UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Berechnung und Visualisierung von DTI-Daten auf Basis der GPU

Studienarbeit

im Studiengang Computervisualistik

vorgelegt von

Jens Eraßmy

Betreuer: Dr. Matthias Raspe
(Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Januar 2010

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

1 Einleitung

Die Diffusions-Tensor-Bildgebung (DTI: diffusion tensor imaging) ist eine Technik aus der Magnetresonanztomografie und basiert auf der Bewegung von Wassermolekülen in organischem Gewebe. Speziell im inhomogenen Gehirngewebe ist die Beweglichkeit durch die faserartige Struktur der weißen Gehirnsubstanz stark eingeschränkt. Anhand der Messergebnisse der DTI und darauf basierenden Berechnungen lässt sich die Faserstruktur geometrisch rekonstruieren. Hierzu existieren bestimmte Verfahren und Visualisierungsmethoden, von denen einige im Rahmen dieser Arbeit erläutert und implementiert werden.
Aufgrund der rasanten Weiterentwicklung der GPU-Hardware wurde Wert darauf gelegt, möglichst viele Arbeitsschritte auf der Grafikhardware durchzuführen. Dies umfasst insbesondere den Einsatz moderner Technologien, vor allem den Geometryshader, welcher nach Möglichkeit für die Berechnung und Visualisierung eingesetzt wurde.
Die entwickeltete Anwendung basiert auf dem GPU-basierten Framework CASACADA [Raspe2009], welches zur Verarbeitung und Visualisierung medizinischer Volumendaten entwickelt wurde.
Im Folgenden werden zuerst die technischen Grundlagen der DTI beschrieben. Darauf folgt ein Überblick über die gängigen Visualisierungsformen. Auf dieser Basis wird die Zielsetzung der Studienarbeit spezifiziert und letzte programmspezifische Grundlagen erläutert. Abschließend folgt die genaue Beschreibung der Implementierung der Applikation, sowie ein abschließendes Resümee.

2 Grundlagen

Zunächst wird ein Überblick über die grundlegende Entwicklung der Diffusions-Tensor-Bildgebung gegeben. Zum allgemeinen Verständnis wird bis zur Theorie der Kernspinresonanz zurückgegangen. Über die Techniken der MRT wird schließlich die Diffusions-Tensor-Bildgebung erfasst.

2.1 Kernspinresonanz

Magnetische Kernspinresonanz (auch NMR: nuclear magnetic resonance) beruht auf den magnetischen Eigenschaften der Atomkerne. Sie entsteht, wenn bestimmte Atomkerne, mit einem von Null verschiedenen Gesamtspin, innerhalb eines homogenen Magnetfeldes plaziert werden. Wenn die Anzahl der Protonen und Neutronen des Atomkerns unsausgewogen ist, existiert ein sogenannter Kernspin. Der Spin oder Eigendrehimpuls stellt eine Art Rotation um die eigene Achse dar (Präzessionsbewegung). Aus

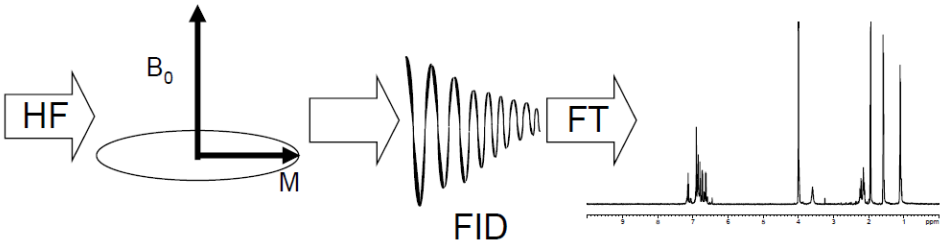

Abbildung 1: Nach Einstrahlung des HF-Impulses wird bei der Relaxation ein abklingendes Signal registriert und ein Spektrum berechnet.

dem Spin der Atomkerne resultiert ein magnetisches Moment. Normalerweise zeigt dieses lokale Magnetfeld am Kern in eine willkürliche Richtung. In einem äußeren statischen Feld richten sich die Spins jedoch parallel oder antiparallel zu diesem aus. Die Spinachse des Kerns präzediert dann mit einer bestimmten Frequenz. Sie wird als Lamour-Frequenz bezeichnet, die sich je nach Stärke des äußeren Feldes zu diesem proportional verändert. Zwischen den beiden möglichen Ausrichtungen des Kerns besteht eine Energiedifferenz. Diese kann durch Energieaufnahme beziehungsweise -abgabe überwunden werden. Hier bewirkt die Einstrahlung eines elektromagnetischen Wechselfeldes mit einem bestimmten Hochfrequenz-Impuls (HF-Impuls) ein Umklappen des Kernspins um 90° aus der Longitudinalmagnetisierung (z-Achse). Die Spinachse präzediert jetzt in der xy-Ebene und die z-Magnetisierung nimmt den Wert Null an. Nach dem Impuls-Signal bewegt sich die Spinachse, in Abhängigkeit von einer gewebespezifischen Zeitkonstanten, wieder in die Ausgangsrichtung des anliegenden Magnetfeldes. Dieser Vorgang beschreibt einerseits das Abklingen der xy-Magnetisierung (T2-/ Spin-Spin-Relaxation) und andererseits die longitudinale Relaxation (T1-/ Spin-Gitter-Relaxation). Die Relaxationszeiten spiegeln sich in verschiedenen Signalstärken für unterschiedliche Gewebearten wieder. Die zugeführte Energie wird als elektromagnetische Hochfrequenzwelle abgestrahlt. Nach der Anregung durch den 90°-HF-Impuls klingt das Signal also ab. Zur Verstärkung des Signals und einer Maximierung des Echos wird nach der HF-Anregung ein zweiter 180°-Impuls eingestrahlt. Mit diesem kann das Signal aus der ersten Anregung refokussiert werden (Spin-Echo). So werden Einflüsse durch Magnetfeld-Inhomogenitäten und Protonen-Interaktionen kompensiert. (siehe in Kapitel 1.2 in [Unrath2006] sowie in [W2009a])

2.2 Magnetresonanzbildgebung

Die Magnetresonanzbildgebung (MRT) oder Kernspintomographie beruht auf den Eigenschaften der Kernspinresonanz und befindet sich seit 1981 im klinischen Einsatz. Entsprechend der NMR misst sie die Signale der Proto-

nen von Molekülen. Die Messergebnisse informieren über die vorhandene Wasserdichte, sowie die Position des beweglichen Wassers im Raum (molekulare Diffusion und Fluss).
Hierzu wird ein beliebiger Körper in einem magnetischen Feld platziert. Durch ortsabhängige Magnetfeldgradienten lassen sich transversale Schnittbilder erzeugen. Die Rekonstruktion der Ortsinformation wird durch drei zusätzliche Magnetspulen in x-, y- und z-Richtung ermöglicht. Jede dieser Spulen baut ein eigenes Magnetfeld auf, welches das bestehende Feld überlagert. So wird garantiert, dass in jedem Punkt eine unterschiedliche Feldstärke existiert, die die Ortsinformation kodiert.
Die entstehenden Tomographien setzen sich aus Bildelementen, den sogenannten Voxeln, mit unterschiedlicher Intensität zusammen. Die räumliche Auflösung ist mit ein bis drei Millimeter sehr fein. Grundlage für die Kontraste sind sowohl die unterschiedlichen T1-, beziehungsweise T2-Relaxationszeiten der Gewebearten, als auch der jeweilige Gehalt von Wasserstoffatomen. Jedoch bietet die konventionelle Magnetresonanztomographie nicht in allen Gewebestoffen gute Kontraste. Insbesondere Messungen der weißen Substanz erscheinen homogen, da die Faserbündel gleiche T1- und T2-Relaxationszeiten besitzen.
Vorteil dieses bildgebenden Verfahrens gegenüber anderen Methoden, wie dem Röntgen oder der Computertomografie, ist, dass keine ionisierende Strahlung Verwendung findet. Negative Auswirkungen auf organische Strukturen bleiben somit aus.

2.2.1 Spin-Echo-Technik

Die Spin-Echo-Technik, wie oben bereits erläutert, gilt als Grundlage der klinischen MRT-Bildgebung. Sie bietet eine annehmbare Messzeit und durch ein hohes Signal-Rausch-Verhältnis (SNR: signal-noise ratio) auch ausreichende Qualität. Somit ermöglicht sie eine gute Differenzierung von Feinstrukturen.

2.2.2 Gradienten-Echo-Technik

Die Gradienten-Echo-Technik verzichtet auf den 180°-Impuls. Dadurch erfolgt die transversale Relaxation schneller. Im Gegensatz dazu entsteht aber ein deutlich geringerer SNR. Eine Verbesserung der Bildqualität wird durch längere Messzeiten, respektive der systeminternen Mittelung wiederholter Sequenzen erreicht. Eine noch schnellere Longitudinalrelaxation kann durch einen HF-Impulswinkel unter 90° erzielt werden.
Der offensichtliche Vorteil resultiert im Endeffekt in einer Verkürzung der Untersuchungszeit der Patienten. Die Anwendung der Gradienten-Echo-Technik ist aber in jedem Fall abhängig von der Fragestellung.

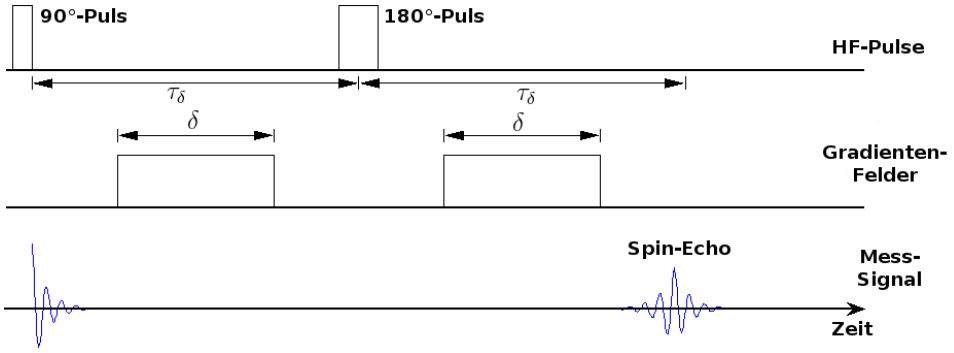

Abbildung 2: Schematische Darstellung der Stejskal-Tanner-Sequenz. Gemessen wird die Diffusionsbewegung in Richtung des anliegenden Gradientenfeldes. Das Signal ist eine Abschwächung des Spin-Echos.

2.2.3 Echo-Planar-Imaging

Diese Technik baut auf der Verwendung von Gradienten-Echo-Sequenzen auf. Dabei werden nach jedem HF-Impuls mehrere Echos ausgelesen. Mit der EPI-Technologie sind Aufnahmen in weniger als 50ms möglich. Daher eignet sich das Echo-Planar-Imaging insbesondere für Anwendungen, bei denen eine höchstmögliche Artefaktreduzierung von Bewegungen als Bedinung gilt und findet daher in der funktionellen Magnetresonanztomografie (fMRT), sowie in der DTI Verwendung. Die erforderten schnellen Gradientenschaltzeiten und hohe Maximalgradienten stellen hohe Anforderungen an die Hardware des Scanners.
(Abschnitt 2.2 siehe [Unrath2006] S. 5-7)

2.3 Diffusions-Tensor-Bildgebung

Die Diffusions-Tensor-Bildgebung basiert auf der Brownschen Molekularbewegung von Wasserteilchen. Diese beschreibt die zufällige Bewegung von Molekülen in Abhängigkeit von den thermischen Verhältnissen. Mit Hilfe der Diffusions-Tensor-Bildgebung war es erstmals möglich nichtinvasiv, strukturelle Information bestimmter komplizierter Gewebearten, wie der weißen Gehirnsubstanz, zu erhalten. Bereits 1965 wurde von Stejskal und Tanner der Grundstein hierfür gelegt, indem sie die Diffusionskonstante von Wasser mit einem Magnet-Resonanz-Tomografen und einem System aus Gradienten-Magnetfeldern ermittelten. Zu Beginn der achtziger Jahre wurden erste Technologien zur Diffusions-MRT vorgestellt. Allgemeines Prinzip war die Kodierung molekularer Diffusion mit Hilfe bipolarer Magnetfeld-Gradienten-Pulse.
Diese sogenannte diffusionsgewichtete MRT ermöglicht die Messung der molekularen Zufallsbewegung der Wasserteilchen. Um die technischen Aspekte der gewöhnlichen Magnetresonanztomografie für die Diffusions-

MRT zu sensibilisieren, muss zwischen den HF-Impulsen und der Datenauslese ein bipolares Paar Gradienten-Pulse geschaltet werden. Das erste geschaltete Gradientenfeld bewirkt, dass die lokale Feldstärke in der entsprechenden Gradientenrichtung variiert. Entlang dieser Richtung kommt es zu einer Dephasierung der Kerne. Durch das zweite, entgegengesetzte Gradientenfeld erfolgt die Re-Dephasierung. Die Kerne werden also wieder in Phase versetzt, vorausgesetzt es ist keine Bewegung der Moleküle erfolgt. Da es aber aufgrund der Brownschen Molekularbewegung zu einer Ortsveränderung einiger Teilchen kommt, bleibt eine Rest-Dephasierung erhalten. Diese beschreibt in einer Abschwächung des Signals die Diffusionsbewegung der Kerne, die sich in Richtung des Gradientenfeldes bewegt haben. Dieser Signalabfall, zeichnet sich umso deutlicher ab, je mehr Bewegung im Gewebe möglich ist. Zusätzlich zu den gradienten-gewichteten Messungen, ist eine weitere Aufnahme notwendig, bei der kein Feldgradient geschaltet ist. Im Vergleich mit deren Ergebnis, kann der Einfluss der Diffusionsbewegung ermittelt werden. Der somit ermittelte scheinbare Diffusionskoeffizient (ADC : apparent diffusion coefficient) beschreibt die Interaktion zwischen diffundierendem Molekül und Gewebestruktur innerhalb des Diffusionszeitraums. Die zu justierenden Messparameter sind die Stärke und Dauer der Diffusion sowie die Zeit zwischen den Einstrahlungen des bipolaren Gradientenimpulses. Insgesamt wird der Betrag der Parameter als Diffusionsgewichtung b festgehalten (b-Wert) . Da die Messsequenz wegen der Vielzahl an erfoderten Aufnahmen sehr anfällig für Bewegungsartefakte ist, wird, wie bereits erwähnt, das schnelle Echo-Planar-Imaging verwendet.
Der Vorteil der Diffusions-Tensor-Bildgebung besteht darin, dass die Diffusion an einem bestimmten Punkt von allen Richtungen gemessen werden kann. Dieser Nutzen zahlt sich aber nur aus, wenn Informationen über anisotropes Gewebe ermittelt werden sollen. In isotropen Strukturen ist die Ausprägung der Diffusion in alle Richtungen identisch. Bei einer üblichen Diffusionszeit von ungefähr 50ms beträgt der durchschnittliche Weg eines Wassermoleküls ungefähr 10m. Dabei stoßen die Moleküle aufeinander, kreuzen sich oder interagieren mit anderen Gewebestrukturen, wie Zellmembranen, Fasern oder Makromolekülen. Bei der Diffusions-MRT wird der Gesamteffekt pro Voxel gemessen. Bei einer Auflösung im Millimeterbereich reflektiert dieses Maß die Verschiebungsverteilung der Wassermoleküle. Diese Auswirkung gibt Hinweise auf die geometrische Organisation des Gewebes. In der Praxis findet die Diffusions-MRT bei der Früherkennung von Schlaganfällen Anwendung und liefert bei Erkrankungen wie Multipler Sklerose durch differenzierte Darstellung des zentralen Nervensystems unterstützende Erkenntnisse. Außerdem hilft die Diffusions-Tensor-Bildgebung bei der Operationsplanung vor Eingriffen im Gehirn, da hier der Erhalt von Nervenfasern von Bedeutung ist. Die ersten klinischen Applikationen wurden in den frühen 90er Jahren vorgeschlagen.

Unter anderem tat sich hier die Anwendung brain ischemia hervor.
Hauptbeobachtungsmerkmal ist die anisotrope Diffusion. Diese beschreibt den dreidimensionalen Prozess der richtungsabhängigen Molekularbewegung und resultiert aus der eigenartigen physikalischen Beschaffenheit des jeweiligen Mediums. Das bedeutet, dass sich die Diffusion entlang der entsprechenden Struktur ausrichtet. Außerdem weist die Anisotropie auf bestehende Hindernisse hin, die die Bewegung der Moleküle in bestimmte Raumrichtungen verhindern. Der Effekt der anisotropen Diffusion ist bei Richtungsänderung des Gradientenpulses feststellbar und repräsentiert für jede untersuchte Einheit des Volumens ein Verhaltensmuster der Diffusion, welches sich graphisch als sogenanntes Diffusionsellipsoid darstellen lässt. Anisotrope Diffusion kann vor allem im Muskelgewebe, und mit Hilfe der DTI auch im Rückenmark und der weißen Gehirnmasse erkannt werden.
Die anisotrope Diffusion in der weißen Substanz beruht auf der besonderen Organisation in Bündeln von parallel verlaufenden myelinisierten axonalen Nervenfasern. Die Diffusion in Richtung der Fasern ist schneller als zur Querrichtung. Somit gibt diese Hinweise auf die Faserstruktur unter der Annahme, dass die Richtung der schnellsten Diffusion die Orientierung der Fasern indiziert.
Entscheidend war der Formalismus des Diffusions-Tensors (Basser et al.). Mit dieser Beschreibung konnte die anisotrope Diffusion vollständig extrahiert und mehr Details über die Gewebestruktur erfasst werden. Infolgedessen traten weitere Studien hervor, die sich größtenteils auf die Optimierung der MRI Sequenzen beriefen.
Die wichtigsten Bestandteile in den aktuellen Studien sind der Zugriff auf den Diffusions-Tensor, die Verarbeitung und Visualisierung der DTI-Daten, sowie Applikationen. Dabei stelllt die Faser-Traktografie im Gehirn eine eine der fortschrittlichsten Anwendungen dar.

2.3.1 Diffusionstensor

Die einfache Diffusions MRI wird durch einen skalaren Parameter beschrieben, den scheinbaren Diffusionskoeffizienten. Entscheidend ist hierbei, dass in dem zu untersuchenden Gewebe, dieser Diffusionskoeffizient größtenteils unabhängig von der Orientierung des Gewebes ist (z.B. graue Materie). In diesem Fall spricht man auch von Isotropie. Ist jedoch die anisotrope Ausdehnung der untersuchten Materie von Bedeutung, reicht eine einzelne skalare Angabe nicht mehr aus. Stattdessen wird ein Tensor **D** notwendig, welcher die Molekularbewegung entlang jeder Raumrichtung und die Korrelation zwischen diesen Richtungen beschreibt:

$$\mathbf{D} = \begin{pmatrix} Dxx & Dxy & Dxz \\ Dyx & Dyy & Dyz \\ Dzx & Dzy & Dzz \end{pmatrix} \tag{1}$$

Die Messungen stehen immer im Bezugsrahmen der Gradienten des MRI Scanners. Dabei reflektieren die nicht-diagonalen Terme die Beziehungen zur molekularen Verschiebung in Querrichtung, da die Signalabschwächung einer Richtung auch von den perpendikulären Richtungen abhängt. Der Effekt der Diffusion auf das MRI Signal spiegelt sich in einer Abschwächung des Signals wieder. Dieser steht in Abhängigkeit zu dem scheinbaren Diffusionskoeffizienten und dem b-Faktor. Der b-Faktor charakterisiert den verwendeten Gradientenpuls und beschreibt die verwendeten Parameter, wie Zeitabfolge, Amplitude und Form in einem statischen Wert.

$$b = \gamma^2 G^2 \delta^2 \left(\Delta - \frac{\delta}{3} \right) \tag{2}$$

Insgesamt ergibt sich die Grundlegende Formel der DTI:

$$S = S_0 e^{-bg^T Dg} \tag{3}$$

2.3.2 Datenakquisition

Die Voraussetzungen zur Bestimmung des Diffusions-Tensors besteht in der Akquise diffusions-gewichteter Bilder entlang bestimmter Gradientenrichtungen. Hierbei wird meist auf das oben erwähnte Echo-Planar-Imaging zurückgegriffen. Durch die Symmetrieeigenschaft des Tensors ist eine Messung entlang von mindestens sechs Raumrichtungen notwendig, sowie eine Bildgebung ohne Diffusionsgewichtung. Aufgrund des sehr geringen Signal-Rausch-Verhältnisses sind wiederholte Messungen erforderlich, aus denen der Durchschnitt ermittelt wird. In Addition dazu ist es außerdem sinnvoll, entlang möglichst vieler Raumrichtungen zu messen um systematische Messfehler gering zu halten.

2.3.3 Datenverarbeitung

Aufgrund des niedrigen SNR ist eine Glättung der Daten auf unterster Stufe meistens Bedingung für eine korrekte Weiterverarbeitung. Die Filterung der Daten mittels Gauss-Maske hat sich dabei als sinnvoll erwiesen. Darauf aufbauend ist der nächste Schritt die Bestimmung der sechs Parameter des Diffusions-Tensors aus den DWI Werten. Da, wie vorhin bereits erwähnt, oft Messungen aus mehr als sechs Gradientenrichtungen vorliegen, ist es erforderlich diese auf sechs unabhängige Tensorparameter zu reduzieren. Mathematische Methoden, wie die multiple lineare Regression oder die Singulärwertzerlegung basierend auf der Methode der kleinsten Quadrate, sind mögliche Lösungsansätze zur Schätzung der korrekten Parameter.

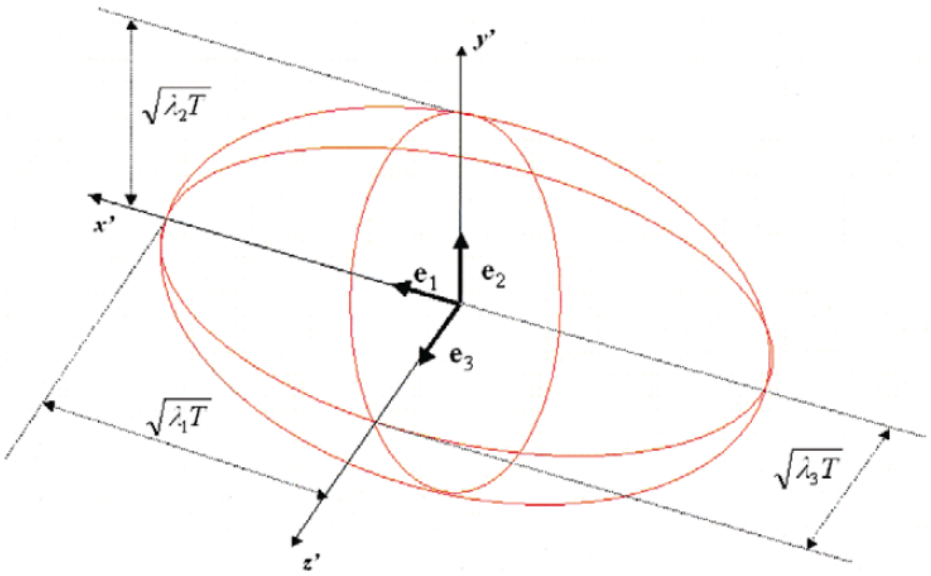

Abbildung 3: Schema der Anisotropie. Dargestellt als Diffusionsellipsoid. Die Oberfläche beschreibt den zurückgelgeten Weg während der Diffusionszeit vom Ursprung aus.

Grundlage des zu lösenden Gleichungsystems ist (Eq.3):

$$\begin{pmatrix} \frac{1}{b}(ln(S_0) - ln(S_1)) \\ \vdots \\ \frac{1}{b}(ln(S_0) - ln(S_i)) \\ \vdots \\ \frac{1}{b}(ln(S_0) - ln(S_n)) \end{pmatrix} = \begin{pmatrix} x_1^2 & y_1^2 & z_1^2 & x_1y_1 & y_1z_1 & z_1x_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^2 & y_i^2 & z_i^2 & x_iy_i & y_iz_i & z_ix_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & y_n^2 & z_n^2 & x_ny_n & y_nz_n & z_nx_n \end{pmatrix} \begin{pmatrix} D_{xx} \\ D_{yy} \\ D_{zz} \\ D_{xy} \\ D_{yz} \\ D_{zx} \end{pmatrix} \quad (4)$$

Der Diffusionstensor bietet nun durch Eigenanalysis die Möglichkeit zur Berechnung der Indizien pro Voxel. Die Bestimmung der Eigenvektoren (e_1, e_2, e_3) und der zugehörigen Eigenwerte (λ_1, λ_2, λ_3) lässt sich durch ein Diffusions-Ellipsoid veranschaulichen. Es visualisiert alle möglichen Parameter in einem geometrischen Objekt und zeigt die zurückgelegte Distanz der Moleküle im Raum in Bezug zu einer bestimmten Zeit. In einem kartesischen Koordinatensystem mit den Raumachsen x, y und z lassen sich die Diffusionsrichtungen und -strecken wie in Abbildung 3 visualisieren. Auf dieser Basis lassen sich weitere Hinweise auf Diffusion bestimmen:

- Durchschnittliche Diffusion
- Grad der der Anisotropie
- Bestimmung der Hauptdiffusionsrichtung und des Diffusionsgrads

Die *durchschnittliche Diffusion Tr(**D**)*beschreibt die durchschnittliche Größe des Ellipsoids und die eventuelle Gegenwart von Hindernissen.

$$Tr(\mathbf{D}) = \frac{D_{xx} + D_{yy} + D_{zz}}{3} \tag{5}$$

Der *Grad der Anisotropie* basiert auf der Kombination aus Termen des diagonalisierten Diffusionstenors, den Eigenwerten. Mögliche Varianten der Darstellung sind die relative Anisotropie (RA), die fraktionelle Anisotropie (FA), so wie das Volumen Verhältnis (VR). Grundlage ist bei allen Formen die Variation der Molekularbewegung und Bezug auf die Präsenz von orientierten Strukturen.

$$FA = \sqrt{\frac{3}{2}} \frac{\sqrt{(\lambda_1 - \lambda_M)^2 + (\lambda_2 - \lambda_M)^2 + (\lambda_3 - \lambda_M)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{6}$$

Die *Hauptdiffusionsrichtung* gibt die räumliche Orientierung der Struktur an. Hierbei besteht die Annahme, dass die Richtung der Fasern sich kolinear mit der Richtung des Eigenvektors des größten Eigenwerts verhält.

(Grundlage des Abschnitts 2.3 sind [Bihan2001], [Basser2002] und [Masutani2002])

3 Visualisierungsmethoden

Vorraussetzung für die Visualierung sind die sechs Tensorparameter. In Abhängigkeit von der erforderten Visualierungsmethode werden die darstellbaren Parameter reduziert, da es nahezu unmöglich ist alle sechs Parameter visuell verständlich darzustellen. Im Folgenden wird auf die gängigsten Visualierungsformen eingegangen, die auch in der zu entwickelnden Applikation verwirklicht werden sollen.
(siehe [Schurade2006] S.12-16 und [Masutani2002]

3.1 Zweidimensional

3.1.1 FA-Map

Eine der gebräuchlichsten Darstellungsformen ist die Karte der Anisotropie. Diese zeigt für eine ausgewählte Schicht des Volumens die anisotrope Gewichtung pro Volumenelement an. Hellere Regionen stellen hierbei Gewebe der weißen Gehirnsubstanz dar, dunkle Bereiche hingegen kennzeichnen Vorkommen der grauen Materie. So lässt sich die zusammenhängende Faserstruktur von anderem Gewebe gut differenzieren.

Abbildung 4: Karte der Anisotropie. Grauwertdarstellung.

3.1.2 Richtungskarte der fraktionellen Anisotropie

Zusätzlich zu der Darstellung eines Graustufenbilds, kann die informelle Visualierung mit Hilfe der Eigenvektoren farblich erweitert werden. Hierzu wird der der Eigenvektor mit dem größten Eigenwert verwendet, also die anzunehmende Hauptrichtung der Diffusion und die damit einhergehende Orientierung der Fasern.
Die Kodierung der Farbinformation beruht auf einem 24-Bit-Farbbereich. Die jeweiligen Farbkanäle können dabei Werte im Bereich von 0 bis 255 annehmen. Die einzelnen Komponenten des Haupteigenvektors lassen sich leicht auf dieses Farbschema übertragen. Zur Unterdrückung isotroper Lokalitäten kann der Farbvektor mit dem vorliegenden FA-Wert gewichtet werden.

3.2 Geometrische Darstellung der Ellipsoide

Diese Darstellungsform basiert auf der Verwendung dreidimensionalgeometrischer Objekte. Die Objekte visualisieren die räumliche Verteilung der Anisotropie und die Hauptdiffusionsrichtung pro Diffusionstensor mit Hilfe von Ellipsoiden, Pfeilen oder anderen geeigneten Formen. Wie in Abbildung 3 gezeigt, repräsentiert ein Diffusionsellipsoid den zurückgelegten Weg der Moleküle in einem bestimmten Zeitabschnitt. Diese Darstellungsmethode eignet sich vor allem für zweidimensionle Schnittbilder, da die Visualsierung für einen dreidimensionalen Raum schnell sehr unübersichtlich wird. Grund dafür ist vor allem die Betrachteransicht auf die Ellipsoide, welche projiziert nicht den gesamten Anisotropieeffekt darstellen

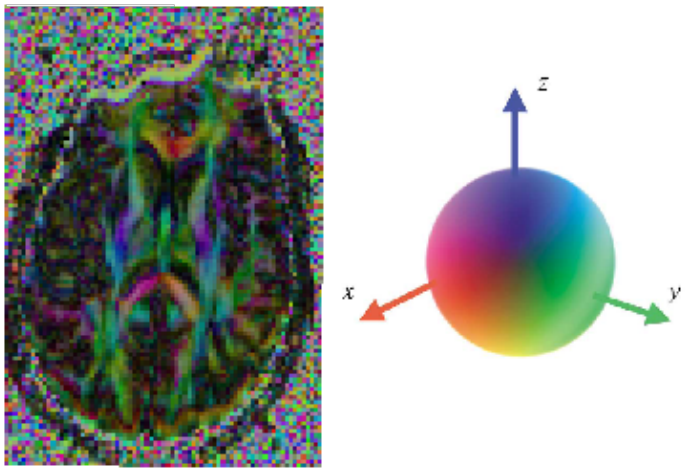

Abbildung 5: Richtungskarte der Anisotropie. Farbkugel repräsentiert Farbkodierung der Richtungen.

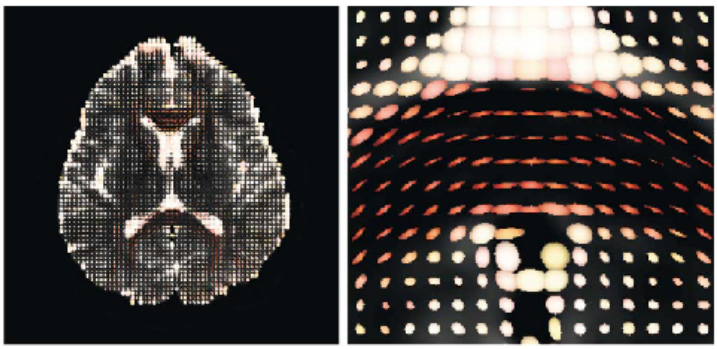

Abbildung 6: Visualisierung der Tensorinformation mittels Ellipsoiden.

können. Abbildung 6 zeigt eine mögliche Ausprägung dieser Methode.

3.3 Dreidimensionale Rekonstruktion

Die Traktografie oder auch Fibertracking ist ein häufg angewandtes Verfahren zur dreidimensionalen Rekonstruktion ganzer Nervenfaserbündel. Mit Hilfe dreidimensionaler Stromlinien (Streamlines) kann der Verlauf der Fasern dargestellt werden. Dabei bestimmt der Eigenvektor mit dem größten Eigenwert den Richtungsverlauf. Zur Untersuchung des Faserverlaufs bieten sich zwei Möglichkeiten an: entweder wird vor der Berechnung eine bestimmte Region ausgewählt (ROI: region of interest), in der das Tracking von sogenannten Saatpunkten aus gestartet wird, oder es wird für jedes Voxel des gesamten Volumens eine Stromlinie propagiert („brute-force“). Zur Berechnung des Faserverlaufs wird im Allgemeinen der FACT-Algorithmus verwendet (fiber assignment by con-

tinuous tracking). Dieser bestimmt die lineare Fortsetzung des Trackings entsprechend der Winkel aufeinanderfolgender Eigenvektoren und dem Wert der fraktionellen Anisotropie. Das Verfolgen der Fasern wird in der Regel, bei einem FA-Wert kleiner als 0.2, respektive einem Skalarprodukt der beiden Vektoren größer als 0.75, abgebrochen.

FACT-Algorithmus:
Pro Voxel wird ausgehend von einem Saatpunkt P_0 eine Linie gestartet. Die einzelnen Liniensegmente werden pro Voxel gezeichnet, wobei die Stützstellen auf der Voxelbegrenzung liegen und die Richtung parallel zum längsten Eigenvektor verläuft [Schurade2006]:

1. Wahl des Saatpunkts P_0.
2. Ermittlung der Faserrichtung des P_i entsprechenden Voxels.
3. Berechnen des Schnittpunkts P_{i+1} des Richtungsvektors mit der Voxelbegrenzung.
4. Bestimmung des Richtungsvektors des angrenzenden Voxels in Richtung der in Schritt 2 ermittelten Faserrichtung.
5. Test auf Abbruchkriterien.
   (a) Abbruchkriterien erfüllt: Hinzufügen des Punktes P_{i+1} zum Polygon. Fortsetzung mit Schritt 2.
   (b) Abbruchkriterien nicht erfüllt: Abbruch des Trackings für die aktuelle Linie.

Nachteil dieser Visualisierungsform ist die Schwierigkeit der Überprüfung auf Korrektheit, da sich die Fasern willkürlich auffächern oder kreuzen können.

4 Zielsetzung

Ziel der Studienarbeit ist die Erstellung einer Beispielapplikation zur Visualisierung von DTI-Daten mit Hilfe der Grafikhardware. Als Grundlage dient das CASCADA-Framework, welches die notwendigen Bausteine zur medizinischen Bildverarbeitung und Visualisierung bereits liefert. Außerdem unterstützt das Framework durch unterschiedliche Abstraktionsebenen die Implementierung. Hierzu folgt später ein umfassender Überblick.
Zunächst sollen die relevanten Informationen für die Erzeugung von geometrischen Repräsentationen aus den Eingabedaten berechnet werden. Dafür werden entsprechende medizinische Bildformate analysiert und

analog dazu die notwendigen Funktionen zum Einlesen der Daten implementiert. Diese legen die Messdaten in geeigneter interner Repräsentation im Speicher ab. Abhängig von der Qualität der Daten soll auf dieser Ebene die Möglichkeit gegeben werden die Daten zu bereinigen. Danach folgen die erforderlichen mathematischen Operationen auf dem Datensatz, um die notwendigen Tensorinformation zu bestimmen. Für die interaktive Visualisierung dieser Daten werden die Möglichkeiten moderner Grafikhardware, insbesondere des Geometry Shaders, ausgenutzt. Die Visualierungsmethoden, die im Rahmen dieser Studienarbeit entwickelt werden sollen, beziehen sich auf die Darstellung der Diffusion mit Hilfe von Glyphen, respektive sogenannter Streamlines innerhalb interaktiv ausgewählter Regionen.
Neben der eigentlichen Umsetzung der Applikation sollen die persönlichen Kenntnisse im Umgang mit OpenGL und GLSL in Bezug auf Shaderprogrammierung, vor allem in Verbindung mit dem Geometry-Shader, vertieft werden.

5 Hintergrund und Technologien

Die entwickelte Applikation basiert auf modernen Techniken der GPU-Programmierung. Von daher soll hier ein grober Überblick über die neuesten Entwicklungen der GPU-Architektur gegeben und die verwendeten Technologien erläutert werden.
In der letzten Jahren hat die programmierbare Grafikhardware zunehmend an Bedeutung gewonnen. Durch die Flexibilisierung der Fixed-Funktion-Pipeline wurde die Verwendung der Grafikhardware auch für nicht-grafikspezifische Anwendungen (GPGPU: General Purpose Computation on Graphics Processing Unit) interessant. Moderne Grafikprozessoren unterstützen voll programmierbare Prozessoreinheiten, sogenannte Shader, welche Vektorrechnungen mit Fließkommawerten bei hoher Präzision ermöglichen (IEEE single precision). Mit Hilfe von Shaderhochsprachen wurden diese Einheiten, welche die Vertex- und Pixelverarbeitung tragen, für die Implementierung zugänglich. Die viel höhere Rechenleistung gegenüber der CPU bringt einen extremen Performanzgewinn bei rechenintensiven Anwendungen, die parallele Datenverarbeitung erlauben, wie beispielsweise in vielen Bereichen der Bildverarbeitung und der Visualisierung. Der enorme Performanz-Zuwachs ist vor allem durch die rasante Entwicklungen auf dem Spiele-Markt zu erklären. Die Portierung, von komplexen, zuvor für die CPU effizient implementierten Algorithmen, ist in de meisten Fällen jedoch mit hohem Aufwand verbunden. Von hardware-technischer Seite stellt der kleinere Grafikkartenspeicher eine Limitierung dar. Außerdem ist die Bandbreite für den erforder-

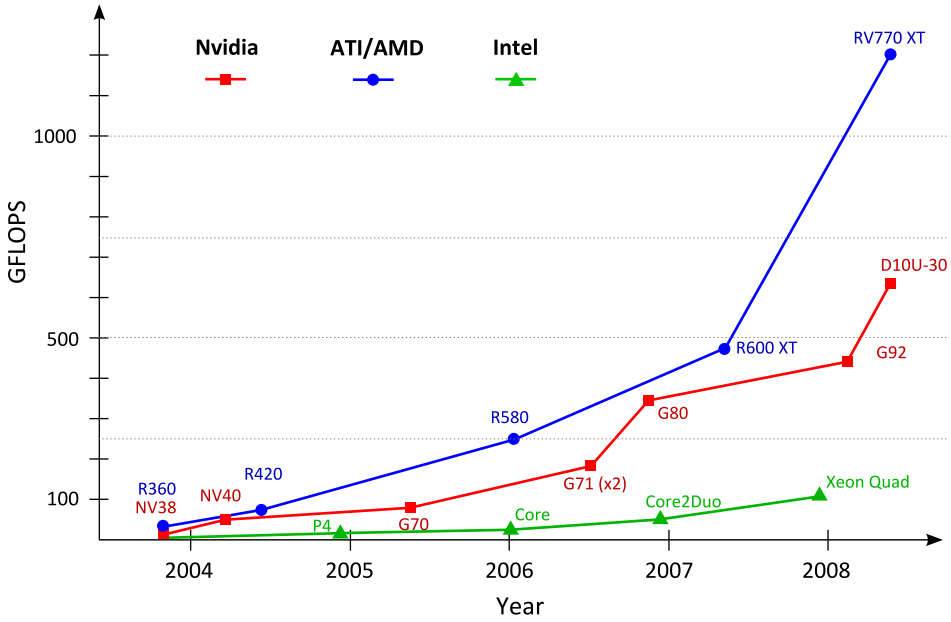

Abbildung 7: Enormer Performanzzuwachs bei Fließkommaberechnungen der GPUs gegegenüber der CPU in den letzten Jahren.

ten Datentransfer zwischen CPU und GPU der Flaschenhals der GPU-Programmierung. (Vgl. [Owens2005])

5.1 GPU-Architektur: OpenGL und GLSL

GLSL wurde als Shaderhochsprache unter OpenGL entwickelt und bringt Unterstützung für die Programmierung des Vertex-, Fragment- und mit Hilfe von Erweiterungen, auch Unterstützung für die Geometry-Shader-Programmierung mit sich. GLSL bietet eine an C angelehnte Syntax.
Der Vertexshader kann im Gegensatz zur statischen Grafik-Pipeline die Vertex-Attribute modifizieren. Darunter fällt unter anderem die Transformation der Position und Normalen, die Pro-Vertex-Beleuchtung und das Generieren, beziehungsweise Transformieren, der Texturkoordinaten. Mögliche Eingabedaten für den Vertexshader sind die jeweilige Position, die Farbe, die Normale und die Texturkoordinaten des Vertex sowie durch die Applikation definierte weitere Attribute.
Nach der Zusammensetzung der Primitive stehen dem Fragmentshader die interpolierten Fragment-Informationen, wie Position, Normale und Farbe, zur Verfügung. Hier übernimmt der Shader die Texturierung und Einfärbung der Fragmente.
Der Einsatz eines Shaders ersetzt die volle Funktionalität für die jeweilige Stufe der statischen Pipeline. Die jeweilgen Shaderprogramme werden

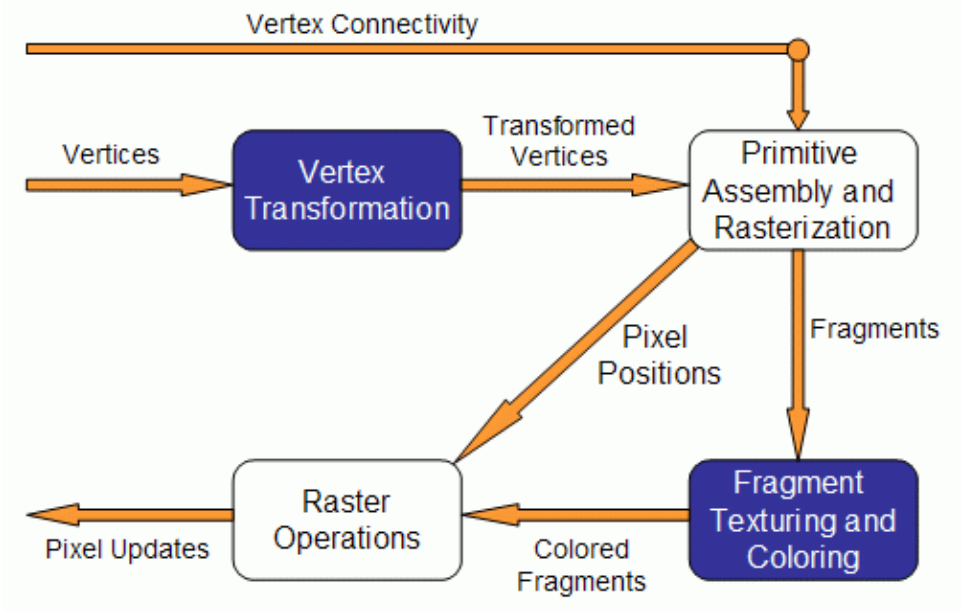

Abbildung 8: Vereinfachtes Diagramm der Grafikpipeline. Blaue Stationen visualisieren den möglichen Einsatz von Vertex- bzw. Fragmentshadern

vom Grafikkartentreiber kompiliert und demzufolge als Programmobjekte auf der GPU festgehalten. Bei Bedarf werden die Shaderprogramme aktiviert, dabei kann jedoch immer nur ein Programm aktiv sein. Ist im Gegensatz dazu kein Shader aktiviert, erfolgt die Verarbeitung der Vertices durch die Fixed-Function-Pipeline. [L3D2009]

5.2 Geometryshader

Der Geometyshader ist ein relativ neuer Shadertyp und wird nach dem Vertexshader respektive nach der Zusammensetzung der Primitive und vor dem Clipping ausgeführt. Der Shader arbeitet also auf Eingabeprimitiven, wie Punkten, Linien und Dreiecken, die nach dem Vertexshader bereit stehen. Mit Hilfe des Geometryshaders kann bestehende Geometrie verändert werden. Einerseite besteht die Option neue Geometriedaten zu erstellen, andererseits alte Geometrie zu verwerfen. Primitive, die den Geometryshader verlassen, werden geclipped und weiterverarbeitet, äquivalent zur Arbeitsweise der OpenGL-Applikation. Die Eingabe-Primitive existieren nach dem Geometryshader-Prozess nicht mehr. Diese können also entfernt werden, indem sie nicht explizit vom Geometryshader emittiert werden. Zusätzlich besteht die Möglichkeit die Primitive vor dem Clipping abzufangen und in einem Vertex Buffer Object (VBO) abzulegen. Dabei kann die Stufe der Rasterisierung deaktiviert werden, wenn noch keine Darstellung erfolgen soll. Des weiteren ist der Zugriff auf Nachbar-Vertices mög-

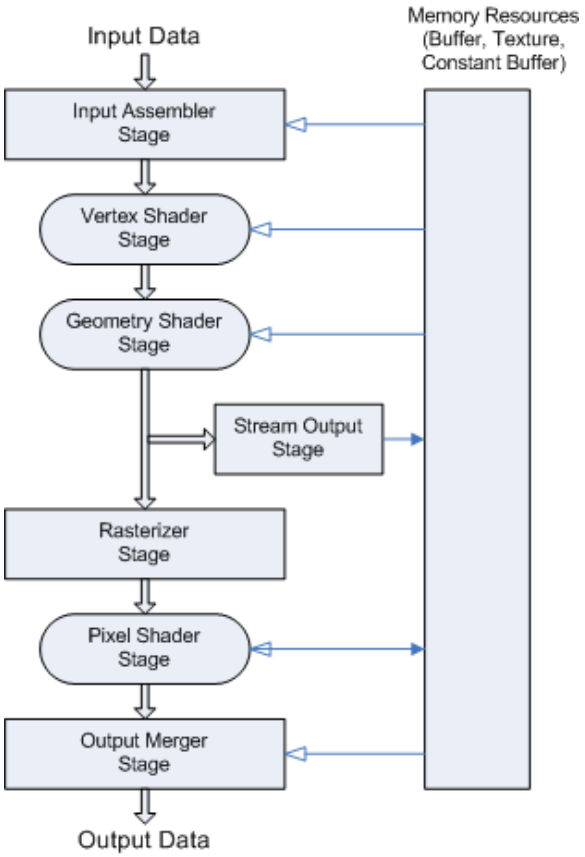

Abbildung 9: Diagramm der Grafikpipeline unter Einsatz des Geometryshaders.

lich, wenn die Eingabedaten entsprechend vordefiniert sind. Je nach GL-Implementation ist auch eine bestimmte Anzahl an Textuzugriffen möglich.
Der Einsatz des Geometryshaders erfordert die Angabe des Typs der Ein- und Ausgabe-Primitive, sowie die genaue Anzahl der auszugebenden Varying-Variablen. Damit sind unter anderem die Gesamtanzahl der Vertices gemeint, die die Shadereinheit verlassen dürfen. [vArb2008], [NVGSEXT]

6 Implementierung

Zunächst soll auf die Architektur des verwendeten Frameworks CASCADA eingegangen werden. Darauf folgt ein Überblick über die Integration der eigenen Applikation sowie Erläuterungen zum verwendeten Datensatz. Anschließend wird anhand des Programmablaufs die genaue Implementation aufgezeigt.

6.1 Architektur des CASCADA-Frameworks

CASCADA ist ein Framework zur Visualierung und Verarbeitung (hauptsächlich medizinischer) Volumendaten auf Basis der GPU. Dabei werden rechenlastige und visualisierende Aufgaben nach Möglichkeit auf der Grafikhardware appliziert. Die meisten visualisierenden und bildverarbeitenden Operationen sind zum Vergleich ebenfalls als CPU-Version realisiert. Die verwendeten Technologien umfassen objekt-orientierte Programmieung mittels C++ und Entwurfsmustern wie das Singleton-, Observer- und Composite-Pattern. Die Grafik- beziehungsweise Shaderprogrammierung ist mit Hilfe von OpenGL und GLSL verwirklicht. Die auf dem System aufgesetzte Benutzeroberfläche basiert auf Qt.
Die Architektur des Frameworks beruht vor allem auf Abstraktion von der konkreten Grafikprogrammierung. Dazu kapseln Sequenzen hierarchisch die Operationen. Eine Sequenz kann dabei aus mehreren Rendervorgängen (Renderpass) bestehen, welche auf auf die existierende Geometrie angewendet werden. Dazu repräsentiert ein Renderpass ein Shaderprogramm, welches zum Beispiel das Rendern in Texturen ermöglicht, die dem nächsten Pass als Eingabe dienen. Des weiteren vereinfacht ein Shadersystem das Erzeugen von Shadern aus mehreren GLSL-Quelldateien und den Umgang mit Uniform-Parametern. Das Erzeugen von Geometrie funtkioniert mit Hilfe von abstrahierten Geometrie-Objekten und der Kapselung von OpenGL-Render-Befehlen.
Da die GPU für das Arbeiten mit zwei-dimensionalen Texturen optimiert wurde, ist eine entsprechende Repräsentation und Transformation der Volumen realisiert worden. Die Volumendaten werden somit auf der CPU,

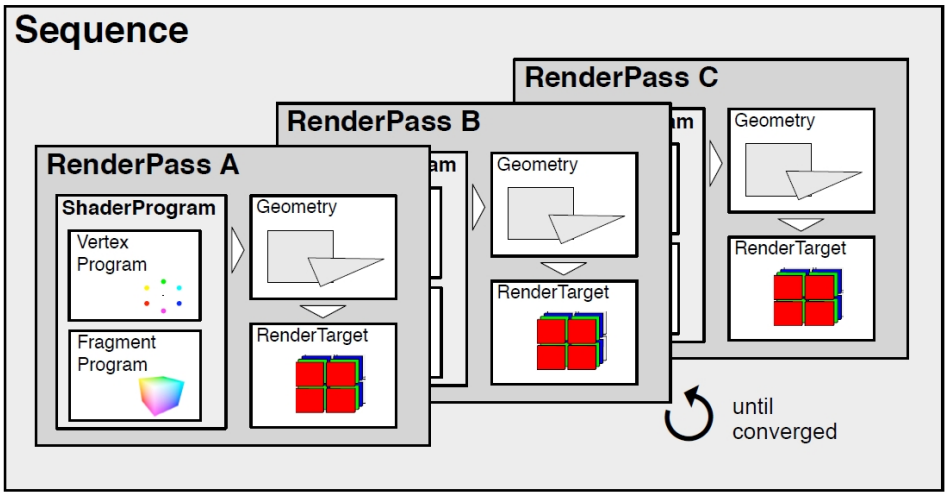

Abbildung 10: Hierarchie der Render-Komponenten in CASCADA

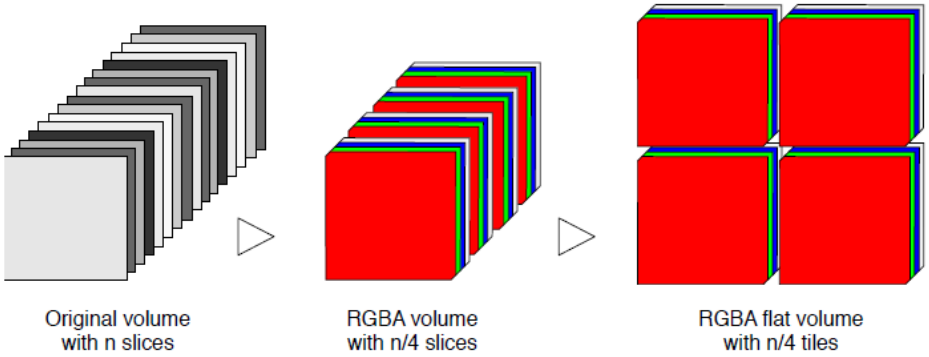

Abbildung 11: Volumenrepräsentation in CASCADA

entsprechend der Textureinheit, als RGBA-Tupel abgelegt. Als Resultat liegen vier aufeinanderfolgende Schnittbilder in einer zweidimensionalen Textur vor. Des weiteren können die immer noch dreidimensionalen Datensätze für die GPU optimiert werden, indem die hintereinanderliegen RGBA-Texturen als sogenannte Flat-3D-Textur repräsentiert werden. Die nun vorliegende zweidimensionale Volumenrepräsentation bietet performanteren Textur-Schreib-Zugriff und Abwärtskompatibilät zu älterer Hardware. Zusätzlich existiert noch eine weitere Volumenkompression, auf die aber hier nicht näher eingegangen werden soll. Eine besondere Erweiterung liefert CASCADA mit der erweiterteten Initialisierung der Uniform-Parameter. Die innerhalb einer Sequenz auf Applikationsebene initialisierten Uniforms werden zunächst in einer Map gespeichert und erst beim Erzeugen des Shaderprogrammobjekts, mit Hilfe einer speziellen Syntax in den Shaderfiles gebunden.

Als Standardvisualisierungsmethode dient die multi-planare Reformierung (MPR). Diese komponiert die Ansichten der Axial-, Coronal- und Sagitalebene in einer dreidimensionalen Darstellung. Durch Benutzereingabe können die Schichten in x-,y- und z- Richtung durchwandert werden. Weitere Visualisierungsformen sind das direkte Volumenrendering (DVR) mittels GPU Ray Casting, Klassifikation durch Transferfunktionen und die Darstellungsmöglichkeit geometrischer Primitive im DVR. (Siehe [Raspe2009] S. 175 ff)

6.2 Integration

Da das CASCADA-Framework in seiner aktuellen Version einen umfassenden Baukasten mit sich bringt, wurde zur leichteren Orientierung und Vermeidung von Konflikten das System auf die, für die Applikation notwenigen Bausteine, reduziert. Das bedeutet, dass vorerst nicht zu verwendende Sequenzen entfernt worden sind. Diese umfassen die Bild-Vorverarbeitungs-Prozesse und nicht benötigte Rendering-Sequenzen. Auf die in das System integrierte Wavelet-Volumekompression wurde auch verzichtet. Das Framework soll also hier eher als Bibliothek verwendet werden, um über die nötigen Abstraktionen, wie das Shader-Management, die Volumenrepräsentation und das Sequenz-System zu verfügen. Außerdem sollen diese nach Anforderung erweitert werden können. Im Folgenden soll dies genauer betrachtet werden.

6.2.1 Volumenrepräsentation und Reader

An Stelle von auf einem skalaren Wert pro Voxel basierenden Volumen, müssen nun Volumen repräsentiert werden, die mehrere Werte pro Voxel besitzen (DTI- oder DWI-Daten). Entsprechend der Anzahl der Voxelwerte werden neue Volumen erzeugt. Dabei entspricht dann jedes Volumen einem Parameter. So kann die bestehende Volumenrepräsentation weiterhin verwendet werden, nur die Anzahl der Volumen vervielfacht sich. Diesbezügliche Anpassungen mussten beim Einlesen der Daten vollzogen werden. Die hauptsächlich betroffenen Klassen waren der RawReader und der VolumeManager. Zudem musste für das Nrrd-Datenformat eine weitere Reader-Klasse konstruiert werden. Hierzu folgen später genaue Angaben. Des Weiteren wurde darauf geachtet, dass Sequenzen und GPU mit mehreren großen Texturen zurechtkommen.

6.2.2 GeometryManager

In der GeometryManager-Klasse musste eine Erweiterung der Vertex-Repräsentation erfolgen. Bisher wurde die Position eines Vertex als 3-Tupel

repräsentiert, wobei die homogene Koordinate auf 1 stand und nicht manipuliert werden konnte. Da das verwendete Geometryshader-Programm für das Fibertracking als Eingabe unterschiedliche w-Komponenten erfordert, muss diese auch variabel definiert werden können. Aus diesem Grund ist eine Umformung der Struktur von Vec3f auf Vec4f erfolgt. Entsprechende Anpassungen in anderen Klassen wurden vorgenommen.

6.2.3 Benutzeroberfläche und Interaktion

Die Benutzeroberfläche wurde einhergehend mit der Entfernung der Sequenzen reduziert, aber nicht weitgehend verändert. Die möglichen Operationen sind entsprechend dem alten Muster über Text-Buttons zugänglich.
Da jetzt mehrere Volumen im Speicher vorliegen, sollte auch die Möglichkeit geboten werden, das gewünschte Volumen anzuzeigen. Dazu wurde eine Auswahlbox in die Menüleiste integriet. Diese wird aktualisiert wenn neue Volumen hinzugefügt werden. Bei Selektion wird das entsprechende Volumen visualisiert.
In der Klasse GLWidget, welche den aktuellen Framebufferinhalt zeichnet, sollte die Darstellung einer ROI möglich werden. Dazu konnten die bestehenden Implementationen für die Mauskontrolle erweitert werden.

6.2.4 Sequenzen

Berechnungen und Visualierungen haben das oben erwähnte Sequenz-System als Grundlage. Für die Glättung und Visualisierung der Glyphen respektive Fasern wurde jeweils eine sogenannte GLRenderSequence implemtiert. Die Berechnung des Tensors erfolgt aufgrund der Komplexität auf CPU Ebene innerhalb einer CPURenderSequence (siehe Abbildung 11). Der weiter unten erläuterte Programmablauf in Abschnitt 7.4 stellt die Sequenzen zueinader in Bezug.

6.3 Datensatz

Die Applikation befindet sich zu dem jetzigen Zeitpunkt, was die Abhängigkeit von verschiedenen Datensätzen angeht, noch in einem relativ statischen Zustand. Die Zugänglichkeit zu DWI- bzw. DTI-Datensätzen ist doch sehr begrenzt und so konnte die Applikation nur auf wenige Datensätze ausgerichtet werden. Einer von ihnen ist der Brain-DT-Datensatz, den das „Scientific Computing and Imaging Institute“ der Universität von Utah auf seiner Internetpräsenz zur Verfügung stellt.Der Datensatz ist vollständig durch drei Dateien definiert: ein DWI-Datensatz bestehend aus zwölf diffusionsgewichteten Messungen, eine B0-gewichtete Referenz-Messung und

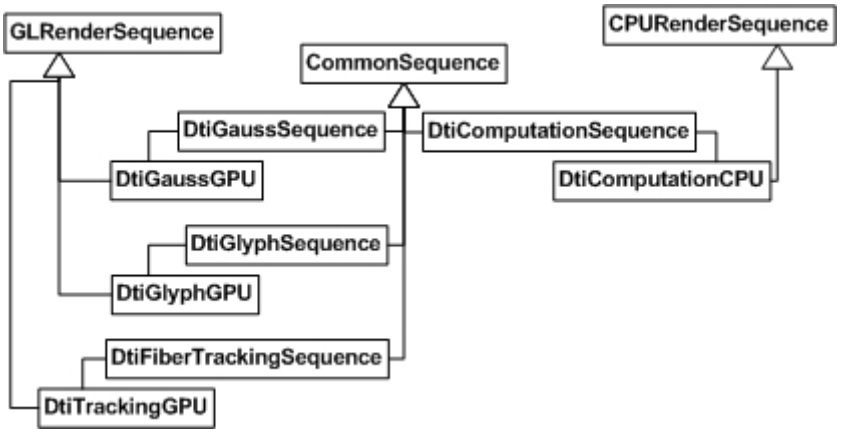

Abbildung 12: Schematisches Klassendiagramm. Integration der Dti-Renderkomponenten.

die Gradientenangaben. Der Datensatz repräsentiert nur einen Teil des gesamten Gehirns: von Oben bis zur ungefähren Mitte (unter dem Corpus Collosum, aber über den Augen). [SCIUtah]

6.4 Programmablauf

Dieser Abschnitt erläutert die Implementation der Applikation in Bezug auf den in Abbildung 12 dargestellten Programmablauf. Vom Einlesen der Datensätze, über die Vorverarbeitung und Berechung bis hin zur gewünschten Visualisierung werden die einzelnen Etappen beschrieben. Zur Unterstützung der DWI- und DTI-Datensätze wurden neue Datenstrukturen und Datenformate verwendet. Die akzeptierten Formate sind RAW-Dateien mit einer zusätzlichen Info-Datei, welche die notwendigen Datenstruktur-Informationen liefert. Des Weiteren werden NRRD-Datensätze unterstützt.

6.4.1 NRRD: Nearly Raw Raster Data

NRRD beschreibt zugleich eine Bibliothek und ein Dateiformat, welches die wissenschaftliche Visualierung und Bildverarbeitung unterstützt. Das Dateiformat umfasst mehrdimensionale Rasterdaten und arbeitet intern mit verschiedenen Datentypen (integrale und Float-Datentypen). Die Bibliothek behandelt auch andere Dateiformate und bietet neben dem Lesen und Schreiben der Daten auch die Möglichkeit auf den Datensätzen zu operieren. In dieser Arbeit sind die Bildanalyse und -verarbeitungsmöglichkeiten der NRRD-Bibliothek jedoch nicht von Bedeutung und sollen daher nicht weiter beachtet werden. Die Anwendung erfordert lediglich das Einlesen von NRRD-Dateien mit Hilfe der Biblio-

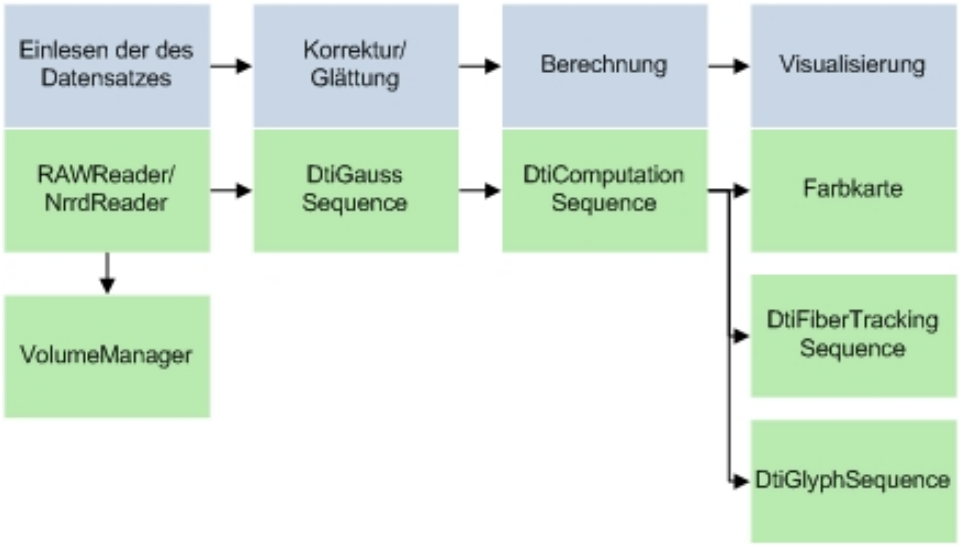

Abbildung 13: Schematische Darstellung des Programmablaufs.

```
NRRD0001
type: float
dimension: 4
sizes: 12 75 109 29
spacings:  NaN 2 2 2
units: "" "mm" "mm" "mm"
endian: big
encoding: raw
```

Abbildung 14: Beispiel: NRRD-Headerdatei.

thek.
Die Daten im Arbeits- und Festplattenspeicher setzen eine strikte lineare Sortierung voraus. Das NRRD-Format organisiert die Beziehung zwischen der Struktur multi-dimensionaler Rasterdaten und der ein-dimensionalen physikalischen Repräsentation. An einem Beispiel (s. Abbildung 14) sollen die Aspekte der Datenorganisation erklärt werden. Die Dimension des Rasters wird durch skalare Werte repräsentiert. Die Anzahl der Skalare steht für die Anzahl der lokalen Komponenten und die Angabe der räumlichen Ausdehnung. Dabei ist die Reihenfolge der Daten grundlegend. Die Werte stehen in der Datei in Reihe und sind von „schnell“ nach „langsam“ sortiert. Dies bedeutet, dass beim sequentiellen Durchlaufen der Daten die erste Achse am schnellsten inkrementiert und die folgenden entsprechend der Reihenfolge langsamer. Die Sortierung der Achsen ist über den imaginären Index der Achsensortierung zu identifizieren. Im

Beispiel also: $12 \rightarrow 75 \rightarrow 109 \rightarrow 29$.
Die Information über die Achsen ist relevant für das Volumen-Rendering. Dazu gibt *spacings* die Abstände der Abtastungen entsprechend der Achsensortierung an. Unter *units* findet sich die reale Ausdehnung pro Voxel wieder. Des Weiteren ist bei *type* der Datentyp angegeben. Mögliche Typangaben sind integrale Typen und Fließkommaangaben mit einfacher und doppelter Präzision analog zur C-Syntax. Das NRRD Format zeigt sich in zwei Ausprägungen, einerseits mit der Endung **.nrrd* für Datensatz inklusive Headerdatei in ASCII-Text, andererseits als Aufspaltung in **.nhdr*-Headerdatei mit Verweis auf externe RAW-Daten. Die Behandlung der Byte-Reihenfolge durch die Bibliothek erfolgt abhängig vom Endian.

6.4.2 RAW Daten und Strukturierung der Info Datei

Die Infodatei enthält die notwendigen Angaben zur korrekten Repräsentation der Daten der RAW-Datei in einer geeigneten Datenstruktur. Notwendige Angaben sind die Anzahl der Werte die pro Voxel repräsentiert werden sowie die Angabe der Volumengröße in Richtung der Hauptachsen, die dimensionale Ausdehnung eines Voxels, der Datentyp und die optionale Angabe von Bezeichnern für die spätere Identifikation der Volumen.

6.4.3 NrrdReader

Der *NrrdReader* empfängt die Eingabedatei als String, identifiziert die Struktur der Daten durch den Nrrd-Header und ließt die Daten entsprechend ein. Zusätzlich wird auf die Existenz eines weiteren B0-Datensatzes untersucht und die notwendige Gradientendatei eingelesen. Für das Einlesen anders strukturierter Dateien muss der Reader noch entsprechend flexibilisiert werden. Für die Anzahl der beinhalteten Volumen in der DWI-Datei wird die gleiche Anzahl an Volumen erstellt und sequentiell mit Werten initialisiert. Bei der verwendeten abstrakten Volumen-Datenstruktur handelt es sich auf CPU Basis um einen Vektor, der die Werte sequentiell ablegt und entsprechende Zugriffsoperatoren unterstützt. Im Folgenden werden die initialisierten Volumen komprimiert. Das bedeutet, dass jeweils vier schichtweise hintereinanderliegende Volumenwerte in einem Vektor, bestehend aus vier Komponenten, abgelegt werden. Die Daten werden also für die spätere Verwendung der oben erwähnten Flat3D-Texturen eingerichtet. Abschließend werden die einzelnen Volumen zur Verwaltung dem VolumeManager hinzugefügt. Die Volumen werden durch diesen in einer Map verwaltet. So stehen die Volumen durch Angabe eines zugehörigen String-Bezeichners global zur Verfügung.
Die eingelesenen Gradientendaten werden für den globalen Zugriff in

das Parameterset aufgenommen. Hierbei werden jeweils drei aufeinanderfolgende Werte als ein Richtungsgradient in einem Vec3f-Vektor zusammengefasst. Die Reihenfolge der Gradienten in der Textdatei muss also entsprechend aufgebaut sein.
Nach der Datenaufnahme ist das System bereit die Volumen entsprechend zu visualieren. Als Standardvisualisierungsform wird das erste, im Volumen-Manager präsente Volumen als multi-planare Darstellung gerendert. Über die Benutzeroberfläche kann mit Hilfe der ComboBox jedes vorhandene Volumen selektiert und angezeigt werden.

6.5 Korrektur/ Filterung - DtiGaussSequence

Zur Verbesserung des geringen Signal-Rausch-Verhältnisses wurde ein einfacher Gauss-Filter implementiert. Die Sequenz erhält alle vorhandenen Volumen des VolumeManagers als Eingabe und führt die implementierte DtiGaussGPU-Sequenz iterativ auf jedem Volumen aus. Die GPU-Sequenz fordert jeweils das Eingabe-Volumen als Flat-3D-Textur an und erzeugt den dazu entsprechenden Parameter im ParameterSet, auf den im verwendeten Shader als Uniform-Textur zugegriffen werden kann. Außerdem wird eine 5x5-Gauss-Filter-Maske vorbereitet, die ebenfalls dem Fragmentshader als Variable zur Verfügung steht. Des Weiteren werden die benötigten Shader-Dateien registriert und mit Hilfe des ShaderManagers zu einem Shaderprogramm-Objekt zusammengefügt.
Der angewandte Vertexshader legt hier lediglich die Texturkoordinaten pro Vertex fest und transformiert diesen entsprechend der ModelViewProjection-Matrix. Die eigentliche Arbeit leistet das Fragmentprogramm. Mit Hilfe der von CASCADA vordefinierten Funktionen können die Texturwerte der Nachbarschaft abgefragt und somit eine Gewichtung durch den gesetzten Filterkern vorgenommen werden. Abschließend werden die Werte aufsummiert und das Ergebnis als gl_FragColor ausgegeben. Da es sich hier um einen Offscreen-Renderpass handelt, also einen Rendervorgang, bei dem die Pixel, die den Fragmentshader verlassen, vorerst nicht dargestellt werden, muss für den Renderdurchlauf eine weitere Flat-3D-Textur als Render-Ziel angegeben werden. Diese Textur entspricht nach der Ausführung des Passes dem gauss-geglätteten Volumen, welches im VolumenManager registriert wird und zur weiteren Verarbeitung beziehungsweise Visualisierung zur Verfügung steht.

6.6 Berechnungen - DtiComputationSequence

Im nächsten Schritt folgen die notwendigen Berechnungen. Auf Basis der zur Verfügung stehenden Daten wird zwischen den erforderlichen mathematischen Operationen differenziert. Bei Vorlage eines DWI-Datensatzes

mit mehr als sechs Gradientenrichtungen muss eine Reduzierung erfolgen, um die sechs unabhängigen Parameter des Diffusionstensors zu bestimmen. Liegen hingegen DTI-Daten vor, bedeutet das, dass die Daten die nötigen Tensorparameter D_{xx},D_{yy},D_{zz},D_{xy},D_{xz} und D_{yz} mitliefern. So entfällt der vorhin erwähnte Schritt. In beiden Fällen wird aber die Analyse der Eigenvektoren und Eigenwerte durchgeführt, da diese Grundvoraussetzung für die Visualierung des Verlaufs der Nervenfaserbündel ist. Aufgrund der Komplexität und des damit verbundenen zeitlichen Aufwands wurde auf eine GPGPU-Implemtierung zur Schätzung der Tensorparameter und der Eigenanalysis verzichtet. Die Berechnungen erfolgen mit Hilfe der GNU Scientific Library auf der CPU.

6.6.1 GNU Scientific Library (GSL)

Die *GNU Scientific Library* ist eine numerische Bibliothek für C und C++. Die GSL steht als freie Software unter der GNU General Public License zur Verfügung. Sie bietet eine Vielzahl an mathematischen Routinen für numerische Berechnungen [GNU]. Unter anderem sind die Singulärwertzerlegung auf Basis der kleinsten Quadrate und die Eigenanalysis für die Applikation von Bedeutung.

6.6.2 DtiComputationSequence

Die DtiComputationSequence kontrolliert vorerst, in Abhängigkeit von den existierenden Volumen, ob vor der Eigenanalysis eine Schätzung der Tensorwerte nötig ist. Sind die Diffusions-Tensor-Volumen nicht vorhanden, werden diese aus den DWI-Volumen errechnet.
Zur Schätzung des Tensors werden, neben den DWI Daten, auch das B0-gewichtete Volumen und die Gradientenrichtungen verwendet. Entsprechend dem Gleichungssystem (4) werden die Vektoren und Matrizen für die Singulärwertzerlegung angelegt. Da die Gradientenmatrix für alle Berechnungen konstant bleibt kann diese extern der folgenden iterativen Berechnung einmalig angelegt werden. Pro Voxelkoordinate wird nun der Vektor auf der linken Seite des Gleichungssystems berechnet. Mit diesem und der Gradientenmatrix können jetzt mit Hilfe der Singulärwertzerlegung von GSL die sechs unabhängigen Tensorparameter bestimmt werden. Für die Tensorparameter werden sechs neue Volumen angelegt und im VolumeManager registriert.
Im Folgenden werden sowohl die Eigenvektoren und Eigenwerte, als auch die fraktionelle und gemittelte Anisotropie pro Tensor berechnet.

6.7 Visualisierung

Wie bereits angesprochen wurden die in Abschnitt 4 genannten Visualierungsmethoden in der Applikation behandelt. Hier wird nun genauer auf die verwendeten Techniken, Algorithmen und Implementationen eingegangen.

6.7.1 Farbwert-Kodierung

Die Farbwertkodierung wurde basierend auf dem Prinzip aus Abschnitt 4.1.2 implementiert. Da die Darstellung im Zusammenspiel mit der multiplanaren Visualsierung angezeigt werden soll, wurde die Implementierung als *OnscreenRenderpass* in das *SimpleVolumeRendering* integriert. Die verwendete Sequenz zur Darstellung der MPR-Ansicht wurde also um einen zusätzlichen Renderpass erweitert. Dieser verwendet zur abschließenden Definition der Pixelfarbe ein abgewandeltes Fragmentshaderprogramm. Der Fragmentshader besitzt Zugriff auf vier Flat3D-Texturen: die jeweiligen Volumen zur Abfrage der Eigenvektorkomponenten x,y und z für die dominante Diffusionsrichtung sowie das Volumen, das die Werte der fraktionellen Anisotropie beinhaltet. Logischerweise werden die zusammengehörigen Parameter alle über die gleichen Texturkoordinaten angefordert. Für das aktuelle Fragment wird der entsprechende RGB-Farbwert gezeichnet. Der Alpha-Kanal bleibt für volle Opazität konstant auf eins. Da die Eigenvektoren auf die Länge eins normiert sind, kann eine direkte Übertragung auf den Farbbereich stattfinden. Lediglich der Betrag der einzelnen Komponenten muss gebildet werden, um auch negative Vektoren darstellen zu können. Die einzelnen Komponenten werden zusätzlich mit dem FA-Wert gewichtet um isotrope Bereiche schwarz darzustellen:

```
gl_FragColor = vec4((vec3(abs(evecXVoxel),
                          abs(evecYVoxel),
                          abs(evecZVoxel))*faVoxel),1);
```

Als Resultat ergibt sich die, in Abbildung 5a gezeigte, Darstellung.

6.7.2 Tensor-Geometrie - DtiGlyphSequence

Zur Visualisierung möglichst vieler Tensorinformationen eignet sich vor allem die Darstellung der Tensor-Geometrie mit Hilfe der in Abschnitt 4.2 aufgezeigten Ellipsoide. Unter Zuhilfenahme des Geometryshaders wurde an dieser Stelle eine ähnliche Repräsentation realisiert. Statt dem Ellipsoid pro Voxel, wird hier aber ein Quader gezeichnet, der die die Diffusionsrichtungen und -wege durch die Seitenvektoren und -längen ausdrückt (siehe Abbildung 15). Dieser Teil der Anwendung, die *DtiGlyphSequence*, wurde als *GLRenderSequence* unter Unterstützung des *OnScreenRenderings* implementiert. Die Sequenz erstellt für jedes Voxel, das sich innerhalb der vom

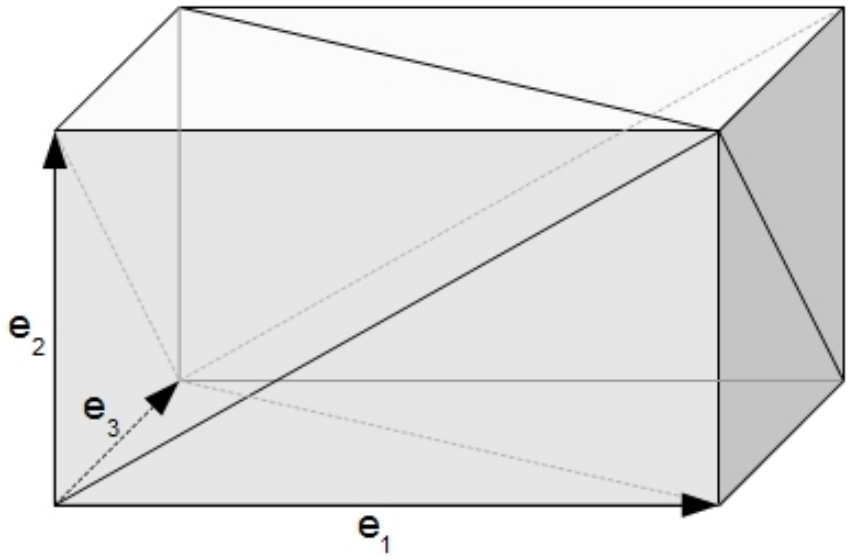

Abbildung 15: Aufbau der Quadergeometrie für die Glyphendarstellung.

Benutzer ausgewählten ROI befindet, einen Eingabevertex. Dieser wird vom Vertexshader einfach nur durchgereicht, eine Transformation findet also zu dem jetzigen Zeitpunkt noch nicht statt. Essentiell ist die Implementierung des Geometryshaderprogramms. Für die hier ankommenden Eckpunktdaten werden nun die notwendigen Texturzugriffe instruiert. Dazu benötigt der Shader die Parameter der drei vorberechneten Eigenvektoren und das FA-Volumen als uniform-Texturen. Zunächst wird die jeweilige Vertexkoordinate auf die entsprechende 2D-Texturkoordinate übertragen. Danach erfolgt die Abfrage des fraktionellen Anisotropiewerts. In Abhängigkeit von einem festgelegten Schwellwert wird hier entschieden, ob die Geometrie an dieser Stelle gezeichnet werden soll. Ist der Wert zu niederig (kleiner 0.2) wird der Vertex später verworfen. Andernfalls werden die weiteren Eigenvektor-Parameter über Texturzugriffe abgefragt. Die einzelnen Eigenvektor-Werte und der FA-Wert werden abschließend in einem *struct* EvecsAndFa abgelegt, um die Parameter in der weiteren Verarbeitung verwenden zu können. Zusätzlich wird in diesem Teil noch die Fragmentfarbe des Geometrieobjekts, wie in Abschnitt 8.1, definiert.
Nun stehen alle Parameter zur Verfügung, um die Quader-Geometrie pro Voxel zu erzeugen. Wie bereits erwähnt, wird die Geometrie nur in Bereichen erzeugt, in denen die Anisotropie hoch genug ist. Um die Geometrie der Voxelgröße anzupassen, werden die Vektoren entsprechend skaliert. Da im Geometryshader keine viereckigen Flächen generiert werden können,

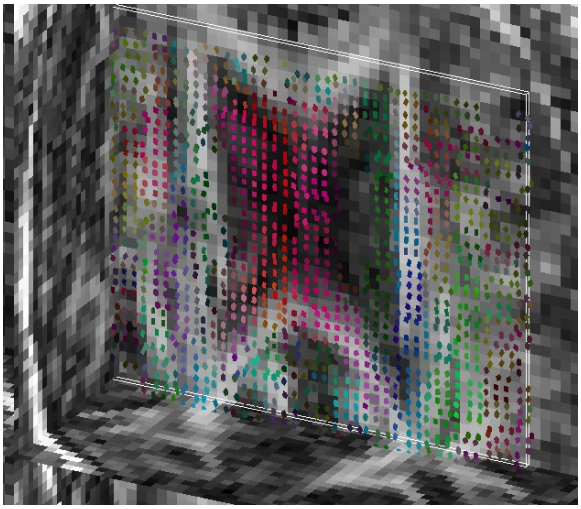

Abbildung 16: Screenshot der Applikation. Ausschnitt der Glyphendarstellung auf Basis der FA-Darstellung des Volumens im MPR.

sondern maximal Dreiecke, bestehen die Bausteine des Quaderes aus Dreiecken. So werden zwei Dreiecke zu einer Seite zusammengefasst. Auf Seite der Applikation wurde über den GeometrieManager die Anzahl der zu erzeugenden Eckpunkte deshalb vorher auf 36 festgelegt und der Ausgabetyp auf GL_TRIANGLE_STRIP gesetzt. Da die Eigenvektoren einen orthogonalen Raum aufspannen, werden aus Ursprung und Vektorlänge die fehlenden Eckpunkte bestimmt. Jeder neu erzeugte Vertex muss nun mit der *ModelViewProjection*-Matrix transformiert werden. Im Anschluss daran wird jeder Vertex mit *EmitVertex()* in die OpenGl-Pipeline übergeben. Da drei Vertices eine Teilfläche definieren wird nach drei emittierten Eckpunkten das Primitiv mit *EndPrimitive()* abgeschlossen. Schließlich weist der Fragmentshader dem Fragment noch die zuvor definierte Farbe zu.

6.7.3 Rekonstruktion der Nervenfasern - DtiFiberTrackingSequence

Der Schwerpunkt der Arbeit liegt auf der Rekonstruktion der Nervenfasern, dem *Fibertracking*. Die DtiFiberTrackingSequence umfasst die Implementierung. Der prinzipielle Aufbau entspricht der vorhin erläuterten DtiGlyphSequence. Auch hier übernimmt der Geometryshader die entscheidenden Aufgaben. Für die sich im Auswahlbereich befindenden Voxel werden aber jeweils zwei Geomtriepunkte erzeugt. Beide Punkte besitzen dieselbe Position, unterscheiden sich aber in der vierten Koordinate (1 oder -1). Diese bestimmt die Richtung des Trackings entlang oder entgegengesetzt des Haupt-Richtungsvektors. Der jeweilige Punkt erreicht den Geometryshader über ein *Passthrough* des Vertexshaders. Aufbauend auf diesem Vertex wird der FA-Wert abgefragt, um zu überprüfen ob die folgen-

den Verarbeitungsschritte notwendig sind. Liegt dieser Wert über dem vordefinierten Schwellwert, werden über die vorhandenen uniform-Texturen die Parameter des dominanten Eigenvektors abgerufen. Die Parameterinformationen werden in einem vec4-Vektor gesichert und die Fragmentfarbe wird nach bekanntem Prinzip definiert.
Hier beginnt der eigentlich Teil des Trackings. Zunächst müssen aber die Limitierungen des Geometryshaders beachtet werden. Da neben der Position, auch die Fragmentfarbe den Geometrieshader als 4D-Vektor verlässt, ist die Anzahl der neu zu erzeugenden Vertices auf 128 begrenzt. Da das Tracking in die beiden entgegengesetzten Richtungen abläuft, kann eine Faser nach einem Durchlauf aus maximal 256 Stützpunkten bestehen. Der Ausgabetyp ist dabei auf GL_LINE_STRIP gesetzt. Nachdem der Startpunkt in der Voxelmitte entsprechend transformiert und emittiert wurde, stehen für das folgende Tracking noch 127 weitere Schritte zur Verfügung. Entsprechend des FACT-Algorithmus wird nun der Schnittpunkt von Richtungsvektor und Voxelbegrenzung berechnet. Dieser Schnittpunkt dient als Stützstelle und zur Berechnung des nächsten Faserabschnitts im folgenden Iterationsschritt. Danach wird die Diffusionsrichtung des nächsten Voxels angefordert. Dazu wird der zuvor berechnete Schnittpunktvektor um einen minimalen Betrag verlängert und zur vorletzten Vertexposition addiert. Die resultierende Position liegt dann innerhalb des nächsten Voxels in Richtung des Faserverlaufs. Über diese Koordinaten finden die notwendigen Texturzugriffe statt, um die Richtung dieses Voxels zu bestimmen. Liegen die Informationen vor, findet ein Test statt, ob das Tracking der aktuellen Faser fortgesetzt wird. Bei Nichterfüllung der Schwellwerttests für die bereits erwähnten Kriterien (FA-Wert und Winkel) wird die Schleife abgebrochen und das Primitiv abgeschlossen. Andernfalls wird das Tracking mit dem Schnittpunkttest fortgesetzt.

6.8 Interaktion

Die Ausführung der Sequenzen wird vom Benutzer gesteuert. So ist es über die Benutzeroberfläche möglich, die einzelnen Schritte bis hin zur Visualisierung modular auszuführen. Nach dem Laden eines Datensatzes kann das Volumen nun vor den Tensorberechnungen geglättet werden. Nach der Bestimmung der Eigenwerte durch die Auswahl der DtiComputationSequence kann eine der drei Visualisierungsmethoden gewählt werden. Es ist ebenfalls möglich die Darstellungen zu kombinieren.
Weiterführende Änderung wurden in der Klasse GLWidget vorgenommen, welche für die Darstellung und die Interaktion der Szene zuständig ist. Zur Definition einer quadratischen Auswahlbox für die ROI mit Hilfe der Maus wurden die bestehenden Maus-Listener-Methoden leicht abgeändert und erweitert. Die dreidimensionale Auswahlbox ist über die minimale und maximale Position vollständig definiert. Beim Klicken mit der Maus und

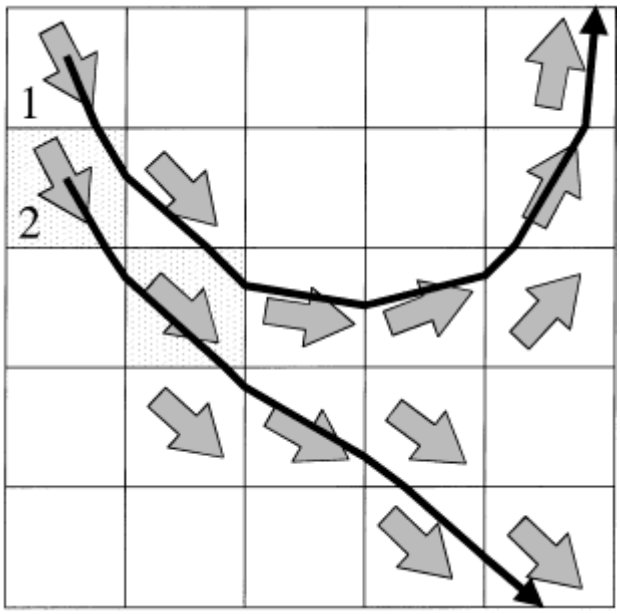

Abbildung 17: Zweidimensionale schematische Darstellung der Faserverlaufsrekonstruktion (FACT). Die grauen Pfeile repräsentieren die dominante Diffusionsrichtung, während die Schwarzen den Faserverlauf nachbilden. Das Tracking beginnt in der Voxelmitte, die Stützstellen liegen auf den Grenzen.

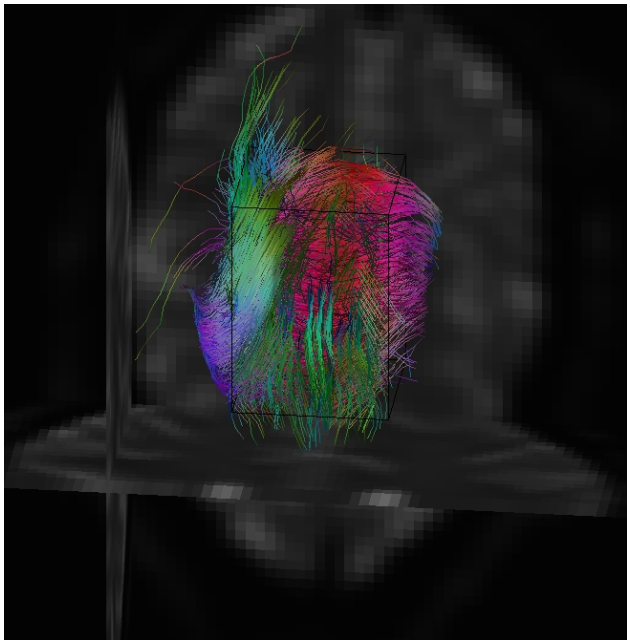

Abbildung 18: Screenshot der Applikation. Ausschnitt der Faserdarstellung.

gleichzeitigem Drücken der Strg-Taste an einer Stelle im GLWidget-Fenster wird der Ursprung der Auswahlbox gespeichert. Dazu werden die zweidimensionalen Mauskoordinaten zum einen in die entsprechenden Texturkoordinaten [0..1] des geladenen Volumens, zum anderen diese wiederum in die Voxelkoordinaten übertragen [1..N]. Beide Repräsentationen werden in Variablen festgehalten. Über das *MouseMoveEvent* wird die Größe der Box bestimmt und funktioniert nach demselben Prinzip. Zudem wird die aktuelle Auswahl direkt dargestellt. Solange die Applikation eine Änderung der Mausposition registriert wird die Auswahlbox neu gezeichnet. Die benötigte Geometrie wird im GeometryManager gelöscht und neu angelegt sowie den bestehenden Renderpasses hinzugefügt. Empfängt die Applikation im Nachhinein das MouseReleaseEvent wird auf eine existierende DtiFiberTrackingGPU-, beziehungsweise DtiGlyphGpu-Sequenz, getestet. Bei Erfolg wird der entsprechenden Sequenz die minimale und maximale ROI-Koordinate übergeben. Die Methode erzeugt die Eingabevertices in dem ausgewählten Bereich neu und fügt die Geometrie dem GeometryManager hinzu. Abschließend erfolgt ein Neuzeichnen der Szene mit der gewählten Tensorgeometrie innerhalb der *Region Of Interest*.

7 Fazit

Abschließend lässt sich festhalten, dass die Visualisierung von Tensordaten kein wissenschaftlich abgeschlossenes Themengebiet ist. Dies beruht nicht zuletzt auf der Modernität der Diffusions-Tensor-Bildgebung. Schon die unterschiedlichen Möglichkeiten der Bildaquisition und die damit einhergehenden Schwierigkeiten, wie hohe Signal-Rausch-Verhältnisse und Artefaktbildung, erfordern eine hohe Präzision bei der Vorverarbeitung der Bilddaten. Die Korrektur der Bilddaten auf einer genaueren Ebene als die implemtierte Rauschunterdrückung mit Hilfe der Gaussfilterung, hätte das Pensum dieser Arbeit jedoch überschritten. Größeres Augenmerk hingegen wurde auf die verschiedenen Visualisierungsformen mit Hilfe der Grafikhardware gerichtet. Über die Schwierigkeiten der Vorverarbeitung hinaus, kann jedoch trotz genauester Berechnung und Darstellung nicht unbedingt eine hundertprozentig korrekte Wiedergabe der tatsächlichen Gewebestruktur gewährleistet werden. Da die Gewebefasern eine weitaus geringere Größe als die Auflösung des Scanners besitzen, kann nur eine grobe Richtungsinformation pro Voxel repräsentiert werden. So können nur große Faserbündel veranschaulicht werden, mit der Bedingung jedoch, dass eine Kreuzung oder Aufspaltung der Faserstränge nicht berücksichtigt werden kann. Bisher ist die Diffusions-Tensor-Bildgebung aber die einzige Option solche Faserverläufe „in vivo“ zu untersuchen. Es existiert die Herausforderung die zugänglichen Informationen pro Voxel möglichst überschaubar und verständlich darzustellen. Die in dieser Arbeit aufge-

führten Methoden repräsentieren grundlegende Visualisierungstechniken. Die Implementation dieser Methoden auf der GPU erforderte bereits einigen Aufwand und kann durchaus weiter ausgebaut werden. Beispielsweise können die Fasern aufgrund der Limitierung des Geometryshaders nur bis zu einer bestimmten Länge dargestellt werden. Hier würde der Einsatz des *Transform Feedbacks* Abhilfe schaffen. Diese Erweiterung erlaubt es, Geometriedaten vor der Rasterisierung abzufangen und in VBOs abzulegen. Erst nach vollständigem Tracking werden die Stromlinien dann gezeichnet. Die Implementierung dieser Option konnte jedoch aus Zeitgründen nicht abgeschlossen werden. Andere Erweiterungsmöglichkeiten wären darüberhinaus der Ausbau auf Hyperstreamlines, die Darstellung ellipsoider Geometrie oder die Integration in eine DVR-Repräsentation des Volumens (bspw. Raycasting). Einen zusätzliche Performanzgewinn würde die Berechnung der Tensordaten (Tensorsschätzung mittels SVD und Eigenanalysis) auf der GPU mit sich bringen, was ursprünglich auch geplant war. Die Einarbeitungszeit in die Thematik und der Umgang mit den wenigen vorhandenen Datensätzen beanspruchten aber leider mehr Zeit als erwartet. Dennoch wurde die Zielsetzung weitestgehend erreicht. Eine interaktive Visualisierung auf Basis der GPU mit Hilfe von Vorberechnungen auf der CPU ist mit der entwickelten Applikation möglich.

Abbildungsverzeichnis

8 Literaturverzeichnis

Literatur

[Basser2002] Basser, P.J., Jones, D.K. 2002. Diffusion-tensor MRI: theory, experimental design and data analysis - a technical review. Bethesda, London.

[Bihan2001] Le Bihan, D., MD, PhD, u.a. 2001. Diffusion Tensor Imaging: Concepts and Applications, in: Journal of Magnetic Resonance Imaging, 13/2001, 534-546.

[CVIT2008] CVIT. Center for Visual Information Technology. Example Codes for Shader Model 4.0. `http://cvit.iiit.ac.in/index.php?page=resources`. Stand: 30. Dezember 2009

[Enders] Ender, F., u.a. Enhanced Visualization of Diffusion Tensor Date for Neurosurgery. Erlangen.

[Faber] Faber, C., Dr. Physik für Mediziner im 1. Fachsemester. Vorlesung. http://bio.physik.uni-wuerzburg.de/people/faber/VORLESUNG/NMR.pdf, Stand 30. Dezember 2009. Würzburg.

[GNU] GNU Operating System. GSL - GNU Scientific Library. http://www.gnu.org/software/gsl/, Stand 30. Dezember 2009.

[Köhn2009a] Köhn, A., Klein, J., Weiler, F., Peitgen, H.-O. 2009. A GPU-based Fiber Tracking Framework using Geometry Shaders. Fraunhofer MEVIS. Bremen.

[Köhn2009b] Köhn, A., Klein, J., Weiler, F. 2009. Real-time GPU-based fiber tracking using geometry shaders (Vortrag, Frauenhofer, MeVis, Institute for Medical Image Computing). Bremen.

[Kreher2005] Kreher, W.B., Tetzlaff Ralf. 2005. DTI-Bildgebung und Fibertracking. Präsentation. Freiburg.

[L3D2009] Lighthouse3D. GLSL Tutorial. http://www.lighthouse3d.com/opengl/glsl/. Stand: 3. Januar 2010

[Masutani2002] Masutani, Y., u.a. 2002. MR diffusion tensor imaging: recent advance and new techniques for diffusion tensor visualization, in: European Journal of Radiology, 46/2003, 53-66. Tokyo.

[Nrrd] Nrrd. Nearly Raw Raster Data. http://teem.sourceforge.net/nrrd/. Stand 30. Dezember 2009.

[NVGSEXT] Nrrd. Nearly Raw Raster Data. http://developer.download.nvidia.com/opengl/specs/GL_EXT_geometry_shader4.txt. Stand 30. Dezember 2009.

[Owens2005] Owens, J.D., u.a. 2005 A Survey of General-Purpose Computation on Graphics Hardware, in: EUROGRAPHICS 2005. 22-51.

[Peeters] Peeters, T.H.J.M, Vilanova, A., ter Haar Romeny, B.M. Visualization of DTI fibers using hair-rendering techniques. Eindhoven.

[Raspe2009] Raspe, M., Dr. 2009. GPU-assisted Diagnosis and Visualization of Medical Volume Data. Dissertation. Koblenz.

[Raspe2007] Raspe, M., Dr. 2007. Using a GPU-based Framework for Interactive Tone Mapping of Medical Volume Data. Koblenz.

[Schurade2006] Schurade, R. 2009. Visualisierung von Nervenfaserflächen im menschlichen Gehirn aus DTI Daten. Diplomarbeit. Leipzig.

[SCIUtah] Diffusion tensor MRI datasets. http://www.sci.utah.edu/~gk/DTI-data/#info, Stand: 30. Dezember 2009

[Unrath2006] Unrath, A. 2006. Richtungsabhängige Farbkodierung des menschlichen Thalamus mittels Diffusion Tensor Imaging. Dissertation zur Erlangung des Doktorgrades der Medizin der Medizinischen Fakultät der Universität Ulm. Ulm.

[vArb2008] von Arb, A.M. 2008. Entwicklung einer Beispielapplikation mit Hilfe von Geometrie-Shadern. Studienarbeit. Koblenz.

[Voss2005] Voss, H.U., u.a. 2005. Understanding functional connectivity of the brain: Diffusion tensor magnetic resonance imaging (Präsentation). New York, Canterbury.

[Wandell2006] Wandell, B., u.a. 2006. Diffusion tensor imaging and fiber tractography of human brain pathways. Stanford.

[Westin2002] Westin, C.-F., Maier, S.E. 2002. A Dual Tensor Basis Solution to the Stejskal-Tanner Equations for DT-MRI. Boston.

[W2009a] Wikipedia, Kernspinresonanzspektroskopie — Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/w/index.php?title=Kernspinresonanzspektroskopie&oldid=67771915, Stand 30. Dezember 2009

[W2009b] Wikipedia, Diffusions-Tensor-Bildgebung — Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/w/index.php?title=Diffusions-Tensor-Bildgebung&oldid=63003642, Stand 30. Dezember 2009

[W2009c] Wikipedia, Magnetresonanztomographie — Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/w/index.php?title=Magnetresonanztomographie&oldid=68594924, Stand 30. Dezember 2009

[Rong1999] Xue, R., u.a. 1999. In Vivo Three-Dimensional Reconstruction of Rat Brain Axonal Projections by Diffusion Tensor Imaging, in: Magnetic Resonance in Medicine, 42/1999, 1123-1127.