Aus der Anatomischen Anstalt
der Ludwig-Maximilians-Universität München

Vorstand: Prof. Dr. med. Winfried Lange

Computergestützte Visualisierung eines human-embryonalen Gehirns

Dissertation
zum Erwerb des Doktorgrades der Medizin
an der Medizinischen Fakultät der
Ludwig-Maximilians-Universität zu München

vorgelegt von

Peter Weinert
aus München
2007

Mit Genehmigung der Medizinischen Fakultät
der Universität München

1. Berichterstatter:	Prof. Dr. rer. nat. Thomas Heinzeller
2. Berichterstatter:	Prof. Dr. med. Hartmut Brückmann
Mitberichterstatter:	Priv. Doz. Dr. med. Peter Alois Winkler
	Prof. Dr. med. Thomas Cremer
Dekan:	Prof. Dr. med. Dietrich Reinhardt
Tag der mündlichen Prüfung:	22.03.2007

Zusammenfassung

In der vorliegenden Arbeit wurde das 3-D-Modell des Gehirns eines frühen humanen Embryos angefertigt, des Weiteren eine 3-D-Software entwickelt, die es erlaubt, das Modell in Echtzeit manipulierbar darzustellen und es schließlich vollwertig stereoskopisch betrachten zu können. Diese Software wird Studierenden auf dem Server des Leibniz-Rechenzentrums zur Verfügung gestellt. Damit können sie am eigenen Rechner virtuelle 3-D-Modelle, die am Lehrstuhl III der Anatomischen Anstalt erarbeitet und bereit gestellt werden, plastisch (auch stereoskopisch) studieren. So besteht in Zukunft die Möglichkeit, die embryonale Entwicklung mit zeitgemäßen Methoden leicht verständlich zu veranschaulichen.

Dem 3-D-Modell diente als Quellmaterial eine Schnittserie aus 574 Schichten eines menschlichen Embryos im Carnegie-Stadium 18. Die Schichten wurden über ein Mikroskop digitalisiert und am Computer wieder räumlich zueinander ausgerichtet. Um die ursprünglichen anatomischen Verhältnisse trotz der verzerrten Schnitte mit dem kommerziell verfügbaren Programm AmiraDev 3.0 möglichst korrekt herzustellen, wurde dieser elementare aber komplizierte Schritt durch selbst entwickelte Techniken unterstützt und sichtbar verbessert. Im so entstandenen Bilderstapel wurde das Gehirn markiert und dann zum virtuellen Modell trianguliert.

Die hier entwickelte 3-D-Software erlaubt es, das willkürlich drehbare 3-D-Modell sowie andere Rekonstruktionen am Rechner anzuzeigen. Eine frei wählbare Schnittebene und die Transparenzfunktion geben Aufschluss über den inneren Aufbau des 3-D-Modells, z. B. über das Ventrikelsystem. In der Programmiersprache C++ wurden hocheffiziente, handoptimierte Bibliotheken für lineare Algebra und Computergrafik entwickelt, die eine ruckfreie Betrachtung ermöglichen. Im Hinblick auf Effizienz, Erweiterbarkeit und Fehlervermeidung wurde auf ein wohl überlegtes Software-Design mit sicherer Semantik Wert gelegt.

Auch wenn ein virtuelles 3-D-Modell bereits einen besseren räumlichen Eindruck als eine plane Abbildung verschafft, kommt eine echte Tiefenwirkung erst durch stereoskopische Darstellung zustande. Diese wurde lege artis als asymmetrische perspektivische Projektion so implementiert, dass sie unkompliziert auf Tastendruck genutzt werden kann. Die ausgereifte Software beherrscht das Anaglyphenverfahren (Rot-Grün-Brille) genauso wie auch aufwendigere Projektionsverfahren.

Die Arbeit stellt darüber hinaus in kurzer Form die für die Programmentwicklung relevanten mathematischen Grundlagen dar. Ferner wird ein Überblick über die im Internet verfügbaren, teils kommerziell vertriebenen Datensätze – speziell zur Embryologie – gegeben und das selbst entwickelte Darstellungsmodell mit seinen Vorteilen und den (selbst auferlegten) Beschränkungen in dieses Bezugssystem eingeordnet.

Abkürzungen und Glossar

2-D	zweidimensional
3-D	dreidimensional
Bit	binary digit *(binäre Einheit)*
CAD	computer aided design *(technisches Zeichenprogramm)*
CCD	charge coupled device *(Fotokamera-Chip)*
CS	Carnegie-Stadium *(Entwicklungsstadium menschlicher Embryonen)*
CT	computed tomography *(Bildgebungsverfahren)*
fps	frames per second *(Geschwindigkeitsmaß bewegter Bilder)*
G	giga *(10^9 nach SI)*
Gi	gibi *(gigabinary $(2^{10})^3$ nach IEC 60027-2 2^{nd} ed. 2000-11)*
GUI	graphical user interface *(grafische Benutzeroberfläche)*
HE	Hämatoxylin-Eosin *(Farbstoff)*
jpeg	joint photographic expert group *(Bildformat)*
k	kilo *(10^3 nach SI)*
Ki	kibi *(kilobinary $(2^{10})^1$ nach IEC 60027-2 2^{nd} ed. 2000-11)*
M	mega *(10^6 nach SI)*
ME	menschlicher Embryo
Mi	mebi *(megabinary $(2^{10})^2$ nach IEC 60027-2 2^{nd} ed. 2000-11)*
MRI	magnetic resonance imaging *(Bildgebungsverfahren)*
OpenGL	open graphics library *(Computergrafikschnittstelle)*
OPT	optical projection tomography *(Bildgebungsverfahren)*
PC	personal computer *(Einbenutzer-Computer)*
Pixel	picture element *(Rasterbildpunkt)*
RGB	red – green – blue *(Farbraum)*
RGBA	red – green – blue – alpha *(Farbraum mit Transparenz)*
ROI	region of interest *(relevanter Ausschnitt im Bild)*
SSD	sum of square differences *(statistisches Maß)*
STL	standard triangulation language *(Dateiformat)*
TIFF	tag image file format *(Bildformat)*
Voxel	volume pixel *(3-D-Pixel)*

Inhalt

1 Einführung

Leonardo da Vincis (1452 – 1519) anatomische Studien begannen um 1469 während seines ersten Florentiner Aufenthaltes. Dabei leitete ihn zunächst der Wunsch, sich für sein künstlerisches Werk möglichst authentische anatomische Grundlagen zu verschaffen. Sein wissbegieriger Geist ließ jedoch nicht lange auf sich warten. Noch vor dem Jahr 1489 wandte er sich in Mailand der Anatomie zu und fertigte der Wissenschaft willen und nicht allein der Kunst wegen anatomische Zeichnungen an. Einer drohenden Todesstrafe und Sanktionen des Papstes trotzend führte er Sektionen und Präparationen an mehr als dreißig Leichen durch. Heute betrachten wir sein Schaffen, wie die Zeichnung eines Feten in geöffneter Gebärmutter, mit gebührendem Respekt. Seine Illustrationen, von denen 228 Blätter in Windsor erhalten geblieben sind, sollten Grundlage eines anatomischen Atlasses werden, den er nie vollenden konnte. Das gelang Andreas Vesalius (1514 – 1564) mit seinem bedeutenden siebenbändigen Werk „*De humani corporis fabrica*“ aus dem Jahr 1543. Vesalius wagte es zur Empörung seiner Kollegen, den seit Jahrhunderten kanonisch nachgebeteten Dogmen eines Claudius Galenus (129 – 199) zu widersprechen, reanimierte die medizinische Forschung und ging als Begründer der modernen Anatomie in die Geschichte ein.

Stetigen Fortschritten der Technik zum Trotz blieb die Zeichnung als wichtigstes Medium weiterhin bestehen, damit aber bis in die heutige Zeit auch ihr Hauptproblem, die Beschränkung des Bildes auf zwei Dimensionen. Konkret heißt das: Die reale Welt wird durch planare Projektion auf Papier gebannt, der Informationsgehalt wird reduziert. Die Tiefeninformation wird zwar auf der Basis von Abtönungen, Licht-Schattenzeichnung oder Schraffuren als kognitive Leistung des Wahrnehmenden hinzu konstruiert, im Bild selbst jedoch ist und bleibt sie für immer verloren. Aufwendige Modelle und Präparate können einem breiten Publikum nur in beschränktem Maße angeboten werden, wie z. B. die detaillierten Embryomodelle aus Wachs der Freiburger Friedrich und Adolf Ziegler, die 1893 auf der „World's Columbian Exposition“ in Chicago einer breiten Öffentlichkeit präsentiert wurden [Hopwood 2002].

Gleichwohl werden Studierende in anatomischen Vorlesungen hauptsächlich mit den klassischen Medien wie Folien, Dias und Zeichnungen konfrontiert. Doch diese werden den Anforderungen einer Vermittlung der räumlichen Verhältnisse nicht vollständig gerecht. Spätestens die Nachbereitung und das Selbststudium werfen Fragen auf, die nur eine freie, dreidimensionale Betrachtung eines Modells oder Präparates klären könnte. Reale Modelle sind in beschränkter Zahl für den topographisch-anatomischen Unterricht vorhanden, aber eben nicht jederzeit frei zugänglich. Eher selten ergibt sich die Chance, sie außerhalb der Vitrine in die Hand zu nehmen und zu studieren. Darüber hinaus sind anatomische Präparate teuer und können nicht beliebig

zerschnitten, zerlegt und wieder zusammengesetzt werden. Ein anatomisches Praktikum, das dem haptischen Bedürfnis des Lernenden entgegenkommt, ist der Präparationskurs. Aber dieser dauert nur ein Semester und anatomische Fragen werden sich nach wie vor stellen. Dann werden meist Abbildungen konsultiert. Der dreidimensionale Eindruck muss durch eigene Vorstellungskraft und Kombinationsvermögen erschlossen werden. Diese wurden wohl deshalb früher in einem medizinischen Eignungstest besonders geprüft. Noch dringlicher als beim Problem der Räumlichkeit des fertig entwickelten Körpers wird die Forderung, moderne Technologien zu nutzen, wenn es um die Embryologie geht. Diese hat neben den drei Raumachsen noch einen vierten Freiheitsgrad, die Zeit der Entwicklung.

Visualisierung sollte also durch moderne Methodik erfolgen und die Daten sollten so präsentiert werden, dass sie der Anwender leichter auswerten und verstehen kann. Die Daten selbst werden durch Beobachtung oder Simulation der Realität gewonnen und einem Visualisierungsprozess unterworfen, der je nach Visualisierungstechnik ein Bild aus da Vincis Zeichenstift, ein Röntgenbild oder ein dreidimensionales virtuelles Modell aus dem Computer generiert. Virtuelle 3-D-Modelle können die Nachteile planer Abbildungen weitgehend beheben und sollten deshalb zunehmend – nicht als Ersatz bestehender Lehrmittel, sondern als vernünftige Ergänzung – eingesetzt werden. Was in diesen Tagen weitgehend fehlt, sind geeignete virtuelle Modelle und an die vorhandene technische Ausrüstung angepasste Visualisierungsprogramme.

Diese Arbeit verfolgt daher drei Ziele. Erstens wurde das Gehirn eines sieben Wochen alten menschlichen Embryos mit ausschließlich am Institut vorhandenen Mitteln dreidimensional rekonstruiert. Zweitens wurde ein Render-Programm entwickelt, das es erlaubt, an einem durchschnittlichen PC diese Rekonstruktion zu betrachten, zu drehen und in beliebigen Ebenen aufzuschneiden, um Innenansichten zu erhalten. Drittens sollte dieses Programm nach Möglichkeit so fortentwickelt werden, dass Rekonstruktionen richtiggehend stereoskopisch betrachtet werden können. Die Hauptaufgabe bei der Durchführung dieses Visualisierungsprozesses lag darin, die Hürden zu erkennen und mit programmiertechnischen Mitteln zu überwinden. Nachfolgende Projekte des Lehrstuhls können sich an dieser Arbeit orientieren und darauf aufbauend einfacher weitere Rekonstruktionen durchführen, um sie schließlich den Studierenden zur Verfügung zu stellen.

2 Material und Methoden

Aus der Sammlung Romeis/Bachmann/Schmitt der Anatomischen Anstalt der Universität München wurde das Präparat menschlicher Embryo Nr. 33 (ME33) mit einer Scheitel-Steiß-Länge von 16.9 mm, was einem Alter von ungefähr sieben Wochen entspricht, ausgewählt. Auf Grund der sichtbaren Fingerstrahlen und der angedeuteten Strahlung in der Fußplatte (Abbildung 1) befindet er sich im Carnegie-Stadium 18. ME33 ist in Paraffineinbettung mit einer Schrittweite von erst 7 µm in 46 Schichten (Objektträger 3 bis 12), dann mit 10 µm in weitere 528 Schichten (Objektträger 13 bis 100) in der Sagittalebene geschnitten und HE gefärbt (Abbildung 2). Objektträger 1 und 2 sind nicht vorhanden. Die Objektträger wurden feucht gereinigt und trockengerieben. Die Reihenfolge der Schnitte auf den einzelnen Objektträgern war nicht markiert. Diese Abfolge ließ sich durch Beachtung kleiner Strukturmerkmale zuverlässig rekonstruieren. Danach wurden alle Schnitte in ihrer ursprünglichen Reihenfolge durchnummeriert.

Abb. 1: Embryo (ME33).
Aufnahme des intakten Embryos, die der Schnittserie beigegeben war.

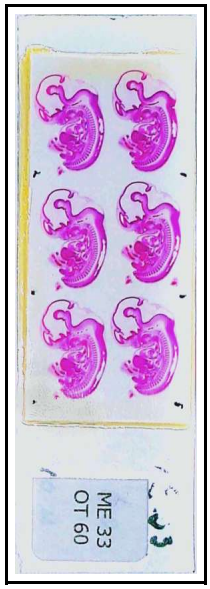


Abb. 2: Objektträger (OT 60).
Übersicht eines Objektträgers mit sechs Serienschnitten.

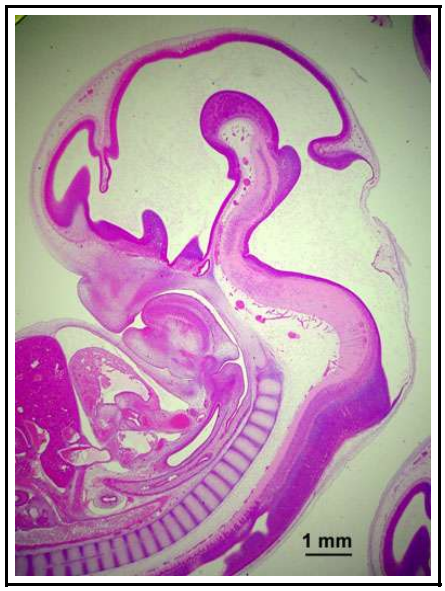


Abb. 3: Digitale Aufnahme (60_6).
Ausschnittvergrößerung mit Kopf und Brust, man beachte die großen Hohlräume des Ventrikelsystems.

Da die Schnitte (maximaler Durchmesser etwa 15 mm) sogar für die übliche Lupenvergrößerung des Mikroskops zu groß waren, wurde zunächst versucht, eine Auflichtapparatur einzusetzen. Die damit erhaltenen Bilder erlaubten jedoch keine ausreichende Nachvergrößerung. Deshalb kam schließlich doch ein Lichtmikroskop (Axiophot von Zeiss mit 3200 K-Lichtquelle und internen Grau- und Blaufiltern) zu Einsatz, und zwar mit einem 1,25-fach vergrößernden Objektiv bei ausgebautem Kondensor. Auf das Mikroskop wurde mittels eines Spezialtubus (Zeiss) eine

Digitalkamera (Cybershot Digital Still Camera DSC-S75 von Sony) aufgesetzt. Die Blende der Kamera konnte wegen des schon definierten Strahlengangs auf offen (entspricht hier 2,2) gestellt werden. Belichtungszeit, Fokus, Weißabgleich und ISO Empfindlichkeit wurden auf automatisch belassen. Als Bildformat wurde JPEG mit einer Auflösung von 2048 x 1536 bei einer Farbtiefe von 3 x 8-Bit gewählt. Das Zoom der Kamera wurde so eingestellt, dass der gesamte zur Verfügung stehende Bildbereich des Mikroskops erfasst wird. Der Zoomfaktor konnte konstant gehalten werden. An den Analogausgang der Kamera wurde ein LC-Display als zusätzliche Kontrolle angeschlossen. Mit dieser Vorrichtung konnten hochwertige digitale Aufnahmen der oberen Körperhälfte erstellt werden, die eine ausreichend detailreiche Auflösung boten (Abbildung 3).

Die Schnitte auf den Objektträgern 19 bis 100 wurden abfotografiert und über die USB-Schnittstelle auf ein Notebook übertragen. Dort wurden sie auf einen fünf Zeichen langen Dateinamen umbenannt (die Nummer des Objektträgers mit führenden Nullen, ein Unterstrich und die Nummer des Schnittes, z. B. 060_6.jpg). Zusätzlich wurde eine Folie mit 0.1 mm Raster (Metrik) und ein Bild ohne Objektträger (zur Beurteilung von Hintergrund und Ausleuchtung) fotografiert.

Vom Notebook wurden die Bilddaten auf das Zielsystem übertragen. Dieses war mit einem AMD Athlon 650 MHz (32-Bit CPU) auf einem Asus K7V (Mainboard) mit VIA VT8371 (KX133 Northbridge) und zwei 256 MiB PC133 SDRAM DIMMs (Hauptspeicher) ausgestattet. Als Grafikchipsatz wurde eine nVidia GeForce2 (GPU) ausgesucht, die auf einer Elsa Gladiac GTS 32 (Grafikboard) arbeitet und mit einem OpenGL-kompatiblen nVidia Detonator Treiber (ICD-konform) betrieben wurde. Als Betriebssystem diente Microsoft Windows 98 SE.

Die Vorarbeiten zur Rekonstruktion, also das Ausrichten und die Segmentierung der Schnitte, sowie die eigentliche 3-D-Rekonstruktion sollte unter Mercurys AmiraDev 3.0[1] erfolgen [Amira; Indeed 3D]. Die Entwicklerversion von „Amira“ war mit Hochschullizenz vom Leibniz-Rechenzentrum erworben worden und stand für ein Jahr als Mitarbeiterkopie zur Verfügung. Die zu dem Zeitpunkt aktuellen Bugfixes und Patches wurden eingespielt.

Das Rendering sollte mit selbst entwickelter Software unter OpenGL schnell, kompakt und unkompliziert geschehen. Der entwickelte Quellcode sollte ferner plattformunabhängig, zumindest aber leicht auf andere Systeme portierbar sein. Als Programmiersprache wurde C++ gewählt, der Code wurde mit g++ aus der GNU Compiler Collection 3.4.2 [gcc] in einer Minimalist GNU for Windows-Umgebung [MinGW] übersetzt. Die Freeware Dev-C++ 4.9.9.2 von Bloodshed wurde als Entwicklungsumgebung (IDE) gewählt [Dev-C++].

1 Im Folgenden kurz als „Amira“ bezeichnet.

3 Ergebnisse

3.1 Bildverarbeitung mit selbst entwickelter Software und „Amira“

3.1.1 Vorfilterung: Datenmenge und -reduktion

Das Rekonstruktionsprogramm „Amira“ versucht bei Erstellung eines Volumendatensatzes diesen sofort unkomprimiert in den Hauptspeicher zu laden. Aber die Rohdatenmenge der 492 Bilder von $2048 \cdot 1536 \cdot 3\,\text{Byte} \cdot 492 = 4428\,\text{MiB}$ [2] ist zu groß. Deswegen musste diese reduziert werden. Das sollte unter Einsatz eines Skriptes automatisiert für den gesamten Datenbestand erfolgen. 32-Bit-Zeiger können maximal $2^{32} = 4\,\text{GiB}$ adressieren. Unter der virtuellen Speicherverwaltung von Windows stehen einem Prozess 2 GiB zur Verfügung, in die anderen 2 GiB wird der Betriebssystemkern eingeblendet. Teile dieses virtuellen Adressraums werden je nach Bedarf auf den physischen Hauptspeicher abgebildet (Working-Set). „Amira“ wurde testweise mit Datensätzen verschiedener Größen (z. B. 1024 x 768er Auflösung) konfrontiert. Es bricht bei einem Speicherüberlauf mit Fehlermeldungen „Couldn't allocate n bytes“ oder einem Absturz ab. Ein Selektieren des nur teilweise geladenen Datensatzes (im Beispiel 341 statt 492 Bilder) in „Amira“ lässt auf die zulässige Maximalgröße der Daten schließen, da dort nur der allozierbare Anteil des Volumendatensatzes auftaucht. Diese Zahlen $1024 \cdot 768 \cdot 8\,\text{Bit} \cdot 341$ ergeben 256 MiB. Nach einigen Stabilitätstest[3] mit „Amira“, Überwachung der Speicherauslastung mit dem Konsolenbefehl „`app maxmalloc`“, Sysinternals „Process Explorer“ und Microsofts „Systeminformation“ wurde die obere Grenze der Datensatzgröße auf die berechneten 256 MiB festgelegt.

Da die Farbinformation der HE-gefärbten Schnitte für die Rekonstruktionsarbeit irrelevant ist, wurden die Bilder in 8-Bit Graustufen konvertiert, was die Datenmenge auf ein Drittel reduziert. Die Auflösung wurde mit Hilfe eines Anti-Aliasing-Filters (siehe Multisampling in Kapitel 5.1.1) auf 819 x 614 Pixel reduziert. Das Ausschneiden der Region-of-interest (ROI) hätte die Bilder auch verkleinert, dies muss jedoch manuell mit jedem einzelnen Bild und mit einer zusätzlichen Rotation zur Einpassung geschehen. Da in „Amira“ jedes Bild die gleiche Auflösung haben muss, wäre nicht einmal ein Drittel des Datensatzes eingespart worden. Die Auflösung wurde entlang der z-Achse nicht reduziert, um sich einer räumlichen Isotropie zu nähern, was zwar nicht unbedingt notwendig ist, jedoch zu ausgewogeneren Ergebnissen führt. Die Bilder wurden dann im unkomprimierten TIF-Format gespeichert. Für alle Schritte der Datenmengenreduktion wurde ein Skript erstellt, mit dessen Hilfe der Prozess für das gesamte Datenverzeichnis automatisch ablaufen konnte.[4]

2 x-, y-Auflösung, Farbtiefe und Anzahl
3 Der Segmentierungseditor fordert nochmals Speicher für seinen (genauso großen) Datensatz an.
4 Ein vergleichbarer Arbeitsgang kann mit Hilfe der Freeware XnView menügesteuert durchgeführt werden [XnView].

Bei genauer Sichtung der Bilder wurden solche ohne ZNS-Anteile, hier die ersten und letzten Bilder, mit einem `u` im Dateinamen als „uninteressant“ markiert (17 Stück). Weitere Bilder wurden bei sichtbaren Beschädigungen des Gehirns, wie dem Umklappen von Rändern oder dem Zerreißen von Strukturen mit einem `d` als „defekt“ markiert (36 Stück). Aus den verbleibenden 439 Schnitten entstand der endgültige Datensatz mit einer Größe von $819 \cdot 614 \cdot 8\,\text{Bit} \cdot 439 \approx 211\,\text{MiB}$. Er wurde somit auf ein Fünftel reduziert und hält die Obergrenze von 256 MiB ein.

3.1.2 Einlesen in „Amira“

Der reduzierte und korrigierte Datensatz konnte, unter anderem wegen der Lücken durch defekte Schnitte, nicht direkt über die von „Amira“ vorgesehene Importfunktion eingelesen werden. Um einen Import dennoch zu ermöglichen, musste nun ein Volumendatensatz mit korrekter Metrik erstellt werden.[5] Auch einer höheren Flexibilität wegen wurde eine info-Datei erstellt [Amira 3.0 Manual 2002]. Dieser Schritt sollte automatisiert werden. Den plattformabhängigen Teil (Dateinamen lesen) erledigte eine Skriptdatei (`make_info.bat`). Laut Konvention sollten die Bilder parallel zur x-y-Ebene, entlang der z-Achse angeordnet sein.

Der Parameter `pixelsize x y z` (`z` ist optional) der info-Datei bestimmte die Metrik der Voxel in µm, d.h. die Länge eines Voxels entlang der Basisachsen. Mit Hilfe einer Mikrometerschraube und des fotografierten Rasters erfolgte die Berechnung. Die gemessene Länge wurde durch die Anzahl der Pixel dividiert: $s_x = \frac{11.7\,\text{mm}}{819} \approx 14.3\,\mu\text{m}$ sowie $s_y = \frac{9.0\,\text{mm}}{614} \approx 14.7\,\mu\text{m}$. Der dritte Wert ergab sich aus der Schnittdicke des Mikrotoms, also ist $s_z = 10\,\mu\text{m}$. Ein Skript erzeugte die Datei `slices.info`, welche die Bilder aus einem Unterverzeichnis `images819/` referenzierte:

```
# slices.info
pathname images819
pixelsize 14.3 14.7 10.0
#
019_1u.tif
...
100_8u.tif
#
end
```

Fehlte noch die Auskommentierung (`#`) der entbehrlichen (mit `d` oder `u` markierten) Bilder. Da die Schrittweite entlang der z-Achse dadurch variierte, wurde die Metrik der z-Achse direkt bei den

5 „Amira“ erwartet bei direkten Import einer Schnittserie eine invariable Schichtdicke. Da dies nicht immer gegeben ist, kann mit der im Folgenden dargestellten Methode dennoch eine korrekte Metrik für „Amira“ erstellt werden.

einzelnen Bildern angegeben. Auf Grundlage von `slices.info` erledigte dies ein eigens entwickelter C++-Parser (`info_conv.exe`)[6]. Die info-Datei sah nun wie folgt aus:

```
# slices2.info
pathname images819
pixelsize 14.3 14.7
#
#019_1u.tif 0
...
027_6.tif   500
#028_1d.tif 510
028_2.tif   520
...
```

samt korrekter Metrik in µm. Bilder 19_1 und 28_1 wurden als Ausschuss erkannt und mit einem # auskommentiert. Die `pixelsize` definierte weiterhin die Einteilung der x- und y-Achsen. Die Einteilung der z-Achse wurde an den einzelnen Bildern inkrementiert.

3.1.3 Registrierung: Ausrichten der Schnitte

Das Ausrichten der Schnitte (Fachterminus: *Registrierung*) bringt die Bilder zur Deckung (siehe Kapitel 5.1.2.1). „Amira“ verwendete als Metrik für die Optimierung der Registrierungs-Funktion (Modul „Compute AlignSlices“) die Minimierung der Abweichungsquadrate (*SSD*)[7]. Dafür sollten jedoch – was realiter selten gegeben ist – die zugrunde liegenden Graustufenhistogramme normiert sein. Im Ergebnis brachte „Amira“ zwei Folgeschnitte schlechter zur Deckung wie es unter visueller Kontrolle manuell möglich war. Deshalb wurde getestet, ob die Funktion durch Streckung der Graustufenhistogramme auf einen normierten Bereich (0x00 bis 0xFF) und Weichzeichnung (3^2 Gauß) verbessert werden konnte. Aber auch damit war die Qualität einer manuellen-visuellen Ausrichtung nicht erreichbar. Ein weiteres von „Amira“ angebotenes Verfahren zur Schnittausrichtung benutzt Landmarken. Da „Amira“ zur Schnittverschiebung nur Translation und Rotation (rigide Transformation) zur Verfügung stellte, genügten im Prinzip zwei Landmarken. Sicherheitshalber wurden im Test vier möglichst weit voneinander entfernt liegende, anatomisch definierte Landmarken gesetzt. Aber auch dieses Verfahren unterlag der manuellen-visuellen Methodik, selbst bei anschließend durchgeführter Feinausrichtung über die Minimierung der SSD.

Wegen der unbefriedigenden Ergebnisse dieser Tests erfolgte die Schnittausrichtung ausschließlich manuell-visuell. Die Klassifikation nach van den Elsen et al. 1993 ergab zwei Dimensionen, intrinsische volle pixel-basierte Registrierung und globale rigide Transformation

6 Die finale info-Datei könnte auch in einem Durchlauf erzeugen werden. Dazu müsste jedoch eine nicht-portierbare Bibliothek eingebunden werden, was der Vorgabe der Plattformunabhängkeit widerspricht.

7 analog zu Formel (2) auf Seite 38

voller Interaktion über SSD-Minimierung einer Modalität. Der Volumendatensatz wurde über `slices2.info` in „Amira“ geladen und interaktiv ausgerichtet. Additive Fehler wurden vermieden, indem der Volumendatensatz überwacht wurde: Zum einen wurde die annähernde Spiegelsymmetrie sichergestellt, indem Abweichungen, die sich beim Vergleich korrespondierender Schnitte der rechten und linken Gehirnhälfte ergaben, visuell identifiziert und manuell korrigiert wurden. Zum anderen wurden in den Volumendatensatz immer wieder virtuelle Schnittebenen um die Transversalachse gelegt und darin die anatomische Konsistenz überprüft, etwa die Konkavität der Innenfläche der Seitenventrikel[8] (Abbildung 4a).

Mit das größte Problem bei der Ausrichtung histologischer Serienschnitte resultierte aus der von Schnitt zu Schnitt leicht variablen Stauchung (vom Schneidevorgang) und Streckung (vom Wasserbad). Dies war mit einer rigiden Transformation, wie sie „Amira“ bot, nicht zu lösen, da sie weder lineare noch elastische Verzerrungen kompensieren konnte. Deshalb wurde nach einer Möglichkeit gesucht, die groben Ausreißer zu definieren und zu eliminieren. Dazu wurde wieder manuell-visuell ausgerichtet, und zwar ausschließlich im Hinblick auf das Gehirn.[9] Dies musste auf Kosten anderer Strukturen geschehen, wie Herz oder Leber, die schlecht ausgerichtet blieben. Abbildung 4a zeigt einen Querschnitt durch den kompletten Datensatz, der wegen der noch vorhandenen Ausreißer sehr unruhig wirkte. Diese störten nicht nur die gesamte Struktur, sondern verschlimmerten die Fehler, die sich bei der Ausrichtung unweigerlich addierten. Beim Durchblättern entlang der z-Achse wurden grobe Ausreißer identifiziert und deren Slice-Nummer protokolliert. Die fertige Protokolldatei `filter1.txt` sah folgendermaßen aus:

```
9
19
21
25
26
39
43
...
```

Slice 9 (entspricht Bild `021_3.tif`) war verzerrt, aber Slice 8 hätte gut auf Slice 10 ausgerichtet werden können. Wenn also ein Slice besonders gestreckt oder gestaucht erschien, die Nachbarn jedoch besser aufeinander passen würden, so sollte der Ausreißer dazwischen aus dem Volumendatensatz entfernt werden. Dies war in „Amira“ nicht direkt möglich. Daher wurde mit einem weiteren C++-Programm `filtered.exe` das auszusortierende Slice mit dem vorhergehenden

8 oder eine zylindrische Struktur eine solche bleibt
9 Das geht in Richtung einer Umwandlung von einer globalen in lokale (bessere) Transformationen.

überschrieben. Danach lagen Schnitte mit tolerabler Verzerrung aneinander und konnten gut ausgerichtet werden. `filtered.exe` erzeugte dazu die Datei `filtered1.info`:

```
...
021_1.tif    90          # slice 7
021_2.tif    100         # slice 8
021_2.tif    110         # slice 9
021_4.tif    120         # slice 10
...
```

Ausreißer-Slice Nummer 9 wurde mit dem vorhergehenden überschrieben (statt Bild `021_3.tif` nun `021_2.tif`). `filtered1.info` wurde in „Amira“ geladen und die Transformationen des letzen Datensatzes übertragen. Die protokollierten Stellen konnten nun sauber ausgerichtet werden. Hierbei wurden 94 Bilder und bei einer nochmaligen Kontrolle weitere 18 überschrieben und zu ihren Nachbarn ausgerichtet. In Abbildung 4b ist im Vergleich zu Abbildung 4a zu sehen, dass grobe Ausreißer beseitigt wurden und die Qualität so sichtbar verbessert wurde.

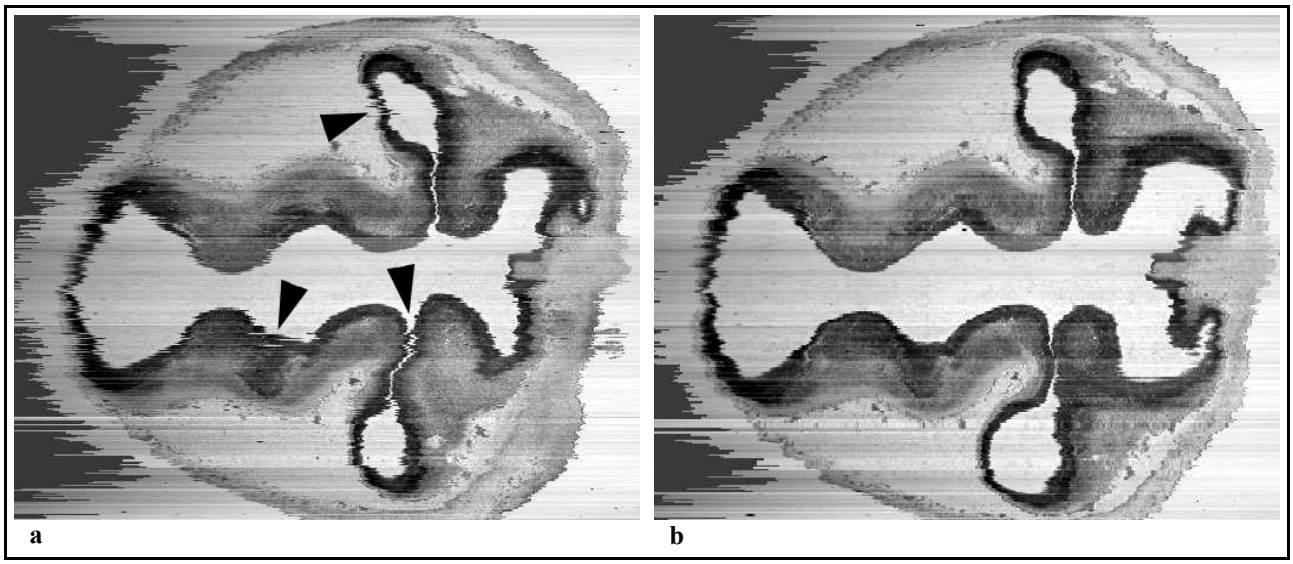


Abb. 4: Querschnitt durch das registrierte Volumen.
Vor (**a**) und nach (**b**) Spezialfilterung. Das Ergebnis in **b** zeigt weniger Fransen als **a** (**Pfeile**), die Ränder wirken glätter. (Da der Volumendatensatz dadurch verändert wurde, weichen die Schnitte leicht voneinander ab.)

3.1.4 Segmentierung: Markierung der relevanten Strukturen

Die Maskierung oder Markierung (Fachterminus: *Segmentierung*) des Datenvolumens ordnet den Voxeln entsprechende anatomisch-topologische Informationen zu, um später eine Oberflächenrekonstruktion durchzuführen. Markiert wurde das ZNS ab der dünnen, aber leicht identifizierbaren Schicht der Arachnoidea als Grenze. Nach innen wurden sinnvollerweise die Ventrikel ausgespart, sie werden so im transparenten oder angeschnittenen Modell als Hohlräume sichtbar.

Obgleich es erstrebenswert schien, die gewünschte Struktur (das Gehirn) in den Schnitten selektiv mit Hilfe von Werkzeugen der Bildverarbeitung hervorzuheben, führte keiner der Versuche mit verschiedenen Filtern, wie Schärfen, Kantenverstärkung des Bildes oder Thresholding des Histogrammes zu befriedigenden Ergebnissen. Von den von „Amira“ im Segmentierungseditor bereitgestellten Funktionen erwies sich nur das „Blow-out-tool“ als hilfreich, freilich auch nur bei dicken, gut kontrastierten homogenen Gehirnanteilen, wie im Hirnstammbereich. Feine Strukturen, etwa das Dach des Rhombencephalons und sämtliche Randbegrenzungen, mussten komplett händisch nachgezeichnet werden. Die Markierung wurde Bild für Bild ohne automatische Interpolation durchgeführt. Nur bei den 112 überschriebenen Bildern wurde eine Interpolation notwendig, aber auch hier erwies sich manuelles Vorgehen der Automatik überlegen.[10]

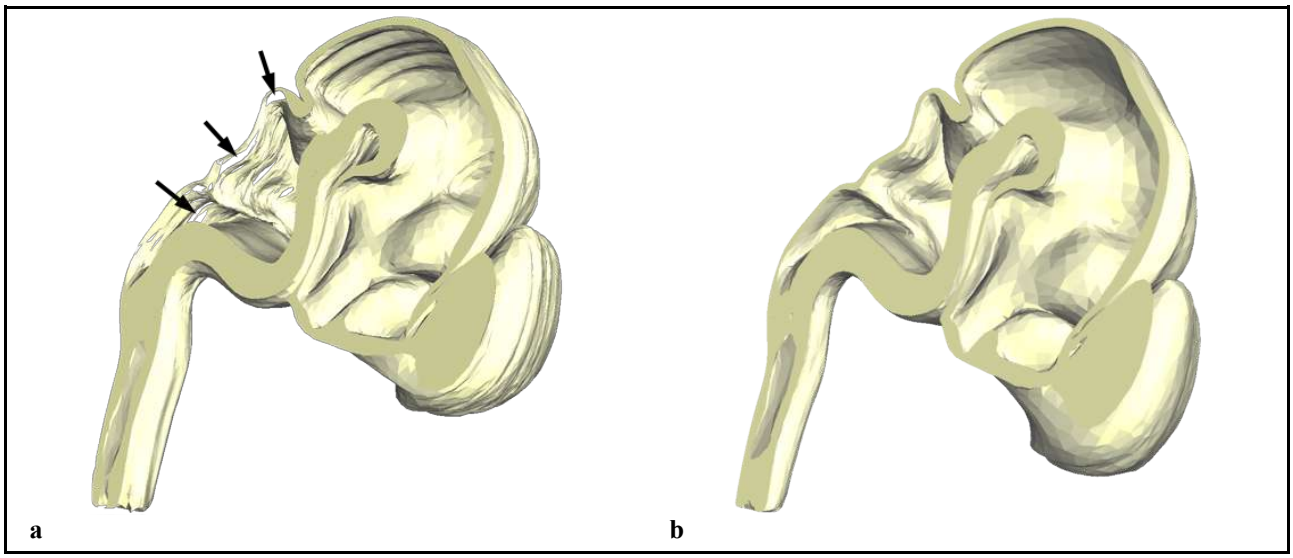


Abb. 5: Schließen der Lücken im Dach des Rautenhirns (aufgeschnittenes Gehirn).
Während in **a** die Löcher (**Pfeile**) irritieren, sind sie in **b** nicht mehr zu finden. (**b** wurde zusätzlich unabhängig davon geglättet, wie die fehlende sagittale Riffelung erkennen lässt.)

Bei feinen Strukturen ergab sich ein weiteres Problem: Sie tendierten dazu, vom Marching-Cubes-Algorithmus übergangen oder bruchstückhaft zu werden, was zu Rissen und Löchern oder zur Unterschlagung von Strukturen wie Gehirnnerven in der Rekonstruktion führte. Um solche Gebilde bestimmt und komplett rekonstruierbar zu erfassen, musste eine Strichstärke ab 10 (bei doppelter Vergrößerung) eingesetzt werden. Manche anatomischen Strukturen wurden dabei inkorrekt vergröbert. Nerven wurden folglich weggelassen. Risse im Dach des vierten Ventrikels wurden durch Identifizierung in der 3-D-Ansicht und nochmalige Verdickung der Strichstärke – bis alle Nachbarschnitte in diesem Bereichen überlappten – geschlossen, wie in Abbildung 5 illustriert.

10 Arbeitssparender Hinweis: Die Markierung eines Schnittes kann für den nächsten/vorigen Schnitt mit der undokumentierten Tastenkombination „`Shift-Space/Backspace`“ übernommen werden, um sie dann nur noch marginal anpassen zu müssen.

3.1.5 Mapping: 3-D-Rekonstruktion

Die eigentliche 3-D-Rekonstruktion (Fachterminus: *Mapping*) erzeugte aus dem segmentierten Datenvolumen eine Oberfläche. „Amira's“ Methode, die Oberfläche zu rekonstruieren, basierte auf dem Marching-Cubes-Algorithmus. Erzeugt wurde ein Netz aus Dreiecken, welches für das spätere Rendering optimal war. Da die Berechnung der Rekonstruktion auf der Basis des Datenvolumens mit $819 \cdot 614\,\text{Byte} \cdot 439 \approx 211\,\text{MiB}$ [11] mangels Speicher abbrach, musste das Volumen auf $409 \cdot 307\,\text{Byte} \cdot 439 \approx 53\,\text{MiB}$ verkleinert werden (Resampling). Die anschließende Funktion „Surface-Generation“ wurde mit eingeschalteten „existing Weights“ sowie „add Border“ durchgeführt. „Existing Weights“ bewirkte die Berechnung auf Subpixel-Ebene, „add Border“ garantierte eine geschlossene Oberfläche. Das resultierende Modell bestand aus rund 3 Millionen Dreiecken bei rund 1 Million Punkten. Für ruckelfreies Rendering der Rekonstruktion wurde die Anzahl der Dreiecke reduziert (auf 25,000) und mit einer Iterationstiefe von 30 geglättet. Diese Arbeitsschritte machten das Modell nicht nur leichter, sondern entfernten auch den Großteil der störenden sagittalen Rillen entlang der z-Achse (Abbildungen 5 und 6).

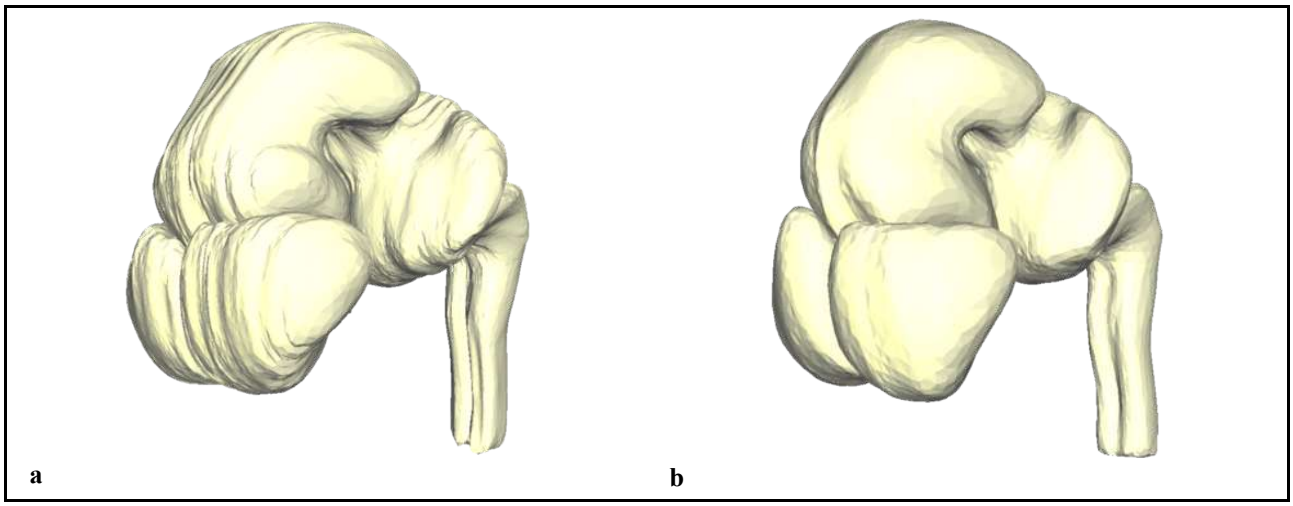


Abb. 6: Beeinträchtigung durch sagittale Rillen.
Die unruhigen sagittalen Kerben aus **a** sind in **b** behoben.

3.2 Rendering: Rekonstruiertes Gehirn in „Amira“

Die geometrischen Primitiven wurden mit dem Computer dargestellt (Fachterminus: *Rendering*). Als Ergebnis der Rekonstruktion zeigt die Abbildung 7 das Gehirn des Embryos ME33. Die Bilder wurden in „Amira“ perspektivisch gerendert, die Oberflächennormalen flossen „per-vertex“ und nicht nur „per-triangle“ in die OpenGL-Pipeline ein (Funktion „Vertex“).

11 Der Marching-Cubes-Algorithmus benötigt mehr Speicher als das Ausrichten und die Segmentierung, da die vielen Zwischenergebnisse und die erzeugten Dreiecke (Fließkommazahlen) einigen Platz in Anspruch nehmen.

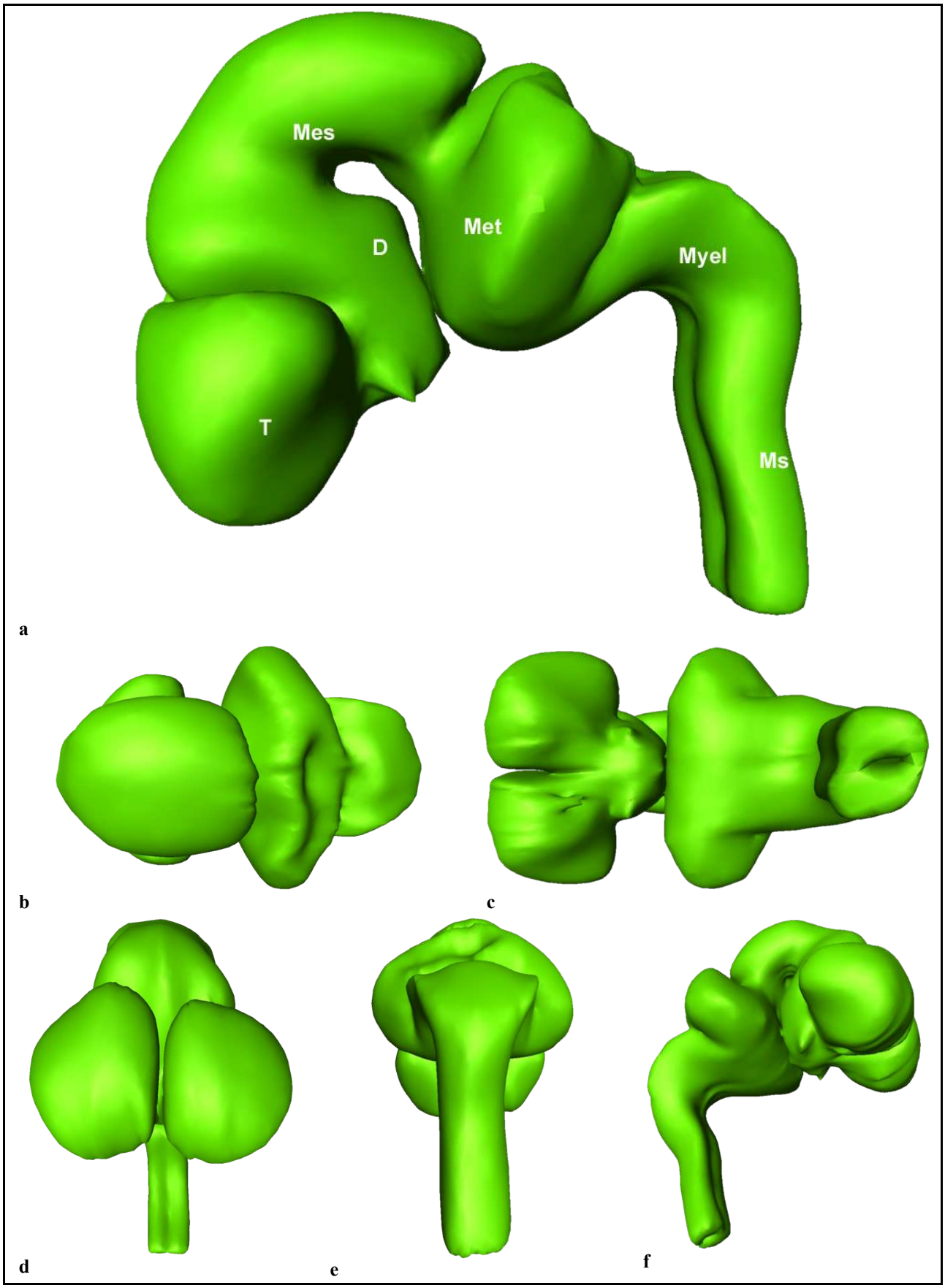


Abb. 7: Gehirn des Embryos ME33.
Seiten- (**a**), Dorsal- (**b**) und Ventralansicht (**c**), sowie von frontal (**d**), occipital (**e**) und eine freie Ansicht (**f**).
T Tel-, **D** Di-, **Mes** Mes-, **Met** Met-, **Myel** Myelencephalon, **Ms** Medulla spinalis

Die Grundgliederung in Rhomb- und Prosencephalon bzw. in Myel-, Met-, Mes-, Di- und Telencephalon ist deutlich erkennbar. Das Mesencephalon ist dem Entwicklungsstadium entsprechend groß (7a und b). Der Boden des Diencephalon zeigt median die Verbindung zur Hypophyse und lateral die Stümpfe der Nervi optici (7a, c und f).

Auf der rechten Endhirnhemisphäre wurde ein Teil des Bulbus olfactorius rekonstruiert, links wurde der Ansatz angedeutet (7c). Das Dach des vierten Ventrikels ist aufgrund der histotechnischen Präparateherstellung eingesunken (7a, b und e). Durch die perspektivische Projektion treten dem Betrachter nahe Strukturen größer hervor als entfernte. Daher stehen die Hemisphären in Abbildung 7d so besonders hervor.

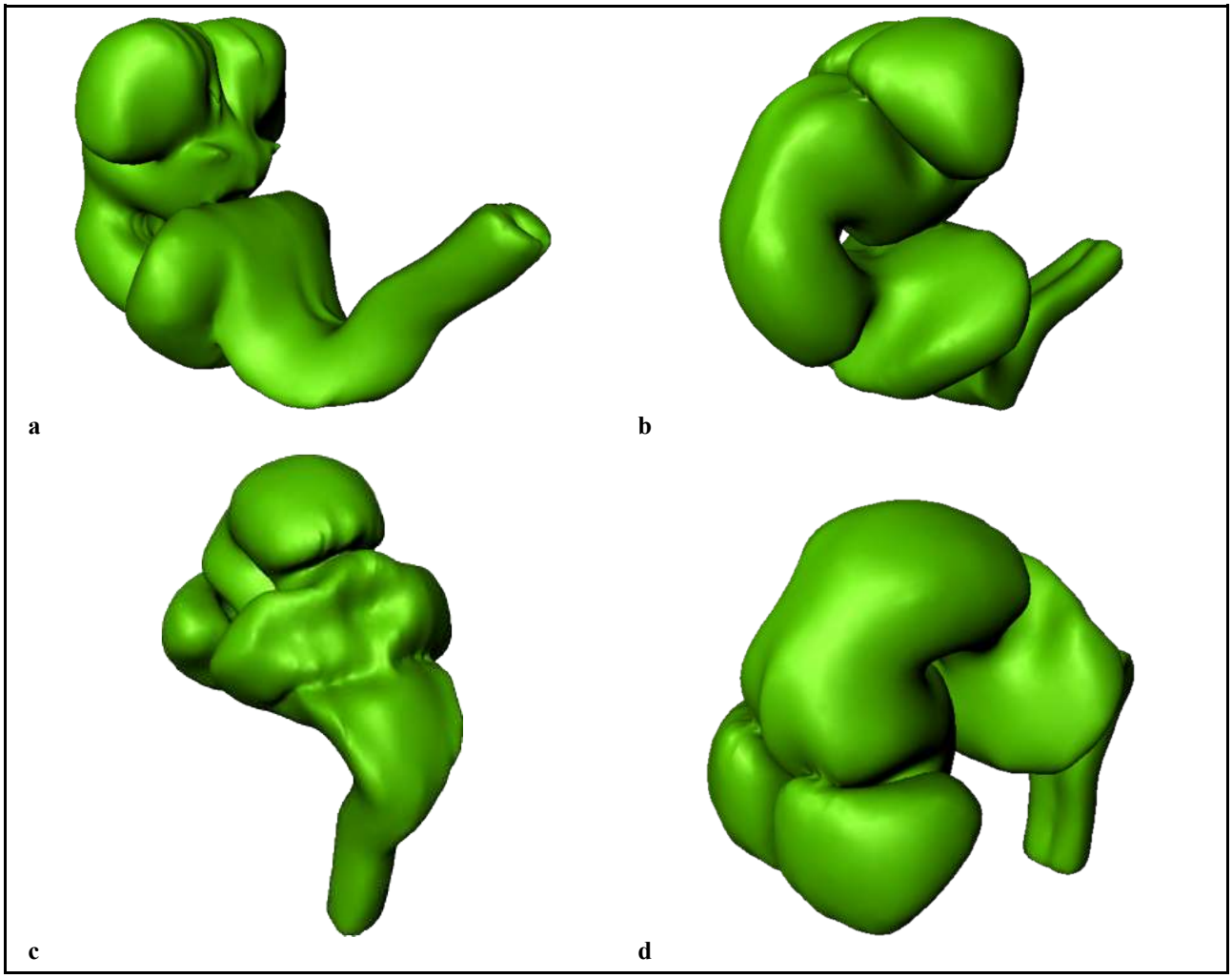


Abb. 8: Schrägansichten des Modells.
Von rechts unten hinten (**a**), von rechts oben vorne (**b**), von links oben hinten (**c**) und von links oben vorne (**d**).

In Abbildung 8 sind verschiedene schräge Blickwinkel auf das Modell dargestellt. Die erhaltenen Bilder stimmen voll mit dem überein, was auf Grund herkömmlicher Lehrbuchabbildungen zu erwarten war. Erst weitere Möglichkeiten zeigen den Vorsprung des virtuellen 3-D-Modells gegenüber den planen Bildern: Drehen des Modells in Echtzeit, Stereoskopie sowie das Potential, das

Modell in beliebigen Ebenen aufzuschneiden und hineinzublicken (Abbildung 9). Diese Funktionen können in der vorliegenden Papierversion der Arbeit nur dokumentiert, nicht wirklich demonstriert werden. Erst die selbst entwickelte Software kann jeder testen (Kapitel 4.7.5.3).

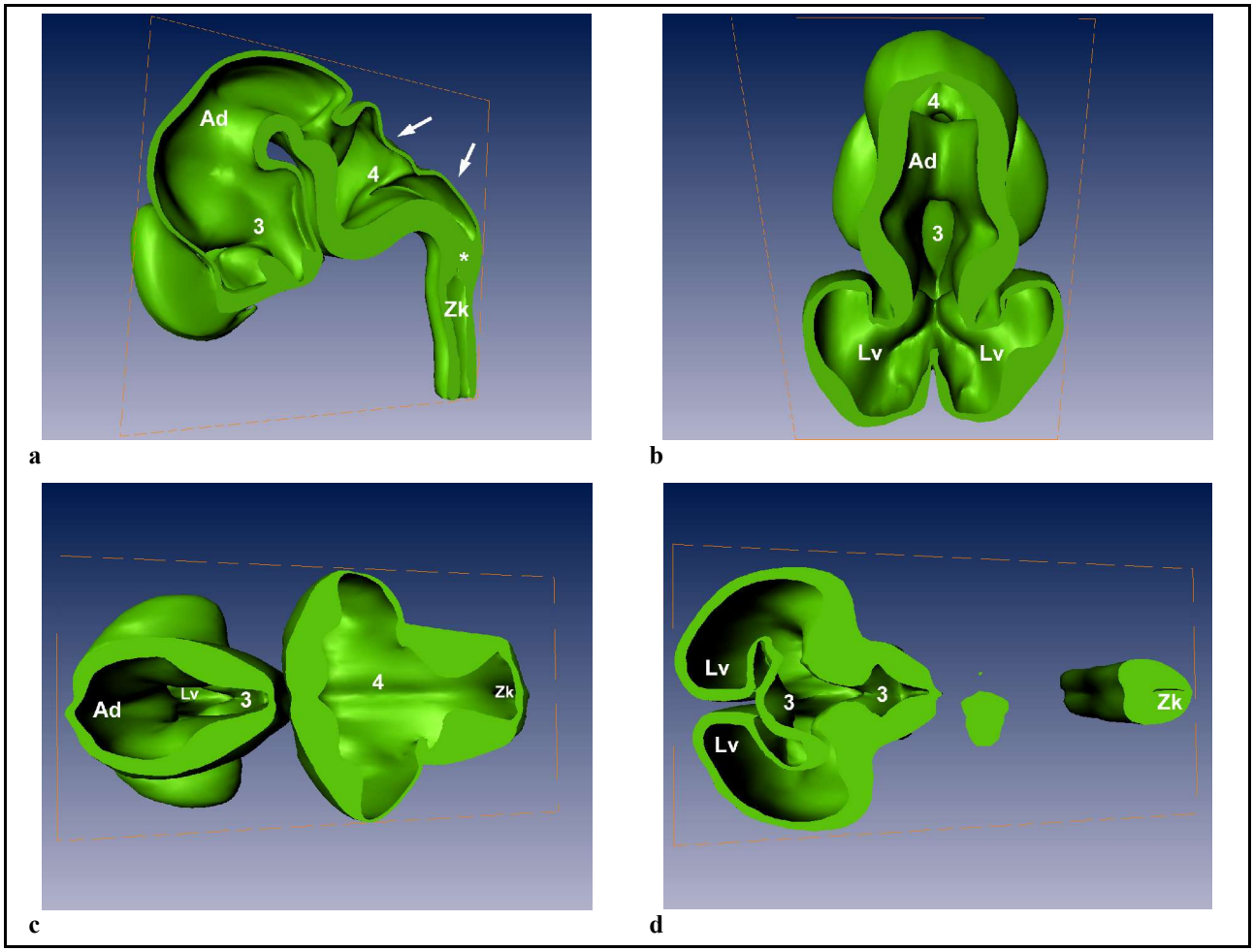


Abb. 9: Einblick in das Modell mit Darstellung des Ventrikelsystems.
Schnitt in der Medianebene (**a**), in einer Frontalebene (**b**) und Schnitte in Horizontalebenen (**c** weiter dorsal gelegen und deshalb das Metencephalon mit erfassend, **d** unterhalb des Metencephalons).
Lv Lateralventrikel, **3** dritter Ventrikel, **4** vierter Ventrikel, **Ad** Aquaedukt, **Zk** Zentralkanal
(Die Pfeile zeigen auf das dünne Dach des Rautenhirns; ***** hier geht der Schnitt durch die linke Seite des Gehirns, die Verbindung von Zentralkanal und vierten Ventrikel verläuft hinter der Schnittebene.)

In Abbildung 9 ist das aufgeschnittene Modell zu sehen. Abbildung 9a zeigt eine Schrägansicht von links vorne auf einen mediosagittal gelegten Schnitt, der die großen dritten und vierten Ventrikel sowie den weiten Aquaedukt eröffnet (auch in b und c). Die Verbindung zum rechten Lateralventrikel liegt rechts von der Schnittebene (sichtbar in d). Der Anschnitt im Übergang vom Myelencephalon zum Rückenmark lässt den Verlauf des Zentralkanals erkennen. Um nun solche Modelle auch ohne „Amira“, mit einer durchschnittlichen Hardware-Ausstattung und geringem Aufwand darstellen, drehen und schneiden zu können, wurde ein neues komfortableres, freies Visualisierungsprogramm entwickelt.

3.3 Entwicklung des Visualisierungsprogramms

3.3.1 Spezifikation

Das Rendering setzt die geometrischen Informationen zu einem dreidimensionalen virtuellen Modell zusammen, beleuchtet es und projiziert es auf den Computerbildschirm. Das fertige Modell sollte mit einem flexiblen, kompakten und schnellen Programm angezeigt werden. Die Spezifikation der Software berücksichtigte die Anforderungen im Hörsaal und die Möglichkeiten des studentischen PCs und erlaubte eine einfache Distribution.

Die Software sollte klein sein, damit sie leicht verteilt werden kann und noch auf einen Wechseldatenträger passt (etwa 3.5" Diskette mit 1440 KiB). Die Software sollte auch auf einem schreibgeschützten Medium ohne Installation und ohne Administratorrechte lauffähig sein. Sie sollte plattformunabhängig implementiert werden, also auf Windows, Linux, Unix oder andere Computersysteme übersetzbar sein. Der Quellcode sollte gut organisiert sein, um schnelles Übersetzen der Einheiten zu gewährleisten. Das Design sollte einzelne Einheiten klar trennen und Abstraktionsebenen beinhalten. Moderne Techniken wie die objekt-orientierte Programmierung (OOP) und die generische Programmierung sollten ein durchdachtes Design ohne Performance-einbußen gewährleisten. Daher wurde als Programmiersprache C++ bestimmt, das sich in diesen Disziplinen als Vorreiter gezeigt hat [BS ISO/IEC 14882:2003 (2nd)].

Für die Grafikschnittstelle fiel die Wahl auf OpenGL. Als verbreitete und professionelle Schnittstelle war sie offen standardisiert, gut dokumentiert und auf vielen Plattformen implementiert [OpenGL]. Als Framework für Callbacks wurde das offene und frei verfügbare LibSigC++ (Lizenz LGPL) gewählt [LibSigC++; Fischer 2001]. Es war aus dem Framework für grafische Benutzeroberflächen (GUI) Gtk-- hervorgegangen und semantisch sicher wie sonst keine andere Callback-Bibliothek implementiert (abgesehen von BOOST [Boost]). LibSigC++ setzt einen standardkonformen Compiler voraus, der sicher mit C++-Templates umgehen kann. Weder Trolltechs QT noch Microsofts .NET konnten solche Callback-Routinen vorweisen.

Im zu entwickelnden 3-D-Viewer sollte die Kommunikation mit OpenGL und mit dem Endanwender über den Signal/Slot-Mechanismus von LibSigC++ abgewickelt werden. Ausser OpenGL und LibSigC++ wurde keine Fremdbibliothek benutzt. Um die Daten aus „Amira“ zu exportieren und sie in die selbst entwickelte Software zu importieren, wurde das binäre Standard-Triangulation-Language-Format (STL) als gemeinsame Schnittstelle ausgesucht [Feng 1997].

3.3.2 Software-Design

Wie in größeren Projekten üblich, wurde der Quelltext in Übersetzungseinheiten unterteilt und diese in Unterverzeichnissen strukturiert. Abhängigkeiten wurden minimiert oder vollständig vermieden. So ist der mathematische Teil (`math`), welcher Vektoren, Punkte, Matrizen, Ebenen und vieles mehr implementiert, völlig unabhängig vom Rest. Im Gegensatz zu anderen Bibliotheken wird es vermieden, OpenGL-Header einzubinden[12]. Sogar Konzepte wie Materialeigenschaften, Farben oder Kamera (`glo`) konnten abstrahiert und völlig unabhängig implementiert werden. Die OpenGL-Schnittstelle wurde durch eine Abstraktionsebene abgeschirmt. Dazu wurde ein Wrapper (`glw`) entwickelt, der die C-Schnittstelle von OpenGL auf ein sauberes Design in C++ abbildet. Erfahrungen aus der hardwarenahen laufzeitkritischen Programmierung sicherten dieses Design vor Geschwindigkeitseinbußen ab. Die Paradigmen der objektorientierten und generischen Programmierung, wie Kapselung oder Typendefinitionen, wurden streng eingehalten[13]. Die Fehlerbehandlung wurde komplett durch Ausnahmen mit throw-Spezifikation realisiert. Sie machen die Ausnahmebehandlung für den Entwickler übersichtlich und bieten dem Compiler Optimierungsmöglichkeiten. Diese fortschrittliche Technik fand sich in sonst kaum einer Bibliothek (in „Amira“ nur teilweise, in GLT oder QGLViewer gar nicht). Die throw-Spezifikationen tauchten sonst nirgends auf[14]. Kostspielige dynamische Allokation auf dem Heap und virtuelle Methoden konnten umgangen werden. Explizite Typumwandlungen wurden aufgrund ihrer Unberechenbarkeit vermieden oder mit `static_cast` durchgeführt. Im Folgenden werden einige Aspekte vorgestellt.

3.3.3 Mathematische Algorithmen

Auf Bibliotheken aus dritter Hand, die höchstperformante multi-threading Frameworks für lineare Algebra anbieten, wie Blitz++, MTL oder Pooma, wurde verzichtet. Diese waren vollständig generisch geschrieben, die Übersetzungszeit wäre daher immens lange gewesen. Expression Templates fordern den Compiler heraus und können den Code aufblähen. Diese Techniken werden eingesetzt, um mit den hochoptimierten Fortran 77/90-Bibliotheken zu konkurrieren. Sie waren zu allgemein implementiert und boten keine speziellen Optimierungen für 3-D-Grafik. Blinde Verwendung solcher Bibliotheken ohne Not und ohne Beachtung der speziellen Rechnerarchitektur (L1-, L2-, L3-Cache, Multiprozessor) und Profiling ist nicht zu empfehlen. Kleinere Bibliotheken, wie GLT oder libQGLViewer, spezialisiert auf 3-D-Programmierung, hatten Designschwächen und Performancelücken. Daher kam selbst entwickelter handoptimierter Code zur Anwendung.

12 GLT bindet etwa in `vector3.cpp` ungeschickterweise `gl.h` und `glu.h` ein [GLT].
13 Was in den meisten Bibliothekten, wie libQGLViewer, nicht der Fall ist [libQGLViewer].
14 Obwohl man sie nicht mehr als zu speziell bezeichnen kann, denn in Java sind sie Pflicht.

3.3.3.1 Punkte

Exemplarisch wird point3.cpp auszugsweise herausgegriffen. Diese Übersetzungseinheit enthält die Implementation für einen geometrischen Punkt (Kapitel 5.2.3). Die 3 bezeichnet die Komponenten p_x, p_y und p_z. Die homogene Komponente h ist fest auf 1 gestellt und belastet den Stack nicht. Sie wird jedoch dem Nutzer transparent zur Verfügung gestellt, er sieht von außen eine (4,1)-Matrix. Das Einführen der homogenen Komponente als Klassenelement, wie in einigen anderen Grafikbibliotheken, wäre, wie gleich gezeigt, nur auf den ersten Blick von Vorteil. Die Typdefinition value_type bereitet die Klasse, wie alle anderen Klassen dieser Bibliothek, auf generische Typisierung vor. Die Header-Datei (point3.h) kann wie folgt skizziert werden:

```
class point3
{
public:
      // TYPES
      typedef float value_type;

      // 'TORS
      point3(void) throw();
      point3(const value_type &x, const value_type &y,
            const value_type &z) throw();

      // DATA ACCESS
      const value_type& x(void) const throw() { return pt_[0]; }
      const value_type& y(void) const throw() { return pt_[1]; }
      const value_type& z(void) const throw() { return pt_[2]; }
      const value_type  h(void) const throw() { return 1; }
      void x(const value_type &s) throw() { pt_[0] = s; }
      void y(const value_type &s) throw() { pt_[1] = s; }
      void z(const value_type &s) throw() { pt_[2] = s; }
      // used as fast interface to 3D driver only, deprecated otherwise
      const value_type* const data(void) const throw() { return pt_; }

      // FUNCTIONS
      point3& operator+=(const vector3 &v) throw();
      point3& operator-=(const vector3 &v) throw();

private:
      // DATA MEMBERS
      value_type  pt_[3];
};

inline const point3 operator+(const point3 &p) throw() { return p; }
inline const point3 operator+(const point3 &p, const vector3 &v) throw();
inline const point3 operator+(const vector3 &v, const point3 &p) throw();
inline const point3 operator-(const point3 &p, const vector3 &v) throw();
template <typename charT, typename traits> std::basic_ostream<charT, traits>&
      operator<<(std::basic_ostream<charT, traits> &os, const point3 &p)
      throw(std::ios_base::failure);
```

Gutes Software-Design überprüft und meldet schon zur Übersetzungszeit fehlerhafte Semantik und gibt einen Syntax- oder Bindefehler aus. So wird verhindert, dass ein Objekt in einem falschen Kontext benutzt wird, ungültige Werte oder unsinniges Verhalten aufweist, die dann erst zur Laufzeit Fehler produzieren. So ist es selbstverständlich mathematisch möglich, zwei Punkte zu addieren, die homogene Komponente wäre dann 2. In einer geometrischen Anwendung soll dies jedoch nicht passieren, die homogene Komponente darf nur die Werte 0 oder 1 annehmen[15]. Daher gibt es die Klassen `point3` und `vector3`, deren Schnittstellen (unter anderem) so entworfen wurden, dass es möglich ist, zwei Vektoren zu summieren, die Addition zweier Punkte jedoch einen Übersetzungsfehler verursacht. So würde direkt auf einen möglichen Programmierfehler hingewiesen. Auch die konsequente Nutzung von `const` im Rückgabewert fördert semantischen Unsinn, wie `(a+b) = c`[16], sofort als Fehler ans Tageslicht [Meyers 2006]. Noch während die Render-Software dieser Arbeit geschrieben wurde, konnten diese Designvorteile genutzt und Bugs mit zeitraubender Fehlersuche vermieden werden. Der Code wurde so für die weitere Benutzung um einiges sicherer.

3.3.3.2 Matrizen

Das Einfrieren der homogenen Komponente bzw. der vierten Zeile einer (4,4)-Matrix hat entscheidende Performancevorteile. Mathematische Operationen können optimiert werden, die Anzahl der Basisoperationen sinkt, die Prozessorbelastung nimmt ab. Der Code wird dabei um die 25% schneller. Eine besonders aufwendige Herausforderung ist die Berechnung der Inversen einer (regulären) Matrix. Sie wird in Render-Anwendungen häufig gebraucht, um zwischen Koordinatensystemen zu wechseln. Sonderfälle gehen leicht von der Hand: Die Inverse einer Translation berechnet sich durch Negierung dreier Elemente der letzten Spalte. Einfach ausgedrückt ist die Inverse einer Skalierungsmatrix durch den Kehrbruch dreier Elemente der Diagonalen. Auch eine Rotation (orthogonale Matrix) kann durch Transposition der Elemente leicht rückgängig gemacht werden (Formeln 6 bis 8 auf Seite 72). Sehr viel komplizierter sieht der allgemeine Fall der Inversion einer (4,4)-Matrix aus (Formel 9). Benötigt werden ohne Optimierung *eine Division*, über *200 Multiplikationen* und um die *100 Additionen*. Da Redundanzen vorkommen, kann dieser Rechenaufwand durch geschicktes Umstellen erheblich minimiert werden. GLT etwa benötigt *eine Division*, *92 Multiplikationen* und *47 Additionen* für die allgemeine Routine und *eine Division*, *43 Multiplikationen* und *20 Additionen* sowie einigen Overhead für die spezialisierte (affine) Routine. Die im Rahmen dieser Arbeit entwickelte Routine verwendet nur *eine Division*, *36 Multiplikationen* und *17 Additionen* ohne Overhead, was die regelmäßig benötigte Inversion höchst effizient macht.

15 Ausnahmen, wie bei der perspektivischen Projektion, werden in der OpenGL-Pipeline transparent durchgeführt.

16 Wird oft versehentlich statt `(a+b) == c` in Bedingungen verwendet, was bei impliziter Konvertierung in `bool` sonst nun mal kein Syntaxfehler wäre. Solch ein Fehler müsste dann ohne erwähnte Semantik mühsam gesucht werden.

3.3.4 Codeoptimierung

Um schnellstmögliche Laufzeitgeschwindigkeit zu erlangen, wurden weitere Techniken verwendet: Übergeben von Referenzen statt von Variablen, konsequente Verwendung des Schlüsselwortes `const` und Umgehung der Erzeugung temporärer Variablen (auch impliziten auf dem Stack). Zugriffe auf den Hauptspeicher und den (assoziativen) Cache wurden so angeordnet, dass „Thrashing“[17] minimiert wurde. Das bedeutete zwar größere Aufmerksamkeit bei der Implementierung, aber auch einen Geschwindigkeitsvorteil. Einige Programmiertechniken wurden von industriell erprobten numerischen Bibliotheken [Blitz++; MTL; Pooma] übernommen, die in besonders laufzeitkritischen Anwendungen wie der Astronomie eingesetzt werden [Vandevoorde, Josuttis 2003].

Dazu gehört unter anderem das loop-unrolling. Profiling ergab beim Kopieren einer Matrix über loop-unrolling einen Wert von 6.590[18], während eine normale Schleife mit 11.040 und der Befehl der C-Standardbibliothek `memcpy()` mit 10.880 deutlich mehr Zeit benötigten. Auffällig war das Versagen von `memcpy()`, das wie die anderen `mem...()` Funktionen zur „Manipulation von Objekten als Zeichenvektoren gedacht“ und „Schnittstelle zu effizienten Routinen“ ist [Kernighan, Ritchie 1990]. Ein interessehalber durchgeführtes Reverse-Engineering brachte die Ursache ans Tageslicht: Die Funktion `memcpy()` wird von `msvcrt.dll` exportiert. Zwar wird der Bereich wie erwartet über den Stringbefehl `REP MOVSD` kopiert. Jedoch steht vor und nach dieser Schleife ein größerer und nutzloser Overhead: Es wird kontrolliert, ob sich die Bereiche überlappen. Dies ist unnötig, der C-Standard setzt dies bei `memcpy()` voraus. (Für überlappende Bereiche gibt es den Befehl `memmove()`.) Weiter wird geprüft, ob die Pointer auf 32-Bit-Adressen ausgerichtet sind. Auch dies sollte in einer effizienten Implementierung unterlassen werden. Sind die Pointer nicht ausgerichtet, so funktioniert der Stringbefehl `MOVSD` trotzdem, nur langsamer (so wie `MOVS`) [Intel 1995]. Legt der Entwickler Wert auf höchste Geschwindigkeit, wird er seinen Compiler veranlassen, die Daten auszurichten.

Eine effiziente Implementierung sollte direkt den breitesten Stringbefehl des Prozessors nutzen (am besten inline) und bei immerhin 128 zu kopierenden Bytes[19] die höchste Geschwindigkeit im Vergleich zu den beiden Hochsprachenmethoden aufweisen. Auch ließe sie sich dann leichter und

17 Assoziativer Cache ist der Einfachheit halber aus Lines aufgebaut (z. B. 32 Byte beim Pentium), die fortlaufenden Hauptspeicheradressen cachen. Daher ist es günstig, direkt aufeinanderfolgende Speicheradressen zu verarbeiten. Wird dies nicht getan, wird also nur ein Wert einer Line benötigt und gleich ein Intervall weitergesprungen, muss die Line wieder überschrieben werden. Dies bezeichnet man als *Thrashing*. So scheint `t[3] = t[7] = t[11] = 0;` nur auf den ersten Blick elegant. Es geht also nicht nur darum, schnellen Maschinencode zu erzeugen, sondern diesen so anzuordnen, dass er möglichst schnell ausgeführt werden kann. Dabei werden heute die Speicherzugriffe optimiert.

18 je niedriger, desto schneller und desto besser

19 Die (4,4)-Matrix wurde aus `double` Elementen zusammengesetzt, die hier jeweils 8 Byte belegen.

sicherer auf eine 64-Bit-Architektur portieren. Da `memcpy()` aus der Runtime von Microsoft jedoch nicht so effizient implementiert wurde, kam in der Eigenentwicklung loop-unrolling zum Einsatz, das keinen Overhead mitbringt. In anderen Kontexten kann auch leichter parallelisiert werden.

Ähnliches zeigte sich auch für die OpenGL-Schnittstelle. Regel 3 der Performance Tipps des „Red Book“ [Woo et al. 1999] empfiehlt statt Routinen aus dritter Hand OpenGL's Matrizen-Routinen (`glRotate()`, `glTranslate()`, `gluLookAt()`, ...) zu verwenden. Profiling ergab jedoch, dass die Eigenentwicklungen schneller arbeiten. Daher wurden alle laufzeitkritischen Routinen von einem Wrapper auf selbst entwickelte, handoptimierte, effizientere Routinen abgebildet (`rotate()`, `translate()`, `look_at()`, ...). Diese profitieren darüber hinaus von der sicheren Semantik, die das Software-Design zur Übersetzungszeit liefert. So erwartet `look_at()` zwei Punkte `point3` und einen Vektor `vector3` und nicht beliebige drei Felder.

3.3.5 C++ IOStream-Inserter Bug

Inserter fügen eine Instanz eines Datentyps als formatierten, internationalisierten String in einen Stream ein. Einmal für einen Datentyp implementiert, geschieht dies transparent innerhalb der gesamten I/O von C++ und Bibliotheken, die den Konventionen folgen. Die vom Autor entwickelte Bibliothek hält die C++-Konventionen ein. Die IOStream-Inserter aus der Bibliothek des hier vorgestellten Programms weichen dennoch vom C++-Standard ab. BS ISO/IEC 14882:2003 (2nd) schlägt unter 26.2.6-15 lib.complex.ops folgende Implementierung vor (exemplarisch für eine komplexe Zahl) [siehe auch ISO C++-Draft Standard 1996; Langer, Kreft 2003]:

```
template<class T, class charT, class traits>
basic_ostream<charT, traits>&
operator<<(basic_ostream<charT, traits>& o, const complex<T>& x)
{
      basic_ostringstream<charT, traits> s;
      s.flags(o.flags());
      s.imbue(o.getloc());
      s.precision(o.precision());
      s << '(' << x.real() << "," << x.imag() << ')' << ends;[20]
      return o << s.str();
}
```

Wenn der Stringstream `s` (durch `bad_alloc`) fehlerhaft wird, so kann der Benutzer der Routine dies nicht feststellen. Das Fehlschlagen des Inserters darf jedoch keinesfalls ignoriert werden. Es könnte so bei einer Dateioperation zu Datenverlust führen. Um einen derartig schweren Fehler erkennen und korrigieren zu können, wurde vom Autor das Fehlerbit `failbit` im Ausgabestream `o` gesetzt:

20 Nur am Rande sei vermerkt, dass auch das Einfügen eines Zeichens `','` statt der Zeichenkette `","` besser wäre.

```
if (s.good())
      return o << s.str();     // OK
else
{
      o.setstate(std::ios_base::failbit);  // Error
      return o;
}
```

Nun wird das Fehlerflag beim Aufrufer ankommen und falls gewünscht[21] eine Ausnahme geworfen. Es wurde `failbit`, nicht `badbit` gewählt, da der Stream `o` nicht zerstört wurde und weiter benutzt werden kann. Er wird unverändert zurückgegeben. So wird das C++-Designparadigma, das besagt, das Auftreten eines Fehlers soll die Daten unkorrumpiert im ursprünglichen Zustand belassen und eine Ausnahme werfen, eingehalten. Im aktuellen ISO/IEC-Standard wird dies nicht korrekt berücksichtigt. Da dieses Verhalten sinnvoll und eben erwartungsgemäß ist, wurde dem Experten der C++-Kommission, der mit der Bibliotheksentwicklung betraut ist, vom Autor dieser Änderungsvorschlag des Standards übergeben und diskutiert und wird in der nächsten Revision C++0x beachtet.

Eine zu erwartende, vermeintlich elegantere Implementation eines Inserters ohne Zuhilfenahme eines `stringstream`s

```
template<class T, class charT, class traits>
basic_ostream<charT, traits>&
operator<<(basic_ostream<charT, traits>& o, const complex<T>& x)
{
      return o << '(' << x.real() << ',' << x.imag() << ')' << ends;
}
```

ist bei genauerer Überlegung fehlerhaft und daher keine Alternative: Manipulatoren wie `setw()` oder `setfill()`, die die Formatierung einer Ausgabe über Flags eines Streams kontrollieren, wirken sich zumeist ausschließlich auf die nächstfolgende Ausgabeoperation aus. Es sollen jedoch alle Ausgabeoperationen eines Inserters, hier also `x.real()` und `x.imag()`, entsprechende Formatierungen erhalten. Daher ist der Umweg über den `stringstream` vonnöten.

21 Standardmäßig wird in dieser Situation nur eine Ausnahme geworfen, wenn die Maske mittels des Members `exception(ios_base::failbit)` gesetzt wurde. Auch der `stringstream s` würde im Fehlerfalle nicht von sich aus eine Ausnahme werfen und daher würde der Fehler nie beim Aufrufer landen.

3.3.6 Stereoskopie

Die gerenderten Bilder vermitteln durch Beleuchtung, Schattierung, Überlappung oder perspektivische Projektion nur einen plastischen Eindruck, sind jedoch faktisch zweidimensional (*monoskop*). Solche Bilder können ohne wahrnehmbaren Verlust genauso gut monokular betrachtet werden. *Stereoskopie* soll die wahrhaftige dreidimensionale Wiedergabe räumlicher Objekte ermöglichen. Ob die Objekte dadurch realistischer werden, sei dahingestellt, die meisten Probanden empfinden den Eindruck jedoch als beeindruckendes Erlebnis.

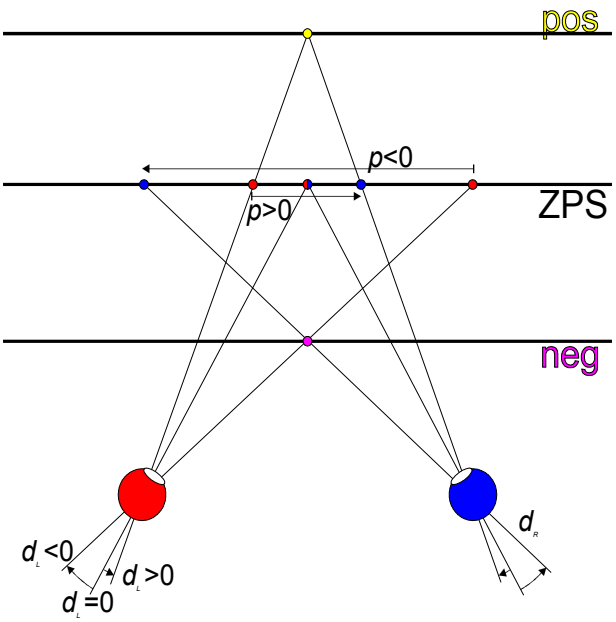


Abb. 10: Retinale Disparität und Parallaxe.
Die Disparität *d* ist die Summe der jeweiligen Winkel im rechten und linken Auge, die Parallaxe *p* ist der Abstand homologer Punkte.
rot linkes Auge, **blau** rechtes Auge, **gelb** positive Ebene, **magenta** negative Ebene, **ZPS** zero parallax setting

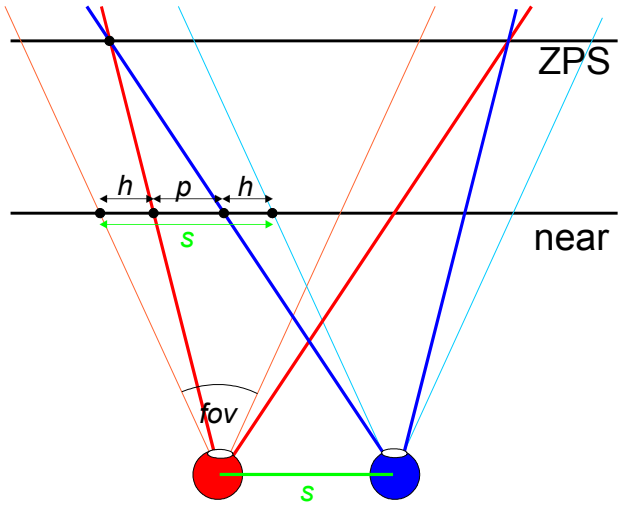


Abb. 11: Asymmetrische perspektivische Projektion.
Geometrischer Zusammenhang stereoskopischer Größen. **s** Augenseparation, **p** Parallaxenextremum, **h** horizontal-image-translation (HIT), **fov** field-of-view (Öffnungswinkel), **hellrot** symmetrische Projektion, **rot** asymmetrische Projektion, **near** vordere Projektionsebene

Dazu wird jedes Auge mit einer eigenen perspektivischen Projektion versorgt. Die sich durch die leicht unterschiedlichen Projektionen ergebende Parallaxe[22] (*p* in Abbildung 10) bedingt zwei Bilder, deren homologe Punkte leicht verschoben sind. Das Gehirn verschmilzt die binokulären disparen Bilder zu einem Bild, die retinale Disparität (*d* in Abbildung 10) zwischen den Bildern liefert die Tiefenwahrnehmung (*Stereopsie*)[23]. Gelingt dies dem Gehirn nicht, entstehen Doppelbilder[24].

22 Bei einer Parallaxe von 0 (homologe Punkte kommen zur Deckung) befindet sich der Punkt in der Bildschirmebene. Negative Parallaxen lassen Punkte aus dem Bildschirm heraustreten, positive Parallaxen setzen Punkte hinter die Bildschirmebene.
23 Dies wurde 1838 von Wheatstone erkannt.
24 die auch durch sogenanntes Übersprechen entstehen (z. B. Nachleuchten des Phosphors im Bildschirm o. ä.)

Physiologisch unterscheidet sich Stereoskopie prinzipbedingt von der Betrachtung eines realen Objektes. In der realen Welt fokussieren die Augenlinsen den Betrachtungspunkt (Akkommodation), auf den die Augenachsen konvergieren (Fixation). In der virtuellen Welt wird diese natürliche Konvergenzreaktion gestört: Die Linsen stellen immer den Bildschirm scharf, während die Achsen auf den imaginären Punkt im Raum konvergieren. In Abbildung 10 konvergieren die Achsen auf den gelben oder magentafarbenen Punkt, die Linsen fokussieren jedoch die Projektionsebene (ZPS). Da das Mißverhältnis zwischen akkommodativer Konvergenz und Akkommodation unphysiologisch ist, können bei einigen Personen asthenoptische Beschwerden, wie Kopfschmerz oder Ermüdung, auftreten. Um das Betrachten angenehm zu gestalten, sollten einige Richtlinien eingehalten werden: Wichtig ist, die retinale Disparität und damit die Parallaxe am Bildschirm zu limitieren. Üblich ist eine Parallaxe $|\beta| = \left|2 \arctan\left(\frac{p}{2a}\right)\right| \leqslant 1.5°$[25]. Auch kann ein Objekt, das vor dem Bildschirm liegt, realiter nicht von diesem verdeckt werden. Also ist bei negativen Parallaxen möglichst darauf zu achten, dass das Objekt nicht vom Bildschirmrand beschnitten wird. Desweiteren darf nur in der Ebene eine Parallaxe auftreten, in der sich die Augen befinden. So wird in der Stereoskopie eine horizontale Parallaxe verwendet, die vertikale Parallaxe muss 0 sein[26].

Für die mathematische Verwirklichung der Stereoprojektion läge es auf der Hand, für die zweite Perspektive das Objekt einfach leicht um die Vertikalachse zu drehen. Dabei wird bei perspektivischer Projektion die unerwünschte vertikale Parallaxe eingeführt[27]. Ausgeklügelter ist es, eine *asymmetrische perspektivische Projektion* (Abbildung 11) einzusetzen, bei der keine vertikale Parallaxe auftritt [Lipton 1997] und daher in dieser Arbeit zur Anwendung kam. Dabei wird die normale perspektivische Projektion mit einer horizontalen Verschiebung kombiniert. Diese komplizierte Transformation kann ebenfalls mit einer (4,4)-Matrix beschrieben werden.

Leider erwartet stereoskopische Software vom Benutzer oftmals, die stereoskopisch relevanten Parameter h, s und ZPS einzugeben. Diese werden neben den üblichen monoskopischen Parametern (near, far, fov, ...)[28] benötigt, um die Projektionsmatrizen zu berechnen. Da kaum ein Benutzer etwas mit h, s oder ZPS anzufangen weiß, wurden vom Autor spezielle Gleichungen aufgestellt. Diese vermögen es allein aus Größe und Auflösung des Bildschirms[29] bzw. der Leinwand und dem Abstand des Benutzers, Werte zu berechnen, die die ZPS in die Mitte des Objekts setzen, den Parallaxenbereich voll ausnutzen und die Parallaxe auf oben erwähnten Bereich beschränken.

25 Die Parallaxe wird bei gegebenem Abstand *a* des Betrachters auch als Winkel β angegeben.
26 Neigt man den Kopf bei einer stereoskopischen Darstellung, wird die 3-D-Wiedergabe eingeschränkt.
27 Anspruchslose Programme verwenden trozdem diese Methode wegen der leichten Realisation.
28 definieren den Pyramidenstumpf (frustum)
29 Diese liefert GLUT standardmäßig über `glutGet(GLUT_SCREEN_WIDTH_MM)` und ähnliche.

Aufgrund der großen Anzahl von Parametern und Gleichungen werden nur drei zentrale Formeln genannt, die nicht über den hohen Entwicklungs- und Implementationsaufwand hinwegtäuschen dürfen. Es ist $s = 2h + p$, der Strahlensatz liefert $h = \frac{s}{2}\left(1 - \frac{zps - near}{zps}\right)$ und $s = near\,\frac{p}{zps - near}$ (Abbildung 11). Die Metrik der realen Welt wird dann in das virtuelle Weltkoordinatensystem übertragen und man erhält die gesuchten stereoskopischen Parameter. Darüber hinaus muss der sogenannte Viewport so gesetzt werden, dass Objekte verzerrungsfrei dargestellt werden und etwa eine Kugel nicht zum Ei wird.

Die Kameraseparation befindet sich in der Software-Referenz [CrystalEyes SDK] fälschlicherweise in der Projektionstransformation. Dies mag die Implementierung einer Stereoskopie vereinfachen, resultiert jedoch in einer nicht ganz korrekten Beleuchtung. Die Beleuchtung wird im VCS[30] der Pipeline durchgeführt. Daher gehört die komplette Kamerakonfiguration (Blickpunkt und -richtung) samt Separation (die den Blickpunkt verändert) in die Viewing-Transformation. Erst jetzt sieht jedes Auge spiegelnde Reflektionen leicht anders, wie es durch die Realität vorgegeben ist. So wird der 3-D-Eindruck in der selbstentwickelten Software mathematisch sauber und besonders realistisch.

Die Software rendert in der Stereooption über vier Frame-Puffer. Dies sind jeweils Vordergrund- und Hintergrundpuffer für das rechte und linke Auge. Da nur wenige, teure Hardware vier Puffer unterstützt (quad-buffered), wurde zusätzlich das Anaglyphenverfahren implementiert. Dieses kommt ohne spezielle Hardware aus. Der Bildinhalt für das linke Auge wird in Rot, für das rechte Auge in der Komplementärfarbe Türkis (= Cyan, Blaugrün) gezeichnet. Durch die Verwendung dieser Komplementärfarben bleibt der Gesamtfarbeindruck (Addition erzeugt Weiß) erhalten und es können Rot-Grün- ebenso wie Rot-Blau-Brillen verwendet werden (Abbildungen 14 und 15). Dies sind zwei Vorteile ohne zusätzliche Nachteile gegenüber der Darstellung in Rot-Grün.

3.3.7 Leistungsspektrum dieses Visualisierungsprogramms

Das Modell wird von zwei Lichtquellen beleuchtet. Abbildung 12 zeigt das an verschiedenen Stellen aufgeschnittene Gehirn. Die Schnittebene kann mit der Maus einfach verändert werden, um eine geeignete Einstellung zu wählen. Die Schnitte hier wurden absichtlich so gewählt, wie sie auch Lehrbücher in Zeichnungen zeigen [etwa Sadler, Langman 1998]. So konnte die Rekonstruktion gut überprüft werden.

30 Render-Pipelines verwenden mehrere Koordinatensysteme (CS), die durch Transformationsmatrizen verknüpft sind. OpenGL arbeitet sich nacheinander vom **O**bject-**CS** über das **W**orld-**CS**, dem **V**iewing-**CS**, dem **C**lipping-**CS** und dem **N**ormalized-**D**evice-**CS** bis zum **D**evice-**CS** vor.

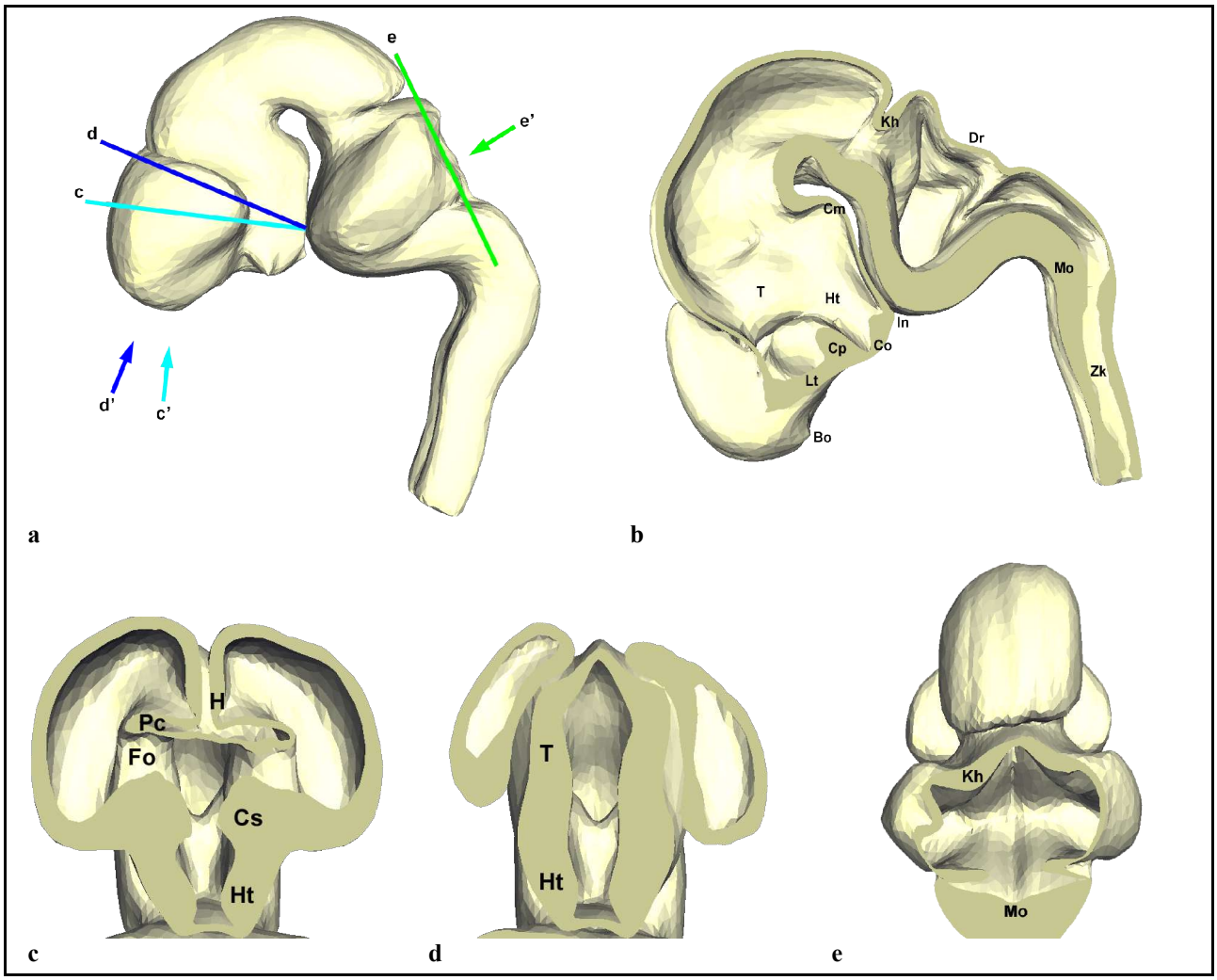


Abb. 12: Verschiedene Schnittansichten von ME33.
Die Linien in **a** zeigen die Schnittführung durch das Prosencephalon (**c** und **d**) und durch das Dach des 4. Ventrikels (**e**). Die **Pfeile (c', d'** und **e')** geben die jeweilige Blickrichtung an. **b** schneidet durch die Medianebene.
T Thalamus, **Ht** Hypothalamus, **Cs** Corpus striatum, **H** Hippocampus, **Pc** Plexus choroideus, **Kh** Anlage des Kleinhirns, **Cm** Corpus mamillare, **In** Infundibulum, **Co** Chiasma opticum, **Cp** Kommissurenplatte, enthält die Commissura anterior, **Lt** Lamina terminalis, **Bo** Bulbus olfactorius, **Fo** Foramen Monroi, **Mo** Medulla oblongata, **Dr** Rautenhirndach, **Zk** Zentralkanal

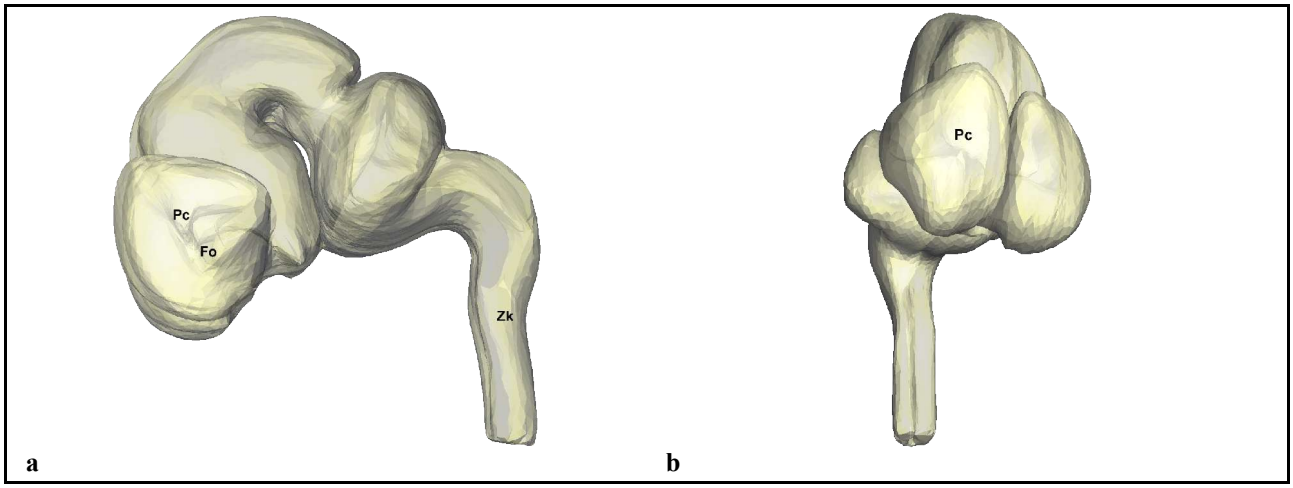


Abb. 13: Transparente Darstellung.
Man beachte, wie innen liegende Strukturen durchscheinen. Der Effekt ist im Modell deutlicher erkennbar als hier in der 2-D-Darstellung. Zwei verschiedene Transparenzstärken kamen zur Verwendung (**a** alpha = 0.4, **b** alpha = 0.6).

Abb. 14: Anaglyphische Bilder des Modells.
Sie erscheinen mit einfachen Rot-Grün- oder Rot-Blau-Brillen plastisch. Ansicht von seitlich (**a**) und schräg von hinten (**b**). Durch **b** wurde dann eine beliebige Schnittebene gelegt, die den 4. Ventrikel sowohl in seinem kaudalen Ende als auch in der seitlichen Ausdehnung nach rechts eröffnet (**c**).

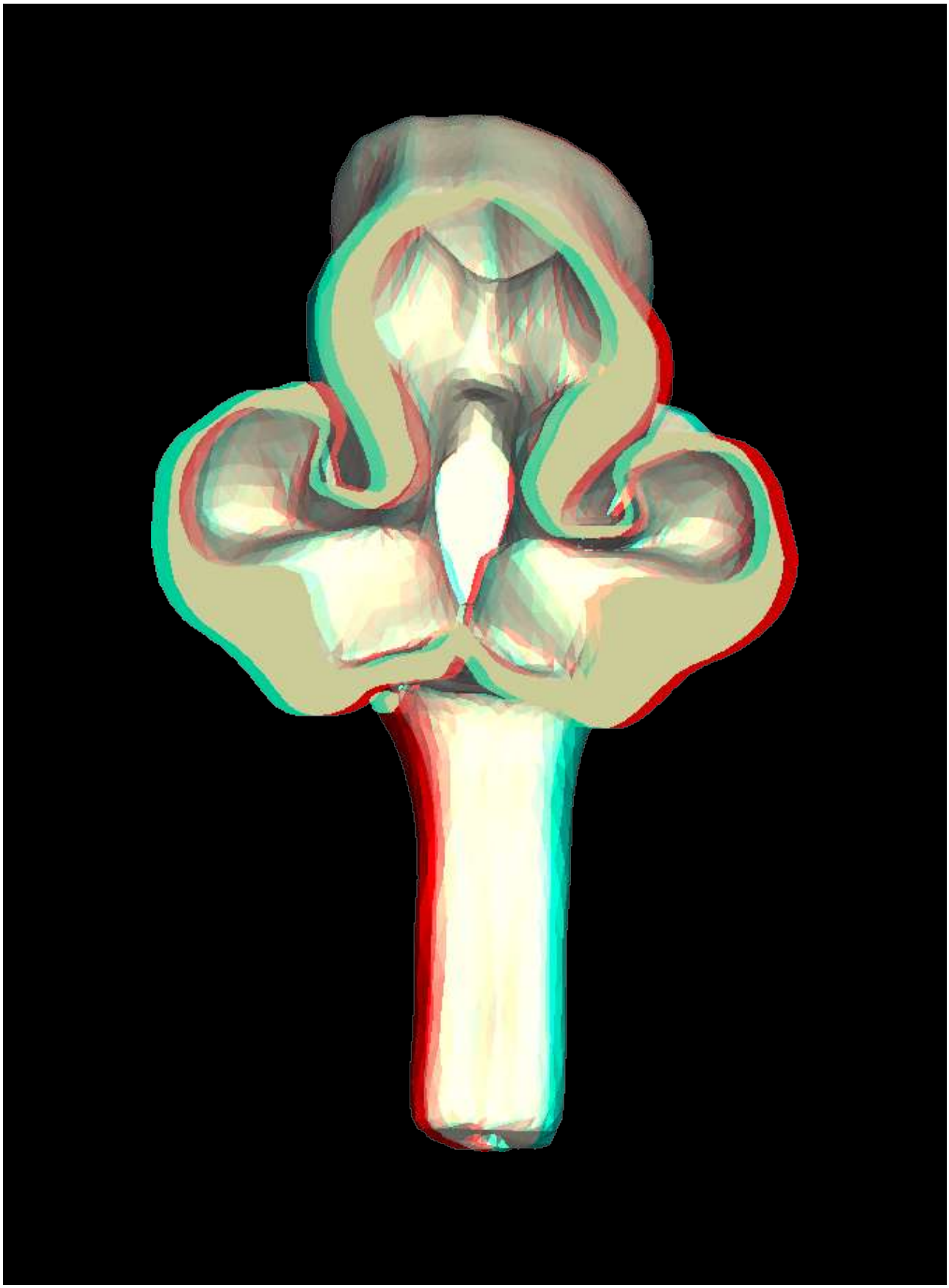
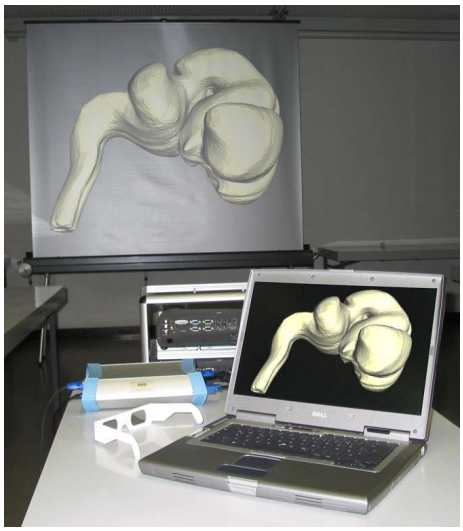


Abb. 15: Anaglyphisches Bild des Modells.
Man erkennt die verschiedenen Parallaxenverschiebungen für rechtes (cyan) und linkes (rot) Auge. Blick von vorne auf das frontal aufgeschnittene Gehirn.

Während in Abbildung 12 das Modell noch opak ist, wird in Abbildung 13 die Fähigkeit demonstriert, es transparent darzustellen. Die großvolumigen inneren Liquorräume können nun betrachtet werden. So sieht man, wie sich die Anlage des Plexus choroideus in die ersten beiden Ventrikel stülpt. Leider kommen die Darstellungen im Screenshot nicht so gut zur Geltung wie im Software-Programm. Dort kann das Modell gedreht werden, wodurch man eine noch bessere Orientierung gewinnt.

Die Fähigkeit der Software zur stereoskopischen Darstellung kann in der gedruckten Version zumindest über das Anaglyphenverfahren demonstriert werden (Abbildungen 14 und 15). Die Software kann aber auch mit komplexeren Verfahren stereoskopische Projektionen rendern. Die Abteilung Virtuelle Realität und Visualisierung des Leibniz-Rechenzentrums der Bayerischen Akademie der Wissenschaften bietet hierfür ausgefeilte Hardware [Brossmann 2006a], die zur Verfügung stand oder vom Autor konstruiert wurde (siehe Abbildung 17). Im Leibniz-Rechenzentrum wurde die Software und das Modell auf mehreren Veranstaltungen genutzt, etwa Schulungen oder der Einweihungsfeier [Brossmann 2006b]. Abbildung 16 zeigt die entwickelte Software in einigen Anwendungsbeispielen.

Die Fotografien aus Abbildung 16 können den plastischen Eindruck natürlich nicht wiedergeben, das Modell wirkt daher unspektakulär flach. Tatsächlich tritt das dargestellte Gehirn plastisch aus der Projektionsanlage heraus und kann in Echtzeit interaktiv gedreht und geschnitten werden. Die fertige Software kann mitsamt Gehirnmodell vom Server des Leibniz-Rechenzentrums heruntergeladen werden (ftp://ftp.lrz-muenchen.de/transfer/anatomieIII/). Man kann auch eigene Rekonstruktionen und Modelle, so sie im binären STL-Format (little-endian für Intel-compatible Prozessoren) vorliegen, laden. Die Bedienung der Software wird in Kapitel 4.7.5.3 auf Seite 50 vorgestellt.

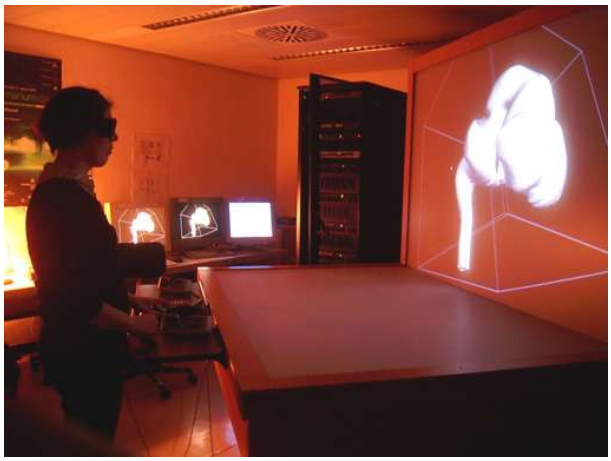


a

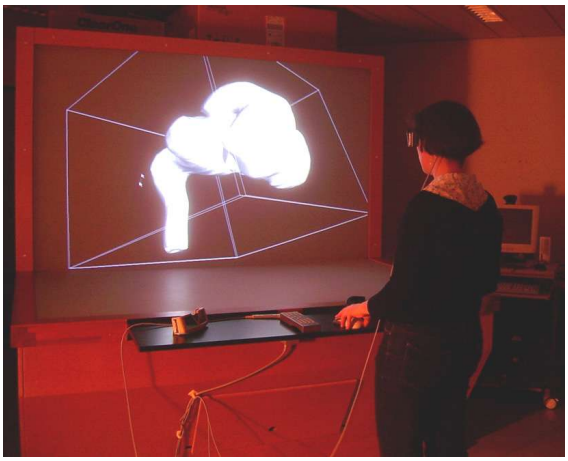
b

c

d



e



f

Abb. 16: Verschiedene Stereoprojektionsverfahren und virtuelle Realität.
a) Polarisationstechnologie, auf dem Tisch ein Notebook mit quad-buffered Grafikkarte, Polarisationsfilterbrille, Bildsplitter, Beamer-Rack (siehe **b**) und spezieller Silberleinwand im Hintergrund.
b) Das vom Autor konstruierte mobile Beamer-Rack mit zwei DLP-Projektoren und linearen Polarisationsfiltern.
c) Head-Mounted-Display (HMD) mit zwei unabhängigen LC-Microdisplays, die ein großes Blickfeld ermöglichen.
d) - f) Stationäre Holobench, die Probandin trägt eine Shutterbrille und hält ein Eingabegerät, links im 19"-Rack der AMD Opteron 64 Orad Cluster mit 5x2 Prozessoren und ganz links die Terminals.

Die mobile Projektionsanlage des Leibniz-Rechenzentrums war nur mit einer starren Beameraufhängung ausgestattet. Für die Stereoprojektion müssen die Beamer jedoch genau aufeinander ausgerichtet werden (auch abhängig vom Leinwandabstand). Daher wurde vom Autor ein neues Beamer-Rack entwickelt und konstruiert (Abbildung 17). Dazu wurden hochwertige feinmechanische Komponenten hergestellt. Die beweglichen Einheiten sind so durchdacht gelagert, dass keine Spannungen im Rahmen auftreten und die Konstruktion stabil bleibt. Das fertige Rack kann die Beamer kinderleicht ohne Werkzeug über drei Drehknöpfe um die drei Raumachsen flexibel und präzise (auf 1000stel Grad genau) ausrichten und so die projizierten Abbildungen (bei Abständen von 1 m bis ∞ m) zur Deckung bringen. Die fertige mobile Einheit kann beim Leibniz-Rechenzentrum ausgeliehen werden und zum Beispiel für eine Präsentation genutzt werden.

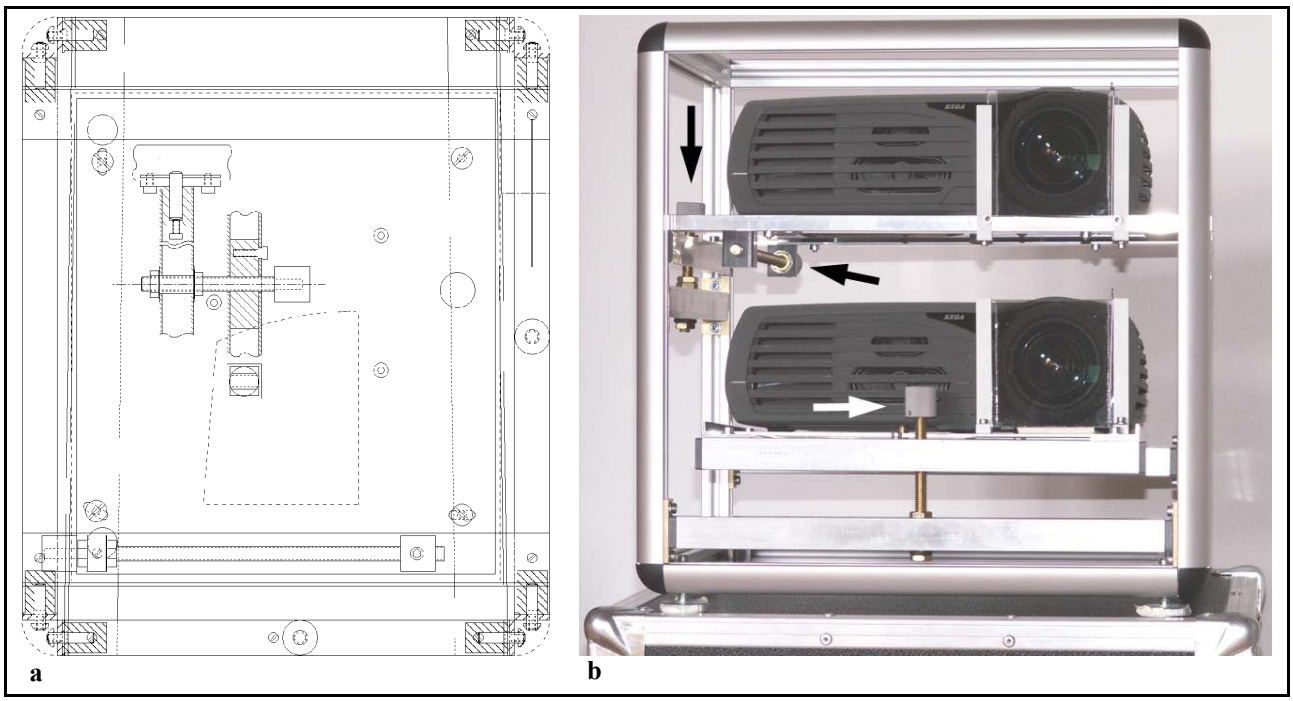


Abb. 17: Konstruktion des Beamer-Racks.
a) Als Grundlage für den Bau wurden CAD-Pläne angefertigt.
b) Das stabile fertige Rack verfügt über drei Drehknöpfe (**Pfeile**), die über Gewindestangen jeweils eine (Teil-) Drehung um eine Raumachse justieren. Die Schienen vor den Linsen halten die Polarisationsfilter. Die Projektoren sind leicht zugänglich, um unkompliziert Konfiguration oder Lampenwechsel zu gestatten.

4 Diskussion

4.1 Material-Qualität

Die histologischen Schnitte des Embryos ME33 stellten eine deutliche Herausforderung für die Visualisierung dar. Das Präparat war für diesen Zweck nicht vorgesehen, wurde jedoch mit den damaligen technischen Mitteln sorgfältig hergestellt. Ein weiterer bildgebender Datensatz, etwa MRI, ist selbstverständlich genauso wenig vorhanden wie extrinsische Marker. Das Harz zum Aufkitten des Deckglases hatte bei einigen Schnitten einen kompletten oder schlimmer einen randbetonten[31] Gelbstich verursacht. Zudem variierte die Intensität der HE-Färbung. Die Einbettung in Paraffin begünstigte Deformationen, aber eine Einbettung und Schneidung in harte Kunststoffe wie Araldit ist bei der Größe des Präparates auch heute noch technisch sehr schwierig. Bei Verwendung weicherer Methacrylate treten ähnliche Nachteile auf wie beim Paraffin. Große Hohlräume wie Ventrikel begünstigten Verzerrungen, Umklappen oder Einknicken von Rändern und Zerreißen von Strukturen. Besonders feine Strukturen, wie das Dach des vierten Ventrikels, waren besonders heftig betroffen, so dass hier ohne Vergröberung der Strichstärke keine geschlossene Oberfläche mehr rekonstruiert werden konnte.

4.2 Lichtmikroskopie

Lichtmikroskopische Präparate können abfotografiert werden und als Quellmaterial dienen. Die Mikrotomie bedingt, dass die einzelnen Schnitte (engl. slices) nicht mehr zueinander ausgerichtet sind. Schlimmer noch, im Rahmen der histotechnischen Präparateherstellung, also der Präparatgewinnung, Fixierung, Entwässerung, Paraffineinbettung, Mikrotomie und dem Streckbad, wird das Präparat verändert. Besonders der Prozess ab der Mikrotomie, die nötig ist, wenn konfokale Mikroskopie wegen eines zu dicken Präparates nicht möglich ist, verzerrt das Material kaum reproduzierbar. Das Mikrotom staucht die einzelnen Schnitte unterschiedlich. Das Streckbad, aus dem sie heraus gefischt werden, soll sie wieder entspannen. Da dies nicht völlig gleichmäßig gelingen kann, werden durch zu große (überstreckte) und zu kleine (gestauchte) Schnitte später im Modell unnatürliche Kanten und Kerben entstehen (Abbildung 4 auf Seite 15). Die Veränderungen sind sogar abhängig von der Gewebeart [Kretschmann et al. 1982] und noch dazu nichtlinear. Sie rückgängig zu machen ist ohne weiteres nicht möglich (Kapitel 4.3 und 5.1.2.1).

31 verfälscht das Graustufenhistogramm besonders

4.3 Registrierung

4.3.1 Metrik

„Amira“ bietet neben Landmarken einen grauwertbasierten Algorithmus und manuelle Interaktion zur Registrierung an. Die Transformation ist jeweils rigide.

Landmarken werden üblicherweise als Vorlauf einer späteren automatischen komplexeren Ausrichtung benutzt, um zu verhindern, dass diese in ein lokales Extremum fällt. Mindestens zwei Markierungen pro Slice, zur höheren Genauigkeit besser mehr, sind dann für jede Schicht anzugeben. Der von „Amira“ verwendete Algorithmus minimiert vermutlich (undokumentiert) das Abstandsquadrat der Landmarkvektoren, also sei die gesuchte Transformation

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \sum_{i=1}^{n} \|\mathbf{f}_i - \mathbf{T} \circ \mathbf{m}_i\|^2 . \tag{1}$$

Die n Markierungen pro Slice (Punkte $\mathbf{m}_i$ (moving) und $\mathbf{f}_i$ (fixed)) werden manuell und sinnvollerweise an klaren anatomischen Strukturen, wie Ränder, Ecken oder Punkten, positioniert. „Amira“ findet das Minimum wohl iterativ, obgleich es mathematisch möglich ist, dieses non-iterativ und daher schneller zu finden [Schormann et al. 1993]. „Amira“ richtet die Schichten einzeln aus, kann aber auch den gesamten Datensatz auf einmal durchgehen. Da es aber keine anatomisch definierbaren Strukturen gibt, die in ausnahmslos allen Schnitten auftreten, schlägt letztere Möglichkeit prinzipiell fehl. Erstere ist wiederum zeitaufwendiger als manuelles Ausrichten. Da die Landmarken manuell positioniert werden müssen, ist das Ergebnis nicht reproduzierbar. Es ist sogar ungenauer als manuelles Ausrichten, da weniger Informationen (Grauwerte) zur Verwendung kommen. In den Paraffinblock artifiziell eingebrachte Marker [Streicher et al. 2000] könnten als topologische Zusatzinformation Landmarken definieren. Die Hauptdomäne solch einer extrinsischen Registrierung ist jedoch eher in der Neurochirurgie und Orthopädie [Simon et al. 1995] zu finden (als globale Angleichung der Koordinatensysteme Patient - Datensatz).

Eine weitere Möglichkeit die Registrierung in „Amira“ durchzuführen, ist eine Ausrichtung via minimalem Abweichungsquadrat der Grauwerte zweier Bilder F und M. $F(\mathbf{p})$ symbolisiert den Grauwert wie in Kapitel 5.1.1 ausgeführt. Die gesuchte Transformation ist

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \sum_{\mathbf{p}} |F(\mathbf{p}) - (\mathbf{T} \circ M)(\mathbf{p})|^2 . \tag{2}$$

Die Summe der Abweichungsquadrate (engl. sum of square differences, SSD) wird in „Amira“ in einen prozentualen Wert umgerechnet und angezeigt, wobei 100% einem idealen Matching entspricht. Dabei ist zu bedenken, dass die SSD bedauerlicherweise sehr sensibel auf akzidentielle Intensitätsabweichungen der beiden Bilder reagiert. Auch ein globaler linearer Kontrast- und

Intensitätsausgleich[32] kann dies nicht verhindern, da die Histogramme sich durch variable Anschnittflächen, etwa der Ventrikel, innerhalb der Schnittserie ändern. Unregelmäßige Störungen, wie die Gelbfärbung durch das Harz, die ungleichmäßige Durchtränkung mit Färbemittel und leichte Toleranzabweichungen der Schichtdicken, führen diesen Algorithmus sowieso in eine Sackgasse. Zu allem Überfluss macht die Quadrierung des Betrags die SSD besonders anfällig für Ausreißer. Eine „mildere" Funktion, etwa eine lineare Abbildung, ist nur geringfügig resistenter (wie im Block-Matching-Algorithmus [Ourselin et al. 2001b; Ourselin et al. 2000]). Besser eignet sich die „Simplified-Local-Correlation-Coefficients"-Metrik (SLCC), die nicht mehr sensibel auf Intensitätsabweichungen reagiert und das Problem löst [Cachier, Pennec 2000].

Zusammengefasst fällt automatisches Ausrichten per SSD zu leicht in lokale Minima. Das manuelle Ausrichten bringt sichtbar bessere Ergebnisse, die sich auch in steigenden Prozentwerten nach Benutzerinteraktion ausdrücken. Die Transformation ist nach manuellem Ausrichten naturgemäß schwer reproduzierbar und aufwendig. Jedoch kann das (lokale) ZNS so vor anderen (globalen) Strukturen besonders beachtet werden. Die drei Freiheitsgrade[33] sind durch Interaktion des Untersuchers mit dem Monitorbild der beiden übereinander gelegten Schnitte zu bewältigen. Da manuelles Ausrichten die besten Ergebnisse liefert, wurde diese Vorgehensweise für diese Arbeit schließlich die Methode der Wahl.

Ein kleiner Ausblick: Das Ausrichten zweier Modalitäten, wie CT auf MRI, wird vielschichtiger. Solche multi-modalen Daten können nur über Umwege mit einer einfachen grauwertbasierten Metrik ausgerichtet werden. Das Extrahieren korrespondierender morphologischer Strukturen (etwa über Dilatation und Erosion) ermöglicht das Ausrichten von CT, PET oder SPECT auf MRI [Maintz et al. 1997]. Auch auf histologische Schnitte kann ausgerichtet werden [Mega et al. 1997]. Diese Methoden enden meist in binärer Segmentation. Verhindern kann dies die Fuzzy[34]-Logik, sie hielt nun auch in die Registrierung Einzug [Maintz et al. 1996].

Als State-of-the-art drängt sich jedoch eine überlegene robuste Technik namens „Mutual-Information-Registrierung" in den Vordergrund. Diese Metrik ist wie geschaffen für Bilder, die zwar das Gleiche zeigen, jedoch unterschiedlich darstellen (eben multi-modal). Der Aufwand eines solchen Algorithmus verschlingt naturgemäß einiges an Rechenkapazität (Formel (3) auf Seite 62). Es ist auch gut für mono-modale Daten als Metrik geeignet. Leider ist dieses Verfahren in „Amira" nicht implementiert.

32 Eine normierte und zentrierte Verteilung erleichtert zumindest die Metrik.
33 rigide 2-D-Transformation, also Translationen entlang der x- und y-Achse und Rotation in der x-y-Ebene
34 1965 von Lotfi Zadeh eingeführt, macht sie sich erst seit der jüngeren Vergangenheit verdient (z. B. in der Steuerungstechnik). Sie benutzt unscharfe graduelle Abstufungen statt binärer Aussagen.

4.3.2 Transformation

Da sich die anatomischen Strukturen von Schnitt zu Schnitt ändern, ist es schwer möglich zu beurteilen, ob eine Änderung zwischen zwei Schnitten auf einer artifiziellen Deformation beruht oder durch strukturelle Änderung verursacht ist. Schon ein einfacher Körper, wie ein schräger Zylinder, kann selbst bei idealen, also ungestreckten und ungestauchten Schnitten, die dann nur eine rigide Transformation benötigen, nicht mehr rekonstruiert werden. Die Information, ob und wie schräg der Zylinder denn ursprünglich war, ist aus den Schnitten nicht mehr ersichtlich. Fehler addieren sich von Schicht zu Schicht, am Ende ist aus dem Zylinder ein welliger Schlauch geworden. Nun sind mikroskopische Präparate noch komplexer als ein Zylinder. So wird anatomisch korrektes Ausrichten zur Herausforderung. Gegenüber vorigem Gedankenexperiment kommen hier noch die nichtlinearen Deformationen hinzu.

Die Verzerrungen von ME33, die durch die Größe des Präparates, der Paraffineinbettung und den großen Ventrikeln ein derartiges Ausmaß haben, dass sie mit rigiden Transformationen nicht in den Griff zu bekommen sind, führen in der Rekonstruktion zu unregelmäßigen vertikalen Furchen. Aufgrund der nichtlinearen Natur der Deformation könnte auch eine affine Transformation nicht die korrekte Lösung liefern. Rigide Transformationen, wie „Amira“ sie anbietet, verschlimmern die Situation. Doch die selbstentwickelte Technik, vorgestellt in Kapitel 3.1.3, konnte die Zerfurchung des Modells minimieren, wie Abbildung 4 auf Seite 15 veranschaulicht. Auch andere Datensätze, die von unterschiedlich deformierten Quellmaterialien, etwa histotechnisch hergestellten Schnittserien, stammen, können zukünftig von dieser Technik profitieren und selbst bei rigider Transformation noch ansehliche Rekonstruktionen mit glatter Oberfläche liefern.

Ausrichtungsmethoden die auf elastischen Transformationen beruhen, etwa finite Elemente, können Deformationen höherer Freiheitsgrade auf elegante, gleichwohl technisch fordernde Weise in den Griff bekommen. Doch solche komplexen Transformationen überkompensieren akzidentiell Deformationen, die anatomisch bedingt sind. Ein vollautomatischer Algorithmus, der die anatomischen Strukturen nicht kennt, die gerade rekonstruiert werden sollen, kann nicht zwischen Artefakt und anatomischem Sachverhalt unterscheiden. Vorgegebene feste 3-D-Geometrien intrinsischer Natur in Form von Oberflächen und morphologischen Kriterien sind eine mögliche Lösung [Maintz et al. 2001; Maintz et al. 1996; Maintz et al. 1994]. Auch die schon angedeuteten extrinsische Marker [Ellis et al. 1996], in der Histologie etwa in Form von geraden Zylindern (durch Laser) oder Fäden, können einer Registrierungsmethode einen klaren Hinweis geben. Transformationen werden über solche Landmarken leichter bestimmbar und verifizierbar.

Ein weiterer Lösungsansatz ist, einen Datensatz über ein Bildgebungsverfahren zu gewinnen, das eine Ausrichtung überflüssig macht, wie etwa die Tomographie. Dieses liefert die Grundlage[35] für die weitere Aufbereitung. Insbesondere CTs, die mit noch kurzwelligerer Strahlung und höherer Dosis als im Krankenhausbereich[36] üblich arbeiten, kommen in Frage. Weitere Verfahren werden ab Kapitel 5.1.1.1 erläutert. Ein Synchroton-CT besitzt eine Ortsauflösung um die 1.5 µm bis 11 µm bei einer maximalen Objektgröße von 20 mm. Ein 100 kV-Tomograph liefert für Objektgrößen bis zu 30 mm eine Ortsauflösung von 2 µm, ein µ-CT 80 µm. Auch µ-MRI findet Verwendung [Smith et al. 1996]. Auf der Basis eines solchen Datensatzes könnte dann ein weiterer (durch Mikroskopie gewonnener) hochauflösender Datensatz multi-modal ausgerichtet werden. Der Entwicklungsaufwand und die Komplexität eines solchen Algorithmus erreicht dann auch ein extrem hohes Maß und wurde bisher nicht überwunden. Einen ersten Schritt in diese Richtung machen Schormann et al. 1995, die einen fertig ausgerichteten histologischen Datensatz über eine globale affine Transformation und automatisch generierte Landmarken ausrichten. Hier wird diejenige Transformation gesucht, welche die globalen Verzerrungen möglichst gut kompensiert, ohne von lokalen Deformationen gestört zu werden. Hierbei wird nicht-iterativ (d. h. schnell) das Abweichungsquadrat möglichst vieler gleichmäßig verteilter Landmarken minimiert. Lokale Deformationen bleiben dennoch in der verbesserten, aber iterativen Version weiterhin erhalten [Schormann et al. 1995]. Auch Ourselin et al. 2001a verwenden das MRI nicht, um die mikroskopischen Schnitte untereinander auszurichten, sondern richten den histologisch gewonnenen Stapel erst rigide aus und dann diesen als Ganzes global auf den MRI-Datensatz. Lokale Deformationen werden auch hier nicht behoben. Wie sich in vorliegender Arbeit jedoch gezeigt hat, sind diese das Hauptproblem. Jene über elastische Transformationen zu kompensieren, bleibt auch weiterhin Herausforderung.

4.4 Segmentierung

Die HE-Färbung hebt das ZNS gegenüber anderen Strukturen weder in Farbton, Sättigung noch Intensität hervor[37]. So wird bei entsprechender Histogrammanpassung die graue Substanz betont, mit ihr aber auch die Leber. Die Arachnoidea bleibt mitsamt Stützgewebe flau. Es ist also nicht möglich, durch Histogrammoptimierung und nachfolgendes Thresholding die geforderte Segmentierung durchzuführen. In nur wenigen Anwendungen (Knochen im CT) kann das Objekt über einen spezifischen Intensitätsbereich gekennzeichnet werden und durch einen einfachen Schwellwertfiltern hervorgehoben werden. Auch automatische intelligente Filter, die die Struktur

35 da Auflösung und Informationsgehalt nicht an histologische Schnitte heranreicht
36 Dort liefert ein aktuelles 2x64-Schicht-Spiral-CT (Siemens Somatom) eine isotrope Ortsauflösung um die 0.4 mm.
37 was durch immunhistochemische Methoden bewerkstelligt werden könnte

analysieren, würden scheitern, da die Struktur innerhalb des zu markierenden Bereichs wechselt und sich nicht ausreichend von den nicht zu segmentierenden Arealen unterscheidet.

Es gibt keinen Königsweg, zu unterschiedlich sind die Bedürfnisse und kein Algorithmus kann automatisch, genau und flexibel zusammen sein. Die Anzahl der Lösungen ist beachtenswert, die der Probleme jedoch gewaltig. Arbeiten, die vollautomatische Techniken beschreiben, sind auf wenige Spezialfälle zugeschnitten. Methoden, die mit beliebigen Datensätzen zurande kommen, erfordern einiges an menschlicher Interaktion. Meist ist für jeden Fall ein individueller Algorithmus zu entwickeln. Keimzellenwachstums-, Konturverfolgungs-, regionen-basierte Verfahren, Kanten-operatoren [Jendrysiak 1999] oder Wasserscheidentransformation [Wegner et al. 1999, Vincent, Soille 1991] sind nur einige Beispiele hierfür. Um den Interaktivitätsaufwand zu minimieren, sind komplexere Verfahren wie morphologische Filter und Analysen vonnöten. Vorherige Filterung der Daten verbessert meist das Ergebnis. Das Einbeziehen benachbarter Schichten ist der alleinigen Betrachtung einer Schicht überlegen, erhöht aber nochmals die Komplexität erheblich [Fathi et al. 1999]. In vielen Arbeitsgruppen kommen semi-interaktive und hoch spezialisierte Algorithmen zur Anwendung. Oft findet eine automatische Vorsegmentierung statt, gefolgt von einer manuellen Nachbearbeitung oder noch mehr Teilschritten. So verwenden zum Beispiel Schiemann et al. 1997 eine Kaskade von Algorithmen, namentlich Thresholding auf der Basis von RGB-Ellipsoiden gefolgt von binärer mathematischer Morphologie und Komponenten-Labeling.

Kurz, eine Universallösung bleibt reine Utopie. Kann keine geeignete Methode gefunden oder entwickelt werden, ist die vollständig manuelle Bearbeitung unumgänglich, wie sie in der Biologie oder Medizin nicht selten verwendet wird. Jedes Ergebnis eines automatischen Segmentierungssystems muss sich schließlich mit demjenigen eines klinisch Sachkundigen messen.

Es ist aber noch nicht damit getan, die Strukturen zu segmentieren. Auf den entstehenden binären Volumendatensatz wird der Rekonstruktionsalgorithmus zurückgreifen. Das Ergebnis der Rekonstruktion wird von der Segmentierung insofern beeinflusst, dass Schwachpunkte im Algorithmus der Rekonstruktion durch geschickte Segmentierung umgangen werden müssen und vice versa oder auf einen anderen (gleichwohl komplexeren) Algorithmus ausgewichen werden muss. Dies zeigte sich beim Marching-Cubes-Algorithmus in Unterbrechungen feiner Strukturen. Nerven können so nicht befriedigend rekonstruiert werden. Um Nerven oder Gefäße ohne Vergröberung oder Unterbrechung rekonstruieren zu können, ist Marching-Cubes fehl am Platze. Er sieht nur die Voxel, aber keinen Bezug dieser zueinander. Abhilfe schaffen könnte die (Subpixel-) Segmentation solcher Strukturen mit Hilfe von Liniensegmenten, die Zusammengehörigkeit beschreiben. Diese werden dann einem spezialisierten Rekonstruktionsalgorithmus übergeben.

4.5 Mapping

„Amira“ beherrscht nur Marching-Cubes. In der Arbeit kam daher der Marching-Cubes-Algorithmus [Lorensen, Cline 1987] zur Anwendung (Kapitel 5.1.2.3). Diese Universallösung spiegelt zwar noch immer den Stand der Technik wieder, schleppt aber einige Probleme ein.

Ungewollte Löcher und Unterbrechungen der zu rekonstruierenden Struktur ergeben sich prinzipbedingt, da etliche Konstellationen uneindeutig sind. Die 15 möglichen Würfelkonfigurationen lassen 33 Triangulationen zu [Chernyaev 1995]. Nur durch aufwendige Erweiterungen des Algorithmus, die eine verbesserte Lookup-Tabelle verwenden, lassen sich diese Zweideutigkeiten auflösen [Lewiner et al. 2003].

Die Schwächen des Verfahrens offenbaren sich auch in der riesigen Datenmenge. Diese lässt sich nur durch Reduktion der Dreiecke in den Griff bekommen, wofür auch einige Methoden zur Verfügung stehen und befriedigende Ergebnisse liefern. Die Funktion in „Amira“ reduziert die Dreiecke auf Kosten der Qualität. Im Ergebnis werden Glättungseffekte sichtbar, die in dieser Arbeit nicht unerwünscht waren. Ist dies störend, kommen ausgefeiltere Verfahren zum Einsatz. Wilmer et al. 1992 erreichen eine Reduktion der Dreiecke um 60% ohne nennenswerten Qualitätsverlust.

Während Marching-Cubes eine explizite Repräsentation der Oberfläche liefert (Dreiecknetze) gibt es auch Methoden, die Oberflächen parametrisch (wie NURBS[38]) oder implizit beschreiben (also durch skalare Funktionen). Implizite Repräsentationen haben den Vorteil, dass sie besser mit Punktwolken, verrauschten Daten und Dynamiken wie den Deformationen umgehen können [Zhao et al. 2001; Zhao, Osher 2002], aber nicht intuitiv sind. Der notwendige mathematische Aufwand ist nicht unerheblich (besonders bei Differentialgleichungssystemen) und daher im nicht rein technischen Bereich kaum zu finden.

Das Gehirn ist in ausreichender Genauigkeit spiegelsymmetrisch. Das ist eine Zusatzinformation, mit der die Daten ohne größeren Aufwand geglättet werden können. Das Gehirn wird gespiegelt, mit dem Original (nach vorheriger Ausrichtung) kombiniert und über Averaging ein neues Modell berechnet, dessen Störungen sich mehr beruhigt haben. Die Schnittebenen von ME33 sind nahezu parallel zur Symmetrieebene. Eine senkrechte Schnittführung könnte Störungen sogar noch besser heraus rechnen. Prinzipiell kann diese Methode Deformationen gleichmäßig verteilen, diese dabei absolut verkleinern und im Idealfall sogar aufheben. Man erhält ein ideal symmetrisches Modell.

38 Non-Uniform Rational B-Splines sind besonders vielseitig.

4.6 Rendering

Das Modell liegt nun trianguliert vor. Diese explizite Repräsentation ist lang erprobt und wird von gängiger Hardware beschleunigt. Eine Render-Pipeline kann mit Dreiecken unkompliziert und zügig umgehen. Leider liegen die Oberflächennormalen im STL-Format nur per-triangle vor und nicht per-vertex. Gleichwohl ist Gouraud-Shading (einfache Interpolation) sinnvoll und wird durchgeführt, damit die Vertices bei Punktlichtquellen verschiedene Farben (Glanzlichter) erhalten. Eine weitere Glättung durch Phong-Shading ist nicht machbar, die Grenzen der Facetten bleiben sichtbar. OpenGL unterstützt von sich aus gar kein Phong-Shading. Dies könnte jedoch mit aktuellen Grafikkarten, deren Shader-Einheiten (per-pixel) programmierbar (GL Shading Language) sind, realisiert werden [Loviscach 2005; Shirley et al. 2005].

Die entwickelte Software rendert triangulierte Modelle leistungsstark. Tests im Leibniz-Rechenzentrum ergaben eine sogar sichtbar höhere Performance gegenüber „Amira“. Die Eigenkreation ist nicht überladen, und dabei leichter zu bedienen (insbesondere die angenehme Möglichkeit eine Schnittebene zu definieren und Stereoskopie zu nutzen). Es ergeben sich so ideale Bedingungen für ihren Einsatzzweck.

Volume-Rendering ergibt auf den ersten Blick beeindruckende Bilder. Sucht man nach Details, fällt jedoch auf wie verschwommen, pixelig oder verrauscht die Information zumeist präsentiert wird. Hinzu kommt, dass Volume-Rendering sehr langsam ist und nur von wenigen speziellen und teuren Karten beschleunigt wird. Auch kann die Größe des Datensatzes nicht ohne schwere Einbußen reduziert werden. Wie eingangs dargelegt, ist die Größe des Datensatzes besonders für 32-Bit-Systeme kritisch.

4.7 3-D-Projekte für Lehre und Forschung

4.7.1 Visible Human Project

Das „*Visible Human Project*“ [VHP] wurde 1986 von der National Library of Medicine [NLM] in Bethesda, Maryland, USA ins Leben gerufen. Ziel war es ein Archiv digitalisierter anatomischer Bilder zu erstellen [Ackerman 1995]. 1989 wurde beschlossen, komplette Datensätze aus CT, MRI und Kryostat-Schnitten menschlicher Leichen zu gewinnen, und zwar eines männlichen und eines weiblichen hingerichteten Delinquenten.

Die MRI-Ebenen des männlichen Leichnams haben einen Abstand von 4.0 mm und eine Auflösung von 256 x 256 Pixel bei 12-Bit Farbtiefe. Die CT-Daten sind mit einem axialen Abstand von 1.0 mm bei 512 x 512 Pixel sowie 12-Bit Farbtiefe etwas feiner, die Kryostat-Schnitte mit einer

Dicke von 1.0 mm wurden zunächst mit einer Auflösung von 2048 x 1216 Pixel bei 24-Bit Farbtiefe aufgenommen. Im November 1994 wurde der Datensatz der männlichen Leiche mit einer Größe von 15 GB fertig gestellt. Die Kryostat-Schnitte wurden im Jahr 2000 nochmals mit 4096 x 2700 Pixel gescannt.

Um die Auflösungen der Vertikalachse an die der transversalen und sagittalen Achse anzugleichen, wurde für den Datensatz der weiblichen Leiche erstere Achse feiner aufgelöst. Das soll laut NLM kubische Voxel ermöglichen. Die isotropen Daten der weiblichen Leiche haben also statt 1.0 mm einen Schichtabstand von 0.33 mm und kamen ein Jahr später auf 40 GB. Die Datensätze werden auf Magnetband kommerziell vertrieben.

Das Projekt wächst stetig weiter. Es soll ein Wissensnetzwerk entstehen, das die visuellen Daten mit textbasierten Arbeiten hierarchisch verknüpft und so eine umfassende Datenbank ermöglicht. Die University of Michigan arbeitet an dem „Next Generation Internet“, über das auf verknüpfte Informationen, 2-D- und 3-D-Visualisierungen zugegriffen werden soll. Des Weiteren sind interessante und vielversprechende quelltextoffene Software-Bibliotheken (ITK, VTK) entstanden und werden im Folgenden aufgrund ihrer Verbreitung im medizinischen Visualisierungsbereich kurz vorstellt.

4.7.1.1 Insight Toolkit

Im *Insight Toolkit* [ITK] sind Bildverarbeitung und Filterung implementiert, wobei besonders die Ausrichtung und Segmentation von Interesse sind. Der Abstraktionslevel hierbei ist naturgemäß hoch, da ausschließlich mathematische Algorithmen, aber keine plattformabhängigen Konzepte benötigt werden. Ibanez et al. 2003 bietet eine vollständige Referenz.

C++-typische Paradigmen wie Ausnahmen und Namespaces werden konsequent eingehalten. Moderne Designs durch objekt-orientierte Programmierung (OOP), wie Iteratoren, Streaming, Ereignisse oder Smartpointer, sind verwirklicht. Die Typen sind generisch gehalten, damit ist Typsicherheit und zugleich Flexibilität gewährleistet. Instanzen von Klassen, die Bilddaten enthalten, werden über die Methoden `SetInput()` und `GetOutput()` mit Filtern zu beliebigen Daten-Pipelines verknüpft.

Die Nachteile liegen auf der Hand: Es handelt sich um eine große, komplexe C++-Bibliothek, die nicht auf schnelle Ausführungsgeschwindigkeit getrimmt wurde. Dem Rechner wie auch dem Entwickler wird einiges abverlangt. Andererseits ist die Thematik eben komplex, deswegen muss einige Einarbeitungszeit, in welche Bibliothek auch immer, investiert werden.

4.7.1.2 Visualisation Toolkit

Hingegen stellt das *Visualisation Toolkit* [VTK] I/O, Mapping, Rendering und eine Schnittstelle zu interpretierten Sprachen zur Verfügung [Schroeder et al. 1996]. Um auch hier möglichst plattformunabhängig zu bleiben, hat VTK sinnvollerweise eine höhere Abstraktionsschicht als OpenGL und ist unabhängig von OpenGLs State-Machine und Pipeline. Durch die Entwicklung einer Grafikschnittstelle die stärker abstrahiert als OpenGL, kann sie bei Bedarf von VTK auf eine andere als die vorhandenen (OpenGL, XGL) portiert werden. So können die Paradigmen der OOP besser eingehalten werden. Vom Design her baut VTK auf einem beliebig verknüpfbarem Netzwerk auf, durch das die Daten fließen. Am Anfang steht daher immer eine Datenquelle, am Ende erzeugt ein Mapper darstellbare Geometrien, dazwischen bearbeiten und manipulieren Filter die Daten. VTK beherrscht viele komplexere Techniken, wie Tensoren, implizite Funktionen, Polygonreduzierung, Delaunay-Triangulierung, Marching-Cubes, Surface-Rendering oder Volume-Rendering, um nur einige zu nennen. Eine grafische Benutzeroberfläche (GUI) ist nicht enthalten, dadurch sind keine Abhängigkeiten vorhanden. So kann eine externe Lösung, wie die Microsoft Foundation Classes [MFC], Trolltechs QT [QT] oder das quelltextoffene und freie Fast Light Tool Kit [FLTK], über Callback-Mechanismen eingebunden werden. Im Umfang der Bibliotheken finden sich zudem Wrapper für die Skript-Sprachen TCL, Phyton und Java.

Die Bibliotheken sind in C++ geschrieben, unterstützen Fabriken, Multi-threading, Streaming und Ereignisse. Das objekt-orientierte Design kapselt Daten gegen direkte Zugriffe durch Algorithmen ab. Leider sind moderne Designs mit Namespaces, generischer Programmierung und Ausnahmen nicht verwirklicht worden. Besonders die spärliche Verwendung von Templates, die sowohl generische Datentypen ermöglichen, als auch abstrakte und leistungsfähige Entwurfsmuster [Vandevoorde, Josuttis 2003], ist enttäuschend. Der mögliche Fortschritt zeigt sich im Vergleich der Designpatterns der „Gang-of-four“ [Gamma et al. 1995] zu Alexandrescu 2001.

4.7.2 Voxel-Man

Am Universitätsklinikum Hamburg-Eppendorf beschäftigt sich das Institut für Mathematik und Datenverarbeitung in der Medizin [IMDM] seit 1970 schwerpunktmäßig mit der Bildverarbeitung in der Medizin, insbesondere der 3-D-Modellierung [Höhne 2002]. Unter dem Namen *Voxel-Man* wurde ein anatomischer Atlas entwickelt [Pommert et al. 2001]. Die originalen VHP-Daten wurden dazu zu kleineren Einheiten reduziert, die 1 mm^3 Voxel enthalten, und dann mehrfach segmentiert [Schiemann et al. 1997]. Die erstellten räumlichen Modelle können dem Benutzer in hoher Qualität präsentiert werden [Höhne 1999; Pommert et al. 2003]. Nachdem zuerst ein Gehirnatlas erschien,

wurden Modelle der inneren Organe fertig gestellt und kommerziell vertrieben [Höhne et al. 2001; Höhne et al. 2003]. Im IMDM wird auch an neuen Segmentierungs-, Visualisierungs- und Informationstechniken gearbeitet [Höhne et al. 1996].

4.7.3 AIR

AIR ist eine quelltextoffene plattformunabhängige Software, die an der University of California, Los Angeles (UCLA) entwickelt wurde [AIR 2002]. Die in C implementierte Bibliothek ist Grundlage von zwei Modulen, die jeweils Ausrichtung von Schnitten mit linearen oder nicht-linearen Transformationen unterstützen. Allerdings stellt AIR nur die mögliche Grundlage einer Software-Eigenentwicklung dar, da keine ausführbaren Dateien angeboten werden und nur Algorithmen zur Ausrichtung von Schnitten implementiert wurden.

4.7.4 3-D-Darstellung embryonaler Strukturen

Blaas et al. 1998 erhielten ihre Daten von einem transvaginalen 3-D-Ultraschall. Von den ab Carnegie-Stadium (CS) 16 untersuchten Embryonen und Feten wurde das Gehirnventrikelvolumen gemessen. Die Bildqualität ist für Sonographie beeindruckend, die Auflösung kann aber nicht an andere Verfahren heranreichen. Details bleiben außen vor, es kann nur eine grobe Orientierung stattfinden. Die Darstellbarkeit der embryonalen und frühen fetalen Morphogenese wurde auch bei Brauer 2004 durch die Sonographie-Auflösung stark limitiert, es konnte jedoch ein grobes Oberflächenmodell demonstriert werden. Im Übrigen muss, um anschauliches Material zu erhalten, vor allem bei der Segmentierung noch einige Interaktion des Untersuchers erfolgen. Der nicht befriedigenden automatischen Segmentierung und der limitierten Auflösung wegen wird das Ergebnis von Ultraschalldaten meist über Volume-Rendering dargestellt [Sakas 1999].

Haque et al. 2001 rekonstruierten Neuralrohre und Somiten vierer humaner Embryonen im CS 12. Als Quellmaterial dienten digitalisierte histologische Schnitte. Die Strukturen wurden in den Bildern manuell nachgezeichnet. Auf die technischen Details der Rekonstruktion wurde nicht weiter eingegangen, das Ziel der Arbeit war nicht die Rekonstruktion an sich, sondern die Anzahl der Somiten zu bestimmen. So lässt sich wohl erklären, dass die Bilder der Arbeit kaum einen plastischen Eindruck vermitteln.

Radlanski et al. 2003 rekonstruierten den Unterkiefer verschiedener humaner Embryonen aus histologischen Schnitten. Mit Lichtmikroskopie konnte eine hohe Auflösung erzielt und so die Differenzierung einzelner Zellen (Knochen, Knorpel, Nerven und Gefäße) ermöglicht werden. Die

Registrierung fand mit Hilfe von Landmarken statt, segmentiert wurde manuell. Die Rekonstruktion wurde mit Analysis (SIS, Münster) durchgeführt.

Das „*Edinburgh Mouse Atlas Project*“ [Emap 2003] hat sich vorgenommen, die Entwicklung von Mäuseembryonen und deren Genexpressionen zu visualisieren. Histologische Schnitte (Theiler-Stadien 1 bis 20) wurden in zwei Schritten registriert (manuelle und elastische Registrierung). Für die elastische Registrierung wurden Landmarken als Finite-Elemente benutzt, was einen hohen manuellen Aufwand und hohe Rechenzeit bedeutet. Die Segmentierung fand manuell statt. Das gewonnene Datenvolumen dient nun als Ausgangspunkt für eine Verknüpfung von Genexpressionen und anatomischen Strukturen. Die Videos auf der Homepage zeigen sich recht grobpixelig, da wohl weniger die Anatomie im Vordergrund steht.

Die Probleme, die die histologischen Schnitte mit sich bringen, werden mit der *Optical-Projection-Tomography* (OPT) (Kapitel 5.1.1.7) umgangen. Hier wird der Vorteil einer fertigen Ausrichtung, den eine Tomographie mit sich bringt, mit Mikroskopie zur Vergrößerung verbunden. Sharpe et al. 2002 demonstrierten diese Technik an Mäuseembryonen, Kerwin et al. 2004 auch an humanen Embryonen (CS 12 bis 23). Leider kann die Auflösung nicht mit der klassischer Mikroskope mithalten. Auch müssen die darzustellenden Objekte möglichst transparent sein und eine geringe Refraktion (für Licht einer Quecksilberdampflampe) besitzen, sonst kann die OPT nicht durchgeführt werden. Bei der Durchleuchtung wird hier nicht entlang einer Linie integriert (Ideal), sondern entlang eines Kegels, weswegen sich systematische Fehler einschleichen.

Der „*Electronic Atlas of the Developing Human Brain*“ [EADHB] beschäftigte sich mit der Visualisierung der Entwicklung des menschlichen Gehirns samt Genexpression (CS 12 bis 23). Auch in diesem Projekt wurde OPT verwendet. Obwohl hier insbesondere die Entwicklung des menschlichen Gehirns dargestellt werden sollte, ist dieses in den Videos kaum zu erkennen. Es kann von anderen Geweben nicht unterschieden werden. In dem Video auf der Homepage, das zumindest ein (hand-)segmentiertes Gehirn darstellt, ist das dünne Dach des Rautenhirns nicht beachtet worden und der vierte Ventrikel kommuniziert nach außen.

Im „*Multidimensional-Human-Embryo-Projekt*“ wurden humane Embryonen der Carnegie-Stadien 10 bis 23 mit einem 9.4 Tesla µ-MR (T1, T2 und Diffusion) erfasst [Smith et al. 1996]. Die 128 x 256 x 256 großen 16-Bit Datenblöcke (39 - 156 µm große Voxel) wurden in Adobe Photoshop mit einfachen Masken segmentiert und über Volume-Rendering auf Silikon-Graphics-Workstations dargestellt. Mit Hilfe von Morphing wurden die einzelnen Stadien verknüpft und Apple-QuickTime-Animationen erstellt. Die Daten wurden im Internet verfügbar gemacht [Smith 2003]. Auch Embryonen verschiedener Tiere sind hier zu finden.

4.7.5 Das Software-Projekt dieser Arbeit

4.7.5.1 Datenimport

Für die Wahl eines geeigneten Dateiformates standen die 3-D-Exportformate von „Amira“ zur Verfügung. Das offene VRML (Web3D Konsortium) ist ein textbasiertes Format und benötigt seiner fest gefügten Struktur und Komplexität wegen einen vollständigen Parser, der den kompletten Funktionsumfang zu implementieren hat, um standardkonform (nach ISO/IEC 14772) zu sein. Auch vollzieht der genügsamere Nachfolger X3D die Ablösung von VRML [Web3D][39]. X3D wird jedoch von „Amira“ nicht unterstützt. IV (OpenInventor) oder SURF („Amira“) bestehen aus einem textorientierten und einem binären Teil. Auch sie sind komplexer als für die hier relevante Rekonstruktion nötig. Das Data Exchange Format (DXF) von Autodesk ist als Austauschformat von AutoCAD bekannt. Da DXF jedoch nicht in Normblättern von ISO, ANSI oder Ähnlichem standardisiert ist, gibt es unzählige Dialekte, Versionen und undokumentierte, sogar verschlüsselte Objekte [CR/LF][40]. Unter diesen Umständen ist das Format für das Projekt dieser Arbeit nicht zu handhaben.

Um das Modell klein zu halten, musste ein binäres Dateiformat gewählt werden. Schließlich fiel die Wahl auf das verbreitete STL-Format (Standard Triangulation Language) [Feng 1997]. Es wird bei Rapid-Prototyping-Fertigungsverfahren angewandt und füttert eigentlich Maschinen der technischen Reproduktion, wie Stereolithographie, mit Daten.[41] Da es zur Schnittstelle vieler CAD-Systeme gehört, gilt STL de facto als Industriestandard. Seine Unflexibilität macht sich in einigen Nachteilen bemerkbar: Es unterstützt keine Farben oder IDs, außer in einer nicht-standardisierten Erweiterung. Den Normalen sind nur Dreiecke (per-triangle normals), nicht einzelne Vertices (per-vertex normals) zugeteilt (Abb. 12 im Vergleich zu Abb. 7). Das schränkt das Ergebnis der Beleuchtung etwas ein[42], spart aber Speicherplatz und Rechenzeit. In STL sind darüber hinaus die Datenblöcke nicht computerfreundlich auf 32- oder 64-Bit Grenzen ausgerichtet (nicht memory aligned). Trotz dieser Nachteile wurde STL als kompaktes Dateiformat gewählt, da es von „Amira“ exportiert werden kann und es leicht zu „parsen“ ist, also der Implementierungsaufwand relativ gering ist. STL verwendet als Datentyp die 32-Bit Gleitkommazahl einfacher Genauigkeit (single-precision) aus dem IEEE Standard [IEEE 754-1985]. 32-Bit ist laut Standard das kleinste Datenformat, reicht aber für Visualisierungszwecke[43] aus, bietet bei 32-Bit-Systemen optimale

39 X3D löst den VRML97-Standard ab und bringt interessante Neuerungen (XML, Profile, Modularität, Skalierbarkeit).
40 So hat bedauernswerterweise die DXF-Version AC1009 aus AutoCAD 12 fast keine Gemeinsamkeiten zu AC1012 aus AutoCAD 13, ein Parser müsste komplett neu entwickelt werden.
41 Es kann daher auch ein reales Modell hergestellt werden.
42 Man kann die einzelnen Dreiecke sehen, es gibt keine sanften Farbübergänge.
43 Selbst aktuelle (Consumer-)Grafikprozessoren rechnen mit maximal 32-Bit Gleitkommagenauigkeit.

Performance [Goldberg 1991] und steht praktisch überall zur Verfügung[44]. STL's einfache Spezifikationen ergeben zwar eine eingeschränkte Flexibilität, sind aber gleichzeitig von Vorteil. So lässt sein einfacher Aufbau erwarten, dass STL auch nach Jahren unverändert verwendet werden kann, was sich von den anderen erwähnten Formaten nicht behaupten lässt.

4.7.5.2 Komplexität

Die Software soll nicht den Anspruch einer „eierlegenden Wollmilchsau“ erfüllen. Es soll eine schlanke, intuitiv zu bedienende und (bezogen auf die Systemvoraussetzungen) anspruchslose Anwendung entstehen. Grafische Spielereien, ein GUI oder enorme Flexibilität können nicht realisiert werden. Hier stehen schon einige umfassende, kommerzielle Anwendungen von Drittherstellern parat. Die Überfrachtung solcher Software schränkt jedoch Bedienbarkeit, Übersichtlichkeit und Performance über die Maßen ein. So lenkt langwieriges Starten und umständliches Bedienen solcher Software von einem Vortrag ab, das Publikum wird ungeduldig und der Dozent unruhig. Hier soll also ein Viewer entstehen, der mal eben das dreidimensionale Objekt anzeigt und in Echtzeit drehen kann. Nicht mehr und nicht weniger. Dem Benutzer soll kein IT- oder Grafikfachwissen abverlangt werden, er soll keinen Einführungskurs belegen müssen. Insbesondere werden die stereoskopischen Parameter über eine Reihe mathematischer Formeln mit sinnvollen Werten belegt, ohne vom Benutzer Kenntnisse über Parallaxe, HIT oder Separation (Kapitel 3.3.6) zu erwarten. Die medizinischen und biologischen Inhalte, nicht die Technik stehen schließlich im Vordergrund.

Die einfache Bedienung der Software soll den Anspruch einer sauberen Implementierung nicht mindern. Im Gegenteil, die Herausforderung soll genau hierin liegen. Das Augenmerk auf die benötigte Funktionalität und das Wesentliche gerichtet, wird ein modernes Software-Design und durchdachte Programmierung gefördert.

4.7.5.3 Bedienung von `pw_viewer`

Hier wird exemplarisch demonstriert, wie die Software bedient wird. Die ausführbare Datei heißt pw_viewer[45]. Sie erwartet als Argument die zu ladenden Datei. Eine binäre STL-Datei cs18.stl wird somit über den Aufruf `pw_viewer cs18.stl` oder noch einfacher mit der Maus über Drag-n-Drop geladen und angezeigt. Kann die Datei nicht gefunden werden, wird eine Fehlermeldung

44 Die nativen Windows 64-Bibliotheken bieten die 64-bittige double-precision nur aus der SSE-Einheit an und double-extended-precision (80-Bit bei Intel) der FPU erschließen erst spezielle numerische Bibliotheken oder Assembler.

45 Sie kann in einem zip-komprimierten Archiv von ftp://ftp.lrz-muenchen.de/transfer/anatomieIII/ geladen werden.

ausgegeben. Optional können dem Programm noch weitere Kommandozeilenargumente in Lang- oder Kurzform mitgegeben werden (lehnt sich an POSIX 1003.2 an):

`--help`	`-h`	bietet Hilfe zu den Kommandozeilenargumenten
`--fullscreen`	`-f`	startet im Vollbildmodus
`--stereo`	`-s`	startet im Stereoskopie-Modus, spezielle Hardware vorausgesetzt
`--anaglyphic`	`-a`	startet im anaglyphischen Modus
`--version`		zeigt Versionsnummer an
`--debug`		zeigt technische Informationen über die Grafikkarte an
`--quiet`	`-q`	Textausgaben werden auf ein Minimum begrenzt

Ein Aufruf `pw_viewer cs18.stl -f -q --stereo` startet das Programm im Vollbild ohne viele informative Textausgaben bei stereoskopischem Betrieb. Kann keine stereoskopiefähige, vierfach gepufferte OpenGL-Grafikkarte gefunden werden wird eine Fehlermeldung ausgegeben und das Programm beendet.

Während des Startvorganges gibt das Programm Informationen auf `stdout` und Fehlermeldungen auf `stderr` aus, die normalerweise beide zum Bildschirm führen, aber auch umgelenkt werden können. Es werden Dateiinformationen, der Fortschritt des Ladevorganges und eine kurze Anleitung ausgegeben. Im OpenGL-Fenster erscheint das 3-D-Objekt.

Die Steuerung des Programms erfolgt mit Tastatur und Maus. Sie wurde analog zur Standardbelegung vieler anderer Programme gewählt, um von Studierenden intuitiv bedient zu werden. Ziehen mit der linken Maustaste führt eine Rotation um die x- und y-Achsen durch. Ziehen mit der rechten Maustaste bewirkt eine Rotation um die z-Achse oder bei der Schnittebene eine Translation. Diese Aktionen können auch analog mit den Tasten W, S (x-Achse), A, D (y-Achse) und Q, E (z-Achse) durchgeführt werden. Die Tasten Y, C bewirken eine Translation der Schnittebene, X setzt sie auf den Ursprung des Koordinatensystems zurück. Ob nun das Modell, die Schnittebene oder beides manipuliert werden soll, wird mit den Tasten 1,2 und 3 festgelegt. Die Tasten N, M verkleinern und vergrößern das Modell. Die Schnittebene kann mit der Taste O, Transparenz mit der Taste T an- und ausgeschaltet werden. Die Esc-Taste verlässt das Programm. Erwähnte Tasten und einige mehr fasst folgende Übersicht zusammen:

W,S,A,D,Q,E	Rotation
Y,X,C	Translation der Schnittebene
1,2,3	ganze Szene / Schnittebene / Modell wird gedreht
O	Schnittebene ein / aus
T	Transparenz ein / aus
V	wechselt zwischen verschiedenen Achsendarstellungen
N,M	Zoom
Strg-F	Vollbild- / Fenstermodus
Strg-A	anaglyphische Darstellung
Strg-S	stereoskopische Darstellung (nur bei Option `--stereo`)
Strg-M	monoskopische Darstellung
Strg-H	Hilfe (zur Tastenbelegung)
Strg-D	Debug-Informationen
Strg-I	Stereoskopieerfahrene können in der Konsole die Metrik anpassen
Strg-R	zeichnet den Bildschirminhalt manuell neu
Esc	schließt das Programm

Am 8.12. und 9.12.2005 wurde die Software und das Modell im Rahmen der Veranstaltung „Datenvisualisierung mit Amira 4.0“ am Leibniz-Rechenzentrum in München dem Fachpublikum vorgestellt. Auch die Stereoskopiefähigkeit wurde über eine Leinwandprojektion mit der aufwendigen Polarisations- sowie der Anaglyphentechnik präsentiert. Bei der Eröffnungsfeier des Neubaus des Leibniz-Rechenzentrums auf dem Garchinger Forschungsgelände am 21.7.2006 wurde die Software samt Modell und Rack dem geladenen Publikum im Visualisierungslabor präsentiert. Über diese Arbeit wurde auch in der zeitgleich herausgegebenen Sonderausgabe der Zeitschrift „Akademie aktuell“ der Bayerischen Akademie der Wissenschaften in dem Artikel „Virtuelle Welten – ganz real“ referiert [Brossmann 2006b]. In dieser Sondernummer ist auch das konstruierte Beamer-Rack für Stereoprojektionen abgebildet.

4.8 Ausblick

Die technischen Voraussetzungen für moderne Visualisierungen sind durchaus vorhanden, sowohl bei den Instituten als auch in der Hand des Studierenden. Hörsäle verfügen heute über digitale Projektoren, bei weiter verbesserter Ausstattung ist sogar eine verblüffend wirkende dreidimensionale Projektion möglich. Zum Beispiel kann ein echter dreidimensionaler Eindruck monochrom durch das einfache Anaglyphenverfahren [Bach 1999] oder in Farbe mittels Liquid-Crystal-Shutter-Brillen [Bungert 1999] erzeugt werden. Groß angelegte Präsentationen erfordern freilich professionelle Display- und Projektionstechniken [Saad, Hevler 1999], wie die Doppel-Projektion mit polarisiertem Licht (Abbildungen 16a, b und 17).

Virtuelle Modelle werden in Klinik und Forschung zukünftig eine immer bedeutendere Rolle spielen. Ermöglichen sie doch eine fortschrittlichere Art der Diagnose und Therapie. Eine räumliche Visualisierung ist schneller und leichter zu verstehen als eine Unzahl von Schnittbildern. Der Arzt wird kognitiv entlastet und kann sich besser auf die relevanten Informationen konzentrieren. Heute schon werden Operationen am Rechner geplant und geübt. Navigationen werden präziser. Pflesser et al. 2000, Pflesser et al. 2002, Petersik et al. 2002 und Petersik et al. 2003 erläutern die Simulation virtueller Osteotomie am Os petrosum. Ellis et al. 1999 beschreiben in vivo durchgeführte Osteotomien der oberen Tibia. Die CT-Kolonographie ist für Patienten weniger aufreibend, komplikationsärmer, kann Stenosen überwinden und den Dünndarm darstellen [Wessling et al. 2005]. Peters et al. 1996 zeigen auf, wie durch Fusion von MRI mit der digitalen Substraktions-angiographie (DSA) die Gefäße während einer neurochirurgischen Operation geschont bleiben. In der neurochirurgischen Navigation spielen die MRI-Rekonstruktionen eine Hauptrolle [Grimson et al. 1998; Grimson et al. 1996]. Eine neurochirurgischen Epilepsietherapie kann zunehmend präziser ausgeführt werden [Chabrerie et al. 1998]. Die Planung eines kiefer- und gesichtschirurgischen Eingriffs ist heute binnen eines einzigen Tages möglich [Oberzig 2004]. In der Strahlentherapie kann durch ausgefeilte Visualisierungstechniken die Bestrahlung gut geplant und der Patient geschont werden [Levoy et al. 1990].

Die Daten aus medizinischen bildgebenden Verfahren können über DICOM standardisierte Schnittstellen leicht ausgetauscht werden und dann angezeigt werden [DICOM; OsiriX; PixelMed]. Virtuelle Modelle sind digital gespeichert. Je nach Leistungsfähigkeit der Zielplattform können diese Modelle jeweils komplex und hochauflösend für parallele Architekturen (Abbildungen 16d-f) und Client/Server Netzwerke [Mayer, Meinzer 1999] oder einfach mit niedriger Polygonzahl für konventionelle PCs erstellt werden. Solche Daten vereinfachen Transport und Vervielfältigung. Manipulationen wie die beliebige Schnittführung durch die Modelle, Transparenz überlagerter

Schichten, auch veränderbare Färbung und realistische oder abstrakte Texturierung oder letztlich dynamische Simulation sind möglich. Sind die virtuellen Modelle erst einmal erstellt, bieten sie eine ökonomische, ethisch leicht handhabbare, verständliche und zukunftsorientierte Art der Wissensvermittlung, die auch den Erwartungen und der Mentalität der heute Studierenden entgegenkommt.

Angesichts dieser weit entwickelten technischen Möglichkeiten ist der faktische Mangel an virtuellen dreidimensionalen Lehrmitteln nur aus den praktischen Gegebenheiten heraus zu erklären. Zum einen fehlen an den vorklinisch-theoretischen Instituten die Mittel, um die apparative Ausstattung weiter zu verbessern und um virtuelle Modelle zu lizenzieren oder zu erstellen, die den hohen Studentenzahlen gerecht werden. Auch die Lizenzierung kommerzieller kostspieliger Programme findet im Budget kaum Platz. Zum anderen darf zwar vorausgesetzt werden, dass jeder Studierende Zugang zu PC oder Notebook hat, aber diese Geräte sind in der Regel nicht geeignet für umfängliche, komplexe Visualisierungsprogramme, die z. B. ein Drehen des Modells in Echtzeit ohne Ruckeln und Stottern erlauben.

Die Arbeit soll dazu beitragen, diese Widrigkeiten zu umgehen und Studierenden eine zusätzliche zeitgemäße Informationsquelle zu erschließen. Diese dient, wie eingangs erwähnt, als Ergänzung und nicht als Ersatz bestehender Lehrmittel. In der Anatomischen Anstalt der Universität München (Lehrstuhl III, Arbeitsgruppe Heinzeller) entstehen weitere Datensätze zu Embryonen verschiedener Entwicklungsstadien (Abbildungen 18 bis 20). Die entwickelte Software ermöglicht nun eine kostengünstige und unkomplizierte Möglichkeit, solche virtuellen Modelle zu studieren. Das konstruierte Beamer-Rack führt einwandfreie stereoskopische Projektionen durch.

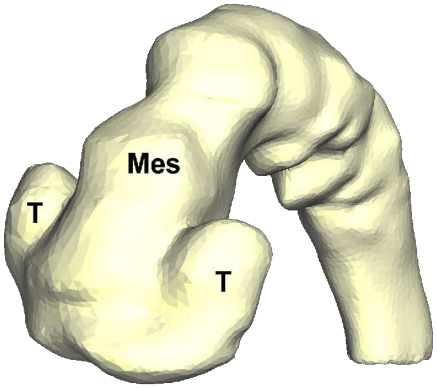


Abb. 18: Anlage des Gehirns.
ME24 hat eine SSL von 3.1 mm.
T Tel-, **Mes** Mesencephalon

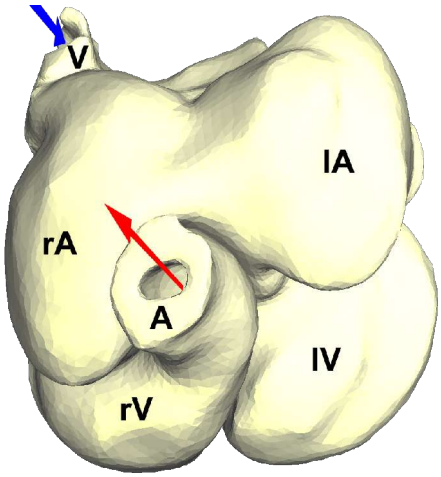


Abb. 19: Herzanlage von ventro-kranial.
ME44 hat eine SSL von 6 mm. Die **Pfeile** zeigen die Blutflussrichtung.
rA rechtes Atrium, **lA** linkes Atrium,
rV rechter Ventrikel, **lV** linker Ventrikel,
A Truncus arteriosus, **V** Sinus venosus

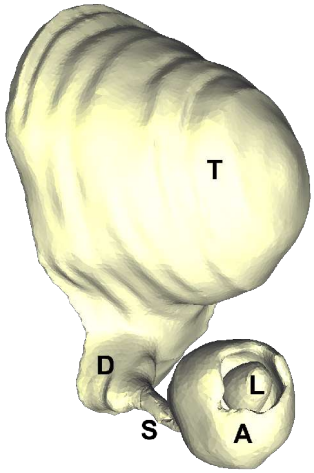


Abb. 20: Anlage des Auges.
ME61 ist 6 Wochen alt.
T Tel-, **D** Diencephalon,
S Augenbecherstiel,
A Augenbecher,
L Linsenbläschen

5 Anhang

5.1 Signal- und Bildverarbeitung

In der *Signalverarbeitung* werden Signale manipuliert. *Signale* aus der realen Welt, wie der elektrische Herzschlag, Geräusche oder Licht, werden zumeist von Sensoren erfasst. Diese Sensoren wandeln die realen Signale heute meist in elektronische Signale um, die dann weiterverarbeitet werden. In der Signalverarbeitung entstehen aus Signalen ständig neue Signale. Mathematisch wird dies als Faltung (Formel 5 auf Seite 67) abstrahiert. Signalverarbeitung kann im kontinuierlichen Fall von analogen Schaltkreisen oder Vorrichtungen[46] vorgenommen werden oder im diskreten Fall von digitalen Prozessoren (DSP, digital signal processing). Die digitale Signalverarbeitung hat die Vorteile, dass sie billig in Fertigung und Entwicklung, nicht von Toleranzen abhängig und daher leicht reproduzierbar ist und keine Nebeneffekte[47] hat.

Digitale *Bildverarbeitung* ist nun ein Spezialgebiet der Signalverarbeitung. Das besondere an einem Bild ist, dass es eine Funktion über den Ort darstellt, während andere Signale meist über die Zeit beobachtet werden. In heutigen Digitalkameras erzeugt Licht über den Photoeffekt auf einem (analogen) CCD (charge-coupled-device)-Chip oder CMOS (complementary metal oxide semiconductor)-Chip Ladungen, deren Höhe von der Helligkeit des Lichtes abhängt und die über die Belichtungszeit integriert werden. Die Ladungen werden vom Chip abgezogen (CCD) oder in ein Potential umgewandelt (CMOS) und in einem AD-Wandler (on-chip) digitalisiert. Dies entspricht der Abtastung.

5.1.1 Abtastung

Im Rahmen der *Abtastung* (engl. sampling) wird im zweidimensionalen Fall typischerweise ein analoges Signal (Licht), mathematisch eine kontinuierliche Funktion f(*i*,*j*) mit $i, j \in \mathbb{R}$, die den Ort auf eine Farbkomponente oder Grauwert abbildet, digital erfasst. Die Diskretisierung der Definitionsmenge (Abtastung des Orts) zu Samples (Pixel) und des Wertebereiches (Quantisierung der Signalamplitude) hat im 2-D-Fall ein Bild $\mathrm{f}_s(x, y)$ der Auflösung $(x_{max}+1)\times(y_{max}+1)$ mit $x \in \{0, 1, \ldots, x_{max}\}$ sowie $y \in \{0, 1, \ldots, y_{max}\}$ zur Folge. Allgemein schreiben wir $F(\mathbf{p})$ für ein Element am Punkt $\mathbf{p}=(x, y)$ eines Bildes *F*. $F(\mathbf{p})$ symbolisiert dabei einen Helligkeits-, Farb- oder Grauwert eines Pixels **p**.

Sampling hat, wie die Signalverarbeitung lehrt, Fehler wie *Aliasing* (da die Forderungen des Nyquist-Theorems oder des Shannon-Sampling-Theorems gerade bei Bildern keinesfalls erfüllt

46 Fotografien wurden früher weichgezeichnet, indem Vaseline auf die Linse aufgetragen wurde.
47 So verursacht ein analoger Tiefpassfilter, wie der Chebyshev, Welligkeit und ungenügende Steilheit.

werden können [Moisan 2003]) und Rauschen zur Folge. Aliasing produziert Signale, die es realiter nicht gibt, im CT etwa beeinflusst unter anderem Aliasing die Größeneinschätzung besonders von kleinen Objekten [Baxter, Sorenson 1981]. Um Aliasing zu verringern wurden Techniken entwickelt, die als *Anti-Aliasing* bezeichnet werden. Im Allgemeinen kommt eine von drei Möglichkeiten zum Einsatz [Hill 2001; Shirley et al. 2005]: Aufwendige *Vorfilterung* (Prefiltering) kann verwendet werden, wenn die Geometrie klar bekannt ist, wie beim Ray-Tracing (Kapitel 5.1.2.4). Für gewöhnlich kommt *Multisampling* zur Anwendung, das für einen Pixel mehrere Samples erzeugt. Das Bild wird sozusagen in einer höheren Auflösung generiert und dann reduziert. Können keine zusätzlichen Samples gewonnen werden, kann *Nachfilterung* (Postfiltering) mit Hilfe eines Weichzeichners (siehe Kapitel 5.2.2.2) erreicht werden.

Die Bilder liegen nach der Abtastung in einem diskreten Ortsbereich vor. Oft wird später wieder ein kontinuierlicher Definitionsbereich gefordert, dann dienen die diskreten Samples als Stützstellen und die benötigte Information wird interpoliert. Dieses *Supersampling* kann auf vielerlei numerische Verfahren zurückgreifen, wie konstante, lineare oder Spline-Interpolation [Bronstein, Semendjajew 1991; Hill 2001]. Können Bilder ihrer Größe wegen nicht verarbeitet werden, muss die Auflösung reduziert werden (*Subsampling*). Subsampling soll, um Aliasing zu vermeiden, analog zum Multisampling nicht durch Verwerfen von Pixeln geschehen (Punktoperation), sondern unter Einbeziehung der Nachbarpixel erfolgen (lokaler Operator).

Im Folgenden sollen einige alternative bildgebende Verfahren erwähnt werden. Diese können in bestimmten Anwendungen, je nach Größe des Präparates, durchaus die klassische Lichtmikroskopie ersetzen und bestimmte bereits diskutierte Probleme umgehen.

5.1.1.1 3-D-Scanner

Unter die Vielzahl der technischen Verfahren zur Erfassung von dreidimensionalen Objekten fallen die berührende und die berührungsfreie Oberflächenabtastung. Ein Beispiel für erstere Methode ist eine Kopierfräsmaschine, Beispiele für letztere sind einfache Digitalisierer [Gembris 1994] oder Laserscanner. Probleme machen hierbei konkave Oberflächen. Mit der CT gemein hat die Oberflächenabtastung eine höhere Auflösung in der gescannten Ebene in Vergleich zur Achse senkrecht zu dieser, sowie die diskreten und damit lückenhaften Informationseinheiten.

5.1.1.2 Tomographie

Die Tomographie ist eine schichtweise Erfassung von Körpern, indem entlang einer Kurve (meist Strecke) einzelne (meist parallele) Schnittebenen erfasst werden. Bekannt wurde dieser Begriff seit

der britische Physiker Godfrey Newbold Hounsfield 1972 das Patent Nr. 1283915 mit dem Titel „A method for an apparatus for examination of a body by radiation such as x-ray or gamma radiation“ zugeteilt bekam. Für die Entwicklung des die Medizintechnik revolutionierenden überlagerungsfreien Röntgenschnittverfahrens teilten sich er und sein unabhängig von ihm arbeitender amerikanischer Kollege Allan M. Cormack 1979 den Nobelpreis für Medizin. Leider hat sich fälschlicherweise mit der Tomographie der Begriff Röntgenschnittverfahren assoziiert, jedoch fällt in die Kategorie Tomographie der Großteil der im Folgenden dargestellten Verfahren, wie Computertomographie (CT), Magnet-Resonanz-Imaging (MRI), Sonographie, Optical-Coherence-Tomography (OCT) oder Optical-Projection-Tomography (OPT).

Moderne Techniken erlauben einen Blick in das Innere des Körpers ohne invasive Maßnahmen. Grob einzuteilen sind diese in Informationsgewinnung durch Projektion (CT, OPT), Sektion (Sonographie, konfokale Mikroskopie, OCT) oder beidem [Sharpe et al. 2002]. Werden die Daten durch Sektion gewonnen, bleibt die räumliche Information erhalten und so können die Rohdaten direkt in Voxel umgewandelt werden. In der projizierenden Tomographie geht eine Dimension zunächst verloren (Integral entlang eines Strahls), kann aber durch Zusatzinformation (mehrere Projektionen durch Rotation) wieder zurückgerechnet werden. Ermöglicht wird die Rückrechnung durch das Fourier-Slice-Theorem. Dieses besagt, dass durch zweidimensionale inverse diskrete Fourier-Transformation aus den Projektionsdaten die Objekte der Ebene näherungsweise hergeleitet werden können [Kak, Slaney 1988].

5.1.1.3 Sonographie

Eine Möglichkeit Daten zu gewinnen ist die 3-D-Ultraschall-Tomographie. Schall im MHz-Bereich wird mittels Piezoelektrik vom auf die Haut aufgesetzten Schallkopf ins Gewebe gesendet. An Gewebegrenzen wird Schall proportional zur Größe der Impedanzänderung reflektiert und von einem Mikrophon detektiert. Aus den Laufzeitunterschieden des Schalls wird die Tiefe der Gewebegrenze berechnet. Je nach Frequenz kann unterschiedlich aufgelöst werden. Eine hohe Frequenz ermöglicht eine bessere Auflösung, die Energie wird jedoch auch in größerem Umfang absorbiert und dann kann nicht besonders gut in die Tiefe geschallt werden. Die gewonnenen Daten können je nach Fähigkeit des Schallkopfes ein-, zwei- oder eben dreidimensional sein.

Ein räumlicher Eindruck der Bilder macht deren Ausmessung und Interpretation einfacher [Gardener 1997], doch kann in der medizinischen Diagnostik auf eine konventionelle Ultraschalluntersuchung durch einen erfahrenen Untersucher nicht verzichtet werden. Da Ultraschalldiagnostik schnell, billig und ungefährlich ist, wird die Nachfrage mit der Entwicklung und Verfeinerung spezieller Visualisierungstechniken immer weiter ansteigen [Sakas et al. 1995; Sakas, Walter 1996].

5.1.1.4 Optical-Coherence-Tomography

Optical-Coherence-Tomography (OCT) ist das Pendant zur Sonographie, nur dass Lichtwellen statt Schallwellen verwendet werden. Die Präparate können bei geringer Auflösung bis zu einer Tiefe von 3 mm erfasst werden.

5.1.1.5 Konfokale Mikroskopie

Marvin Lee Minskys Entwicklung, seinerseits die Mikroskopie seinen speziellen Anforderungen anzupassen, brachte 1955 die damals kaum beachtete konfokale Mikroskopie hervor. Erst Laser- und DSP-Technologien als Dreingaben zur Idee der konfokalen Mikroskopie, präziser nun als C*onfocal-Laser-Scanning-Microscopy* (CLSM) zu bezeichnen, ermöglicht in unseren Tagen eine 3-D-Darstellung von begrenzt dicken Präparaten. Damit tritt diese Technik ihren Siegeszug in der Mikroskopie an.

Analog zur Fluoreszenzmikroskopie ist das Präparat mit einem Fluorochrom einzufärben (Sekundärfluoreszenz). In der Fluoreszenzmikroskopie tritt gefiltertes Licht aus einer Quecksilberdampflampe über einen dichromatischen Spiegel in den Strahlengang und regt den Farbstoff an, der Licht einer längeren Wellenlänge durch denselben dichromatischen Spiegel als Sperrfilter (und zusätzlicher Optik, versteht sich) auf den Schirm bringt.

Bei der CLSM dient ein *Laser* als Beleuchtung, um besonders hohe Intensitäten zu erreichen. Hinzu kommen zwei um eine Achse bewegliche Spiegel, die das Präparat mit 3-300 fps in einer Auflösung von üblicherweise 512 Pixel oder weniger abtasten (*scanning*). Die zentrale Prämisse der konfokalen Mikroskopie bildet ein unscheinbares Nadelloch in der zum Fokus konjugierten Ebene (conjugate to the focal plane, *confocal*). Dieses blendet den nicht fokussierten Beleuchtungsanteil (Hintergrundlichtschleier) aus und erhöht zudem nebenbei die Auflösung. Das Licht des jeweiligen Punktes wird nun von einem Detektor erfasst und von einem Computer als Pixel/Voxel gespeichert.

Da das Nadelloch nur Licht aus der fokussierten Ebene passieren lässt, wird nur eine dünne Schicht des Präparates gezeigt. Nun können noch weitere Ebenen fokussiert werden und so Voxel für Voxel ein diskreter dreidimensionaler Datensatz aufgebaut werden. Die Auflösung beträgt auf der horizontalen Ebene um die 0.2 µm und entlang der vertikalen Achse 0.5 µm [Weeks; Ladic 1995]. Dieser Volumendatensatz wird meist durch spezielle Filter (Deconvolution[48]) geschärft und über direktes Volume-Rendering dargestellt, das an die Datensätze der CLSM angepasst wurde [Razdan et al. 2001; Sakas et al. 1996].

48 Deconvolution rechnet räumliche Unschärfe zurück und erhöht die Bildqualität immens.

CLSM liefert automatisch einen fertig ausgerichteten Datensatz. Dies gilt jedoch nicht mehr, wenn mehrere Farbstoffe verwendet werden und somit mehrere Scans nötig sind. Diese Datensätze sind untereinander nicht ausgerichtet. Ein bedeutender Nachteil ist darüber hinaus, dass das Präparat durch den starken Laser beschädigt wird (Blending). So ist die maximale Anzahl der abtastbaren Ebenen begrenzt. Die Zahl ist stark vom Aufbau abhängig. Die maximale vertikale Ausdehnung des Präparates beträgt ungefähr bis 1 mm, bei optisch weniger transparenten Präparaten, wie Pflanzen, sogar nur um 200 µm [Haseloff 2003]. Als Färbung kommen nur fluoreszierende Farbstoffe in Frage. Auch die Präparateherstellung und die Wahl des Durchmessers des Pinholes (bestimmend für die Beugung) erfordern einiges an Erfahrung.

5.1.1.6 Dual-Inline-Holographie

Bei der *Dual-Inline-Holographie* (DIH) dient ein Laser, der auf ein Nadelloch kleiner der Wellenlänge gerichtet wird als Lichtquelle [Xu et al. 2001]. Das Nadelloch wird dabei neues Wellenzentrum (Beugung an einer kreisförmigen Öffnung) und so Punktlichtquelle für das Präparat. Dieses beugt wiederum das Licht und erzeugt so ein Muster, das, wegen der Zentralprojektion stark vergrößert, als Hologramm auf einem CCD detektiert wird. Ein Computer berechnet aus dem Hologramm wieder den Volumendatensatz des Präparates.

Die Vorteile liegen im kostengünstigen Aufbau und der Möglichkeit lebende Zellen zu beobachten, da keine Färbung nötig ist. Ein fertiger Volumendatensatz ist schnell erstellt. Die theoretische Auflösung liegt bei der vom Lichtmikroskop (Optik). Die tatsächliche bestimmt auch der CCD-Chip. Da für die Rekonstruktion nur ein Bild nötig ist und die Belichtungszeit im Sekundenbruchteil liegt, kann sogar das Wachstum von lebenden Zellen als Video aufgezeichnet werden. Die Einstellung und Messung der Parameter, wie Durchmesser des Präparates, der Abstand zur Punktlichtquelle und zum CCD-Chip machen das Verfahren natürlich aufwendiger im Vergleich zur konventionellen Lichtmikroskopie.

5.1.1.7 Optical-Projection-Tomography

Eine bemerkenswerte nichtzerstörende Technik, Präparate mit einer Ausdehnung um die 15 mm dreidimensional zu erfassen, ist die *Optical-Projection-Tomography* (OPT) [Sharpe et al. 2002]. Wie in einer Projektionstomographie üblich, wird das Präparat von verschiedenen Winkeln aus auf einen Detektor (meist CCD) abgebildet. Nur kommt bei der OPT ein Linsensystem (Mikroskop) hinzu, um das Bild zu vergrößern. Als Beleuchtung dient eine Quecksilberdampflampe. Aufgrund der Optik schleicht sich ein systematischer Fehler (durch Fokussierung) ein, da keine

Linienintegrale gebildet werden, sondern ein spitzer gerader Kegel (Öffnungswinkel um 4.2°) integriert wird. Um dennoch von zumindest einer Hälfte des Präparates eine scharfe Abbildung zu erlangen, wurde die Rotationsachse aus der fokussierten Ebene verschoben.[49] Genaueres ist im Internet zu finden [Sharpe, MRC Genex Homepage], dort sind die Ergebnisse als Videos von Mäusembryonen dokumentiert.

OPT ist eine ökonomische und schnelle Lösung, um einen Volumendatensatz von kleineren Präparaten anzufertigen. Es steht eine große Auswahl an Färbemethoden zur Verfügung. Die Präparate müssen jedoch transparent genug sein und wie beim CT eine geringe Refraktion aufweisen. Die Auflösung kann mit der Mikroskopie jedoch nicht mithalten.

5.1.2 Visualisierung

Der Visualisierungsprozeß veranschaulicht Daten, die gemessen, berechnet oder beobachtet wurden [VisTutor 2003]. Verschiedenste Datentypen können visualisiert werden. Zum einen gibt es Daten vom Typ der booleschen Werte, sogennante *Aussagen*. Solche Aussagen können entweder wahr oder falsch sein (Prinzip vom ausgeschlossenen Dritten), ein Punkt mag zu einem Organ gehören oder nicht (wahr oder unwahr). Zum anderen gibt es Daten, die durch reelle Zahlen ausgedrückt werden, sogennante *Skalare*. Eine physikalische Größe setzt sich bekanntermaßen aus Maßzahl und Einheit zusammen. Die Maßzahlen der SI-Basiseinheiten sind ebensolche Skalare. Schließlich gibt es auch physikalische Größen wie Kraft, Geschwindigkeit und Beschleunigung, die eine Richtung und einen Betrag haben. Sie werden durch Vektoren beschrieben. (Mehr über Vektoren und Matrizen in der 3-D-Grafik in Kapitel 5.2.3.)

Egal ob es sich um Bevölkerungswachstumsraten oder CT-Bilder handelt, die Visualisierungs-pipeline besteht aus drei Schritten, die die Daten zu einem Bild weiterverarbeiten, nämlich Filterung, Mapping und zuletzt Rendering.[50] Im ersten Schritt, der *Filterung*, werden die Rohdaten so weiterverarbeitet, dass sie dem empirischen Modell genügen. Wenn zum Beispiel die Oberfläche der Organe gesucht ist, werden die Daten in eine dreidimensionale Matrix umgewandelt, die einen Ortsbereich auf die Organzugehörigkeit abbildet. Beim nächsten Schritt in der Pipeline, dem *Mapping*, wird eine geometrische Darstellung ermittelt, die im letzten Schritt, dem *Rendering*, in der gewünschten Form ausgegeben wird.

49 Mit DSP könnte man sicher auch diese Unschärfe zurückrechnen.
50 Die Datenakquisition gehört demnach nicht zum Visualisierungsprozess, ist jedoch unumgänglich.

5.1.2.1 Registrierung

Die *Registrierung* (auch Alignment, Fusion oder Ausrichten) ist der Prozess eine Transformation $\hat{\mathbf{T}}$ zu finden, die die Pixel eines Bildes auf die korrespondierenden Pixel eines anderen Bildes abbildet. $\hat{\mathbf{T}}$ richtet also ein variables Bild *M* (movable) auf ein fixes Bild *F* (fix) so aus, dass gleiche Strukturen übereinander zu liegen kommen.

Als Quellmaterial werden je nach Anwendung unterschiedlichste Daten verwendet. Die *intrasubject* Registrierung richtet Datensätze aus, die zu verschiedenen Zeitpunkten vom gleichen Patienten gewonnen wurden, um etwa Tumorwachstum oder eine digitale Substraktionsangiographie darzustellen. Bei der *intersubject* Registrierung werden Datensätze verschiedener Patienten aneinander ausgerichtet, meist um Statistiken anzufertigen oder Vergleiche anzustellen. Bei einer *single/mono-modalen* Registrierung liefert immer das gleiche bildgebende Verfahren die Daten. Hingegen richtet die *multi-modale* Registrierung Bilder verschiedener Verfahren, wie CT, MRI, PET oder Mikroskopie, aufeinander aus.

Die Ausrichtung kann die gesuchte Transformation entweder direkt berechnen oder es kommt ein Optimierungsalgorithmus zum Einsatz. Dieser besteht im Allgemeinen aus den vier Bausteinen *Transformation*, *Bildinterpolation*, *Ähnlichkeitsmetrik* und *Optimierung*. Nachdem eine initiale Transformation (etwa das neutrale Element) gewählt wurde, wird folgende Pipeline iteriert: Die Transformation[51] berechnet das variable Bild neu. Da die Gitter der beiden Bilder nun kaum mehr zur Deckung kommen, ist die Interpolation des Ausgangmaterials nötig. Danach liefert die Ähnlichkeitsmetrik als quantitative Abschätzung in der Regel einen Skalar. Dieser Skalar zeigt die Übereinstimmung des variablen mit dem fixen Bild, die am Ende der Ausrichtung maximal sein soll. Nachdem der Optimierer mit Hilfe der Metrik als Kostenfunktion eine neue Transformation gewählt hat, beginnt die Schleife von vorne.

Im Folgenden wird ein Überblick aus der Sicht eines Entwicklers über die Aspekte der Methoden zur Ausrichtung gegeben. Eine weitere Darstellung findet sich bei Maintz, Viergever 1996 und Maintz, Viergever 1998.

Die *Transformation* kann verschiedene Freiheitsgrade aufweisen. Rigide Transformationen erlauben nur Rotation und Translation. Sie können im 2-D-Fall durch eine (3,3)-Matrix beschrieben werden, deren Determinante 1 und deren unterste Reihe $(0\,0\,1)$ ist. Affine Transformationen unterstützen zusätzlich lineare Skalierung[52], die Determinante muss nicht mehr 1 sein. Die Beschränkung für die unterste Reihe fällt jedoch erst mit der Projektionstransformation. Dann sind

51 Aus technischen Gründen wird die Umkehrabbildung benutzt, also wird das neue Pixelgitter (grid) auf die Quellpixel abgebildet.

52 Scherung kann durch Rotation und Skalierung ausgedrückt werden, ist also nicht fundamental.

auch lineare Verzerrungen möglich. Noch komplexere Transformationen sind ausschließlich nichtlinear. Elastische Transformation kann komplexe Deformationen ausgleichen. Verschiedene mathematische Ansätze aus dem ITK-Paket (Thirion's Demons `itk::DemonsRegistrationFilter` [Thirion 1998] oder das Finite-Elements-Model (FEM) in `itk::fem::FEMRegistrationFilter`) stehen quelltextoffen bereit.

Für die *Interpolation* stehen die üblichen Algorithmen zur Verfügung. Die Pixel dienen dabei als Stützstellen. Einfache Methoden verwenden konstante oder lineare Interpolation. Bei den anspruchsvolleren Methoden setzt sich die interpolierende Spline-Funktion (stückweise Polynome) [Bronstein, Semendjajew 1991; Hill 2001], bei gleicher Rechenarbeit von anderen Verfahren (wie Lagrange oder sinc) ab [Meijering 2000].

Bei der *Ähnlichkeitsmetrik* gibt es eine Vielzahl von Möglichkeiten. So stellt sich die Frage was eigentlich zur Deckung gebracht werden kann. Die *extrinsische* Ausrichtung verwendet artifizielle Marker wie Schrauben oder stereotaktische Rahmen, die dem Patienten angebracht werden. Die *intrinsische* Ausrichtung kommt ohne solche Objekte aus, sie beruht auf einzelnen prominenten Punkten (Landmarken) wie einem Knochenvorsprung, auf Segmentation von Kurven und Oberflächen wie Haut, Knochen oder Sulci [Cachier et al. 2001] oder verwendet das gesamte (meist Graustufen-)Bild. Letzteres Verfahren ist am viel versprechendsten, jedoch bei extremen Rechenaufwand und hoher Komplexität. Um dann eine Metrik zu berechnen, kommen verschiedenste Algorithmen zum Einsatz. Im einfachsten Fall etwa die Minimierung des Abweichungsquadrates (sum of square differences, SSD) (Formel 2 auf Seite 38) oder als besonders fortschrittliche, aber komplexe Methode die Mutual-Information (Formel 3).

Mutual-Information ist ein sehr robuster Algorithmus, der sogar Bilder unterschiedlicher Modalitäten untereinander auszurichten vermag. Viola, Wells 1997 und Wells et al. 1996 übertragen die Entropie von der Nachrichtentechnik auf die medizinische Bildverarbeitung. Die Mutual-Information-Metrik I eines fixen Bildes F und eines variablen Bildes M kann als

$$I(F, \mathbf{T} \circ M) = H(\mathbf{T} \circ M) - H_F(\mathbf{T} \circ M) \tag{3}$$

$$\text{mit der Shannon Entropie } H = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$$

definiert werden. Entropie ist als eine Maßzahl für Information zu verstehen. Diese berechnet sich als ein Wahrscheinlichkeitsmaß aus n Ereignissen der jeweiligen Wahrscheinlichkeit p_i. Die gesuchte Transformation ist somit

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} I(F, \mathbf{T} \circ M) \,. \tag{4}$$

Interpretieren lassen sich die Formeln so: Das eine Bild ist so zu transformieren bis dessen Information möglichst dem anderen Bild gleicht. Dann bringt das zweite Bild kaum einen Informationszuwachs mehr, die bedingte Entropie H_F in (3) wird minimal, der gesamte Term (3) wird maximal und (4) liefert die entsprechende Transformation. Eine Hinführung auf das ungewöhnliche Thema liefern Pluim et al. 2003. Mutual-Information ist anderen Techniken überlegen, so auch nichtlinearen oder statistischen Ansätzen, ja sogar der extrinsischen Registrierung[53]. So kann ein 3-D-Datensatz sehr genau auf das ursprüngliche Objekt im Raum (z. B. das Objektkoordinatensystem eines CT auf das Weltkoordinatensystem im OP) ausgerichtet werden, indem von diesem möglichst mehrere 2-D-Bilder gemacht werden [Leventon et al. 1997]. Sie ist eine der wenigen bemerkenswerten Möglichkeiten funktionale Bildgebung (fMRI, PET, SPECT) an nativer Bildgebung (CT, MRI) auszurichten [Tsai et al. 1999]. ITK bietet für die hoch rechenintensive Mutual-Information-Metrik zwei Implementationen an: `itk::MutualInformation-ImageToImageMetric` [Viola, Wells 1997] und `itk::MattesMutualInformationImageToImage-Metric` [Mattes et al. 2001].

Der *Optimierer* ist das waltende Gehirn und regelt die Schleife. Er muss das Optimierungsproblem lösen und die beste Transformation finden. Als Kostenfunktion steht ihm die Metrik zur Verfügung. Als Algorithmen dienen die üblichen Lösungen für Optimierungsprobleme, wie Evolution, Downhill Simplex, Levenberg-Marquardt (alle in ITK) und viele mehr. So kann bei einer single-modality Registrierung der schwarz-weiß Gradient für die Optimierung verwendet werden (`itk::RegularStepGradientDescentOptimizer`). Um die Geschwindigkeit zu erhöhen und um nicht in lokale Extrema zu fallen, können multiresolution Ansätze (Pyramide) helfen: Durch Subsampling werden mehrere Auflösungen erzeugt und mit der gröbsten die Optimierung begonnen. Die erhaltene Transformation dient als Initialwert für die nächst feinere Auflösung, bis man sich zur Originalauflösung vorgearbeitet hat. Andererseits kann auch eine schnelle, aber ungenaue Optimierung die Initialtransformation für einen leistungsfähigeren Algorithmus liefern.

5.1.2.2 Segmentierung

Segmentierung wird der Schritt genannt, der das gewünschte Objekt im Datensatz markiert. Anders gesagt, bezeichnet dieser Schritt die Zuordnung eines Voxels zu einer entsprechenden anatomischen Einheit, also das Verbinden von Bilddaten mit medizinischem Wissen. Aufgrund der diskreten Natur der Ursprungsdaten fallen die Grenzen zweier Objekte meist in ein oder mehrere Pixel (Partialvolumeneffekt) zusammen, getrennte Zuordnungen können dann nur auf Subpixel-Ebene getroffen werden.

53 Diese wird gerne im kurativen Bereich (z. B. OP, Radiation) angewendet.

5.1.2.3 Mapping

Mapping wandelt den Volumendatensatz in eine geometrische Repräsentation um (3-D-Rekonstruktion). Die Grenzen der binär segmentierten Objekte (Isosurface) werden als Oberflächennetz beschrieben. Die dreidimensionale Oberfläche wird meist aus Dreiecken aufgebaut (Triangulation).

Das Verfahren, das standardmäßig zur Verwendung kommt, ist der *Marching-Cubes-Algorithmus* [Lorensen, Cline 1987]. Dabei wird in jedem Schritt ein Würfel (*cube*) mit acht Voxeln an den Ecken betrachtet. Für jede Ecke wird kontrolliert, ob sie innerhalb oder außerhalb des Objektes liegt. Die gesuchte Oberfläche schneidet die Würfelkanten so, dass die inneren Voxel auf der einen Seite und die äußeren auf der anderen Seite liegen. Dabei gibt es $2^8 = 256$ Möglichkeiten. Lässt man die symmetrischen Fälle weg, reduziert sich die Anzahl auf 15. Diese Fälle werden linear interpoliert (adaptive Marching-Cubes), trianguliert und die erhaltenen Vertices mit Normalen versehen. Dann kann mit dem nächsten Würfel fortgefahren (*marching*) werden. Auftretende Zweideutigkeiten behindern eine korrekte Oberflächenrekonstruktion. Da die entstehende Datenmenge gewaltig ist, wird in einem weiteren Schritt die Anzahl der Dreiecke reduziert.

5.1.2.4 Rendering

Kurz gesagt ist Rendering der Prozess, realistische Abbilder einer Welt zu erzeugen. Das Rendering projiziert die ermittelte dreidimensionale geometrische Repräsentation auf eine Ebene, gibt diese grafisch aus und steht somit am Ende der Visualisierungspipeline. Zwei übliche Techniken existieren für Volumendatensätze: indirektes Surface-Rendering und direktes Volume-Rendering.

Um dreidimensionale Objekte auszugeben, wird häufig das *Surface-Rendering* (Oberflächen-Rendering) angewandt. Das Verfahren ist wohldefiniert und dessen Algorithmen in handelsüblichen Grafikchipsätzen implementiert. Die (schon heraussegmentierte) Oberfläche des Objekts wird modelliert, und zwar durch Vernetzung einfacherer geometrischer Formen (Primitiven), wie Dreiecken, Polygonen oder Splines [Schönherr 2004]. Die Anzahl der Primitiven übersteigt meist handhabbare Dimensionen und muss daher reduziert werden. Haben diese einen geringeren Speicherplatzbedarf, können sie bei limitierter Bandweite schneller übertragen werden (auch vom Hauptspeicher in den Grafikkartenspeicher) und können durch gängige Grafikkartenhardware schnell gerendert werden [Bertuch 2000; Bertuch 2002]. Das Rendering hat in diesem Fall eine gute Ausführungsgeschwindigkeit, da der Datensatz nur einmal durch das aufwendige Mapping in Geometrien umgewandelt wird. Besonders günstig sind Dreiecke, da deren Ecken immer in einer Ebene liegen.

Allerdings gehen beim Mapping Informationen verloren, etwa zu kleine oder unscharfe Strukturen, aber auch die (bei Tumoren wertvolle) Grauwertinformationen. Innen liegende Objekte bleiben verdeckt. Sie können zwar durch transparente Oberflächen sichtbar gemacht werden, doch geht hierbei der Großteil des Geschwindigkeitsvorteils verloren, da aufwendigere Algorithmen zur Anwendung kommen[54]. Im Extremfall können die Ansprüche zum extrem rechenintensiven *Ray-Tracing* führen. Hier wird mit Strahlenrückverfolgung versucht, die optischen Gesetze (Reflexion, Diffraktion, Refraktion, Absorption, Wellenlänge) und die Physiologie des Sehens (Farben) möglichst realistisch zu imitieren (nicht zu simulieren) [Rückert 1988; Hill 2001; Shirley et al. 2005]. Dazu wird ein Beleuchtungsmodell, wie das Phong-Modell[55] (samt Interpolation via Gourand- und Phong-Shading) entwickelt, das möglichst naturgemäß und schnell zu berechnen ist.

Für *direktes Volume-Rendering* ist eine vorherige Segmentation nicht unbedingte Voraussetzung. Es arbeitet direkt mit den Farbinformationen der Voxel. Anatomisches Wissen ist daher nicht nötig, es müssen keine Strukturen verarbeitet werden. Das ist ein großer Vorteil, denn der Schritt der Segmentation ist sehr aufwendig, wenn (bei malignen Tumoren) überhaupt möglich. Die erzeugten Bilder wirken ästhetisch und faszinierend. Wenn sie dennoch segmentiert wurden, sind sie informativer [Hesser, Männer 1999]. Diese Render-Technik benötigt darüber hinaus kein Mapping, da als Repräsentation schließlich keine geometrischen Grundformen verwendet werden, sondern das Datenvolumen direkt dargestellt wird. Doch sind Render-Algorithmen noch immer Grundlage intensiver Forschung. Einiger Entwicklungsaufwand, viel Speicherplatz und eine schnelle Plattform sind Grundvoraussetzung, da der Großteil der Implementationen auf unkonventioneller Software [Levoy 1988; Meinzer et al. 1991] basiert und auf Hardwarebeschleunigung[56] kaum zurückgegriffen werden kann.

Die Interpretation[57] von und die Navigation[58] in diesen wolkig anmutenden Bildern stellt sich schwieriger dar als beim Oberflächen-Rendering. Problematisch wird es, ein realistisches Beleuchtungsmodell zu simulieren. Während bei obigen Primitiven direkt Oberflächennormalen zu bestimmen sind, gelingt dies hier nicht mehr. Über Gradienten im Subvoxelbereich können diese dennoch berechnet werden [Tiede et al. 1998].

54 Genaugenommen müssen für jeden Blickwinkel alle Vertices sortiert werden, was sehr langwierig ist.
55 Aus Oberflächennormale, Sehstrahl und Lichtstrahlen werden Streulicht, diffuse und spiegelnde Reflexion berechnet.
56 Es gibt Hardware von Mitsubishi, die Rendering von Volumen beschleunigt [VolumePro 500].
57 Vorausgehende Segmentation erleichtert diese doch erheblich.
58 Vergrößerung der Voxel offenbart deren diskrete Natur.

5.2 Mathematische Grundlagen der Computergrafik

Viele der zitierten Arbeiten setzen mathematisches Fachwissen voraus. Dem Verständnis auch der in dieser Arbeit besprochenen Formeln und Zusammenhänge sowie der entwickelten Software dient dieser Teil des Anhangs.

5.2.1 Faltung

Faltung (engl. *convolution*) beschreibt, wie aus einem Eingangssignal das gewünschte Ausgangssignal wird. Sie hat in der Signalverarbeitung, also auch in der Bildverarbeitung zentrale Bedeutung.

Betrachten wir zum Einstieg den kontinuierlichen eindimensionalen Fall. Ein Eingangssignal s(t) soll gemessen werden. Dazu wird dieses von einer Reihe von Impulsen zerlegt, also eine Impuls-Dekomposition durchgeführt. Ein Signal kann nun Sample für Sample betrachtet werden. Ein *Impuls* ist eine Abbildung, die nur an einer Stelle ungleich 0 ist. Die *Dirac*[59]*-Delta-Funktion* $\delta(t)$ ist ein Impuls, dessen Fläche auf 1 normiert ist, also der *Einheitsimpuls*. Sie kann als

$$\delta(t) = \lim_{n\to\infty} \delta_n(t) = \lim_{n\to\infty} n\,\mathrm{rect}(nt)$$

mit $\mathrm{rect}(t) = \begin{cases} 1 & |t| < 1/2 \\ 0 & \text{sonst} \end{cases}$ definiert werden. Die geforderte Eigenschaft $\int_{-\infty}^{\infty} \delta(t)\,dt = 1$ wird erfüllt. Ein Einheitsimpuls tastet ein Signal s(t) an genau einem Punkt $t = t'$ ab und liefert als System-Antwort einen idealen Messwert $\mathrm{s}(t') = \int_{-\infty}^{\infty} \mathrm{s}(t)\,\delta(t-t')\,dt$. Einheitsimpulse ahmen demnach ein ideales Messgerät nach. Die Wirklichkeit sieht anders aus, ein Messinstrument ist träge und wird vom Signal über einen größeren Bereich beeinflusst. Diese individuelle Eigenschaft eines Messinstruments wird als *Impuls-Antwort* h(t) dargestellt und wird erhalten, wenn ein Einheitsimpuls zum Zeitpunkt $t = t_0$ als Eingangssignal dient:

$$\mathrm{h}(t-t_0) = \int_{-\infty}^{\infty} \delta(t'-t_0)\,\mathrm{h}(t-t')\,dt'$$

Mit dem gewonnenen Wissen der Impuls-Antwort h(t) kann die System-Antwort m(t) eines beliebigen Signals s(t) vorausgesagt werden. Das spielt eine zentrale Rolle in der Faltung. Aus der unbekannten Blackbox ist ein wohldefiniertes Modell entstanden. Es ergibt sich daher als Messkurve oder *System-Antwort*

$$\mathrm{m}(t) = \int_{-\infty}^{\infty} \mathrm{s}(t')\,\mathrm{h}(t-t')\,dt'$$

59 Paul Adrien Maurice Dirac (1902 – 1984), britischer Physiker und Nobelpreisträger (Physik 1933)

was als *Faltung* bezeichnet wird und lautet in üblicher Kurzschreibweise

$$\mathrm{m}(t)=(\mathrm{s}*\mathrm{h})(t)\,. \tag{5}$$

Die *diskrete Faltung* bei einer Schrittweite von τ entspricht

$$m_i=\mathrm{m}(\tau i)=\tau\sum_{k=-\infty}^{\infty}\mathrm{s}(\tau k)\;\mathrm{h}(\tau(i-k))\,.$$

Zusammengefasst ist Faltung ein linearer Operator der das Ausgangssignal (System-Antwort m) als Integral des Produktes des Eingangssignals s mit gewichteten Koeffizienten (Impuls-Antwort h) beschreibt. Aus zwei Signalen wird ein drittes erzeugt. Genaueres ist in einschlägiger Literatur über Digital Signal Processing (DSP) zu finden [Smith 1999; Kak, Slaney 1988].

5.2.2 Fourier und Filter

5.2.2.1 Fourier-Transformation

Der französische Mathematiker und Physiker Baron Jean Baptiste Joseph Fourier (1768 – 1830) begründete ab 1820[60] die Fourier-Analyse und die Fourier-Reihen. Die *Fourier-Analyse* oder *harmonische Analyse* zerlegt eine Funktion in eine Reihe elementarer Sinus- und Kosinusfunktionen. Diese Reihe aus trigonometrischen periodischen Funktionen heißt *Fourier-Reihe*. Unendliche Reihen werden jedoch zweckdienlich abgebrochen und ein endliches trigonometrisches Polygon als Approximation erzeugt.

Die *Fourier-Transformierte* (Frequenzdomäne) definiert sich (im kontinuierlichen Fall) als

$$\mathrm{S}(\omega)=\int_{-\infty}^{\infty}\mathrm{s}(t)\mathrm{e}^{-i\omega t}\,dt$$

während sich die *inverse Fourier-Transformierte* (Orts- oder Zeitdomäne) als

$$\mathrm{s}(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\mathrm{S}(\omega)\mathrm{e}^{i\omega t}\,d\,\omega$$

darstellt, mit $\mathrm{e}^{i\omega t}=\cos(\omega t)+i\sin(\omega t)$, $i^2=-1$, der Kreisfrequenz $\omega=2\pi f$ und der Frequenz *f*.

In der Frequenzdomäne lassen sich bestimmte Probleme leichter lösen. Die Faltung stellt sich dort statt eines Integrals als einfache Multiplikation dar[61]. Die Impuls-Antwort kann als Filterkern dienen. Bestimmte Filter, etwa Tiefpassfilter, können in der Frequenzdomäne leichter definiert werden. Zur Verwendung kommen hierfür die verschiedensten Funktionen. Eine wichtige Rolle zur Weichzeichnung spielt die Gauß-Funktion.

60 Tatsächlich wurde die Arbeit schon 1807 fertig gestellt, jedoch wegen der Bedenken Joseph Louis Lagranges (1736 – 1813) zurückgewiesen.

61 Wird bei der Fast-Fourier-Transformations-Faltung (FFT) genutzt.

5.2.2.2 Filter

Ein Filter in der Bildverarbeitung manipuliert ein Bild und hebt zumeist die wichtigen Informationen heraus, während unerwünschte Informationen und Fehler, wie das Rauschen, verschwinden. Dass dies in der Regel nur näherungsweise gelingt, versteht sich von selbst. Je nachdem ob nun für den jeweiligen neuen Pixel nur der Eingangspixel, seine Nachbarschaft oder das ganze Bild als Ortsfunktion einbezogen werden, wird von Punktoperation, lokaler Operation oder globaler Operation gesprochen.

Punktweise Operatoren dienen oft der Histogrammanpassung. Lineare Operatoren können dort den dynamischen Umfang der Intensitäten verändern oder die Intensitätsverteilung auf definierten Mittelwert und Standardabweichung setzen und normalisieren. Dies ist oft für medizinische Ausrichtung (Kapitel 5.1.2.1) sinnvoll. Eine Sigmoidkurve als Beispiel eines nichtlinearen Operators betont bestimmte Intensitätswerte und findet seine Bestimmung oft in der Segmentation (Kapitel 5.1.2.2). Als deren Extremausführung kann der Schwellwertoperator (Thresholding) angesehen werden. Dieser punktweise Operator erzeugt einen binären Datensatz.

Erst *lokale Operatoren* sind Filter im Sinne der Signalverarbeitung. Sie beziehen die Nachbarpixel mit in die Berechnung des aktuellen Pixels ein (Faltung). So kann auch in der Frequenzdomäne gearbeitet werden. Dort werden dann bestimmte Frequenzen verstärkt oder abgeschwächt. Unentbehrlich ist hierbei natürlich die Fourier-Transformation, die die Zeit-/ Ortskurve in ihre Frequenzdomäne überführt und zurück. *Tiefpassfilter* dämpfen oder beseitigen hochfrequente Anteile. So werden hochfrequente Störungen im Rahmen der Rauschunterdrückung gemindert. Dabei werden jedoch auch kleine Details verwischt. Bilder werden so von Median- oder Gauß-Filtern geglättet. Die Gauß-Funktion ist Fourier-symmetrisch, sieht also in beiden Domänen wie eine Glockenkurve aus. Darüber hinaus ist sie im n-dimensionalen Fall isotropisch, die Wirkung hängt also nur vom eindimensionalen Abstand zweier Werte, nicht von der n-dimensionalen Position ab. Dadurch können die Dimensionen separiert werden und in der Implementation durch n eindimensionale Filterkerne ersetzt werden. Nicht zuletzt ist sie näherungsweise lokal begrenzt, mit wachsender Entfernung sinkt der Einfluss auf das Ergebnis, es genügt alson nur die nähere Umgebung zu betrachten. Im Gegensatz dazu sollen *Hochpassfilter* niederfrequente Anteile dämpfen und dienen daher der Kantenverstärkung. Gradienten, wie Hell-Dunkel-Übergänge und Kanten, werden durch Hochpassfilterung hervorgehoben. So kann der Laplace-Operator Bilder schärfen, erzeugt dabei jedoch auch Artefakte, wie Hochfrequenzrauschen. Das Bild wird körniger.

Um Berechnungen elegant durchzuführen, werden Filter gerne durch Kerne beschrieben. So hat ein 3^2-Gauß-Operator den Filterkern $G=\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$, während $L=\begin{bmatrix}-1 & -1 & -1\\ -1 & +8 & -1\\ -1 & -1 & -1\end{bmatrix}$ hochpass filtert. Diese Kerne kennt man auch aus der Bildverarbeitung, wo als Signal s(t) ein Bild F und als Impuls-Antwort h(t), dem Filter, der Kern H angewendet wird. Über eine Faltung mit der Gauß-Funktion als Kernel, lässt sich dann leicht ein gewichteter lokaler Durchschnittswert $(F*G)(\mathbf{p})$ errechnen, was einer Weichzeichnung beziehungsweise einem Tiefpassfilter entspricht.

Lokale Operatoren betrachten näherungsweise eine begrenzte Umgebung. Würde der Gauß-Operator, der sich im Unendlichen der Null annähert, nicht abgeschnitten werden, so würden unnötig viele mathematische Operationen durchgeführt werden. *Globale Operatoren* schließlich können eine solche Approximation auf eine begrenzte Nachbarschaft nicht hinnehmen und verwenden alle verfügbaren Pixel. Sie beanspruchen deswegen viel Rechenkapazität.

5.2.3 Lineare Algebra

5.2.3.1 Vektoren

Die Vektorrechnung als Teilbereich der linearen Algebra schafft einen idealen mathematischen Rahmen für die Computergrafik. Einige Besonderheiten dieser dreidimensionalen analytischen Geometrie sollen im Folgenden erläutert werden. Der zweidimensionale Fall ist entsprechend ähnlich gelagert (die z-Komponente fehlt) und wird daher nicht weiter erwähnt.

Ein *Vektor* darf als gerichtete Strecke angesehen werden. Mathematisch stammen unsere Vektoren **v** aus einem euklidischen Vektorraum **V**, einem Vektorraum, der mit Hilfe des Skalarproduktes eine Metrik in Form von Länge (Betrag, euklidische Norm ||**v**||) des Vektors und Winkel φ zwischen zwei Vektoren erhalten hat.

Einem *Koordinatensystem* $\Sigma=\{\vartheta, \mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z\}$ sind Ursprung ϑ und linear unabhängige[62] Basisvektoren $\mathbf{e}_x$ (Abszisse), $\mathbf{e}_y$ (Ordinate) und $\mathbf{e}_z$ (Applikate) zugrunde gelegt. Damit lässt sich jeder Vektor **v** als Linearkombination $\mathbf{v}=v_x\mathbf{e}_x+v_y\mathbf{e}_y+v_z\mathbf{e}_z$ darstellen. Die Koeffizienten[63] heißen *affine* (auch *kontravariante*) *Koordinaten* oder *Parallelkoordinaten*, kurz $\mathbf{v}=(v_x, v_y, v_z)$ oder als

62 also nicht komplanar – nicht einer Ebene parallel
63 Sind die Koeffizienten null, erhält man den Nullvektor **o** = (0, 0, 0).

Spaltenmatrix $\mathbf{v}=\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$. In einem *rechtshändigen Koordinatensystem*, welches meist verwendet wird, sind die Basisvektoren wie der Daumen, der Zeigefinger und der Mittelfinger der rechten Hand angeordnet. Beruhen die Basisvektoren auf orthogonalen Einheitsvektoren (**i**, **j** und **k**), dann spricht man von einem *kartesischen* (auch *rechtwinkligen* oder *orthonormalen*) Koordinatensystem, die Koeffizienten heißen *kartesische Koordinaten*.

In der Computergrafik ist eine Unterscheidung zwischen Vektoren (Richtungen) $\mathbf{v}=v_x\mathbf{e}_x+v_y\mathbf{e}_y+v_z\mathbf{e}_z+0\vartheta$ und Punkten (Bezug zum Ursprung) $\overrightarrow{\vartheta P}=P=p_x\mathbf{e}_x+p_y\mathbf{e}_y+p_z\mathbf{e}_z+1\vartheta$ praktisch. Im *homogenen Koordinatensystem* (auch *Verhältniskoordinatensystem*) bestimmt das Verhältnis der Koeffizienten zur homogenen Komponente[64] h die Lage des Punktes $P=(p_x, p_y, p_z, h)=\frac{p_x}{h}\mathbf{e}_x+\frac{p_y}{h}\mathbf{e}_y+\frac{p_z}{h}\mathbf{e}_z+\vartheta$ zum Ursprung. Üblicherweise ist bei einem Punkt $h = 1$ (allgemein $h \neq 0$) und bei einem Vektor $h = 0$.[65] Homogene Koordinaten ermöglichen perspektivische Transformation und Translation über einfache Multiplikation mit einer Matrix. In der Computergrafik ist ein rechtshändiges orthonormales homogenes Koordinatensystem üblich.

Eine wichtige Operation ist das *Skalarprodukt* $\mathbf{ab}=|\mathbf{a}||\mathbf{b}|\cos\varphi$ (φ ist der Winkel zwischen den Vektoren), was im kartesischen Koordinatensystem $a_xb_x+a_yb_y+a_zb_z$ entspricht. Es ist $\mathbf{aa}=|\mathbf{a}|^2$ und wenn $\mathbf{ab} = 0$ ist $\mathbf{a}\perp\mathbf{b}$. Es gelten die Kommutativität $\mathbf{ab}=\mathbf{ba}$, Assoziativität nur mit einem Skalar $(s\,\mathbf{a})\mathbf{b}=s(\mathbf{ab})$ (jedoch ergibt $(\mathbf{ab})\mathbf{c}$ nicht unbedingt $\mathbf{a}(\mathbf{bc})$) und Distributivität $\mathbf{a}(\mathbf{b}+\mathbf{c})=\mathbf{ab}+\mathbf{ac}$.

Eine weitere Operation zweier Vektoren ist das *Vektorprodukt* oder *Kreuzprodukt* $\mathbf{a}\times\mathbf{b}$. Dies ergibt einen Vektor der Länge $|\mathbf{a}||\mathbf{b}|\sin\varphi$ (Fläche des aufgespannten Parallelogramms), der rechtshändig orthogonal auf **a** und **b** steht. Im kartesischen Koordinatensystem gilt daher $\mathbf{a}\times\mathbf{b}=\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}=\begin{pmatrix} a_yb_z-a_zb_y \\ a_zb_x-a_xb_z \\ a_xb_y-a_yb_x \end{pmatrix}$. Es ist $\mathbf{a}\times\mathbf{b}=\mathbf{o}$, falls **a** und **b** linear abhängig. Es gilt die Distributivität $\mathbf{a}\times(\mathbf{b}+\mathbf{c})=\mathbf{a}\times\mathbf{b}+\mathbf{a}\times\mathbf{c}$ und die Assoziativität nur mit einem Skalar $(s\,\mathbf{a})\times\mathbf{b}=s(\mathbf{a}\times\mathbf{b})$ (jedoch ergibt $(\mathbf{a}\times\mathbf{b})\times\mathbf{c}$ nicht unbedingt $\mathbf{a}\times(\mathbf{b}\times\mathbf{c})$). Die Kommutativität $\mathbf{a}\times\mathbf{b}=-(\mathbf{b}\times\mathbf{a})$ ist invers. Es ist $\mathbf{i}\times\mathbf{j}=\mathbf{k}$, aber $\mathbf{e}_x\times\mathbf{e}_y$ ergibt nicht unbedingt $\mathbf{e}_z$.

64 eingeführt durch den deutschen Mathematiker und Astronomen August Ferdinand Möbius (1790 – 1868)

65 Nähert sich die homogene Komponente dem Grenzwert 0, so entfernt sich ein Punkt immer weiter vom Ursprung, bis er „nur“ noch eine Richtung repräsentiert und zum Vektor geworden ist (unendlich weit entfernter Punkt).

5.2.3.2 Matrizen

Matrizen ermöglichen die bequeme lineare Manipulation von Vektoren und können zwischen Koordinatensystemen transformieren. Eine (m,n)-Matrix $\mathbf{A}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \\ a_{m1} & & a_{mn} \end{pmatrix}=(a_{ik})$ der Elemente a_{ik} wird als rechteckiges Feld notiert. In der 3-D-Computergrafik sind (3,3)- oder für homogene Systeme (4,4)-Matrizen[66] üblich. Bei der *Einheitsmatrix* **N**, dem neutralen Element der Multiplikation, sind die Hauptdiagonalelemente 1, alle anderen Elemente 0, also $n_{ik}=\begin{cases} 0 & \text{für } i\neq k \\ 1 & \text{für } i=k \end{cases}$.

Die *Transponierte* einer Matrix ist $\mathbf{A}^{\mathrm{T}}=(a_{ki})$. Eine wichtige Größe einer quadratischen Matrix ist ihre *Determinante* $D=\det \mathbf{A}=\det (a_{ik})=|a_{ik}|$ und definiert sich als $\sum_{\pi}(-1)^{j(\pi)}a_{1i_1}a_{2i_2}\ \ldots\ a_{ni_n}$[67].

Die Determinante einer *singulären* Matrix ist gleich 0. Die Vektoren solch einer Matrix sind dann linear abhängig, das neue Koordinatensystem hat weniger Dimensionen. Geometrisch wird der Raum auf eine Ebene (oder sogar weniger) abgebildet. Die Matrix ist daher nicht umkehrbar[68]. Eine *reguläre* Matrix hat eine Determinante ungleich 0 und besteht aus linear unabhängigen Vektoren. Die Größe der Determinante gibt bemerkenswerterweise das Maß der Skalierung einer Transformation an, das heißt wenn D = 1 dann ändern sich die Volumina, Flächen und Längen von Objekten nicht. Wenn D = 2 dann verdoppeln diese sich entsprechend.

5.2.3.3 Affine Transformation

In der Computergrafik wird durch die Multiplikation mit einer Matrix **A** das Koordinatensystem gewechselt. Solche affinen Transformationen bestehen aus den fundamentalen Komponenten Translation, Skalierung und Rotation. Die *Inverse* (Reziproke) einer Matrix ist $\mathbf{A}^{-1}$ und es gilt, falls **A** regulär ist, $\mathbf{A}\mathbf{A}^{-1}=\mathbf{A}^{-1}\mathbf{A}=\mathbf{N}$. Durch die Multiplikation mit der Inversen $\mathbf{A}^{-1}$ kann die Transformation **A** rückgängig gemacht werden.

Eine Translationsmatrix und deren Inverse sehen folgendermaßen aus:

66 Auch quadratische Matrix (n=m) der Ordnung 3 bzw. 4 genannt.

67 π symbolisiert die n! Permutationen der Zahlen 1 bis n, der ausformulierte Term ist in der Literatur nachzuschlagen. Der Term besteht aus immerhin n!-1 Additionen und n!(n-1) Multiplikationen...

68 Die Determinante geht als Reziproke in die Gleichung ein und bedingt dann eine Division durch 0.

$$\mathbf{A}=\begin{pmatrix}1 & 0 & 0 & a_x\\ 0 & 1 & 0 & a_y\\ 0 & 0 & 1 & a_z\\ 0 & 0 & 0 & 1\end{pmatrix} \text{ mit } \mathbf{A}^{-1}=\begin{pmatrix}1 & 0 & 0 & -a_x\\ 0 & 1 & 0 & -a_y\\ 0 & 0 & 1 & -a_z\\ 0 & 0 & 0 & 1\end{pmatrix}. \tag{6}$$

Eine Skalierung wird bewerkstelligt durch die symmetrische Matrix

$$\mathbf{A}=\begin{pmatrix}a_x & 0 & 0 & 0\\ 0 & a_y & 0 & 0\\ 0 & 0 & a_z & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \text{ mit } \mathbf{A}^{-1}=\begin{pmatrix}1/a_x & 0 & 0 & 0\\ 0 & 1/a_y & 0 & 0\\ 0 & 0 & 1/a_z & 0\\ 0 & 0 & 0 & 1\end{pmatrix}, \tag{7}$$

eine Rotation um die z-Achse liefert schließlich die orthogonale Matrix

$$\mathbf{A}=\begin{pmatrix}\cos(\alpha) & -\sin(\alpha) & 0 & 0\\ \sin(\alpha) & \cos(\alpha) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \text{ mit } \mathbf{A}^{-1}=\mathbf{A}^{\mathrm{T}}. \tag{8}$$

Rotationsmatrizen um die anderen Achsen sind analog aufgebaut. Um Transformationen zu kombinieren, werden sie multipliziert. Die Inverse einer allgemeinen (n,n)-Matrix erhält man über

$$\mathbf{A}^{-1}=\frac{1}{\det\mathbf{A}}\begin{pmatrix}A_{11} & \cdots & A_{1n}\\ \vdots & \ddots & \vdots\\ A_{n1} & \cdots & A_{nn}\end{pmatrix}^{\mathrm{T}}, \tag{9}$$

wobei A_{ik} das algebraische Komplement[69] (Adjunkt) des Elements a_{ik} ist. Ausformuliert entsteht ein recht umfangreiches Gleichungssystem von über 300 Operationen bei einer (4,4)-Matrix.

69 Das Adjunkt ist die mit $(-1)^{i+k}$ multiplizierte Unterdeterminante einer Matrix, die durch Weglassen der *i*-ten Zeile und der *k*-ten Spalte aus der ursprünglichen Matrix erzeugt wird.

6 Verzeichnisse

6.1 Abbildungen

6.2 Literatur

1. Ackerman, M. J.: Accessing the Visible Human Project. URL *http://www.dlib.org/dlib/october95/10ackerman.html* 1995
2. AIR: Automated image registration. URL *http://bishopw.loni.ucla.edu/AIR3/* 2002
3. Alexandrescu, A.: *Modern C++ design: Generic programming and design patterns applied*. 1. Aufl. Boston : Addison-Wesley, 2001
4. Amira: 3D Data Visualization. URL *http://www.amiravis.com*
5. Amira 3.0: *User's guide and reference manual*. 1. Aufl. Berlin : ZIB, Indeed, 2002
6. Bach, M.: Räumlich durchs Auge. In: *c't* (1999) 7, 158-162
7. Baxter, B. S. ; Sorenson, J. A.: Factors affecting the measurement of size and CT number in computed tomography. In: *Investigative Radiology* 16 (1981) 4, 337-441
8. Bertuch, M.: Aufklärung in 3D. In: *c't* (2002) 15, 194-201
9. Bertuch, M.: Polygonfabriken. In: *c't* (2000) 8, 202-207
10. Blaas, H.-G. ; Eik-Nes, S. H. ; Berg, S. ; Torp, H.: In-vivo three-dimensional ultrasound reconstructions of embryos and early fetuses. In: *Lancet* 352 (1998) 10, 1182-1186
11. Blitz++: Object-Oriented Scientific Computing. URL *http://www.oonumerics.org/blitz*
12. Boost: C++ libraries. URL *http://www.boost.org*
13. Brauer, M.: *Embryonale Oberflächenmorphologie in der transvaginalen 3D-Sonographie*. 1. Aufl. Berlin : Dissertation, 2004
14. Bronstein, I. N. ; Semendjajew, K. A.: *Taschenbuch der Mathematik*. 25. Aufl. Stuttgart : B. G. Teubner, 1991
15. Brossmann, C.: Virtual Reality im LRZ. URL *http://www.lrz-muenchen.de/services/peripherie/vr/* 2006
16. Brossmann, C.: Virtuelle Welten - ganz real. In: *Akademie Aktuell* (2006) 2, 28-33
17. BS ISO/IEC 14882:2003 (2nd): *The C++ Standard incorporating Technical Corrigendum No. 1*. 1. Aufl. West Sussex : Wiley, 2003
18. Bungert, C.: Raumöffner. In: *c't* (1999) 7, 172-178
19. Cachier, P. ; Mangin, J.-F. ; Pennec, X. ; Rivière, D. ; Papadopoulos-Orfanos, D. ; Régis, J. ; Ayache, N.: Multisubject non-rigid registration of brain MRI using intensity and geometric features. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (2001) , 734-742
20. Cachier, P. ; Pennec, X.: 3D non-rigid registration by gradient descent on a Gaussian-windowed similarity measure using convolutions. In: *Mathematical Methods in Biomedical Image Analysis* (2000) , 182-189
21. Chabrerie, A. ; Ozlen, F. ; Nakajima, S. ; Leventon, M. ; Atsumi, H. ; Grimson, E. ; Keeve, E. ; Helmers, S. ; Riviello, J. Jr. ; Holmes, G. ; Duffy, F. ; Jolesz, F.: Three-dimensional reconstruction and surgical navigation in pediatric epilepsy surgery. In: *Journal of Pediatric Neurosurgrey* 27 (1998) , 304-310
22. Chernyaev, E. V.: Marching cubes 33: Construction of topologically correct isosurfaces. In: *Technical Report CERN* (1995) 17,
23. CR/LF: Carriage Return / Line Feed. URL *http://www.crlf.de/Dokumente/Dxf-R13/DXF13.HTM*
24. CrystalEyes SDK: A Stereographics produkt. URL *http://www.stereographics.com/products/crystaleyes/dl_body_crystaleyes_sdks/ce_sdk.pdf* 1997
25. Dev-C++: Bloodshed Software. URL *http://www.bloodshed.net*
26. DICOM: Digital Imaging and Communications in Medicine. URL *http://medical.nema.org/dicom/*
27. EADHB: The electronic atlas of the developing human brain. URL *http://www.ncl.ac.uk/ihg/EADHB/home.html*
28. Ellis, R. E. ; Toksvig-Larsen, S. ; Marcacci, M. ; Caramella, D. ; Fadda, M.: Use of biocompatoble fiductial marker in evaluation the accuracy of CT image registration. In: *Investigative Radiology* 31 (1996) 10, 658-667
29. Ellis, R. E. ; Tso, C. Y. ; Rudan, J. F. ; Harrison, M. M.: A surgical planning and guidance system for high tibial osteotomy. In: *Computer Aided Surgery* 4 (1999) 5, 264-274
30. Emap: Edinburgh mouse atlas project. URL *http://genex.hgu.mrc.ac.uk/* 2003
31. Fathi, M. ; Schönewald, S. ; Hoffmann, I.: Ein intelligentes System zur Segmentierung dreidimensionaler NMR-Tomogramme. In: *Spektrum der Wissenschaft Digest* (1999) 2, 102-104
32. Feng, W.: Standard triangulation language (STL) file format. URL *http://www.vr.clemson.edu/rp/rp_stlfile.htm* 1997
33. Fischer, O.: Rückrufaktion. In: *iX* (2001) 7, 161-163

34. FLTK: Fast Light Tool Kit. URL *http://www.fltk.org*
35. Gamma, E. ; Helm, R. ; Johnson, R. ; Vlissides, J.: *Design patterns: Elements of reusable object-oriented software*. 1 . Aufl. Boston : Addison-Wesley, 1995
36. Gardener, J.: Medical Physics UCL MGI group 3D ultrasound. URL *http://www.medphys.ucl.ac.uk/mgi/US-3D/vis-3dus.htm* 1997
37. gcc: GNU Compiler Collection. URL *http://gcc.gnu.org*
38. Gembris, D.: Streifenweise. In: *c't* (1994) 6, 204-227
39. GLT: OpenGL C++ Toolkit. URL *http://www.nigels.com/glt/*
40. Goldberg, David: What every computer scientist should know about floating-point arithmetic. In: *ACM Computing Surveys* 23 (1991) 1, 5-48
41. Grimson, E. ; Leventon, M. ; Ettinger, G. J. ; Chabrerie, A. ; Ozlen, F. ; Nakajima, S. ; Atsumi, H. ; Kikinis, R. ; McBlack, P.: Clinical experience with a high precision image-guided neurosurgery system. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (1998) , 63-73
42. Grimson, W. E. L. ; Ettinger, G. J. ; Kapur, T. ; Leventon, M. ; Wells III, W. M. ; Kikinis, R.: Utilizing segmented MRI data in image-guided surgery. In: *Int. Journal of Pattern Recognition & Artificial Intelligence* 11 (1997) 8, 1367-1397
43. Haque, M. ; Ohata, K. ; Takami, T. ; Soares, S. B. Jr. ; Aree, S. N. ; Hakuba, A. ; Hara, M.: Development of lumbosacral spina bifida: Three-dimensional computer graphic study of human embryos at carnegie stage twelve. In: *Pediatric Neurosurgery* (2001) 35, 247-252
44. Haseloff, J.: Old botanical techniques for new microscopes. In: *BioTechniques* 34 (2003) 6, 1174-1182
45. Hesser, J. ; Männer, R.: Realistische Reisen durch den menschlichen Körper sind jetzt möglich. In: *Spektrum der Wissenschaft Digest* (1999) 2, 104-106
46. Hill, F. S. Jr.: *Computer graphics using OpenGL*. 2. Aufl. London : Prentice Hall, 2001
47. Höhne, K. H.: Medical image computing at the Institute of Mathematics and Computer Science in Medicine. In: *IEEE Trans. Med. Imaging* 21 (2002) 7, 713-723
48. Höhne, K. H.: Phantastische Reisen durch den menschlichen Körper. In: *Spektrum der Wissenschaft Digest* (1999) 2, 80-87
49. Höhne, K. H. ; Petersik, A. ; Pflesser, B. ; Pommert, A. ; Priesmeyer, K. ; Riemer, M. ; Schiemann, T. ; Schubert, R. ; Tiede, U. ; Frederking, H. ; Gehrmann, S. ; Noster, S.: *Voxel-Man 3D navigator: Brain and skull (3 CDs)*. 2. Aufl. Heidelberg : Springer, 2001
50. Höhne, K. H. ; Pflesser, B. ; Pommert, A. ; Priesmeyer, K. ; Riemer, M. ; Schiemann, T. ; Schubert, R. ; Tiede, U. ; Frederking, H. ; Gehrmann, S. ; Noster, S. ; Schumacher, U: *Voxel-Man 3D navigator: Inner organs (DVD)*. 2. Aufl. Heidelberg : Springer, 2003
51. Höhne, K. H. ; Pflesser, B. ; Pommert, A. ; Riemer, M. ; Schiemann, T. ; Schubert, R. ; Tiede, U.: A new representation of knowledge concerning human anatomy and function. In: *Yearbook of medical informatics 1996: Integration of information for patient care* (1996) , 525-529
52. Hopwood, N.: *Embryos in wax. Models from the Ziegler studio*. 1. Aufl. Cambridge : University of Cambridge, 2002
53. Ibanez, L. ; Schroeder, W. ; Ng, L. ; Cates, J.: The ITK software guide. URL *http://www.itk.org/ItkSoftwareGuide.pdf* 2003
54. IEEE 754-1985: IEEE Standard 754-1985 for Binary Floating-point Arithmetic. In: *SIGPLAN* 22 (1987) 2, 9 - 25
55. IMDM: Universitätsklinikum Hamburg-Eppendorf: Institut für Mathematik und Datenverarbeitung in der Medizin. URL *http://www.uke.uni-hamburg.de/zentren/experimentelle_medizin/informatik/index.en.html*
56. Indeed 3D: ZIB, Mercury Computer Systems. URL *http://www.indeed3d.com*
57. Intel: *Pentium Processor Family Developer' Manual Volume 3: Architecture and Programming Manual*. 1. Aufl. Mt. Prospect : McGraw-Hill, 1995
58. ISO C++-Draft Standard: X3J16/96-0225. URL *http://www.open-std.org/jtc1/sc22/open/n2356/* 1996
59. ITK: Insight Toolkit. URL *http://www.itk.org*
60. Jendrysiak, U.: Segmentierung von Schnittbildern. In: *Spektrum der Wissenschaft Digest* (1999) 2, 94-98
61. Kak, A. C. ; Slaney, M.: *Principles of computerized tomographic imaging*. 1. Aufl. New York : IEEE Press, 1988
62. Kernighan, B. W. ; Ritchie, D. M.: *Programmieren in C*. 2. Aufl. München : Hanser, 1990
63. Kerwin, J. ; Scott, M. ; Sharpe, J. ; Puelles, L. ; Robson, S. C. ; Martinez-de-la-Torre, M. ; Ferran, J. L. ; Feng, G. ; Baldock, R. ; Strachan, T. ; Davidson, D.; Lindsay,S.: 3 dimensional modelling of early human brain development using optical projection tomography. In: *BMC Neuroscience* 53 (2004) 1, 27-37

64. Kretschmann, H. J. ; Tafesse, U. ; Herrmann, A.: Different volume changes of cerebral cortex and white matter during histological preparation. In: *Microscopica Acta* 86 (1982) 1, 13-24
65. Ladic, L.: The process of making 3-D reconstructions from LSCM data. URL *http://www.cs.ubc.ca/spider/ladic/overview.html* 1995
66. Langer, A. ; Kreft, K.: *Standard C++ IOStreams and Locales*. 3. Aufl. Boston : Addison-Wesley, 2003
67. Leventon, M. ; Wells III, W. M. ; Grimson, W. E. L.: Multiple view 2D-3D mutual information registration. In: *Image Understanding Workshop* (1997) , 625-630
68. Levoy, M.: Display of surfaces from volume data. In: *IEEE Computer Graphics and Applications* 8 (1988) 3, 29-37
69. Levoy, M. ; Fuchs, H. ; Pizer, S. M. ; Rosenman, J. ; Chaney, E. L. ;Sherouse, G. W. ; Interrante, V. ; Kiel, J.: Volume rendering in radiation treatment planning. In: *Proc. 1st Converence on Visualization inBiomedical Computing* (1990) , 4-10
70. Lewiner, T. ; Lopes, H. ; Vieira, A. W. ; Tavares, G.: Efficient implementation of marching cubes' cases with topologicalguarantees. In: *Journal of Graphics Tools* (2003) 8, 1-15
71. libQGLViewer: Qt OpenGL Toolkit. URL *http://www-imagis.imag.fr/Membres/Gilles.Debunne/CODE/QGLViewer/*
72. LibSigC++: Callback-Framework. URL *http://libsigc.sourceforge.net/*
73. Lipton, L.: Stereographics developers' handbook. URL *http://www.stereographics.com/support/downloads_support/handbook.pdf* 1997
74. Lorensen, W. E. ; Cline, H. E.: A high resolution 3D surface construction algorithm. In: *Computer Graphics* 21 (1987) 4, 163-169
75. Loviscach, J.: Pixelwerker. In: *c't* (2005) 20, 200-211
76. Maintz, J. B. A. ; van den Elsen, P. A. ; Viergever, M. A.: 3D multimodality medical image registration using morphological tools. In: *Image and vision computing* (2001) 19, 53-62
77. Maintz, J. B. A. ; van den Elsen, P. A. ; Viergever, M. A.: Registration of 3D medical images using simple morphological tools. In: *Lecture notes in computer science IMPI 97* 1230 (1997) , 204-217
78. Maintz, J. B. A. ; van den Elsen, P. A. ; Viergever, M. A.: Registration of SPECT and MR brain images using a fuzzy surface. In: *Medical Imaging 96 SPIE* 2710 (1996) , 821-829
79. Maintz, J. B. A. ; van den Elsen, P. A. ; Viergever, M. A.: Using geometrical features to match CT and MR brain images. In: *Medical imaging, analysis of multimodality 2D/3D images* 19 (1994) , 1-11
80. Maintz, J. B. A. ; Viergever, M. A.: A survey of medical image registration. In: *Medical Image Analysis* 2 (1998) 1, 1-36
81. Maintz, J. B. A. ; Viergever, M. A.: An overview of medical image registration methods. In: *Symposium of Belgian hospital physicists association* (1996) 12, 1-22
82. Mattes, D. ; Haynor, D. R. ; Vesselle, H. ; Lewellen, T. K. ; Eubank, W.: Non-rigid multimodality image registration. In: *Medical Imaging 2001: Image Processing* (2001) , 1609 - 1620
83. Mayer, A. ; Meinzer H.-P.: High performance medical image processing in client/server-environments. In: *Computer Methods and Programs in Biomedicine* 58 (1999) , 207-217
84. Mega, M. S. ; Chen, S. S. ; Thompson, P. M. ; Woods, R. P. ; Karaca, T. J. ; Tiwari, A. ; Vinters, H. ; Small, G. ; Toga, A.: Mapping histology to metabolism: Coregistration of stained whole-brain sections to premortem PET in Alzheimer's Disease. In: *Neuroimage* 5 (1997) , 147-153
85. Meijering, E.: Spline interpolation in medical imaging: Comparison with other convolution-based approaches. In: *Signal Processing X: Theories and Applications-Pro* 4 (2000) , 1989-1996
86. Meinzer, H.-P. ; Meetz, K. ; Scheppelmann, D. ; Engelmann, U. ; Baur, H. J.: The Heidelberg ray tracing model. In: *IEEE Computer Graphics and Applications* 11 (1991) , 34-43
87. Meyers, S.: *Effektiv C++ programmieren*. 3. Aufl. München : Addison-Wesley, 2006
88. MFC: Microsoft Foundation Classes. URL *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcmfc98/html/mfchm.asp*
89. MinGW: Minimalist GNU for Windows. URL *http://www.mingw.org*
90. Moisan, L.: Modeling and image processing. URL *www.cmla.ens-cachan.fr/Utilisateurs/moisan/echant.pdf* 2003
91. MTL: The Matrix Template Library. URL *http://www.osl.iu.edu/research/mtl*
92. NLM: National Library of Medicine. URL *http://www.nlm.nih.gov*
93. Oberzig, K.: Ein neues Lächeln aus dem Computer. URL *http://www.scienzz.com/magazin/art165.html* 2004
94. OpenGL: Open Graphics Library. URL *http://www.opengl.org*

95. OsiriX: DICOM PACS workstation. URL *http://homepage.mac.com/rossetantoine/osirix/*
96. Ourselin, S. ; Bardinet, E. ; Dormont, D. ; Malandain, G. ; Roche, A. ; Ayache, N. ; Tandé, D. ; Parain, K. ; Yelnik, J.: Fusion of histological sections and MR images: Towards the construction of an atlas of the human basal ganglia. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (2001) , 743-751
97. Ourselin, S. ; Roche, A. ; Prima, S. ; Ayache, N.: Block Matching: A general framework to improve robustness of rigid registration of Medical Images. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (2000) , 557-566
98. Ourselin, S. ; Roche, A. ; Subsol, G. ; Pennec, X. ; Ayache, N.: Reconstructing a 3D structure from serial histological sections. In: *Image and Vision Computing* 19 (2001) 1, 25-31
99. Peters, T. M. ; Davey, B. L. K. ; Munger, P. ; Comeau, R. M. ; Evans, A. ; Olivier, A.: Three-dimensional multi-modal image-guidance for neurosurgery. In: *IEEE Transactions on medical imaging* 15 (1996) 2, 121-128
100. Petersik, A. ; Pflesser, B. ; Tiede, U. ; Höhne, K. H. ; Leuwer, R.: Realistic haptic interaction in volume sculpting for surgery simulation. In: *Surgery simulation and soft tissue modeling, Proc. IS4TM 2003* (2003) , 194-202
101. Petersik, A. ; Pflesser, B. ; Tiede, U. ; Höhne, K. H. ; Leuwer, R.: Realistic haptic volume interaction for petrous bone surgery simulation. In: *Computer assisted radiology and surgery, Proc. CARS 2002* (2002) , 252-257
102. Pflesser, B. ; Petersik, A. ; Tiede, U. ; Höhne, K. H. ; Leuwer, R.: Volume cutting for virtual petrous bone surgery. In: *Comput. Aided Surg.* 7 (2002) 2, 74-83
103. Pflesser, B. ; Tiede, U. ; Höhne, K. H. ; Leuwer, R.: Volume based planning and rehearsal of surgical interventions. In: *Computer assisted radiology and surgery, Proc. CARS 2000* (2000) , 607-612
104. PixelMed: PixelMed Java DICOM Toolkit. URL *http://www.pixelmed.com*
105. Pluim, J. P. W. ; Maintz, J. B. A. ; Viergever, M. A.: Mutual-information-based registration of medical images: a survey. In: *IEEE Transactions on medical imaging* 22 (2003) , 986-1004
106. Pommert, A. ; Höhne, K. H. ; Pflesser, B. ; Richter, E. ; Riemer, M. ; Schiemann, T. ; Schubert, R. ; Schumacher, U. ; Tiede, U.: Creating a high-resolution spatial/symbolic model of the inner organs based on the Visible Human. In: *Yearbook of Medical Informatics 2003: Quality of Health care: The role of informatics* (2003) , 530-537
107. Pommert, A. ; Höhne, K. H. ; Pflesser, B. ; Richter, E. ; Riemer, M. ; Schiemann, T. ; Schubert, R. ; Schumacher, U. ; Tiede, U.: Ein realistisches dreidimensionales Modell der Inneren Organe auf der Basis des Visible Human. In: *Bildverarbeitung für die Medizin 2001: Algorithmen - Systeme - Anwendungen, Informatik aktuell* (2001) , 72-76
108. Pooma: Parallel object-oriented methods and applications. URL *http://acts.nersc.gov/POOMA.html*
109. QT: Trolltech QT. URL *http://www.trolltech.com*
110. Radlanski, R. J. ; Renz, H. ; Klarkowski, M. C.: Prenatal development of the human mandible. In: *Anat Embryol* 207 (2003) 9, 221-232
111. Razdan, A. ; Patel, K. ; Farin, G. E. ; Capco, D. G.: Volume visualization of multicolor Laser Confocal Microscope data. In: *Computers & Graphics* 25 (2001) 3, 371-382
112. Rückert, J.: Ray-Tracing. In: *Informatik-Spektrum* (1988) 11, 40-42
113. Saad, A. ; Hevler, A.: Das plastische Panoptikum. In: *c't* (1999) 7, 164-171
114. Sadler, T. W. ; Langman, J.: *Medizinische Embryologie*. 9. Aufl. Stuttgart : Thieme, 1998
115. Sakas, G.: Dreidimensionale Bildrekonstruktion aus Ultraschall-Daten. In: *Spektrum der Wissenschaft Digest* (1999) 2, 88-93
116. Sakas, G. ; Schreyer, L.-A. ; Grimm, M.: Preprocessing and volume rendering of 3D ultrasonic data. In: *IEEE Computer Graphics and Applications* 15 (1995) 4, 47-54
117. Sakas, G. ; Vicker, M. G. ; Plath, P. J.: Case study: Visualization of laser confocal microscopy datasets. In: *Proc. IEEE Visualization '96* (1996) , 375-380
118. Sakas, G. ; Walter, S.: Extracting surfaces from fuzzy 3D-ultrasound data. In: *ACM Computer Proc. SIGGRAPH-95* (1996) , 465-474
119. Schiemann, T. ; Tiede, U. ; Höhne, K. H.: Segmentation of the Visible Human for high quality volume based visualization. In: *Med. Image Analysis* 1 (1997) 4, 263-271
120. Schönherr, M.: Welten-Bau. In: *c't* (2004) 2, 196-201
121. Schormann, T. ; Dabringhaus, A. ; Zilles, K.: Statistics of deformations in histology and application to improved alignment with MRI. In: *IEEE Transactions on Medical Imaging* 14 (1995) 1, 25-35
122. Schormann, T. ; von Matthey, M. ; Dabringhaus, A. ; Zilles, K.: Alignment of 3-D brain data sets originating from MR and histology. In: *Bioimaging* (1993) 1, 119-128
123. Schroeder, W. J. ; Martin, K. M. ; Lorensen, W. E.: The design and implementation of an object-oriented toolkit for 3D graphics and visualization. URL *http://www.vtk.org/pdf/dioot.pdf* 1996

124. Sharpe, J.: OPT microscopy at the MRC Human Genetics Unit, Edinburgh UK. URL *http://genex.hgu.mrc.ac.uk/OPT_Microscopy*

125. Sharpe, J. ; Ahlgren, U. ; Perry, P. ; Hill, B. ; Ross, A. ; Hecksher-Sorensen, J. ; Baldock, R. ; Davidson, D.: Optical projection tomography as a tool for 3D microscopy and gene expression studies. In: *Science* 296 (2002) 5567, 541-545

126. Shirley, P. ; Ashikhmin, M. ; Gleicher, M. ; Marschner, S. R. ; Reinhard, E. Sung, K. ; Thompson, W. B. ; Willemsen, P.: *Fundamentals of computer graphics*. 2. Aufl. Wellesley : A K Peters, 2005

127. Simon, D. ; O'Toole, R. V. ; Blackwell, M. ; Morgan, F. ; DiGioia, A. M. ; Kanade, T.: Accuracy validation in image-guided orthopaedic surgery. In: *Proc. of the 2. Int. Symposium on medical robotics and computer assisted surgery* (1995) , 185-192

128. Smith, B. R.: The multi-dimensional human embryo. URL *http://embryo.soad.umich.edu/* 2003

129. Smith, B. R. ; Linney, E. ; Huff, D. S. ; Johnson, G. A.: Magnetic resonance microscopy of embryos. In: *Computerized Medical Imaging and Graphics* 20 (1996) 6, 483 - 490

130. Smith, S. W.: *The scientist and engineer's guide to digital signal processing*. 2. Aufl. San Diego : Californial Technical Publishing, 1999

131. Streicher, J. ; Donat M. A. ; Strauss, B. ; Sporle, R. ; Schughart, K. ; Müller, G. B.: Computer-based three-dimensional visualization of developmental gene expression. In: *Nat Genet* 25 (2000) 2, 147-152

132. Thirion: Image matching as a diffusion process: an analogy with Maxwell's demons. In: *Medical Image Analysis* 2 (1998) 3, 243-260

133. Tiede, U. ; Schiemann, T. ; Höhne, K. H.: High quality rendering of attributed volume data. In: *Proc. IEEE Visualisation* (1998) , 255-262

134. Tsai, A. ; Fisher, J. ; Wible, C. ; Wells III, W. M. ; Kim, J. ; Willsky, A.: Analysis of functional MRI data using Mutual Information. In: *Medical Image Computing and Computer Assisted Intervention* (1999) 2,

135. van den Elsen, P. A. ; Pol, E. J. D. ; Viergever, M. A.: Medical image matching - a review with classification. In: *IEEE Engineering in medicine and biology* 12 (1993) 1, 26-39

136. Vandevoorde, D. ; Josuttis, N. M.: *C++ Templates: The complete guide*. 1. Aufl. Boston : Addison-Wesley, 2003

137. VHP: Visible Human Project. URL *http://www.nlm.nih.gov/research/visible/visible_human.html*

138. Vincent, L. ; Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. In: *IEEE Trans on Pattern analysis and machine intelligence* 13 (1991) 6, 583-598

139. Viola, P. ; Wells III, W. M.: Alignment by maximization of mutual information. In: *International Journal of Computer Vision* 24 (1997) 2, 137-154

140. VisTutor: Universität Tübingen. URL *www.gris.uni-tuebingen.de/projects/vis/coursebook/* 2003

141. VolumePro 500: Mitsubishi Electronics real time visualization. URL *www.mitsubishielectric.com/news/1999/113099.html* 1999

142. VTK: Visualization Toolkit. URL *http://www.vtk.org*

143. Web3D: Web3D Consortium. URL *http://www.web3d.org*

144. Weeks, E.: How does a confocal microscope work?. URL *http://www.physics.emory.edu/~weeks/confocal/*

145. Wegner, S. ; Oswald, H. ; Wust, P. ; Fleck, E.: Segmentierung mit der Wasserscheidentransformation. In: *Spektrum der Wissenschaft Digest* (1999) 2, 99-101

146. Wells III, W. M. ; Viola, P. ; Atsumi, H. ; Nakajima, S. ; Kikinis, R.: Multi-modal volume registration by maximization of mutual information. In: *Medical Image Analysis* (1996) 1, 35-52

147. Wessling, J. ; Domagk, D. ; Domschke, W. ; Heindel, W.: Virtuelle CT-Kolonographie - Alternative zur konventionellen Koloskopie?. In: *Deutsches Ärzteblatt* 102 (2005) 43, 2337-2342

148. Wilmer, F. ; Tiede, U. ; Höhne, K. H.: Reduktion der Oberflächenbeschreibung triangulierter Oberflächen durch Anpassung an die Objektform. In: *Mustererkennung Proc. DAGM-Symposium* 14 (1992) , 430-436

149. Woo, M ; Neider, J. ; Davis, T. ; Shreiner, D.: *OpenGL Programming Guide*. 3. Aufl. Massachusetts : Addison-Wesley, 1999

150. XnView: Bildverarbeitung. URL *http://xnview.de*

151. Xu, W. ; Jericho, M. H. ; Meinertzhagen, I. A. ; Kreuzer, H. J.: Digital in-line holography for biological applications. In: *Proc. of the National Academy of Sciences of the USA (PNAS)* 98 (2001) 20, 11301-11305

152. Zhao, H. ; Osher, S.: Visualisation, analysis and shape reconstruction of unorganized data sets. In: *Geometric level set methods in imaging vision and graphics* (2002) , 1-25

153. Zhao, H. ; Osher, S. ; Fedkiw, R.: Fast surface reconstruction using the level set method. In: *Proc IEEE Variational and level set methods in computer vision* (2001) , 8p

Danksagung

Ich danke dem Direktor der Anatomischen Anstalt der Universität München, Herrn Prof. Lange, für das Bereitstellen des Arbeitsplatzes. Ich danke Herrn Prof. Heinzeller (Anatomische Anstalt der Universität München, Lehrstuhl III) für seine intensive Betreuung. Ich danke Frau Beate Aschauer (Anatomische Anstalt der Universität München, Lehrstuhl III, Arbeitsgruppe Heinzeller) für ihre unermüdliche Hilfsbereitschaft. Ich danke meinem Kollegen Herrn Christian Brossmann (Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften) für die Beratung und die Bereitstellung spezieller Hardware. Ich danke Herrn Prof. Hölzel und Herrn Dr. Müller (Institut für Medizinische Informationsverarbeitung, Biometrie und Epidemiologie) für die richtungweisenden Anregungen. Ich danke Marlis Friedl für die Korrektur. Ich danke meinen Eltern für ihre Unterstützung.

Zum Autor

Peter Weinert.

Hochschulreife auf dem Gymnasium Unterhaching.
Studium der Humanmedizin an der Ludwig-Maximilians-Universität München.
Doktorand an der Anatomischen Anstalt der Universität München.
Approbation als Arzt.

Forschung und Entwicklung auf den Gebieten
medizinische Visualisierung, 3-D-Rekonstruktion, Stereoskopie,
lineare Algebra, analytische Geometrie, dynamische Systeme,
Grafik-Programmierung, Software-Design, -Entwicklung und -Optimierung.