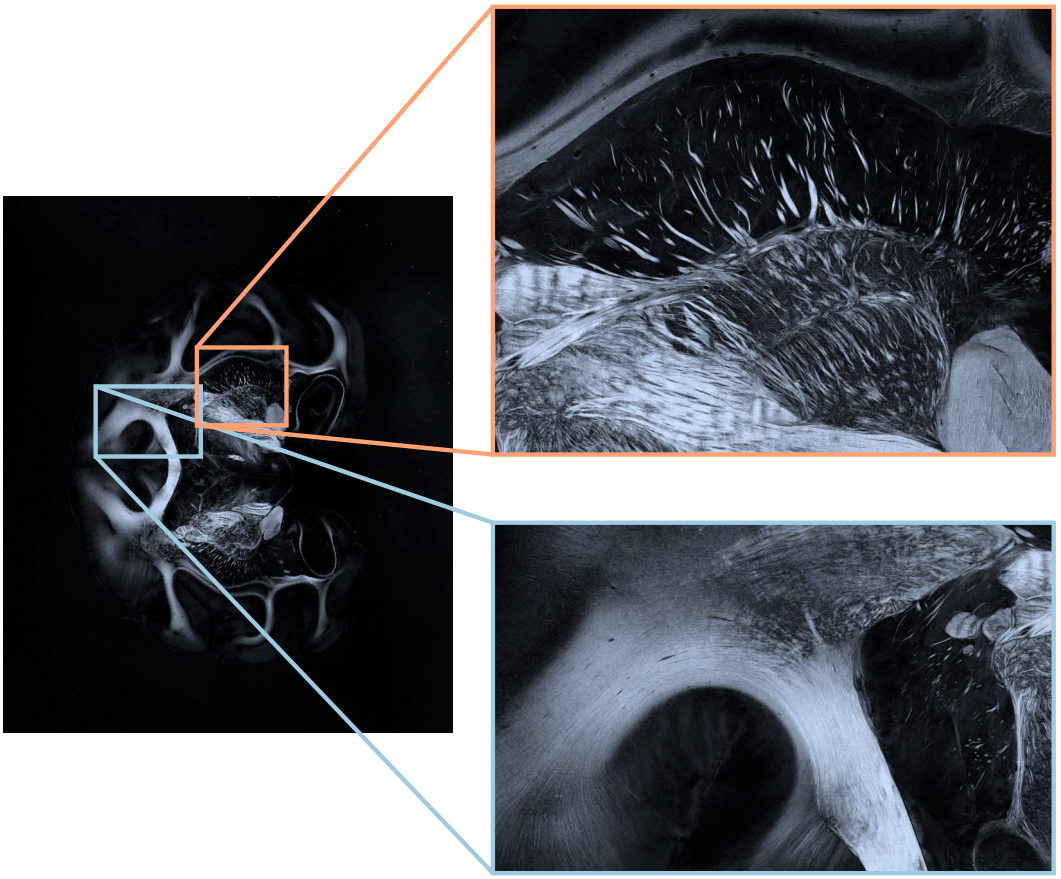


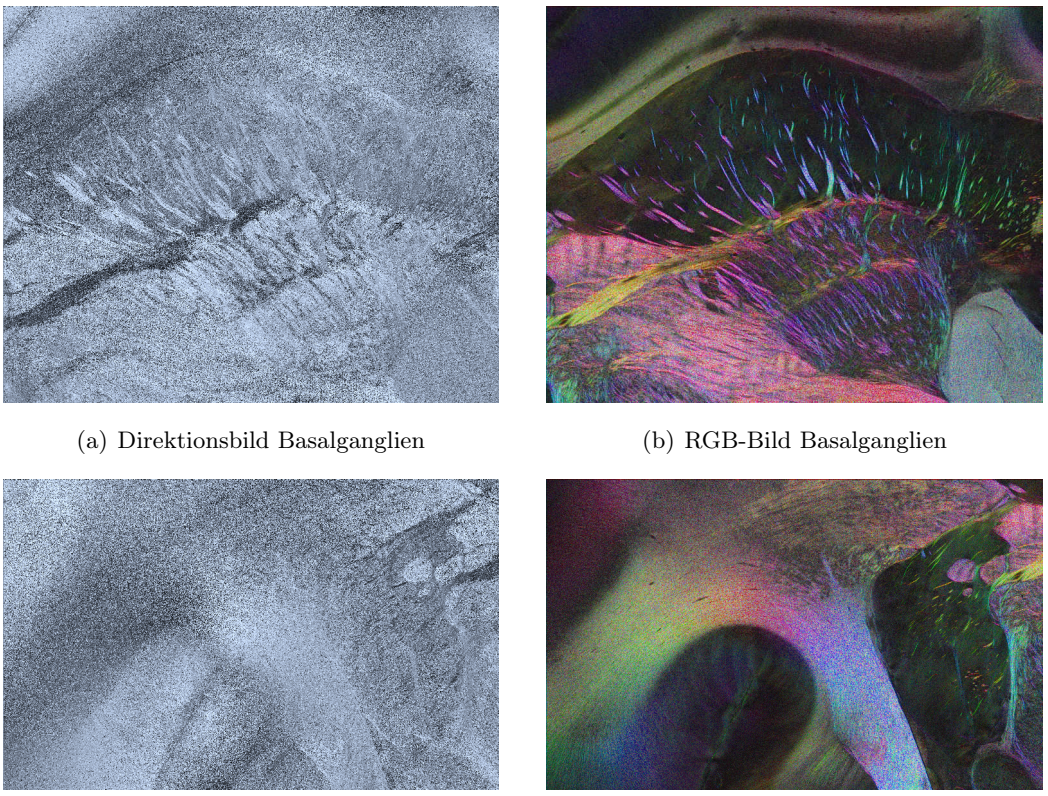
Abbildung 4.10.: Betrachtete Ausschnitte des Mikroskopgesamtbildes. Kleinste (Basalganglien, rechts oben) und größte vorhandene Struktur (Teil des Corpus callosum, rechts unten)

Dabei handelt es sich um den Bereich des größten Faserstranges, dem Corpus callosum (blauer Ausschnitt) und den Bereich der feinsten Strukturen, den Basalganglien (oranger Ausschnitt). Auf diese beiden Bereiche werden wie im Folgenden erst die 2D-STA und dann die 3D-STA angewendet.

2D

Obwohl der 7×7-Tensorkern bei Anwendung auf LAP-Bilder nur die allerfeinsten Strukturen sauber erkannt hat und bei größeren Strukturen nur an den Rändern korrekte Ergebnisse geliefert hat (Abb. 4.4), ist es bei Mikroskopdaten trotzdem sinnvoll, ihn zu nutzen. Durch die sehr hohe Auflösung der Bilder besteht hier die Möglichkeit, dass einzelne Fasern innerhalb der großen Stränge sichtbar sind und somit auch von der STA erkannt werden können.

Daher wurde auf den beiden ausgewälten Regionen der Mikroskopbilder als erstes eine STA mit 3×3-Glättungskern, 5×5-Gradientenkern und 7×7-Tensorkern durchgeführt (Abb. 4.11).



(a) Direktionsbild Basalganglien

(b) RGB-Bild Basalganglien

(c) Direktionsbild Hauptstrang

(d) RGB-Bild Hauptstrang

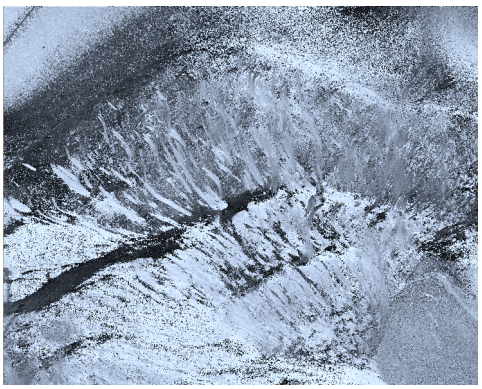
Abbildung 4.11.: Direktions- und RGB-Bilder nach Anwendung der 2D-STA auf beide Regionen des Mikroskopbildes. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 7×7-Tensorkern.

Betrachtet man die RGB-Bilder, bestätigt sich die aufgestellte Vermutung. Im Gegensatz zu den LAP-Daten werden allen vorhandenen Strukturen plausibel wirkende Richtungen zugeordnet, unabhängig von Strukturgröße und Position innerhalb der Struktur.

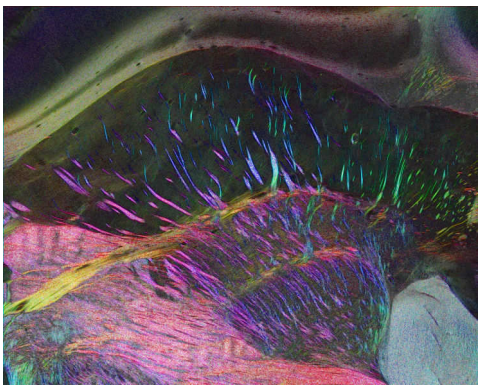
An dieser Stelle werden die Vorteile der Mikroskopbilder gegenüber den LAP-Bildern bei der STA sehr deutlich. Da einzelne Fasern erkennbar sind, ist die Erkennung von Strukturen grundsätzlich auch mit Faltungskernen möglich, die wesentlich kleiner sind als die Größe der Gesamtstruktur.

Da die Direktionsbilder recht hohe Rauschanteile enthalten, ist die Gesamtqualität des Bildes dennoch nicht zufriedenstellend. Rauschen bedeutet in diesem Kontext immer eine Streuung der berechneten Richtungen, sodass die Ergebnisse der einzelnen Pixel als unsicher angesehen werden müssen, obwohl das Gesamtbild auf den ersten Blick plausibel wirkt.

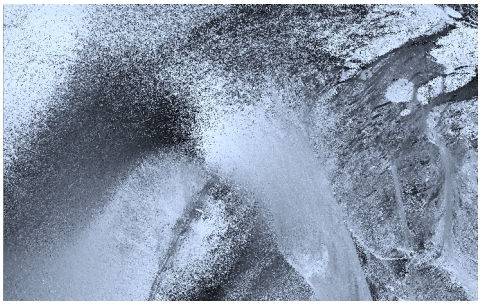
Daher wurde der Tensorkern erst auf 21×21 (Abb. 4.12) und im Anschluss auf 51×51 (Abb. 4.13 und 4.14) vergrößert, um zufällige Gradienten und somit auch das Rauschen der Direktion zu minimieren. Ein nahezu vollständiger Rückgang des Rauschens ist vor allem beim 51×51-Tensorkern sehr gut erkennbar.



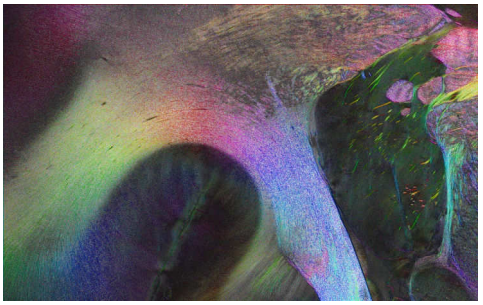
(a) Direktionsbild Basalganglien



(b) RGB-Bild Basalganglien

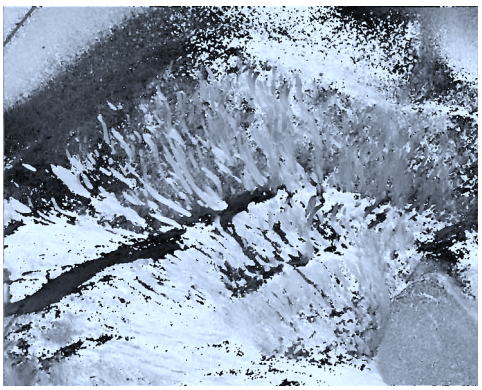


(c) Direktionsbild Hauptstrang

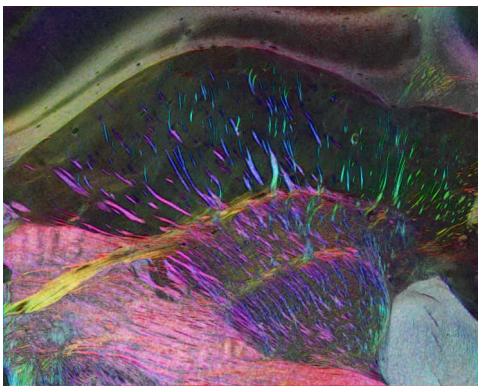


(d) RGB-Bild Hauptstrang

Abbildung 4.12.: Direktions- und RGB-Bilder nach Anwendung der 2D-STA auf beide Regionen des Mikroskopbildes. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 21×21-Tensorkern.

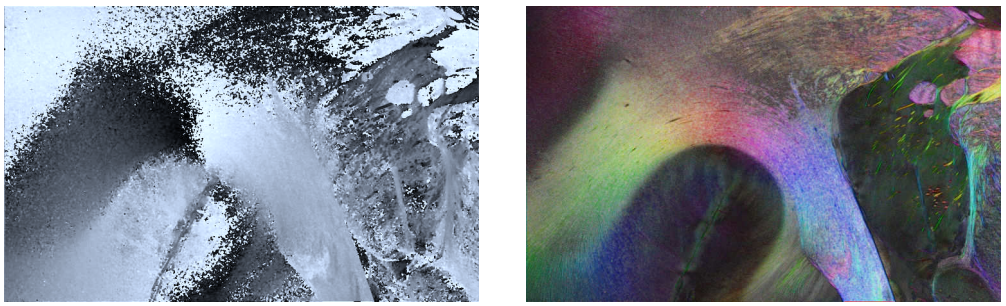


(a) Direktionsbild Basalganglien



(b) RGB-Bild Basalganglien

Abbildung 4.13.: Direktions- und RGB-Bild nach Anwendung der 2D-STA auf das Bild der Basalganglien. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 51×51-Tensorkern.



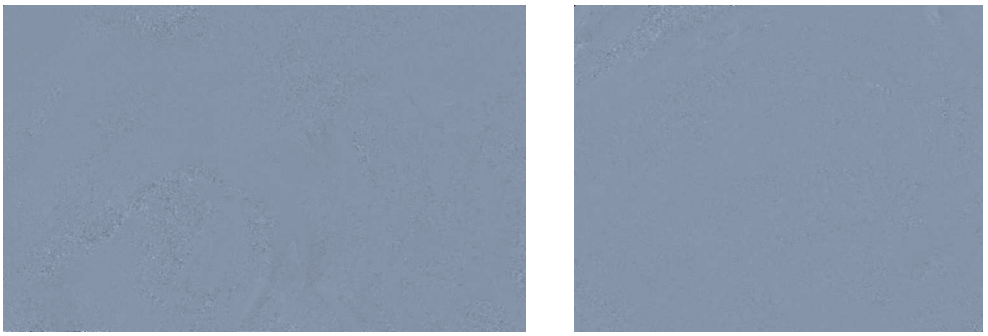
(a) Direktionsbild Hauptstrang (b) RGB-Bild Hauptstrang

Abbildung 4.14.: Direktions- und RGB-Bild nach Anwendung der 2D-STA auf das Bild des Hauptstrangs. Genutzt wurden ein 3×3-Glättungskern, ein 5×5-Gradientenkern und ein 51×51-Tensorkern.

Insgesamt sehen die RGB-Bilder für alle betrachteten Tensorkerne vernünftig aus. Bei genauerer Betrachtung fällt auf, dass auch sie ein leichtes Rauschen enthalten, das bei größeren Kernen schwächer wird. Im Vergleich mit LAP-Bildern liefert der 51×51-Tensorkern allgemein sehr gute Ergebnisse für alle Strukturgrößen, was an der wesentlich höheren Auflösung der Bilder und der Erkennbarkeit einzelner Fasern liegt.

3D

Im nächsten Schritt wird auch auf diese Bilder eine 3D-STA angewendet (Abb. 4.15). Wie dabei gut zu sehen ist, sind die berechneten Inklinationswinkel auch bei Mikroskopdaten vollkommen unbrauchbar. Sie liegen überall im Bereich von 0 Grad, was auch hier auf vergleichsweise große Gradienten in z-Richtung hindeutet.



(a) Inklinationsbild Hauptstrang (b) Inklinationsbild Basalganglien

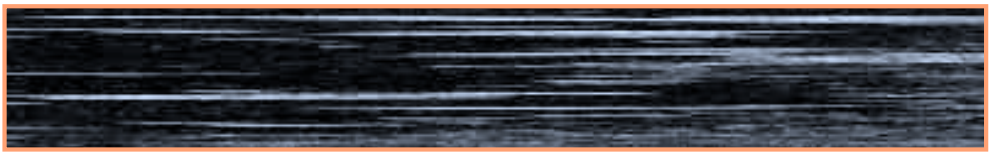
Abbildung 4.15.: Inklinationsbilder nach Anwendung der 3D-STA auf beide Regionen des Mikroskopbildes. Genutzt wurden ein 11×11×11-Glättungskern, ein 5×5×5-Gradientenkern und ein 3×3×3-Tensorkern.

Wie bereits bei den LAP-Bildern, lohnt an dieser Stelle ein Blick auf einen Querschnitt des Datensatzes durch die verschiedenen Schnitte (Abb. 4.16). Dabei zeigen sowohl der originale Querschnitt, als auch der gezoomte Querschnitt keinerlei erkennbare

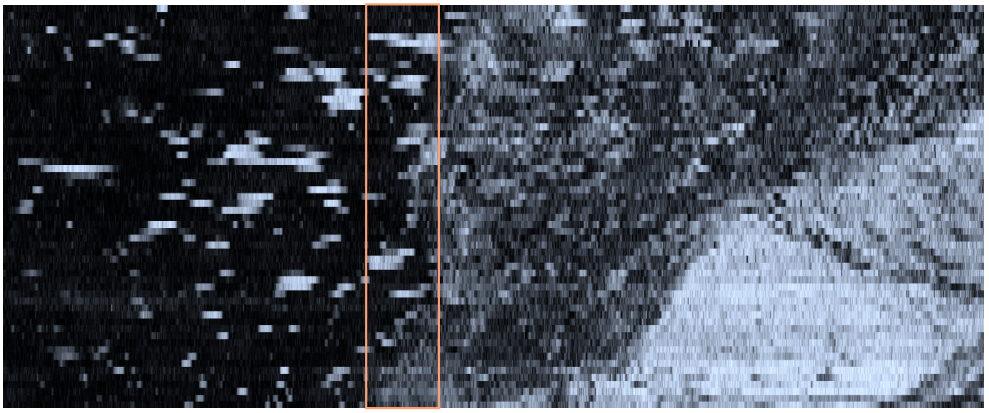
Struktur, insbesondere in z-Richtung. Dies ist der Grund für die Größe der Gradienten in z-Richtung, die eine korrekte Inklinationsberechnung verhindern.

Besonders deutlich wird es, wenn man den Querschnitt bezüglich der Seitenverhältnisse korrigiert. Da die Pixelabstände, wie in Abschnitt 4.3.2 erwähnt, in z-Richtung um den Faktor 46 größer sind als innerhalb des Schnittes, muss der Querschnitt in der Höhe um diesen Faktor gestreckt werden, um einem echten Querschnitt mit tatsächlichen Seitenverhältnissen möglichst nahe zu kommen.

(a) Querschnitt



(b) gezoomter Querschnitt



(c) Querschnitt mit korrekten Pixelabständen

Abbildung 4.16.: Querschnitt durch den Basalganglien-Stack

Man erkennt in der linken Hälfte des skalierten Querschnittes zwar vorhandene Strukturen, diese verlaufen aber meist nur innerhalb eines oder zweier Schnitte. Daher sind in z-Richtung keine sinnvollen Gradienten zu erwarten. In der rechten Hälfte des Bildes verlaufen die einzelnen Strukturen durch viele Schnitte hindurch, allerdings mit über Schnitte hinweg stark schwankender gemessener Intensität. Dieses Rauschen in z-Richtung verhindert ebenfalls die Berechnung sinnvoller Gradienten.

Ebenso können in z-Richtung einzelne Fasern nicht aufgelöst werden, da die Schnittdicke den Durchmesser einzelner Fasern um ein Vielfaches übersteigt. Die gemessene Intensität hängt hier zusätzlich von der Anzahl übereinander liegender Fasern innerhalb des Schnittes ab. Diese Schwankung wäre zwar mit Hilfe von noch stärkerer Glättung beherrschbar, allerdings zerstört man so nahezu sämtliche Strukturinformation.

Diese Probleme sind ebenfalls an dieser Stelle nicht lösbar, da hierfür die Schnittdicke so weit reduziert werden müsste, dass sie in der Größenordnung des Pixelabstandes innerhalb der Schnitte liegt. Dies ist zum jetzigen Zeitpunkt nicht möglich.

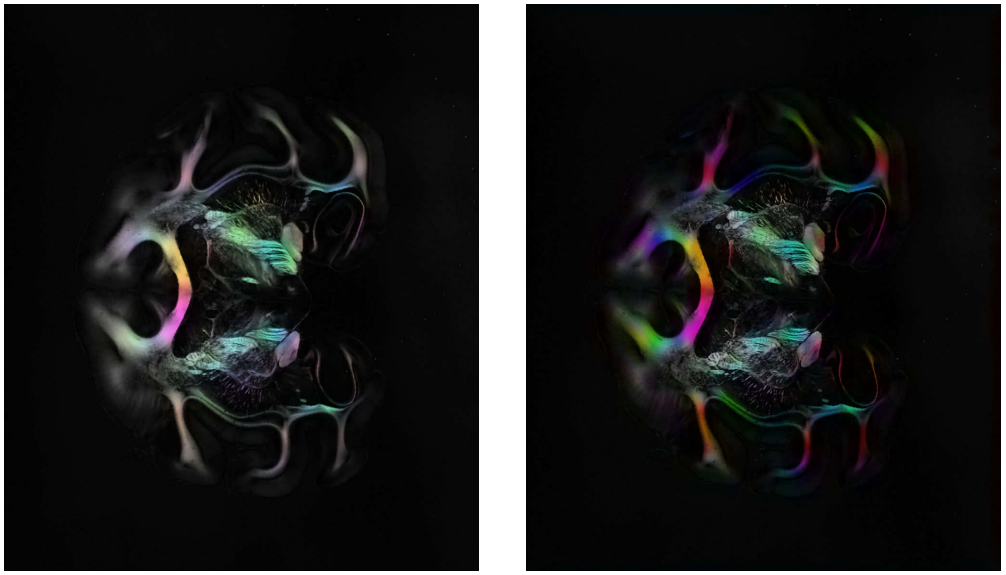
4.4. Richtungsvergleich

Zur Bestätigung der erzielten Ergebnisse kann man diese mit alternativen Berechnungsmethoden vergleichen. Da eine 3D-STA der PLI-Bilder nicht möglich ist, können nur die Ergebnisse der 2D-STA mit anderen Daten verglichen werden.

4.4.1. OrientationJ

OrientationJ, entwickelt von der Biomedical Imaging Group der École polytechnique fédérale de Lausanne, ist ein Plugin für das Bildbearbeitungsprogramm ImageJ, das für größenbeschränkte Daten ebenfalls eine 2D-STA durchführen kann [OriJ]. Um eigene Ergebnisse mit denen von OrientationJ vergleichen zu können, müssen kleine Anpassungen vorgenommen werden. So läuft bei OrientationJ die 0-Grad-Richtung senkrecht, während sie bei PLI waagrecht verläuft. Zusätzlich bietet OrientationJ keine Möglichkeit, Sättigung und Grundhelligkeit eines Pixels durch Anwenden der Wurzelfunktion zu verstärken. In den für den Vergleich erzeugten Bildern wurden daher alle Richtungen um 90 Grad gedreht und die Aufhellung bzw. Sättigungsverstärkung entfernt.

Stellt man das aus den LAP-Daten berechnete RGB-Bild von OrientationJ ($\sigma = 10$) dem mit 3×3-Glättungskern, 5×5-Gradientenkern und 51×51-Tensorkern (d.h. $\sigma = 12.5$) berechneten Ergebnisbild dieser Arbeit gegenüber (Abb. 4.17), sieht man, dass sehr ähnliche Ergebnisse erzielt wurden. Allerdings ist die Farbsättigung an sehr vielen Stellen des eigenen Bildes höher, was auf bessere Kohärenzwerte schließen lässt. Grundsätzlich sind die bestimmten Richtungen aber gleich, da die Farbtöne übereinstimmen.

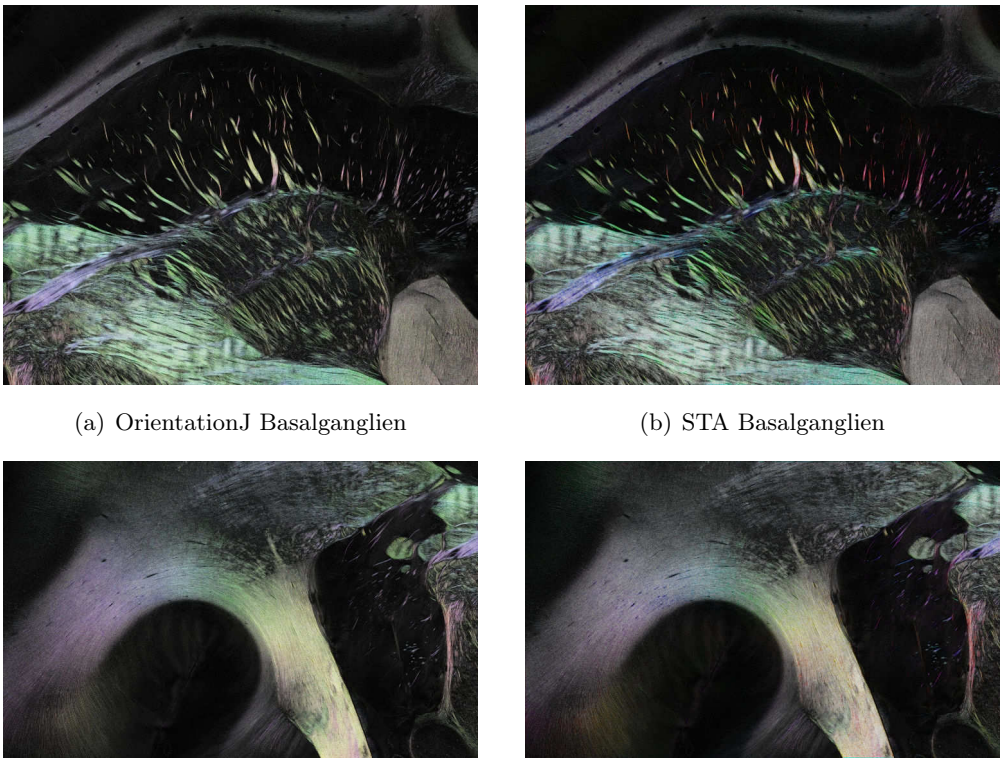


(a) OrientationJ LAP-Bild (b) STA LAP-Bild

Abbildung 4.17.: Vergleich der RGB-Bilder von OrientationJ mit $\sigma = 10$ und der eigenen 2D-STA zum LAP-Datensatz mit 3x×3-Glättungskern, 5×5-Gradientenkern und 51×51-Tensorkern (d.h. $\sigma = 12.5$)

Genau wie die Ergebnisse, die aus den LAP-Daten berechnet wurden, lassen sich auch die Ergebnisse, die auf den Mikroskopbildern basieren, gegenüberstellen (Abb. 4.18). Hierfür wurde bei OrientationJ sowohl für die Basalganglien, als auch für den Hauptstrang $\sigma = 10$ gewählt. Die eigene Strukturtensoranalyse basiert auf 3×3-Glättungskern, 5×5-Gradientenkern und 51×51-Tensorkern (d.h. $\sigma = 12.5$).

Man sieht sofort, dass die Ergebnisbilder nahezu identisch sind. Wie beim LAP-Bild ist die Farbsättigung der eigenen Ergebnisse stellenweise etwas höher. Grundsätzlich bestätigen die Ergebnisbilder von OrientationJ aber die eigenen Ergebnisse.



(a) OrientationJ Basalganglien

(b) STA Basalganglien

(c) OrientationJ Hauptstrang

(d) STA Hauptstrang

Abbildung 4.18.: Vergleich der RGB-Bilder von OrientationJ und der eigenen STA der beiden Mikroskopdatensätze

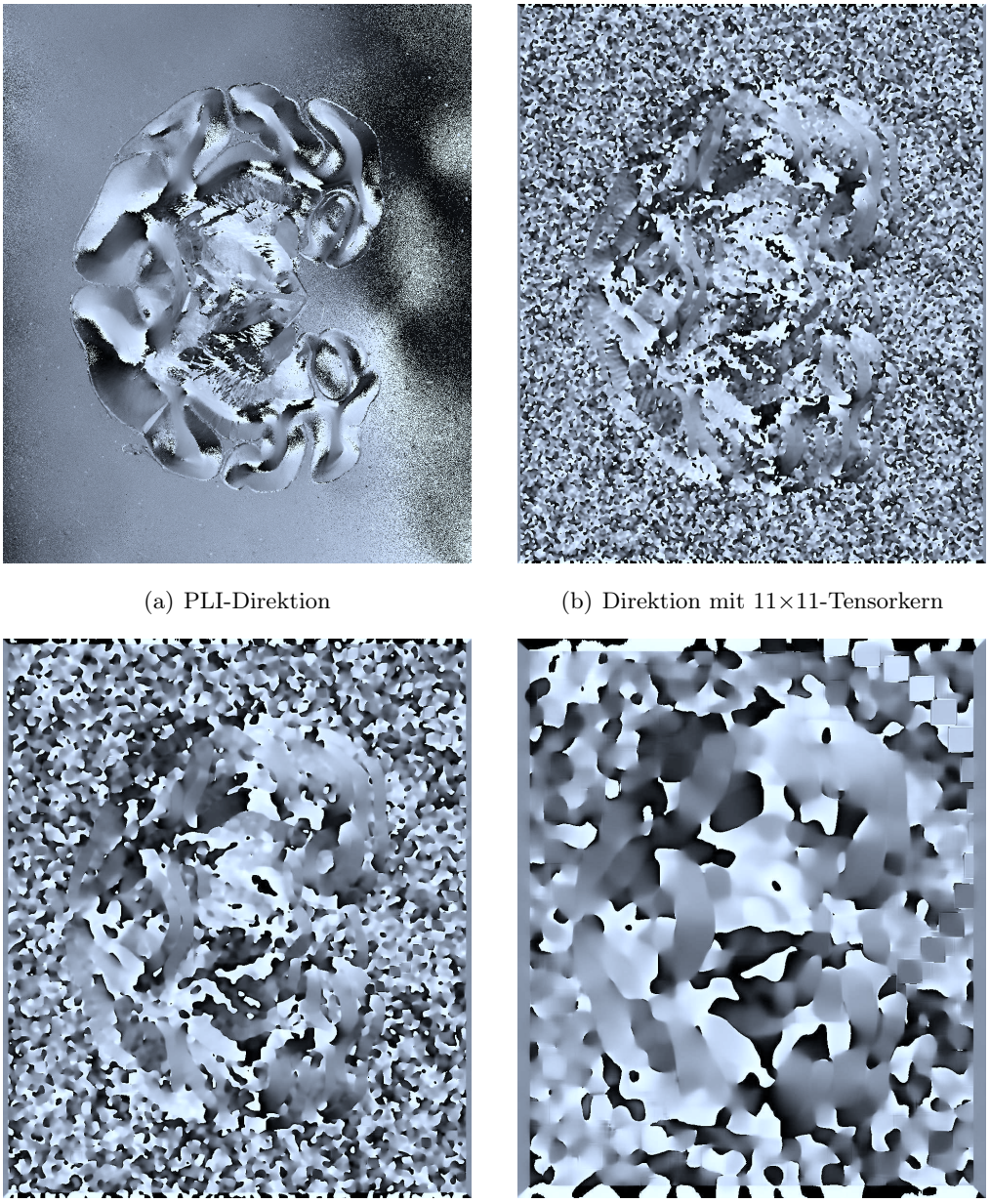
4.4.2. PLI-Richtungen

Neben den Ergebnissen von OrientationJ ist es sinnvoll, die eigenen Ergebnisse mit denen des PLI-Workflow zu vergleichen. Das bedeutet einen direkten Vergleich der berechneten Direktionsbilder.

LAP

Vergleicht man das durch den PLI-Workflow berechnete LAP-Direktionsbild mit denen der STA (Abb. 4.19), fällt auf, dass sich die Ergebnisbilder der STA stark vom PLI-Bild

unterscheiden. Bei Verwendung von Tensorkernen der Größen 21×21 und insbesondere 51×51 werden vorhandene Strukturen so stark verwaschen, dass Ähnlichkeiten nur schwer erkennbar sind, auch wenn in einzelnen Regionen die Direktionswerte übereinstimmen. Im Vergleich mit dem Direktionsbild, das auf dem 11×11-Tensorkern beruht, sind die Ähnlichkeiten besser erkennbar. Allerdings ist dieses Bild stark verrauscht und auch hier sind vor allem die eigentlich klaren Kanten nicht wirklich erkennbar.



(a) PLI-Direktion

(b) Direktion mit 11×11-Tensorkern

(c) Direktion mit 21×21-Tensorkern

(d) Direktion mit 51×51-Tensorkern

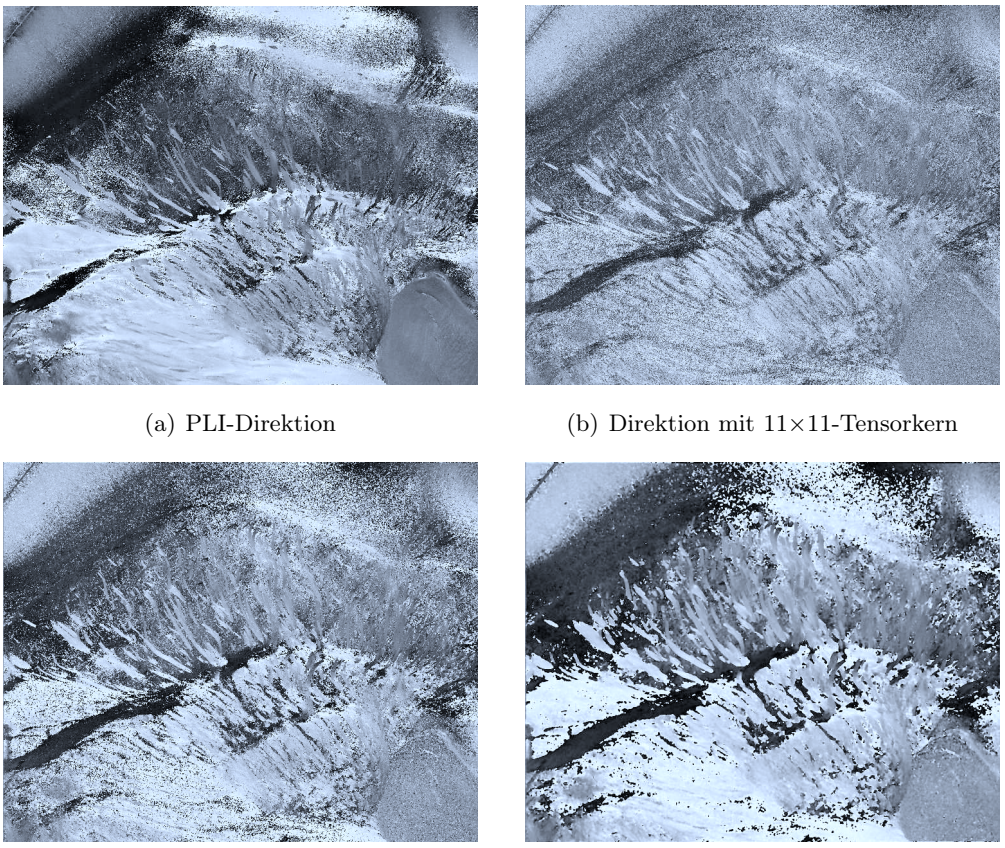
Abbildung 4.19.: Vergleich der für das LAP-Bild berechneten Direktionen von PLI-Workflow und 2D-STA mit verschieden großen Tensorkernen

In einem direkten Vergleich liegen die STA-Direktionsbilder der LAP-Daten weit hinter den Ergebnissen des PLI-Workflows. Begründen lässt sich das durch die geringe Auflösung der LAP-Bilder und die Tatsache, dass PLI aus 18 Bildern und physikalischen Informationen seine Ergebnisse berechnet, die STA jedoch nur auf Gradienten innerhalb des Retardierungsbildes aufbaut.

Mikroskop

Angewendet auf Mikroskopdaten erzeugt die Strukturtensoranalyse wesentlich bessere Direktionsbilder. Ein Vergleich der Direktionsbilder von PLI und STA für beide betrachteten Hirnregionen zeigt die hohe Qualität der Ergebnisse (Abb. 4.20 und 4.21).

In beiden Regionen ist das Direktionsbild jeweils bei Berechnung mit einem 11×11-Tensorkern verrauscht und enthält in mittleren und großen Strukturen nicht zuverlässig dieselbe Richtung wie PLI. Eine Vergrößerung des Tensorkerns auf 21×21 und 51×51 liefert schließlich den PLI-Ergebnissen sehr ähnliche Direktionsbilder.



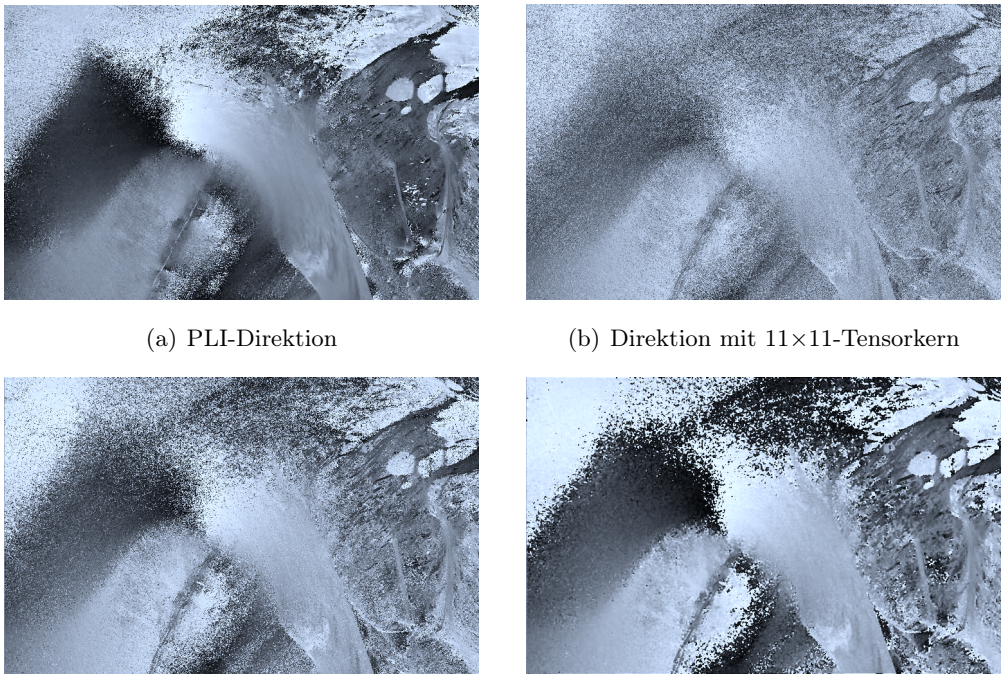
(a) PLI-Direktion

(b) Direktion mit 11×11-Tensorkern

(c) Direktion mit 21×21-Tensorkern

(d) Direktion mit 51×51-Tensorkern

Abbildung 4.20.: Vergleich der für das Mikroskopbild der Basalganglien berechneten Direktionen von PLI-Workflow und 2D-STA mit verschieden großen Tensorkernen



(a) PLI-Direktion

(b) Direktion mit 11×11-Tensorkern

(c) Direktion mit 21×21-Tensorkern

(d) Direktion mit 51×51-Tensorkern

Abbildung 4.21.: Vergleich der für das Mikroskopbild des Hauptstrangs berechneten Direktionen von PLI-Workflow und 2D-STA mit verschieden großen Tensorkernen

Alle markanten Strukturen sind dort erkennbar und die Richtungen selbst entsprechen nahezu den PLI-Richtungen. Der wohl deutlichste Unterschied ist das verhältnismäßig grobkörnige Rauschen am Rand der großen Strukturen, was jedoch in der Größe des Tensorkerns begründet liegt.

Zusammenfassend bieten die Ergebnisse der Strukturtensoranalyse also eine sehr gute Möglichkeit, einzelne Ergebnisse des PLI-Workflows zu verifizieren. Momentan beschränkt sich dies ausschließlich auf die Direktionsbilder.

5. Implementierung

Bei der Implementierung der vorgestellten Algorithmen sind verschiedene Eigenschaften von PLI und den erzeugten Daten zu beachten. So werden, wie bereits in Kapitel 2 erwähnt, bei Mikroskopaufnahmen pro Gehirn etwa 2000 Schnitte mit je 3500 Kacheln erzeugt. Für jede dieser Kacheln werden 18 Bilder einer Größe von 2048×2048 Pixeln aufgenommen.

Bei einem Speicherverbrauch von 4 B pro Pixel (bei einfacher Genauigkeit des Datentyps) haben nur die $2000 \cdot 3500 \cdot 18 = 126$ Millionen Bilder der Rohdaten einen Speicherplatzbedarf von je $2048 \cdot 2048 \cdot 4\,\mathrm{B} = 16\,\mathrm{MiB}$, d.h. insgesamt ca. 1.9 PiB. Hinzu kommt noch der Speicherplatz für die Bilder für Retardierung, Transmittanz, Direktion, Inklination und Segmentierung.

5.1. HDF5

Um all diese Dateien übersichtlich und geordnet zu speichern, kann das Hierachical Data Format 5, kurz HDF5, genutzt werden [HDF5]. Es wurde vom National Center for Supercomputing Applications als Nachfolger des veralteten HDF4 entwickelt. Da HDF4 noch signed Integer-Werte für die Adressierung der Daten nutzte, war die Dateigröße auf etwa 2 GiB beschränkt. Dieses Limit und andere Einschränkungen, wie etwa die fehlende Möglichkeit für parallelen I/O, machten HDF4 für immer mehr Anwendungen unbrauchbar.

Das Nachfolgeformat HDF5 hat diese Einschränkungen nicht mehr. Sein Hauptanwendungsbereich ist die Speicherung riesiger Datenmengen im wissenschaftlichen Umfeld. Da der HDF5-Container nach außen hin für das Betriebssystem nur eine einzige Datei ist, können insbesondere die Probleme, die durch die enorme Anzahl an Dateien auftreten, durch HDF5 gelöst werden.

Die Dateien selbst sind hierarchisch organisiert. Wie in einem Dateisystem ist jedes Objekt innerhalb einer Datei über einen definierten Pfad erreichbar (Abb. 5.1).

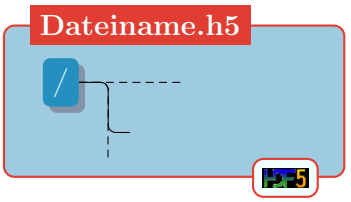


Abbildung 5.1.: Wurzelverzeichnis

Der Pfad jeder HDF5-Datei beginnt mit dem Wurzelverzeichnis „/“. Tieferliegende Pfadstrukturen ergeben sich aus den Namen der geschachtelten Objekte.

Die Anzahl verschiedener Objekte ist dabei stark beschränkt, was die Komplexität verringert und die Ähnlichkeit zu einem Dateisystem erhöht.

Gruppen Ähnlich zu Ordnern in einem Dateisystem, können in einer HDF5-Datei sogenannte *Gruppen* unterhalb des Wurzelverzeichnisses beliebig verschachtelt werden (Abb. 5.2). Dabei ist weder die Anzahl nebeneinander liegender Gruppen, noch die Schachtelungstiefe beschränkt, einzige Forderung ist die Eindeutigkeit eines Pfades. Somit können keine zwei Gruppen mit gleichem Namen unterhalb der selben Gruppe existieren.

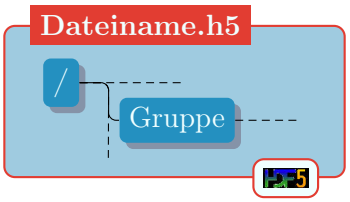


Abbildung 5.2.: Gruppe

Datensätze So wie Dateien in einem Dateisystem abgelegt werden können, werden *Datensätze* in HDF5-Dateien abgelegt. Sie liegen an beliebiger Position der Gruppenhierarchie innerhalb der HDF5-Datei (Abb. 5.3). Wie bei Gruppen muss auch der Pfad von Datensätzen, d.h. der Name innerhalb einer Gruppe, eindeutig sein.

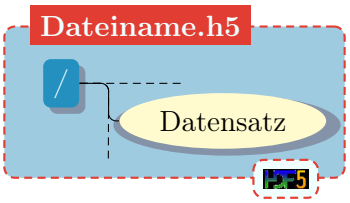


Abbildung 5.3.: Datensatz

Logisch gesehen sind diese Datensätze n-dimensionale Matrizen eines wählbaren Datentyps, der für alle Einträge gleich ist. Die Anzahl der Dimensionen und die jeweilige Größe werden beim Erzeugen des Datensatzes festgelegt und sind beliebig. Diese Eigenschaften sind nach dem Erzeugen nicht mehr änderbar.

Da Dimensionsanzahl und -größe beliebig sind, ist es abhängig von den Nutzdaten möglich, dass der Datensatz so groß ist, dass er nicht mehr vollständig in den Arbeitsspeicher eines Rechners passt. Um solche Datensätze trotzdem verarbeiten zu können, bietet HDF5 die Möglichkeit, diese teilweise einzulesen.

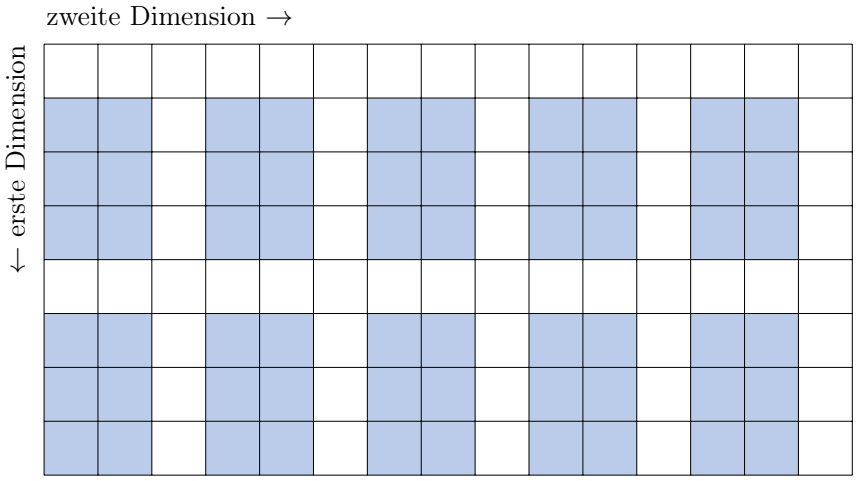


Abbildung 5.4.: Die dunkel markierten Pixel eines Bildes werden ausgewählt, wenn man in einem Datensatz der Größe 8×15 einen *Unterbereich* auswählt mit folgenden Parametern: Offset: (1,0) – Count: (2,5) – Stride: (4,3) – Block: (3,2)

Diese *Unterbereiche* sind über vier Parameter eindeutig anwählbar (Abb. 5.4). Parameter sind dabei die Verschiebung des Startpunktes zum Einlesen in jeder Dimension (*Offset*), die Anzahl der einzulesenden Blöcke pro Dimension (*Count*), die Anzahl der Elemente pro Dimension, nach der ein weiterer Block eingelesen wird (*Stride*) und die Anzahl der Elemente der Dimensionen eines Blockes (*Block*).

Das Auswählen von Unterbereichen und die Einschränkung auf einen einzigen Datentyp pro Datensatz ermöglichen zudem parallelen I/O. So können Daten, sofern vom Dateisystem unterstützt, z.B. blockweise und nicht überlappend von verschiedenen Prozessoren gleichzeitig eingelesen werden.

Attribute Zu jeder Gruppe und jedem Datensatz unterhalb des Wurzelverzeichnisses kann man *Attribute* hinzufügen (Abb. 5.5), etwa um Metadaten oder kleine Vorschaubilder, sogenannte Thumbnails, zu speichern. Metadaten eines Hirns wären dabei medizinische Informationen, wie die Schnittrichtung, das Alter der Person oder technische Daten, wie die Belichtungszeit bei der Aufnahme.

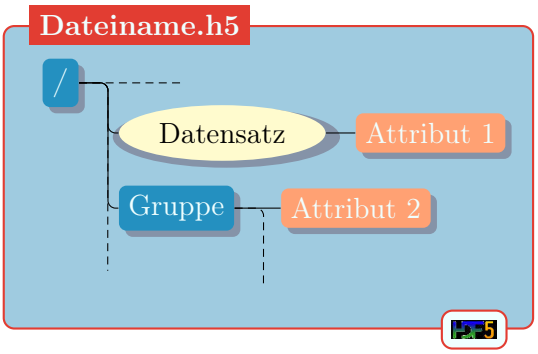


Abbildung 5.5.: Attribut

Die Attribute sind logisch gesehen nichts anderes als ein Datensatz. Allerdings macht HDF5 bei Attributen Einschränkungen gegenüber reinen Datensätzen, die diese klar voneinander abgrenzen. So gibt es beispielsweise keine Möglichkeit, um Unterbereiche von Attributen auszuwählen.

Links Es ist ebenfalls möglich, Objekte in HDF5-Dateien zu verlinken. Dabei wird zwischen internen und externen *Links* unterschieden. Ein interner Link erlaubt es, dass ein Datensatz oder eine Gruppe über mehrere Pfade innerhalb der Datei erreichbar ist, ohne die Daten an zwei Positionen doppelt speichern zu müssen (Abb. 5.6).

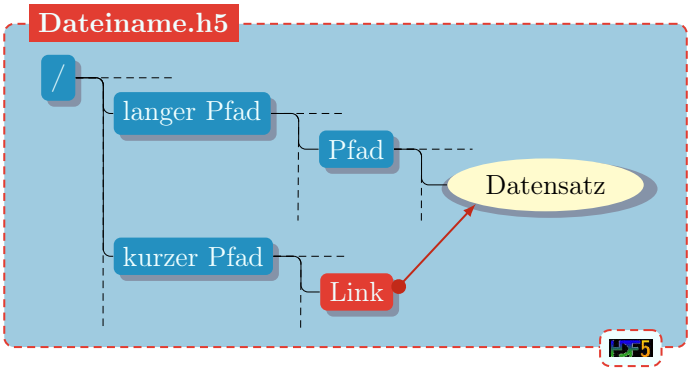


Abbildung 5.6.: Beispiel für die Verwendung eines internen Links. Der Datensatz „`Daten`“ ist sowohl unter „`/langer Pfad/Pfad/Daten`“, als auch unter „`/kurzer Pfad/Daten`“ erreichbar.

Bei der Verarbeitung der Objekte einer Datei muss keine Rücksicht auf eventuelle Links genommen werden. Öffnet man einen Link, so wird automatisch das verlinkte Objekt geöffnet.

Externe Links (Abb. 5.7) ermöglichen es, vorhandene Daten auf mehrere Dateien zu verteilen. Öffnet man einen externen Link, wird automatisch das Objekt in der Zieldatei geöffnet. Auch dieser Vorgang geschieht implizit und bedarf keiner besonderen Beachtung seitens des Nutzers.

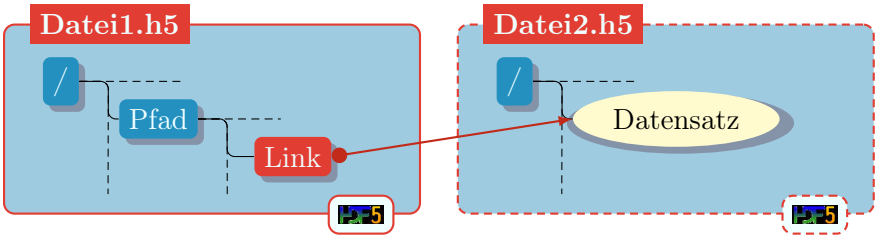


Abbildung 5.7.: Beispiel für die Verwendung eines externen Links. Die in der Datei `Datei2.h5` liegenden Daten „`Datensatz`“ sind auch über den Pfad „`/Pfad/Datensatz`“ in der Datei `Datei1.h5` erreichbar.

Vorhandene Dateistruktur Beim PLI-Workflow wird für jeden Schnitt eine separate HDF5-Datei mit den einzelnen Datensätzen angelegt. Um diese Schnitte organisiert abzulegen, erzeugt man für jedes Gehirn eine zentrale HDF5-Datei und einen Ordner mit dem Namen des Hirns, z.B. „`Brain`“. Die HDF5-Datei „`Brain.h5`“ enthält die externen Links zu den HDF5-Dateien aller verfügbaren Schnitte, die im Ordner „`Brain`“ abgelegt werden.

Die Rohdaten werden dabei, räumlich korrekt angeordnet, kachelweise als ein Datensatz nebeneinander abgelegt. Benötigt man nur einzelne Kacheln, so sind diese als Unterbereich auszulesen. Dadurch sind einzelne Schnitte als Ganzes nur noch schwer handhabbar. Dies liegt hauptsächlich an der enormen Menge an Rohdaten (984 GiB pro Schnitt).

Möchte man beispielsweise nur den Retardierungsdatensatz eines Schnittes übermitteln, dann ist dies nicht möglich, da die HDF5-Datei nur als Ganzes verschickt werden kann. Dies bedeutet, dass alle vorhandenen Datensätze eines Schnittes ebenfalls übermittelt werden müssten.

Um eine saubere Trennung der Daten zu erhalten, werden die Daten ein weiteres Mal aufgeteilt. Die Datei, die bisher alle Daten eines Schnittes enthält, wird so modifiziert, dass sie nur noch Links enthält. Diese Links zeigen auf nach Art der Daten unterteilte HDF5-Dateien.

Diese wiederum liegen in einem Unterordner, dessen Name den Hirnnamen und die Schnittnummer enthält, z.B. „`Brain_125`“. Für Rohdaten, Retardierung, Transmittanz, Segmentierung, Kalibrierung, Direktion und Inklination wird je eine Datei erzeugt. Die Transmittanzdatei enthält z.B. die normale, die normierte und die gefilterte Transmittanz (Abb. 5.8).

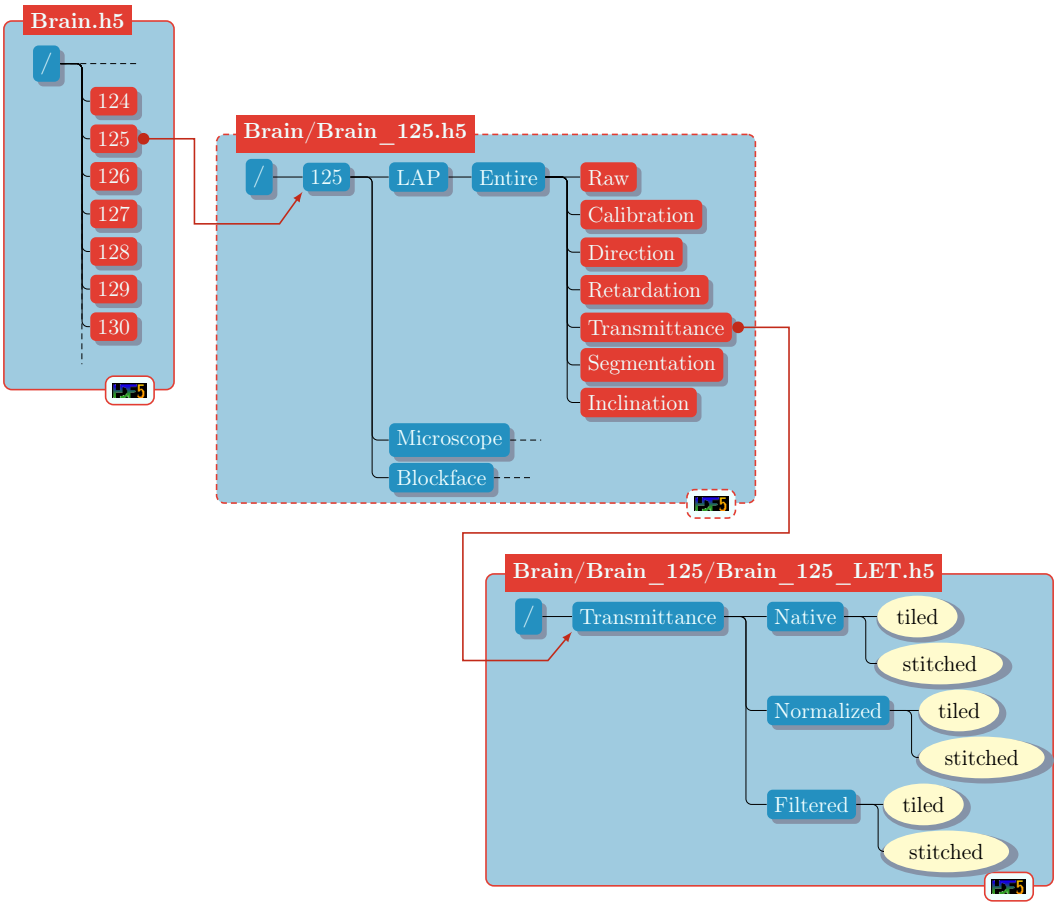


Abbildung 5.8.: Ein Ausschnitt aus der gewählten HDF5-Struktur. Eine Datei pro Gehirn (z.B. „`Brain.h5`“) enthält Links zu allen Schnitten, die in einem Ordner mit Hirnnamen (z.B. „`Brain`“) liegen. Diese wiederum enthalten Links zu den verschiedenen Datensätzen des Schnittes, die in einem Unterordner mit Hirnnamen und Schnittnummer (z.B. „`Brain_125`“) liegen.

5.2. Parallelisierung

Alle Laufzeitmessungen wurden auf dem Ausschnitt des Mikroskopdatensatzes mit 58 Schnitten zu je 10000×8000 Pixeln, der die Basalganglien zeigt, durchgeführt. Ein einzelner Prozessor braucht für die Durchführung einer 3D-Strukturtensoranalyse dieses Datensatzes etwa 8 Stunden und 20 Minuten unter Verwendung von 3×3×3-Glättungskern, 5×5×5-Gradientenkern, 7×7×7-Tensorkern und anschließender Berechnung des RGB-Bildes.

Hochgerechnet auf den vollständigen Datensatz eines Hirnes mit 2000 Schnitten und 140000×100000 Pixeln pro Schnitt würde die gesamte Berechnung ca. 5 Jahre und 9 Monate dauern. Um diese nicht vertretbare Laufzeit auf ein sinnvolles Maß zu reduzieren, ist eine Parallelisierung des Algorithmus notwendig.

5.2.1. Rechnerklassifikation

Allgemein lassen sich Rechnerarchitekturen nach Flynn in vier Klassen unterteilen [MF67]. Die Unterscheidung erfolgt dabei anhand der Anzahl der Daten- und Befehlsströme (Tab. 5.1).

	Single **D**ata	**M**ultiple **D**ata
Single **I**nstruction	**SISD**	**SIMD**
Multiple **I**nstruction	**MISD**	**MIMD**

Tabelle 5.1.: Rechnerklassifikation nach Flynn anhand der Anzahl der Daten- und Befehlsströme

SISD Single Instruction, Single Data. Rechner dieser Klasse sind Einprozessorsysteme, die einzelne Anweisungen auf einzelnen Daten sequentiell abarbeiten. Zu dieser Klasse gehören Rechner, die nach der Von-Neumann- bzw. Harvard-Architektur aufgebaut sind.

SIMD Single Instruction, Multiple Data. Ein System, das dieselben Instruktionen gleichzeitig auf vielen Daten ausführen kann. Ein Beispiel für ein solches System ist der Vektorrechner/-prozessor. Häufigster Anwendungsfall ist die Verarbeitung von Bild-, Ton- und Videodaten, da dort sehr oft dieselbe Operation auf kleinen Teilen der Gesamtdaten auszuführen ist.

MISD Multiple Instruction, Single Data. Eine sehr umstrittene Klassifizierung, da das gleichzeitige Ausführen mehrerer Befehle auf denselben Daten eigentlich nicht möglich ist. Viele ordnen dieser Klasse allerdings fehlertolerante Systeme zu, die redundante Berechnungen auf gleichen Daten ausführen.

MIMD Multiple Instruction, Multiple Data. Eine Klasse von Rechnern, bei der verschiedene Instruktionen gleichzeitig auf verschiedenen Daten ausgeführt werden können. Zusätzlich zu den meisten Supercomputern zählt mittlerweile auch nahezu jeder Heim-PC zu dieser Klasse, da diese meistens Mehrkernprozessoren mit eigenen Registersätzen für Befehle und Daten sind.

5.2.2. Speicherorganisation

Um parallele Programme für ein System entwickeln zu können, ist es unerlässlich zu wissen, wie der Speicher des Systems organisiert ist und auf welche Teile davon ein Prozessor zugreifen kann. Bei Mehrprozessorsystemen unerscheidet man dabei grundsätzlich drei Arten der Speicherorganisation.

Distributed Memory

Eine erste Art der Organisation ist dabei ein System mit den Prozessoren einzeln zugeteiltem Speicher (*Distributed Memory*). Dabei kann jeder Prozessor grundsätzlich nur auf seinen eigenen Speicher direkt zugreifen (Abb. 5.9). Um auf die Daten eines anderen Prozessors zuzugreifen, müssen Nachrichten zwischen den beiden Prozessoren verschickt werden, d.h. es muss explizite Kommunikation erfolgen.

Die Programmierung eines solchen Systems kann beispielsweise mit MPI erfolgen. MPI, kurz für *Message Passing Interface*, ist ein Standard zum Austausch von Nachrichten in verteilten Computersystemen [MPI]. Das parallele Programm wird dabei auf jedem Prozessor einzeln gestartet, es laufen also mehrere Instanzen desselben Programmes. Die Prozessoren erhalten einen eindeutigen Rang, anhand dessen sie untereinander kommunizieren können.

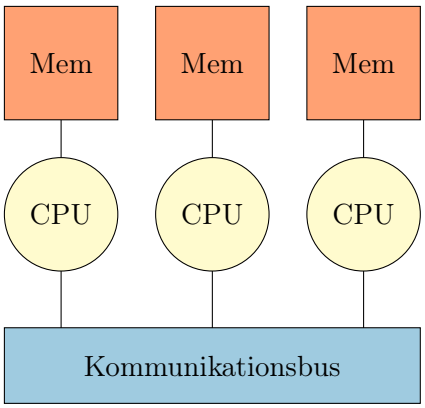


Abbildung 5.9.: Schematische Darstellung eines Systems mit verteiltem Speicher (*Distributed Memory*)

Shared Memory

Im Gegensatz zum *Distributed Memory* ist *Shared Memory* direkt an den Kommunikationsbus angebunden, sodass jeder Prozessor auf den gesamten Speicherbereich zugreifen kann (Abb. 5.10). Ein solches System kann zwar ebenfalls mit MPI programmiert werden, dies ist allerdings sehr ungünstig. Da bei MPI jede Programminstanz ein eigener Prozess mit eigenem Speicherbereich ist, entspricht jede Kommunikation einem Kopiervorgang innerhalb des eigentlich gemeinsam zugreifbaren Speichers.

Eine Lösung dieses Problems ist die Nutzung von OpenMP [OMP]. Anders als bei MPI kann mit OpenMP, kurz für *Open Multi-Processing*, nur ein *Shared Memory*-System programmiert werden. OpenMP erzeugt Parallelität durch das Ausführen eines Programmes mit mehreren Threads, die alle Zugriff auf den gemeinsamen Speicher haben. Die Kommunikation zwischen den Threads erfolgt dabei immer implizit, da der Nutzer sich nur um die Aufteilung separater und gemeinsamer Speicherbereiche kümmern kann.

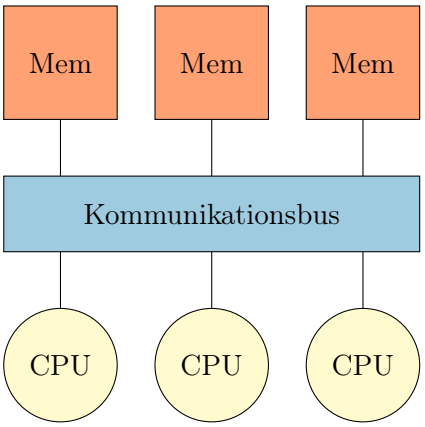


Abbildung 5.10.: Schematische Darstellung eines Systems mit gemeinsamem Speicher (*Shared Memory*)

Kombinationen

Heutige Supercomputer sind meistens eine Kombination aus beiden Techniken, sie sind aus sogenannten *Nodes* aufgebaut (Abb. 5.11). Die Nodes selbst sind ein *Shared Memory*-System. Kommunikation zwischen den Nodes ist jedoch nur explizit möglich, d.h. wie in einem *Distributed Memory*-System.

Die Programmierung (bei der Nutzung mehrerer Nodes) erfolgt dabei mit reinem MPI oder hybrid, d.h. mit OpenMP innerhalb der Nodes und MPI zur Kommunikation zwischen den Nodes.

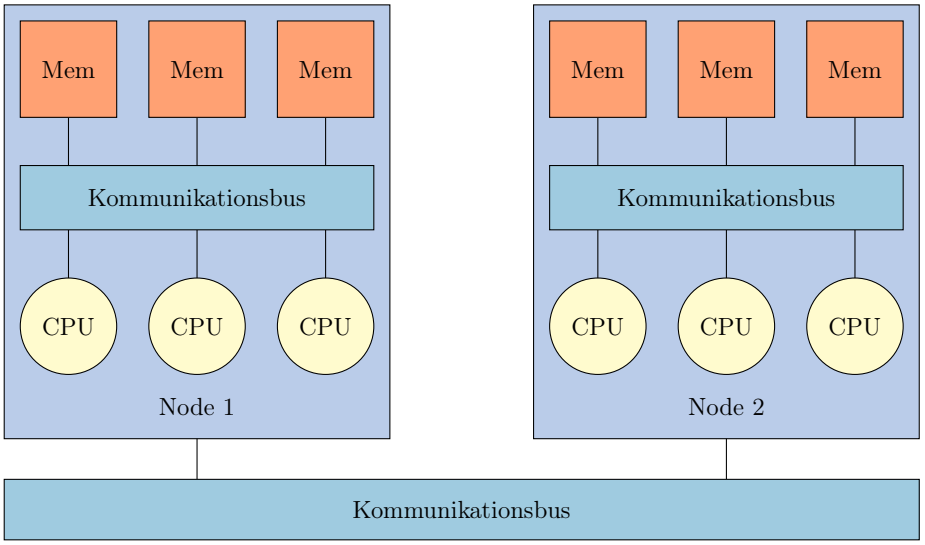


Abbildung 5.11.: Schematische Darstellung eines Systems mit *Nodes*, d.h. einer Kombination aus gemeinsamem Speicher innerhalb der Nodes und verteiltem Speicher zwischen den Nodes

5.2.3. Parallelisierungsstrategien

Für die Parallelisierung des Codes auf logischer Ebene, d.h. die Aufteilung der Rechenlast auf die einzelnen Prozessoren/Threads, gibt es ebenfalls verschiendene Möglichkeiten. Der Einfachheit halber wird im Folgenden nur noch von Prozessoren gesprochen.

Cloning

Die einfachste Methode der Parallelisierung ist das gleichzeitige Ausführen desselben Programmes auf vielen Prozessoren. Da keine wirkliche Aufteilung der Arbeit erfolgt, ist die Anwendung dieser Methode nur sinnvoll, wenn es sich beispielsweise um ein Problem handelt, das auf Zufallszahlen beruht. Durch das mehrfache gleichzeitige Lösen können so mehr Ergebnisdaten in derselben Zeit erzeugt werden, die am Schluss nur zusammengeführt werden müssen.

Funktionale Zerlegung

Je nach Anwendungsfall ist auch eine funktionale Zerlegung des Programmcodes möglich. Dabei wird das Programm in einzelne Komponenten bzw. Module zerlegt, die einzeln bearbeitet werden können. Abhängigkeiten zwischen den einzelnen Systemen müssen mit Kommunikation gelöst werden.

Ein möglicher Anwendungsfall ist beispielsweise eine Klimasimulation. Dabei könnte die Gesamtsimulation sehr vereinfacht in die Simulation von Atmosphäre, Gewässern und Land aufgeteilt werden, welche auf verschiedene Prozessoren verteilt werden können.

Gebietszerlegung

Ist eine funktionale Zerlegung nicht möglich, besteht dennoch sehr oft die Möglichkeit ein Problem in kleine Teilprobleme zu zerlegen und auf die Prozessoren zu verteilen. Diese Gebietszerlegung ist insbesondere bei der Bildverarbeitung sehr oft der richtige Parallelisierungsansatz, da sehr viele Berechnungen unabhängig für jedes Pixel durchgeführt werden können.

Bei der Verteilung der Teilprobleme auf die Prozessoren gibt es wiederum verschiedene Möglichkeiten.

Blockweise Bei der blockweisen Verteilung (Abb. 5.12) wird das Gebiet in Blöcke aufgeteilt, die den einzelnen Prozessoren zugewiesen werden. Jeder Prozessor erhält dabei einen Block. Um eine möglichst gute Lastverteilung zu erreichen, sollten die Blöcke zumindest annähernd gleich groß sein. Die blockweise Verteilung gehört zur Gruppe der statischen Verteilungen, da bereits vor Programmstart bestimmt werden kann, welchem Prozessor welches Gebiet zugeordnet wird.

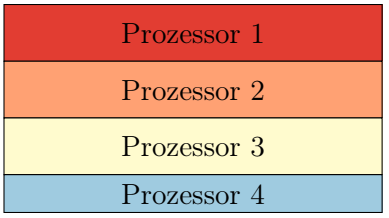


Abbildung 5.12.: Blockweise Verteilung mit vier Prozessoren

Zyklisch Im Vergleich zur blockweisen Verteilung wird der Datensatz bei der zyklischen Verteilung in kleinere, exakt gleich große Blöcke aufgeteilt. Diese Blöcke werden reihum an die einzelnen Prozessoren verteilt und von ihnen bearbeitet (Abb. 5.13). Dadurch kann es passieren, dass einzelne Prozessoren weniger Blöcke bearbeiten müssen als andere.

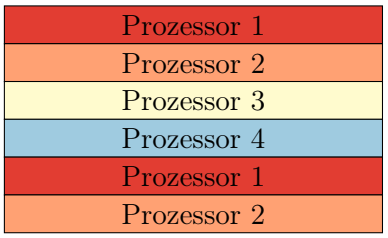


Abbildung 5.13.: Zyklische Verteilung mit vier Prozessoren

Die zyklische Verteilung gehört ebenfalls zur Gruppe der statischen Lastverteilung. Ihr Vorteil ist, dass ein Prozess nicht darauf achten muss, wie groß der ihm zugewiesene Block ist und ob etwa die Daten vollständig in den Arbeitsspeicher passen. Bei blockweiser Verteilung würde dies von der Anzahl der eingesetzten Prozessoren abhängen, hier hängt es von der anfänglichen Wahl der Blockgröße ab.

Master/Worker Neben der statischen Lastverteilung gibt es auch Konzepte mit dynamischer Lastverteilung, deren Zuordnungen von Blöcken zu Prozessoren nicht vorhersagbar sind. Dynamische Lastverteilung ist insbesondere dann sinnvoll, wenn die nötige Rechenleistung pro Block unvorhersagbar schwankt. Durch solche Schwankungen kann es bei statischer Lastverteilung vorkommen, dass einzelne Prozessoren nur sehr schlecht ausgelastet sind, während andere vergleichsweise sehr lange rechnen müssen.

Um dieses Problem zu lösen, wird beim Master/Worker-Schema jedem Prozessor (außer dem Master) am Anfang ein Block zugeordnet. Sobald ein Block abgearbeitet ist, weist der Master diesem Prozessor einen neuen Block zur Bearbeitung zu. So erreicht man eine sehr gute Auslastung der Prozessoren, allerdings auf Kosten eines Prozessors, der nicht für Berechnungen zur Verfügung steht.

5.2.4. Gewählter Parallelisierungsansatz

Das Programm zur Berechnung ist in diesem Fall durch Domain Decomposition parallelisierbar. Da in den vorhandenen Berechnungsschritten – Glättung, Gradienten und Tensoren – keine Abhängigkeiten zwischen Nachbarpixeln bestehen, kann man das Bild in Streifen aufteilen, die dann von den verschiedenen Prozessoren berechnet werden.

An den Übergängen der Berechnungsschritte müssen Barrieren eingebaut werden, denn um beispielsweise Gradienten zu bestimmen, muss das geglättete Bild bereits fertig berechnet sein. Ebenso müssen auf jedem Prozessor die zu verarbeitenden Daten überlappend eingelesen werden (Abb. 5.14), da für die Faltung andernfalls Daten fehlen würden.

Die Verteilung der Daten auf die einzelnen Prozessoren kann nach den soeben beschriebenen Mustern erfolgen. Das Master-/Worker-Schema ist bei der vorliegenden Problemstellung von Nachteil, da in jedem Streifen der gleiche Berechnungsaufwand nötig ist. Anders als bei der Berechnung der Inklination (Abschnitt 2.2.5), wo der Hintergrund aus der Berechnung ausgenommen wurde, verarbeitet die Strukturtensoranalyse jedes Pixel gleich. Der zu erwartende Gewinn durch den Ausgleich ungleicher Lastverteilung fällt in diesem Fall weg.

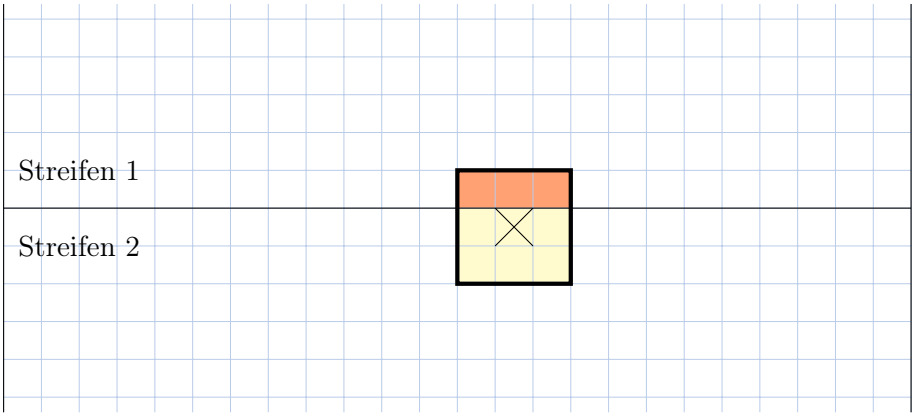


Abbildung 5.14.: Für die Berechnung der Glättung des mit einem Kreuz gekennzeichneten Pixels mit einem 3×3-Kern sind die eingefärbten Pixel nötig. Die orange eingefärbten Pixel gehören allerdings zu einem anderen Streifen und müssen daher überlappend eingelesen werden.

Um sich nicht um eventuelle Speicherprobleme durch zu große Blöcke kümmern zu müssen, fällt die blockweise Verteilung ebenfalls weg. Es wird mit Hilfe der zyklischen Verteilung parallelisiert. Verteilt man jedoch jeden Schnitt einzeln nach diesem Muster, kann ein enormes Ungleichgewicht bei der Lastverteilung auftreten.

Bei der einfachen Form der zyklischen Verteilung würde jeder Schnitt auf jeder Berechnungsstufe gleich auf die Prozessoren verteilt. Das würde aber bedeuten, dass bei einer Bildgröße und Prozessorzahl, bei der nicht alle Prozessoren die gleiche Anzahl Streifen erhalten, der erste Prozessor pro Schnitt einen Streifen mehr berechnen muss. Bei 2000 Schnitten würde dies eine erhebliche Ungleichverteilung der Last bedeuten. Aus diesem Grund wurde eine angepasste Form der zyklischen Verteilung implementiert.

Bei dieser wird die Zyklizität auch über Schnitte hinweg erhalten. Verteilt man beispielsweise 2 Schnitte mit je 6 Streifen auf 4 Prozessoren, bedeutet dies, dass wenn Prozessor 2 den letzten Streifen von Schnitt 1 bearbeitet hat, Prozessor 3 den ersten Streifen von Schnitt 2 bearbeitet. Dadurch erreicht man, dass in diesem Beispiel jeder Prozessor 3 Streifen berechnet (Abb. 5.15). Wären die Prozessoren auf beide Schnitte so verteilt worden wie bei Schnitt 1, dann hätten manche 2, manche 4 Streifen berechnen müssen.

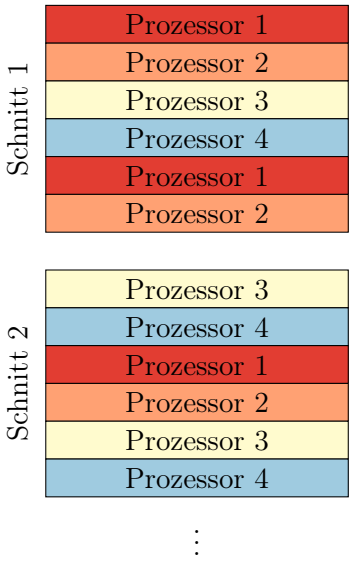


Abbildung 5.15.: Angepasste zyklische Verteilung von zwei Schnitten mit je sechs Streifen an vier Prozessoren

5.2.5. Laufzeitverhalten

Um eine sinnvolle Analyse der Laufzeiten durchführen zu können, muss das Programm immer mit denselben Parametern auf denselben Daten ausgeführt werden. Dazu wird das Programm auf dem bereits am Kapitelanfang gewählten Datensatz mit den dort beschriebenen Parametern aufgerufen, jeweils mit unterschiedlicher Anzahl an Prozessorkernen.

Eine Betrachtung der Laufzeiten (Tabelle 5.2) zeigt, dass die Laufzeit mit jeder Erhöhung der Anzahl der Prozessorkerne kleiner wird.

# Kerne	Laufzeit (s)
1	29 765.9
2	15 968.3
4	8309.2
8	4321.3
12	2947.9
16	2434.4
24	1583.5
32	1090.3
48	773.7
64	627.8
80	494.0
96	430.7

Tabelle 5.2.: Laufzeiten für die Berechnung des Testdatensatzes mit 3×3×3-Glättungskern, 5×5×5-Gradientenkern, 7×7×7-Tensorkern und anschließender Berechnung des RGB-Bildes

Die Laufzeit scheint sich mit jeder Verdopplung der Prozessorkerne annähernd zu halbieren (Tab. 5.2). Insbesondere sinkt die Gesamtverarbeitungszeit des Datensatzes von 8 Stunden und 20 Minuten auf 7 Minuten und 10 Sekunden unter Verwendung von 96 Prozessorkernen. Für das Gesamthirn würde das ein Senken der Verarbeitungszeit von 5 Jahren und 9 Monaten auf 30 Tage bedeuten.

Um die Entwicklung der Laufzeit genauer zu untersuchen, schaut man sich den Speedup-Graphen (Abb. 5.16) an. Der Speedup S für n Prozessoren ist definiert als Quotient aus serieller Laufzeit (T_S) und paralleler Laufzeit ($T_P(n)$):

$$S(n) = \frac{T_S}{T_P(n)} \tag{5.1}$$

Der Verlauf des tatsächlichen Speedup (blau) im Vergleich zum idealen Speedup (grau) zeigt, dass gute Werte erreicht werden. Schwankungen im Speedup selbst können durch die Hardware erklärt werden. So enthält jeder Rechenknoten 24 Prozessoren, die die Bandbreite des Knoten umso mehr beanspruchen, je mehr von ihnen aktiv sind. Daher zeigt der Speedup-Graph leichte Einbrüche bei ganzzahligen Vielfachen von 24. Andere Schwankungen können durch die allgemeine Auslastung des Systems und dadurch bedingte eventuelle Verzögerungen im Gesamt-I/O erklärt werden.

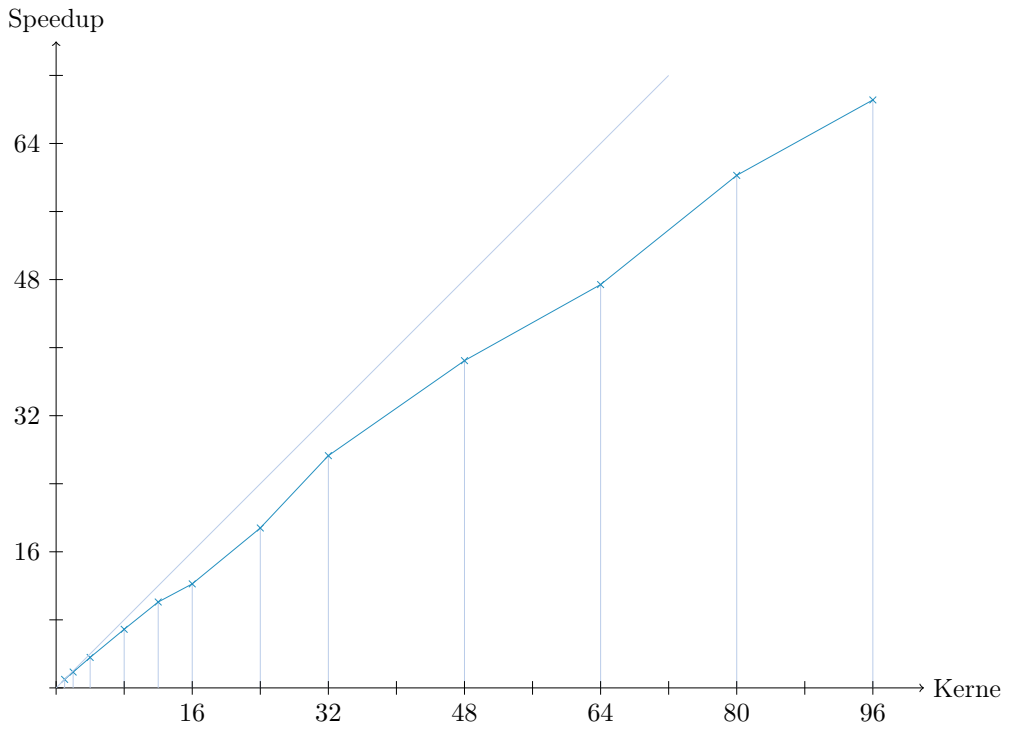


Abbildung 5.16.: Speedupkurve der gemessenen Laufzeiten der STA

Um eine Prognose zur weiteren Entwicklung des Speedup zu machen, ist es sinnvoll, die Effizienz E zu betrachten (Abb. 5.17). Diese ist definiert als Quotient aus Speedup ($S(n)$) und Prozessorzahl (n):

$$E(n) = \frac{S(n)}{n} = \frac{T_S}{n \cdot T_P(n)} \tag{5.2}$$

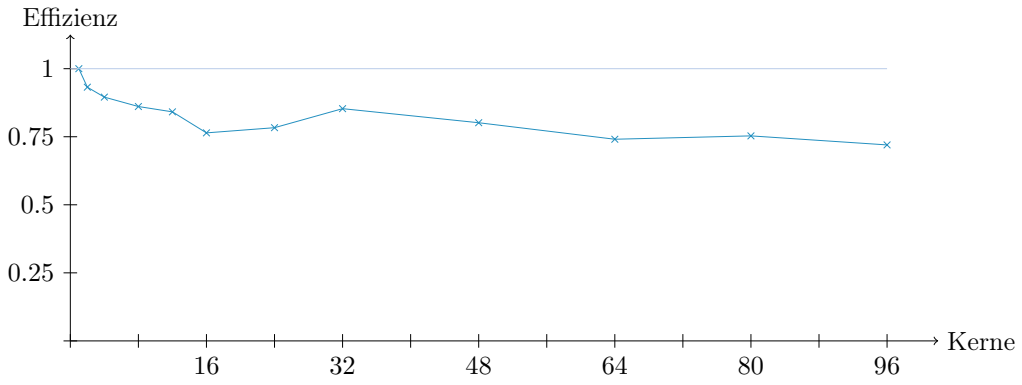


Abbildung 5.17.: Effizienz

Man erkennt dort, dass sich der Wert etwa bei 0.75 einpendelt und dann langsam absinkt. Das bedeutet, dass für 400 Prozessoren ein Speedup von etwa 250 bis 300 möglich wäre. Allgemein ist die Anzahl der einsetzbaren Prozessoren jedoch dadurch limitiert,

dass jeder Schnitt bei einer festen Streifenbreite von 100 Pixeln in maximal 1000 Streifen aufgeteilt werden kann. Das bedeutet, dass maximal 1000 Prozessoren eingesetzt werden können und im Idealfall, d.h. bei nur sehr langsam sinkender Effizienz, ein Speedup von etwa 600 bis 700 möglich ist. Dadurch würde die Gesamtverarbeitungszeit eines Hirnes auf etwa 72 Stunden sinken.

5.2.6. GPUs

Eine Analyse der Laufzeiten (Abb. 5.18) zeigt, dass die Berechnung von Direktion und Inklination mit knapp zwei Dritteln den größten Anteil der Berechnungen ausmacht und somit auch das größte Einsparpotential bietet.

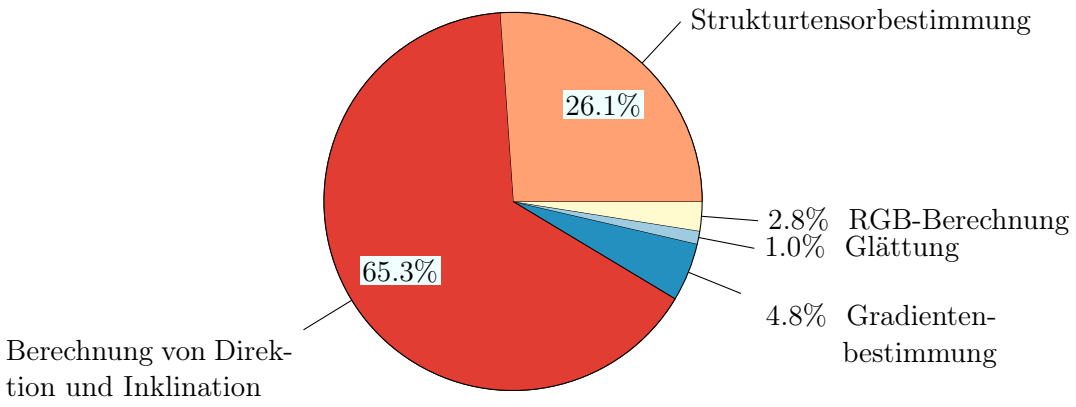


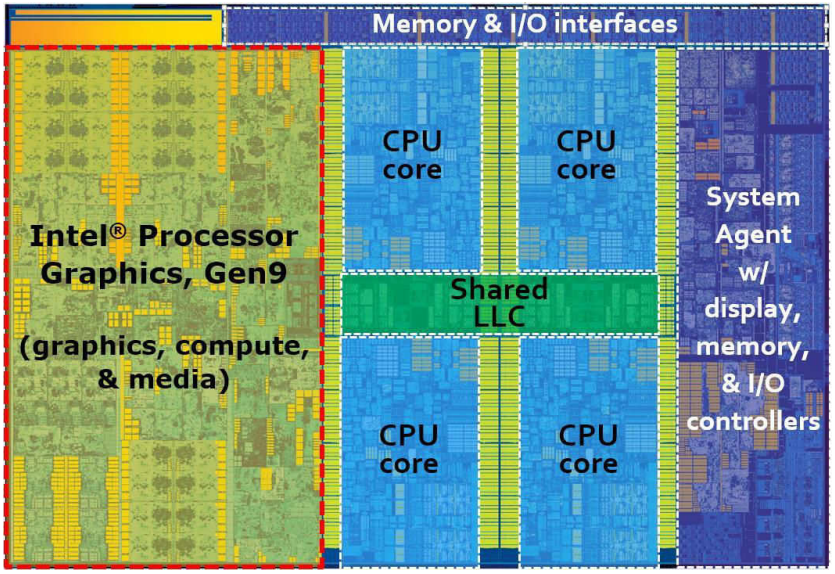
Abbildung 5.18.: Laufzeitanteile ohne GPU-Nutzung

Die Berechnung dieser beiden Winkel kann nach Bestimmung der Strukturtensoren durch das Lösen eines Eigenwert-/Eigenvektorproblems für jedes Pixel unabhängig stattfinden. Daher bietet es sich an, diese Teile des Programmes von der CPU auf die GPU auszulagern. Diese ist für massiv parallele Berechnungen ideal geeignet.

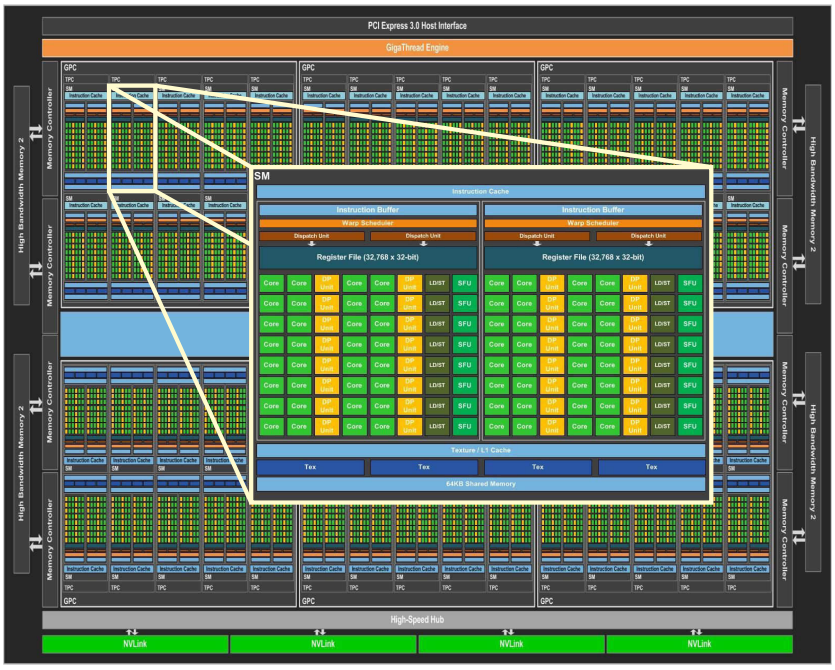
GPU-Architektur

Ursprünglich für die Berechnung des Bildschirminhaltes gedacht, haben sich Grafikkarten immer mehr zu einer Art Parallelrechner entwickelt. Sie haben den klassischen Prozessor in der Rechenleistung weit abgehangen (5300 GFLOPs [NVI] vs. ≈1000 GFLOPs [INT], jeweils doppelte Genauigkeit).

Im Gegensatz zum klassischen Prozessor (CPU), der verschiedenste Aufgaben wahrnehmen muss und daher einen komplexen Befehlssatz hat, haben Grafikprozessoren (GPUs) einen Primärzweck: In komplexen 3D-Anwendungen hohe Bildraten liefern. Dafür müssen möglichst viele Pixel des Bildes gleichzeitig berechnet werden können. Daher wird der Großteil der Chip-Fläche für tatsächliche Berechnungen genutzt, während in einer CPU viel Platz für Caches, Kontrolllogik und integrierter Grafik genutzt wird. Die Kernanzahl moderner GPUs (Abb. 5.19b) liegt dabei im vierstelligen Bereich, während CPUs (Abb. 5.19a) gerade erst in den zweistelligen Bereich vordringen.



(a) CPU-Aufbau



(b) GPU-Aufbau

Abbildung 5.19.: Schematischer Aufbau einer Intel i7-6700k CPU [INTEL] und einer NVIDIA Tesla P100 GPU [NVI]. Die grünen und gelben Rechtecke im gezoomten Bild stellen jeweils einen GPU-Rechenkern dar.

GPUs bieten daher allein durch die enorme Anzahl an Kernen ein riesiges Potential zur Parallelisierung, das durch Programmiertechniken wie CUDA auch für andere Zwecke als Bildausgabe nutzbar gemacht wird.

CUDA

Die von NVIDIA entwickelte *Compute Unified Device Architecture* (CUDA) bietet die Möglichkeit, von der Bildschirmausgabe unabhängige Berechnungen auf der GPU auszuführen.

Vereinfacht gesagt erfolgt die parallele Berechnung auf einer GPU in 5 Schritten (Abb. 5.20). Nachdem in einem ersten Schritt Speicherplatz auf der GPU reserviert wurde, müssen die zu verarbeitenden Daten in einem zweiten Schritt zur GPU kopiert werden. Im dritten und wichtigsten Schritt erfolgt die parallele Berechnung. Dafür wird nach dem Prinzip der Gebietszerlegung der sogenannte Berechnungskernel auf den unterteilten Daten parallel aufgerufen. Anschließend müssen die berechneten Ergebnisse wieder von der GPU zurückkopiert werden und der Speicher auf der GPU wieder freigegeben werden.

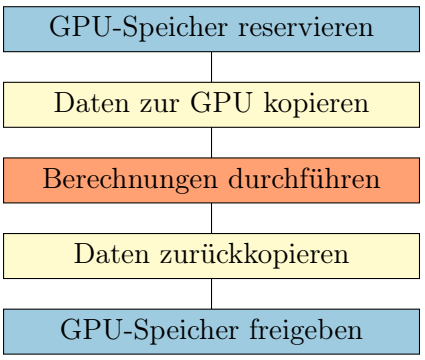


Abbildung 5.20.: Vorgehen bei Berechnungen auf einer GPU

Laufzeitverhalten

Im vorhandenen Anwendungsfall werden daher zur GPU-Parallelisierung die berechneten Strukturtensoren zur GPU kopiert und dort parallel die Eigenwerte und -vektoren bestimmt.

Zur Laufzeitmessung wird das Programm mit den in Abschnitt 5.2.5 beschriebenen Parametern und verschiedenen Kernanzahlen aufgerufen, diesmal mit Zuhilfenahme der vorhandenen GPUs zur Direktions- bzw. Inklinationsberechnung.

# Kerne	Laufzeit ohne GPU (s)	Laufzeit mit GPU (s)
1	29 765.9	9561.6
2	15 968.3	4976.5
4	8309.2	2539.0
8	4321.3	1281.1
12	2947.9	889.7
16	2434.4	707.2
24	1583.5	468.1
32	1090.3	375.6
48	773.7	260.0
64	627.8	205.9
80	494.0	159.1
96	430.7	138.9

Tabelle 5.3.: Laufzeiten für die Berechnung des Testdatensatzes mit 3×3×3-Glättungskern, 5×5×5-Gradientenkern, 7×7×7-Tensorkern und anschließender Berechnung des RGB-Bildes

Vergleicht man die Laufzeiten des Programms mit und ohne GPU-Nutzung (Tab. 5.3), so sieht man, dass die Nutzung der GPUs die Laufzeiten stark reduziert. Bei Nutzung eines Prozessorkerns sinkt die Laufzeit beispielsweise von 8 Stunden und 20 Minuten auf etwa 2 Stunden und 40 Minuten. Die Beschleunigung um den Faktor 3 entspricht an dieser Stelle dem maximal möglichen Speedup, der hier durch GPU-Nutzung erreichbar ist. Er entspricht nach dem Gesetz von Amdahl dem Kehrwert des nicht parallelisierbaren Anteiles eines Programmes, d.h. $\frac{1}{35\%} \approx 3$ [GA67].

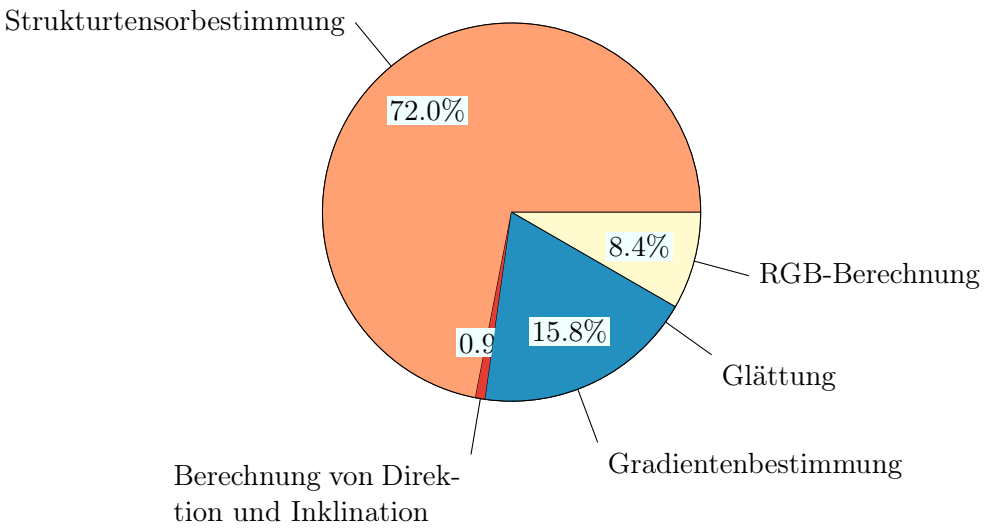


Abbildung 5.21.: Laufzeitanteile mit GPU-Nutzung

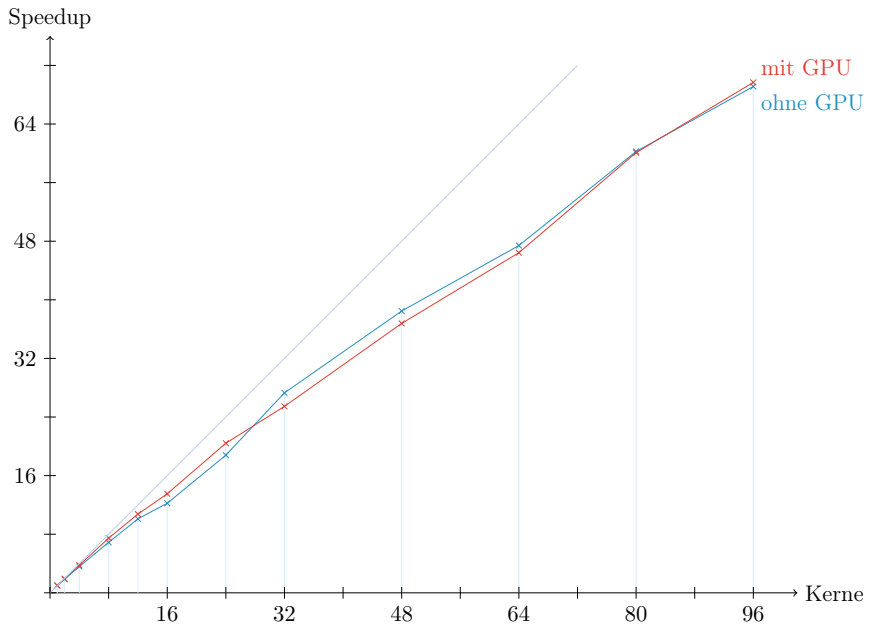


Abbildung 5.22.: Speedupkurven mit und ohne GPU-Nutzung im Vergleich

Eine erneute Analyse der Laufzeiten zeigt, dass der Laufzeitanteil der Berechnung von Direktion und Inklination bei GPU-Nutzung von etwa zwei Drittel auf nicht einmal 1% der bereits verminderten Laufzeit gesunken ist (Abb. 5.21).

Um eine etwas genauere Prognose der Laufzeiten für größere Prozessorzahlen machen zu können, schaut man sich auch an dieser Stelle noch einmal den Speedup an (Abb. 5.22). Insbesondere die Ähnlichkeit zum Verlauf des Speedup ohne GPU erlaubt eine wiederholte Prognose der Laufzeiten. Diese lässt sich wie vorher treffen, nur mit einer allgemeinen Drittelung der Laufzeit durch den Einsatz von GPUs. Somit sinkt die prognostizierte Laufzeit der Verarbeitung eines gesamten Gehirns auf etwa 24 Stunden.

6. Zusammenfassung und Ausblick

In dieser Arbeit wurde die 3D-Strukturtensoranalyse auf hochauflösende Hirnbilder angewendet, die mit Hilfe von Polarized Light Imaging (PLI) aufgenommen wurden. PLI ist eine Aufnahmetechnik, die auf der Doppelbrechung von Licht an myelinisierten Nervenfasern beruht. Die Bilder werden dabei als Gesamtbild eines Schnittes mit dem Large Area Polarimeter (LAP) oder als einzelne Kacheln mit dem Polarisationsmikroskop (PM) in sehr hoher Auflösung aufgenommen. Durch Anwendung des PLI-Rekonstruktionsworkflows werden aus den so aufgenommenen Bildern Informationen über die räumliche Orientierung der Fasern extrahiert. Dazu gehören sowohl der Winkel einer Faser in der Schnittebene und der Winkel zwischen einer Faser und ihrer Projektion auf die Schnittebene (Kap. 2).

Um die Korrektheit der berechneten Winkel zu bestätigen, können sie durch alternative Methoden berechnet werden. Eine Methode dafür ist die Strukturtensoranalyse (STA). Vor der Anwendung auf PLI-Bilder wurde die STA in einem ersten Schritt für den zweidimensionalen Fall erklärt und erfolgreich auf ein allgemeines Beispielbild angewendet. Im Anschluss daran wurde sie auf drei Dimensionen erweitert. Hierfür wurden alle nötigen Anpassungen an den einzelnen Schritten erläutert und anschließend die 3D-STA ebenfalls erfolgreich auf ein konstruiertes Testbild angewendet (Kap. 3).

Vor der tatsächlichen Anwendung auf PLI-Bilder mussten an der STA noch kleinere Änderungen vorgenommen werden. Dazu gehörten die Nutzung der elliptischen Gauß-Funktion und die Erweiterung der Operatoren zur Bestimmung von Gradienten auf eine beliebige Größe.

Bei der Anwendung selbst traten einige Hindernisse auf. So ist bei LAP-Bildern die unterschiedliche Größe von Strukturen ein großes Problem. Je nach Größe des Gebietes für den Strukturtensor wird entweder eine große Struktur nicht vollständig erkannt, oder aber feine Strukturen werden stark verwaschen dargestellt. Bei PM-Bildern, bei denen auch in großen Strukturen einzelne Fasern erkennbar sind, tritt dieses Problem nicht auf. Dafür ist es jedoch in beiden Fällen nicht möglich, eine 3D-STA durchzuführen, bzw. einen korrekten Raumwinkel zu extrahieren. Grund hierfür ist eine nicht ausreichende Qualität der Registrierung der einzelnen Schnitte und im Falle des PM eine zu geringe Auflösung in z-Richtung, d.h. eine zu hohe Schnittdicke (Kap. 4).

Anschließend wurde die Implementierung beschrieben. Hierbei wurden inbesondere die Art der Speicherung der Daten im speziell für große Datenmengen entwickelten Format HDF5 und die Parallelisierung der Algorithmen dargestellt. Das Programm wurde erst auf CPUs und anschließend auch auf GPUs parallelisiert, um vertretbare Laufzeiten zu erreichen und die vorhandene Infrastruktur bestmöglich auszunutzen (Kap. 5).

In einer zukünftigen Version wäre es möglich, das angesprochene Problem der Erkennung unterschiedlich großer Strukturen durch die mehrmalige Anwendung der STA mit unterschiedlich großen Tensorkernen zu lösen. Dabei könnte etwa anhand des besten

Kohärenzwertes entschieden werden, welche Richtung das endgültige Ergebnisbild in einem Punkt hat. Eine andere Möglichkeit wäre, das Bild mit dem kleinsten Tensorkern als Basis zu nehmen und nur Richtungen von Pixeln, die einen gewissen Kohärenzwert unterschreiten, durch Richtungen aus gröberen Bildern auszutauschen, sofern ein besserer Kohärenzwert vorliegt.

Ebenso könnte man die Ergebnisse der Strukturtensoranalyse in eine wiederholte Durchführung der Registrierung einfließen lassen, um bessere Ergebnisse zu erzielen.

Hinsichtlich der Implementierung wäre es möglich, trotz der theoretisch fehlenden Notwendigkeit einmal das Master-/Worker-Schema zu implementieren, um unterschiedliche Dateizugriffszeiten und Ein-/Ausgabegeschwindigkeiten der Prozessorkerne abzufangen. Insbesondere bei einer sehr großen Anzahl an Prozessorkernen könnte der Speedup dadurch noch gesteigert werden. Um die Gesamtlaufzeit der STA eines gesamten Hirnes deutlich unter 24 Stunden zu senken, könnten weitere Möglichkeiten der Optimierung des Programmcodes untersucht werden, sowohl im Algorithmus selbst, als auch durch weitere Parallelisierung. Dafür könnte beispielsweise eine Auslagerung der zu berechnenden Faltungen auf die GPU überprüft werden.

A. Koordinatensysteme

Betrachtet man die Lage der einzelnen Winkel, lässt sich daraus ein Koordinatensystem erzeugen. So liegt beispielsweise der Direktionswinkel innerhalb der Ebene so, dass ein Winkel von 0° einer von links nach rechts laufenden Faser entspricht. Der Winkel verläuft im mathematisch positiven Sinne (Abb. A.1).

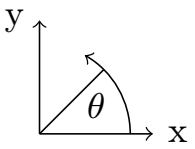


Abbildung A.1.: Lage der Direktion

Der Inklinationswinkel ist positiv, wenn die Faser zum darüber liegenden Schnitt zeigt (Abb. A.2). Das bedeutet, dass die positive z-Achse ebenfalls nach oben zeigt. An dieser Stelle fällt auf, dass sich das Koordinatensystem der Winkel von dem der Daten unterscheidet. Denn während die positve z-Achse der Fasern nach oben zeigt, zeigt die der Schnitte nach unten. Das liegt daran, dass die Schnitte von oben nach unten erzeugt, verarbeitet und nummeriert abgespeichert werden.

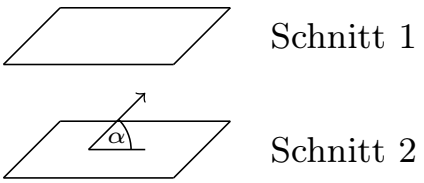


Abbildung A.2.: Lage der Inklination

Ebenso unterscheiden sich auch x- und y-Achse von den Datenkoordinaten. Während bei den Datenkoordinaten erst die Zeile und dann die Spalte adressiert wird, arbeitet man bei Vektoren erst mit der x- und dann mit der y-Achse. Die x-Achse entspricht hier aber der Spalten-Achse, die y-Achse der negativen Zeilen-Achse (Abb. A.3).

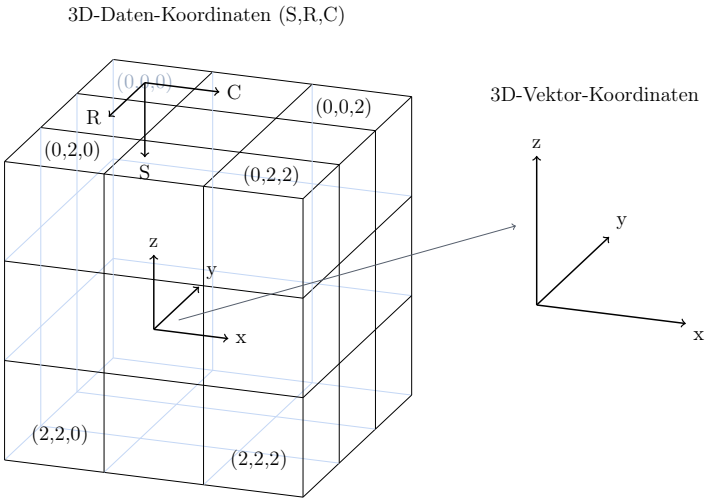


Abbildung A.3.: Gegenüberstellung von Daten- und Vektorkoordinatensystemen

B. Supercomputer

Jülich Research on Exascale Cluster Architectures (JURECA)

Der Supercomputer JURECA steht allen Mitarbeitern im Juelich Supercomputing Centre für Simulationen zur Verfügung [JURE].

Rechenknoten

- 1872 Rechenknoten
  - Prozessor: 2 Intel Xeon E5-2680 v3 Haswell 12-Kern Prozessoren (2.5 GHz)
  - Hauptspeicher: DDR4 2133 MHz
    - 1605 Knoten mit 128 GiB
    - 128 Knoten mit 256 GiB
    - 64 Knoten mit 512 GiB
  - GPU: 2 NVIDIA K80 (Tesla); je 4992 CUDA Kerne, 2x24 GiB GDDR5 Hauptspeicher (auf 75 Knoten)

Visualisierungsknoten

- 12 Visualisierungsknoten
  - Prozessor: 2 Intel Xeon E5-2680 v3 Haswell 12-Kern Prozessoren (2.5 GiHz)
  - Hauptspeicher: DDR4 2133 MHz
    - 10 Knoten mit 512 GiB
    - 2 Knoten mit 1024 GiB
  - GPU: 2 NVIDIA K40 (Tesla); je 2880 CUDA Kerne, 2x12 GiB GDDR5 Hauptspeicher

Gesamtsystem

- 45216 Prozessorkerne
- 1.8 (CPU) + 0.44 (GPU) Petaflops Peak Performance
- 100 GiB pro Sekunde Speicheranbindung an JUST [JUST]

Literaturverzeichnis

[AA+11] Markus Axer, Kathrin Amunts, David Grässel, Christoph Palm, Jürgen Dammers, Uwe Pietrzyk, Karl Zilles – A novel approach to the human connectome: Ultra-high resolution mapping of fiber tracts in the brain, *Neuroimage*, 15.01.2011; 54(2):1091-1101.

[AC+09] `http://www.ncbi.nlm.nih.gov/pubmed/19226510`, 2009. Letzter Zugriff: 18.07.2017

[AS78] Alvy Ray Smith – Color gamut transform pairs, *Computer Graphics*, 12:12 - 19, 1978.

[AS+16] Markus Axer, Sven Strohmer, David Gräßel, Oliver Bücker, Melanie Dohmen, Julia Reckfort, Karl Zilles, Katrin Amunts – Estimating Fiber Orientation Distribution Functions in 3D-Polarized Light Imaging, *Frontiers in Neuroanatomy*, 2016; 10: 40.

[ATAN] `http://www.cplusplus.com/reference/cmath/atan/`. Letzter Zugriff: 21.06.2017

[ATAN2] `http://www.cplusplus.com/reference/cmath/atan2/`. Letzter Zugriff: 21.06.2017

[BG87] J. Bigun, G. Granlund – Optimal orientation detection of linear symmetry, Proceedings of the IEEE First International Conference on Computer Vision, London (1987), S. 433-438

[BMBF] `http://www.bmbf.de/press/3413.php`, 2013. Letzter Zugriff: 16.05.2017

[CAT] `http://www.publicdomainpictures.net/view-image.php?image=181282`, Auflösung 1920x1280, Letzter Zugriff: 22.05.2017

[DA+10] Jürgen Dammers, Markus Axer, David Grässel, Christoph Palm, Karl Zilles, Katrin Amunts, Uwe Pietrzyk – Signal enhancement in polarized light imaging by means of independent component analysis, *Neuroimage*, 49:1241 - 1248, 2010.

[DS15] Daniel Schmitz – Comparison of Diffusion Tensor Imaging and 3D Polarized Light Imaging for the investigation of the human brain's nerve fiber architecture, Masterarbeit an der Heinrich-Heine Universität Düsseldorf, S.20, 2015

[GA67] Gene Amdahl – Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities, AFIPS Conference Proceedings, Band 30, 1967, S. 483–485.

[HDF5] `https://support.hdfgroup.org/HDF5/doc/UG/HDF5_Users_Guide-Responsive%20HTML5/index.html`, Letzter Zugriff: 28.06.2017.

[HSV] `https://de.wikipedia.org/wiki/HSV-Farbraum#/media/File:HSV_cone.png`, 2006. Letzter Zugriff: 02.06.2017

[INTEL] `https://software.intel.com/sites/default/files/managed/c5/9a/The-Compute-Architecture-of-Intel-Processor-Graphics-Gen9-v1d0.pdf`, Letzter Zugriff: 31.07.2017.

[INT] `https://newsroom.intel.com/editorials/new-intel-core-x-series-processors-scale-accessibility-and-performance-go-extreme/`, Letzter Zugriff: 31.07.2017.

[JP70] J. M. S. Prewitt – Object enhancement and extraction, Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld, Eds., New York: Academic Press, 1970, S. 75-149, vorgestellt beim Stanford Artificial Intelligence Project (SAIL), 1968.

[JURE] `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/Configuration/Configuration_node.html`, 2016. Letzter Zugriff: 04.05.2017

[JUST] `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Datamanagement/OnlineStorage/JUST/JUST_node.html`, 2016. Letzter Zugriff: 27.06.2017

[M14] Andreas Müller – Optimierung und Parallelisierung der Inklinationsberechnung von Nervenfasern im PLI-Rekonstruktionsworkflow, Seminararbeit FH Aachen, 2014

[MF67] Michael J. Flynn – Very High-Speed Computing Systems. *Proceedings of the IEEE*, 54:1901 - 1909, 1967.

[MPI] `http://www.mcs.anl.gov/research/projects/mpi/`, Letzter Zugriff: 25.07.2017.

[NVI] `https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf`, Letzter Zugriff: 31.07.2017.

[OMP] `http://www.openmp.org/`, Letzter Zugriff: 25.07.2017.

[OriJ] `http://bigwww.epfl.ch/demo/orientation/`, Letzter Zugriff: 12.07.2017.

[S5x5] `https://software.intel.com/en-us/node/504204`. Letzter Zugriff: 21.06.2017

[SF68] Irwin Sobel, Gary Feldman – A 3x3 Isotropic Gradient Operator for Image Processing

[SHF11] Robert F. Schmidt, Manfred Heckmann, Florian Lang – Physiologie des Menschen: mit Pathophysiologie, S. 72, 31. Auflage, Springer-Verlag, 2011.

[SU05] Scott E. Umbaugh – Computer Imaging: Digital Image Analysis and Processing, CRC Press, 2005, S. 139.

[S+15] Schober M. et al. – Reference Volume Generation for Subsequent 3D Reconstruction of Histological Sections. In: Handels H., Deserno T., Meinzer HP., Tolxdorff T. (eds) Bildverarbeitung für die Medizin 2015. Informatik aktuell. Springer Vieweg, Berlin, Heidelberg

[W13] Anna Westhoff – GPU-accelerated Segmentation of high-resolution Human Brain Images acquired with Polarized Light Imaging, Masterarbeit an der FH Aachen, 2013