UNIVERSITÄT LEIPZIG
INSTITUT FÜR INFORMATIK (IfI)

Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Kombinierte Visualisierung von EEG- und Diffusions-MRT-Nervenfaser-Daten

Diplomarbeit

Leipzig, August 2010

vorgelegt von
Cornelius Müller
Studiengang Diplom Informatik

Betreuender Hochschullehrer: Prof. Dr. Gerik Scheuermann
Abteilung für Bild- und Signalverarbeitung

INHALTSVERZEICHNIS

1

EINLEITUNG

Bereits vor 80 Jahren wurde die *Elektroenzephalographie*, abgekürzt *EEG*, entwickelt [1]. Sie ist damit das älteste Verfahren, um am lebenden Organismus Aufnahmen der Gehirnaktivität zu machen. Sie basiert darauf, dass die Nervenzellen elektrische Felder erzeugen, welche außerhalb des Gehirns messbar sind. Diese Messung geschieht durch Elektroden, welche auf der Kopfoberfläche angebracht werden.
Neben der klassischen Darstellung als Zeitreihen der einzelnen Elektroden, dem *Elektroenzephalogramm*, wurden im Laufe der Zeit weitere Möglichkeiten entwickelt, die Daten auszuwerten. Eine relativ junge Methode ist die *Quelllokalisation*, also das Zurückrechnen, in welchem Bereich des Gehirns das gemessene Signal erzeugt wurde.
Eine andere Möglichkeit, Informationen über das lebende Gehirn aufzunehmen, ist die *Magnetresonanztomographie*, abgekürzt *MRT*. Sie arbeitet mit starken Magnetfeldern und erlaubt, dreidimensionale Bilder des Körpers zu erhalten. Ein spezielles Verfahren ist die *Diffusions-Magnetresonanztomographie*, welche die Richtung der Diffusion von Wassermolekülen misst. Dies erlaubt Rückschlüsse über die Nervenfaserrichtungen innerhalb des Gehirns. So können mit Hilfe der *Traktographie* Nervenfaserbündel rekonstruiert werden.

Die durch derartige Methoden aufgenommene Menge an Rohdaten wird immer größer, weshalb ein Mensch direkt daraus kaum Erkenntnisse gewinnen kann. Deshalb ist eine Vorverarbeitung und Verdichtung der Daten durch Computerprogramme nötig. Da der Mensch Bilder am leichtesten interpretieren und darin Besonderheiten erkennen kann, ist eine *Visualisierung* besonders gut geeignet. Folglich wurden für die Elektroenzephalographie und die Magnetresonanztomographie verschiedene Visualisierungstechniken entwickelt.

Allerdings stellt jede Messmethode andere Strukturen und Gegebenheiten besonders gut dar und hat Probleme mit anderen. So hat die Elektroenzephalographie eine hohe zeitliche Auflösung, die räumliche Auflösung aber ist auf den Abstand der Elektroden begrenzt, welcher üblicherweise einige Zentimeter beträgt. Die vorgestellten MRT-Methoden liefern dagegen Daten mit hoher räumlicher Auflösung, welche allerdings nur statisch sind. Um ein umfassenderes Bild zu erhalten, können die verschiedenen Methoden miteinander kombiniert werden. Dies erlaubt, die jeweiligen Stärken zu nutzen und dadurch Schlüsse zu ziehen, die durch die Auswertung der einzelnen Methoden allein nicht möglich wären.

Es gibt bislang allerdings keine Versuche, die relativ junge Rekonstruktion der Nervenfaserbündel mit der Elektroenzephalographie zu verbinden. Die reine Quelllokalisation von EEG-Daten erlaubt, den Bereich des Gehirns zu ermitteln, der wahrscheinlich für das gemessene Signal verantwortlich ist. Es ist aber nur in Ausnahmefällen so, dass lediglich ein Bereich an der Bewältigung einer Aufgabe beteiligt ist. Meist sind es mehrere, welche miteinander interagieren. Die dafür nötigen Verbindungen sind die Nervenfaserbündel, welche mit der Diffusions-Magnetresonanztomographie extrahiert werden können. Stellt man also neben den aus den EEG-Daten errechneten Quellen auch noch die Verbindungen dieser Quellen zu den anderen Bereichen des Gehirns dar, so erhält man einen umfassenderen Einblick in die Funktionsweise des Gehirns.

Ziel dieser Diplomarbeit ist die Entwicklung einer interaktiven Visualisierung von EEG-Daten und deren Quellen in Kombination mit Nervenfaserbündeldaten. Dazu soll als Softwaregrundlage das derzeit in Entwicklung befindliche *OpenWalnut* genutzt werden. Dabei handelt es sich um einen Softwarerahmen zur medizinischen Visualisierung mit Schwerpunkt auf die interaktive Darstellung von Gehirndaten. Die Darstellung und Selektion von Nervenfaserbündeln wurde darin bereits implementiert. Damit bietet es sich an, dieses System als Grundlage zu nehmen.
Im Rahmen dieser Arbeit soll OpenWalnut so erweitert werden, dass es mit EEG-Daten umgehen kann. Diese Daten sollen als anpassbares Elektroenzephalogramm und als Interpolation auf der dreidimensionalen Kopfoberfläche dargestellt werden können. Aus diesen Daten können Quellpositionen errechnet werden. Diese sollen zusammen mit den Nervenfaserbündeln, welche durch den Quellbereich verlaufen, angezeigt werden. Dadurch soll

ein nutzbares System für diesen neuen Ansatz entstehen, welches zeigen soll, dass die kombinierte Visualisierung sinnvoll ist.

Diese Arbeit wird durch Kapitel 2 fortgesetzt, in welchem die zugrundeliegenden Messmethoden beschrieben werden, also die Elektroenzephalographie und die Diffusions-Magnetresonanztomographie. Darauf folgt eine Übersicht über Arbeiten anderer Autoren, welche mit dieser in Zusammenhang stehen. In Kapitel 4 wird genauer auf die Softwaregrundlage OpenWalnut eingegangen, sowie die Erweiterungen beschrieben, die daran nötig waren. Das Design und die Implementierung der Datenhaltung und Visualisierung wird in Kapitel 5 erläutert. Im abschließenden Kapitel werden die Ergebnisse zusammengefasst und ein Ausblick auf mögliche weiterführende Forschung gegeben.

2

BESCHREIBUNG DER DATEN

In dieser Diplomarbeit wird die kombinierte Visualisierung zweier unterschiedlicher Arten von Hirndaten behandelt. In den folgenden Abschnitten werden diese einzeln beschrieben. Dabei wird mit der *Elektroenzephalographie* begonnen und danach die *Diffusions-Magnetresonanztomographie* beschrieben.

2.1 ELEKTROENZEPHALOGRAPHIE

Das menschliche Gehirn besteht aus circa 10^{10} *Nervenzellen*, welche untereinander durch zahlreiche *Nervenfasern* verbunden sind. Der Informationsaustausch zwischen diesen Zellen, welche auch *Neuronen* genannt werden, basiert auf elektrischen Ladungen. Falls genug Nervenzellen gleichzeitig aktiv sind, wird ein elektrisches Feld erzeugt, welches stark genug ist, dass es auch außerhalb des Gehirns gemessen werden kann und Rückschlüsse auf seine Arbeitsweise liefert.
Im Falle der *Elektroenzephalographie*, abgekürzt *EEG*, geschieht diese Messung durch *Elektroden*, welche auf der Kopfoberfläche befestigt werden. Da diese elektrische Spannung zwischen zwei Punkten messen, sind mindestens zwei Elektroden notwendig, um einen Wert zu erhalten. Bei einer EEG-Aufnahme werden mehrere Elektroden nach einem bestimmten System an verschiedene Positionen platziert, wie zum Beispiel dem verbreiteten *internationalen 10-20-System* [2].
Bei diesem wird die Entfernung zwischen dem *Nasion* am vorderen Teil des Schädels und dem *Inion* an der Rückseite gemessen. Diese Strecke wird aufgeteilt in einen 10-Prozent-Schritt, vier 20-Prozent-Schritten und schließlich wieder einmal 10 %. Daher kommt der Name *10-20-System*. Genauso wird mit der Strecke zwischen dem linken und dem rechten Ohr verfahren. Auf den Kreuzungspunkten des daraus entstehenden Gitters werden Elektroden laut Abbildung 2.1 platziert. Es entsteht eine Montage mit 21 Elektro-

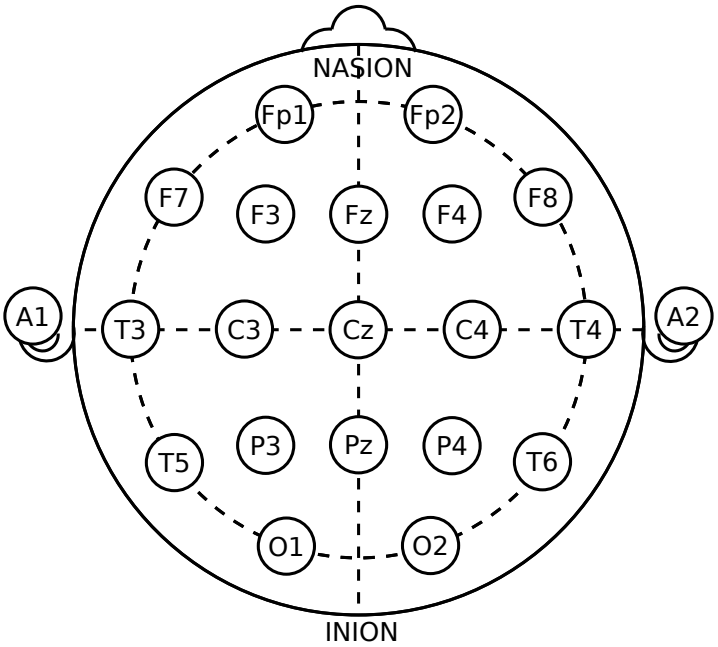

Abbildung 2.1: Elektrodenpositionen und -namen des internationalen 10-20-Systems. Dieses Bild wurde zur uneingeschränkten Nutzung freigegeben („public domain“).

den. Sind mehr Elektroden gewünscht, so können sie zwischen den Elektroden des 10-20-Systems angebracht werden, also zum Beispiel so, dass es mehr 10-Prozent-Schritte gibt.
Die Namen der Elektroden bestehen meist aus einem Großbuchstaben und einer Zahl. Der Großbuchstabe steht für den Teil der Großhirnrinde, über dem die Elektrode liegt: *Frontallappen* (F), *Temporallappen* (T), *Parietallappen* (P) und *Okzipitallappen* (O). Deren Lage ist auf Abbildung 2.2 zu sehen. C steht für den *Sulcus centralis* (lateinisch für *Zentralfurche*), welcher den Frontal- von dem Parietallappen trennt. Fp bedeutet *prefrontal*, also noch vor dem Frontallappen liegend. A1 und A2 sind zwei Elektroden, die an den Ohrläppchen befestigt sind und Referenzzwecken dienen. Die Zahl bezeichnet die Gehirnhälfte: Elektroden mit ungeraden Zahlen liegen auf der linken Hälfte, gerade Zahlen auf der rechten. Ein kleines z ist für Elektroden auf der Mittellinie („Zero“). Da die Namen der Kanäle auch nach Erweiterung mit mehr Elektroden gleich bleiben, lässt sich aus der Benennung die genaue Position ableiten.
Die gemessenen Spannungswerte können dann in Abhängigkeit von der Zeit graphisch dargestellt werden. Diesen Graph nennt man *Elektroenzephalogramm*. In der Regel werden dabei die Graphen der einzelnen Elektroden

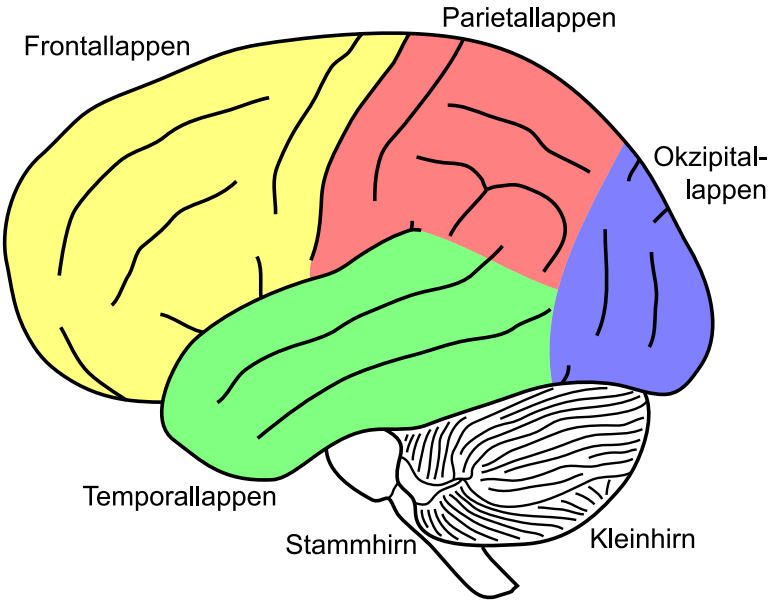

Abbildung 2.2: Seitenansicht eines menschlichen Gehirns mit farblich hervorgehobenen Großhirnlappen. Erstellt von NEUROtiker, Wikimedia Commons, lizenziert unter CreativeCommons by-sa-3.0, `http://creativecommons.org/licenses/by-sa/3.0/legalcode`.

übereinander angeordnet, so dass gleichzeitige Schwankungen in den verschiedenen Kanälen gut sichtbar sind. Ein Beispiel davon ist in Abbildung 2.3 zu sehen, welche eine Aufnahme von einem Kind mit Epilepsie zeigt. Es ist deutlich zu sehen, wie alle Kanäle ab der Hälfte des dargestellten Zeitbereichs gleichzeitig andere Wellen als zuvor zeigen.

2.1.1 *Quelllokalisation*

Da beim EEG das elektrische Feld nur auf der Kopfoberfläche gemessen wird, sind nicht direkt Aussagen über das elektrische Feld im Gehirn möglich, welches allerdings meist von Interesse ist. Deshalb wurden Verfahren entwickelt, aus den Spannungswerten an den einzelnen Elektroden Aussagen über den Bereich im Gehirn zu treffen. Dies ist umso zuverlässiger möglich, je mehr Elektroden benutzt werden.
Ist das elektrische Feld innerhalb des Gehirns bekannt, so kann mit Hilfe der *Maxwellschen Gleichungen* das Feld auf der Kopfoberfläche errechnet werden. Dies ist umso genauer möglich, je näher das Modell des Kopfes mit dem tatsächlichen Kopf übereinstimmt. Grundsätzlich wird in ei-

Abbildung 2.3: Elektroenzephalogramm von einem Kind mit Epilepsie. Erstellt von Der Lange, Wikimedia Commons, lizenziert unter CreativeCommons by-sa-2.0, `http://creativecommons.org/licenses/by-sa/2.0/legalcode`.

nem Kopfmodell zwischen drei Bereichen unterschieden: dem innerhalb des Schädels, den Schädel selber, und dem Bereich außerhalb; jeweils mit unterschiedlichen Leitereigenschaften. Diese Aufgabenstellung wird *Vorwärtsproblem* genannt.

Analog ist die Aufgabenstellung, aus dem elektrischen Feld auf der Kopfoberfläche das Feld innerhalb des Gehirns zu berechnen, das *inverse Problem*. Dieses ist allerdings *schlecht gestellt*, da verschiedene Felder innerhalb des Gehirns das gleiche Feld außerhalb erzeugen können.

Daher wird ein vereinfachtes Modell des Feldes innerhalb aufgestellt, welches sich durch wenige Parameter darstellen lässt. In der Regel wird ein *Dipolmodell* genutzt, welches davon ausgeht, dass das gesamte elektrische Feld durch ein oder mehrere Dipolquellen erzeugt wird, welche feste Positionen, Richtungen und Stärken besitzen. Dies stimmt mit der medizinischen Interpretation überein, nach welcher für bestimmte Beobachtungen bestimmte Areale des Gehirns verantwortlich sind.

Die Lösung des inversen Problems besteht daraus, eine Menge von Dipolen zu finden, welche ein elektrisches Feld auf der Kopfoberfläche erzeugen, welches möglichst stark mit dem gemessenem Feld übereinstimmt. Dabei ist der Unterschied zwischen den beiden Feldern die *Zielfunktion*, welche

durch eine geeignete *Optimierungsstrategie* minimiert wird. Dieses Festlegen der Dipolquellen wird dabei *Quelllokalisation* genannt.
Es gibt zahlreiche Verfahren, wie dies genau getan wird. Die Dissertation von Knösche [3] liefert einen Vergleich der Wichtigsten. Die Verfahren unterscheiden sich unter Anderem dadurch, wie die Dipole angeordnet sind: Einige fixieren eine große Menge an Dipolen auf einem gleichmäßigen Gitter überall innerhalb des Gehirns, und errechnen deren Richtung und Stärke. Andere Verfahren geben nur wenige Dipole aus, deren Positionen beliebig sein können.
Für diese Diplomarbeit sind die Inverslösungen mit wenigen Dipolen an freien Positionen besser geeignet, da diese Positionen direkt weitergenutzt werden können, um die dadurch verlaufenden Nervenfaserbündel anzuzeigen. Bei den Verfahren mit einem festen Gitter müssten die konkreten Positionen nachträglich bestimmt werden, welches zusätzlichen Aufwand bedeutet.

2.2 DIFFUSIONS-MAGNETRESONANZTOMOGRAPHIE

Das EEG eignet sich zur Aufnahme der zeitlichen Schwankungen der Aktivitäten des Gehirns. Zur statischen Aufnahme der Anatomie des Gehirns wird üblicherweise die *Magnetresonanztomographie* genutzt, abgekürzt *MRT*. Die Magnetresonanztomographie liefert hochauflösende dreidimensionale Bilder des gesamten Kopfes. Die Bilder bestehen aus einzelnen Bildelementen, den *Voxeln*, in welchen eine gemessene Intensität als Helligkeitswert gespeichert ist. Diese Intensitäten basieren auf den *Kernspineigenschaften* des Wasserstoffs, welche zum Beispiel von der Protonendichte und der Homogenität des Gewebes abhängen. Dabei ist MRT noninvasiv und nutzt keine potentiell schädliche Strahlung, sondern starke Magnetfelder, welche in der Regel keine negative Wirkung auf den menschlichen Organismus haben. Ein Schnitt aus einer MRT-Aufnahme ist in Abbildung 2.4 zu sehen.
Allerdings ist dies nicht ausreichend, um Strukturen innerhalb der *weißen Gehirnsubstanz* zu erkennen – sie erscheint als homogene Masse. Die weiße Substanz besteht hauptsächlich aus Nervenfasern, welche die Bereiche der *grauen Substanz* miteinander verbinden. Dabei verlaufen häufig viele Nervenfasern zusammen und bilden damit *Nervenfaserbündel*.

Zur Sichtbarmachung der Struktur innerhalb der weißen Substanz eignet sich die *Diffusions-Magnetresonanztomographie*, deren Grundlagen be-

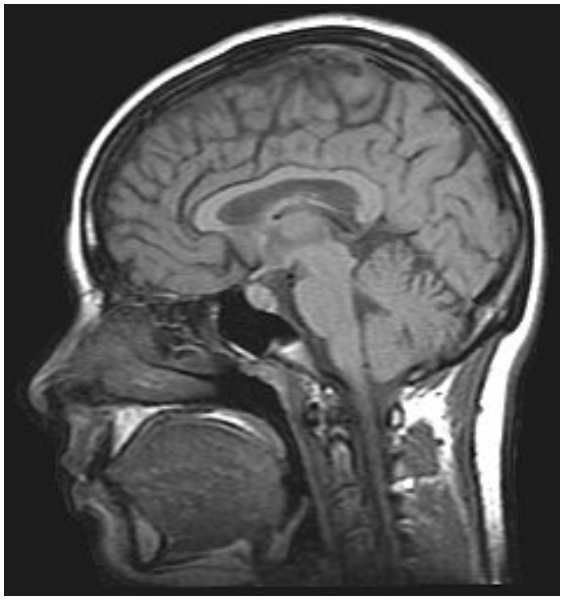

Abbildung 2.4: Schnitt aus einer Magnetresonanztomographieaufnahme, welches ein menschliches Gehirn von der Seite zeigt. Dieses Bild wurde von NASA zur uneingeschränkten Nutzung freigegeben („public domain“).

reits 1965 von Stejskal und Tanner [4] gelegt wurden. Sie misst für jeden Voxel die Stärke der Diffusion der Wassermoleküle in verschiedene Richtungen.
In frei beweglichem Wasser ist die Diffusion in alle Richtungen gleich stark, also *isotrop*. Dies ändert sich allerdings, wenn die Bewegung des Wassers eingeschränkt ist, wie es im Gehirn der Fall ist. Die Wassermoleküle bewegen sich besonders stark in die Richtungen, in welche Nervenfasern verlaufen, während die Beweglichkeit senkrecht zu den Fasern eingeschränkt ist. Die Diffusion ist also *anisotrop*. Dadurch lässt sich von der Diffusion des Wassers auf den Verlauf von Nervenfasern schließen.
So wird zum Beispiel die Hauptrichtung der Diffusion errechnet und graphisch dargestellt [5], da dies in der Regel mit der Richtung der meisten Nervenfasern übereinstimmt. Dazu können die verschiedenen Richtungen durch unterschiedliche Farben kodiert werden, wie in Abbildung 2.5 zu sehen. Dabei wird der Richtungsvektor auf [0, 1] skaliert und in seine x-, y- und z-Komponente zerlegt. Diese drei Werte werden den drei Farbkanälen

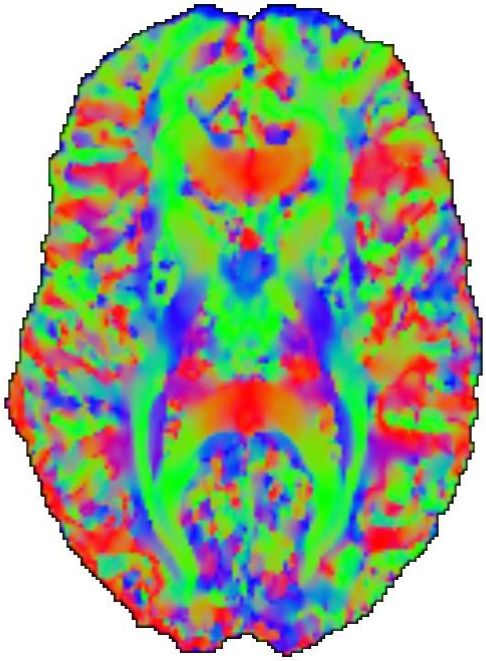

Abbildung 2.5: Farbige Darstellung der Hauptrichtung der Diffusion. Entnommen aus [6].

des *RGB*-Farbsystems[1] zugeordnet. Dadurch erscheint die x-Richtung rot, die y-Richtung grün und die z-Richtung blau. Ist die Hauptrichtung der Diffusion eine Mischung aus den Grundrichtungen, so ist auch die Farbe eine Mischung der Grundfarben.
Außerdem kann errechnet werden, wie stark die Hauptrichtung gegenüber den anderen Richtungen überwiegt, was meist als Wert zwischen 0 und 1 angegeben wird, der *fraktionellen Anisotropie*. Sie ist ein Maß für die Homogenität der Nervenfaserbündelrichtungen innerhalb der weißen Substanz; außerdem ist sie ein Indikator für einige Erkrankungen, wie zum Beispiel der Alzheimer-Krankheit.

Von besonderem Interesse für die Neurowissenschaft ist die Möglichkeit, den Verlauf von Nervenfaserbündeln zu rekonstruieren und als dreidimensionale Linien darzustellen. Dazu wird üblicherweise an einem bestimmten

1 Im *RGB*-Farbsystem entstehen Farben durch das Mischen der drei Grundfarben Rot, Grün und Blau. Eine Farbe ist dabei durch die Angabe der Werte für die einzelnen drei Farben eindeutig definiert.

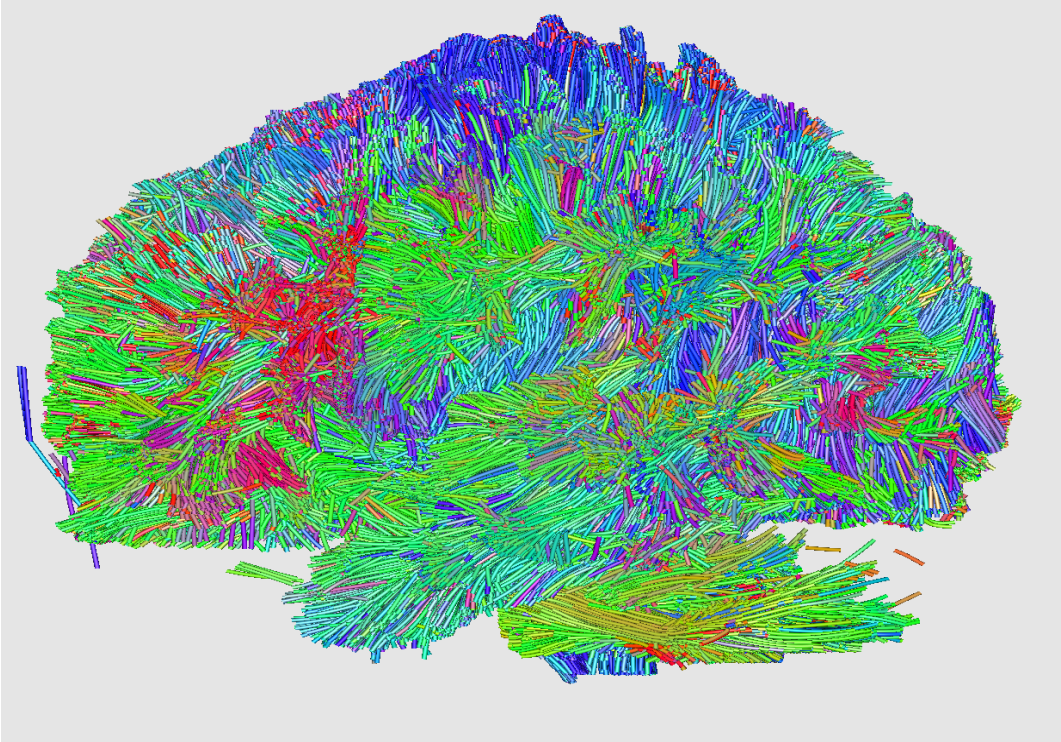

Abbildung 2.6: Alle errechneten Nervenfaserbündel eines menschlichen Gehirns. Dabei hängen die Farben der Linien von deren Richtung ab.

Punkt begonnen und dann der Hauptrichtung der Diffusion gefolgt. Dies wird bis zu einem bestimmten Abbruchkriterium weitergeführt, wie zum Beispiel, dass ein bestimmter Schwellwert der Anisotropie unterschritten wird oder der Winkel zwischen dem nächsten Abschnitt und dem bisherigen Faserbündel zu groß ist. Dieses Verfahren [7] wird *Traktographie* genannt.
Errechnet man die Nervenfaserbündel für jeden Voxel, so erhält man ein Bild der Nervenverbindungen des gesamten Gehirns, wie in Abbildung 2.6 zu sehen. Dies ist zwar rechnerisch aufwendig, lohnt sich aber dennoch, da diese Berechnung nur einmal durchgeführt werden muss. Später können genau die Faserbündel ausgewählt werden, welche gerade von Interesse sind. Abbildung 2.7 zeigt eine solche Auswahl. Beide Bilder wurden in OpenWalnut erstellt.
Eine solche Auswahl von interessanten Faserbündeln wird in der vorliegenden Diplomarbeit genutzt. Dabei wird durch die Ergebnisse von EEG-Quelllokalisation bestimmt, welche Faserbündel interessant sind.

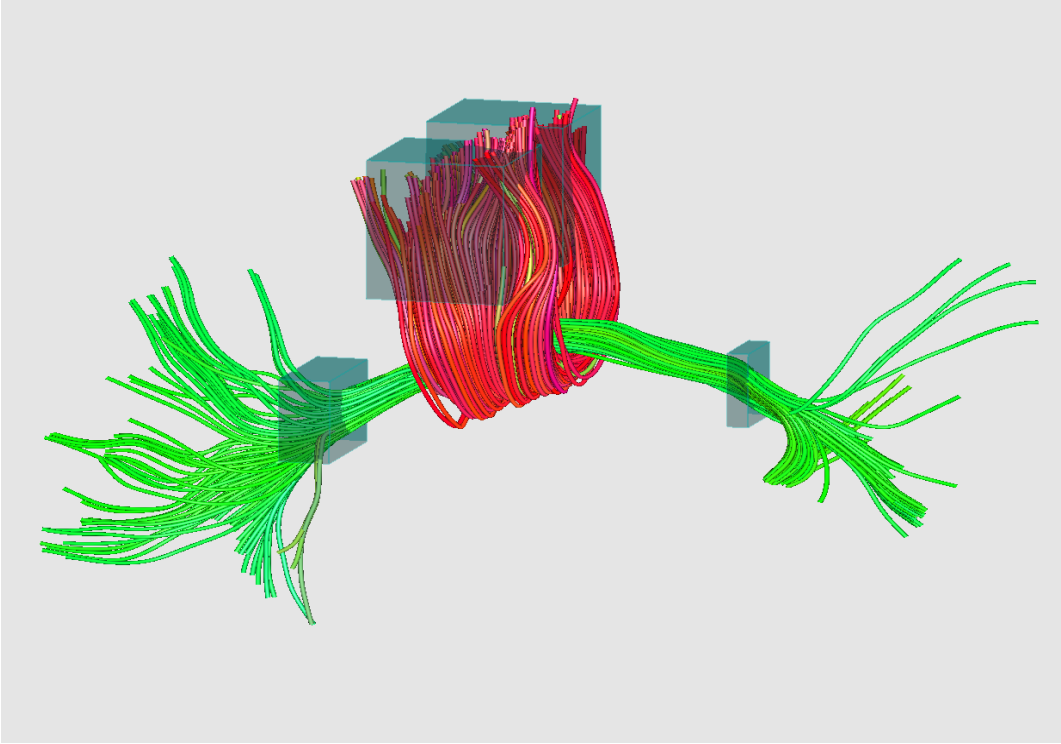

Abbildung 2.7: Eine Auswahl der Faserbündel, welche durch bestimmte Bereiche verlaufen, die als türkisfarbene Boxen zu sehen sind.

3

VERWANDTE ARBEITEN

In diesem Kapitel werden einige andere Arbeiten vorgestellt, welche mit der vorliegenden Arbeit zusammenhängen. Dabei wird im ersten Abschnitt auf die Selektion von Nervenfaserbündeln eingegangen, welche die Grundlage für die zu entwickelnde Visualisierung ist. Im zweiten Abschnitt werden bisherige Visualisierungsmöglichkeiten von EEG-Daten gezeigt, zum Teil auch in Kombination mit bildgebenden Verfahren der Medizin, um festzustellen, was in diesem Bereich bereits getan wurde.

3.1 NERVENFASERBÜNDELSELEKTION

Bei der Auswahl von Nervenfaserbündeln sind zunächst zwei Aufgaben wichtig: Zum einen die Frage, auf welche Weise die Faserbündel definiert werden, welche angezeigt werden sollen. Und dann das Problem, diese Auswahl so schnell zu machen, dass sie auf einen üblichen PC interaktiv benutzt werden kann.

Eine Forschergruppe der Stanford University [8, 9] hat ein System zur Faserbündelauswahl entwickelt, welches zur Definition der anzuzeigenden Faserbündel Bereiche in der Form von *achsenparallelen Quadern* nutzt: Es werden die Faserbündel dargestellt, welche durch einen so festgelegten Bereich verlaufen. Mehrere Quader können durch *boolesche Operatoren* kombiniert werden. So können zum Beispiel alle Faserbündel ausgewählt werden, welche durch alle der drei definierten Bereiche verlaufen, oder durch einen, aber nicht durch einen anderen. Außerdem können statistische Merkmale der Faserbündel wie Länge oder Gekrümmtheit zur Selektion genutzt werden. Das System ist auf einen üblichen PC lauffähig, allerdings ist es während der Selektion nicht besonders schnell, nur 3–8 Bilder pro Sekunde. Folglich ist damit nur die erste der beiden Aufgaben gelöst.

Blaas et al. [10] entwickelten ein ähnliches System, allerdings sind dort als Bereiche beliebige *konvexe Polyeder* möglich. Auch in ihrem System werden die Faserbündel angezeigt, welche durch die definierten Bereiche verlaufen und sie können ebenfalls durch boolesche Operatoren kombiniert werden. Technisch nutzen sie allerdings einen *kd-Baum*, eine Datenstruktur zur schnellen Auswahl von Punkten, welche in einem bestimmten Bereich liegen. Dadurch ist ihre Methode um mindestens eine Größenordnung schneller, so dass mit dem *DTII* genannten Programm interaktiv gearbeitet werden kann. Folglich sind damit die beiden grundlegenden Aufgaben gelöst. Über die Faserbündelselektion hinaus geht der *FiberNavigator*, welcher hauptsächlich von Ralph Schurade als Teil seiner Diplomarbeit [6] entwickelt wurde und als Open-Source veröffentlicht [11] ist. Er benutzt ebenfalls einen kd-Baum zur schnellen Auswahl von Faserbündeln, welche in einem definierten Bereich liegen, wobei ebenfalls mehrere Bereiche kombiniert werden können. Darüber hinaus beherrscht er den Umgang mit anderen Gehirndaten. So kann er zum Beispiel verschiedene MRT-Bilder als achsenparallele Schichten anzeigen oder daraus Isoflächen berechnen und einfärben. Ein ausgewählter Nervenfaserstrang kann dazu genutzt werden, eine Schnittfläche zu verformen, so dass sie dem Verlauf des Faserstrangs entspricht. So sind kombinierte Visualisierungen von verschiedenen aus Magnetresonanztomographie erstellten Daten möglich. EEG-Daten können aber nicht genutzt werden. Ein Screenshot ist in Abbildung 3.1 zu sehen.

Die Lösung der beiden Aufgaben, also die schnelle Auswahl von Nervenfaserbündeln, welche durch einen bestimmten Bereich verlaufen, ist wichtig für die zu entwickelnde Visualisierungsmethode. In dieser sollen interaktiv die Faserbündel angezeigt werden, welche durch eine aus EEG-Daten errechnete Quellposition verlaufen.
Der FiberNavigator ist der Vorgänger von *OpenWalnut*, einem Softwarerahmen für medizinische Visualisierung. Die zu entwickelnde Visualisierung nutzt OpenWalnut als Softwaregrundlage, wie im Kapitel 4 genauer beschrieben wird.

3.2 VISUALISIERUNG VON EEG-DATEN

Mit der schnellen Selektion von Nervenfaserbündeln ist die Grundlage für die neue Visualisierung gelegt. Im folgenden Abschnitt wird untersucht, welche anderen Methoden der Visualisierung von EEG-Daten bisher ent-

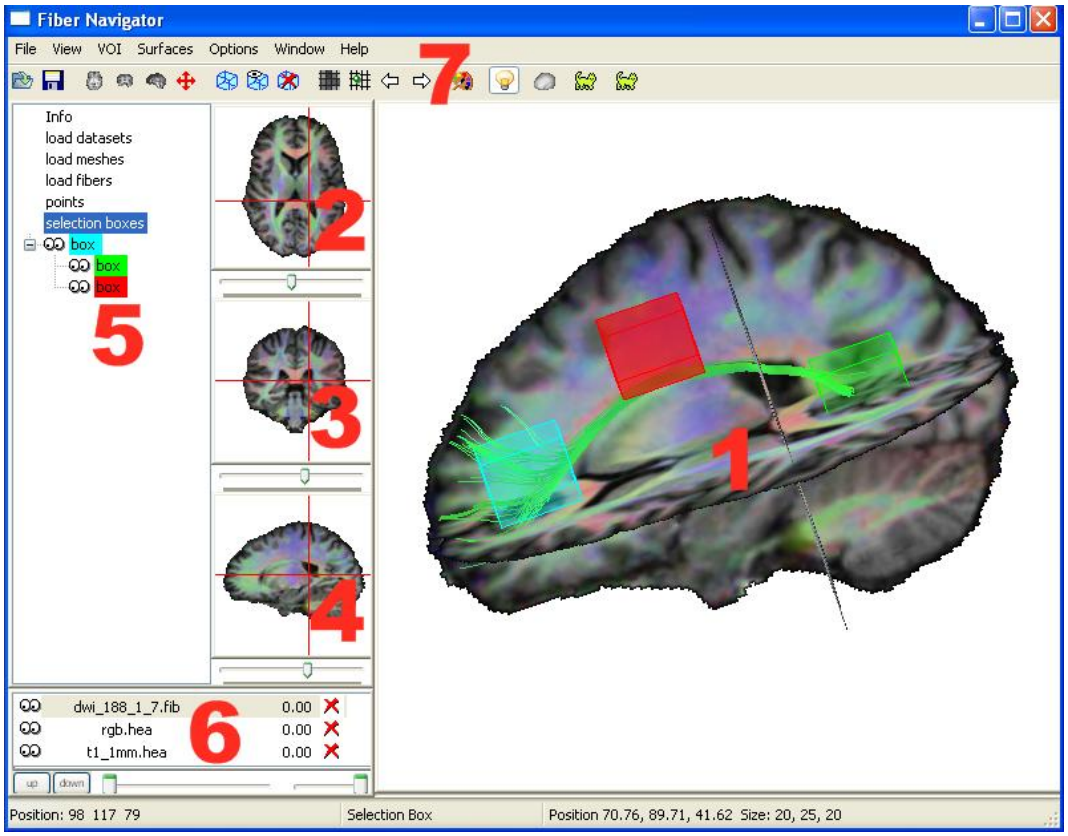

Abbildung 3.1: Screenshot des FiberNavigators unter Windows. Die Zahlen bezeichnen die Bereiche der Programmoberfläche: Hauptanzeige (1), Navigationsfenster (2–4), allgemeine Informationen sowie Auflistung der Auswahlboxen (5), Liste der geladenen Datensätze (6) und die Symbolleiste (7). Entnommen aus [6].

wickelt wurden. Dabei wird insbesondere darauf eingegangen, in wie weit zusätzliche Arten von Gehirndaten in Kombination mit den EEG-Daten genutzt werden.

So stellen Heinonen et al. 1999 [12] ein Programm vor, mit welchem es möglich ist, EEG-Daten auf der Kopfoberfläche darzustellen. Dabei wurde Wert auf die Geschwindigkeit gelegt, so dass es auch auf damaligen PCs interaktiv nutzbar war. Es können MRT-Aufnahmen genutzt werden, um das 3D-Modell des Kopfes zu erhalten, auf welchem die Daten interpoliert werden.
Jovanov et al. [13, 14] arbeiteten ebenfalls an einem Programm zur Visualisierung von EEG-Daten einschließlich deren Interpolation auf der Kopfoberfläche. Es ist nicht möglich, das Kopfmodell aus MRT-Daten erstellen zu lassen. Die Neuerung lag dagegen darin, dass die Benutzeroberfläche über *VRML*[2] realisiert wurde, wodurch die Arbeit über das Internet erledigt werden kann, ohne an einen bestimmten Ort gebunden zu sein. Außerdem beherrscht die Software neben der Visualisierung auch eine *Sonifikation*, also eine Umsetzung der Daten in Klänge.
Samardzic, Jovanov und Starcevic, welche auch an dem eben beschriebenen Artikel beteiligt waren, stellen in einer 2000 veröffentlichten Arbeit [15] zwei Programme vor, welche sich auf die Visualisierung der EEG-Daten auf der Kopfoberfläche spezialisieren. Beide Programme nutzen ein Kopfmodell aus dem *3DStudio*-Programm, also ohne MRT-Datengrundlage. Der Schwerpunkt wurde auf die schnelle Echtzeitdarstellung gelegt, so dass auch auf den damaligen PCs eine interaktive Nutzung möglich war.
EEGLAB [16] ist ein Open-Source-Programm für die plattformübergreifende *Matlab*-Umgebung, welches sehr viele Funktionen für den Umgang mit EEG-Daten vereint. Dazu gehört auch Informationsvisualisierung verschiedener Graphen sowie der Kopfoberfläche als 2D-Aufsicht. Kombinationen mit bildgebenden Verfahren der Medizin sind dabei nicht implementiert; der Schwerpunkt liegt bei mathematischen Analysen der Daten.
Ebenfalls Open-Source und in Matlab lauffähig ist *Brainstorm* [17]. Brainstorm konzentriert sich allerdings auf Visualisierung und Quelllokalisation, dabei werden auch MRT-Daten integriert. Es beherrscht verschiedene Verfahren für das Vorwärtsproblem und die Inversrechnung. Die Ergebnisse der Inversrechnung können farbcodiert auf MRT-Schnittebenen angezeigt

2 *VRML* ist die Abkürzung für *Virtual Reality Modeling Language* und bezeichnet eine Beschreibungssprache, welche dazu dient, 3D-Szenen über das Internet darzustellen.

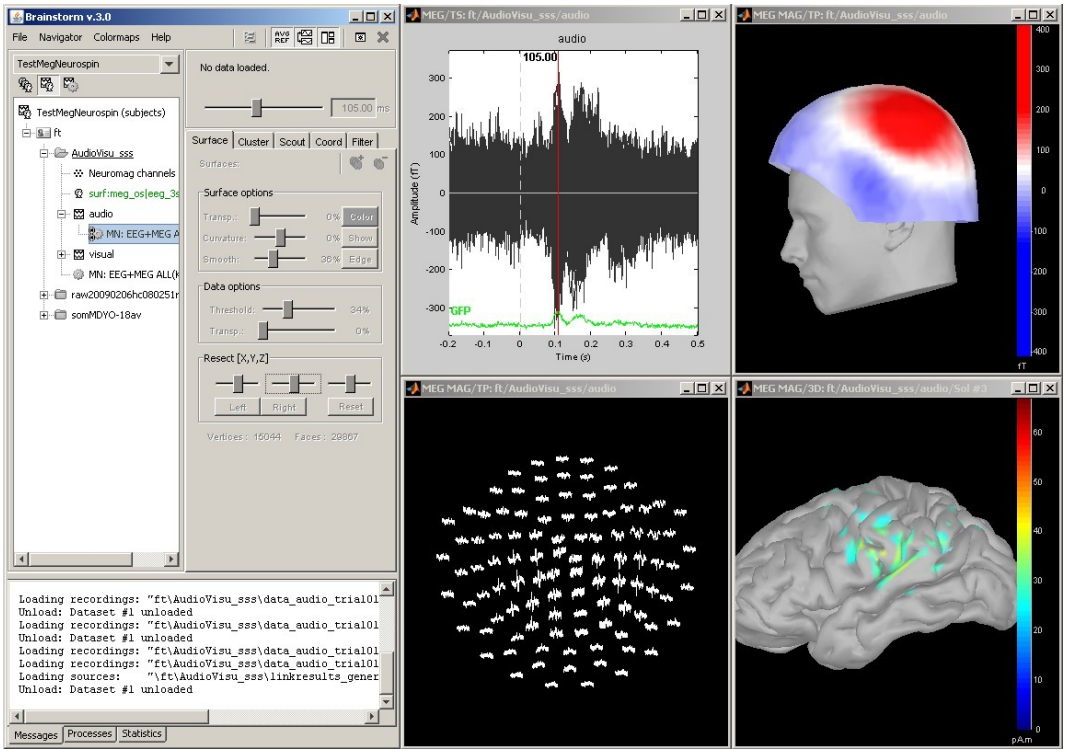

Abbildung 3.2: Screenshot von Brainstorm. Entnommen von [18].

werden oder auf Oberflächen gerendert werden, welche aus MRT-Daten extrahiert wurden, wie in der Abbildung 3.2 unten rechts zu sehen ist. Eine Kombination mit Diffusions-MRT oder Nervenfaserbündeldaten wie in der vorliegenden Arbeit findet allerdings nicht statt.
Wieder mehr Wert auf die mathematische Analyse legt die Arbeit von Kulish et al. [19]. Dort wird eine neue Technik der *nichtlinearen Spektralanalyse* vorgestellt, welche sich für die ebenfalls nichtlinear verhaltenden Prozesse im Gehirn besser eigenen soll als die üblicherweise genutzte Spektralanalyse durch eine lineare Fouriertransformation. Diese neue Analyse wird für eine Volumenvisualisierung der Gehirnaktivität genutzt. Dabei wird die Aktivität in dem Bereich des Kopfes interpoliert und die aktiven Stellen durch dreidimensionale transparente Tropfen angezeigt, den sogenannten „Blobs“. Diese sind in Abbildung 3.3 zu sehen. Das dabei genutzte Kopfmodell hat allerdings keine MRT-Grundlage.

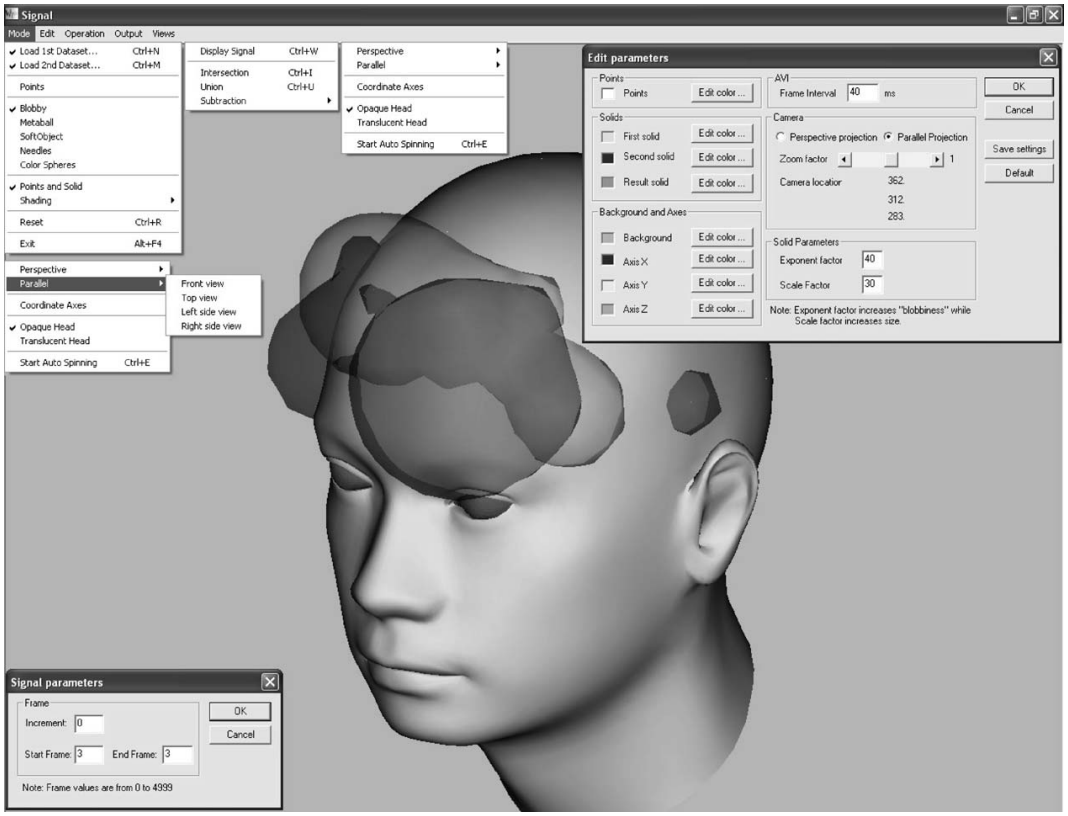

Abbildung 3.3: Volumenvisualisierung der Gehirnaktivität. Entnommen aus [19].

Die Kombination von EEG-Daten mit *SPECT*-Bildern[3] behandelt eine Arbeit von Goszczyńska et al. [20]. Darin werden die dazu nötigen Grundlagen geschaffen, indem die Elektroden in den SPECT-Bildern sichtbar gemacht werden. Dies ist nötig, um deren Positionen zu ermitteln und damit EEG und SPECT zu kombinieren.
Mit der Visualisierung von reinen EEG-Daten beschäftigt sich ten Caat [21]. Dabei entwickelte er die *tiled parallel coordinate map*, eine Möglichkeit, mehrere Graphen von parallelen Koordinaten anzuordnen; sowie *functional unit maps*, welche funktional zusammengehörige Elektroden erkennt und anzeigt. Keines der beiden Verfahren nutzt die räumliche Anordnung der Elektroden oder bildgebende Verfahren; es handelt sich um reine Informationsvisualisierung.

Wie zu sehen ist, werden die Möglichkeiten der Darstellung von EEG-Daten immer vielfältiger und durch die fortführende Verbesserung der Computerleistung sind auch anspruchsvolle dreidimensionale Visualisierungen möglich. Durch die Kombination mit anderen medizinischen Daten ist ein tieferer Einblick in die Funktionsweise des Gehirns möglich.
Bisher gibt es allerdings keine Arbeiten, welche EEG mit Nervenfaserbündeldaten kombinieren. Diese vielversprechende Idee wurde im Rahmen der vorliegenden Diplomarbeit zum ersten Mal implementiert und vorgestellt.

3 *SPECT* steht für *Single Photon Emission Computed Tomography* und ist ein bildgebendes Verfahren, welches die Verteilung eines Radiopharmakons im Körper zeigt. Dies erlaubt Rückschlüsse über die Funktion des untersuchten Organs.

4

OPENWALNUT

4.1 ZIEL

OpenWalnut ist ein Projekt, welches einen Softwarerahmen für die medizinische Visualisierung schaffen will. Daran beteiligt ist die *Abteilung für Bild- und Signalverarbeitung* [22] der *Universität Leipzig* und die Gruppe *Kortikale Netzwerke und Kognitive Funktionen* [23] des *Max-Planck-Institut für Kognitions- und Neurowissenschaften*.
Die Idee ist es, eine Basissoftware zu schaffen, welche grundlegende Funktionalität zur Verfügung stellt, die von vielen Visualisierungen wiederverwendet werden kann. Darunter fallen zum Beispiel eine grafische Benutzeroberfläche, 3D-Engine, häufig genutzte Datenstrukturen und das Laden und Speichern von Daten. Neue Algorithmen können als eigenständige Module geschrieben werden, ohne derartige grundlegende Funktionalitäten neu entwickeln zu müssen.
Die Software wird dabei als Open-Source-Projekt entwickelt und unter einer öffentlichen Website [24] zur Verfügung gestellt. Damit ist es möglich, dass sie nicht nur der Universität Leipzig und dem Max-Planck-Institut als Basis zur Entwicklung und dem Ausprobieren neuer Algorithmen dient, sondern auch anderen Gruppen, welche medizinische Visualisierungen erstellen wollen.
Zunächst soll OpenWalnut den *FiberNavigator* ersetzen, welcher in Abschnitt 3.1 vorgestellt wurde. Folglich unterstützt OpenWalnut genauso wie der FiberNavigator den Umgang mit Hirndaten. Es kann MRT-Aufnahmen laden und in Form von Schnittebenen, Isoflächen oder als Direct Volume Rendering darstellen. Vorberechnete Nervenfaserdaten können als Linien oder Röhren angezeigt, interaktiv ausgewählt und weiterverarbeitet werden.
Daraus ist der Name *OpenWalnut* entstanden, welcher auf die ähnliche Form des menschlichen Gehirns und einer geöffneten Walnuss (englisch

Abbildung 4.1: Logo von OpenWalnut. Entnommen von [24].

„walnut“) anspielt. Ein Bild des Gehirns ist daher auch Bestandteil des Logos, welches in Abbildung 4.1 zu sehen ist.
OpenWalnut wird für Linux, Windows und für Mac OS X entwickelt. Dabei wird als Programmiersprache das objektorientierte C++ verwendet, da es systemnah und damit schneller ist als die meisten anderen modernen Sprachen. Dies ist essentiell für Visualisierungen mit ihren zum Teil sehr aufwendigen 3D-Grafiken.

4.2 Programmoberfläche

Die Programmoberfläche von OpenWalnut ist in Abbildung 4.2 zu sehen. Der Screenshot wurde unter Windows erstellt und stellt drei Arten von Gehirndaten dar: Im vorderen Gehirnbereich wurden einige Nervenfaserbündel ausgewählt. Mithilfe von MRT-Aufnahmen wird die Anatomie durch Direct Volume Rendering dargestellt. Und drei achsenparallele Schnittebenen zeigen durch eine Farbkarte die fraktionelle Anisotropie an, welche in Abschnitt 2.2 beschrieben wurde.
Den Großteil des Fensters nimmt dabei die *Hauptanzeige* ein, welcher die genannten Daten dreidimensional darstellt. Links davon sind drei *Navigationsfenster*, welche jeweils einen Schnitt in einer Hauptrichtung anzeigen. Die Position des Schnittes lässt sich durch einen Schieberegler verschieben. Jedes Navigationsfenster ist ein eigenes *Dock Widget* – eine spezielle Anzeigefläche, welche der Endbenutzer in verschiedene Bereiche des Hauptfenster verschieben kann. Außerdem ist es auch möglich, es als eigenständiges *Floating Window* anzuzeigen, unabhängig vom Hauptfenster. Ebenfalls als Dock Widget realisiert wurde der *Dataset Browser* auf der rechten Seite. Er enthält eine Auflistung der *Module*, welche für die Visualisierung verantwortlich sind, einschließlich der Datenmodule, welche die Dateien laden. Die Module werden im nächsten Abschnitt erklärt. Auch die Bereiche, nach welchen die Faserbündel ausgewählt werden, sind hier verzeichnet. Ein sol-

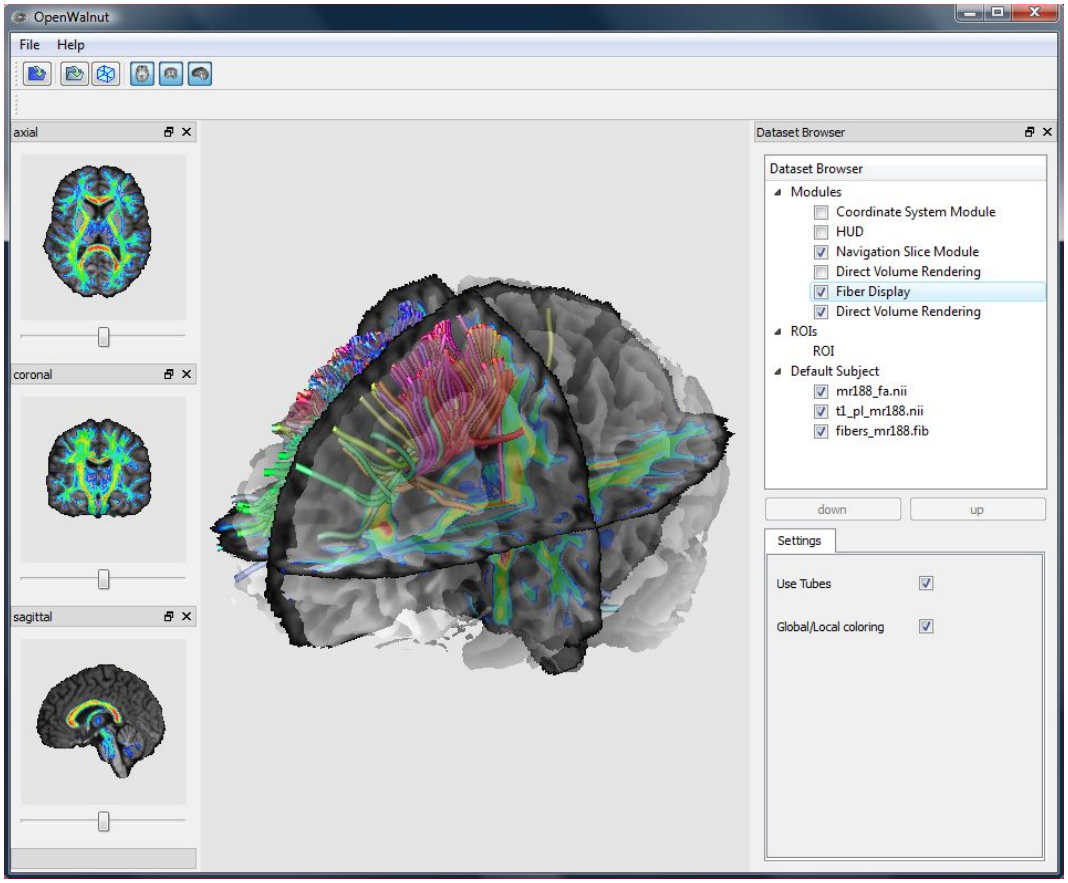

Abbildung 4.2: Screenshot der Programmoberfläche von OpenWalnut unter Windows. Entnommen von [24].

cher Bereich wird bei OpenWalnut *Region-of-Interest*, kurz *ROI* genannt. Im unteren Bereich des Dataset Browsers können die Einstellungen des derzeit ausgewählten Moduls geändert werden. Auf diese sogenannten *Properties* wird ebenfalls im nächsten Abschnitt näher eingegangen. Außerdem enthält das Programmfenster noch eine Menüleiste sowie Symbolleisten.

4.3 AUFBAU

OpenWalnut besteht aus sechs Teilen, den sogenannten *Libraries*: *common*, *dataHandler*, *graphicsEngine*, *gui*, *qt4gui* und *kernel* [24, CodeOrganization].
Common enthält Code, welcher von allen anderen Teilen häufig benutzt wird. Dazu gehören mathematische Grundstrukturen, Funktionen zur Verarbeitung von Zeichenketten und wiederbenutzbare Datenstrukturen.
Die *dataHandler-Library* enthält die grundlegenden Datensätze wie MRT-Bilder oder Nervenfaserdaten sowie die Möglichkeit, diese zu laden und zu speichern.
GraphicsEngine kümmert sich um die 3D-Darstellung. Dabei wird das Open-Source-Toolkit *OpenSceneGraph* [25] genutzt.
Die *gui-Library* besteht nur aus wenigen abstrakten Klassen, welche als Schnittstelle zu einer konkreten grafischen Benutzeroberfläche dienen. Die anderen Teile der Software, insbesondere die Module (siehe unten), können nur diese Schnittstellen nutzen, wenn sie Funktionen der GUI benutzen wollen. Dadurch ist die tatsächlich benutzte GUI leichter austauschbar.
Die derzeit einzige implementierte GUI basiert auf *Qt* [26] Version 4 und ist in der Library *qt4gui* enthalten.
Der letzte Teil, *kernel*, besteht aus den *Modulen*, welche die eigentlichen Visualisierungen enthalten, sowie der Funktionalität, diese untereinander und mit den anderen Teilen der Software zu verbinden.

Die Module folgen dabei dem Prinzip des Datenflusses, hier auch als „Hidden Visual Programming“ bezeichnet: Jedes Modul besitzt Eingänge und Ausgänge, welche mit anderen Modulen verbunden werden können und über welche Datenaustausch stattfindet. Da alle Verbindungen unidirektional sind, ist durch die Verbindungsstruktur der Module der Weg der Daten festgelegt. Diese Struktur kann als gerichteter Graph aufgefasst werden. Die Quellen dieses Graphen, also die Module ohne Eingänge, sind dabei üblicherweise Datenmodule, welche aufgenommene oder vorberechnete Daten aus Dateien laden und somit anderen Modulen zur Verfügung

stellen. Die Module mit Ein- und Ausgängen verarbeiten diese Daten weiter, zum Beispiel wenden sie bestimmte Filteroperationen darauf an. Die Senken, also die Module ohne Ausgänge, dienen in der Regel der Visualisierung der geladenen und eventuell bearbeiteten Daten. Das im Rahmen dieser Diplomarbeit entstandene *EEG View Modul* ist ein solches Modul ohne Ausgang.
Zusätzlich zu den Daten an den Eingängen hängen die Module auch noch von den *Properties* ab, welche bereits bei der Beschreibung der Programmoberfläche erwähnt wurden. Durch sie hat man die Möglichkeit, zusätzliche Parameter anzugeben. Diese können zum Beispiel den Dateinamen der zu ladenden Datei enthalten, die Größe eines Filters oder verschiedene Darstellungseinstellungen.
Sie sind über die grafische Benutzeroberfläche im Dataset Browser zu sehen und änderbar. Ein Modul kann auf seine Properties wie auf Variablen zugreifen. Neben dem Auslesen des aktuellen Werts und Setzen eines neuen kann auch abgefragt werden, ob der Wert geändert wurde und das Modul kann über Änderungen benachrichtigt werden.
Durch das Konzept von eigenständigen Modulen, welche nur über wenige, genau definierte Schnittstellen miteinander agieren, lassen sich leicht Erweiterungen programmieren. Jedes Modul hat eine bestimmte Aufgabe, welche sich im Idealfall unabhängig von den anderen Modulen testen lässt. Durch die Zusammenarbeit von mehreren Modulen können kompliziertere Probleme gelöst werden, welche durch den Austausch von einzelnen Modulen oder Neukombination verändert werden können.

4.4 ERWEITERUNGEN

Für die Realisierung des Visualisierungsmoduls „EEG View“ (siehe Abschnitt 5.2) waren einige Erweiterungen des OpenWalnut-Softwarerahmens nötig. Diese wurden als Teil dieser Diplomarbeit implementiert, sind aber auch von anderen Modulen nutzbar.

4.4.1 *Custom Widgets*

Die größte dieser Erweiterungen sind die sogenannten *Custom Widgets*. Diese werden als Anzeigefläche des Elektroenzephalogramms benötigt.
Das Problem ergibt sich daraus, dass OpenWalnut ursprünglich nur genau die vier Anzeigefenster enthielt, welche im Screenshot von oben (Abbil-

dung 4.2) zu sehen sind: die Hauptanzeige und die drei Navigationsfenster für die drei achsenparallelen Schnittebenen. All diese Anzeigen stellen die gleiche 3D-Szene dar. Sie haben jeweils eine Kamera, welche unterschiedliche Transformationen vornimmt, um die 3D-Weltkoordinaten in Bildschirmkoordinaten umzurechnen. Die Position und andere Parameter der Kamera wie Zoomfaktor sind durch Mausbewegungen änderbar. Aber all dies ist für die Darstellung des Elektroenzephalogramms ungeeignet. Man benötigt ein weiteres Anzeigefenster, welches eine unabhängige Szene darstellt und in der direkt die 2D-Bildschirmkoordinaten benutzt werden können, ohne Manipulationen durch Mausbewegungen.
Für diese Anzeigefenster werden *Dock Widgets* benutzt, so wie bereits für die drei Navigationsfenster und den Dataset Browser. Die Eigenschaften der Dock Widgets, innerhalb des Hauptfensters frei verschiebbar zu sein oder auch unabhängig vom Hauptfenster als eigenständiges *Floating Window* angezeigt werden zu können, sind ebenfalls für die neuen Anzeigefenster gut geeignet. Diese Funktionalität wird von *Qt* zu Verfügung gestellt, so dass dafür kein weiterer Implementierungsaufwand anfällt. Da die neuen Anzeigefenster beliebigen Inhalt enthalten können, werden sie *Custom Widgets* genannt.
Es ist gewünscht, dass sie von einen Modul aus erstellt werden können. Die Module aber können keine der Qt-Funktionen direkt nutzen, da sie nur von der abstrakten *gui-Library* abhängen (siehe oben, Abschnitt 4.3 Aufbau). Folglich wurde ein System von Klassen erstellt, welches in Abbildung 4.3 zu sehen ist: Die konkrete Klasse *WQtCustomDockWidget* repräsentiert das Custom Widget und wurde von *QDockWidget* abgeleitet, der entsprechenden Qt-Klasse für Dock Widgets. Außerdem wurde *WCustomWidget* erstellt, eine abstrakte Oberklasse von *WQtCustomDockWidget*, welche Teil der *gui-Library* ist und damit von einem Modul aus genutzt werden kann, wie zum Beispiel dem EEG View Modul (*WMEEGView*).
Ebenfalls in der *gui-Library* befindet sich eine abstrakte Funktion, um ein neues Widget zu erstellen:

```
openCustomWidget(title, projectionMode, shutdownCondition)
```

Diese Funktion wird in der *qt4gui-Library* implementiert. Da sie aber in der Regel von einem Modul aufgerufen wird, läuft sie auch in dem Thread dieses Moduls. Die Konstruktion eines Dock Widget ist aber nur von einem Qt-Thread aus möglich. Deshalb muss die Aufgabe, ein neues Widget zu erstellen, weitergeleitet werden. Dies geschieht, indem ein spezielles Event

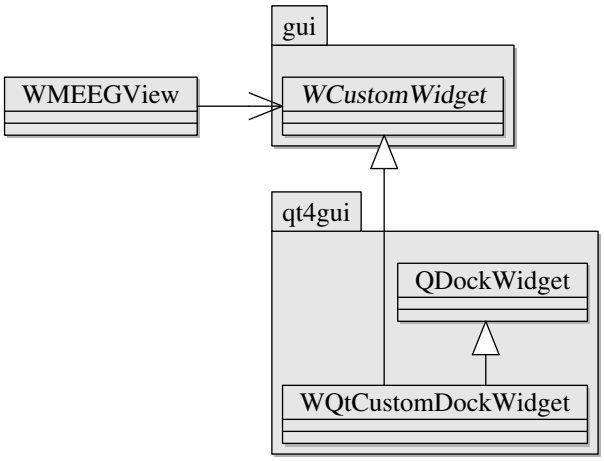

Abbildung 4.3: System der Klassen, welche ein Custom Widget repräsentieren.

an das Qt-Hauptfenster geschickt wird. Dieses erzeugt daraufhin das Widget. Ein Zeiger auf das neu erstellte Widget wird über einen in OpenWalnut implementierten Signal-Slot-Mechanismus an die *openCustomWidget*-Funktion weitergeleitet. Diese gibt ihn als Rückgabewert zurück an das Modul.
Die *openCustomWidget*-Funktion wartet, während das Hauptfenster das Dock Widget erstellt. Da sie in einem Modulthread läuft, ist in dieser Zeit das Modul untätig. Dies führt insbesondere dann zu einem Problem, wenn das Hauptfenster einen Fehler beim Erstellen des Widgets hat, so dass die *openCustomWidget*-Funktion und damit das Modul mit dem Warten nicht aufhört, also nicht mehr reagiert. Um dieses Problem einzudämmen, wird der *openCustomWidget*-Funktion die *shutdownCondition* des Moduls übergeben. Dadurch ist es möglich, dass nicht mehr gewartet wird, wenn die *shutdownCondition* ausgelöst wird, also wenn das Modul beendet werden soll.
Jedes Custom Widget besitzt einen eindeutigen Namen, welcher bei der Erstellung als *title*-Parameter angegeben werden muss. Über diesen Namen kann man später ein Zeiger zu dem Widget erhalten oder es wieder schließen. Dabei kann es auch vorkommen, dass mehrere Module das gleiche Widget benutzen. Deshalb wird gezählt, wie oft ein Widget benutzt wird und es wird erst dann entfernt, wenn es nicht mehr benötigt wird.
Außerdem hat jedes Custom Widget seine eigene Szene, so dass Grafikelemente unabhängig von anderen Anzeigen dargestellt werden können. Die Art der Kamera wird ebenfalls bei der Erstellung durch den *projectionMode*-Parameter festgelegt. Speziell für die Anzeige des Elektroenzephalo-

gramms wurde ein neuer Kameramodus ermöglicht. In diesem stimmen die Weltkoordinaten und die Bildschirmkoordinaten überein und diese Zuordnung lässt sich auch nicht durch Mausbewegungen ändern.

4.4.2 *Sonstige Erweiterungen*

Neben den Custom Widget wurden auch einige andere Erweiterungen für diese Arbeit implementiert, welche nicht Teil des *EEG View Moduls* sind.

Ein Problem war, dass *Regions-of-Interest* sich nicht von einem Modul heraus löschen ließen. Das *EEG View Modul* soll aber die Fähigkeit besitzen, Faserbündel in einem bestimmten Bereich auszuwählen. Dafür sind die kurz ROIs genannten Regions-of-Interest nötig, um den Bereich zu definieren. Diese müssen nicht nur erstellt werden können, sondern auch wieder gelöscht werden.
Dies funktionierte deshalb nicht, da die ROIs doppelt gespeichert werden: zum einen im *ROI-Manager* in der *kernel-Library* und zum anderen als Teil des *Dataset Browsers* in der *qt4gui-Library*. Der ROI-Manager erlaubt das Löschen von ROIs, aber auf den Dataset Browser kann man von Modulen heraus nicht direkt zugreifen, da sie nicht von *qt4gui* abhängen (siehe oben, Abschnitt 4.3 Aufbau). Das Löschen von ROIs war also nur möglich, wenn es vom Dataset Browser initiiert wurde (wie es üblicherweise über die GUI geschieht).
Also wurde eine Verbindung des Dataset Browser mit dem ROI-Manager implementiert. Möchte ein Modul eine ROI löschen, so ruft sie die *removeRoi*-Funktion des ROI-Managers auf. Dieser löscht daraufhin die ROI bei sich und benachrichtigt durch einen Signal-Slot-Mechanismus die Qt-GUI. Diese ist in der gleichen Library wie der Dataset Browser, so dass sie ihm ein spezielles Event schicken kann. Dadurch wird der Dataset Browser angewiesen, die ROI auch bei sich zu löschen, so dass die ROI in keiner Liste mehr gespeichert ist.
Durch diesen Mechanismus ist es einem Modul möglich, Regions-of-Interest zu erstellen und zu löschen.

Außerdem wurde ein kleines Hilfsmodul entwickelt, welches Nervenfaserdaten im 3D-Raum transformieren kann, das *Fiber Transform Modul*.
Das Problem war, dass die verschiedenen Daten, welche kombiniert dargestellt werden sollen, nicht unbedingt das selbe Koordinatensystem benut-

zen. So haben die gleichen Stellen des Kopfes nicht die gleichen Koordinaten. Folglich muss einer der beiden Datensätze so transformiert werden, dass die Stellen übereinstimmen.
Dazu wurde ein Modul implementiert, welches einen Nervenfaserdatensatz sowohl als Ein- als auch als Ausgang besitzt. Das Modul enthält eine Matrix und einen Vektor, womit alle Punkte jedes Nervenbündels des Eingangsdatensatzes multipliziert bzw. addiert werden, um sie affin zu transformieren. Der entstehende Datensatz wird an den Ausgang gelegt und kann wahlweise auch in einer Datei gespeichert werden, um später schneller zur Verfügung zu stehen.

5

DESIGN UND IMPLEMENTIERUNG

In diesem Kapitel wird das Design und die Implementierung der Programmteile beschrieben, welche direkt dafür verantwortlich sind, EEG-Daten zu visualisieren und den Zusammenhang zu den Nervenfaserdaten herzustellen.
Dazu wird im Abschnitt 5.1 erklärt, wie die EEG-Daten aus den entsprechenden Dateien geladen werden, wie sie im Programm repräsentiert werden und wie darauf zugegriffen werden kann. Im darauf folgenden Abschnitt wird das *EEG View Modul* beschrieben, welches die Visualisierung vornimmt. Dabei wird einzeln auf die drei wichtigsten Funktionen eingegangen: die Darstellung des Elektroenzephalogramms in 2D, die Darstellung der Kopfoberfläche in 3D und die Kombination dieser Daten mit den Nervenfaserdaten.

5.1 DATENHALTUNG

5.1.1 *Dateiformate*

Bei dem Laden von EEG-Daten muss beachtet werden, dass es dafür zahlreiche verschiedene Formate gibt. Das *Max-Planck-Institut für Kognitions- und Neurowissenschaften* arbeitet hauptsächlich mit den Programmen von *ANT* [27], welche ihr eigenes Format mit der Dateiendung *CNT*[4] nutzen [28, Abschnitt 7.3 Signal Data Files]. Deshalb ist es nötig, dass dieses Format gelesen werden kann.
Dafür wurde die Open-Source-Bibliothek *Libeep* [29] genutzt, welche von ANT veröffentlicht wurde, um anderen Programmen die Möglichkeit zu geben, ihre EEG-Formate zu lesen und zu schreiben.

4 *CNT* steht für „**con**tinuous EEG data file“.

Die Bibliothek ist in *C* geschrieben und kann deshalb auch von *C++* heraus ohne einen zusätzlichen Wrapper benutzt werden. Sie bietet Funktionen, um die unterstützen EEG-Dateien einzuladen, Metadaten zu erhalten und die Werte an den Elektroden zu beliebigen Zeitpunkten auszulesen. Damit ist es möglich, die EEG-Daten zu laden, ohne die Art der Speicherung genau zu kennen.

Für die Anzeige der Kopfoberfläche und die Quelllokalisation müssen außerdem die Positionen der Elektroden auf dem Kopf bekannt sein. Diese Daten werden üblicherweise nicht zusammen mit den Spannungsdaten gespeichert, sondern in einer separaten Datei, da häufig viele Messungen mit der gleichen Elektrodenkonfiguration durchgeführt werden.
Die Programme von ANT nutzen dazu ebenfalls ein eigenes Format mit der Dateiendung *ELC* [30, Electrode file format]. In diesem Format werden die Positionen zusammen mit den Namen der dazugehörigen Elektroden gespeichert, so dass eine Zuordnung zwischen beiden möglich ist. Die Daten werden als menschenlesbarer Text gespeichert, so dass man die ELC-Dateien vergleichsweise einfach ohne Hilfsmittel laden kann.

5.1.2 *Implementierung*

Eine Besonderheit bei EEG-Daten gegenüber den meisten anderen Hirndaten ist, dass sie so groß werden können, dass sie nicht vollständig in den Hauptspeicher geladen werden können. Folglich müssen Methoden entwickelt werden, welche die Datei geöffnet lassen und nur die Daten lesen, welche gerade benötigt werden. Dabei ist es wünschenswert, dass der Programmteil, der die Daten benutzt, nicht wissen muss, wie dieser Mechanismus funktioniert.
Dafür wurde die Datenhaltung auf *zwei Ebenen* aufgeteilt. Die *untere Ebene* übernimmt das direkte Lesen der Datei. Die *obere Ebene* speichert einen Teil der Daten, so dass später schneller darauf zugegriffen werden kann und stellt die Sicht auf die Daten für die verarbeitende Programmteile wie dem *EEG View Modul* bereit.

Untere Ebene

Die *untere Ebene* übernimmt die Aufgabe, verschiedene Dateiformate alle über die gleiche Schnittstelle lesbar zu machen. Dazu wurde die abstrakte Klasse *WPagerEEG* geschrieben, welche rein virtuelle get-Methoden für

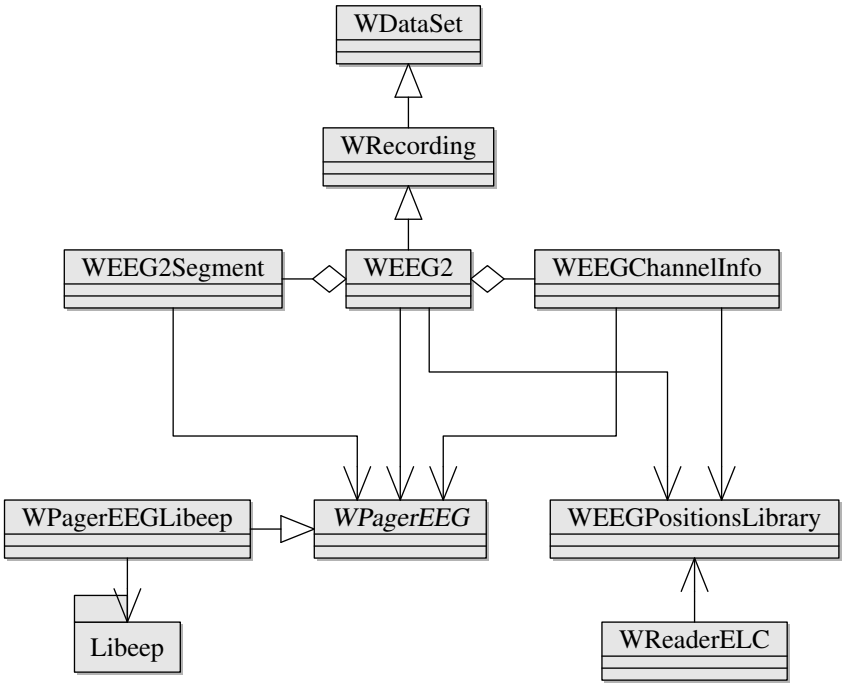

Abbildung 5.1: Beziehungen zwischen den an der Datenhaltung beteiligten Klassen.

alle auszulesenden Daten hat. Für jedes unterstützte Dateiformat existiert eine abgeleitete Klasse, welche die get-Methoden überschreibt, so dass sie die entsprechenden Daten aus den Dateien lesen. Im Moment gibt es nur eine abgeleitete Klasse, *WPagerEEGLibeep*, welche CNT-Dateien mit Hilfe der oben beschriebenen *Libeep*-Bibliothek lesen kann. Siehe Abbildung 5.1 für eine Übersicht der Beziehungen zwischen den an der Datenhaltung beteiligten Klassen.
Es ist angedacht, für weitere Formate weitere spezialisierte Klassen zu schreiben. Dies ist ohne Änderungen des restlichen Programms möglich, da niemals die spezialisierte Klasse direkt benutzt wird, sondern stets die abstrakte Basisklasse *WPagerEEG*.

Die Elektrodenpositionen werden zunächst unabhängig von den anderen Daten in der Klasse *WEEGPositionsLibrary* gespeichert. Diese Klasse ermöglicht die Abfrage der Position zu einem bestimmten Namen einer Elektrode. Der Zusammenhang mit den anderen geladenen Daten wird erst in der oberen Ebene hergestellt.
Die *WEEGPositionsLibrary* enthält dabei direkt die Daten. Da die Elektrodenpositionen nur sehr wenig Speicher verbrauchen, ist ein Nachladen-bei-

Bedarf unnötig. Die *WEEGPositionsLibrary* muss daher nicht die Datei kennen, aus der sie erzeugt wurde und ist damit unabhängig vom Dateiformat. Die Elektrodenpositionen können also genauso geladen werden, wie die restlichen Daten in OpenWalnut: Einem spezialisierten *WReader* wird in seinem Konstruktor der Dateiname der zu ladenden Datei übergeben. Die read-Methode gibt den Datensatz zurück. Für das oben beschriebene *ELC*-Format wurde der *WReaderELC* implementiert. Andere Dateiformate könnten mit einer anderen von *WReader* abgeleiteten Klasse geladen werden, welche ebenfalls ein *WEEGPositionsLibrary*-Objekt erzeugt.

Obere Ebene

Die *obere Ebene* ist der Teil der Datenhaltung, der von den verarbeitenden Programmteilen gesehen wird. Dabei wurde darauf geachtet, die Struktur der Daten auf die Struktur der Klassen abzubilden. Dies führte zu den drei Klassen *WEEG2*, *WEEG2Segment* und *WEEGChannelInfo* wie in Abbildung 5.1 zu sehen.
Die Grundklasse ist *WEEG2*. Sie ist von *WRecording* abgeleitet, welches eine Basisklasse für alle Zeitreihen-Datensätze ist, wie zum Beispiel auch *MEG*[5]. *WRecording* und damit auch *WEEG2* sind Spezialisierungen von *WDataSet* und können damit von einem OpenWalnut-Modul einem anderen Modul im Sinne des Datenflussnetzwerks übergeben werden (siehe Abschnitt 4.3). *WEEG2* besteht aus einer Liste der Segmente (*WEEG2Segment*), also der kontinuierlichen Abschnitte der Aufnahme, und einer Liste von Informationen über die einzelnen Kanäle (*WEEGChannelInfo*). Außerdem werden allgemeine Metainformationen gespeichert wie der Dateiname und die Aufnahmefrequenz.
In *WEEG2Segment* sind die Werte der Zeitreihe gespeichert. Dies ist die Klasse, über welche die eigentlichen Spannungsdaten abgefragt werden können.
WEEGChannelInfo enthält Metadaten über einen bestimmten Kanal der Aufnahme. Dies ist zur Zeit der Name des Kanals, die Einheit (meist Mikrovolt) und die Elektrodenposition auf der Kopfoberfläche.

Die drei Klassen bekommen ihre Daten von *WPagerEEG*. Dabei werden derzeit alle Metadaten bereits bei der Erstellung des Objekts im Konstruktor ausgelesen und zwischengespeichert. Nur die Spannungsdaten werden

5 *MEG* steht für *Magnetoenzephalographie* und misst die magnetische Aktivität des Gehirns, im Gegensatz zur elektrischen Aktivität bei *EEG*.

auf Grund ihrer Größe erst dann aus der Datei geladen, wenn auf sie zugegriffen werden soll. Dazu muss beim Lesen der Spannungsdaten immer der Bereich angegeben werden, welcher benötigt wird.
Die Entscheidung, welche Daten zwischengespeichert werden, ist relativ einfach änderbar. Dazu muss nur die Implementierung der jeweiligen Klasse angepasst werden, andere Teile des Programm müssen nicht umgeschrieben werden. Dies ist möglich, da jede Klasse im Konstruktor einen Zeiger zu *WPagerEEG* übergeben bekommt und alle Abfragen über get-Methoden gemacht werden. So ist angedacht, die zuletzt gelesenen Spannungsdaten zu cachen, was allerdings bisher nicht nötig war, da die Geschwindigkeit auch ohne Caching den Anforderungen genügt.
Die Elektrodenposition liest *WEEGChannelInfo* aus einer ebenfalls im Konstruktor übergebenen *WEEGPositionsLibrary*. Erst hier wird die Position dem entsprechenden Kanal zugeordnet. Auch die Position wird zwischengespeichert, so dass nach der Erstellung eines *WEEGChannelInfo*-Objekts keine *WEEGPositionsLibrary* mehr nötig ist.
Derzeit werden beim Laden eines EEG-Datensatzes die Elektrodenpositionen aus einer ELC-Datei gleichen Namens mit angepasster Dateiendung gelesen. Dies ist nötig, da es in der OpenWalnut-GUI noch nicht möglich ist, mehrere Ausgangsdateien für einen Datensatz zu wählen. Sobald diese Funktion verfügbar ist, kann implementiert werden, dass die Auswahl einer ELC-Datei vom Benutzer vorgenommen wird.

5.2 EEG VIEW MODUL

Für die Visualisierung wurde ein neues OpenWalnut-Modul (siehe Abschnitt 4.3) geschrieben. Das *EEG View* genannte Modul beherrscht die Darstellung des Elektroenzephalogramms in 2D, die Darstellung der Kopfoberfläche in 3D und die Kombination dieser Daten mit den Nervenfaserdaten. Auf diese drei Funktionalitäten wird in den folgenden Abschnitten nacheinander eingegangen.

5.2.1 *Elektroenzephalogramm*

Für das Elektroenzephalogramm wird ein *Custom Widget* verwendet, welches die Darstellung eines 2D-Graphs unabhängig von den anderen geladenen Daten in einen eigenen Anzeigefenster gestattet, siehe Abschnitt 4.4.1.

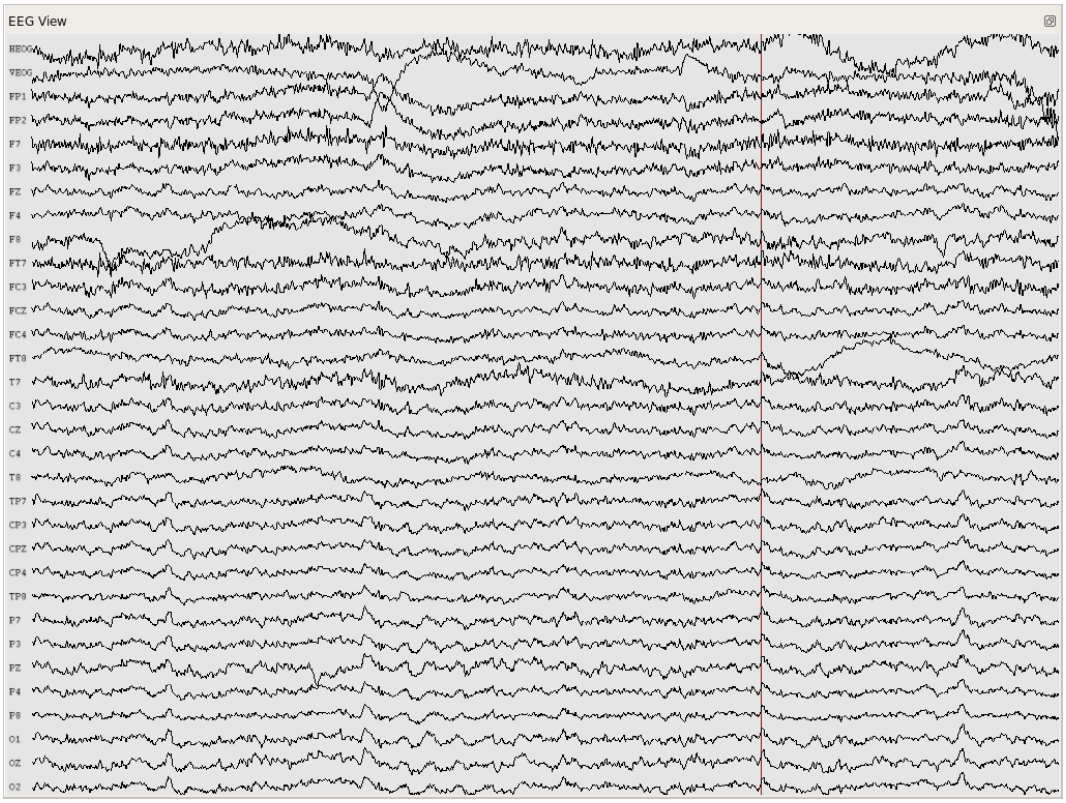

Abbildung 5.2: EEG View Modul: Elektroenzephalogramm.

In Abbildung 5.2 sieht man ein solches Anzeigefenster mit geladenen Daten. Links werden die Namen der Elektroden dargestellt und der Rest der Anzeige sind die Zeitreihen der einzelnen Kanäle. Ein Zeitpunkt ist durch eine rote Linie markiert.
Durch Klicken der linken Maustaste kann ein anderer Zeitpunkt markiert werden. Die Anzeige kann verschoben werden, indem man sie mit der mittleren Maustaste festhält und die Maus in die gewünschte Richtung bewegt. Die Größe des angezeigten Zeitfensters kann mit der rechten Maustaste geändert werden, ebenfalls durch Festhalten und Ziehen. Mit dem Mausrad allein kann man die Empfindlichkeit der dargestellten Spannung verändern. Dreht man es mit festgehaltener rechten Maustaste, so ändert man den Abstand der einzelnen Kanäle untereinander.

Intern werden bei diesen Eingaben die in Abschnitt 4.3 beschriebenen *Properties* geändert. Die Properties, welche an der Darstellung des Elektroenzephalogramms beteiligt sind, kann man in Abbildung 5.3 sehen. *Labels*

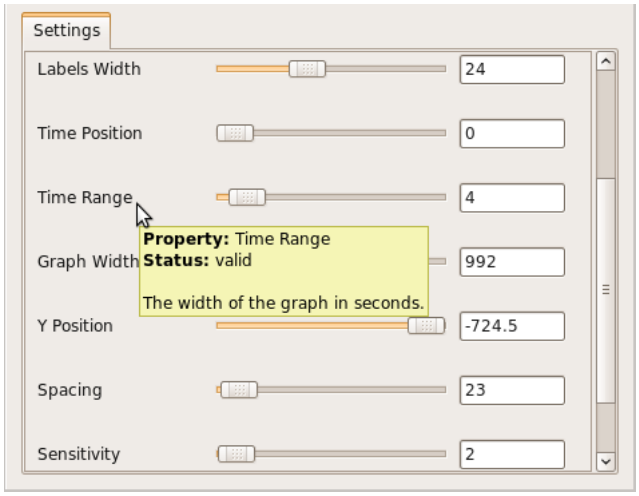

Abbildung 5.3: EEG View Modul: Properties für die Darstellung des Elektroenzephalogramms.

Width bezeichnet den Platz in Pixel, der für die Anzeige der Elektrodennamen reserviert ist. *Time Position* ist der Startzeitpunkt, ab welcher die Anzeige des EEGs beginnt, *Time Range* ist die Größe des angezeigten Ausschnitts. Die Property *Graph Width* wird angepasst, wenn die Größe des Anzeigefensters verändert wird und bezeichnet die Breite des Anzeigefensters ohne den Platz für die Elektrodennamen. Die Verschiebung des angezeigten Bereichs in vertikaler Richtung wird durch *Y Position* angegeben. *Spacing* bezeichnet den Abstand der einzelnen Kanäle untereinander, *Sensitivity* die Empfindlichkeit der dargestellten Spannung. Das Modul besitzt außerdem noch die Properties *Draw Electrodes*, *Draw Head Surface*, *Draw Labels* sowie *Color Sensitivity*, welche für die Darstellung der Kopfoberfläche in 3D wichtig sind und im entsprechenden Abschnitt genauer beschrieben werden.
Bei der Abbildung der Properties handelt es sich um einen Screenshot des Dataset Browsers, welcher ein Teil der GUI OpenWalnuts ist. Die Properties können auch von dort aus geändert werden, was insbesondere beim Testen des Programms hilfreich ist.
Die Eingaben nimmt dabei die Klasse *WEEGViewHandler* entgegen, welche von *osgGA::GUIEventHandler* [31, Abschnitt 3.2.4 Picking] erbt. Diese besitzt die Methode *handle*, welche bei Eingaben aufgerufen wird. Die Implementierung dieser Methode in der *WEEGViewHandler*-Klasse fragt die Art der Eingabe ab und ändert die entsprechende Property.

Ändert sich eine Property, so reagiert darauf ein ein System aus *OSG Callbacks* [31, Abschnitt 3.2.2 Callbacks]. Diese passen die Anzeige entsprechend an. Dabei gibt es keine direkte Verbindung zwischen dem Event-Handler für die Eingabe und den Callbacks für die Ausgabe. Die Zusammenarbeit geschieht dadurch, dass sie auf die gleichen Properties zugreifen. Durch diese Vorgehensweise sind Eingabe (welche die Properties ändern) und Ausgabe (welche die Properties lesen) klar voneinander getrennt und können unabhängig voneinander implementiert und geändert werden.

Wird das darzustellende Zeitfenster geändert, so werden die nicht mehr benötigten Daten aus dem Speicher gelöscht und nur die neuen Daten ausgelesen. Auf diese Weise ist der Speicherbedarf minimal, da nur die Werte im Speicher liegen, die gerade angezeigt werden.
Es wäre auch möglich, nicht mehr benötigte Daten erst später zu löschen oder bei dem Anfordern von neuen Daten mehr auszulesen, als im Moment benötigt werden. Dadurch würde bei mehrmaligen kleinen Veränderungen des Zeitfensters Lesezugriffe eingespart werden und damit schneller reagiert werden können. Dies wurde allerdings nicht implementiert, da die Anzeige bereits schnell genug reagiert und solche Optimierungen auf Kosten des Speicherbedarfs gehen.
Die Darstellung des Graphen erfolgt mit Hilfe von *GL-Line-Strips* durch die Grafikkarte.

Die Speicherung des markierten Zeitpunkts erfolgt als *WEEGEvent*-Objekt. Diese Klasse speichert nicht nur die Zeit in Sekunden, sondern auch die graphische Repräsentation in Form einer roten Linie sowie auch die Spannungswerte jedes Kanals zu dem Zeitpunkt. Dabei wird linear interpoliert, was der Darstellung von geraden Linien zwischen den gegebenen Werten entspricht. Bei den in EEG-Messungen üblichen Aufnahmefrequenzen ist diese Art der Interpolation ausreichend.
Programmteile, welche den markierten Zeitpunkt oder die Spannungswerte an diesem Zeitpunkt benötigen, bekommen einen Zeiger auf das *WEEG-Event*-Objekt und können damit die benötigten Daten direkt abfragen, ohne auf den EEG-Datensatz zuzugreifen. Dies ist insbesondere von Vorteil, wenn diese Daten mehrfach benötigt werden, da so ein mehrfaches Lesen und Interpolieren eingespart wird.

5.2.2 *Kopfoberfläche*

Der erste Schritt, um die EEG-Daten in Zusammenhang mit anderen Daten zu bringen, ist die Darstellung der Positionen der Elektroden in 3D. Dies geschieht im EEG View Modul im Hauptanzeigefenster, in der auch die anderen Hirndaten angezeigt werden.
Es können drei Darstellungen gemeinsam oder einzeln aktiviert werden: Kugeln an den Elektrodenpositionen, die Namen des Kanäle oberhalb der Positionen und ein Dreiecksgitter zwischen den Positionen, welches die Kopfoberfläche abbilden soll. Die Auswahl, welche dieser Darstellungen aktiviert ist, erfolgt über die drei Properties *Draw Electrodes*, *Draw Labels* und *Draw Head Surface*.
Dabei ist es möglich, dass ein Datensatz nicht zu jedem Kanal eine Position enthält, so dass die 3D-Darstellung aus weniger Elektroden besteht als das Elektroenzephalogramm. Abbildung 5.4 zeigt die drei Darstellungen von einem Datensatz mit 32 Kanälen, bei dem allerdings nur von 30 Elektroden die Position bekannt ist und somit nur diese für die 3D-Darstellung genutzt werden können.
Zur Berechnung des Dreiecksgitters zwischen den Elektrodenpositionen wird die *Delaunay-Triangulation* [32] genutzt. Diese arbeitet im $\mathbb{R}^2$, so dass eine Projektion der Positionen auf eine Ebene nötig ist. Die einfachste Projektion aller Punkte auf die x-y-Ebene erzeugt unschöne Artefakte an Stellen, wo die Elektrodenpositionen fast übereinander sind, wie zum Beispiel in der Nähe der Ohren. Dies ist in Abbildung 5.5 anhand der blauen Punkte zu sehen. Die blauen Punkte im oberen Teil stellen beispielhaft eine simple Elektrodenanordnung dar. Nach der Projektion auf die x-y-Ebene haben die Punkte 1 und 2 sowie die Punkte 5 und 6 jeweils die selbe Position.
Deshalb wird vor die Projektion noch eine Transformation gestellt. Dazu wird der Schwerpunkt aller Positionen berechnet. Dieser ist in der Abbildung als schwarzes Kreuz zu sehen. Danach werden alle Punkte, welche unterhalb des Schwerpunkts liegen, von diesem weggedrückt und alle Punkte oberhalb werden in die Nähe des Schwerpunkts gezogen. In dem Beispiel werden die Punkte 1 und 6 vom Schwerpunkt weggedrückt und die Punkte 3 und 4 in seine Nähe gezogen; die Punkte 2 und 5 ändern sich nicht, da sie auf der Höhe des Schwerpunkts liegen. Es ergeben sich die durch rote Kreise markierten Punkte 1′ bis 6′. Die Verschiebung ist umso stärker, je weiter der Punkt vom Schwerpunkt entfernt ist.
Wie zu sehen ist, wird durch diese Transformation die Kopfoberfläche an der Unterseite auseinandergezogen. In der Projektion auf die x-y-Ebene

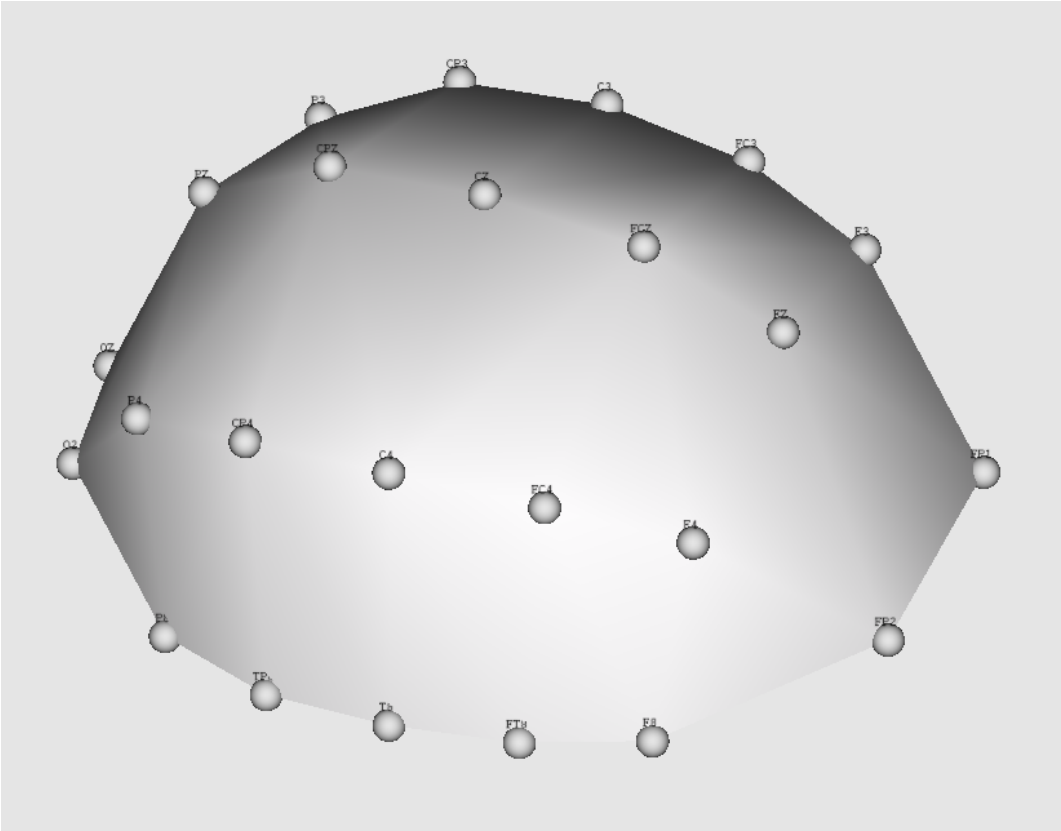

Abbildung 5.4: EEG View Modul: Elektrodenpositionen, Namen der Elektroden und Kopfoberfläche bestehend aus 30 Elektroden.

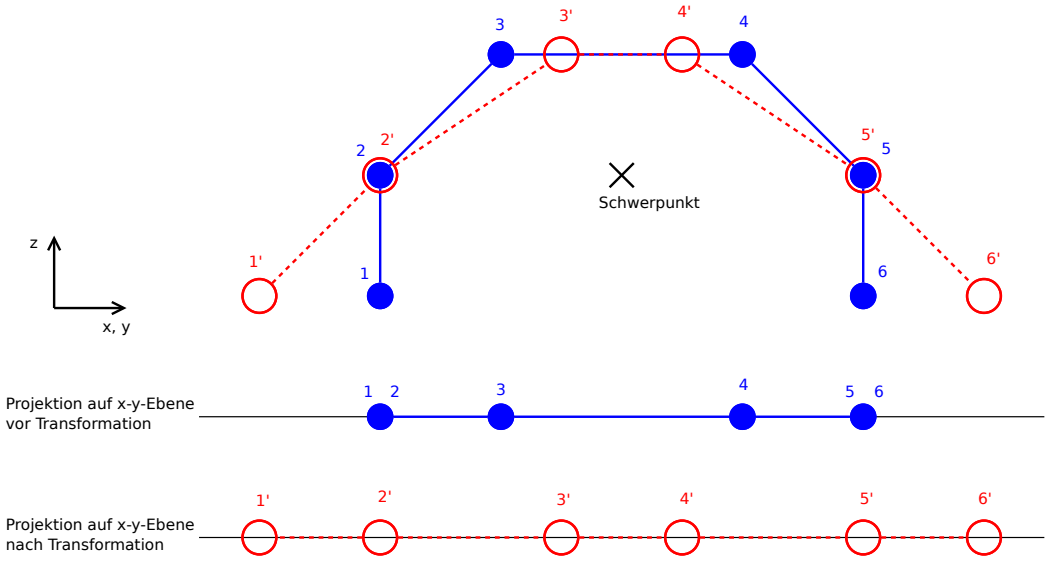

Abbildung 5.5: Verdeutlichung der Transformation der Elektrodenpositionen, um eine zufriedenstellende Triangulation zu erhalten. Die blauen Punkte stellen ein Beispiel eines Kopfmodells dar. Nach der Projektion auf die x-y-Ebene haben einige Punkte die selbe Position. Die roten Kreise sind die Elektrodenpositionen nach der Transformation. Die Projektion dieser Positionen lässt sich gut triangulieren.

sind Punkte, welche vorher übereinander lagen, nun nebeneinander. Auf diese Projektion der transformierten Punkte wird die Delaunay-Triangulation angewandt, welche ein zufriedenstellendes Dreiecksnetz erzeugt. Zur endgültigen Darstellung werden die untransformierten Punkte genutzt, welche durch die Dreiecke der entsprechenden transformierten Punkte verbunden sind.

Wird eine Zeitposition in der 2D-Anzeige markiert, so ändert sich die Farbe der Kopfoberfläche entsprechend der Spannungswerte an den Elektroden zu dieser Zeit. Abbildung 5.6 zeigt die gleiche Kopfoberfläche wie Abbildung 5.4 nach Auswahl eines Zeitpunkts. Negative Spannungen werden dabei blau dargestellt, Werte im Bereich von Null weiß und große Werte rot.
Die Zuordnung der Spannungswerte zu den Farben ist dabei variabel. Intern wird eine Colormap gespeichert, welche aus der gleichmäßigen linearen Interpolation zwischen den drei Farben besteht. Dazu wird die *osgSim::ColorRange*-Klasse von *OpenSceneGraph* benutzt, welche aus einer Zahl im Intervall -1 bis 1 einen Farbwert ermittelt. Welche Spannung auf die 1 bzw. -1 abgebildet wird, also die Empfindlichkeit der Farbdarstel-

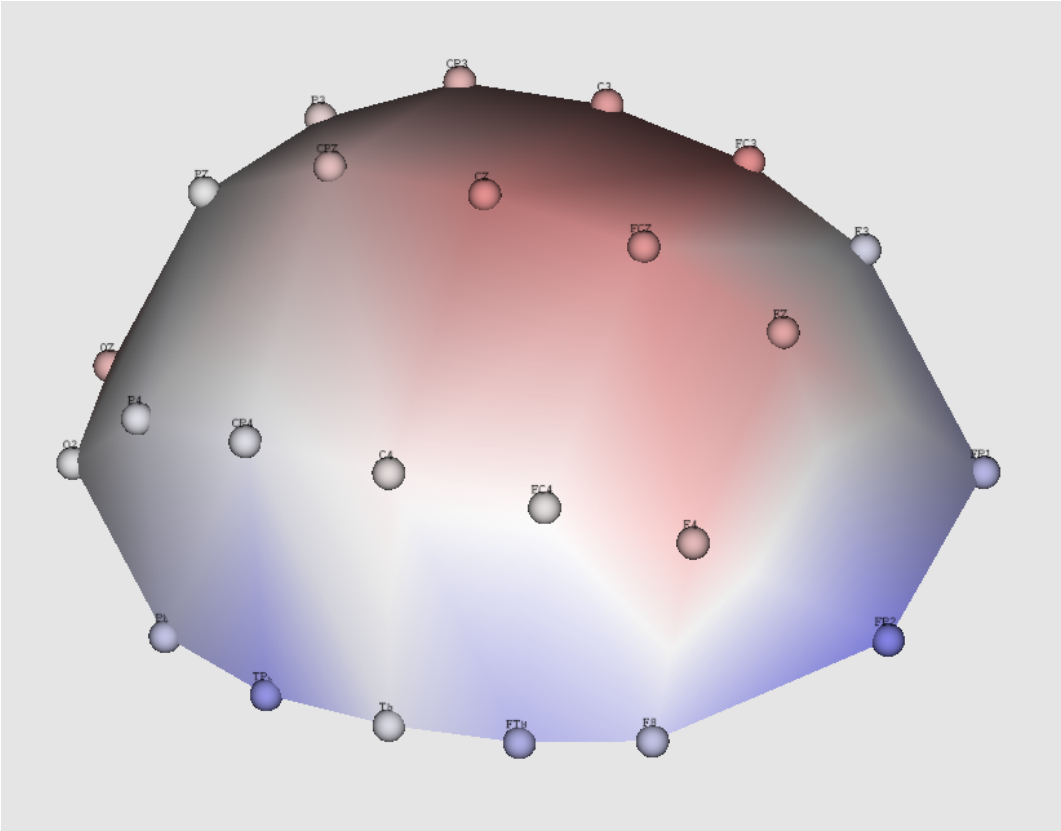

Abbildung 5.6: EEG View Modul: Darstellung der Spannungswerte zu einem bestimmten Zeitpunkt auf der Kopfoberfläche.

lung, ist dabei durch die Property *Color Sensitivity* änderbar. Diese kann man nicht nur in der GUI editieren, sondern auch durch Bewegen des Mausrads bei gedrückter linker Maustaste.
Die Kugeln an den Elektrodenpositionen erhalten die Farbe der entsprechenden Elektrode. Bei der Kopfoberfläche zwischen den Elektroden ist allerdings Interpolation nötig, um auch für die Stellen, für die kein Wert gegeben ist, eine sinnvolle Farbe darzustellen.
Dies kann mit Hilfe der Grafikkarte gelöst werden. Dabei ist es nicht ausreichend, die Farben der Dreieckseckpunkte richtig zu setzen, da so innerhalb des Dreiecks diese Farben gemischt werden, was nicht der gewünschten Colormap entsprechen muss. Zum Beispiel würde bei den verwendeten Farben zwischen einem roten und einem blauen Eckpunkt die Farbe Lila entstehen, nicht Weiß, wie zu wünschen ist.
Eine zufriedenstellende Interpolation wurde mit Hilfe einer 1-D-Textur implementiert. Dazu wird die Colormap als eindimensionale Textur gespeichert. Jeder Eckpunkt erhält nun eine Texturkoordinate, welche vom Spannungswert der entsprechenden Elektrode und der Property *Color Sensitivity* abhängt. Die Grafikkarte interpoliert damit innerhalb des Dreiecks die Texturkoordinate und übersetzt diese erst nach der Interpolation in eine Farbe, so dass die richtige Colormap dargestellt wird. Auf diese Weise wird ohne Belastung des Hauptprozessors die richtige lineare Interpolation der Spannungswerte auf der Kopfoberfläche realisiert.

5.2.3 *Nervenfaserbündel auswählen*

Eine der Ziele dieser Arbeit ist es, die EEG-Daten zusammen mit Nervenfaserbündeldaten darzustellen. Dabei kam die Idee, dazu die *Quelllokalisation* zu nutzen. Mit ihr ist es möglich, aus dem EEG die Positionen der Dipolquellen innerhalb des Gehirns zu einem bestimmten Zeitpunkt zu bestimmen, wie in Abschnitt 2.1.1 beschrieben. Diese Position kann dann dazu genutzt werden, die Nervenfaserbündel anzuzeigen, welche in deren Nähe liegen. Dadurch erhält man ein Bild der Verbindungen, welche die Quellposition mit anderen Hirnregionen hat.
In Abbildung 5.7 ist zu sehen, wie dies im EEG View Modul umgesetzt wurde. Wenn zusätzlich zu den EEG-Datensatz ein Faserbündeldatensatz geladen ist und eine Zeitposition markiert wurde, dann wird an einer bestimmten Position eine *Region-of-Interest* gesetzt, so dass nur die Faserbündel zu sehen sind, welche durch sie verlaufen. Die Möglichkeit, die Fa-

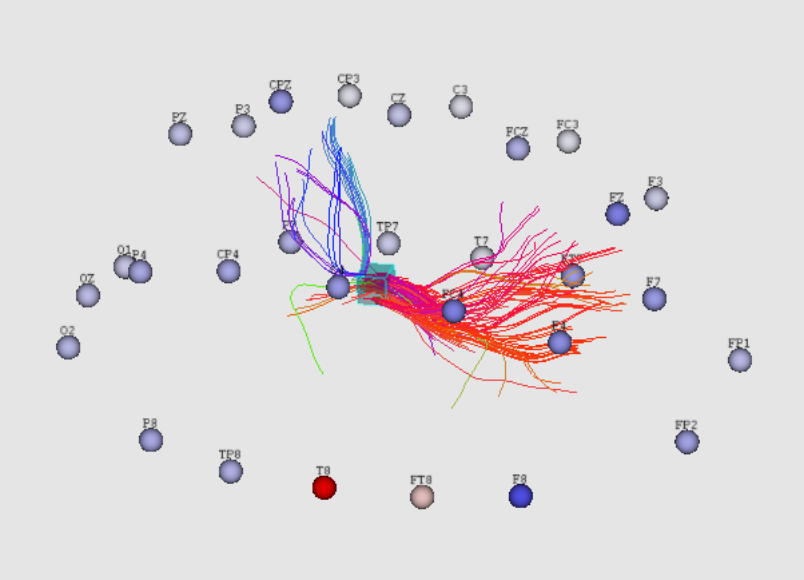

Abbildung 5.7: EEG View Modul: Elektrodenpositionen mit den Nervenfaserbündeln, welche durch die türkis dargestellte ROI verlaufen, deren Position durch die Spannungswerte an den Elektroden berechnet wurde.

serbündel auszuwählen, welche durch eine gegebene ROI verlaufen, war bereits in OpenWalnut implementiert. Um ROIs aus einem Modul heraus zu ändern, musste OpenWalnut erweitert werden, wie in Abschnitt 4.4.2 beschrieben.
Da die Quelllokalisation rechnerisch sehr aufwendig ist, sollte sie nicht während der Laufzeit berechnet werden. Sinnvoll ist, sie vorher von einem darauf spezialisierten Programm erstellen zu lassen und dessen Ergebnis in OpenWalnut einzuladen. Mit den Nervenfaserbündeldaten wird ebenfalls auf diese Weise verfahren.
Im Rahmen dieser Diplomarbeit war es allerdings zeitlich nicht mehr möglich, ein geeignetes Programm zur Quelllokalisation auszuwählen und dessen Ausgabedateiformat von OpenWalnut lesbar zu machen.
Deshalb ist die derzeit angezeigte Position der ROI nicht die tatsächliche Position der Dipolquelle. Es wird zwar aus den Spannungswerten an den Elektroden ein Punkt errechnet, so dass sich die ausgewählten Faserbündel bei unterschiedlichen Zeitpunkten ändern, allerdings beruht diese Berechnung auf keiner medizinischer Grundlage. Der berechnete Punkt ist eine Linearkombination der Elektrodenpositionen, welche mit den jeweiligen Spannungen so gewichtet werden, dass er innerhalb der konvexen Hülle liegt.

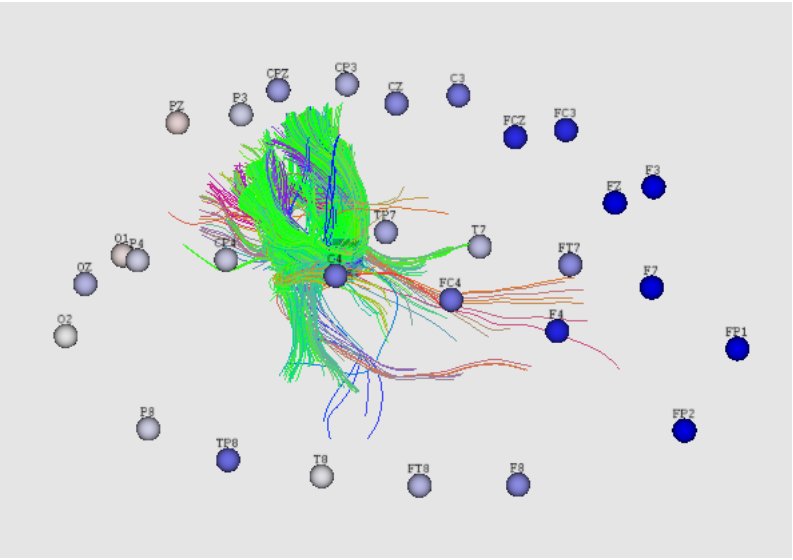

Abbildung 5.8: EEG View Modul: Ausgewählte Nervenfaserbündel von oben betrachtet, zusammen mit den Elektrodenpositionen.

Dadurch ist ein Proof-of-Concept entstanden, welches eine Quellposition, wenn auch unbegründet, auf eine interaktive Weise visualisiert.

Durch die Kombination der verschiedenen Darstellungen und der Navigation im 3D-Raum sind verschiedene Ansichten möglich. So zeigen Abbildung 5.8 und 5.9 die gleichen EEG-Daten in Kombination mit den gleichen Nervenfaserbündeldaten zum gleichen ausgewählten Zeitpunkt. Im ersten Bild wird der Kopf seitlich von oben betrachtet. Die Kopfoberfläche wurde nicht dargestellt, um freie Sicht auf die Faserbündel zu erlauben. Im zweiten Bild ist die Perspektive von unten. Da so die Faserbündel vor der Kopfoberfläche liegen, kann diese dargestellt werden. Man sieht deutlich die interpolierte Spannungsverteilung.

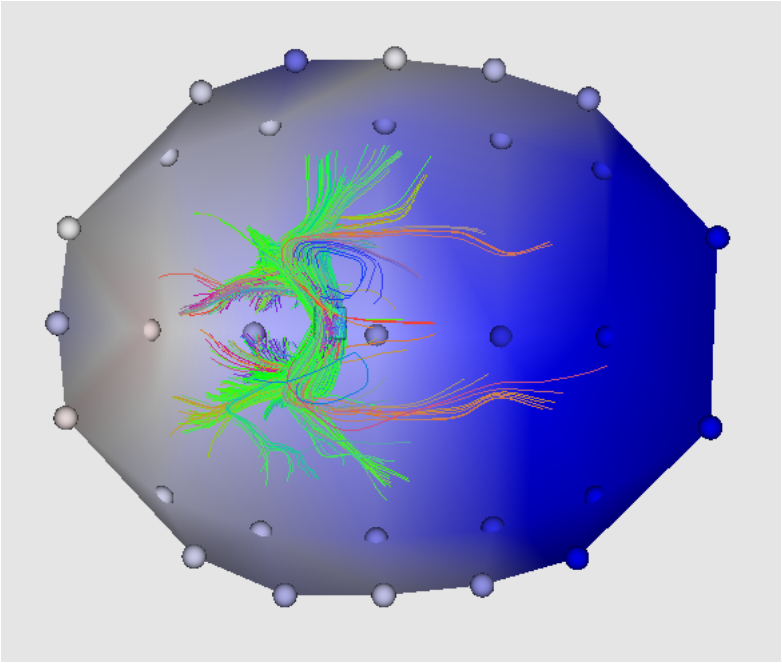

Abbildung 5.9: EEG View Modul: Ausgewählte Nervenfaserbündel von unten betrachtet, zusammen mit der Kopfoberfläche und den Elektrodenpositionen.

6

ZUSAMMENFASSUNG UND AUSBLICK

Das entwickelte Visualisierungssystem erlaubt die vielfältige Darstellung von EEG-Daten. Sie können als klassisches Elektroenzephalogramm angezeigt werden. Die Positionen der Elektroden können im dreidimensionalen Raum dargestellt werden, einschließlich der Kopfoberfläche zwischen ihnen. Wird ein Zeitpunkt markiert, so können auf dieser Fläche die interpolierten Spannungswerte durch Farben kodiert verdeutlicht werden. Außerdem wird aus den Spannungswerten eine Position errechnet, welche für die Kombination mit Nervenfaserbündeldaten genutzt wird. Dazu werden die Faserbündel ausgewählt, welche in der Nähe der errechneten Position verlaufen.
Dabei muss beachtet werden, dass die derzeitig errechnete Position nicht unbedingt die tatsächliche Quelle der beobachten EEG-Daten ist. Dies ist allerdings möglich, indem man das Programm so erweitert, dass es vorberechnete Resultate der Inversrechnung einlädt und nutzt. Außerdem stammen auf den gezeigten Bildern die EEG- und Nervenfaserbündeldaten nicht von der gleichen Person und sind nicht aufeinander registriert.
Dennoch ist das Ziel dieser Arbeit erreicht: Es entstand ein nutzbares System zur interaktiven, kombinierten Visualisierung von EEG- und Nervenfaserbündeldaten. Es wurde gezeigt, dass das Konzept umsetzbar ist, beide Methoden gemeinsam darzustellen.
Auf dieser Erkenntnis aufbauend sollte das System durch obige Erweiterung fertiggestellt werden, damit es für neurologische Untersuchungen angewandt werden kann. Außerdem sind auch an der Visualisierung noch Verbesserungen möglich. So können im Elektroenzephalogramm die einzelnen Kanäle durch unterschiedliche Farben deutlicher voneinander getrennt werden. Wird eine Zeitposition markiert, so können Informationen über diese Position in Textform ausgegeben werden. Die Geschwindigkeit

des Datenzugriffs ist zwar bereits ausreichend, falls es aber nötig ist, kann sie durch Caching noch verbessert werden.
Die Anzeige der Kopfoberfläche in 3D kann noch verfeinert werden. Zur Zeit wirkt sie zumindest bei wenigen Elektroden noch eckig. Es ist angedacht, sie zu glätten, indem sie als *Subdivision Surface* [33] dargestellt wird. Bei dieser Technik werden die Dreiecke, aus denen die Oberfläche besteht, in mehrere kleinere, feinere Dreiecke unterteilt. Durch rekursive Fortsetzung dieses Verfahren wird eine glatte Fläche erzeugt. Außerdem kann man versuchen, die Kopfoberfläche transparent darzustellen, um sie und die Nervenfaserbündel innerhalb des Gehirns leichter gleichzeitig anzeigen zu können. Weiterhin sollte die Zuordnung der Spannungswerte zu den dargestellten Farben angezeigt werden.

LITERATURVERZEICHNIS

[1] Berger, Hans: *Über das Elektrenkephalogramm des Menschen*. Zeitschrift für die gesamte Neurologie und Psychiatrie, 87(1):527–570, Dezember 1929.

[2] Jasper, Herbert H.: *The ten-twenty Electrode System of the International Federation*. Electroencephalography and Clinical Neurophysiology, 10(2):371–375, Mai 1958.

[3] Knösche, Thomas Reiner: *Solutions of the Neuroelectromagnetic Inverse Problem – An Evaluation Study*. Doktorarbeit, Universiteit Twente Enschede, 1997.

[4] Stejskal, E. O. und J. E. Tanner: *Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient*. The Journal of Chemical Physics, 42(1):288–292, Januar 1965.

[5] Westin, C.-F., S. E. Maier, H. Mamata, A. Nabavi, F. A. Jolesz und R. Kikinis: *Processing and visualization for diffusion tensor MRI*. Medical Image Analysis, 6(2):93–108, Juni 2002.

[6] Schurade, Ralph: *Visualisierung von Nervenfaserflächen im menschlichen Gehirn aus DTI Daten*. Diplomarbeit, Universität Leipzig, Januar 2009.

[7] Mori, Susumu und Peter C. M. van Zijl: *Fiber tracking: principles and strategies – a technical review*. NMR in Biomedicine, 15(7-8):468–480, 2002.

[8] Akers, David, Anthony Sherbondy, Rachel Mackenzie, Robert Dougherty und Brian Wandell: *Exploration of the Brain's White Matter Pathways with Dynamic Queries*. In: *Proceedings of IEEE Visualization 2004*, Seiten 377–384, 2004.

[9] Sherbondy, Anthony, David Akers, Rachel Mackenzie, Robert Dougherty und Brian Wandell: *Exploring Connectivity of the Brain's White Matter with Dynamic Queries*. IEEE Transactions on Visualization and Computer Graphics, 11(4):419–430, 2005.

[10] Blaas, Jorik, Charl P. Botha, Bart Peters, Frans M. Vos und Frits H. Post: *Fast and Reproducible Fiber Bundle Selection in DTI Visualization*. In: Silva, Claudio, Eduard Gröller und Holly Rushmeier (Herausgeber): *Proceedings of IEEE Visualization 2005*, Seiten 59–64, 2005.

[11] Schurade, Ralph: *FiberNavigator*. Webseite. `http://code.google.com/p/fibernavigator/`, Zugriff am 27. April 2010.

[12] Heinonen, Tomi, Antti Lahtinen und Veikko Häkkinen: *Implementation of Three-Dimensional EEG Brain Mapping*. Computers and Biomedical Research, 32(2):123–131, April 1999.

[13] Jovanov, Emil, Dušan Starčević, Aleksandar Samardžić, Andy Marsh und Željko Obrenović: *EEG analysis in a telemedical virtual world*. Future Generation Computer Systems, 15:255–263, 1999.

[14] Jovanov, Emil, Dusan Starcevic, Andy Marsh, Zeljko Obrenovic, Vlada Radivojevic und Aleksandar Samardzic: *Multi Modal Presentation in Virtual Telemedical Environments*. In: Sloot, Peter, Marian Bubak, Alfons Hoekstra und Bob Hertzberger (Herausgeber): *High-Performance Computing and Networking*, Band 1593 der Reihe *Lecture Notes in Computer Science*, Seiten 964–972. Springer-Verlag, April 1999.

[15] Samardzic, Aleksandar B., Emil S. Jovanov und Dusan B. Starcevic: *Real-time Visualization of Brain Electrical Activity*. Real-Time Imaging, 6(1):69–76, Februar 2000.

[16] Delorme, Arnaud und Scott Makeig: *EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis*. Journal of Neuroscience Methods, 134(1):9–21, März 2004.

[17] Mosher, John C., Sylvain Baillet, Felix Darvas, Dimitrios Pantazis, Esen Kucukaltun-Yildirim und Richard M. Leahy: *BrainStorm Electromagnetic Imaging Software*. International Journal of Bioelectromagnetism, 7(2):189–190, 2005.

[18] University of Southern California: *Brainstorm*. Webseite. `http://neuroimage.usc.edu/brainstorm/`, Zugriff am 22. Juni 2010.

[19] Kulish, Vladimir, Alexei Sourin und Olga Sourina: *Human electroencephalograms seen as fractal time series: Mathematical analysis and visualization*. Computers in Biology and Medicine, 36:291–302, 2006.

[20] Goszczyńska, Hanna, Leszek Królicki, Adam Bajera, Ewa Zalewska, Leszek Kowalczyk, Piotr Walerjan, Andrzej Rysz und Krystyna Kolebska: *The Procedure for SPECT and BEAM Images Adjustment Visualisation of EEG Electrodes in SPECT Images*. Polish Medical Journal of Medical Physics and Engineering, 13(3):115–125, 2007.

[21] Caat, Michael ten: *Multichannel EEG Visualization*. Doktorarbeit, Rijksuniversiteit Groningen, 2008.

[22] Universität Leipzig: *Abteilung für Bild- und Signalverarbeitung*. Webseite. `http://www.informatik.uni-leipzig.de/bsv/`, Zugriff am 26. April 2010.

[23] Max-Planck-Institut für Kognitions- und Neurowissenschaften: *Kortikale Netzwerke und Kognitive Funktionen*. Webseite. `https://www.cbs.mpg.de/groups/support/1/`, Zugriff am 26. April 2010.

[24] Universität Leipzig: *OpenWalnut*. Webseite. `http://www.openwalnut.org`, Zugriff am 26. April 2010.

[25] OSG Community: *OpenSceneGraph*. Webseite. `http://www.openscenegraph.org`, Zugriff am 27. April 2010.

[26] Nokia Corporation: *Qt*. Webseite. `http://qt.nokia.com`, Zugriff am 27. April 2010.

[27] ANT B.V.: *Firmenauftritt*. Webseite. `http://www.ant-neuro.com`, Zugriff am 30. April 2010.

[28] ANT B.V., Enschede: *EEProbe Reference Manual*, März 2003.

[29] ANT B.V.: *Libeep*. Webseite. `http://sourceforge.net/projects/libeep/`, Zugriff am 30. April 2010.

[30] ANT B.V., Enschede: *ASA Help*, September 2009.

[31] Martz, Paul: *OpenSceneGraph Quick Start Guide*. Skew Matrix Software LLC, Louisville, CO, 2007.

[32] Delaunay, Boris N.: *Sur la sphère vide*. Bulletin de l'Académie des Sciences de l'URSS, (6):793–800, 1934.

[33] Catmull, E. und J. Clark: *Recursively generated B-spline surfaces on arbitrary topological meshes*. Computer-Aided Design, 10(6):350–355, November 1978.

ABBILDUNGSVERZEICHNIS

ERKLÄRUNG

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet.
Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig, den 19. August 2010

(Cornelius Müller)