Rekonstruktion und Vermessung der Geometrie von Neuronen-Zellkernen

Diplomarbeit
bei Prof. Dr. Gabriel Wittum

vorgelegt von Sean Gillian Queisser
am Fachbereich der Mathematik der
Universität Heidelberg

Heidelberg, 30. November 2005

Great music lies at the intersection of
mathematics and spiritual education
-Ralph W. Emerson

Inhaltsverzeichnis

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel verwendet; alle Stellen, die dem Wortlaut oder Sinne nach anderen Werken entnommen sind, wurden durch Angabe der Quellen als Entlehnung deutlich gemacht.

Heidelberg, 30. November 2005: .....

Einleitung

Die Schönheit der Mathematik liegt mitunter darin, dass es sie einem ermöglicht in vielen Bereichen der Wissenschaft tätig zu werden. Aus diesem Grund kam ich – zunächst als wissenschaftliche Hilfskraft am *Interdisziplinären Zentrum für wissenschaftliches Rechnen* (IWR) – in Kontakt mit neurobiologischen Forschungsthemen. Eines dieser Themen, die *Gentranskription*, lieferte die Grundlage für diese Arbeit.
Um diesen Begriff verständnisvoller zu machen und zu erklären, woher der Anreiz kam, der Gentranskription meine Arbeit zu widmen, beginne ich etwas weiter oben im biologischen Kontext, im menschlichen Gehirn.

Eine Vielzahl von Neuronen bilden ein gigantisches Netzwerk, die Steuerzentrale des menschlichen Körpers. Doch schon eine einzelne Komponente dieses Netzwerks weist einen Aufbau zur Bewältigung vieler komplexer biologischer und biochemischer Vorgänge auf.
Grob unterteilt besitzt eine solche Zelle einen informationsaufnehmenden und -weiterleitenden Bereich (*Dendriten*, *Axon*) und einen informationsverarbeitendes Zentrum, der Zellkörper, in dem sich neben vielen anderen *Organellen* der Zellkern befindet.

Als Speicherort für die *DNA*, übernimmt der Zellkern viele regulierende Aufgaben, u.a. die Regulierung der Gentranskription, welche einen Schritt in der Produktionskette von DNA zu Proteinen beschreibt. Zur Gentranskription sei an dieser Stelle soviel gesagt: Kalziumdiffusion in und aus dem Zellkern reguliert die Gentranskription. Diese These wird durch etliche Erkenntnisse, wie sie in Bading[2], Echevarria[7], Hardingham[4] und Chawla[6] enthalten sind, gestützt.

Biologisches Interesse besteht also darin, die Diffusion von Kalziumionen am Zellkern zu verstehen. Neueste mikroskopische Untersuchungen der Morphologie des Neuronenzellkerns weisen morphologische “Sonderheiten” auf.
Der Zellkern – bis dato in allen medizinischen und biologischen Lehrbüchern als kugelförmig beschrieben – besitzt *Invaginationen* der Zellkernmembran. Eine Antwort darauf, ob es sich um *tubuläre* Invaginationen wie es Fricker[5]

vorschlägt und/oder um "Einstülpungen" handelt, liefert diese Arbeit.
Es ist anzunehmen, dass sich der Diffusionsprozeß durch diese Invaginationen im Vergleich zu einer kugelförmigen Geometrie stark ändern wird.

Berechtigte Fragen an dieser Stelle sind:

1. Wie sieht die dreidimensionale Geometrie des Neuronenzellkerns aus?
2. Wie ändert sich seine Oberfläche bzw. sein Volumen aufgrund solcher Invaginationen?
3. Was sind die Eigenschaften der Diffusion über die Zellkernoberfläche?

Diese Fragen gestalten die Sprungstelle von der Neurobiologie zur Mathematik und riefen damit ein Projekt zwischen dem Interdisziplinären Zentrum für Neurobiologie (IZN) der Universität Heidelberg unter Leitung von Hilmar Bading und der Abteilung *Simulation in Technology* (SiT) unter Leitung von Gabriel Wittum ins Leben.
Sie liefern gleichzeitig die stufenweise Vorgehensweise des Projekts:

1. **Rekonstruktion** der Geometrie von Neuronenzellkernen.
2. **Vermessung** der Oberfläche und des Volumens des Zellkerns.
3. **Mathematische Simulation** der Diffusion von Kalzium auf der rekonstruierten Geometrie des Zellkerns.

Die Rekonstruktion und Vermessung sind Inhalt dieser Arbeit.

In Kapitel 1 werden die biologischen Grundlagen zum Verständnis der Gentranskription geschaffen.

Die Mikroskopie ist der erste Schritt in Richtung 3D-Visualisierung des Zellkerns. Welche Art der Mikroskopie und dessen optimale Einstellungen für die dann folgende Mathematik verwendet werden muss, ist Thema von Kapitel 2.

Kapitel 3 widmet sich den mathematischen Grundlagen der Bildaufarbeitung. Dienlich für den Aspekt Bildverarbeitung in den Schritten

- **Filterung**
- **Segmentierung**
- **Visualisierung**

war das Projekt *NeuRA* (*Neuron Reconstruction Algorithm*), geführt von Gabriel Wittum und Bert Sakmann zur automatischen Rekonstruktion der Morphologie von Neuronen.
Der dazu von Roland Schulte implementierte Filter wird für die Bildaufarbeitung der Mikroskopiedaten von Neuronenzellkernen verwendet.

Kapitel 4 liefert Einblick in den Aufbau des Filters. Da dieser für die Geometrie von Dentriten entwickelt wurde, wird der Filter in Kapitel 5 für die Zellkerngeometrie angepasst, numerisch optimiert und auf Testgeometrien sowie auf reellen Geometrien getestet.

Nächster Schritt ist die Segmentierung des Bildes, also die Trennung von Bild und Hintergrund. In Kapitel 6 werden mögliche Formen der Segmentierung besprochen und ein passender Segmentierungsalgorithmus durch Tests auf gefilterten Bilddaten ermittelt.

In Kapitel 7 werden die Verfahren zur Vermessung und Visualisierung von Zellkernen entwickelt.
Um Aussagen über Oberfläche und Volumen machen zu können wurde eine Testreihe mit 20 Zellkernen durchgeführt. Die Testreihen und Visualisierungsergebnisse werden in Kapitel 8 vorgestellt.

Kapitel 1

Die Gehirnzelle

1.1 Aufbau der Zelle

> Die Zelle ist von einer Membran umschlossen und enthält membranumgrenzte Organellen; die Membranen bestehen aus einer Lipiddoppelschicht und eingelagerten Proteinen.
> Schmitt[16]

1.1.1 Die Membran

Lipide sind langkettige Moleküle die in einen *hydrophilen* und *hydrophoben* Bereich unterteilt werden können. Die sogenannten Phospholipide bilden zu 40% die Membranen. Die restlichen 60% der Membran sind Proteine.
Eine oder mehrere lange Kohlenwasserstoffketten bilden den hydrophoben (wasserabweisenden) Baustein. Ans Ende dieser Kette ist ein hydrophiler (wasserfreundlicher) Teil gebunden, im Fall der Phospholipide ist dies ein *Phosphatidylamin*. Das *Phosphatidylcholin* ist das am häufigsten vertretene Lipid in einer Membran. Dieses besitzt einen hydrophilen Kopf mit zwei langen Kohlenwasserstoffketten.
Der Regel folgend, dass sich hydrophile an hydrophile Komponenten anlagern, genauso hydrophobe Gruppen an hydrophobe binden, bilden Lipide in einem hydrophilen Medium (z.B. Wasser) Doppelschichten, die *Bilipidschichten*, aus. Der hydrophile Kopf der Lipide ist dabei dem Medium zugewandt, während die hydrophoben Ketten eine wasserabweisende Ölphase bilden (Abb. 1.1).
In die Membran eingebettet sind Moleküle, wie z.B. Cholesterin und unterschiedliche Proteine. Die Proteine übernehmen dort spezifische Aufgaben für die Informationsverarbeitung in der Zelle.

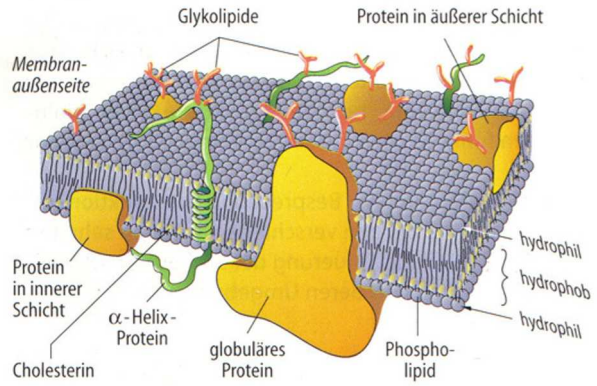

Abbildung 1.1: Schematischer Aufbau einer Plasmamembran (Schmitt[16])

1.1.2 Das Zellinnere

Der Innenraum der Zelle – umschlossen von der Zellmembran – ist gefüllt mit Zytosol. Zytosol besteht zu 20 % aus Eiweiß und ist von gelantiner Konsistenz. Darin gelöst sind anorganische und organische Ionen sowie verschiedene Organellen, wie es in Abbildung 1.2 schematisch dargestellt ist.
Die frei beweglichen Ionen spielen zum Auslösen von Impulsen in der Zelle eine wichtige Rolle. Die Konzentration der Ionen – hauptsächlich Kalzium- und Natriumionen – ist im *Intrazellulärraum* anders als im *Extrazellulärraum*. Dadurch entsteht ein Konzentrationsgradient, der über die Membran ausgeglichen werden kann. Der beim Ausgleich entstehende Ionenfluss ist einer der zentralen Vorgänge in der Informationsvermittlung der Zelle.
Die durch Kalziumionen gesteuerte Informationscodierung und Informationsvermittlung spielt bei der Steuerung der Gentranskription eine zentrale Rolle. Wir gehen deshalb in Abschnitt 1.5 näher auf diese Thematik ein. Doch zunächst wollen wir uns den verschiedenen Mechanismen des Stoffaustauschs widmen.

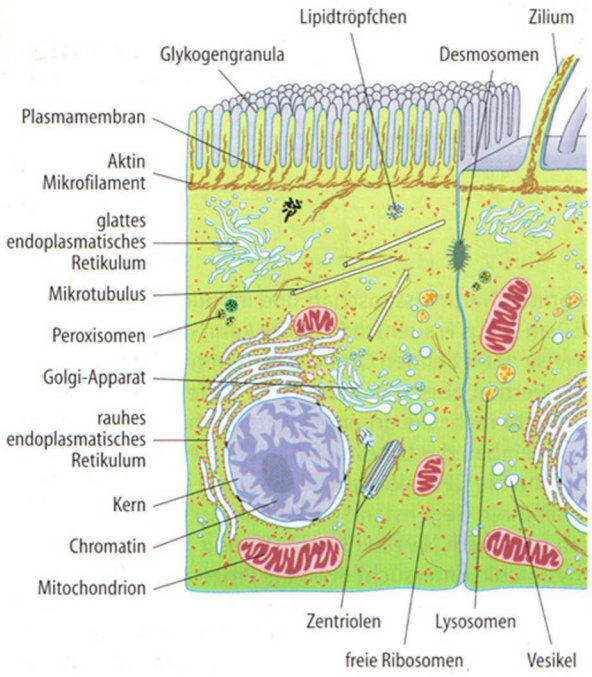

Abbildung 1.2: Schematischer Aufbau einer Zelle (Schmitt[16])

1.2 Stoffaustausch und Steuerung

Der *Stoffaustausch* bezeichnet den hochdynamischen Gleichgewichtszustand einer Zelle mit ihrer Umgebung und analog dazu den Zustand zwischen dem Inneren der Zelle und den eingeschlossenen Organellen (z.B. dem Zellkern). Der Austausch von Ionen findet auf zwei verschiedene Weisen statt:

Diffusion:

Diffusion ist ein passiver Stoffaustauschprozeß. Grundlage der Diffusion ist das Fick'sche Diffusionsgesetz (siehe Abschnitt 3.3.1) dem frei bewegliche Moleküle genügen.
Der Austausch von Ionen findet über Transportproteine statt (Schmitt[16]). Die von den Transportproteinen gebildeten Kanäle sind ionenspezifisch. Deshalb besitzt die Membran für jedes Molekül spezifische Kanäle.

Konzentrationsunterschiede zwischen Intra- und Extrazellulärraum werden durch den Diffusionsprozeß ausgeglichen. Messungen weisen jedoch ein sogenanntes *Ruhepotential* von ca. -70 mV nach. Dieses wird durch den zweiten Austauschprozeß aufrecht erhalten.

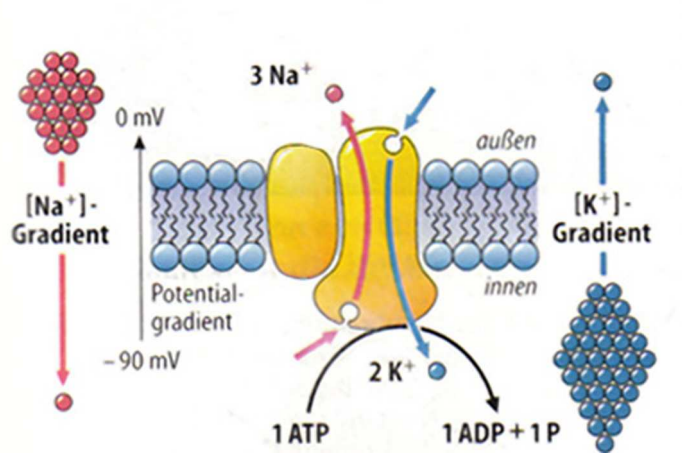

Abbildung 1.3: Kaliumspezifischer Kanal (Schmitt[16])

Aktiver Transport:

Ionenpumpen arbeiten entgegen dem *Konzentrationsgradienten*. Abbildung 1.3 zeigt den schematischen Aufbau einer Natrium/Kalium-Pumpe. Da die Pumpe dem Gradienten entgegen arbeitet, wird Energie verbraucht (ATP wird zu ADP und Phosphat umgesetzt).
Die in die Membran eingebetteten Proteine, welche als Ionenpumpen fungieren, sind also in der Lage das Membranpotential zu verändern. Dieser Vorgang dient zur Steuerung von Zellfunktionen:

> Steuernde Informationen werden innerhalb der Zelle durch sekundäre Botenstoffe (second messengers), wie Erhöhung der Kalzium-Konzentration, zyklisches Adenosinmonophosphat (cAMP), Inositotriphosphat, Diacylglycerin und ähnliche Stoffe weitergeleitet. (Schmitt[16])

Die nächste Frage die geklärt werden muss, ist *wie* und unter *welchen Bedingungen* Information in der Zelle weitergeleitet und verarbeitet wird.

1.3 Informationscodierung

Wie in Abschnitt 1.2 schon erwähnt, besitzt eine Nervenzelle ein Membranruhepotential von ca. -70 mV. Durch schnelle *Depolarisation* findet ein Ionenfluss (Kalium- bzw. Kalziumionen) statt, der nach Erreichen eines maximalen Flusses abklingt und sich schließlich wieder das Ruhepotential einstellt. Dieser Vorgang wird als *Aktionspotential* bezeichnet. Abbildung 1.4 zeigt die Graphen verschiedener Aktionspotentiale.

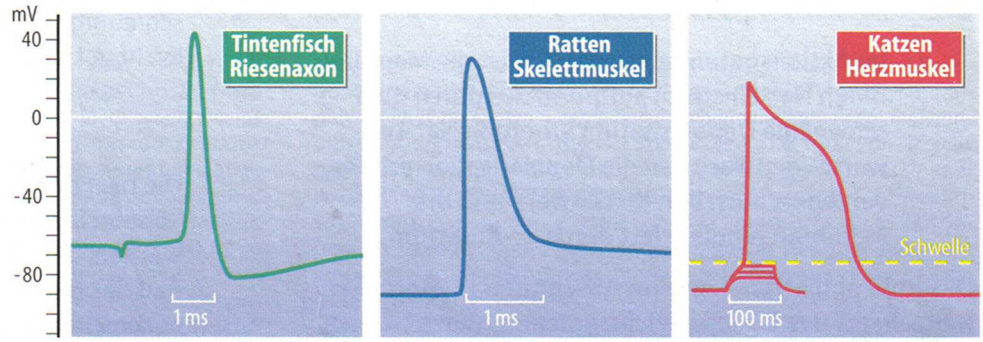

Abbildung 1.4: Verschiedene Aktionspotentiale (Schmitt[16])

Man trennt das Aktionspotential in eine *Depolarisationsphase* (von Beginn der Zeitmessung bis zum Erreichen des Maximum) und *Repolarisation* (vom Maximum bis zur Wiedereinstellung des Membranruhepotentials).

Wichtig: Ein Aktionspotential läuft nur dann ab, wenn eine genügend hohe Depolarisation stattfindet.

Das Aktionspotential erzeugt eines Stromfluss in dem zum einen Information für die Zelle (in unserem Fall für den Zellkern) codiert ist, zum anderen diese Information weitergeleitet wird.

1.4 Der Zellkern

1.4.1 Aufbau und Aufgaben des Zellkerns

Der *Zellkern* (engl.: *cell nucleus*) ist eine Organelle der Zelle (Abb. 1.5). Wir wollen an dieser Stelle den Aufbau und die unterschiedlichen Aufgaben des Zellkerns erläutern.

Aufbau des Zellkerns:

Anders als die in Abschnitt 1.1.1 beschriebene Membran der Zelle, besteht die *Kernhülle* (engl.: *nuclear envelope*) aus einer doppelten Membran. Die äußere Membran ist die Fortsetzung des *endoplasmatischen Retikulums*. Verknüpft mit der inneren Membran ist ein zweidimensionales Netzwerk, vergleichbar mit einem Stoffgewebe. Dieses besteht unter anderem aus *Lamin B*, einem faserförmigen Protein (Lodish[14]). Das Protein-Netzwerk dient der Strukturerhaltung der Kernhülle.
Die Membran trennt also den Innenraum des Zellkerns vom äußeren

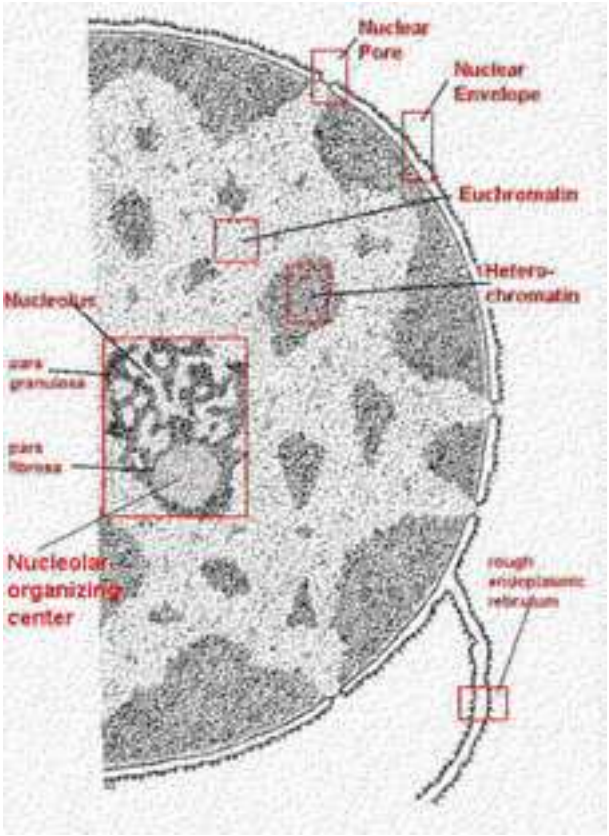

Abbildung 1.5: Schema eines Zellkerns (http://cellbio.utmb.edu/cellbio/nucleus.htm)

Zytoplasma, lässt jedoch spezifischen Molekültransport über feine *Kernporen* in der Membran zu. So lässt sich Information und Energie (ATP) in und aus dem Kern transportieren.
Im Kern befindet sich der *Nucleolus* der die *DNA*, *RNA*, *Ribosomen* sowie *Proteine* beherbergt.
Neben dem Nucleolus existiert noch das *Chromatin* innerhalb der Kernhülle. Darin enthalten sind die sogenannten *Chromosomen* – kleine Pakete aus DNA und Proteinen.

Aufgaben des Zellkerns:

Der Zellkern übernimmt eine Vielzahl von Aufgaben, unter anderem

- Ermöglichung der Zellteilung
- Transport von regulierender Information und Genprodukten per Diffusion über Membranporen
- Codierung von Information für Proteine

- Aufbau von Ribosomen
- DNA-Reproduktion
- Gentranskription

1.5 Gentranskription

Wenden wir uns nun dem Thema Gentranskription, das diesem Projekt zu Grunde liegt, zu. Zur Produktion von Proteinen wird die DNA zunächst in RNA (*Ribonucleinsäure*) umgewandelt. Anschließend wird die RNA in Proteine übersetzt (Abb. 1.6).
Der erste Schritt wird als *Gentranskription* bezeichnet, der zweite als *Translation*.
Wir stellen uns in diesem Abschnitt folgende Fragen:

1. Welche Rolle spielt Kalzium bei diesem Prozeß?
2. Wieso soll die Diffusion von Kalzium mathematisch simuliert werden?

Im Folgenden wird die Wichtigkeit von Kalzium im Rahmen der Gentranskription und die Vielfältigkeit der Wirkungsbereiche herausgearbeitet.

1.5.1 Die Kalziumhypothese

In Neuronen regulieren Kalziumionen die Gentranskription, die durch synaptische Aktivität ausgelöst wird. Die Zustände und der Verlauf der neuronalen Aktivität werden durch das ausgeschüttete Kalzium codiert (Abb. 1.7). Dieser Code trägt die Information des Kalziumeintritts in den Kern und die Amplitude des Ionenstroms sowie die Dauer und räumliche Eigenschaften des Kalziumflusses. Mit Hilfe der Kalziumverschlüsselung werden die spezifischen *firing patterns* des Neurons in Gentranskriptionsaufgaben übersetzt (Bading[2]).
Bading[2] stellt folgende Hypothese auf:

> Der durch Synapsen aktivierte Kalziumfluss in den Kern nimmt langanhaltende Veränderungen in neuronalen Funktionen vor, welche die Gentranskription beeinflussen.

1.5.2 Wirkungsbereiche der Kalziumdiffusion

Kalzium ist ein *second messenger* (siehe Schmitt[16]) und ist in praktisch allen Zellen vertreten. Man unterscheidet dabei zwischen *nuclear Calcium*, dem Kalziumanteil im Zellkern und dem *cytoplasmic Calcium*, dem Kalzium im Zytoplasma. Kalzium im Kern reguliert den *Proteintransport* über die

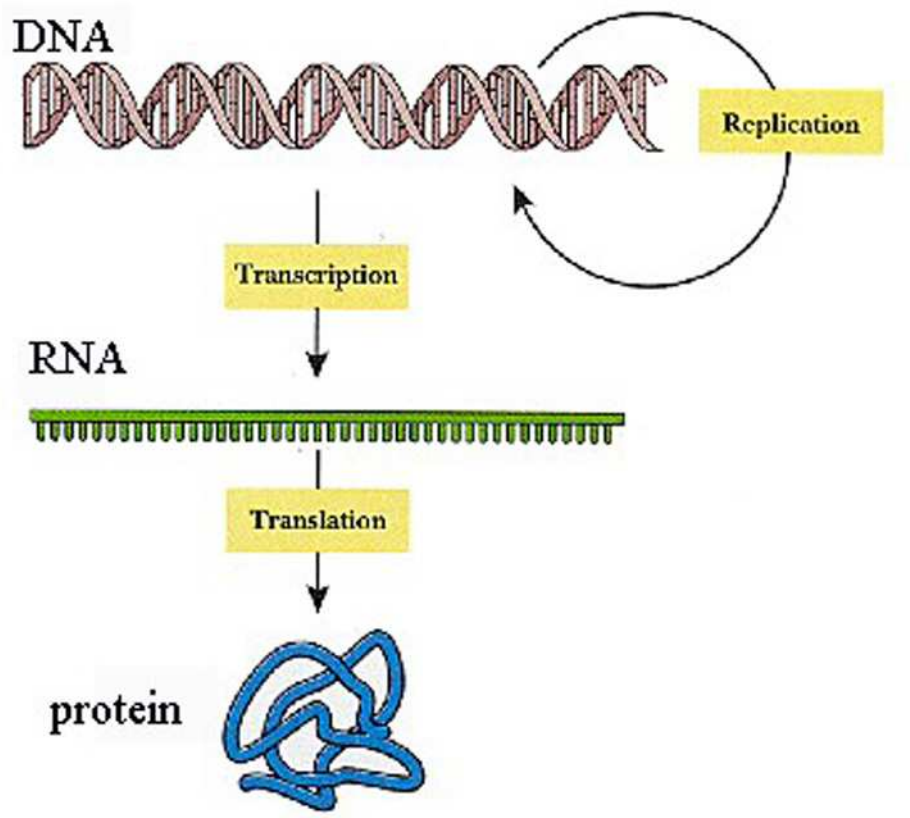

Abbildung 1.6: Gentranskription (http://cellbio.utmb.edu/cellbio/ribosome.htm)

Kernhülle (Stehno-Bittel[13] & Perez-Terzic[3]) sowie die Transkription einiger Gene (Chawla[6] & Hardingham[11]).
Kernkalzium wird über einen bestimmten Mechanismus kontrolliert. In Echevarria[7] wird vermutet, dass Kalziumdiffusion – vom Zytoplasma über die Kernhülle in den Zellkern – der regulierende Vorgang dieses Mechanismus ist.
Bading[2] stellt folgende wichtige kalziumabhängige Punkte zusammen:

1. **Induktion von Gentranskription ist maßgeblich für langanhaltende adaptive Veränderung des Nervensystems.**

   Versuche zeigen, dass Informationen und Erinnerungen im Gehirn durch einen Prozeß dauerhaft gespeichert werden, der Änderungen in der neuronalen Genexpression vornimmt.
   Auslöser für diesen Prozeß ist der Eintritt von Kalzium durch NMDA-Rezeptoren in der Plasmamembran in die Neuronen.

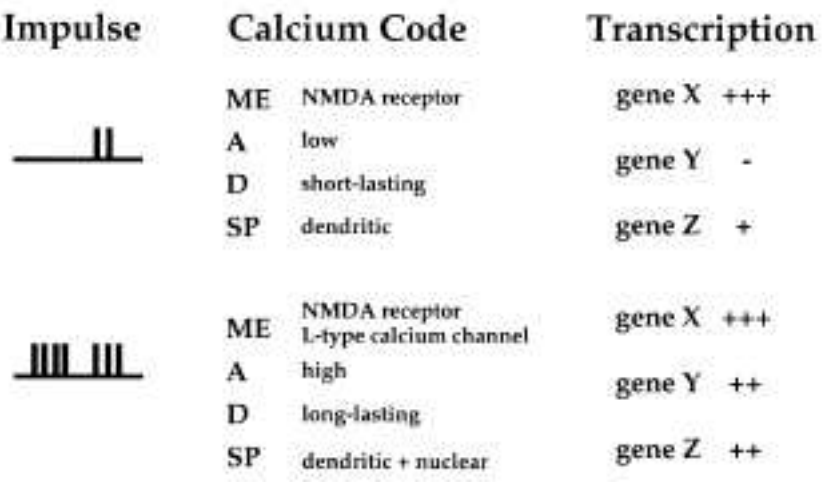

Abbildung 1.7: **Über einen Kalziumcode wird neuronale Aktivität in Transkription umgesetzt.** Hier: Schemata verschiedener Kalziumcodes mit der zugehörigen induzierten Transkription (Induktion von Transkription: +, schwach; ++, mäßig; +++, stark; -, keine Induktion). ME, Art des Eintritts; A, Amplitude; D, Dauer; SP, räumliche Eigenschaften. (Bading[2])

2. **Die Kalziumcodierung kontrolliert abgestufte Genexpression.**

   Durch die Veränderung der Kalziumkonzentration im Zellkern wird die Transkription gesteuert. Die *Amplitude* und *Dauer* des Kalziumflusses sind Teil des Codes der den Umfang der Transkriptionsrate und die Art der Transkription bestimmt.

   Zwei Prozeße übersetzen elektrischen Reiz in Transkription:

   (a) Kalzium aktiviert andere signaltragende Moleküle im Zytoplasma, die das kalziuminduzierte Signal zum Kern tragen und Transkription auslösen. In manchen Neuronen findet jedoch auch eine Translokation eines Kalzium/Calmodulin-Komplexes von den Dendriten zum Kern statt (Deisseroth[12]).

   (b) Kalzium selbst ist der Signalträger und aktiviert im Kern kalziumbindende Proteine, welche anschließend die Transkription aktivieren. Zusätzlich zum einströmenden Kalzium durch NMDA-Rezeptoren ist vermutlich ein interner Kalzium-Speicher beteiligt.

3. **Kalziumaktivierte Transkription ist möglicherweise notwendig für dauerhafte Veränderungen der synaptischen Aktivitäten.**

   Bei der Untersuchung von Stimuli welche sogenannte *long-term potentiation* (LTP) induzieren und Genexpression einleiten fällt auf, dass die Transkriptionsaktivierung nicht notwendigerweise langanhaltende LTP produzieren und keine Garantie für anhaltende neuronale Plastizität sind.
   Andere Versuche sind Indikatoren, dass gerade kalziumabhängige Transkription zur Induktion von langanhaltender LTP führt (Abb. 1.8).

4. **Kalzium ist Regulator der transkriptionsabhängigen Arten der neuronalen Plastizität.**

   Es besteht die Annahme, dass Kalzium den Lernprozess und das Erinnerungsvermögen reguliert, beides plastizitätsabhängige Vorgänge. Kernkalzium könnte auch für pathologische Zustände im Gehirn sorgen. Nicht-physiologische Reize sorgen möglicherweise für fehlerhafte Kalziumsignale, die falsches Transkriptionsverhalten verursachen. Diese Kalziumsignale könnten dann beispielsweise für Epilepsie verantwortlich sein.

Durch die oben aufgeführten Punkte dürfte bestärkt worden sein, wie wichtig das detaillierte Verständnis von Kalziumdiffusion ist.
Die mathematische Simulation kann einen wesentlichen Schritt dazu beitragen, solch ein Verständnis zu erlangen.

1.5.3 Morphologie des Zellkerns

Kalzium spielt – wie Abschnitt 1.5.2 zeigt – eine zentrale Rolle in verschiedenen Bereichen. Es stellt sich die Frage, wie das Kalzium auf so unterschiedliche Weisen aktiv sein kann. Einen zentralen Beitrag könnte die Morphologie des Zellkerns liefern.
Eine kugelförmige Geometrie des Zellkerns würde indizieren, dass Kalzium im gesamten Kern auf die selbe Weise und gleichstark wirkt. Die Vielfalt der regulatorischen Aufgaben des Kalziums wäre mit dieser Geometrie nicht zu erklären.
Echevarria[7] untersucht deshalb, ob die Morphologie des Zellkerns feinere Strukturen besitzt welche das Kalzium nicht global, sondern lokal in diskreten Regionen des Zellkern agieren lässt.
Die in diesem Projekt unternommenen mikroskopischen Untersuchungen (Abb. 2.2) der Zellmorphologie zeigen Einstülpungen *beider* Membranen

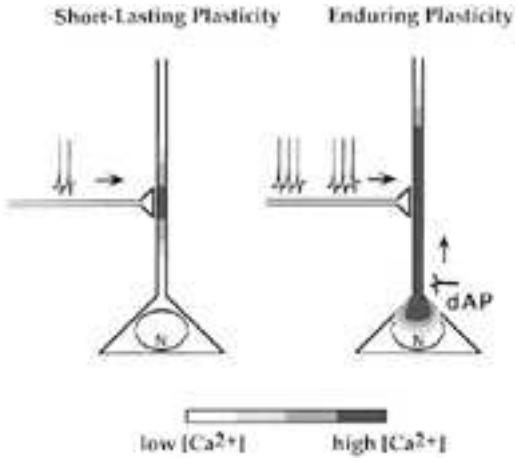

Abbildung 1.8: Kürzere Kalziumaktivität bewirkt kurzanhaltende Plastizität, während lange Kalziumaktivität dauerhafte Plastizität bewirkt. (Bading[2])

der Kernhülle.
Es besteht die Vermutung, dass diese Einstülpungen durch Verkürzung der Diffusionsstrecken im Kern die Kinetik verändern, d.h. sie ermöglichen schnelleres "Füllen" des Kerns, schnellere Signalabnahme und somit schnellere Wiederherstellung der Ausgangskonzentration.
Die Einstülpungen könnten für eine größere Membranoberfläche oder eine Verkleinerung des Zellkernvolumens sorgen. Dies wird in Kapitel 8 näher untersucht.
In beiden Fällen wird die Veränderung direkten Einfluss auf den Diffusionsprozeß von Kalzium haben.
In Kapitel 2 wird man erkennen, dass aus den zweidimensionalen Schnitten (Abbildungen 2.2 und 2.5) nicht ersichtlich ist, ob es sich um Einstülpungen oder tunnelförmige Gebilde handelt. Um die Morphologie der Invaginationen richtig zu interpretieren und die dadurch resultierende Oberflächen- oder Volumenveränderung zu ermitteln bedarf es einer dreidimensionalen Rekonstruktion der Zellkerne.

Kapitel 2

Mikroskopie des Zellkerns

Die Mikroskopie bildet den ersten Schritt in der Geometrie-Rekonstruktion eines Zellkerns. Zunächst muss festgestellt werden welche Form der Mikroskopie für unsere Zwecke brauchbar ist. Weiterhin müssen in Hinblick auf die Filterung und Segmentierung der Mikroskopiedaten passende Einstellungen des Mikroskops festgelegt werden.
Mit Hilfe der Elektronenmikroskopie wurden erstmals die in der Einleitung erwähnten Invaginationen sichtbar. Es wird sich jedoch zeigen, dass für eine 3D-Rekostruktion Aufnahmen eines konfokalen Fluoreszenzmikroskops notwendig sind.
In diesem Kapitel stellen wir deshalb das Prinzip der *Elektronenmikroskopie* und der *konfokalen Mikroskopie* vor.

2.1 Elektronenmikroskopie

2.1.1 Schematischer Aufbau der Elektronenmikroskopie

In der Elektronenmikroskopie existieren verschiedene Verfahren, wie z.B. die *Rasterelektronenmikroskopie* oder die *Transmissionselektronenmikroskopie*.
Die Elektronenmikroskopie ähnelt vom Aufbau her der Lichtmikroskopie, wobei an Stelle der Lichtquelle ein energietragender Elektronenstrahl zur Anregung der zu mikroskopierenden Probe dient.
Als Elektronenquelle dient eine metallische Kathode, beispielsweise aus Wolfram. Diese Kathode wird erhitzt, dabei werden Elektronen aus der Oberfläche des Drahts freigesetzt und bewegen sich in einem elektrischen Feld in Richtung einer Anode. Um Energieverlust, Streuung und Wechselwirkung zwischen Elektronen und Molekülen zu vermeiden, muss zwischen Kathode und Anode ein Vakuum erzeugt werden.
Mit Hilfe einer Magnetspule werden die emmitierten Elektronen zu einem Strahl gerichtet. Hinter der Magnetspule befinden sich zwei Kondensorlinsen, die den Elektronenstrahl auf das Objekt leiten. Hinter dem Objekt befinden sich drei in Reihe geschaltete Linsen, die *Objektivlinse*, *Zwischenlinse*

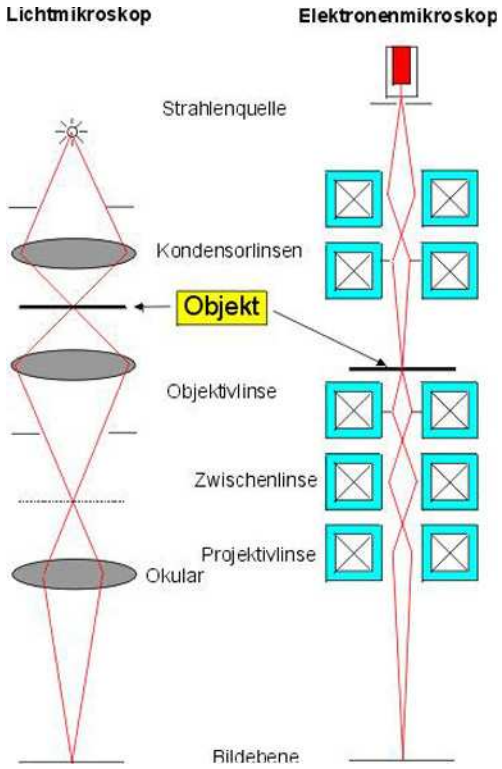

Abbildung 2.1: Schematischer Aufbau des Transmissionselektronenmikroskops (http://www.zoologie-skript.de/methoden/mikros/tem.htm)

und die *Projektivlinse*. Die Objektivlinse filtert mit einer integrierten Blende streuende Elekronen. Die Zwischen- und Projektivlinse dienen zur Endvergrößerung des Bildes welches auf einen Floureszenzschirm projeziert wird.
In Abbildung 2.1 ist der schematische Aufbau der Elektronenmikroskopie gezeigt, im Vergleich zur klassischen Lichtmikroskopie.

2.1.2 Eigenschaften der Elektronenmikroskopie

Die Transmissionselektronenmikroskopie erlaubt eine Visualisierung von Strukturen im Nanometerbereich (bis zu 0,2 Nanometern), d.h. eine Visualisierung auf atomarer Ebene ist möglich.
Durch Zwischenlinse und Projektivlinse kann das erzeugte Bild 100-fach bis 800 000-fach vergrößert werden.
Wie schon erwähnt, konnte mit Hilfe der Elektronenmikroskopie die Doppelmembran des Zellkerns aufgenommen werden. Hier wurden Einstülpungen beider Membranen festgestellt, wie sie in Abbildung 2.2 zu sehen sind.
Um mehrere Ebenen des Gewebes zu mikroskopieren müssen Gewebeschnitte angefertigt und einzeln mikroskopiert werden. Es ist somit sehr

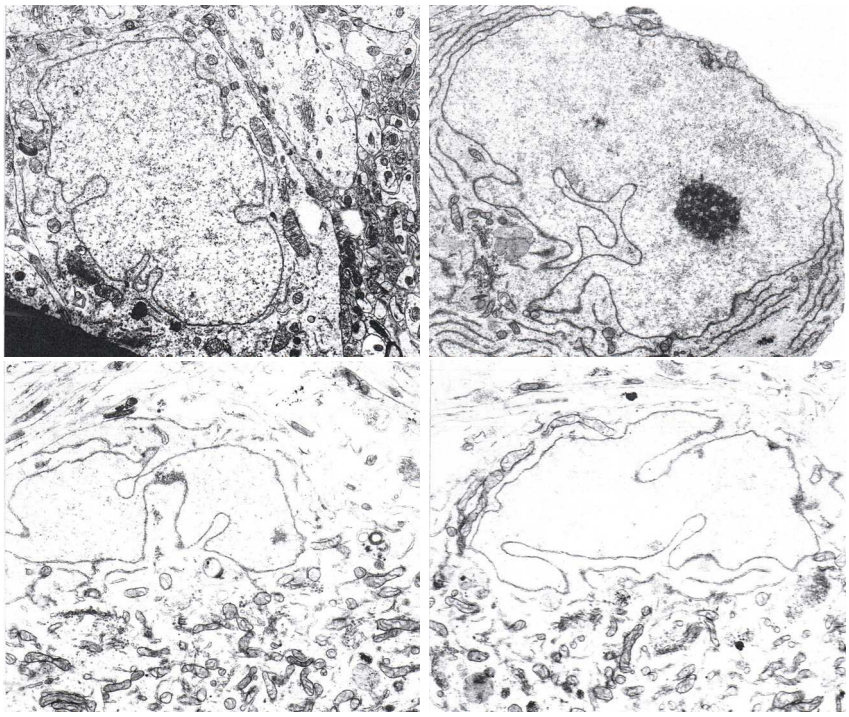

Abbildung 2.2: Elektronenmikroskopische Aufnahmen von Zellkernen (A. Hellwig (IZN), 2003)

schwierig und mit einem sehr großen Zeitaufwand verbunden oder gar unmöglich die Schichtweise angefertigten Aufnahmen den einzelnen Kernen zuzuordnen und daraus eine Bildsequenz einzelner Kerne anzufertigen.
Eine 3D-Rekonstruktion der Zellkerne ist aus den elektronenmikroskopischen Daten also nicht zu gewinnen. Wir stellen im folgenden Abschnitt die konfokale Lichtmikroskopie vor, die uns die Möglichkeit bietet, Bildsequenzen einzelner Kerne anzufertigen.

2.2 Konfokale Fluoreszenzmikroskopie

2.2.1 Schematischer Aufbau der konfokalen Fluoreszenzmikroskopie

Wie der Name schon sagt, wird ein Mikroskopiebild mittels Fluoreszenz erzeugt. Der erste Schritt bei diesem Mikroskopieverfahren ist also die zu visualisierende Membran mit einer Fluoreszenz zu versehen.
In den hier verwendeten Mikroskopiedaten wird der Fluoreszenzfarbstoff *alexa 488* verwendet. Es emmitiert Licht im Wellenbereich von ca. 500 - 600 nm und besitzt sein Emmisionsmaximum bei 520 nm.
Als Energiequelle zur Anregung des Fluoreszenz dient ein Laserstrahl, der

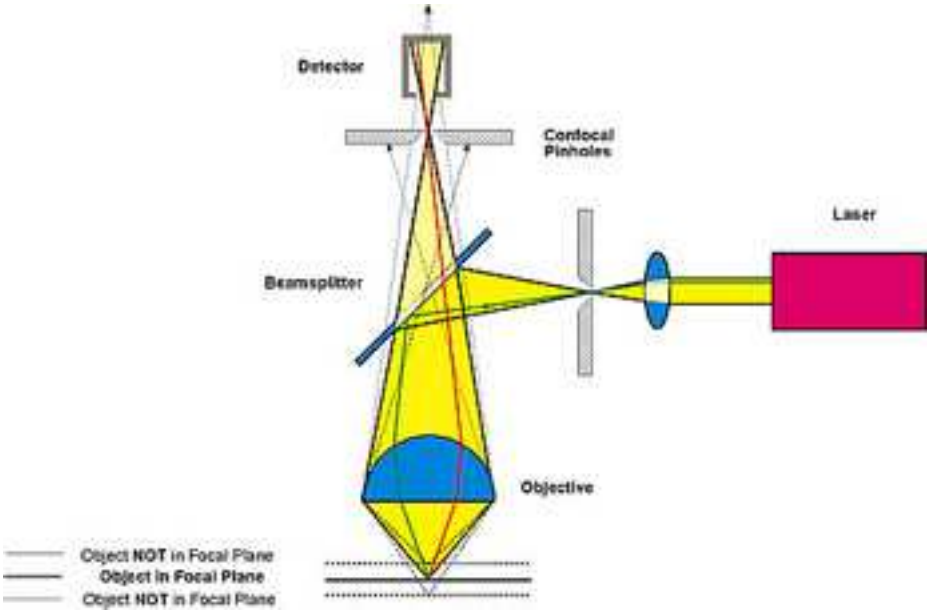

Abbildung 2.3: Schematischer Aufbau der konfokalen Fluoreszenzmikroskopie (http://www.confocal-microscopy.com/website/sc_llt.nsf)

zunächst eine *Lochblende* passiert und auf einen *dichroischen Spiegel* trifft. Reflektiert von diesem Spiegel, wird der Laserstrahl durch eine Linse auf das Objekt gelenkt. Hier werden die fluoreszierenden Molekü le angeregt und emmitieren Licht im oben genannten Wellenbereich.
Das emmitierte Licht wird von der Linse fokussiert und passiert eine zweite Lochblende und trifft schließlich auf einen *Detektor*, dem *Photomultiplier* (PMT).
Durch die erste Lochblende wird der Laser auf genau einen Punkt einer Ebene der Probe gerichtet. D.h. im bis jetzt beschriebenen Vorgehen wird lediglich ein Punkt der Probe aufgenommen (Abb. 2.3).
Zwei Scanspiegel die von Stellmotoren gesteuert werden dienen dazu, den Laser in x- und y-Richtung zu variieren. Ist eine Ebene in x- und y-Richtung *gescannt*, wird die Fokusebene in z-Richtung verstellt und gescannt. Die Position der Bildpunkte wird in einem angschlossenen Computer gespeichert (Abb. 2.4).

2.2.2 Eigenschaften der konfokalen Mikroskopie

Bei der konfokalen Mikroskopie gibt es etliche wählbare Parameter die Einfluss auf das Mikroskopiebild haben. Die mathematischen Verfahren, wie *Filterung* und *Segmentierung* stellen gewisse Anforderungen an das Bild.
In diesem Abschnitt werden Eigenschaften der konfokalen Mikroskopie benannt. Es wird erläutert wie die Parameter zu wählen sind, bzw. auf

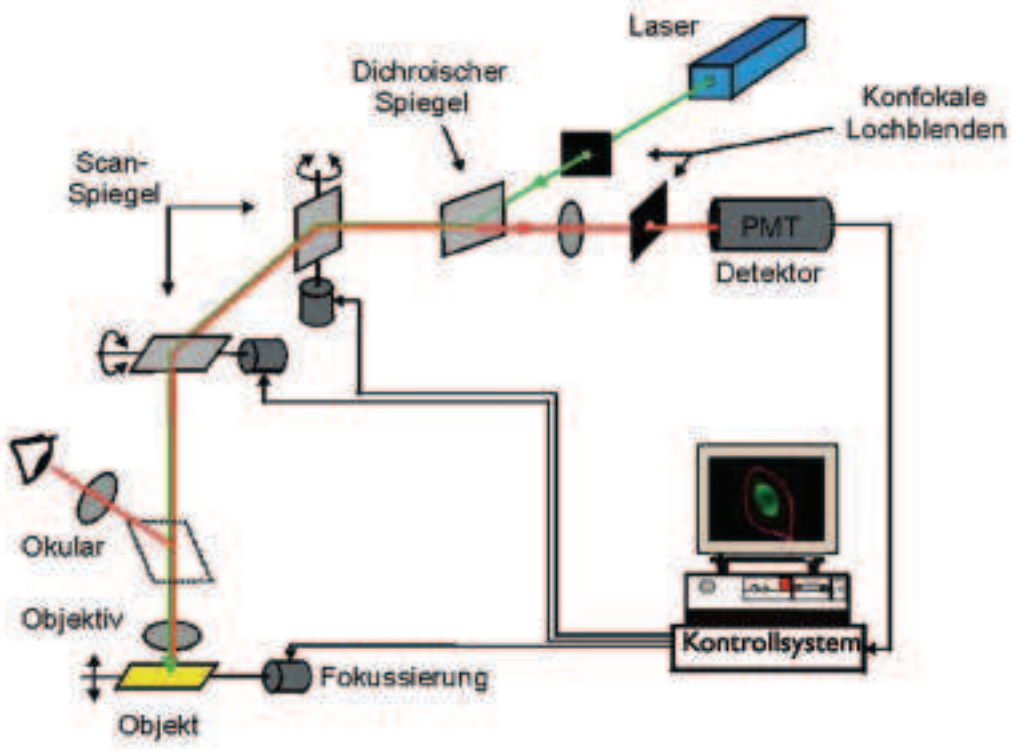

Abbildung 2.4: Schematischer Aufbau eines konfokales Laser-Scanning Mikroskops (http://www.zoologie-skript.de/methoden/fluo/scannl.htm)

was bei der Einstellung des Mikroskops in Hinblick auf die mathematischen Verfahren zu achten ist.

Scanner:

Beim Scanner lassen sich *Scangeschwindigkeit*, *Pixelzeit*, *Pixelgröße* und das *Bildformat* in x/y-Ebene festlegen.
Die Scangeschwindigkeit legt die Pixelzeit fest. Die Pixelzeit ist das Zeitintervall in dem ein Bildpunkt bestrahlt wird und somit die Anzahl *Photonen* pro Bildpunkt steuert.
Mit der Pixelzeit lässt sich also das Rauschen im Bild verändern. Dies ist für unsere Zwecke wichtig, da eine gewisse Grauwertdifferenz zwischen Struktur und Hintergrund wichtig ist.
Als Bildformat wird bis Kapitel 6 eine Auflösung von 512×512 verwendet. Ab Kapitel 7 wird die Auflösung aus mathematischen und rechenzeitlichen Gründen auf 128×128 gesetzt.
Die geringere Auflösung stellt kein Problem dar, da die wichtigen Strukturen der Membran im Verhältnis zur Auflösung so groß sind, dass sie nicht verloren gehen.

z-Schrittweite:

Ein Motor im Mikroskop steuert die Schrittweite in z-Richtung. Die optimale Schrittweite wird von Fall zu Fall vom Mikroskop ermittelt. Die Größe der z-Schrittweite ist für die spätere Rekonstruktion wichtig um die Dimensionsverhältnisse des Zellkerns zu berücksichtigen.

Mikroskop-Objektiv:

Die Wahl des Objektivs beeinflusst die Bildqualität und die Geschwindigkeit in welcher der verwendete Farbstoff ausbleicht. Es ist bei der Wahl des Objektivs darauf zu achten, das möglichst viel Signal in den unteren Schichten des Kerns vorhanden ist um eine 3D-Rekonstruktion möglich zu machen.

In Abbildung 2.5 sind konfokale Fluoreszenzmikroskopiebilder mit verschiedenen Parametereinstellungen dargestellt.

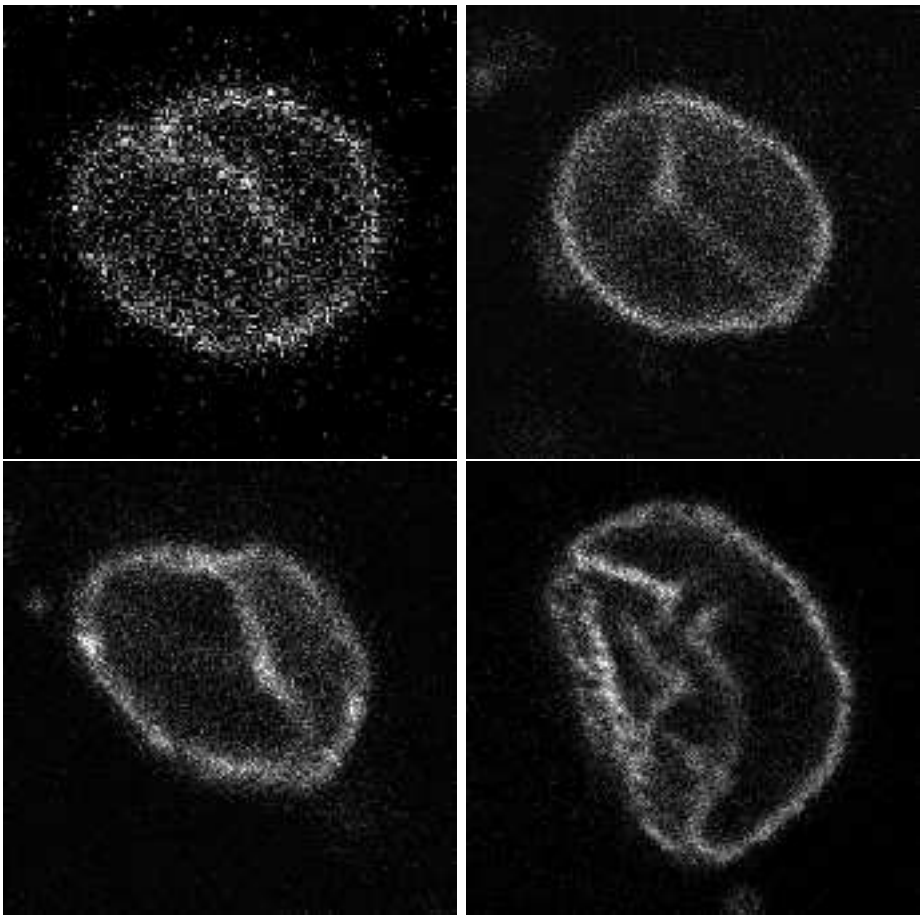

Abbildung 2.5: Beispiele von konfokalen Fluoreszenzaufnahmen mit verschiedenen Parametern (M. Wittmann (IZN), 2005)

Kapitel 3

Filterung mit Hilfe nichtlinearer anisotroper Diffusion

3.1 Einleitung

In Kapitel 2 wurde die Mikroskopie von Zellkernen erläutert. Beim Anfärben der Membran verteilt sich der Farbstoff nicht gleichmäßig. Außerdem kann es passieren, dass freier, ungebundener Farbstoff mikroskopiert wird, der nicht Teil der Membran ist. Dadurch entstehen Lücken in der Oberflächenstruktur der Membran und es entsteht Hintergrundstruktur, die nicht zum Kern gehört.

Wichtige Aufgaben des Filters sind daher

1. Schließung von Lücken in der Struktur
2. Glättung der Struktur
3. Glättung des Hintergrundrauschens

Die Filterung ist eine der stärksten Mittel in der digitalen Bildverarbeitung. Für uns ist der Schritt des Filterns wichtig, da es ohne den Filterprozess unmöglich ist, die dreidimensionale Rekonstruktion eines Zellkerns durchzuführen.
Ein ganz neues Feld in der Bildverarbeitung öffneten die *Partiellen Differentialgleichungen* (PDE) (engl.: *partial differential equations*).
Methoden der Bildverarbeitung, denen partielle Differentialgleichungen zu Grunde liegen weisen sehr stabile Algorithmen auf. Diese Methoden eignen sich daher sehr gut zur Implementierung.

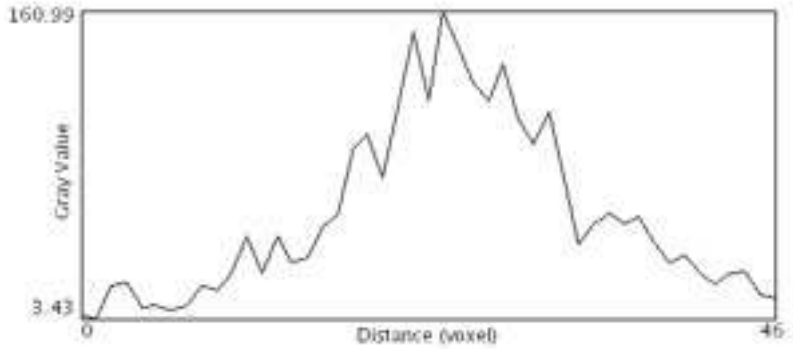

Abbildung 3.1: Profil einer ungefilterten Membran (Queisser (SiT), 2005)

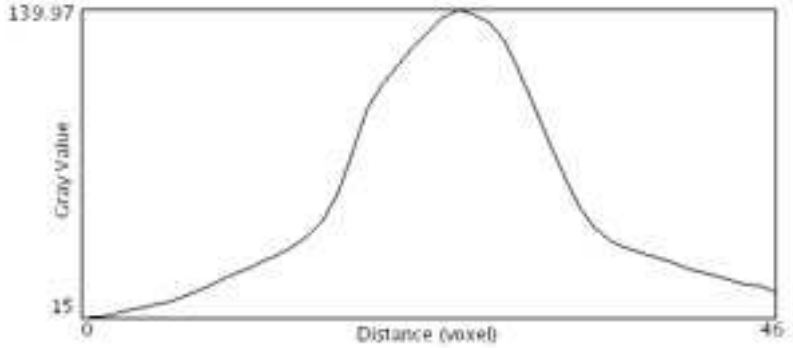

Abbildung 3.2: Profil einer gefilterten Membran (Queisser (SiT), 2005)

Man unterscheidet dabei zwischen

- *linearen* Methoden
- *nichtlinearen* Methoden
- *isotropen* Methoden
- *anisotropen* Methoden

Aufgabe dieses Kapitels ist, diese Methoden zu untersuchen und die für unsere Aufgabenstellung ideale Lösung zu entwickeln.

Der Begriff *Filter* wird definiert, es werden verschiedene Filter vorgestellt und deren mathematische Vorgehensweise erklärt. In Kapitel 3 wird der von Roland Schulte implementierte *nichtlineare anisotrope Diffusionsfilter* mathematisch hergeleitet und dessen Aufbau in Kapitel 4 vorgestellt.
Wir werden uns deshalb in Kapitel 3 und 4 zum Teil an Schulte[17] orientieren.

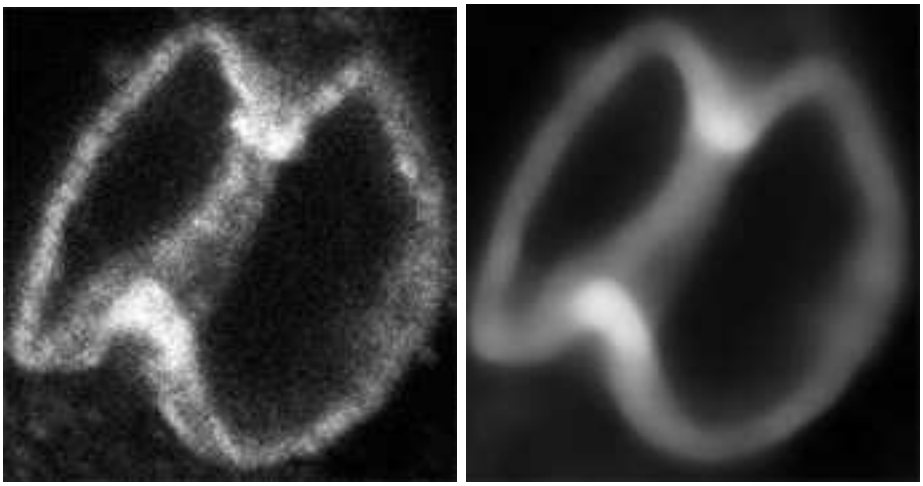

Abbildung 3.3: *links*: Ungefilterte Membran, *rechts*: Mediangefilterte Membran (Queisser (SiT), 2005)

Definition:

Ein Filter ist eine Abbildung F die auf dem Bilddatensatz $\tilde{u}$ operiert:

$$\begin{array}{rccc} F: & \mathbb{R}^n & \longrightarrow & \mathbb{R}^n \quad n \in \mathbb{N} \\ & \tilde{u} & \longmapsto & F(\tilde{u}) \end{array} \tag{3.1}$$

$F(\tilde{u}) =: u$ ist der gefilterte Datensatz.

Abbildung 3.1 zeigt das Profil einer ungefilterten Membran. Die Abbildung F glättet die lokalen Maxima/Minima und erzeugt ein geglättetes Profil, wie es in Abbildung 3.2 zu sehen ist.

3.2 Gängige isotrope Filter

3.2.1 Medianfilter

Einer der einfachsten Filter ist der *Medianfilter*. Dieser Filter geht folgendermaßen vor:
Um das zu filternde Bildvoxel x wird ein Gebiet $\Omega(x)$ gelegt. Dann wird aus den Voxelwerten in Ω das arithmetische Mittel gebildet:

$$u(x) = \sum_{y \in \Omega(x)} g(.) \cdot \tilde{u}(y)$$

$g(.)$ ist eine Gewichtsfunktion. Im Fall des Medianfilters ist g die konstante Funktion $g := \frac{1}{|\Omega(x)|}$.
Einer der Nachteile des Medianfilters ist daher, dass weiter entfernt liegende

Bildpunkte gleich stark gewichtet werden wie nahe gelegene. Abbildung 3.3 zeigt das Resultat eines Medianfilters. Der *Gaußfilter* (*Gaussian blur*) behebt dieses Problem.

3.2.2 Gaußfilter

Der Gaußfilter ersetzt die konstante Gewichtsfunktion durch eine von den Bildpunkten abhängige Funktion g:

$$u(x) = \sum_{y \in \Omega(x)} g(y - x) \cdot \tilde{u}(y)$$

$g := G_\sigma$ ist die Gausssche Normalverteilung, definiert als

$$G_\sigma(x) := \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.2}$$

Der Gaußfilter zieht somit die Entfernung $|x - y|$ der Punkte x und y in Betracht.
Dadurch ist der in Abschnitt 3.2.1 beschriebene Nachteil des Medianfilters behoben, dass der Gaußfilter jedoch keine Ideallösung für unsere Zellkernaufnahmen ist (siehe auch Abbildung 3.4), wird sich im folgenden Abschnitt zeigen.

3.3 Die Mathematik der nichtlinearen anisotropen Diffusion

3.3.1 Die Diffusionsgleichung

Ein physikalischer Diffusionsprozess wird mathematisch durch eine parabolische partielle Differentialgleichung, der Diffusionsgleichung

$$\frac{\partial u}{\partial t} = \nabla u \qquad \text{auf} \quad \mathbb{R}^+ \times \mathbb{R}^n \tag{3.3}$$

beschrieben.
Dabei ist u eine von Ort und Zeit abhängige Konzentrationsfunktion. Sie beschreibt die örtliche und zeitliche Veränderung der Konzentration in einem Medium.
Diesen Diffusionsprozess wollen wir uns für die Filterung von Bilddaten zu Nutze machen. Das zu filternde Bild besteht aus Voxeln mit Werten im Intervall $[0, 255]$, im skalierten Fall im Intervall $[0.1]$ (wir werden im Folgenden vom skalierten Fall ausgehen). Diese Werte repräsentieren die zugehörige Graustufe des Voxels. Fasst man die Grauwerte als Konzentrationsdichte auf, lässt sich die obige Diffusionsgleichung auf unseren Bilddatensatz anwenden. Dieses Vorgehen lässt sich durch zwei physikalische Gesetzmäßigkeiten begründen.

1. Ein Gemisch mit einem Konzentrationsgefälle gleicht dieses aus.
   Das Gefälle wird durch den Konzentrationsgradienten ∇ beschrieben.
   Diesem Gradienten wirkt ein Fluss ϕ entgegen.
   Daraus ergibt sich das sogenannte *Fick'sche Gesetz*:

$$-\nabla u = \phi$$

2. *Massenerhaltung*: Die Masse in einem Volumen V bleibt erhalten, d.h. existiert ein Fluss über den Rand ∂V so fließt der gleiche Anteil Masse über den Rand nach außen, wie nach innen.
   Als mathematische Gleichung formuliert, erhält man

$$-\int_{\partial V} \phi \cdot \vec{n}\, \mathrm{d}s = \int_V \frac{\partial u}{\partial t}\, \mathrm{d}x$$

   oder

$$\frac{\partial u}{\partial t} = -\nabla \cdot \phi$$

In Worten: Der Fluss über den Rand ∂V entlang der Normalen $\vec{n}$ ist gleich der zeitlichen Konzentrationsänderung im Volumen V.

3.3.2 Lineare Diffusion

Wir wollen die oben aufgestellte Diffusionsgleichung weiter analysieren.
Betrachten wir dazu erneut das *Fick'sche Gesetz*:

$$-\nabla u = \phi$$

Man erkennt, dass der Fluss ϕ linear vom Gradienten abhängt. Diese Form von Diffusion heißt deshalb *lineare Diffusion*.
Wir haben bereits einen Filter kennengelernt, der sich der linearen Diffusion bedient, der Gaußfilter. Diesen kann man durch eine *Faltung* der Bilddaten mit der Gaußfunktion G_σ erklären.

$$u(x,\sigma) = (\tilde{u} * G_\sigma)(x) = \int_{\mathbb{R}^n} \tilde{u}(y) \cdot \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} e^{-\frac{(x-y)^2}{2\sigma^2}}\, \mathrm{d}y \qquad (3.4)$$

Dabei regelt die Standardabweichung σ die Gewichtung der Bildpunkte in der Diffusion.
Abbildung 3.4 zeigt eine Gaußgefilterte Membran. Man erkennt eine Glättung des Bildes, welche aber mit dem Verlust an Struktur einher geht. Um später den Kern ausmessen und rekonstruieren zu können, muss die Geometrie des Kerns möglichst unverändert bleiben. Der Gaußfilter ist für diesen Fall also ungeeignet.

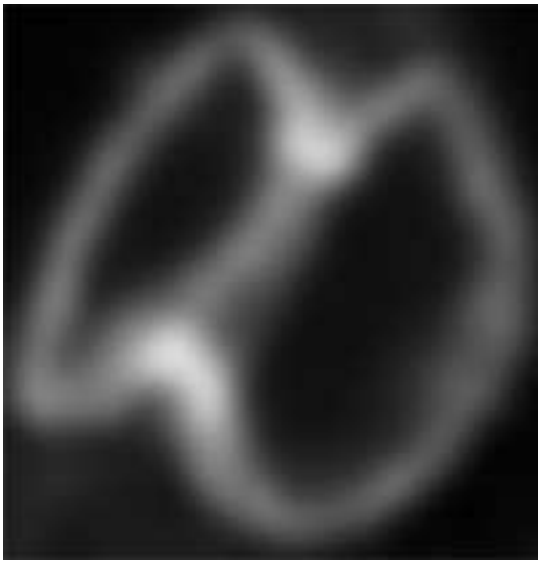

Abbildung 3.4: Gaußgefilterte Membran (Queisser (SiT), 2005)

3.3.3 Nichtlineare Diffusion

Der nächste Schritt muss also sein, einen Filter zu finden, der einerseits die Struktur des Kerns glättet, andererseits aber die Ränder der Membran scharf hält.
Möchte man die Diffusion in Richtung der Ränder dämpfen, darf der Fluss nicht mehr linear vom Gradienten ∇ abhängen. Wir müssen die Gleichung $-\nabla u = \phi$ mit einer Dämpfungsfunktion g versehen. g hängt von ∇u ab und besitzt die folgenden Eigenschaften:

1. $g(0) = 1$
2. g ist monoton fallend

Das Fick'sche Gesetz hat jetzt die Form:

$$g(|\nabla u(x,t)|) \cdot \nabla u(x,t) = \phi(x,t)$$

Die so konstruierte Diffusion ist nichtlinear und heißt nach ihren Erfindern *Perona-Malik-Diffusion*:

$$\frac{\partial u}{\partial t} = \nabla \cdot (g(|\nabla u|)\nabla u)$$

Perona und Malik waren die ersten, die 1987 parabolische Differentialgleichung im Rahmen nichtlinearer Diffusionsfilter anwandten (Weickert[19]).
In Weickert[19] ist g von $u_\sigma(x,t) := (u * G_\sigma)(x,t)$ – dort als *edge detector*

bezeichnet – abhängig. Da wir uns in Abschnitt 3.3.4 auf Weickert[19] beziehen, schreiben wir die Perona-Malik-Diffusion als

$$\frac{\partial u}{\partial t} = \nabla \cdot (g(|\nabla u_\sigma|^2)\nabla u) \tag{3.5}$$

Als Dämpfungsfunktion g stellt Schulte[17] diese drei Varianten vor:

1. **Perona-Malik-Diffusivität**

$$g(s) = \frac{c}{1 + (s/\lambda)^{\alpha+1}} \tag{3.6}$$

   Dabei sind c und $\alpha > 0$ frei wählbare Konstanten.

2. **Weickert-Diffusivität**

$$g(s) = 1 - \exp\left(\frac{c}{(s/\lambda)^m}\right) \tag{3.7}$$

   mit den Konstanten c und m.

3. **Black-Sapiro Diffusivität**

$$g(s) = \begin{cases} \frac{1}{2}\left(1 - \left(\frac{s}{\lambda \cdot \sqrt{5}}\right)^2\right)^2 & |s| \leq \lambda \cdot \sqrt{5} \\ 0 & \text{sonst} \end{cases} \tag{3.8}$$

Abbildung 3.5 zeigt die Graphen zu den unterschiedlichen Dämpfungsfunktionen.
Jaakko Astola[1] stellt in seinem Werk *Fundamentals of Nonlinear Digital Filtering* eine ganze Reihe von nichtlinearen Filtern vor. In der Literatur werden die Begriffe *nichtlineare Diffusion* und *anisotrope Diffusion* unterschiedlich verwendet.
Deshalb fügen wir an dieser Stelle eine Begriffserklärung ein:

1. Bei der *nichtlinearen* Diffusion wird die Diffusionsstärke von einer wie oben definierten Funktion g gesteuert.

2. Bei der *anisotropen* Diffusion wird die skalare Steuerung durch einen *Diffusionstensor* ersetzt.

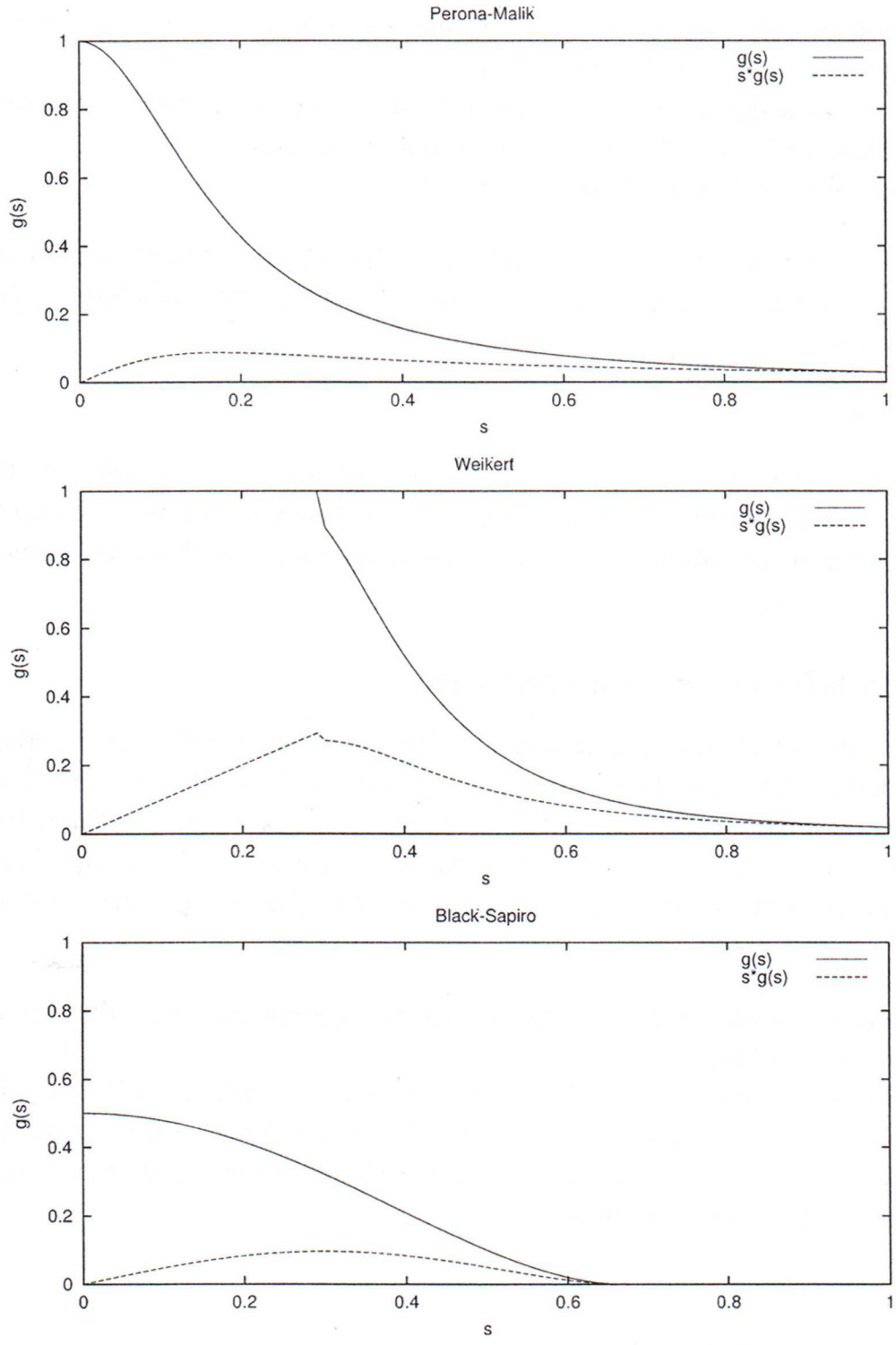

Abbildung 3.5: Graphen verschiedener Gewichtsfunktionen (Schulte[17])

3.3.4 Nichtlineare anisotrope Diffusion

Die anisotrope Diffusion bietet mehrere Vorteile gegenüber der isotropen Filterung. Zwei davon spielen in unserem Fall eine große Rolle:

1. Die Diffusion findet *entlang* der ermittelten Struktur statt und wird orthogonal dazu gedämpft.

2. Lücken in der Struktur werden geschlossen.

Bei der isotropen Diffusion hat man eine *Fluss-Gradient*-Beziehung von der Form

$$\phi = -g\nabla u$$

Der Fluss ϕ ist also immer parallel zum Gradienten ∇u. In diesem Abschnitt führen wir eine Methode ein, welche die Diffusion in Richtung der zu filternden Struktur lenkt. Dazu wird der *Diffusionstensor* eingeführt.

Definition:

Der *Diffusionstensor* ist eine *lineare Abbildung*

$$\begin{array}{rccccc} D: & \mathbb{R}^n & \longrightarrow & \mathbb{R}^n & & n \in \mathbb{N} \\ & u & \longmapsto & D(x) & & \end{array} \tag{3.9}$$

mit den Eigenschaften:

- D besitzt n normierte, orthogonale Eigenvektoren $v_1, \ldots, v_n$ und die dazugehörigen Eigenwerten $\lambda_1, \ldots, \lambda_n$.
- $v_1 \parallel \nabla u_\sigma, \quad v_2 \perp \nabla u_\sigma$

Als Flussgleichung erhalten wir

$$\phi = D(u) \cdot \nabla u$$

Die nichtlineare anisotrope Diffusion wird somit beschrieben durch

$$\frac{\partial u}{\partial t} = \nabla \cdot (D(u) \cdot \nabla u) \tag{3.10}$$

Beschränken wir uns zunächst auf den zweidimensionalen Fall.
Mit Hilfe des Diffusionstensors D kann der Fluss ϕ nun folgendermaßen geschrieben werden:

$$\begin{array}{rcl} \phi & = & D \cdot \nabla(u) \\ & = & D \cdot (\alpha_1 v_1 + \alpha_2 v_2) \quad = \quad D\alpha_1 v_1 + D\alpha_2 v_2 \\ & = & \lambda_1 \alpha_1 v_1 + \lambda_2 \alpha_2 v_2 \end{array}$$

mit $\nabla u = \alpha_1 v_1 + \alpha_2 v_2 \quad \alpha \in \mathbb{R}$.

λ_1 und λ_2 dienen also als Regler für die Flussstärke in Richtung der Eigenvektoren v_1 und v_2. Um Glättung entlang der Strukturkanten zu fördern und gleichzeitig orthogonal dazu zu dämpfen, schlägt Weickert[19] folgende Eigenwerte vor:

$$\begin{aligned} \lambda_1(\nabla u_\sigma) &:= g(|\nabla u_\sigma|^2), \\ \lambda_2(\nabla u_\sigma) &:= 1. \end{aligned}$$

Für den zweidimensionalen Fall ist diese Methode ausreichend. Ist dieses Verfahren auf dreidimensionale Datensätze anwendbar?
Bei unseren Zellkernaufnahmen müssen wir auf verschiedene Geometrien Rücksicht nehmen:

- Innerhalb des Zellkerns ist keine Membranstruktur vorhanden. Die Diffusion soll also in alle drei Richtungen stattfinden um möglichst hohe Homogenität zu erreichen.
- Die Membran kann lokal als der $\mathbb{R}^2$ aufgefasst werden. Die Diffusion soll nur in *zwei* Richtungen stattfinden.
- Die Membran besitzt Einstülpungen. Die Enden mancher Einstülpungen besitzen eine lokal *lineare* Struktur. Die Diffusion soll also nur in einer Richtung erfolgen.

Mit der oben beschriebenen Diffusionsteuerung über die Eigenvektoren des Diffusionstensors kann nur eine Richtung definiert werden, in der die Diffusion stattfinden soll und eine in der sie gedämpft wird.
Da wir drei verschiedene Geometrietypen zu berücksichtigen haben, bei denen die Diffusion in mehrere Richtungen gelenkt bzw. gedämpft werden muss, brauchen wir ein anderes Verfahren um die Diffusion in die gewünschten Richtungen zu lenken.

3.4 Kernstrukturerkennung

3.4.1 Physikalische Trägheitsmomente

Die Mechanik beschäftigt sich unter anderem mit der Bewegung von Körpern. Ein Spezialfall ist die Rotation eines Körpers um eine feste Achse. Die unter diesem Aspekt hergeleitete *kinetische Energie* führt uns zu dem Begriff des *Trägheitstensors*. Dieser Trägheitstensor lässt sich zur Identifikation der Geometrie eines Körpers in den Kategorien *linear*, *planar* und *isotrop* nutzen.
Bei der Herleitung werden wir uns zunächst mit dem diskreten Fall beschäftigen. Wir betrachten also ein aus Massepunkten bestehendes System

und gehen anschließend auf den kontinuierlichen Fall über. An diesem Fall wird demonstriert wie sich der Trägheitstensor zur Strukturerkennung nutzen lässt.
Um den Trägheitstensor herzuleiten benötigen wir einige Größen aus der Physik.

- Die Masse eines Körpers wird definiert durch

$$M = \sum_i m_i \tag{3.11}$$

- Der Schwerpunkt eines Körpers wird bestimmt durch

$$\mathbf{R} = \frac{1}{M} \sum_i m_i \mathbf{r}_i \tag{3.12}$$

- Winkelgeschwindigkeit ω:

$$\omega = (\omega_1, \omega_2, \omega_3) \tag{3.13}$$

Weiter wählen wir ein kartesisches Koordinatensystem, dessen Ursprung im Schwerpunkt sp_K des Körpers K liegt.
Im diskreten Fall wird die *kinetische Energie* definiert als:

$$T = \frac{1}{2} \sum_i m_i \cdot \dot{\mathbf{r}}_i^2 \tag{3.14}$$

Dabei ist m_i die Masse des i-ten Massenpunktes und r_i der Ortsvektor des i-ten Teilchens bzgl. der raumfesten Basis $B = \{e_1, e_2, e_3\}$.
Die kinetische Energie lässt sich in einen *Rotations-* und *Translationsanteil* aufspalten (siehe Nolting[15]).

$$T = \frac{1}{2} M \dot{\mathbf{r}}_0^2 + \frac{1}{2} \sum_i m_i (\omega \times \mathbf{r}_i)^2 = T_T + T_R$$

Die Berechnung der Rotationsenergie führt uns schließlich auf den Trägheitstensor. Für T_R erhält man

$$T_R = \frac{1}{2} \sum_{l,m=1}^{3} J_{lm} \omega_l \omega_m$$

Dabei sind die J_{lm} die Matrixeinträge des Trägheitstensor und haben die Gestalt

$$J_{lm} = \sum_i m_i \left(\mathbf{r}_i^2 \delta_{lm} - x_{il} x_{im} \right) \tag{3.15}$$

Der Trägheitstensor als Matrix geschrieben:

$$(J_{lm})_{1\leq l,m\leq 3} = \begin{pmatrix} \sum_i m_i \left(x_{i2}^2 + x_{i3}^2\right) & -\sum_i m_i x_{i1} x_{i2} & -\sum_i m_i x_{i1} x_{i3} \\ -\sum_i m_i x_{i2} x_{i1} & \sum_i m_i \left(x_{i1}^2 + x_{i3}^2\right) & -\sum_i m_i x_{i2} x_{i3} \\ -\sum_i m_i x_{i3} x_{i1} & -\sum_i m_i x_{i3} x_{i2} & \sum_i m_i \left(x_{i1}^2 + x_{i2}^2\right) \end{pmatrix}$$

Dabei wurde verwendet, dass $\mathbf{r}_i = \sum_{j=1}^{3} x_{ij}\mathbf{e}_j$. Um vom diskreten Fall auf den kontinuierlichen Fall überzugehen, führen wir eine Dichtefunktion $\rho(\mathbf{r})$ ein und erhalten die integrale Darstellung des Trägheitstensors in der Form

$$J_{lm} = \int_V \rho(\mathbf{r}) \left(r^2 \delta_{lm} - x_l x_m\right) \mathrm{d}V \tag{3.16}$$

über ein Volumenstück V.
In Schulte[17] wird eine äquivalente Darstellung des Trägheitstensors implementiert. Wir wollen deshalb die dort verwendete Darstellung aus Gleichung (3.16) herleiten.

Behauptung: Die Trägheitsmatrix J_K für einen Körper K lässt sich schreiben als

$$J_K = \int_V \rho(y) \left(|y - sp_K|^2 \,\mathbb{E} - (y - sp_K) \otimes (y - sp_K)\right) \mathrm{d}y$$

Beweis: Vergleichen wir der Reihe nach die einzelnen Komponenten der Gleichungen.

1. Der Vektor y entspricht dem Vektor $\mathbf{r}$. Der Unterschied liegt darin, dass y bezüglich dem kartesischen Koordinatensystem mit dem Ursprung in Null dargestellt ist und $\mathbf{r}$ – wie oben erwähnt – Koordinaten des um sp_K verschobenen Koordinatensystem trägt.

2. Das Kroneckersymbol δ übernimmt in Koordinatenschreibweise die Aufgabe der Einheitsmatrix $\mathbb{E}$.

3. Das Tensorprodukt lässt sich bezüglich der Basis

   $$B = \{e_1 \otimes e_1, e_1 \otimes e_2, e_1 \otimes e_3, e_2 \otimes e_1, \ldots, e_3 \otimes e_3\}$$

   schreiben als

   $$(y - sp_K) \otimes (y - sp_K) = \mathbf{x} \otimes \mathbf{x} = \begin{pmatrix} x_1 x_1 & x_1 x_2 & x_1 x_3 \\ x_2 x_1 & x_2 x_2 & x_2 x_3 \\ x_3 x_1 & x_3 x_2 & x_3 x_3 \end{pmatrix}$$

Damit ist die Gleichheit der einzelnen Terme gezeigt. Wir wollen nun die neue Schreibweise in einer leicht vereinfachten Form übernehmen. Zur Strukturerkennung dienen die *Eigenvektoren* und *Eigenwerte* des Trägheitstensors, genauer die Eigenvektoren und die *Differenzen* der Eigenwerte. Wir haben also das System

$$
\begin{aligned}
|y - sp_K|^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot v_1 - \begin{pmatrix} x_1x_1 & x_1x_2 & x_1x_3 \\ x_2x_1 & x_2x_2 & x_2x_3 \\ x_3x_1 & x_3x_2 & x_3x_3 \end{pmatrix} \cdot v_1 &= \lambda_1 v_1 \\
|y - sp_K|^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot v_2 - \begin{pmatrix} x_1x_1 & x_1x_2 & x_1x_3 \\ x_2x_1 & x_2x_2 & x_2x_3 \\ x_3x_1 & x_3x_2 & x_3x_3 \end{pmatrix} \cdot v_2 &= \lambda_2 v_2 \\
|y - sp_K|^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot v_3 - \begin{pmatrix} x_1x_1 & x_1x_2 & x_1x_3 \\ x_2x_1 & x_2x_2 & x_2x_3 \\ x_3x_1 & x_3x_2 & x_3x_3 \end{pmatrix} \cdot v_3 &= \lambda_3 v_3
\end{aligned}
$$

Vernachlässigt man in den drei Gleichungen den ersten Summanden, hat das neue Gleichungssystem dieselben Eigenvektoren und dieselben Differenzen der Eigenwerte.
Wir können also für unsere Zwecke den Trägheitstensor vereinfachen und schreiben den *vereinfachten Trägheitstensor* T_K als:

$$T_K = \int_K \rho(y) \left((y - sp_K) \otimes (y - sp_K) \right) \, \mathrm{d}y \tag{3.17}$$

3.4.2 Der Trägheitstensor zur Strukturerkennung

Da der Trägheitstensor eine *symmetrische, positiv definite* Matrix ist, besitzt er ein orthonormales System aus Eigenvektoren mit zugehörigen positiven Einheitsvektoren. Für den Trägheitstensor gilt folgender

Satz: Sei T_K der symmetrische Trägheitstensor. Dann existiert eine orthogonale Matrix $S \in O(n, \mathbb{R})$, so dass gilt

$$ST_KS^{-1} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}, \qquad S^TS = SS^T = \mathbb{E}$$

für die Eigenwerte $\lambda_1 \geq \ldots \geq \lambda_n$ aus $\mathbb{R}$.

Zum Beweis siehe Weissauer[20].

Die Matrix S führt eine sogenannte *Hauptachsentransformation* durch. Die Standardbasis wird in die Basis der Eigenvektoren transformiert. Die

Eigenvektoren heißen *Hauptträgheitsachsen*, die Eigenwerte *Hauptträgheitsmomente*.
Diese werden wir berechnen, um die Struktur des Körpers als linear, planar oder isotrop zu identifizieren.

Betrachten wir beispielsweise einen Quader mit den Seitenlängen a, b, c, wobei $a > b \approx c$. Richten wir diesen der Einfachheit halber am kartesischen Koordinatensystem aus, d.h. $S = \mathbb{E}$ und wählen die konstante Dichtefunktion $\rho(y) = 1$.
Aufgrund der orthogonalen Ausrichtung sind alle Nebendiagonaleinträge gleich Null. Die Hauptdiagonaleinträge berechnen sich folgendermaßen:

$$\begin{aligned} T^{11}_{Quader} &= \int_{m=0}^{c}\int_{l=0}^{b}\int_{k=0}^{a} 1 \cdot x_{11}^2 \,\mathrm{d}k\,\mathrm{d}l\,\mathrm{d}m = \frac{1}{12}a^3bc \\ T^{22}_{Quader} &= \int_{m=0}^{c}\int_{l=0}^{b}\int_{k=0}^{a} 1 \cdot x_{22}^2 \,\mathrm{d}k\,\mathrm{d}l\,\mathrm{d}m = \frac{1}{12}ab^3c \\ T^{33}_{Quader} &= \int_{m=0}^{c}\int_{l=0}^{b}\int_{k=0}^{a} 1 \cdot x_{22}^2 \,\mathrm{d}k\,\mathrm{d}l\,\mathrm{d}m = \frac{1}{12}abc^3 \end{aligned}$$

$$\Rightarrow T_{Quader} = \frac{1}{12}\begin{pmatrix} a^3bc & 0 & 0 \\ 0 & ab^3c & 0 \\ 0 & 0 & abc^3 \end{pmatrix}$$

Man sieht also mit $a > b, c$ ist das erste Trägheitmoment viel größer als die anderen beiden und dem Körper kann eine vornehmlich *lineare* Struktur zugeordnet werden.
Im Fall dass $a \approx b > c$ erhält man über die Trägheitmomente die Vorhersage einer *planaren* Struktur, etc.
Diese Vorhersage aus den Trägheitsmomenten wollen wir nun nutzen um die Diffusion der Bilddaten zu steuern.

3.4.3 Steuerung der Diffusion

Sammeln wir nun die Daten die zur Steuerung der Diffusion nötig sind. Die zu filternden Bilddaten sind unterteilt in Voxel mit diskreten Grauwerten, einem Bildvektor $u(y)$ den wir als konstant auf den einzelnen Voxeln annehmen und dem Voxelvolumen v_y. Wir definieren die Begriffe *Körper*, *Körpermasse*, *Massenschwerpunkt* und können damit den Trägheitstensor für Bilddaten berechnen und schließlich die Diffusion wie gewünscht steuern.

- **Körper**: Wir definieren eine Umgebung V um den Bildpunkt x der als physikalischer Körper aufgefasst wird. Der Einfachheit halber wählen wir die Umgebung V so, dass ein Bildvoxel entweder ganz in V enthalten ist, oder außerhalb von V liegt.
- **Körpermasse**:

$$m_V(x) = \sum_{y \in V(x)} v_y \cdot u(y) \tag{3.18}$$

- **Massenschwerpunkt**:

$$sp_V(x) = \frac{1}{m_V(x)} \sum_{y \in V(x)} v_y \cdot u(y) \cdot y \tag{3.19}$$

Damit ergibt sich für den vereinfachten Trägheitstensor:

$$T_V(x) = \sum_{y \in V(x)} v_y \cdot u(y) \cdot ((y - sp_V(x)) \otimes (y - sp_V(x))) \tag{3.20}$$

Es lässt sich mit den drei Eigenvektoren v_1, v_2, v_3 und Eigenwerten $\lambda_1 \geq \lambda_2 \geq \lambda_3$ der Diffusionstensor D (3.9) beschreiben durch

$$D = \begin{pmatrix} v_{1_1} & v_{2_1} & v_{3_1} \\ v_{1_2} & v_{2_2} & v_{3_2} \\ v_{1_3} & v_{2_3} & v_{3_3} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \cdot \begin{pmatrix} v_{1_1} & v_{2_1} & v_{3_1} \\ v_{1_2} & v_{2_2} & v_{3_2} \\ v_{1_3} & v_{2_3} & v_{3_3} \end{pmatrix}^T \tag{3.21}$$

Ersetzt man die Eigenwerte durch Variablen t_1, t_2, t_3 so lässt sich die Diffusion über diese steuern, also

$$D := \begin{pmatrix} v_{1_1} & v_{2_1} & v_{3_1} \\ v_{1_2} & v_{2_2} & v_{3_2} \\ v_{1_3} & v_{2_3} & v_{3_3} \end{pmatrix} \cdot \begin{pmatrix} t_1 & 0 & 0 \\ 0 & t_2 & 0 \\ 0 & 0 & t_3 \end{pmatrix} \cdot \begin{pmatrix} v_{1_1} & v_{2_1} & v_{3_1} \\ v_{1_2} & v_{2_2} & v_{3_2} \\ v_{1_3} & v_{2_3} & v_{3_3} \end{pmatrix}^T \tag{3.22}$$

3.5 Formulierung des numerischen Problems

Wir haben in den Abschnitten 3.3 und 3.4 die mathematischen Grundlagen geschaffen um die nichtlineare anisotrope Diffusion, angewandt auf Mikroskopiedaten, zu beschreiben.
In diesem Abschnitt übertragen wir diese Grundlagen auf ein numerisches Problem das es zu lösen gilt.
Dazu muss die in Abschnitt 3.3.4 definierte Differentialgleichung (3.10) auf einem beschränkten Gebiet Ω definiert werden.

3.5.1 Das Anfangs-Randwert-Problem

Bisher wurden sämtliche Differentialgleichungen auf $\mathbb{R}^+ \times \mathbb{R}^n$ definiert. Wir haben es bei den Mikroskopiedaten mit Bildstapeln der Auflösung r und Anzahl n Bildebenen zu tun, d.h. die Differentialgleichung muss auf einem Gebiet $\Omega \subset \mathbb{R}^3$ bestehend aus $r^2 \cdot n$ Voxeln definiert werden.
Wie ist dann die Differentialgleichung auf dem Rand definiert?
Im Rahmen der Theorie der partiellen Differentialgleichungen stellt Schwarz[18] folgende *Randwert-Bedingungen* zusammen:

1. $u = \varphi$ auf dem Rand $\partial\Omega$ des Gebiets Ω. Diese Bedingung heißt *Dirichlet-Bedingung*. Im Rahmen der Wärmediffusion heißt dies nichts anderes als dass eine bestimmte Temperatur auf dem Rand $\partial\Omega$ gehalten wird.

2. $\frac{\partial u}{\partial n} = \gamma$ auf dem Rand $\partial\Omega$. Hier wird entlang der Normalen n die Derivation auf dem Rand vorgegeben, d.h. die Diffusion über den Rand wird festgelegt. Diese Randbedingung nennt sich *Neumann-Bedingung*. Der Fall $\frac{\partial u}{\partial n} = 0$ heißt *natürliche Randbedingung* oder *sealed-end*-Bedingung und beschreibt den isolierten Zustand des Gebiets Ω.

3. Eine Kombination der Dirichlet- und Neumann-Bedingung beschreibt die *Cauchy-Bedingung* $\frac{\partial u}{\partial n} + \alpha u = \beta$ auf Ω.
   Dabei sind φ, γ, α und β gegebene Funtionen auf dem Rand $\partial\Omega$.

Für unseren Fall wählen wir die natürliche Randbedingung $\frac{\partial u}{\partial n} =: \nabla u \cdot \vec{n} = 0$ damit wir ein System ohne Informationsverlust über den Rand erhalten.
Geben wir nun noch eine *Anfangsbedingung* $u(x, 0) = u_0(x)$ auf dem Abschluss $\bar{\Omega}$ von Ω vor, können wir das zu lösende *Anfangs-Randwert-Problem* stellen:

$$\begin{array}{ll} \frac{\partial u}{\partial t} = \nabla \cdot (D(u) \cdot \nabla u) & \text{auf } \mathbb{R}^+ \times \Omega \\ u(x, 0) = u_0(x) & \text{auf } \bar{\Omega} \\ (D(u) \cdot \nabla u) \cdot \vec{n} = 0 & \text{auf } \mathbb{R}^+ \times \partial\Omega \end{array}$$

3.5.2 Diskretisierung des Anfangs-Randwert-Problem

Zur Berechnung des in Abschnitt 3.5.1 formulierten Anfangs-Randwert-Problems gibt es zunächst zwei verschiedene Herangehensweisen, eine *analytische* und eine *numerische*.
Die Differentialgleichung $\frac{\partial u}{\partial t} = \nabla \cdot (D(u) \cdot \nabla u)$ ist auf einem *kontinuierlichen* Gebiet $\mathbb{R}^+ \times \Omega$ definiert worden. Der analytische Ansatz bedeutet, eine *exakte* Lösung der Differentialgleichung auf dem definierten Gebiet zu finden. In Forster[8] werden etliche solcher analytischen Lösungsverfahren besprochen. So schön diese Verfahren sind, sind sie jedoch nicht geeignet um von einem Computer übernommen zu werden. Auch falls eine solche analytische

Lösung exitieren sollte, ist diese aus Stabilitätsgründen meist nicht in einem Programm verwendbar.
Die auf dem kontinuierlichen Gebiet $\mathbb{R}^+ \times \Omega$ definierte Differentialgleichung muss nun auf ein *diskretes* Gebiet übertragen werden und dort diskret gelöst werden. Dieser Schritt in Richtung numerischer Lösung nennt sich *Diskretisierung*.
Die auf einem diskreten Gebiet gelöste Differentialgleichung ist eine Annäherung an die analytische Lösung, dessen Fehler jedoch in Griff gehalten werden kann.
Nachdem wir unser Gebiet Ω in eine Anzahl $|G|$ von Raumpunkten G diskretisiert haben, müssen die Variablen der Differentialgleichung ebenfalls diskretisiert werden.

Diskretisierung des Ortsterms:

In der Implementierung des Filters nach Schulte[17] wird zur Diskretisierung des Ortsterms das *Finite Volumen Verfahren* verwendet. Wir widmen uns diesem Verfahren. Für andere Verfahren, wie das *Finite Differenzen Verfahren* und das *Finite Elemente Verfahren* sei auf Wittum[22] bzw. Schwarz[18] verwiesen.

Begriffserklärung:

1. *Triangulierung*: Die räumlichen Punkte $g \in G$ werden linear miteinander verbunden, im zweidimensionalen Fall zu Dreiecken und Vierecken, im dreidimensionalen Fall zu Tetraedern und Hexaedern. Diese Strukturierung heißt Triangulierung.

2. *Gitter*: Die Vereinigung der Punkte (*Knoten*) $g \in G$ und der Triangulierung bildet ein Gitter. Dabei unterscheidet man zwischen *strukturiertem* und *unstrukturiertem* Gitter. $\mathbb{Z} \times \mathbb{Z}$ mit einer quadratischen Triangulierung beispielsweise ist ein strukturiertes Gitter.

Wir wählen ein strukturiertes Hexaedergitter mit den Voxelmittelpunkten als Knoten.

Finite Volumen Verfahren:

Definition: Ein Volumen $V_i \in \Omega$ heißt *Kontrollvolumen* zum Punkt $x_i \in G$ wenn es folgende Eigenschaften besitzt:

- $x_i \in V_i$
- $\bigcup_i V_i = \Omega$
- $V_i \cap V_j = 0$ für $i \neq j$ auf $\Omega \backslash Gitter$

Wir konstruieren solche Kontrollvolumina indem wir die Massenschwerpunkte mit den dazugehörigen Kantenmittelpunkten (im zweidimensionalen Fall) bzw. den Flächenschwerpunkten (im dreidimensionalen Fall) verbinden.
Somit sind die Bildvoxel gerade die Kontrollvolumina.

Integration über V_i:

$$\int_{V_i} \frac{\partial u}{\partial t} \, \mathrm{d}x = \int_{V_i} \nabla \cdot (D(u) \nabla u) \, \mathrm{d}x \quad \forall V_i \tag{3.23}$$

Das Finite Volumen Verfahren basiert auf der *Greenschen Formel* (Forster[9]):

Satz: Sei $A \subset \mathbb{R}^n$ eine kompakte Teilmenge mit glattem Rand, $\nu : \partial A \to \mathbb{R}^n$ das äußere Einheits-Normalenfeld und $U \supset A$ eine offene Teilmenge von $\mathbb{R}^n$. Dann gilt für jedes stetig differenzierbare Vektorfeld $F : U \to \mathbb{R}^n$

$$\int_A divF(x) \, \mathrm{d}^n x = \int_{\partial A} < F(x), \nu(x) > \, \mathrm{d}S(x)$$

Dabei ist S ein Flächenelement.
Zum Beweis verweisen wir auf Forster[9]. Mit der Greenschen Formel lässt sich das Volumenintegral aus (3.23) in ein Randintegral umschreiben:

$$\int_{V_i} \nabla \cdot (D(u) \nabla u) \, \mathrm{d}x = \int_{\partial V_i} (D(u) \nabla u) \cdot \vec{n} \, \mathrm{d}S \quad \forall V_i \tag{3.24}$$

Approximation von u:

Wir approximieren u in den Gitterpunkten durch

$$u(x) = \sum_{j=1}^{|G|} \alpha_j \varphi_j(x) \tag{3.25}$$

mit den *Ansatzfunktionen* φ_j.

Es gilt

- $\varphi_j(x_i) = \delta_{ij}$
- $Träger(\varphi_j)$ = Elemente die x_j als Eckpunkt besitzen

Es folgt $u(x_i) = \alpha_i$. Die α_i sind somit die diskrete Darstellung von u.
Aus den Gleichungen (3.24) und (3.25) erhalten wir

$$\begin{aligned}\int_{\partial V_i} (D(u)\nabla u)) \cdot \vec{n}\,\mathrm{d}S &\approx \int_{\partial V_i} \left(D(u)\nabla \left(\sum_j \alpha_j \varphi_j \right) \right) \cdot \vec{n}\,\mathrm{d}S \\ &= \sum_j \alpha_j \int_{\partial V_i} (D(u)\nabla\varphi_j) \cdot \vec{n}\,\mathrm{d}S \quad \forall V_i \quad (3.26)\end{aligned}$$

Diskretisierung des Zeitterms:

Als numerisches Verfahren zur Diskretisierung des Zeitterms wird ein *Vorwärtsdifferenz*-Verfahren angewandt (siehe Wittum[22]). Dabei wird die Ableitung $\frac{\partial u}{\partial t}$ durch einen rechtsseitigen Differenzenquotient ersetzt. Man erhält die Approximation

$$\int_{V_i} \frac{\partial u}{\partial t} \approx \int_{V_i} \frac{u^{t+1} - u^t}{\tau}\,\mathrm{d}x \quad \forall V_i$$

mit Zeitschrittweite τ.
Die nächste Approximation die wir vollziehen ist

$$\int_{V_i} \frac{u^{t+1} - u^t}{\tau}\,\mathrm{d}x \approx \frac{\alpha_i^{t+1} - \alpha_i^t}{\tau} \cdot |V_i| \quad \forall V_i$$

3.5.3 Numerische Lösung des diskretisierten Problems

Berechnung der α_i:

Aus den Gleichungen der Diskretisierung des Zeitterms und der Approximation des Integrals können die folgenden Rechenschemata abgeleitet werden:

1. Explizites Verfahren:

$$\alpha_i^{t+1} = \alpha_i^t + \frac{\tau}{|V_i|} \left(\sum_j \alpha_j^t \int_{\partial V_i} (D(u^t)\nabla\varphi_j) \cdot \vec{n}\,\mathrm{d}s \right) \quad \forall V_i \qquad (3.27)$$

Man erkennt, die Berechnung des neuen Werts im Zeitschritt $t+1$ hängt ausschließlich von den Werten des vorherigen Schrittes ab. Daher heißt dieses Verfahren *explizites* Verfahren.

2. Semiimplizites Verfahren:

$$\alpha_i^{t+1} = \alpha_i^t + \frac{\tau}{|V_i|}\left(\sum_j \alpha_j^{t+1} \int_{\partial V_i} (D(u^t)\nabla\varphi_j)\cdot\vec{n}\,\mathrm{d}s\right) \quad \forall V_i \qquad (3.28)$$

Dieses Schema hängt von α^{t+1} und α^t ab, ist also *semiimplizit*.

In unserem Fall wird aus Stabilitätsgründen das semiimplizite Verfahren benutzt.Wir definieren

$$a_{ij} := \frac{1}{|V_i|}\left(\int_{\partial V_i} (D(u^t)\nabla\varphi_j)\cdot\vec{n}\,\mathrm{d}s\right)$$

und formulieren Gleichung (3.28) in Matrixschreibweise:

$$\alpha^t = (\mathbb{E} - \tau A)\,\alpha^{t+1} \qquad (3.29)$$

Als letzter Schritt bleibt die

Approximation des Integrals:

Wir führen das Integral – also die kontinuierliche Darstellung – in die diskrete Darstellung einer Summe über:

$$\begin{aligned} a_{ij} &= \frac{1}{|V_i|}\left(\int_{\partial V_i} (D(u^t)\nabla\varphi_j)\cdot\vec{n}\,\mathrm{d}s\right) \\ &\approx \frac{1}{|V_i|}\sum_k (D(u)\nabla\varphi_j(ip_k))\cdot\vec{n}_k\cdot|S_k| \end{aligned} \qquad (3.30)$$

Bezeichnungen:

- Integrationspunkte ip_k = Stützstellen auf ∂V_i.
- $\vec{n}_k$ = Normale auf ∂V_i in ip_k.
- $|S_k|$ = ein- oder zweidimensionales Volumen des Teilstücks $v_k \ni ip_k$.

Zuletzt müssen die Ansatzfunktionen $\varphi_j(ip_k)$ geeignet gewählt und deren Ableitung $\nabla\varphi_j(ip_k)$ berechnet werden.
Dazu wird ein *Referenzelement* definiert, in unserem Fall ist dies ein Würfel

mit Kantenlänge 1. Zu diesem Referenzelement werden lokale Ansatzfunktionen Φ_i aufgestellt und setzt diese in Beziehung zu den φ_j:

$$\varphi_j\left(T_e(\xi,\eta,\vartheta)\right) = \Phi_{loc(j,e)}(\xi,\eta,\vartheta) \tag{3.31}$$

ξ, η, ϑ sind die Koordinaten im Referenzraum, die Funktion $loc(j,e)$ eine Funktion die dem Element e und Knoten j den Index der Ansatzfunktion zu j auf dem Element e zuweist.
Die Transformationsfunktion T_e verbindet das Referenzelement mit den Elementen e. In unserem Fall ist $T_e = id$, falls die Basisvektoren beider Räume bis auf Skalierung gleich sind.
Unter diesen Voraussetzungen erhält man

$$\nabla\varphi_j(ip(x,y,z)) = \nabla\Phi_{loc(j,e)}\left(ip(\xi,\eta,\vartheta)\right)$$

Um das Referenzelement zu beschreiben sind folgende trilineare Funktionen implementiert:

$$\begin{aligned}
\Phi_0(\xi,\eta,\vartheta) &= (1-\xi)(1-\eta)(1-\vartheta)\\
\Phi_1(\xi,\eta,\vartheta) &= \xi(1-\eta)(1-\vartheta)\\
\Phi_2(\xi,\eta,\vartheta) &= (1-\xi)\eta(1-\vartheta)\\
\Phi_3(\xi,\eta,\vartheta) &= \xi\eta(1-\vartheta)\\
\Phi_4(\xi,\eta,\vartheta) &= (1-\xi)(1-\eta)\vartheta\\
\Phi_5(\xi,\eta,\vartheta) &= \xi(1-\eta)\vartheta\\
\Phi_6(\xi,\eta,\vartheta) &= (1-\xi)\eta\vartheta\\
\Phi_7(\xi,\eta,\vartheta) &= \xi\eta\vartheta
\end{aligned}$$

Um die natürliche Randbedingung zu berücksichtigen, summieren wir nur über die inneren Integrationspunkte ip_{in}, die Matrixeinträge a_{ij} berechnen sich also aus

$$a_{ij} = \frac{1}{|V_i|}\sum_{ip_{in}}(D(u)\nabla\varphi_(ip_{in}))\cdot\vec{n}_k\cdot|S_k|$$

Die Information, *welche* Knoten gekoppelt sind und *wie* diese gekoppelt sind, lässt sich mit Hilfe eines *Sterns* darstellen.

Bei einer linearen Diffusion ($D = diag(1,1,1)$) erhält man folgenden Stern (Schulte[17]):

$$\begin{bmatrix} 0.0469 & 0.1563 & 0.0469 \\ -0.1563 & 0.1875 & -0.1563 \\ 0.0469 & 0.1563 & 0.0469 \end{bmatrix} \begin{bmatrix} 0.1563 & 0.1875 & 0.1563 \\ -0.1875 & -3.375 & -0.1875 \\ 0.1563 & 0.1875 & 0.1563 \end{bmatrix} \begin{bmatrix} 0.0469 & 0.1563 & 0.0469 \\ -0.1563 & 0.1875 & -0.1563 \\ 0.0469 & 0.1563 & 0.0469 \end{bmatrix}$$

Zur Erläuterung der Schreibweise einer Matrix in Form eines *k-Punkt-Sterns* verweisen wir auf Wittum[23].
Unser anfänglich analytisches mathematisches Problem der nichtlinearen anisotropen Diffusion haben wir nun in diskreter Darstellung formuliert und daraus ein Gleichungssystem abgeleitet. Der letzte Abschnitt dieses Kapitels liefert eine Lösungsmethode des entstandenen Gleichungssystems.

3.5.4 Löser

Wie lösen wir ein Gleichungssystem? Eine erste Idee wäre das *Gauß-verfahren.*
Bei einem gegebenen Gleichungssystem $Ax = b$ welches eindeutig lösbar ist, zerlegt man nach Gauß die Matrix A sukzessive, bis A auf Dreiecksgestalt gebracht ist. Man beginnt in der ersten Spalte und eliminiert alle Einträge unter dem Diagonaleintrag. Diese Elimination lässt sich mit Hilfe einer Eliminationsmatrix T^1 durchführen. Diese Matrix T^1 muss gespeichert werden.
So verfährt man mit allen anderen Spalten und erhält pro Spalte eine Eliminationsmatrix T^i mit unterer Dreiecksgestalt. Das Matrixprodukt T der T^i ist wieder eine Dreiecksmatrix und besitzt die Eigenschaft

$$A = \begin{pmatrix} * & & * \\ & \ddots & \\ 0 & & * \end{pmatrix} \cdot T^{-1}$$

Das Gleichungssystem kann nun durch Rückwärtseinsetzen gelöst werden. Eine ausführliche Beschreibung des Gaußverfahrens findet man in Wittum[21].

Bei diesem eher naiven Ansatz stoßen wir auf verschiedene Probleme:

1. Die Größe des Gleichungssystems führt zu Speicherproblemen
2. Die Komplexität des Lösungsverfahrens führt zu sehr großem Zeitaufwand

Die Matrix aus unserem Gleichungssystem lässt sich durch einen Stern wie in Abschnitt 3.5.3 beschreiben. Viele der Matrixeinträge sind Null und nur die Knoten, die mit einem ausgewählten Knoten gekoppelt sind, sind von Null verschieden. In unserem Fall wird ein Knoten durch 27 Matrixeinträge beschrieben. Diese Matrix gehört zu den *dünn besetzen* Matrizen (engl.: *sparse matrices*). Eine Matrix $A = (a_{ij})_{i,j\in\mathbb{N}}$ gilt dann als dünn besetzt, wenn gilt

$$\#\{a_{ij} : a_{ij} \neq 0\} \in \mathcal{O}(n) \tag{3.32}$$

Die Tatsache, dass wir mit einer dünn besetzten Matrix arbeiten, machen wir uns zu Nutze.
Speichert man lediglich alle von Null verschiedenen Einträge, so ist das Problem der Speicherkapazität behoben. Wendet man jedoch das Gaußverfahren auf diese Matrix an, entstehen an Stellen die zuvor Null waren nun nicht-Null Einträge, d.h. die Matrix verliert die Eigenschaft (3.32).
Wir benötigen also ein neues Lösungsverfahren und bedienen uns deshalb eines *Mehrgitterverfahrens*. Bevor die Funktionsweise des Mehrgitterverfahrens erläutert werden kann, stellen wir vorher die Eigenschaften der *iterativen* Löser zusammen.

Iterative Löser:

Wir definieren eine Folge $(x^i)_{i\in\mathbb{N}}$ durch $x^{i+1} = f(x^i)$ aus dem sich das lineare konsistente Verfahren

$$x^{i+1} = x^i - K^{-1}(Ax^i - b) \tag{3.33}$$

ableiten lässt. K heißt eine zu A *ähnliche* Matrix, K^{-1} die *angenäherte Inverse* von A. Die spezielle Wahl von K führt zu verschiedenen iterativen Lösungsverfahren:

1. *Richardson-Verfahren*: $K = \kappa\mathbb{E}$
2. *Jacobi-Verfahren*: $K = D$
3. *Gauß-Seidel-Verfahren*: $K = L + D$
4. *Gauß-Verfahren*: $K = A$

Für diese Verfahren wurde die Matrix A in eine untere Dreiecksmatrix L, eine Diagonalmatrix D und eine obere Dreiecksmatrix U zerlegt. Durch die Annäherung von A erhält man einen Invertierungsaufwand in $\mathcal{O}(n)$.
Bei den Verfahren (1)-(3) und anderen Verfahren wie die ILU-Zerlegung (engl.: *incomplete lower-upper decomposition*) sowie das Gradientenverfahren (siehe dazu Hackbusch[10]) erhält man die Dünnbesetztheit der Matrix und hat zusätzlich einen linearen Invertierungsaufwand. Die im Rahmen des Gaußverfahrens erwähnten Probleme sind dadurch behoben.
Ein weiteres Problem eröffnet sich jedoch; die langsame Konvergenz. Dies führt uns zum oben erwähnten Mehrgitterverfahren.

Mehrgitterverfahren:

Wendet man beispielsweise das Jakobiverfahren auf die gegebene Differentialgleichung an, stellt man fest das hochfrequente Eigenvektoren des Gleichungssystems stark gedämpft werden, dagegen werden Eigenvektoren mit niedriger Frequenz nur schwach gedämpft. Man bezeichnet solche Verfahren deshalb auch als *Glätter* (engl.: *Smoother*).
Nach einigen Glättungsschritten weisen diese Verfahren sehr langsame Konvergenz auf. An dieser Stelle wechselt man deshalb auf ein gröberes Gitter. Aufgrund der Glattheit der Funktionen ist dies möglich. Diesen Schritt nennt man *Restriktion*. Der Defekt wird nun auf dem gröberen Gitter gelöst und im nächsten Schritt wieder auf das feinere Gitter *prolongiert*.
Der Algorithmus eines Zweigitterverfahrens hat folgende Gestalt (Wittum[23]):
Gegeben sei ein Startwert $u^{(0)}$. Dann berechnet sich $u^{(i+1)}$ aus $u^{(i)}$ durch

$\bar{u}_2 = S_2^{\nu_1} u_2^{(i)}$	Glättung auf Gitter 2 (feines Gitter)
$d_2 = A_2 \bar{u}_2 - g_2$	Berechnung des Defekts
$d_1 = r d_2$	Restriktion des Defekts
$\tilde{u}_1 = A_1^{-1} d_1$	Exakte Lösung auf Gitter 1 (grobes Gitter), Approximation der Korrektur, nicht der Lösung
$\tilde{u}_2 = \bar{u}_2 - p\tilde{u}_1$	Korrektur durch Prolongation der Lösung von Gitter 1
$u_2^{(i+1)} = S_2^{\nu_2} \tilde{u}_2$	Nachglättung

S definiert dabei die Iterationsmatrix. Im Falle des Jakobiverfahrens angewandt auf (3.33) ist $S = \mathbb{E} - D^{-1}A$. g beschreibt die rechte Seite der Differentialgleichung – hier also $g = \nabla \cdot (D(u) \cdot \nabla u)$ – d den *Defekt*.
Zur Restriktions- und Prolongationsvorschrift verweise ich auf Wittum[23].

Kapitel 4

Aufbau des Filters

Die in Abschnitt 3.5 hergeleitete Lösungsmethode der in Kapitel 3 definierten Diffusionsgleichung wurde im Rahmen des Projekts *NeuRA* von Roland Schulte[17] implementiert.
Dieses Kapitel liefert einen Überblick über die verschiedenen Kategorien des Filters und deren dazugehörigen Klassen. Dabei wird das Hauptaugenmerk auf den Klassen des Filters liegen, die eine numerische Optimierung in Bezug auf konfokale Mikroskopiedaten von Neuronenzellkernen ermöglichen.

4.1 Kategorien des Filters

Die Klassen des Filters können in sieben Kategorien unterteilt werden:

1. Hilfsstrukturen
2. Zentrale Datenstruktur
3. Berechnung der Trägheitsmomente
4. Matrix-Datenstruktur
5. Diskretisierung
6. Löser
7. Steuerung des Filters

Die logischen Zusammenhänge sind in Abbildung 4.1 verdeutlicht. Abbildung 4.2 listet die Klassen zu den verschiedenen Kategorien auf.
Für mehr Information über den Aufbau der Klassen aus den Kategorien *Hilfsstrukturen* und *Zentrale Datenstruktur* verweise ich auf Schulte[17].

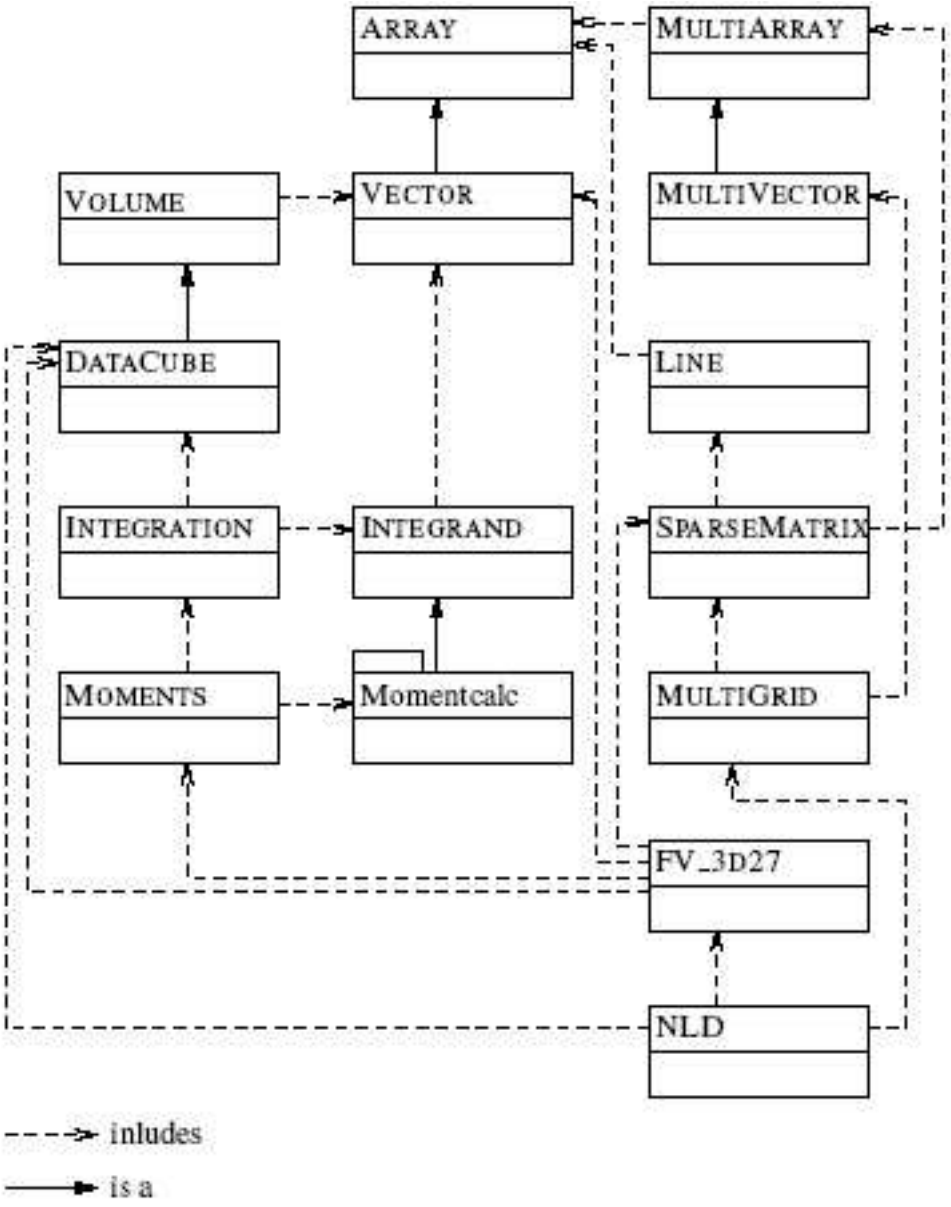

Abbildung 4.1: Übersicht der Filter-Klassen (Schulte[17])

Kategorie	Klassen
Hilfsstrukturen	ARRAY VECTOR MULTIARRAY MULTIVECTOR
Zentrale Datenstruktur	VOLUME DATACUBE
Berechnung der Trägheitsmomente	INTEGRATION INTEGRAND Momentcalc MOMENTS
Matrix-Datenstruktur	SPARSEMATRIX LINE
Diskretisierung	FV_3D27
Löser	MULTIGRID
Steuerung des Filters	NLD

Abbildung 4.2: Klassenkategorien (Schulte[17])

4.2 Trägheitsmomente

Die Idee in Abschnitt 3.4 zur Ermittlung der Diffusionsrichtungen war die Berechung der physikalischen Trägheitsmomente, deren Eigenwerte und Eigenvektoren Aufschluss über die Geometrie des zu filternden Objekts liefern.
Es muss also die Gleichung (3.16) berechnet werden, sowie deren Eigenwerte und Eigenvektoren. Dies geschieht über die Klassen aus Kategorie *Berechung der Trägheitsmomente*.
In der Klasse INTEGRATION werden Methoden zur Integration über ein gewähltes Gebiet zur Verfügung gestellt. Berechnet wird dabei eine diskrete Form

$$T_V = \sum_{p \in V} g(u(p), p) \tag{4.1}$$

Wählbare Parameter sind das Integrationsgebiet V und die Funktion g.

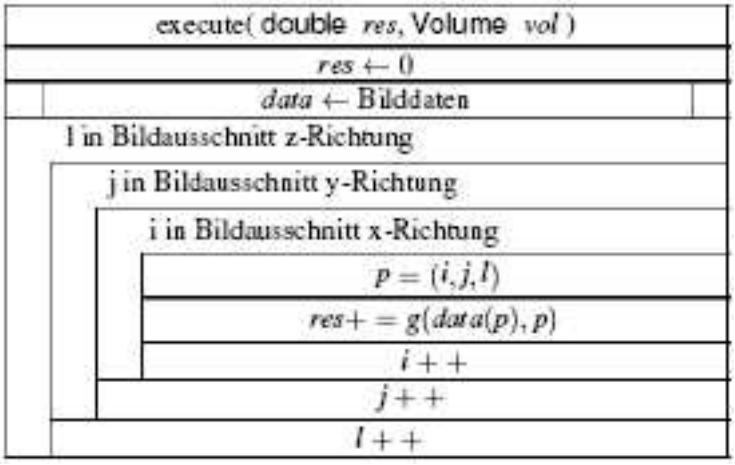

Abbildung 4.3: Integration auf einem kubischen Gebiet (Schulte[17])

Das Integrationsgebiet V:

Als Integrationsgebiet stehen zwei Möglichkeiten zur Verfügung:

1. *Kubisches* Integrationsgebiet
2. *Kugelförmiges* Integrationsgebiet

Der Algorithmus für die Integration auf einem kubischen Gebiet ist in Abbildung 4.3 dargestellt. In diesem Fall schließt das Integrationsgebiet immer eine ganze Anzahl von Bildpunkten ein, beim kugelförmigen Integrationsgebiet ist dies nicht mehr der Fall. Das kugelförmige Gebiet schließt manche Bildpunkte nur zum Teil ein.
Um den Anteil des im Integrationsgebiet liegenden Bildpunktes zu ermitteln wird folgende Funktion eingeführt:

$$w(x) = \begin{cases} 1 & \text{Voxel ganz im Integrationsgebiet enthalten} \\ 0 & \text{Voxel außerhalb des Integrationsgebiets} \\ a(x) & \text{sonst} \end{cases}$$

Liegt ein Voxel nur zum Teil in der Kugel, wird dieses rekursiv in kleinere Teile zerlegt, bis zu einer festgelegten Verfeinerungsstufe. Die verfeinerten Quader die ganz in der Kugel enthalten sind werden mit 1 gewichtet, die nur teilweise enthaltenen mit 0.5. Das Gewicht des Voxels berechnet sich aus der Summe der kleineren Quader.
Um die unterschiedliche Gewichtung bei der Integration zu berücksichtigen, ergibt sich

$$T_V = \sum_{p \in V} w(p) \cdot g(u(p), p) \tag{4.2}$$

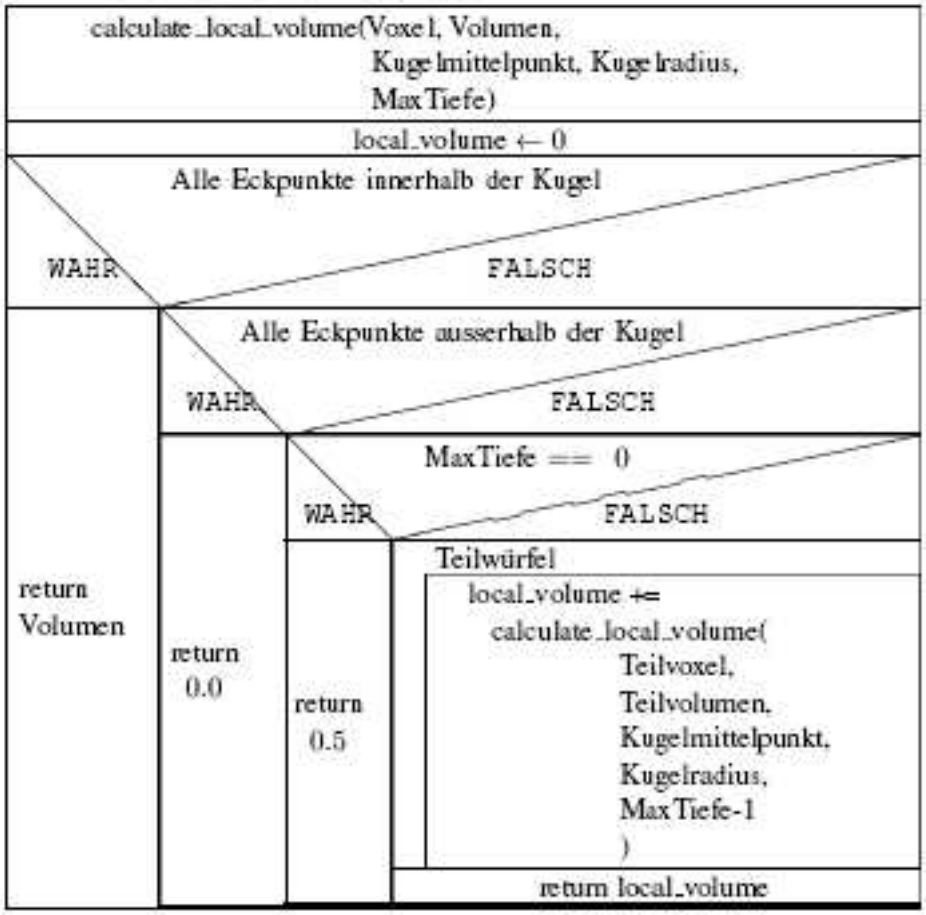

Abbildung 4.4: Integration auf einem kugelförmigen Gebiet (Schulte[17])

Die Gewichtung wird gemäß Abbildung 4.4 im Algorithmus eingebunden.

Die Funktion g:

Über die Klasse INTEGRAND wird die Wahl der Funktion g getroffen. g kann für folgende Aufgaben aufgerufen werden:

- Berechnung des Volumens:

  $$g(u(p), p) = 1$$

  (Implementiert in der Klasse VOLUMEINTEGRAND)

- Berechnung der Masse:

  $$g(u(p), p) = u(p)$$

  (Implementiert in der Klasse MASSINTEGRAND)

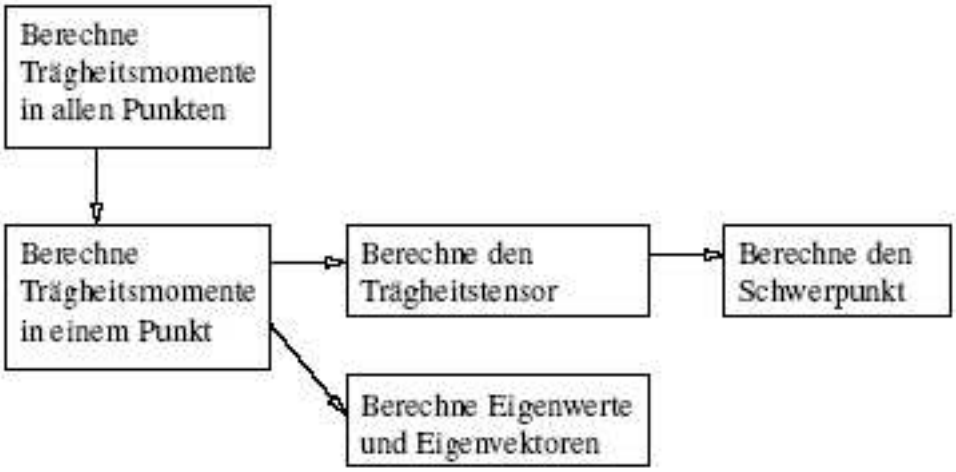

Abbildung 4.5: Methoden von MOMENTS (Schulte[17])

- Berechnung des Schwerpunkts:

$$g(u(p),p) = \begin{cases} u(p) \cdot p_x & \text{in der Klasse FOCALPOINT\_X} \\ u(p) \cdot p_y & \text{in der Klasse FOCALPOINT\_Y} \\ u(p) \cdot p_z & \text{in der Klasse FOCALPOINT\_Z} \end{cases}$$

  (für $p = (p_x, p_y, p_z)$)

- Berechnung der Matrixeinträge des Trägheitstensors (Implementiert in den INERTIAN_TENSOR-Klassen)

MOMENTS ist eine Klasse die Methoden zur Berechnung von Schwerpunkt, Trägheitstensor und Trägheitsmomenten enthält. Abbildung 4.5 zeigt den Zusammenhang der verschiedenen Methoden. Die Berechnung der Trägheitsmomente für alle Punkte des DATACUBE (eine zentrale Datenstruktur zur Speicherung der Bilddaten, siehe Abb. 4.2) ist zeitlich nicht optimiert.
Für ein kubisches Integrationsgebiet wird in Schulte[17] eine Beschleunigung durch Zusammenfassung von Punkten entwickelt. Bei der Wahl eines kugelförmigen Integrationsgebiets wird eine zeitliche Optimierung durch *Fast-Fourier-Transformation* erreicht. Zur Herleitung und Implementierung wird an dieser Stelle auf Schulte[17] verwiesen.
In der Kette der zu berechnenden Größen folgt die Berechnung der Eigenwerte und Eigenvektoren. Dazu wurde ein QR-Algorithmus verwendet auf den hier nicht näher eingegangen wird. Das Konzept des QR-Algorithmus findet sich in Schwarz[18].

4.3 Diskretisierung

FV_3D27 ist die Klasse die das Anfangsrandwertproblem

$$\begin{array}{ll} \frac{\partial u}{\partial t} = \nabla \cdot (D(u) \cdot \nabla u) & \text{auf } \mathbb{R}^+ \times \Omega \\ u(x,0) = u_0(x) & \text{auf } \bar{\Omega} \\ (D(u) \cdot \nabla u) \cdot \vec{n} = 0 & \text{auf } \mathbb{R}^+ \times \partial\Omega \end{array}$$

auf einem Hexaedergitter der Kantenlänge 1 löst. Die Methoden von FV_3D27 führen eine *Raumdiskretisierung* nach der Vorschrift

$$a_{ij} = \frac{1}{|V_i|} \sum_k (D(u) \nabla \varphi_j(ip_k)) \cdot \vec{n}_k \cdot |S_k|$$

durch. Diese Matrix wird mit Hilfe der Klasse SPARSEMATRIX als dünn besetzte Matrix gespeichert. Die *Zeitdiskretisierung* erfolgt nach

$$A = \mathbb{E} - \tau A$$

(siehe Abschnitt 3.5.3)

Wählbare Parameter:

Folgende Parameter stehen für die Filterregulierung zur Verfügung:

1. Das Integrationsgebiet auf dem die Trägheitsmomente berechnet werden lässt sich durch integration_size_x, integration_size_y, integration_size_z einstellen.

2. Die in Abschnitt 4.2 beschriebenen Integrationsgebiete lassen sich mit Hilfe von geometry_type wählen.

3. Mit ip_flag wählt man zentrierte Integrationspunkte auf dem subcontrol volume (IP_USUAL) oder verschobene Integrationspunkte (IP_BND).

4. Die Wahl der Anisotropieparameter t_1, t_2, t_3 wird mit fixed_coeffs getroffen. Ist fixed_coeffs aktiviert (YES), wird folgende Belegung gewählt:

$$\begin{aligned} t_1 &= 1 \\ t_2 &= g(c_l) \\ t_3 &= g(1 - c_i) \end{aligned}$$

   g ist dabei entweder (3.6), (3.7) oder (3.8). Sind diese deaktiviert (NO), können die Anisotropiekoeffizienten frei gewählt werden.

5. Die freie Wahl der Anisotropiekoeffizienten findet mit anicoeff1, anicoeff2, anicoeff3 statt.

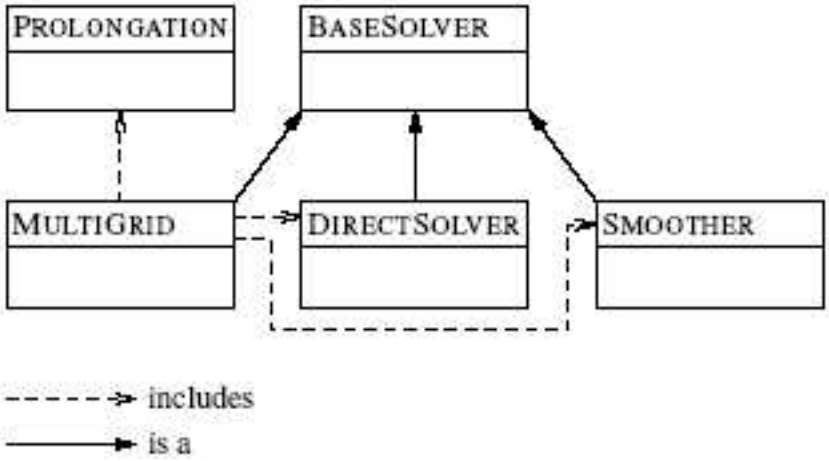

Abbildung 4.6: Zusammenhang der Löser-Klassen (Schulte[17])

6. DependenceType gibt die drei Optionen (3.6), (3.7) und (3.8) frei mit den Eingaben PERONA_MALIK, WEIKERT und BLACK_SAPIRO.
7. lambda legt den Parameter von g fest.

4.4 Löser

Der Löser für das Gleichungssystem welches die Diffusion beschreibt wurde in der Klasse MULTIGRID – von Christoph Reisinger geschrieben – implementiert. Abbildung 4.6 zeigt die an der Lösung beteiligten Klassen. Dabei übernimmt

- PROLONGATION den Transfer
- SMOOTHER die Glättung
- DIRECTSOLVER die Lösung auf dem groben Gitter
- BASESOLVER die Funktionen der iterativen Verfahren

Wählbare Parameter in MULTIGRID:

1. Transferoperationen
2. Glätter
3. Anzahl der Vorglättungsschritte
4. Anzahl der Nachglättungsschritte
5. Art des Mehrgitterzyklus

6. Grobgitterlöser
7. Anzahl der Gitterlevel
8. Reduktion des Anfangsdefekts
9. Maximale Zyklenzahl

4.5 Die Klasse NLD: Steuerelement des Filters

Die Hauptklasse Nld ist die Steuerklasse des Filters. Zum einen lassen sich die Diskretisierungsparameter in Fv_3d27 festlegen, zum anderen kann die Zeitdiskretierung parametrisiert werden. Zur Verfügung stehen die Parameter

1. time_steps zur Wahl der Zeitschritte
2. tau zur Wahl der Zeitschrittweite
3. precision zur Wahl der Defektreduzierung
4. levels zur Wahl der Mehrgitterlevel

Im folgenden Kapitel werden die in diesem Kapitel aufgeführten Klassen mit ihren freien Parametern in Hinblick auf das Filterresultat auf konfokalen Mikroskopiedaten von Neuronenzellkernen getestet und eine numerische Optimierung für die Bildqualität durchgeführt.

Kapitel 5

Numerische Optimierung des Filters

Ziel dieses Kapitels ist es, eine stabile Einstellung der Parameter des Filters zu finden, für die eine optimale Filterung von Zellkernmikroskopiedaten erreicht wird. Optimale Filterung hat zwei Bedeutungen: Entweder wenn eine ideale Bildqualität für darstellungstechnische Zwecke erreicht wird oder in Hinblick auf die Rekonstruktion, wenn eine Extrahierung der Geometrie des Kerns möglich ist.
Jeder Zellkern zeichnet sich durch eine sehr individuelle Geometrie aus, ähnlich wie ein Fingerabdruck. Es ist also anfangs nicht klar, dass eine feste Wahl der Parameter für alle Zellkernaufnahmen möglich ist.
Um später aus den gefilterten Daten Aussagen über die Oberfläche sowie das Volumen machen zu können, ist es wichtig die Struktur des Kerns möglichst unverändert zu lassen. Bei der Wahl der Parameter wird also die Geometrie des Zellkerns unsere Richtlinien bestimmen.
Im Rahmen dieses Projekts werden ausführliche Testreihen – zunächst auf Testgeometrien, dann auf Mikroskopiedaten – durchgeführt. Dabei werden 30 Zellkerne untersucht. Um die Übersichtlichkeit dieses Kapitels zu wahren, werden wir uns bei der Visualisierung der Ergebnisse auf einen Zellkern beschränken.
Wir untersuchen zunächst auf Testgeometrien die Diffusionsrichtungen, das Integrationsgebiet und Zeitschrittweite sowie Zeitschrittzahl und stellen diese Parameter so ein, dass später für die Filterung von echten Mikroskopiedaten die beste Qualität erreicht wird.
Dazu wurden verschiedene Testgeometrien in der Klasse Testcubes angelegt:

1. Plane3d: Ebene im Raum mit einem Voxel Dicke.

2. Plane3d_Holes: Ebene im Raum mit Löchern vom Durchmesser 2, 4, 6, 8 und 10 Voxeln.

3. Sphere3d: Kugel im Raum mit Radius 52.
4. Sphere3d_Holes: Kugel im Raum mit Radius 52 und Löchern mit den Durchmessern 7, 12 und 22.

Alle weiteren Parameter werden anschließend auf echten Zellkerndaten optimiert.
Folgende Parameter werden in diesem Kapitel optimiert:

1. time_steps = 2
2. tau = 2.0
3. epsilon = 0,001
4. levels = 1
5. integration_size = 10^3 Voxel
6. GeometryType = CUBE
7. ip_flag = IP_USUAL
8. anicoeffs: anicoeff1 = 1, anicoeff2 = 1, anicoeff3 = 0

Die Parameter behalten die oben angegebenen Voreinstellungen, außer denen die in den einzelnen Abschnitten variiert und geprüft werden.
Im Folgenden werden die Werte der Anisotropiekoeffizienten als 3-Tupel geschrieben.

5.1 Parameteruntersuchung auf Testgeometrien

5.1.1 Die Diffusionsrichtungen

In Abschnitt 4.3 aufgeführt, besitzt man verschiedene Möglichkeiten der Diffusionssteuerung. Wir bedienen uns dazu der Parameter anicoeff1, anicoeff2 und anicoeff3.
Gewünscht ist die Diffusion entlang der Zellkernmembran, also eine gekrümmte zweidimensionale Oberfläche.

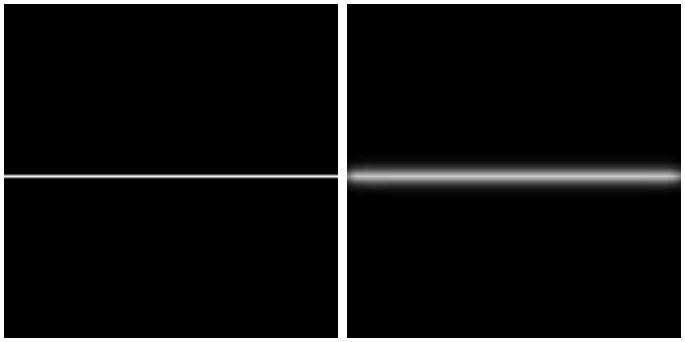

Abbildung 5.1: links: Querschnitt einer ungefilterten Ebene ohne Löcher, rechts: In allen Raumrichtungen gefilterte Ebene (Queisser (SiT), 2005)

Untersuchung auf einer Ebene ohne Löchern:

Betrachten wir zunächst eine Ebene ohne Löcher und variieren die Anisotropiekoeffizienten:

1. **Einstellung der Anisotropiekoeffizienten auf (1,1,1):**

   Wir lassen ungedämpfte Diffusion in alle drei Raumrichtungen zu. Abbildung 5.1 zeigt den Effekt des Filters auf die Ebene.
   Die Punkte der Ebene sind mit dem Grauwert 1 initialisiert, der Rest des Würfels mit 0. Die Filterung sollte deshalb keinen sichtbaren Effekt auf die Ebene haben. Aufgrund der Diffusion in alle Raumrichtungen erhält man jedoch eine Verbreiterung der Ebene in z-Richtung.

2. **Einstellung der Anisotropiekoeffizienten auf (1,1,0):**

   Die Diffusion in z-Richtung wird auf 0 gesetzt und volle Diffusion in x- und y-Richtung zugelassen. Abbildung 5.2 zeigt den gewünschten Effekt, die Ebene bleibt unverändert in z-Richtung.

3. **Einstellung der Anisotropiekoeffizienten auf (1,0,0):**

   Wird die Diffusion lediglich in x-Richtung zugelassen, erhält man wie in Abbildung 5.2 keine Änderung in z-Ebene. Wir werden später jedoch sehen, dass zwischen den Einstellungen (1,1,0) und (1,0,0) unterschieden werden muss.

Die Untersuchung auf einer Ebene mit Löchern mit den Voxeldurchmessern 2, 4, 6, 8 und 10 (Abbildung 5.3) liefert, wie zu erwarten, die gleichen Resultate wie die Untersuchung auf Plane3d.

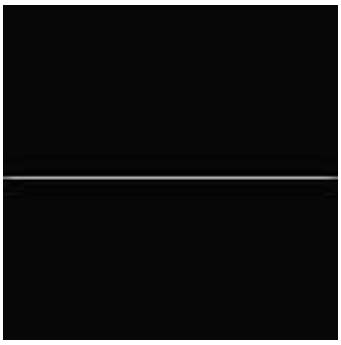

Abbildung 5.2: Querschnitt einer in xy-Ebene gefilterten Ebene (Queisser (SiT), 2005)

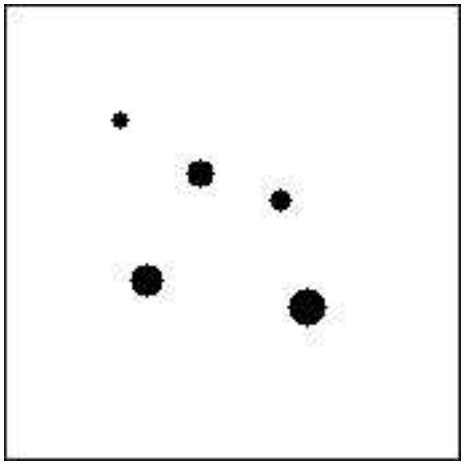

Abbildung 5.3: Ebene mit Löchern verschiedener Durchmesser (Queisser (SiT), 2005)

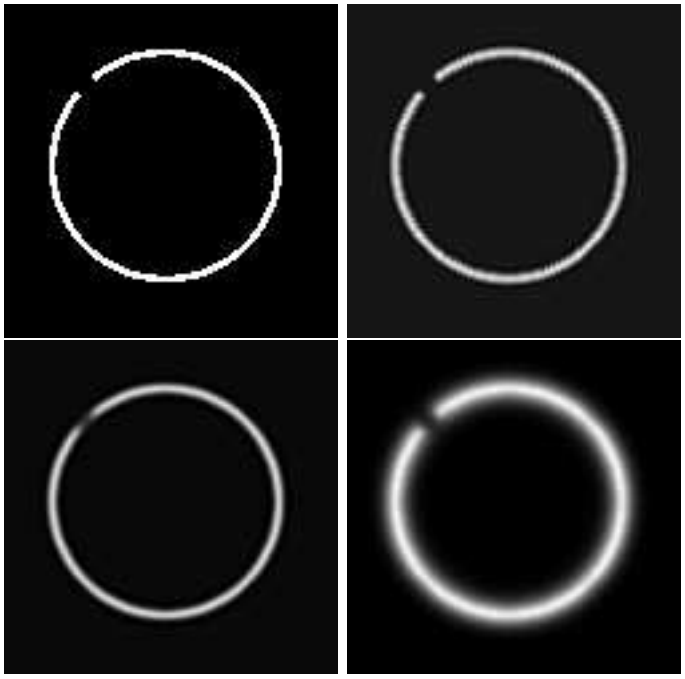

Abbildung 5.4: l.o.: Ungefiltert, r.o.: Gefiltert mit (1,0,0), l.u.: Gefiltert mit (1,1,0), r.u.: Gefiltert mit (1,1,1), (Queisser (SiT), 2005)

Untersuchung auf einer Kugel mit Löchern:

Bisher haben wir Testgeometrien ohne Krümmung betrachtet. Die Tangentialrichtung auf einer Kugel ändert sich, wie auch bei der Zellkernmembran, punktweise. Wir wollen nun untersuchen, wie sich der Filter bei punktweiser Richtungsänderung verhält.
Parallel dazu betrachten wir den Effekt des Filters auf Lücken in der Kugeloberfläche.

Veränderung der Anisotropiekoeffizienten:

Abbildung 5.4 zeigt die unterschiedlichen Ergebnisse bei der Veränderung der Diffusionsrichtungen, Abbildung 5.5 liefert dazu eine dreidimensionale Darstellung.
Obwohl sich die Tangentialrichtung in jedem Punkt ändert erhält der Filter bei der Koeffizienteneinstellung (1,1,0) den Durchmesser der Kugelschale und bewirkt eine Glättung der Oberfläche. In den anderen Fällen bleibt entweder der Durchmesser erhalten, gleichzeitig ist der Glättungseffekt nicht optimal, oder ein Glättungseffekt ist sichtbar, der jedoch mit einer starken Geometrieveränderung einher geht. An dieser Stelle wird der Unterschied zwischen den Einstellungen (1,0,0) und (1,1,0) deutlich, was bei der

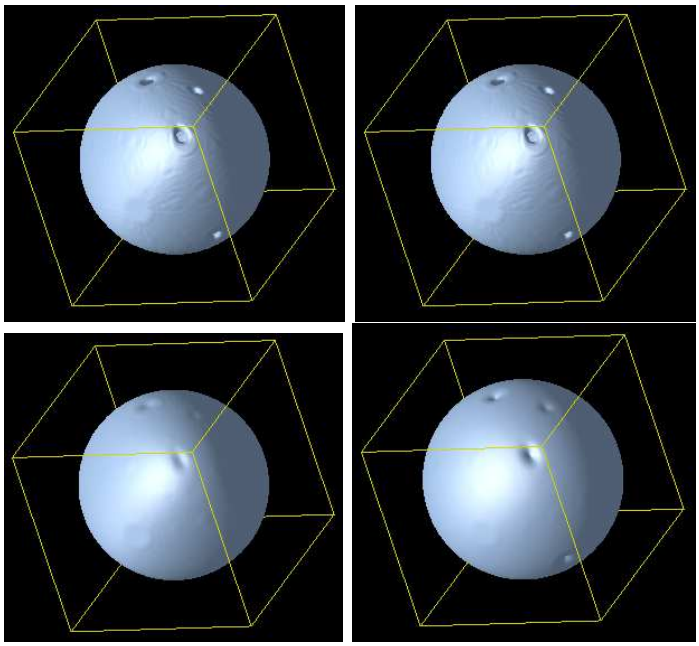

Abbildung 5.5: l.o.: Kugel mit Löchern ungefiltert, r.o.: Gefiltert mit (1,0,0), l.u.: Gefiltert mit (1,1,0), r.u.: Gefiltert mit (1,1,1), (Queisser (SiT), 2005)

Untersuchung der Ebene noch nicht zu erkennen war. Abbildung 5.6 zeigt die unterschiedlichen Grauwertverteilungen nach den drei Filterprozessen. Da die Zellkernmembranoberfläche eine kugelähnliche Geometrie vorweist, ist mit der Wahl der Anisotropiekoeffizienten anicoeff1 = 1, anicoeff2 = 1 und anicoeff3 = 0 der beste Effekt zu erwarten.

5.1.2 Das Integrationsgebiet

Wie groß sollte das Integrationsgebiet gewählt werden?
Im Vordergrund steht die Strukturerkennung, d.h. bei Löchern in der Struktur muss das Integrationsgebiet groß genug sein um den Fortlauf der Struktur zu erkennen. Auf der anderen Seite sollte das Integrationsgebiet nicht zu groß sein, denn sonst könnten lokale Richtungsänderungen der Struktur wie z.B. Einstülpungen der Membran die Hauptdiffusionsrichtungen verfälschen. Wir testen deshalb verschiedene Größen des Integrationsgebiets.

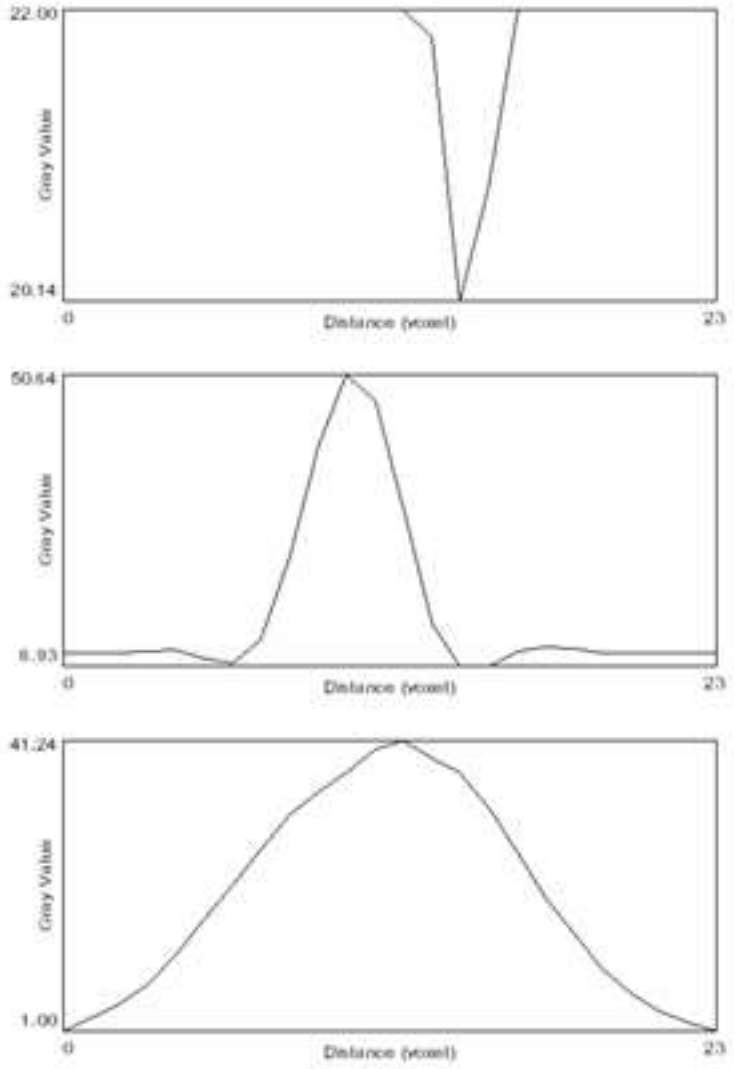

Abbildung 5.6: Grauwertverteilung nach Filterung mit (1,0,0), (1,1,0), (1,1,1), (Queisser (SiT), 2005)

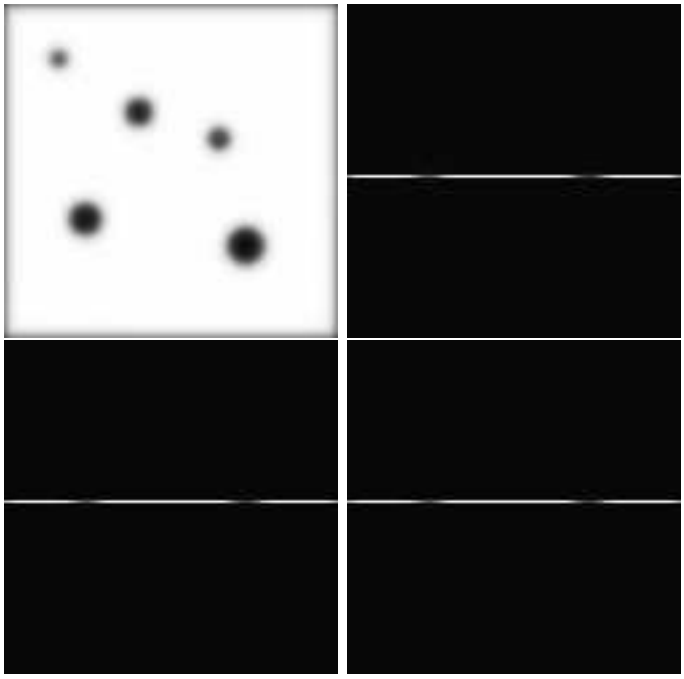

Abbildung 5.7: l.o.: Gefilterte Ebene mit integration_size = 2 von oben, r.o.: Querschnitt der gefilterten Ebene mit integration_size = 2, l.u.: Gefiltert mit integration_size = 10, r.u.: Gefiltert mit integration_size = 30, (Queisser (SiT), 2005)

Untersuchung auf einer Ebene mit Löchern:

Wir wählen drei verschiedene Größen des Integrationsgebiet:

- Kleines Integrationsgebiet: integration_size = 2
- Mittleres Integrationsgebiet: integration_size = 10
- Großes Integrationsgebiet: integration_size = 30

Abbildung 5.7 zeigt den Effekt der Filterung mit variabler Integrationsgröße. Der Vergleich der Grauwertverteilung in der ungefilterten und in der gefilterten Ebene in Abbildung 5.8 demonstriert das Resultat der Filterung entlang der linken Lücke (im Querschnitt betrachtet). Bei diesem Test ist noch kein wesentlicher Unterschied zwischen den unterschiedlichen Integrationsgebieten zu erkennen. Dass die Größe des Integrationsgebiets jedoch eine Rolle spielt, wird sich im nächsten Test zeigen.

Untersuchung auf einer Kugel mit Löchern:

Bei der Untersuchung einer Ebene mit Löchern scheinte die Wahl des Integrationsgebiets keine Unterschiede in der Filterung zu bewirken. Wir wollen dies weiter untersuchen. Dazu wählen wir wieder die Kugel mit Löchern. Die unterschiedlichen Ergebnisse sind in Abbildung 5.9 im Querschnitt zu sehen.
Einen großen Unterschied erkennt man bei der Betrachtung der dreidimensionalen Darstellung in Abbildung 5.10. Abbildung 5.11 zeigt die zugehörigen Grauwertverteilungen.
Da wir mit der Kugel wieder näher an der Geometrie eines Zellkerns sind als mit der Ebene, ist auch für die Mikroskopiedaten das beste Resultat mit einer mittleren oder großen Integrationsgebietgröße zu erwarten. Die Tatsache, das die Membranoberfläche eines Zellkerns – anders als bei der Kugel – Variationen im Krümmungsgrad besitzt und feinere Strukturen vorhanden sind, muss man darauf achten, dass man nicht durch ein zu großes Integrationsgebiet Strukturinformation durch die Filterung verliert.

5.1.3 Zeitschrittweite und Zeitschrittzahl

Die Zeitschrittweite und Zeitschrittzahl sind die letzten Parameter die wir auf der Kugeltestgeometrie untersuchen. Diese zwei Parameter nehmen wesentlichen Einfluss auf das Resultat der Filterung, weshalb wir deren Effekt zunächst in dem kontrollierbaren Feld der Testgeometrien prüfen wollen.

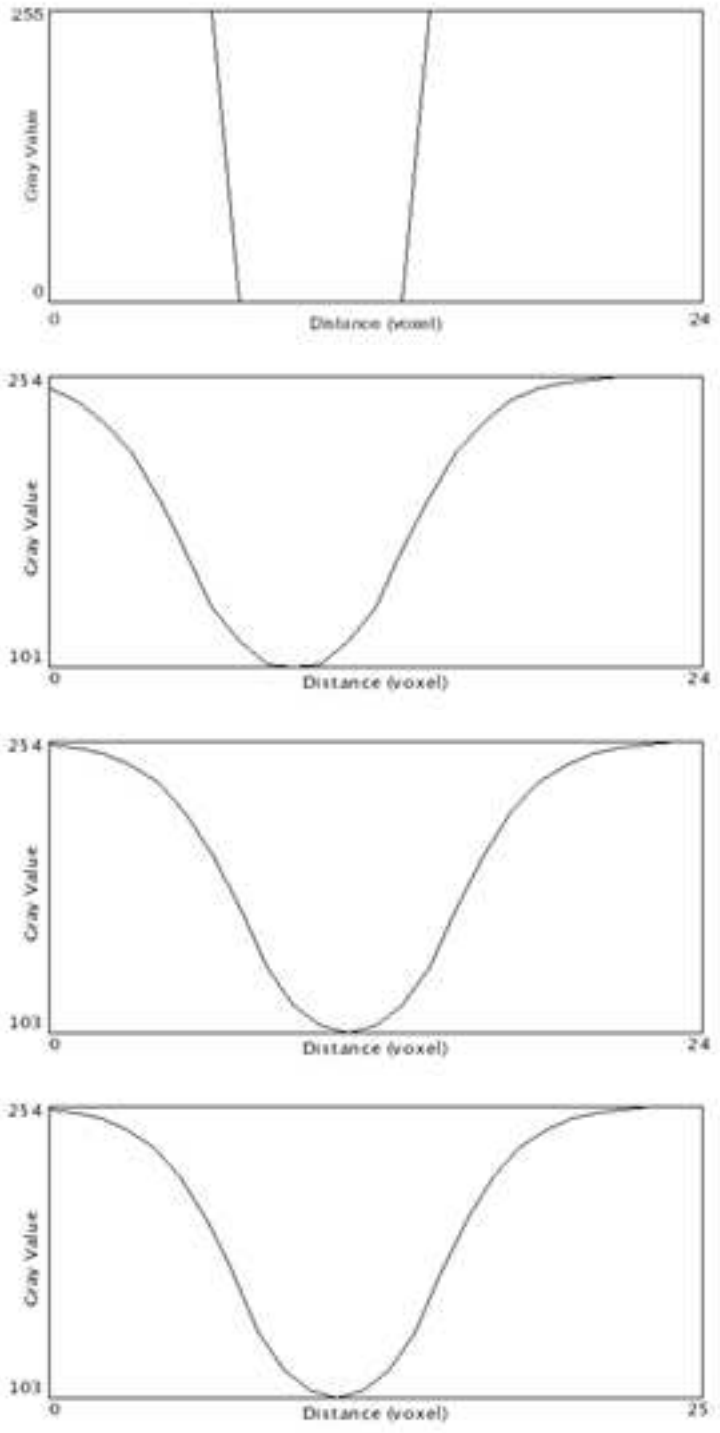

Abbildung 5.8: *von oben nach unten*: 1. Grauwertverteilung entlang der linken Lücke des ungefilterten Bilds, 2. Grauwertverteilung des gefilterten Bilds mit integration_size = 2, 3. Grauwertverteilung des gefilterten Bilds mit integration_size = 10, 4. Grauwertverteilung des gefilterten Bilds mit integration_size = 30, (Queisser (SiT), 2005)

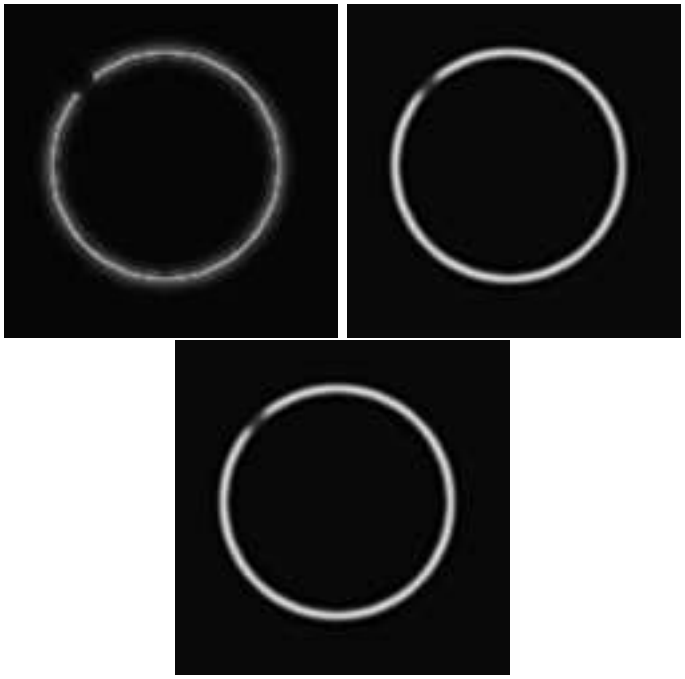

Abbildung 5.9: *l.o.*:Integrationsgebiet = 2; *r.o.*: Integrationsgebiet = 10; *unten*: Integrationsgebiet = 30, (Queisser (SiT), 2005)

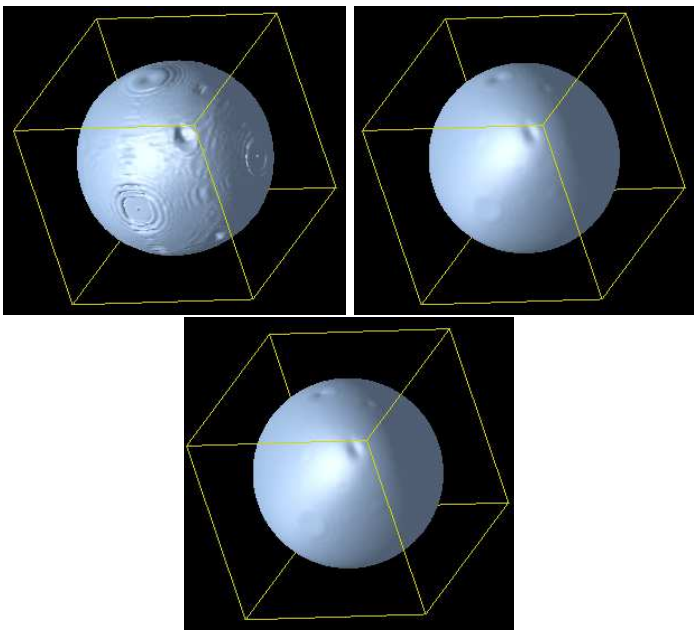

Abbildung 5.10: Gefilterte Kugel mit Löchern mit Integrationsgebieten 2 (l.o.), 10 (r.o.), 30 (unten), (Queisser (SiT), 2005)

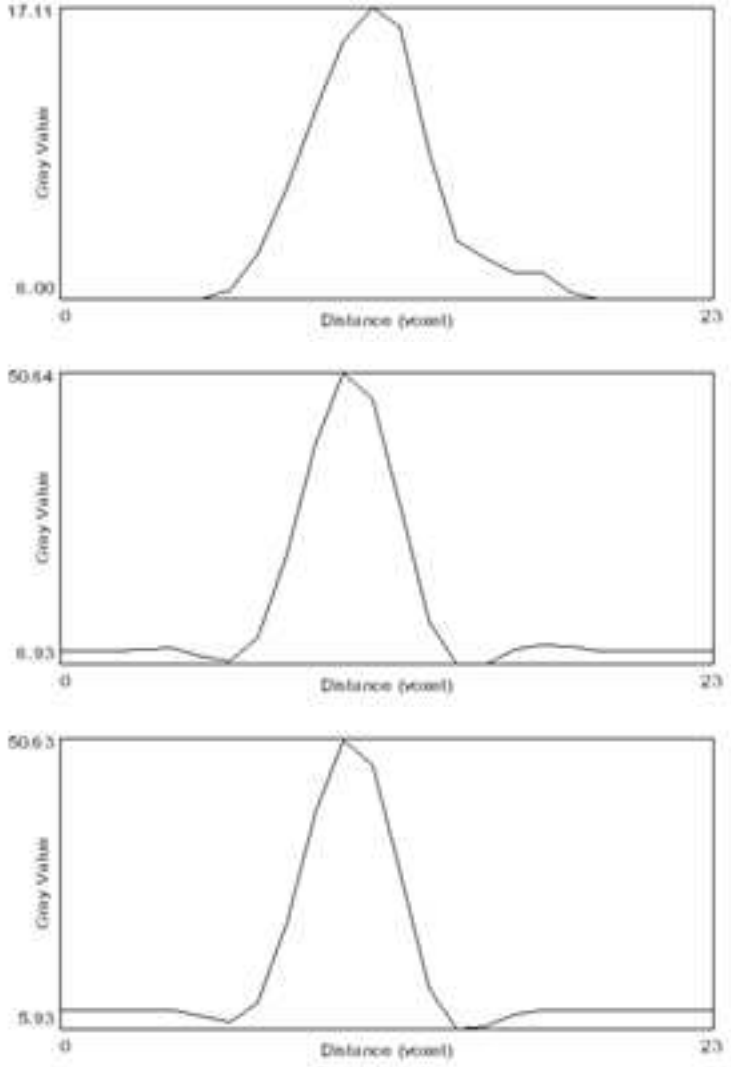

Abbildung 5.11: Grauwertverteilung durch das Kugelloch mit Integrationsgebieten 2, 10, 30 (von oben nach unten), (Queisser (SiT), 2005)

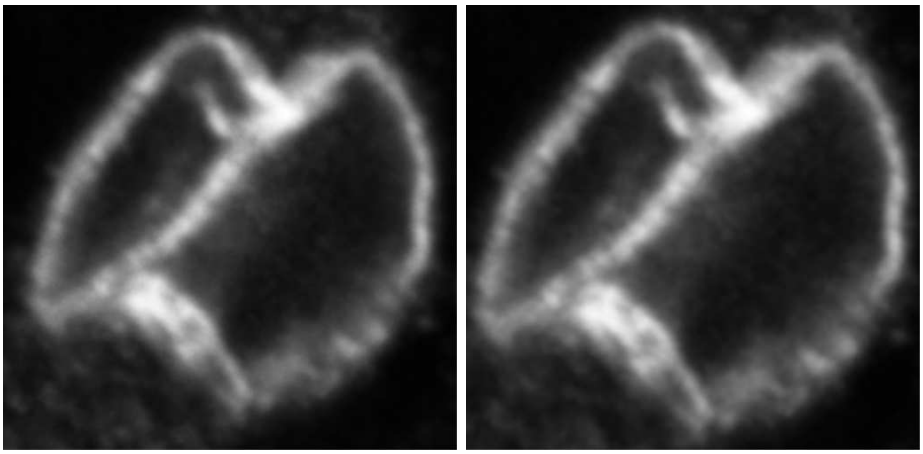

Abbildung 5.12: links: gefiltert mit tau = 10 und time_steps = 1, rechts: tau = 1 und time_steps = 4, (Queisser (SiT), 2005)

Bei der folgenden Testreihe betrachten wir die drei Löcher in der Kugel mit den Durchmessern 7, 12 und 22 Voxeln und variieren die Zeitschrittweite. Zu jeder Zeitschrittweite wird geprüft, wieviele Zeitschritte zur Schließung der Löcher notwendig sind.
Ein Loch gilt als geschlossen, wenn folgendes Kriterium erfüllt ist:

> Ein Loch ist dann geschlossen, wenn dieses nach einer globalen Segmentierung mit dem Schwellenwert 0.3 – d.h. jeder Wert unterhalb des Grauwerts 0.3 wird auf 0 gesetzt, jeder darüber liegende auf 1 – verschwindet.

Tabelle 5.1 stellt die Ergebnisse dieser Versuchsreihe zusammen. Optisch ist bei Anwendung auf den Testgeometrien kein Unterschied zwischen der Wahl einer kleinen Zeitschrittweite zusammen mit einer größeren Zahl von Zeitschritten und einer großen Zeitschrittweite zusammen mit wenigen Zeitschritten zu erkennen.
Falls dies auch für reelle Daten der Fall sein sollte, ist die Wahl einer Zeitschrittweite von ca. tau = 2 aus rechenzeitlichen Gründen zu empfehlen. Da das zugrunde liegende numerische Problem relativ statisch ist – d.h. die Bilddaten ändern sich zwar in der Zeit aufgrund der Filterung, der Diffusionstensor wird dadurch jedoch kaum beeinflusst – ist die Wahl der Zeitschrittweite physikalisch und numerisch nicht eingeschränkt.
Abbildung 5.12 zeigt das Ergebniss der Filterung einmal mit großer, einmal mit kleiner Zeitschrittweite.

Zeitschrittweite	10.0	2.0	1.0	0.5
Anzahl Zeitschritte: Loch 1 (d = 10)	1	2	3	10
Anzahl Zeitschritte: Loch 2 (d = 15)	1	3	5	11
Anzahl Zeitschritte: Loch 3 (d = 20)	1	4	5	17

Tabelle 5.1: Messergebnisse zu Zeitschrittweite und Zeitschrittzahl

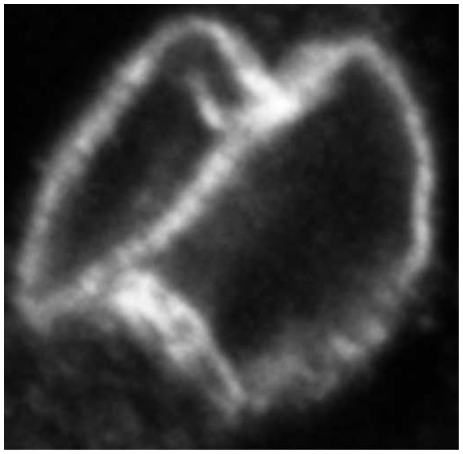

Abbildung 5.13: gefiltertes Bild mit Idealparametern, (Queisser (SiT), 2005)

5.2 Einstellung des Filters für konfokale Mikroskopiebilder von Zellkernen

5.2.1 Übernahme der auf TESTGEOMETRIES untersuchten Parameter

Die in Abschnitt 5.1 untersuchten Parameter wurden in Hinblick auf Zellkerngeometrie untersucht und bieten mit den dort vorgeschlagenen Einstellungen eine feste Voreinstellung die auf konfokale Mikroskopiedaten angewandt werden kann und im allgemeinen Rahmen die optimalen Ergebnisse produziert.
Die Wahl der Parameter ist natürlich weiterhin abhängig davon, welches Ziel mit der Filterung verfolgt wird. Möchte man bestimmte Teile des Kerns möglichst genau visualisieren? Oder benötigt man vollkommen geschlossene Strukturen zur Vermessung, bei der kleine Details nicht in Gewicht fallen?
In den folgenden Kapiteln wird sich zeigen, wie die Parameter auf spezielle Aufgaben angepasst werden müssen. Abbildung 5.13 zeigt jedoch zunächst das Ergebnis der Filterung von mikroskopischen Aufnahmen mit den in Abschnitt 5.1 optimierten Parametern.

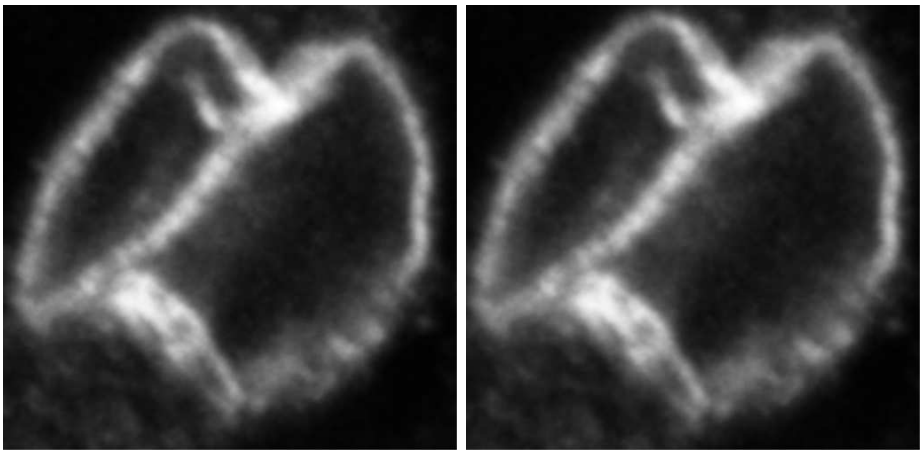

Abbildung 5.14: links: kubusförmiges Integrationsgebiet, rechts: kugelförmiges Integrationsgebiet, (Queisser (SiT), 2005)

5.2.2 Geometrie des Integrationsgebiets

Ein weiterer Parameter der zu untersuchen bleibt, ist die Geometrie des Integrationsgebiets. Zur Verfügung stehen hier ein *kubusförmiges* Integrationsgebiet oder ein *kugelförmiges* Integrationsgebiet.
Für die unterschiedlichen Integrationsgebiete sind zwei verschiedene Algorithmen implementiert, die in Kapitel 4 erwähnt sind (siehe dazu Abb. 4.3 und Abb. 4.4).
Da der Zellkern eine kugelähnliche Geometrie besitzt, wäre es anzunehmen, dass ein kugelförmiges Integrationsgebiet die Strukur besser erfasst und somit ein besseres Filterergebnis erzielt.
Abbildung 5.14 vergleicht den Effekt der zwei Geometrien des Integrationsgebiets.

Die Geometrie scheint bei diesen Daten keine große Rolle zu spielen. Ein Vergleich der Rechenzeit – beim Kubus liegt diese bei ca. 20 Minuten, bei der Kugel bei ca. 24 Stunden – bevorzugt die Wahl des kubusförmigen Integrationsgebiets.
Die großen zeitlichen Unterschiede liegen in den verschiedenen Varianten der zeitlichen Optimierung (siehe Abschnitt 4.2). Die Zusammenfassung von Punkten, die nur bei kubusförmigem Integrationsgebiet anwendbar ist, liefert eine bessere zeitliche Optimierung als die im kugelförmigen Integrationsgebiet angewandte Fast-Fourier-Transformation.

5.2.3 Art des Lösers und Glätters

In Schulte[17] wurden verschiedene Löser implementiert:

1. *Gauss-Seidel*-Löser
2. *BiCG-Stab*-Löser

Weiterhin werden unterschiedliche Glättungsverfahren untersucht, zum einen lineare Transfermethoden, zum anderen matrixabhängige Transfermethoden.
Die verschiedenen Möglichkeiten werden in Schulte[17] auf Mikroskopiedatensätzen von Neuronen ausreichend untersucht und optimiert. Die dort vorgeschlagenen Lösungs- und Glättungsverfahren können für Mikroskopiedaten von Neuronenzellkernen übernommen werden.
Das Resultat dieser Untersuchung schlägt die Verwendung des BiCGStab-Verfahrens vor, welchen in diesem Kapitel stillschweigend verwendet wurde. Zu Messergebnissen bzgl. des zeitlichen und qualitativen Unterschieds der verschiedenen Löser und Glätter verweise ich an dieser Stelle auf Schulte[17].
Damit ist die Untersuchung und numerische Optimierung der wählbaren Parameter abgeschlossen. Es besteht die Möglichkeit einer Grundeinstellung der Parameter, die trotz der stark variierenden Geometrie der Zellkerne nicht notwendigerweise von Fall zu Fall geändert werden müssen.
Der Filterprozeß bewirkt die Glättung der Membran des Zellkerns, eine Grundvoraussetzung zur Extrahierung der Geometrie. Was noch fehlt ist ein Verfahren welches das Signal der Membran vom Hintergrundrauschen trennt.
Das folgende Kapitel liefert dieses Verfahren.

Kapitel 6

Segmentierung

6.1 Prinzip der Segmentierung

Die für das menschliche Auge bereits aus den Rohdaten sichtbare Struktur des Kerns, ist für den Computer zunächst eine Ansammlung von Bildpunkten mit den zugehörigen Grauwerten.
Durch die Filterung bilden wir eine größere Zusammengehörigkeit der Bildpunkte, welche die Kernmembran zusammensetzen.
Nun gilt es, den Computer diese Zusammengehörigkeit erkennen zu lassen und die zur Membran gehörenden Bildpunkte vom Rest des Bildes zu trennen. Im Anschluss an die Filterung führen wir eine *Segmentierung* des Bildes durch, ein Verfahren zur Extrahierung der Zellkernmembran.
Dabei wird das Bild mit einer Grauwertverteilung im Intervall $I = [0, 255]$ (oder im skalierten Fall in $I_s = [0, 1]$) in ein binäres Bild transformiert. Die zur Struktur gehörenden Punkte sollen dabei auf den Grauwert 255 (bzw. 1), der Rest auf 0 gesetzt werden. Im Folgenden gehen wir, wenn nicht explizit definiert, immer vom skalierten Fall aus.
Es bleibt jedoch die Entscheidungsfrage, welche Bildpunkte zur Struktur gehören und welche zum Hintergrund. Diese Entscheidung treffen verschiedene Segmentierungsalgorithmen.

Konvention:
Wir unterscheiden bei den Mikroskopiebildern zwischen *Bild* und *Hintergrund*. Dabei bezeichnet das Bild die Membranstruktur, der Hintergrund den Rest des Mikroskopiebildes.

6.2 Verschiedene Segmentierungsalgorithmen

Die zu bewältigenden Aufgaben in der Bildverarbeitung sind sehr vielfältig. Es ist einleuchtend, dass man keine Universallösung im Bereich der Segmentierung finden wird, d.h. ein Segmentierungsalgorithmus für alle Aufgaben.

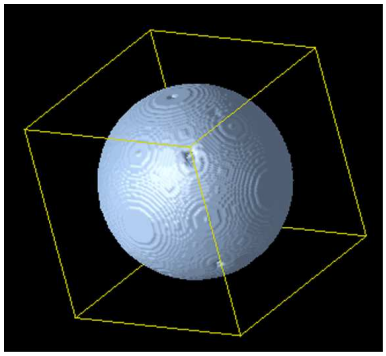

Abbildung 6.1: Global segmentiert mit $\vartheta = 0.3$, (Queisser (SiT), 2005)

Wir müssen also einen geeigneten Segmentierungsalgorithmus finden. Dabei werden wir uns hier auf sogenannte *Thresholding*-Segmentierungen beschränkten. Diese Art von Segmentierung ist dann empfehlenswert, wenn das Bild Grauwerte in einem Intervall $I_B = [r_0, r_1], \quad 0 \leq r_0 \leq r_1 \leq 1$ besitzt, die Grauwerte des Hintergrunds in $I_H = [h_0, h_1], \quad 0 \leq h_0 \leq h_1 \leq 1$ liegen, so dass $I_B \cap I_H$ "klein" oder idealerweise $I_B \cap I_H = \emptyset$. Diese Voraussetzung ist in unserem Fall gegeben. Bei der Thresholding-Segmentierung unterscheidet man grob zwischen einer *globalen* und *lokalen* Segmentierung.

6.2.1 Globale Segmentierung

Das Prinzip der globalen Segmentierung ist sehr einfach. Zu wählen ist ein globaler Schwellenwert (engl.: *global threshold*). Der Segmentierungsalgorithmus untersucht jeden Bildpunkt. Liegt dessen Grauwert über diesem Threshold ϑ wird ihm der Wert 1 zugeordnet, liegt dessen Grauwert unter ϑ gehört der Punkt zum Hintergrund und wird auf 0 gesetzt. Diese Segmentierungsfunktion ist definiert durch

$$s(x, y) := \begin{cases} 1 & \text{für } f(x, y) \geq \vartheta \\ 0 & \text{sonst} \end{cases}$$

f ist eine Funktion die jedem Bildpunkt seinen Grauwert zuordnet.
Betrachten wir dazu unsere in Kapitel 5 eingeführte Testgeometrie sphere3D_holes.
Das Bild wird zunächst gefiltert, und anschließend mit dem globalen Threshold $\vartheta = 0.3$ segmentiert (Abbildung 6.1).

Man erkennt, die Löcher in der Kugel werden kleiner, da die Lücke durch den Filterprozeß zunächst mit Grauwerten "aufgefüllt" wird, dessen zugehörige Bildpunkte durch die Segmentierung der Kugel zugeordnet werden. Bei der Wahl eines anderen Thresholds, z.B. $\vartheta = 0.2$ ändert sich das Bild, Abbildung 6.2 zeigt, dass sich in diesem Fall die Lücke komplett schließt.

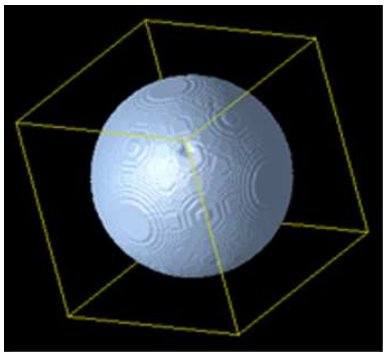

Abbildung 6.2: Global segmentiert mit $\vartheta = 0.2$, (Queisser (SiT), 2005)

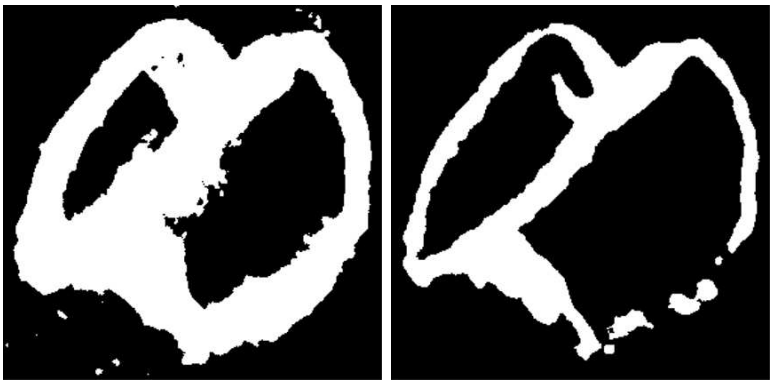

Abbildung 6.3: links: Global segmentiert mit $\vartheta = 0.2$, rechts: $\vartheta = 0.4$, (Queisser (SiT), 2005)

Die Aufteilung der Punkte in Bild und Hintergrund ist stark abhängig von der Wahl des globalen Thresholds.
Wenden wir diese Segmentierung als nächstes auf unseren Beispielfall an. Abbildung 6.3 zeigt das Resultat nach der Filterung und Segmentierung einmal mit dem Threshold $\vartheta = 0.2$ und einmal mit $\vartheta = 0.4$. Wie man sieht erhält man sehr unterschiedliche Ergebnisse. Wie man im linken Bild erkennen kann, verliert man durch die Segmentierung an manchen Stellen die filigraneren Strukturen der Einstülpungen, im rechten Bild werden Lücken in der Membran leider nicht geschlossen.

Die Resultate der globalen Segmentierung sind also zum Einen stark abhängig vom gewählten Threshold (dies birgt somit eine gewisse Willkür bei der Zuordnung der Punkte in Bild und Hintergrund), zum Anderen wird die globale Segmentieren den verschiedenen Wünschen nicht gerecht, filigrane Strukturen zu erhalten und gleichzeitig an anderer Stelle größere Lücken in der Membran zu schließen.
Aufgrund der Tatsache, dass der bei der Mikroskopie verwendete Farbstoff

beim Scannen (siehe Abschnitt 2.2.2) ausbleicht, ist das Signal in den unteren Regionen des Kerns schwächer als als in den oberen. Dementsprechend sollte für verschiedene Regionen des Bildes ein anderen Threshold gewählt werden.
Der sehr starre Algorithmus der globalen Segmentierung eignet sich für unseren Fall also nicht.

6.2.2 Lokale Segmentierung

Weit dynamischere Algorithmen als den, der in Abschnitt 6.2.1 besprochenen globalen Segmentierung liefert die lokale Segmentierung. Anstatt für das gesamte Bild einen Schwellenwert anzugegeben, passen wir diesen an die lokalen Gegebenheiten des Bildes an.
Wie in Abschnitt 6.2.1 zu erkennen gegeben wurde, haben wir gewisse Anforderungen an den Segmentierungsalgorithmus. Filigrane Einstülpungen die in manchen Fällen auch nah beieinander verlaufen sollen erhalten werden, gleichzeitig müssen Lücken in der Membranstruktur geschlossen werden. Dies ist, wie wir in Kapitel 7 sehen werden, für die Rekonstruktion des Kerns unverzichtbar.
Um diese scheinbar gegensätzlichen Wünsche in Einklang bringen zu können, wählen wir die lokale Segmentierung. In diesem Bereich finden sich diverse Methoden, wie die *Region-Growing*-Segmentierung oder die lokale Segmentierung nach der Otsu-Methode.

6.2.3 Otsu-Methode

Die Segmentierung nach Otsu, auch *Clustering* genannt, geht von folgender Idee aus:

> Die Gesamtmenge der Bildpunkte lässt sich in zwei Gruppen (*Cluster*) aufteilen. Der eine Cluster beinhaltet die Punkte, die zur Struktur des Kerns gehören, der andere die Hintergrundpunkte.

Diese zwei Cluster – als Histogramm dargestellt – überlappen jedoch. Die Otsu-Methode liefert ein Verfahren, welches Bildpunkte mit der größten Wahrscheinlichkeit dem richtigen Cluster zuzuordnen. Der Graph der Grauwertverteilung besteht aus zwei überlappenden Glockenkurven welche die Cluster beschreiben. Das Verfahren berechnet nun lokal einen idealen Threshold, der diese zwei Kurven mit der geringsten Fehlerwahrscheinlichkeit trennt.

Wir wollen dieses Konzept mathematisch beschreiben:
Die *Varianz* der Punkte innerhalb der Cluster lässt sich wie folgt beschreiben:

$$\sigma_I^2(\vartheta) = q_1 \cdot \sigma_1^2(\vartheta) + q_2 \cdot \sigma_2^2(\vartheta) \tag{6.1}$$

mit

$$\begin{aligned}
\vartheta &= \text{Threshold} \\
q_1(\vartheta) &= \sum_{i=0}^{\vartheta-1} p(i) \\
q_2(\vartheta) &= \sum_{i=\vartheta}^{N-1} p(i) \\
\sigma_1(\vartheta) &= \text{Varianz im ersten Cluster} \\
\sigma_2(\vartheta) &= \text{Varianz im zweiten Cluster} \\
p(i) &= \text{Wahrscheinlichkeit eines Punktes mit Grauwert } i
\end{aligned}$$

und N die Anzahl der verschiedenen Grauwerte (hier z.B. $N = 255$).
Die Klasse der "Zwischenpunkte" im Überlappungsbereich besitzt dann die Varianz

$$\sigma_B^2 = \sigma^2 - \sigma_I^2(\vartheta) \tag{6.2}$$

Gleichung(6.2) lässt sich mit Hilfe der zugehörigen Erwartungswerte μ schreiben als

$$\sigma_B^2 = q_1(\vartheta) \cdot (\mu_1(\vartheta) - \mu)^2 + q_2(\vartheta) \cdot (\mu_2(\vartheta) - \mu)^2 \tag{6.3}$$

Mit $\mu = q_1\mu_1 + q_2\mu_2$ folgt

$$\sigma_B^2 = q_1(\vartheta)q_2(\vartheta)\left(\mu_2(\vartheta) - \mu_2(\vartheta)\right)^2 \tag{6.4}$$

Der optimale Threshold ist also der, der σ_B maximiert.

Dazu müssen folgende Punkte abgearbeiten:

1. Aufteilen der Bildpunkte in zwei Cluster getrennt durch einen anfänglich gewählten Threshold.
2. Berechnen der Erwartungswerte der zwei Cluster.
3. Berechnen von $q_1(\vartheta)q_2(\vartheta)\left(\mu_2(\vartheta) - \mu_2(\vartheta)\right)^2$.

Abbildung 6.4: Segmentiert mit dem Otsu-Verfahren, (Queisser (SiT), 2005)

Wie Abbildung 6.4 zeigt, erhält die Otsu-Segmentierung feinere Strukturen und schließt gleichzeitig Lücken in der Membran.

Wir sind nun an einem Punkt angekommen, an dem die ersten dreidimensionalen Darstellungen gewonnen werden können. Im nächsten Abschnitt werden erste Resultate vorgestellt, gleichzeitig aber Mängel festgestellt, die den Weg zum weiteren Vorgehen weisen werden.

6.3 Erste Resultate

Als erste Visualisierungsmethode werden wir das Software-Paket *DataExplorer* verwenden. Dazu müssen die Rohdaten der Mikroskopie wie folgt aufgearbeitet werden:

1. Filterung der Daten im tiff-Format.
2. Segmentierung der Daten mit Hilfe der Otsu-Methode.
3. Einlesen der tiff-Datei mit anschließender Streckung in z-Richtung
4. Umwandeln in das *DataExplorer*-kompatible dx-Format.

Die Streckung des Bildes ist an dieser Stelle notwendig, da fixierte Zellkerne sehr flach sind und man deshalb bei einer Visualisierung mit dem *DataExplorer* wenig erkennen würde.
Zur Realisierung der Punkte (2)-(4) wurde ein Programm in der Sprache C++ geschrieben.
Abbildung 6.5 zeigt die dreidimensionale Struktur unseres Beispielkerns. Aus mikroskopischen Gründen folgt die *äußere Spur* der Membran den Einstülpungen *nicht*. Als Ganzes betrachtet sieht man also die eigentlich

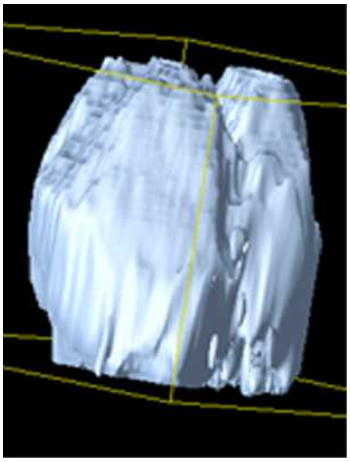

Abbildung 6.5: Dreidimensionale Darstellung mit dem *DataExplorer*, (Queisser (SiT), 2005)

interessante *innnere Spur*, die den Einstülpungen der Membran folgt, nicht.

Konvention:
Die Zellkernmembran besitzt durch die Streuung des Mikroskopiesignals eine gewisse Dicke. Wir unterscheiden deshalb zwischen der *äußeren* Spur und der *inneren* Spur.

Um einen Einblick ins Innere des Kern zu bekommen, wurde das obige Programm ergänzt. Es werden drei Schnittebenen in x-, y- und z-Richtung festgelegt und der Kern an diesen Ebenen aufgeschnitten. Das Programm legt zusätzlich zur Datei mit dem gesamten Kern für jede Kernhälfte eine weitere Datei an.
Die Abbildungen 6.6, 6.7 und 6.8 geben Einblick ins Innere verschiedener Kerne. Bei der geöffneten Darstellung sind die Einstülpungen der Membranen zu erkennen.

Diese Visualisierungsmethode weist jedoch etliche Mängel auf. Führen wir zunächst nochmal die gewünschten Ziele auf:

- 3D-Visualisierung der *inneren* Spur der Membran mit ihren Einstülpungen.
- Vermessung der Oberfläche der inneren Spur der Membran und dem eingeschlossenen Volumen.

Unser Verfahren bietet lediglich die Visualisierung als Ganzes. Weder kann der innere Bereich der Zelle isoliert visualisiert werden, noch diese Geometrie vermessen werden. Als nächstes muss also die Extrahierung des Innenraums des Zellkerns in Angriff genommen werden.

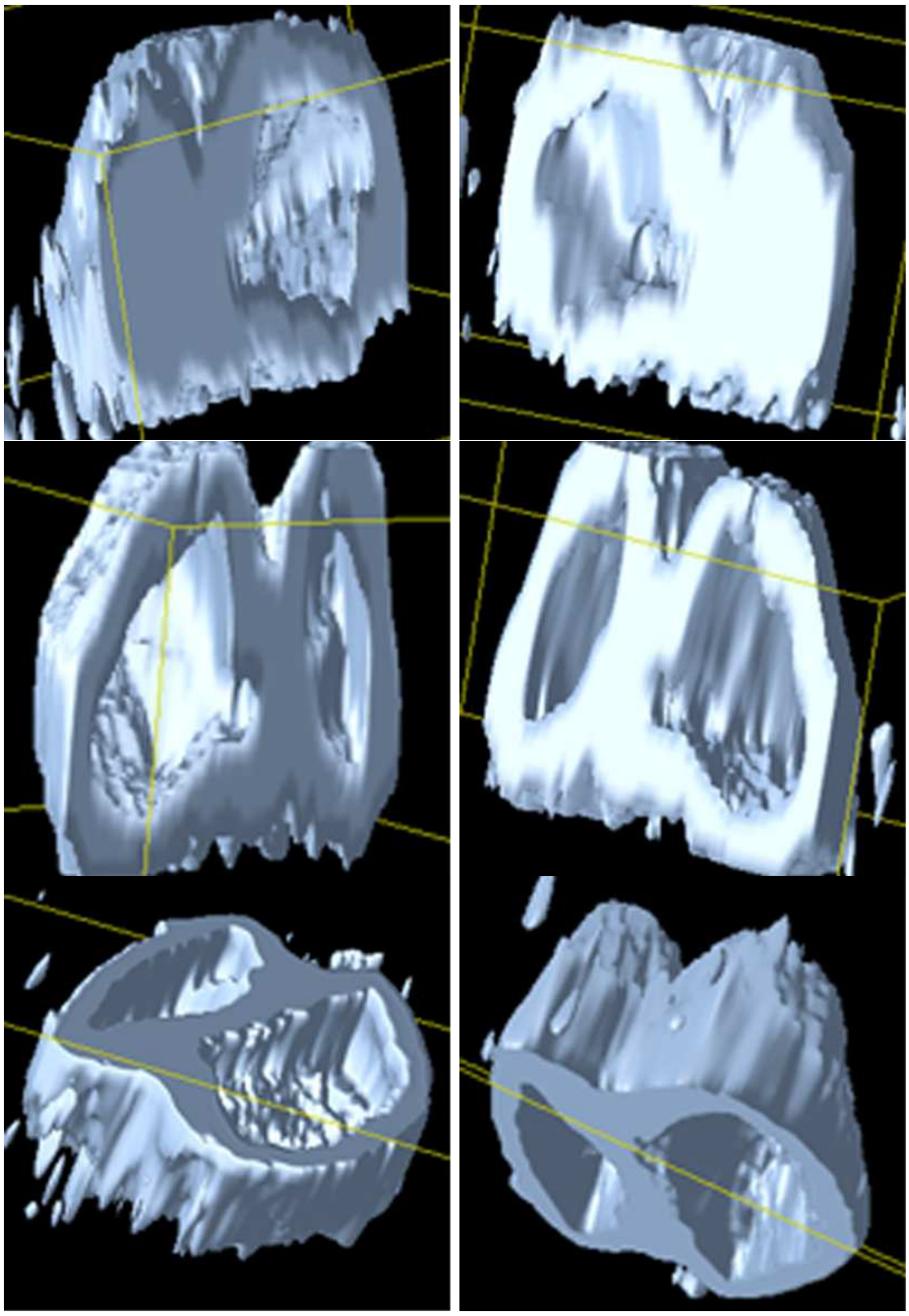

Abbildung 6.6: Blick in den Innenraum von Kern 1, (Queisser (SiT), 2005)

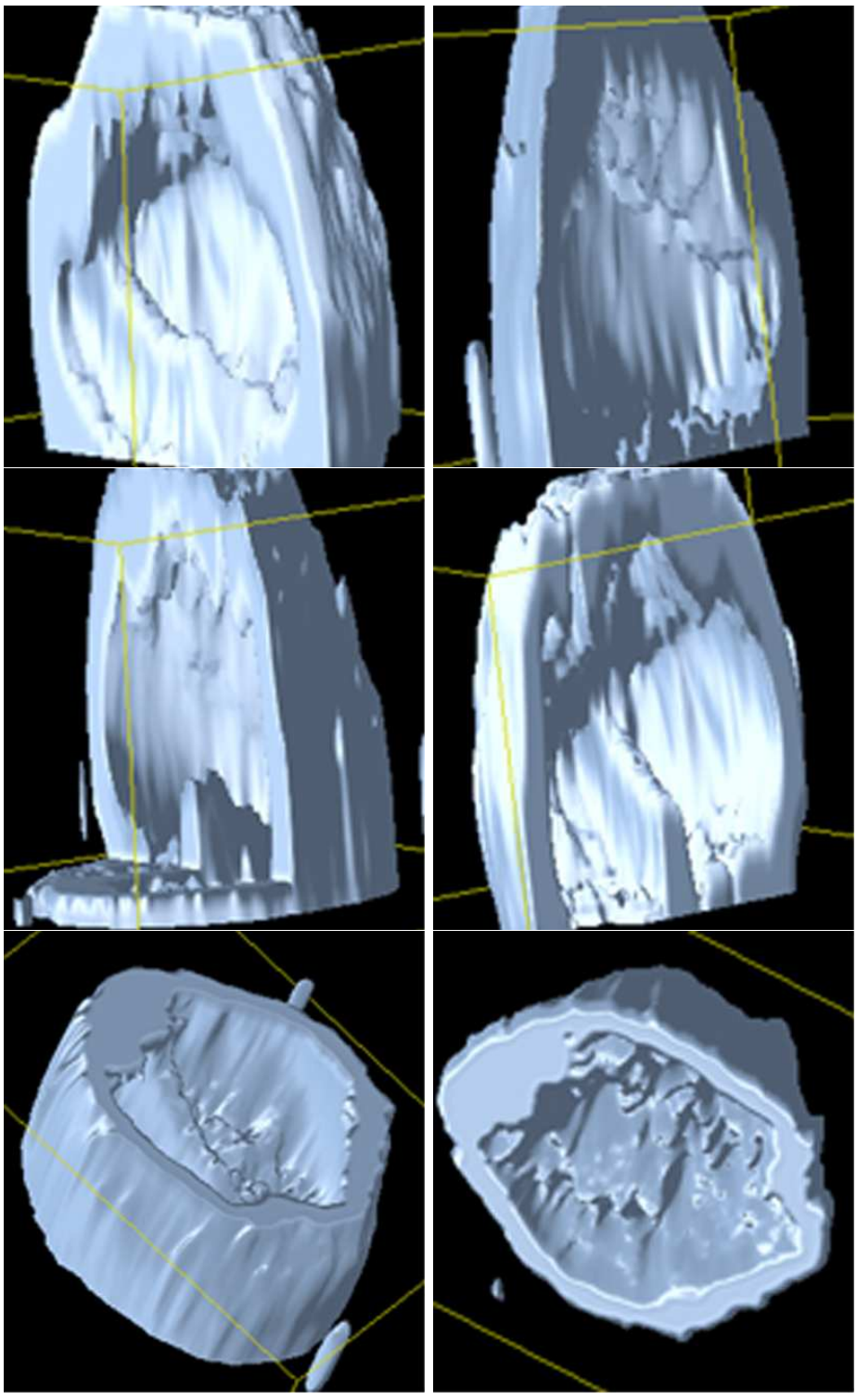

Abbildung 6.7: Blick in den Innenraum von Kern 2, (Queisser (SiT), 2005)

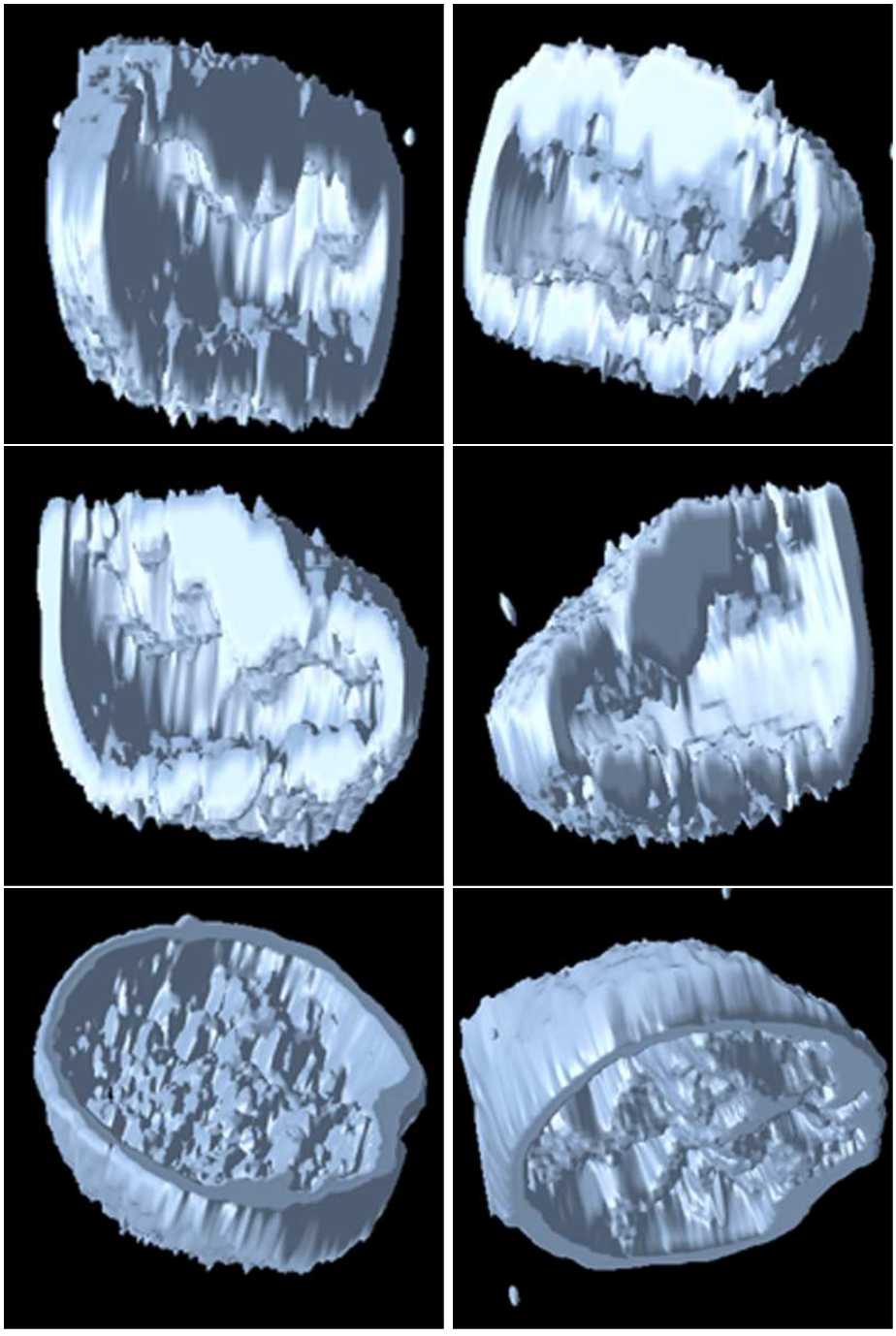

Abbildung 6.8: Blick in den Innenraum von Kern 3, (Queisser (SiT), 2005)

Kapitel 7

Vermessung des Zellkerns

Im letzten Schritt des Projekts wird der Zellkern vermessen.

Als Erstes soll dabei die Oberfläche des Zellkerns berechnet werden. Die Oberfläche des Zellkerns bildet – wie in Kapitel 6 bereits erwähnt – die *innere* Spur der Membran.
Wir benötigen also ein Verfahren, welches diese Oberfläche extrahiert. Ist dies möglich, kann einmal dessen Oberflächengröße bestimmt werden, zum anderen lässt sich die Oberfläche graphisch darstellen und wir haben damit eine Möglichkeit die Innenseite der Membran mit ihren Einstülpungen zu visualisieren. Damit ist auch das in Abschnitt 6.3 beschriebene Problem behoben.

Wir gehen folgendermaßen vor:
Sämtliche Punkte mit einem vorgegebenen Grauwert werden ermittelt und zu geschlossenen *Dreiecksgittern* verbunden. Durch die gleichmäßige Signalstreuung in der Mikroskopie besitzen äußere und innere Spur denselben Grauwert. In diesen ermittelten Daten ist also sowohl die äußere als auch die innere Spur der Zellkernmembran enthalten. Die erzeugten Dreiecksgitter müssen anschließend getrennt werden.
Das Dreiecksgitter des inneren Bereichs wird dann in das Software-Paket *ICEM CFD* eingelesen, welches aus dem Dreiecksgitter ein Oberflächengitter erzeugt.

Als Zweites soll das Volumen des Zellkerns berechnet werden. Aus dem im ersten Schritt gewonnenen Oberflächengitter kann mit *ICEM CFD* ein Volumengitter erzeugt werden, welches den Zellkern beschreibt. Das Volumen dieses Gitters wird zuletzt berechnet.
In diesem Kapitel werden alle Verfahren beschrieben, die zur Realisierung der oben aufgeführten notwendigen Teilschritte entwickelt bzw. verwendet wurden.

7.1 Berechnung der Membranoberfläche

Zur Erstellung des Oberflächengitters, welches den Zellkern beschreibt wurde ein von Alexander Heusel entwickeltes Programm zur *Erstellung* und *Separierung* von *Isoflächen* verwendet und im Rahmen dieses Projekts erweitert.
Um den Innenraum des Zellkerns zu extrahieren, ist zunächst eine Aufarbeitung der Mikroskopiedaten nötig, und zwar in den Schritten

1. Filterung der Daten.
2. Segmentierung der Daten.
3. Erneute Filterung.

Für den ersten Filterprozeß wird folgende Einstellung des Filters verwendet:

1. time_steps $= 4$
2. tau $= 2.0$
3. epsilon $= 10^{-3}$
4. levels $= 1$
5. integration_size $= 10^3$ Voxel
6. GeometryType $=$ CUBE
7. ip_flag $=$ IP_USUAL
8. anicoeffs: anicoeff1 $= 1$, anicoeff $= 1$, anicoeff $= 0$

Die Segmentierung verläuft nach der in Abschnitt 6.2.3 beschriebenen Otsu-Methode. Im zweiten Filterprozeß wird der Zeitschrittparameter von 4 auf 2 gesetzt. Die zweite Filterung ist nötig, um das nach der Segmentierung entstandene *binäre* Bild zu glätten und damit ein glattes Oberflächengitter zu erhalten, dessen Oberfläche der Zellkernoberfläche entspricht.

7.1.1 Das Programm iso_surf

Das Programm iso_surf legt einen Würfel an, dessen Eckpunkte den Mittelpunkten der Bildvoxel entsprechen, d.h. der Würfel ist in einem Würfel bestehend aus 8 Voxeln enthalten (siehe Abb. 7.1). Anschließend wird der Würfel in fünf Tetraeder gemäß Abbildung 7.2 zerlegt.

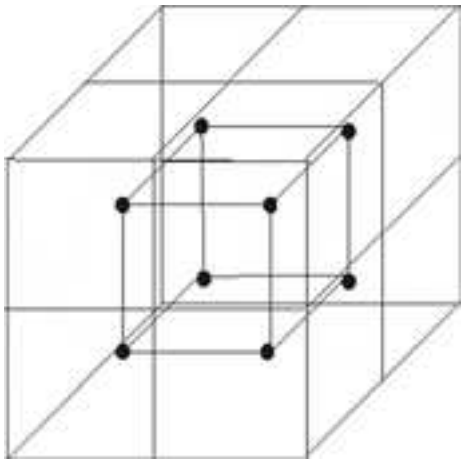

Abbildung 7.1: 8 Bildvoxel, in der Mitte Würfel mit Voxelmittelpunkte als Eckpunkte, (Queisser (SiT), 2005)

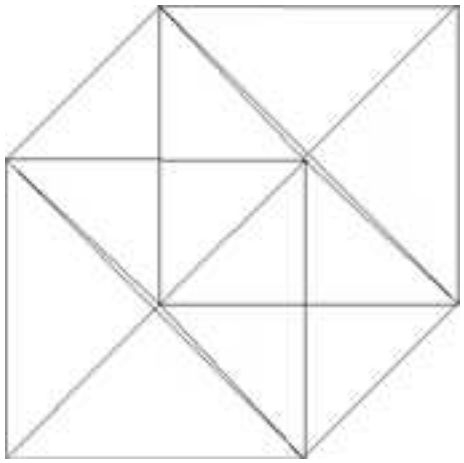

Abbildung 7.2: Würfel zerlegt in 5 Tetraeder, (Queisser (SiT), 2005)

```
112383 224404
0 51 60.95 14.95
1 51 61 14.9478
2 50.9508 60.9508 14.9508
3 50.9496 61 14.9496
4 50.7391 61 15
5 51 60.7857 15
6 50.8667 60.8667 15
7 52 60.872 14.872
8 52 61 14.8667
9 51.8769 60.8769 14.8769
```

```
112378 60 76.1318 41.1318
112379 59.0261 76.0261 41.0261
112380 61 76.0588 41.0588
112381 60.1318 76.1318 41.1318
112382 61.0598 76.0598 41.0598
0 0 1 2
1 3 1 2
2 3 4 2
3 0 5 2
4 6 5 2
5 6 4 2
```

Abbildung 7.3: links: Liste der Isoflächenpunkte, rechts: Liste der Dreieckspunkte, (Queisser (SiT), 2005)

Die Grauwerte der Tetraedereckpunkte sind aus der raw-Datei bekannt, die den *Mikroskopie-Stack* speichert. Zwischen diesen Grauwerten wird entlang der Tetraederlinien *linear* interpoliert und die Position des Punktes mit dem festgelegten Threshold-Grauwert bestimmt.
Die so ermittelten Punkte werden anschließend zu Dreiecken verbunden. Der Würfel tastet Schritt für Schritt den gesamten Stack ab und bestimmt die Positionen aller Punkte die auf dem Threshold liegen.

Die Gitterinformation wird anschließend in einer `txt`-Datei gespeichert, welche eine indizierte Liste aller Punktkoordinaten enthält, sowie eine indizierte Liste aller erzeugten Dreiecke. Abbildung 7.3 zeigt Auszüge aus einer solchen `txt`-Datei.

Das Programm arbeitet also folgende Schritte ab:

1. Erstellen eines Würfels mit Voxelmittelpunkten als Eckpunkte.
2. Zerlegung des Würfels in Tetraeder.
3. Interpolieren der Grauwerte entlang der Tetraederlinien.
4. Bestimmen der Position des Punktes mit dem festgelegten Threshold-Grauwert.
5. Verbinden der Punkte zu Dreiecken.
6. Erstellen einer `txt`-Datei mit Koordinaten- und Dreiecksinformation.

Modifikation:

Für die in diesem Projekt aufgenommenen Mikroskopiebilder ist für jeden Kern eine Modifikation des Programms iso_surf vorzunehmen.
Das Programm legt intern ein Koordinatensystem mit einer Voxeleinheit an. Hat ein Voxel also die Größe $0.12 \times 0.12 \times 0.16\ \mu m^3$ wird die Einheit des Koordinatensystems in x- und y-Richtung mit $0.12\ \mu m$ und in z-Richtung mit $0.16\ \mu m$ als Koordinateneinheit festgelegt.
In diesem Beispiel wäre das erzeugte Gitter also mit dem Faktor $\frac{3}{4}$ gestaucht. Aus der (bei jeder Mikroskopieaufnahme unterschiedlichen) Voxelgröße ist also der Streckungs- bzw. Stauchungsfaktor in z-Richtung zu berechnen und an entsprechender Stelle des Algorithmus einzufügen.

7.1.2 Das Programm iso_split

Das in Abschnitt 7.1.1 vorgestellte Programm iso_surf erzeugt Dreiecksgitter geschlossener Isoflächen. Die erstellte Datei trägt jedoch die gesamte Information aller Isoflächen.
Wir benötigen jedoch ausschließlich die Isofläche der inneren Spur der Membran. Das von Alexander Heusel geschriebene Programm iso_split liefert die Möglichkeit geschlossene Isoflächen aus der vom Programm iso_surf erstellten txt-Datei zu trennen und in separate Dateien zu exportieren.

Beachte:

Es ist wichtig, dass das Bild in der Aufarbeitungsphase so gefiltert und segmentiert wird, das die innere und äußere Spur der Membran in sich geschlossen sind, da diese sonst von iso_surf zu *einer* geschlossenen Isofläche verbunden wird und im Separierungsschritt nicht getrennt werden.

Das Programm iso_split liest die von iso_surf erstellte Datei ein und schreibt die Punkt- und Dreiecksinformation in Listen. Es wird ein Startpunkt gewählt, dieser in eine Bearbeitungsliste übertragen und aus der ursprünglichen Liste gelöscht. In dieser Liste werden nun alle Nachbarn dieses Punkts gesucht, ebenfalls übertragen und aus der Originalliste gelöscht. Wurden alle Nachbarn des Punkts ermittelt, wird dieser aus der Bearbeitungsliste gelöscht. Dieses Verfahren wird wiederholt bis keine Nachbarpunkte mehr gefunden werden. An dieser Stelle ist eine in sich geschlossene Isofläche gefunden worden. Die Isofläche wird in eine wie in Abbildung 7.3 gezeigte Datei exportiert.
So fährt der Algorithmus fort, bis die Originalliste leer ist. Dieses Verfahren nennt man auch *Marching-Front*-Verfahren, da dessen Bearbeitungsfront sich in alle Richtungen “Flächenbrandartig” ausbreitet.

Folgende Schritte werden ausgeführt:

1. Einlesen der von iso_surf erstellten `txt`-Datei.
2. Importieren der Punkt- und Dreiecksinformation in Listen.
3. Wahl eines Startpunkts und Bearbeitung der Punktliste nach einem Marching-Front-Verfahren.
4. In sich geschlossene Isoflächen werden in `txt`-Dateien geschrieben.

Um das Dreiecksgitter weiter zu bearbeiten, also zum einen in *ICEM CFD* einlesen und bearbeiten zu können, zum anderen mit dem Simulationstool *UG* kompatibel zu machen, benötigen wir anstatt der von iso_split erstellten `txt`-Datei ein `lgm`-Format.

7.1.3 Erstellen einer `lgm`-Ausgabe

Im Rahmen dieses Projekts wurde das Programm iso_split erweitert um die Gitterinformation als `lgm`-Datei auszugeben. Abbildung 7.4 zeigt die `lgm`-Datei eines Einheitswürfels, dessen Würfelseiten entlang ihrer Diagonalen in Dreiecke zerlegt wurden.

Wie man sieht sind folgende Informationen zu ermitteln:

1. **Domain-Info**: Information über die Geometrie.
2. **Line-Info**: Angabe aller im Gitter enthaltenen Linien. Eine Linie wird durch zwei *Punktindizes* beschrieben.
3. **Surface-Info**: Angabe der einzelnen Oberflächen des Gitters. In unserem Fall existiert lediglich eine Oberfläche. Dazu muss über *left* und *right* (siehe Abb. 7.4) Innen- und Aussenraum des Zellkerns definiert werden und sämtliche Punkt-, Linien- und Dreiecksinformation geschrieben werden.
4. **Point-Info**: Liste aller Punkte des Gitter mit ihren Raumkoordinaten.

Beachte:

Was aus Abbildung 7.4 nicht hervorgeht, ist:

1. Linien dürfen nur einmal in der *Line-Info* vorkommen.
2. Dreiecke in der *Surface-Info* besitzen eine Orientierung, d.h. die Reihenfolge der drei Punkte, die ein Dreieck beschreiben, spielt eine Rolle.

In einem *Post-processing*-Prozeß wird die in iso_split erstellte Punkt- und Dreiecksliste bearbeitet. Dabei werden die Dreiecke orientiert und die noch nicht vorhandene Linieninformation erzeugt.

```
# Domain-Info
name = sp3d
problemname = problem
convex = 1

# Unit-Info
unit 1 DUMMY

#Line-Info
line 0: points: 0 1;
line 1: points: 1 2;
line 2: points: 2 3;
line 3: points: 3 0;
line 4: points: 4 5;
line 5: points: 5 6;
line 6: points: 6 7;
line 7: points: 7 4;
line 8: points: 0 4;
line 9: points: 1 5;
line 10: points: 2 6;
line 11: points: 3 7;


#Surface-Info
surface 0: left=1; right=0; points: 0 1 2 3;
           lines: 0 1 2 3; triangles: 0 1 2; 0 2 3;
surface 1: left=1; right=0; points: 1 0 4 5;
           lines: 0 8 4 9; triangles: 0 4 5; 1 0 5;
surface 2: left=1; right=0; points: 1 2 5 6;
           lines: 1 10 5 9; triangles: 1 5 6; 1 6 2;
surface 3: left=1; right=0; points: 2 3 6 7;
           lines: 2 11 6 10; triangles: 2 6 7; 2 7 3;
surface 4: left=1; right=0; points: 0 3 4 7;
           lines: 3 8 7 11; triangles: 0 3 7; 0 7 4;
surface 5: left=1; right=0; points: 4 5 6 7;
           lines: 4 5 6 7; triangles: 4 7 6; 4 6 5;

#Point-Info
0 0 0;
1 0 0;
1 1 0;
0 1 0;
0 0 1;
1 0 1;
1 1 1;
0 1 1;
```

Abbildung 7.4: Beispiel einer `lgm`-Datei, (Queisser (SiT), 2005)

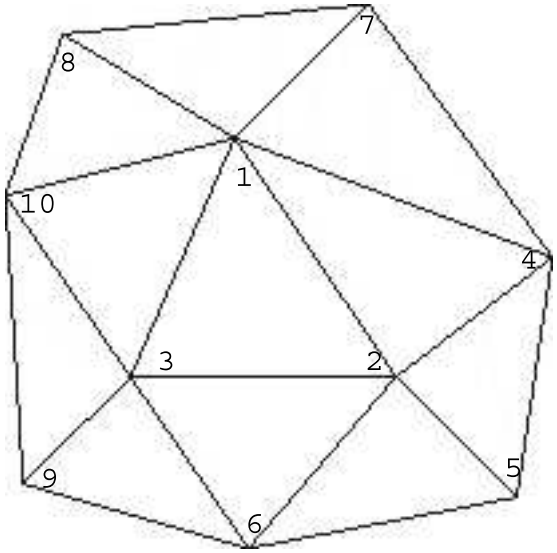

Abbildung 7.5: Auszug aus einem Dreiecksgitter, (Queisser (SiT), 2005)

Dreiecksorientierung:

Der erste Schritt im Post-processing ist die Orientierung der Dreiecke. Dazu werden diese in eine Liste triangles eingelesen. Eine weitere Liste sort nimmt das erste Element von triangles auf und sucht seine Nachbardreiecke um diese zu orientieren.
Die Idee zur Orientierung ist folgende:

> Jede Linie mit Punkten i und j des Gitters muss einmal von i nach j und von j nach i durchlaufen werden.

Das erste Element in sort ist das zu bearbeitende Dreieck und gibt vor wie die Linien der Nachbardreiecke durchlaufen werden müssen. Abbildung 7.5 zeigt einen Ausschnitt eines Dreiecksgitters. In diesem Beispiel betrachten wir als erstes das Dreieck Δ_{123}. Da die erste Linie von 1 nach 2 durchlaufen wird muss diese im Nachbardreieck Δ_{124} von 2 nach 1 durchlaufen werden. Δ_{124} wird also zu Δ_{214} umorientiert und gespeichert. So verfährt man mit allen Nachbardreiecken, aus Δ_{236} wird Δ_{326}, aus Δ_{3110} wird Δ_{1310} usw.
Wird ein Dreieck von triangles nach sort übernommen wird es in triangles gelöscht, da es nicht mehr bearbeitet werden darf, ist ein Dreieck in sort bearbeitet worden wird es auch aus dieser Liste gelöscht. Die Nachbardreiecke werden immer ans Ende von sort kopiert um analog zu Abschnitt 7.1.2 ein Marching-Front-Verfahren zu erhalten.

Der Orientierungsalgorithmus arbeitet also folgende Punkte ab:

1. Suchen der Nachbardreiecke des ersten Elements von sort und schreiben dieser in sort.
2. Orientieren der Nachbardreiecke und speichern der orientierten Dreiecke in einer separaten Liste.

Linienerzeugung:

Zur Linienerzeugung geht man ähnlich wie bei der Dreiecksorientierung vor. Mit Hilfe der Dreiecksliste triangles und einer line-Liste wird in jedem Schritt das erste Dreieck in triangles bearbeitet und anschließend gelöscht bis triangles leer ist.
In jedem Bearbeitungsschritt wird geprüft, ob die Linien des Dreiecks schon in der Liste lines vorhanden sind. Sind diese noch nicht vorhanden, werden diese Linien in lines aufgenommen, ansonsten verworfen.

Hinweis:

Falls man anstatt einer einzigen *Surface* das Oberflächengitter in mehrere Surfaces zerlegen möchte, steht noch ein Algorithmus in der Erweiterung von iso_split zur Verfügung, welcher den verschiedenen Oberflächen die zugehörigen Linien zuordnet. Dabei werden den Dreiecken die Indizes ihrer Linien zugeordnet.
Da die Filter und Programme, die in den folgenden Schritten noch verwendet werden müssen, für wenige Oberflächen mit vielen Dreiecken zeitlich optimiert sind, empfiehlt es sich, nicht jedes Dreieck als eine Surface zu speichern. Wir erstellen also im Folgenden immer eine Surface die alle Dreiecke, Linien und Punkte enthält.

Punktinformation:

Die Punktinformation wird als einfache Liste in die `lgm`-Datei geschrieben. Anders als bei der `txt`-Datei werden jedoch lediglich die Koordinaten der Punkte gespeichert, *ohne* deren Index.

7.1.4 Erzeugen eines *ICEM CFD*-Projekts

Um eine Geometrie in *ICEM CFD* einlesen zu können, muss die oben erzeugte `lgm`-Datei in ein `tin`-Format umgewandelt werden. Dazu steht ein von Andreas Hauser geschriebener lgm2tetin-Filter zur Verfügung.
Nun kann die tin-Datei in ein *ICEM CFD*-Projekt importiert werden. Das Programm erstellt ein Oberflächengitter dessen Oberfläche direkt berechnet wird. Die Ausgabe trägt die Einheit *Quadratpixel* und kann mit Hilfe der Voxelgröße in μm^2 umgerechnet werden.

Mit iso_surf, dem erweiterten Programm iso_split, dem lgm2tetin-Filter und *ICEM CFD* steht ein Verfahren, das die Oberfläche des Zellkerns berechnet.

7.2 Berechnung des Kernvolumens

Um Information über den Einfluss der Membraneinstülpungen zu bekommen, muss noch das Volumen des Zellkerns berechnet werden. Dazu müssen zunächst alle Schritte von Abschnitt 7.1 durchlaufen werden. Im Anschluss daran wird ein *Volumengitter* erzeugt.

7.2.1 Erzeugen eines Volumengitters

Mit *ICEM CFD* kann aus dem Oberflächengitter ein Volumengitter erzeugt werden. Die Feinheit des Volumengitters kann so gewählt werden, dass eine ausreichend genaue Annäherung an die Geometrie des Zellkerns erreicht wird. Bei dem Volumengitter handelt es sich um ein *Tetraedergitter*.
Die Volumengitterinformation wird von *ICEM CFD* in zwei Dateien gespeichert. Die eine enthält die Tetraederdaten, die andere die Koordinaten der Tetraedereckpunkte. Aus diesen Dateien muss nun das Volumen des Gitters berechnet werden.

7.2.2 Das Programm VolCal

Da *ICEM CFD* keine interne Routine zur Volumenberechnung besitzt, wurde hierfür ein Programm VolCal in der Programmiersprache *Perl* geschrieben.
VolCal arbeitet Schritt für Schritt die Tetraeder aus der ersten Datei ab. Es sucht dazu in der zweiten Datei die Punktkoordinaten und berechnet das Tetraedervolumen Vol_{Tet} nach

$$Vol_{Tet} = \frac{1}{12} \cdot \begin{vmatrix} x_0 - x_1 & y_0 - y_1 & z_0 - z_1 \\ x_0 - x_2 & y_0 - y_2 & z_0 - z_2 \\ x_0 - x_3 & y_0 - y_3 & z_0 - z_3 \end{vmatrix} \tag{7.1}$$

Dabei sind (x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2) und (x_3, y_3, z_3) die Eckpunkte der Tetraeder.
Die einzelnen Tetraedervolumen werden zum Gesamtvolumen des Tetraedergitters aufsummiert.

Es ist nun möglich, die Oberfläche und das Volumen eines Zellkerns zu berechnen. Wir sind jetzt in der Lage den Einfluss der Membraneinstülpungen auf Oberfläche und Volumen zu untersuchen.

Kapitel 8

Mess- und Visualisierungsergebnisse

In Kapitel 7 wurde ein Verfahren entwickelt das es ermöglicht Zellkerne zu vermessen. Eines der Ziele dieses Projekt war es, die schon in der Einleitung erwähnten *Membraneinstülpungen* zu untersuchen.
Die Frage die es zu klären gilt ist Folgende:

> Wird durch die Einstülpungen die Membranoberfläche größer oder "verdrängen" die Einstülpungen Kernvolumen?

Um auf diese Frage eingehen zu können müssen zunächst eine Reihe von Zellkernen mit Einstülpungen nach der Oberfläche und dem Volumen vermessen werden.
Um diese Werte zu vergleichen muss noch eine Referenz gesetzt werden. Dazu wurde eine zweite Reihe von Zellkernen vermessen, diesmal jedoch Zellkerne *ohne* Einstülpungen. Ein Vergleich der Oberflächen- und Volumenwerte liefert Auskunft über die Membraneinstülpungen.

8.1 Messergebnisse

Die folgenden Messergebnisse basieren auf einer Testreihe, bei der 10 Zellkerne mit Einstülpungen und 10 ohne Einstülpungen vermessen wurden.

8.1.1 Zellkernoberfläche

Tabelle 8.1 zeigt die Ergebnisse der Oberflächen-Messreihe der Versuchsgruppe, den Zellkernen mit Einstülpungen. Die Oberflächen der Kontrollgruppe sind in Tabelle 8.2 dargestellt.

Name	Art	Oberfläche	Pixelgröße
stack_c3	Infoldings	164,04	0,0153 μm^2
stack_c4	Infoldings	131,32	0,0153 μm^2
stack_c5	Infoldings	162,13	0,0153 μm^2
stack_c7	Infoldings	100,76	0,0226 μm^2
stack_c8	Infoldings	109,43	0,0226 μm^2
stack_c9	Infoldings	187,17	0,0130 μm^2
stack_i-1	Infoldings	209,72	0,0347 μm^2
stack_i-4	Infoldings	210,42	0,0347 μm^2
stack_i-5	Infoldings	205,71	0,0347 μm^2
stack_inf1	Infoldings	185,78	0,1405 μm^2

Tabelle 8.1: Oberflächen von Kernen mit Einstülpungen

Name	Art	Oberfläche	Pixelgröße
stack_con1	Control	135,11	0,0199 μm^2
stack_con3	Control	150,48	0,0216 μm^2
stack_con4	Control	114,73	0,0216 μm^2
stack_con5	Control	126,40	0,0111 μm^2
stack_con6	Control	121,64	0,0111 μm^2
stack_con8	Control	114,34	0,0136 μm^2
stack_con10	Control	125,71	0,0166 μm^2
stack_e-1	Control	171,51	0,0335 μm^2
stack_e-2	Control	148,71	0,0347 μm^2
stack_e-3	Control	150,35	0,0347 μm^2

Tabelle 8.2: Oberflächen von Kontroll-Kernen

Wichtig: Je nach Originalgröße der Kerne wird für jeden Zellkern ein individueller Zoomfaktor zur Vergrößerung verwendet. Dieser Zoomfaktor ändert die Voxelgröße und muss deshalb in die Oberflächen- und Volumenwerte integriert werden.

8.1.2 Zellkernvolumen

Nach dem Verfahren aus Abschnitt 7.2 wurde von allen untersuchten Kernen das Volumen berechnet. Tabellen 8.3 und 8.4 stellen die Ergebnisse zusammen. Dabei ist die zugehörige Voxelgröße und der entsprechende Zoomfaktor integriert.

Name	**Art**	**Volumen**	**Voxelgröße**	**Zoomfaktor**
stack_c3	Infoldings	147,06	$2{,}48 \cdot 10^{-3}\ \mu m^3$	15,06
stack_c4	Infoldings	106,51	$2{,}48 \cdot 10^{-3}\ \mu m^3$	15,06
stack_c5	Infoldings	117,11	$2{,}48 \cdot 10^{-3}\ \mu m^3$	15,06
stack_c7	Infoldings	83,66	$3{,}68 \cdot 10^{-3}\ \mu m^3$	12,37
stack_c8	Infoldings	74,18	$3{,}68 \cdot 10^{-3}\ \mu m^3$	12,37
stack_c9	Infoldings	155,37	$2{,}12 \cdot 10^{-3}\ \mu m^3$	16,32
stack_i-1	Infoldings	165,35	$5{,}65 \cdot 10^{-3}\ \mu m^3$	9,99
stack_i-4	Infoldings	138,84	$5{,}65 \cdot 10^{-3}\ \mu m^3$	9,99
stack_i-5	Infoldings	135,13	$5{,}65 \cdot 10^{-3}\ \mu m^3$	9,99
stack_inf1	Infoldings	168,07	$2{,}29 \cdot 10^{-3}\ \mu m^3$	15,69

Tabelle 8.3: Volumen der Versuchsgruppe

Name	**Art**	**Volumen**	**Voxelgröße**	**Zoomfaktor**
stack_con1	Control	140,54	$3{,}25 \cdot 10^{-3}\ \mu m^3$	13,17
stack_con3	Control	148,22	$3{,}52 \cdot 10^{-3}\ \mu m^3$	12,64
stack_con4	Control	99,31	$3{,}52 \cdot 10^{-3}\ \mu m^3$	12,64
stack_con5	Control	150,12	$1{,}81 \cdot 10^{-3}\ \mu m^3$	17,66
stack_con6	Control	135,29	$1{,}81 \cdot 10^{-3}\ \mu m^3$	17,66
stack_con8	Control	133,66	$2{,}22 \cdot 10^{-3}\ \mu m^3$	15,94
stack_con10	Control	122,87	$2{,}71 \cdot 10^{-3}\ \mu m^3$	14,42
stack_e-1	Control	117,68	$5{,}45 \cdot 10^{-3}\ \mu m^3$	10,16
stack_e-2	Control	141,39	$5{,}65 \cdot 10^{-3}\ \mu m^3$	9,99
stack_e-3	Control	144,71	$5{,}65 \cdot 10^{-3}\ \mu m^3$	9,99

Tabelle 8.4: Volumen der Kontrollgruppe

Name	q_i
stack_c3	0,90
stack_c4	0,81
stack_c5	0,72
stack_c7	0,83
stack_c8	0,68
stack_c9	0,83
stack_i-1	0,79
stack_i-4	0,66
stack_i-5	0,66
stack_inf1	0,90

Tabelle 8.5: Volumen/Oberflächen-Verhältnisse der *Infolding*-Kerne

8.1.3 Das Volumen/Oberflächen-Verhältnis

Um eine qualitative Aussage über den Einfluss der Membraneinstülpungen auf Oberfläche und Volumen des Zellkerns zu erhalten, berechnen wir das *Volumen/Oberflächen*-Verhältnis. Dazu führen wir folgende Größen ein:

$$
\begin{aligned}
q_i &:= \frac{\text{Volumen des } \textit{Infolding}\text{-Kerns}}{\text{Oberfläche des } \textit{Infolding}\text{-Kerns}} \\
q_c &:= \frac{\text{Volumen des } \textit{Control}\text{-Kerns}}{\text{Oberfläche des } \textit{Control}\text{-Kerns}} \\
r_i &:= \text{Arithmetisches Mittel der } q_i \\
r_c &:= \text{Arithmetisches Mittel der } q_c
\end{aligned}
$$

Tabellen 8.5 und 8.6 listen die untersuchten Kerne mit den zugehörigen q_i bzw. q_c auf.

Aus den Ergebnissen von Tabelle 8.5 und 8.6 ergibt sich:

$$
\begin{aligned}
r_i &= 0,78 \\
r_c &= 0,99
\end{aligned}
$$

Die Durchschnittsoberfläche der Versuchsgruppe liegt bei 166,65, die der Kontrollgruppe bei 138,90. Das Durchschnittsvolumen der Versuchsgruppe ist 129,13, das der Kontrollgruppe 133,38.

Name	q_c
stack_con1	1,04
stack_con3	0,98
stack_con4	0,86
stack_con5	1,19
stack_con6	1,11
stack_con8	1,17
stack_con10	0,98
stack_e-1	0,69
stack_e-2	0,95
stack_e-3	0,96

Tabelle 8.6: Volumen/Oberflächen-Verhältnisse der *Control*-Kerne

8.2 Visualisierungsergebnisse

Zur Darstellung der Zellkerne wurde das Visualisierungstool *VTK* verwendet. *VTK* visualisiert das in Kapitel 7 erzeugte Oberflächengitter. Dieser Abschnitt dient als Zusammenstellung einiger 3D-Rekonstruktionsergebnisse von Neuronenzellkernen. Diese sind in Abbildungen 8.1 bis 8.7 gezeigt.

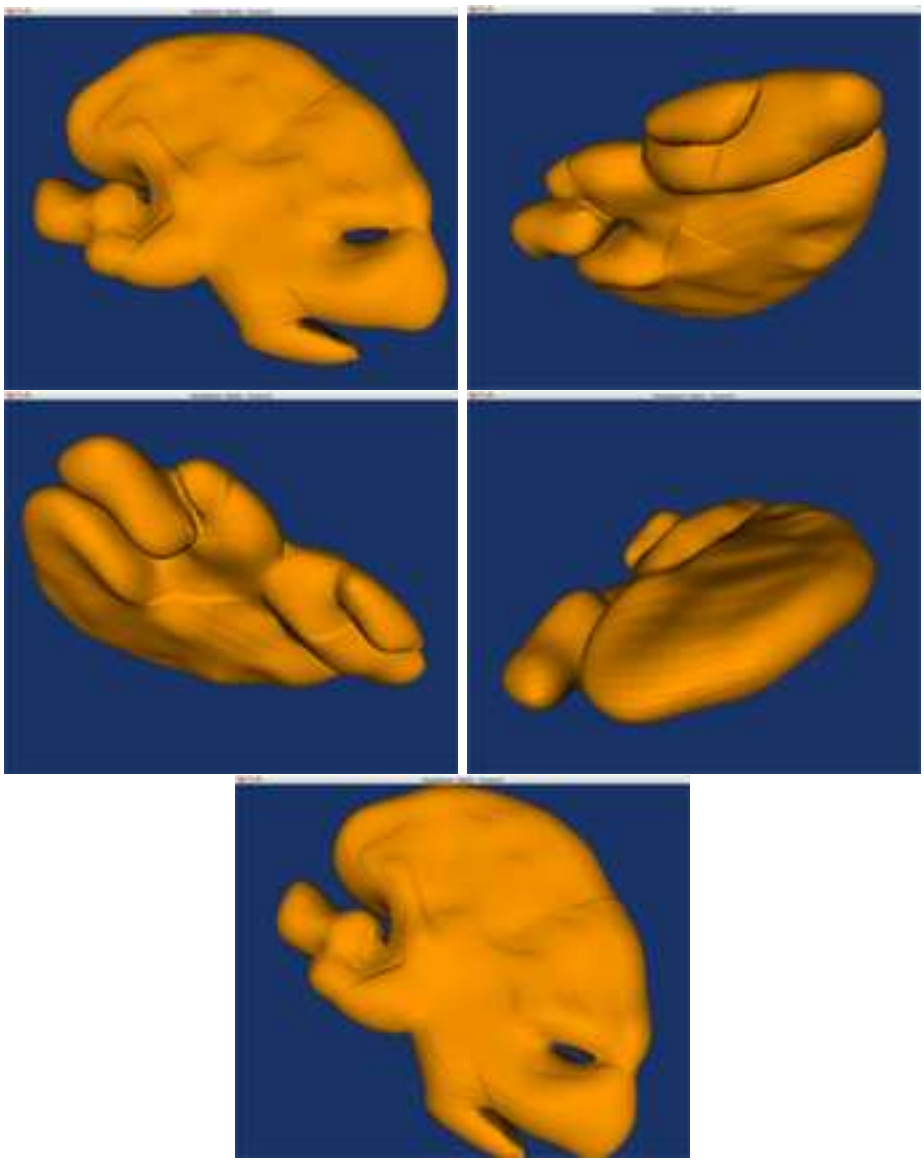

Abbildung 8.1: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

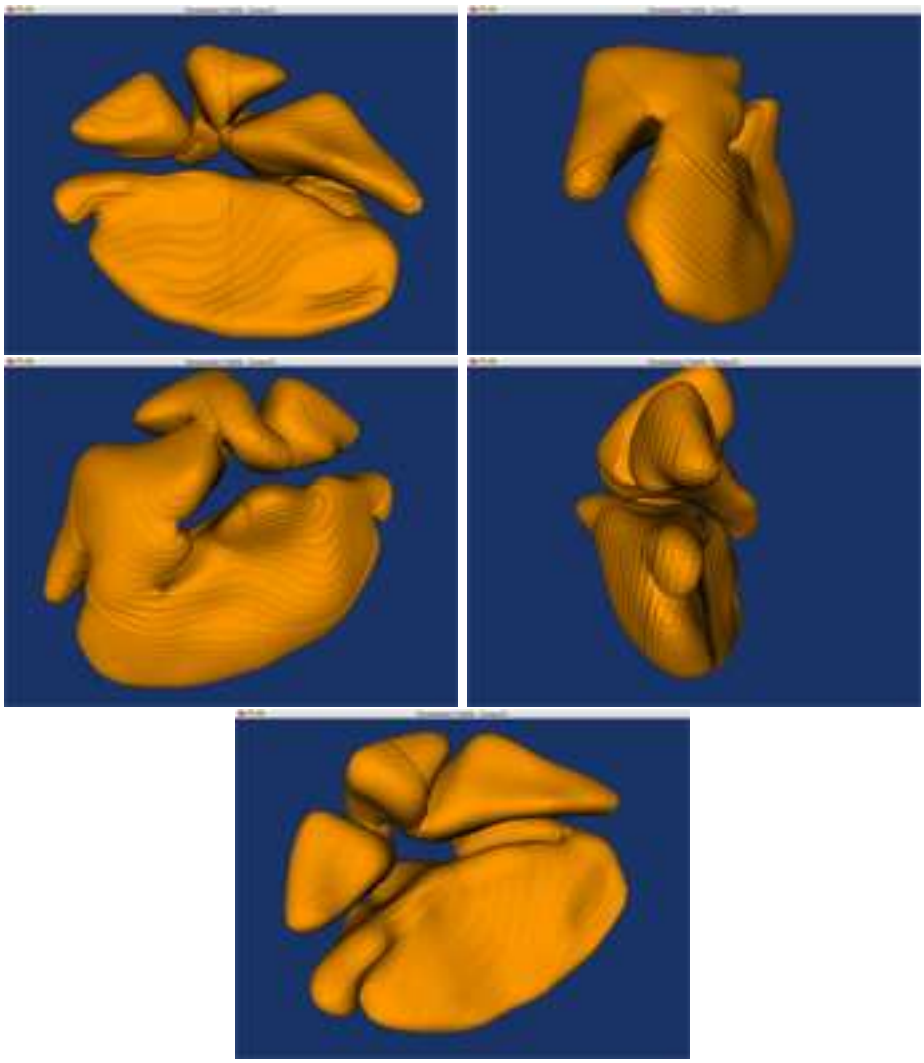

Abbildung 8.2: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

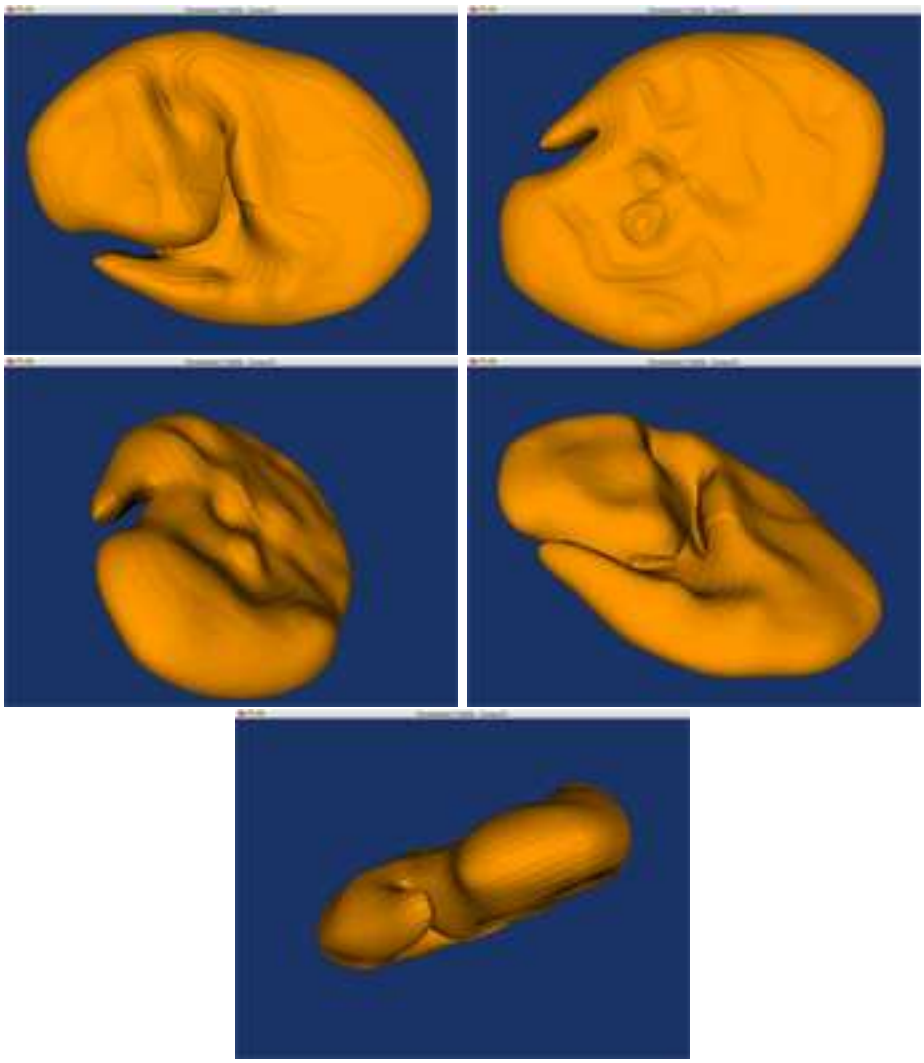

Abbildung 8.3: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

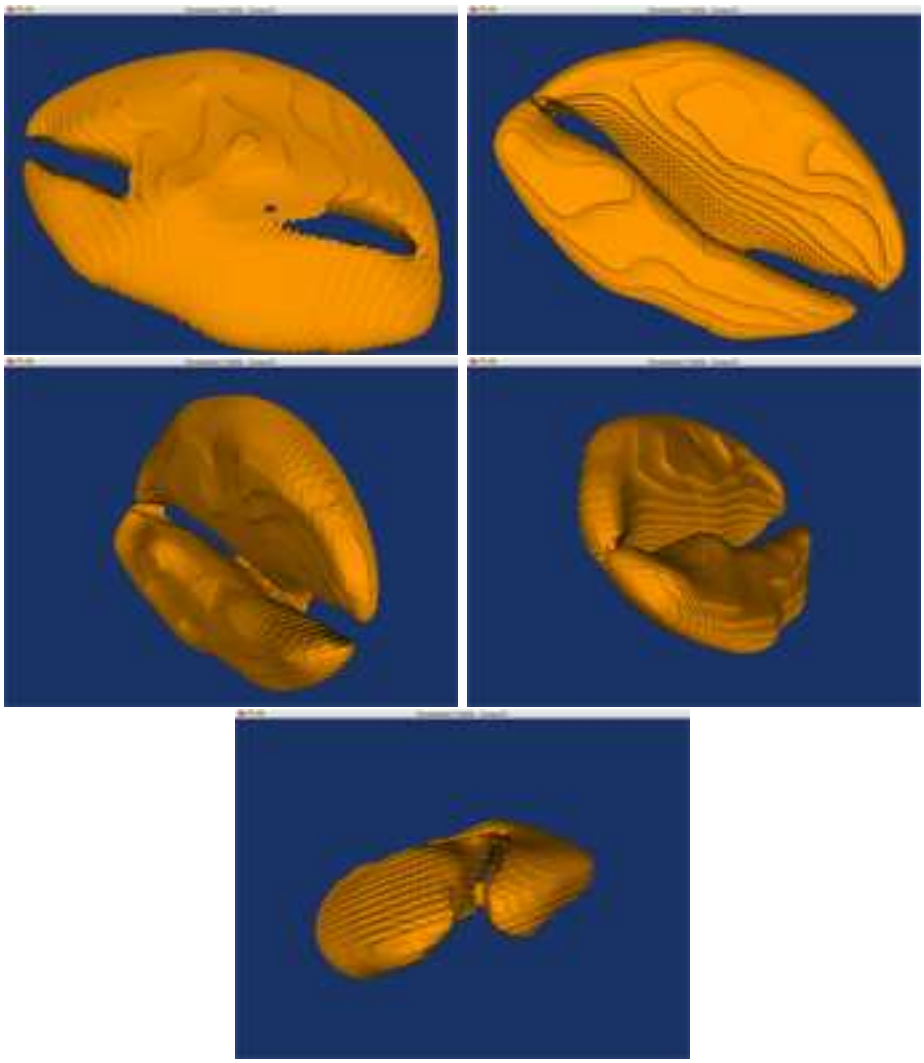

Abbildung 8.4: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

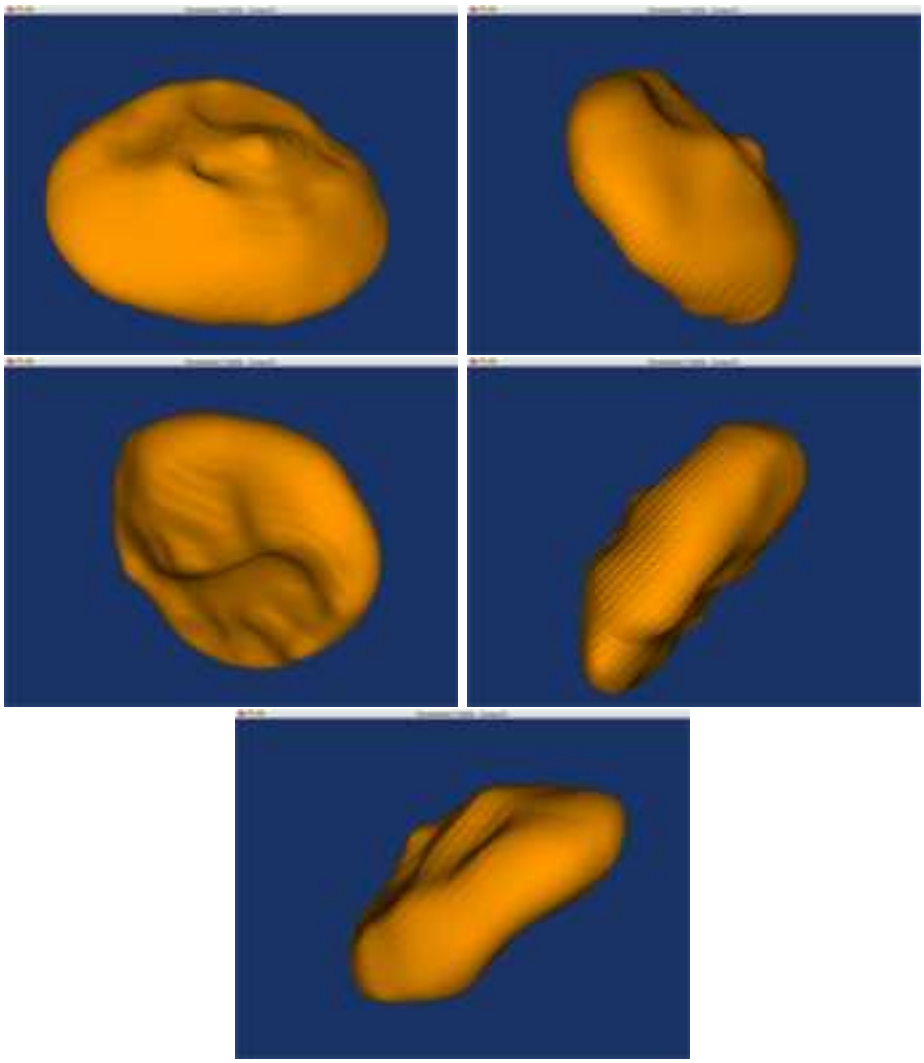

Abbildung 8.5: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

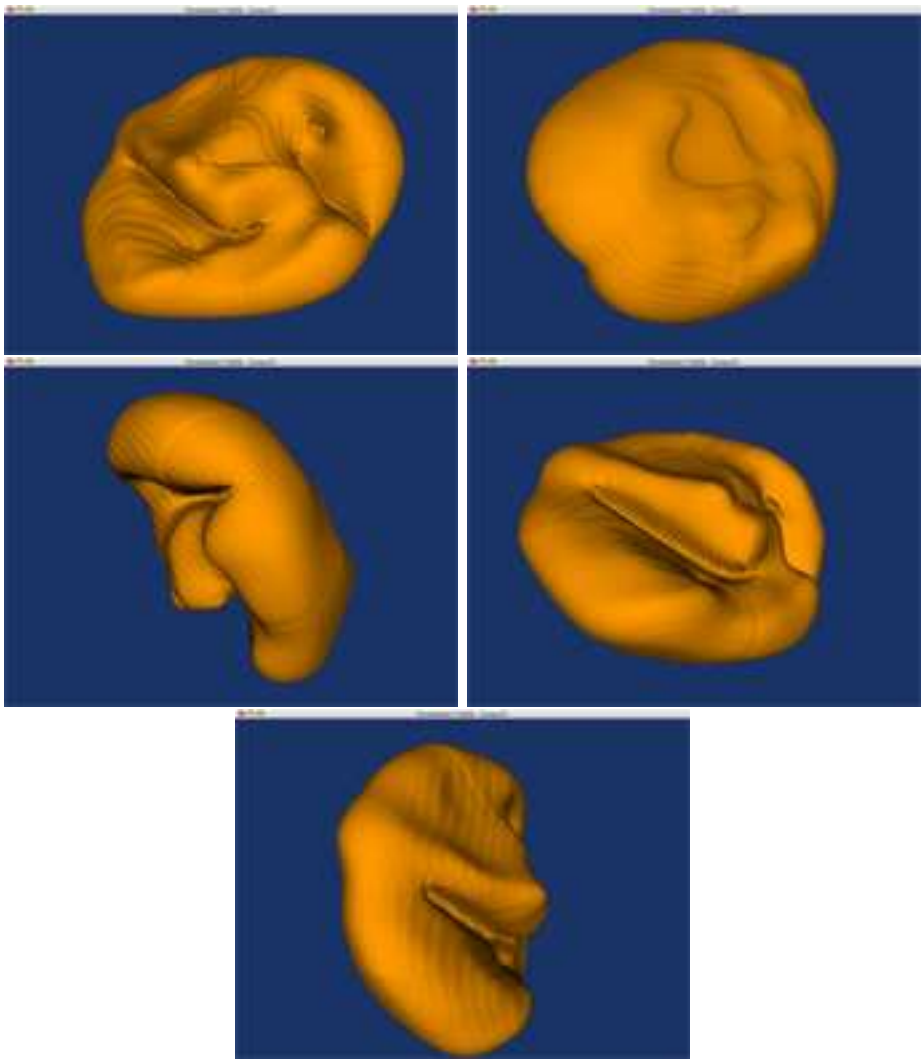

Abbildung 8.6: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

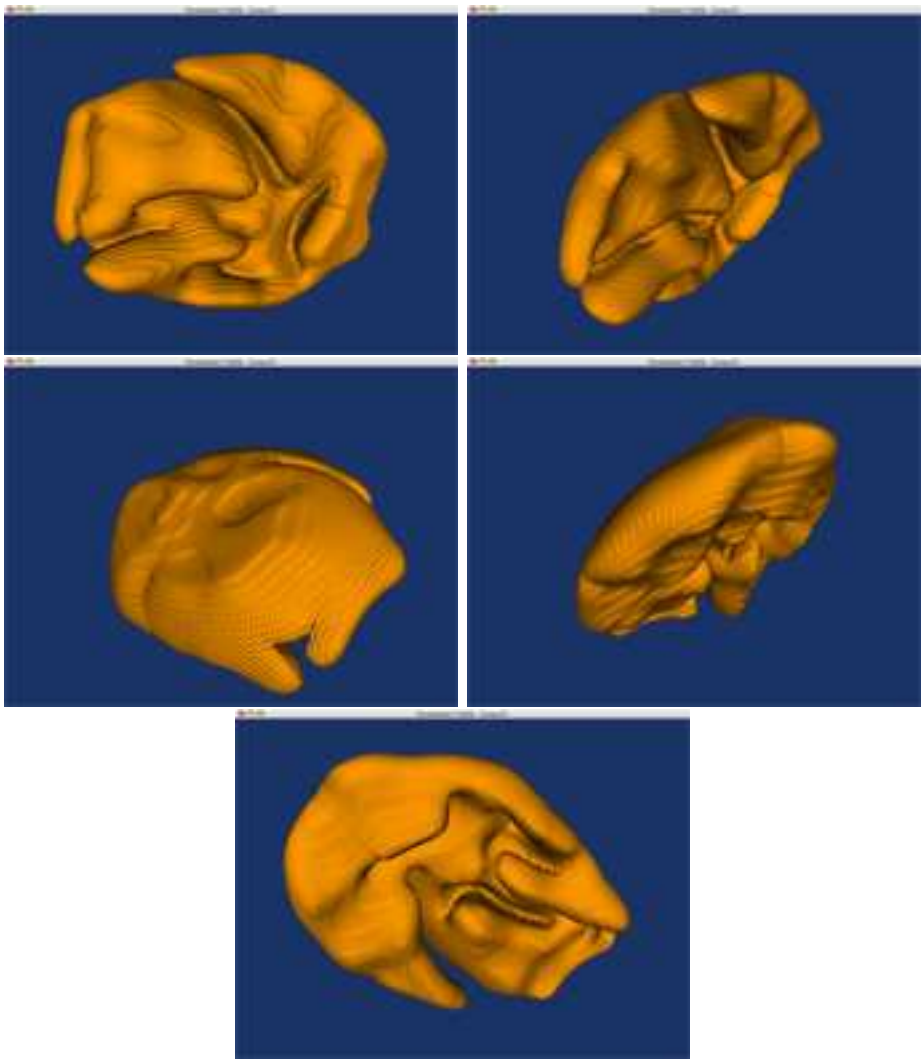

Abbildung 8.7: 3D-Rekonstruktion eines Zellkerns, (Queisser (SiT), 2005)

8.2.1 Resultate und Schlussfolgerungen

Wir wollen abschließend noch einmal auf die Motivationen dieses Projekts zurückkommen. Mikroskopische Untersuchungen von Neuronen-Zellkernen lassen generell lediglich eine zweidimensionale Betrachtung zu.
Somit war es schwierig, durch den gesamten Kern verlaufende Membranstrukturen richtig zu interpretieren. Es war also ein Anliegen, eine dreidimensionale Darstellungsmöglichkeit von Zellkernen zu entwickeln.
Dies ist in dieser Arbeit geschehen. Aufgrund der dreidimensionalen Visualisierung (siehe Abschnitt 8.2) lassen sich die Invaginationen darstellen und eindeutig interpretieren.
In allen hier untersuchten Kernen lassen sich ausschließlich Invaginationen in Form von "Einstülpungen" feststellen. Tubuläre Invagination wie es in Fricker[5] vermutet wird, lassen sich nicht finden.
Diese Einstülpung lassen sich möglicherweise weiter kategorisieren. Abbildung 8.2 zeigt einen Kern mit tief in den Kern reichende Einstülpungen, wohingegen Abbildung 8.4 nur *eine* kurze Einstülpung zeigt.
Die Kategorisierung war jedoch kein Ziel dieses Projekts und ist eine Aufgabe für die Neurobiologie.

Zwar hat sich durch die dreidimensionale Visualisierung die Art der Invaginationen der Zellkernmembran geklärt, es bleiben jedoch noch offene Fragen.
An dieser Stelle sind verschiedene Entstehungsmöglichkeiten der Einstülpung denkbar.
Ausgehend von einem Kern ohne Einstülpungen, könnten diese beispielsweise durch Bildung neuer Membran, die sich in den Kern wölbt, entstehen.
Oder die Membran des Kerns wölbt sich ohne Neubildung ins Innere und verdrängt dabei Volumen aus dem Kern.
Dazu wurden die Kerne weiter untersucht, auf die Größe ihrer Membranoberfläche und ihres Volumens. Die Messreihen in Abschnitt 8.1 zeigen die Ergebnisse dieser Untersuchung. Die dort eingeführten Größen r_i und r_c liefern den Durchschnitt der Volumen/Oberflächen-Verhältnisse der Versuchsgruppe ($r_i = 0,78$) und der Kontrollgruppe ($r_c = 0,99$).
Durch die Einstülpungen lässt sich also ein markanter Unterschied im Volumen/Oberflächen-Verhältnis erkennen, der kleinere Wert r_i entsteht scheinbar durch eine größere Durchschnittsoberfläche.
Die größere Kernmembran der Versuchsgruppe bietet eine größere Diffusionsoberfläche für Kalziumionen, was den Diffusionsprozeß beeinflusst.
Die Diffusion kann durch eine erhöhte Anzahl von Kernporen vermutlich beschleunigt werden.
Ein Vergleich der Durchschnittsoberflächen von Versuchs- und Kontrollgruppe zeigt eine um 20 % größere Zellkernoberfläche.
Ein weiterer Punkt, der schon in der Einleitung angesprochen wurde, ist

das Thema Kalziumdiffusion und definiert den Ausblick dieser Arbeit.
Dieses Projekt liefert zusätzlich den Ansatz zur mathematischen Simulation der Diffusion von Kalziumionen über die Zellkernmembran. Die entwickelten Methoden liefern Oberflächen- und Volumengitter auf denen weiterführend die Diffusion mathematisch simuliert werden kann.
Die Simulation dürfte der Antwort auf die Frage

> Was bewirken die Einstülpungen der Zellkernmembran aus neurobiologischer Sicht, speziell wie ändert sich der Diffusionsprozeß aufgrund der Membraneinstülpungen?

näher kommen.
Die Tatsache, dass das Volumen/Oberflächen-Verhältnis von 0,99 auf 0,78 abnimmt und die Kernoberfläche um 20 % größer ist als bei der Kontrollgruppe zeigt, dass sich die Eigenschaften der Kalziumdiffusion durch die Membraneinstülpungen ändern werden.

Mit diesem Projekt konnten wichtige und aktuelle neurobiologische Fragen in Hinblick auf die Morphologie der Neuronen-Zellkerne beantwortet werden. Es schaffte gleichzeitig die Grundlage zur Bearbeitung eines weiteren neurobiologischen Forschungsbereichs, der Kalziumdiffusion.

Danksagung

Ich möchte mich an dieser Stelle bei einer Reihe von Leuten bedanken, die aktiv oder passiv zum Gelingen dieser Arbeit beigetragen haben.

Ich bedanke mich bei Gabriel Wittum, der mich während der Zeit als Hilfswissenschaftler in seiner Abteilung und schließlich als Diplomant konstant unterstützte und durch positive Motivation den Spaß am Projekt erhielt. Es war zu jeder Zeit möglich, Ideen für das Projekt sowie zukunftsorientierte Themen zu diskutieren und umzusetzen.

Für eine sehr gute Zusammenarbeit und für die vielen Mikroskopiedaten danke ich Hilmar Bading und Malte Wittmann.

Ich danke Christine für die fachliche Unterstützung, das angenehme Arbeitsklima und entspannende Kaffeepausen.

Meinen Eltern, die mir eine wundervolle Studienzeit möglich machten und mir in jeder Studiumsphase beiseite standen gebührt besonderer Dank.

Weiterhin danke ich meinen Geschwistern, Leah und Jamie, die mich dabei unterstützten nicht allzu schnell erwachsen zu werden.

Auch meiner Freundin Anna möchte ich danken; für Motivation, liebevolle Unterstützung und Verständnis für gelegtliche “Vernachlässigungen” aufgrund der Diplomarbeit.

Ich danke auch meinen Mitbewohnern, die für willkommene Abwechslung in meinem Studium sorgten.

Literaturverzeichnis

[1] Jaakko Astola. *Fundamentals of nonlinear Digital Filtering*. CRC Press LLC, 1997.

[2] Hilmar Bading. Transcription-dependent neuronal plasticity. *Eur. J. Biochem.*, 267:5280–5283, 2000.

[3] C. Perez-Terzic et al. D.e. conformational states of the nuclear pore complex induced by depletion of nuclear calcium stores. *Science*, 273:1875–1877, 1996.

[4] Giles E. Hardingham et al. Distinct funtions of nuclear and cytoplasmic calcium in the control of gene expression. *Nature*, Seiten 360–365, 1996.

[5] Marc Fricker et al. Interphase nuclei of many mammalian cell types contain deep, dynamic, tubular membrane-bound invaginations of the nuclear envelope. *The Journal of Cell Biology*, 136(3):531–544, February 10 1997.

[6] Sangeeta Chawla et al. Cbp: A signal-regulated transcriptional coactivator controlled by nuclear calcium and cam kinase iv. *Science*, 281:1505–1509, September 4 1998.

[7] Wihelma Echevarria et al. Regulations of calcium signals in the nucleus by a nucleoplasmic reticulum. *Nature Cell Biology*, 5:440–446, May 2003.

[8] Otto Forster. *Analysis 2*. Vieweg, fünfte Auflage, 1999.

[9] Otto Forster. *Analysis 3*. Vieweg, dritte Auflage, 1999.

[10] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. Teubner Stuttgart, 1993.

[11] G. et al. Hardingham. Dinstinct functions of nuclear and cytoplasmic calcium in the control of gene expression. *Nature*, 385:260–265, 1997.

[12] R.W. Tsien K. Deisseroth, E.K. Heist. Calmodulin translocation to the nucleus supports creb phosphorylation in hippocampal neurons. *Nature*, 392:198–202, 1998.

[13] Clapham L. Stehno-Bittel, C. Perez-Terzic. D. e. diffusion across the nuclear envelope inhibited by depletion of the nuclear calcium store. *Science*, 270:1835–1838, 1995.

[14] Harvey Lodish. *Molecular Cell Biology.* W. H. Freeman and Company, vierte Auflage, 1999.

[15] W. Nolting. *Grundkurs: Theoretische Physik, Klassische Mechanik, Band 1.* Zimmermann-Neufang, dritte Auflage, 1989.

[16] Schmitt. *Physiologie des Menschen.* Springer, 27. Auflage, 1997.

[17] Roland Schulte. *Filterung von 3D-Neuronenaufnahmen durch nichtlineare anisotrope Diffusion.* Diplomarbeit, Universität Heidelberg, 2003.

[18] H.R. Schwarz. *Numerische Mathematik.* B.G. Teubner Stuttgart, vierte Auflage, 1997.

[19] Joachim Weickert. *Anisotropic Diffusion in Image Processing.* B.G. Teubner Stuttgart, 1998.

[20] Rainer Weissauer. *Lineare Algebra.* 2001. Vorlesungsskript, Universität Heidelberg.

[21] Gabriel Wittum. *Einführung in die Numerische Mathematik.* 1998. Vorlesungsskript, Universität Heidelberg.

[22] Gabriel Wittum. *Diskretisierungsverfahren für partielle Differentialgleichungen.* 1999. Vorlesungsskript, Universität Heidelberg.

[23] Gabriel Wittum. *Mehrgitterverfahren.* 1999. Vorlesungsskript, Universität Heidelberg.