Institut für Informatik
der Technischen Universität München

Methoden zur Erhöhung des visuellen Realismus für die Simulation minimal-invasiver chirurgischer Eingriffe

Arne Radetzky

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Gudrun J. Klinker, Ph.D.
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Rüdiger Westermann
2.	Univ.-Prof. Dr. Bernhard Preim, Otto-von-Guericke-Universität Magdeburg

Die Dissertation wurde am 28.05.2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.10.2003 angenommen.

KURZFASSUNG

Ein Hauptziel für die Anwendung von Chirurgiesimulatoren ist die Erzeugung einer virtuellen Trainingsumgebung für angehende Chirurgen, mit der sie die verschiedenen Schritte chirurgischer Eingriffe am Computer ohne Risiko für den Patienten trainieren können.

Eine maßgebliche Voraussetzung für ein realistisches Training am Computer ist die simulierte Interaktion mit virtuellen medizinischen Instrumenten, wobei speziell die virtuelle Deformation und Entfernung von Gewebe von Bedeutung ist. Für diese Anwendung wurde ein Neuro-Fuzzy Modell entwickelt, das ein Feder-Masse System simuliert und die Beschreibung von visuellem Deformationsverhalten des simulierten Gewebes in der Form von Expertenwissen und medizinischer Terme erlaubt. Darüber hinaus können die für das zugrundeliegende Feder-Masse System benötigten physikalischen Parameter unter Benutzung von Zeitreihendaten realer Gewebedeformationen automatisch erlernt werden.

Eine weitere Voraussetzung für realistisches, chirurgisches Training ist die Qualität und Interaktivität der Visualisierung. Verschiedene Methoden für die interaktive Visualisierung und Manipulation dreidimensionaler, digitaler medizinischer Datensätze, differenziert nach den Themen Volume-Rendering und Surface-Rendering, werden beschrieben. Um den Realitätslevel weiter zu erhöhen, wurde eine Texturierungsmethode entwickelt, die das interaktive Platzieren hochauflösender Bilder anatomischer Details auf jede Position der Modelloberfläche ermöglicht.

Abschließend werden die Ergebnisse in Form einer Beschreibung und Evaluierung zweier Chirurgiesimulatoren präsentiert, die unter Benutzung der beschriebenen Deformations- und Visualisierungsmethoden entwickelt worden sind.

ABSTRACT

A main goal of surgical simulators is the creation of virtual training environments for prospective surgeons. Thus, they can rehearse the various steps of surgical procedures on a computer system without any risks for the patient.

One main condition for realistic training is the simulated interaction with virtual medical instruments. In particular, the virtual deformation and transection of tissues are important. For this application, a neuro-fuzzy model has been developed, which simulates a spring-mass system and allows the description of the visual deformation behavior of the simulated tissue by means of expert knowledge in form of medical terms. In addition, the physical parameters required for the underlying spring-mass system can be automatically learned by using time series data from real tissue deformations.

Another condition for realistic surgical training is the quality and interactivity of visualization. Various methods for the interactive visualization and manipulation of three-dimensional digital imaging datasets are described. These methods are mainly separated in the topics volume-rendering and surface-rendering. To further increase the level of realism, a texturing method has been developed, allowing interactive placement of high-resolution bitmaps of anatomical details to any desired position on the model's surface.

Finally, the results are presented in form of the description and evaluation of two surgical simulators, which were developed by using the described deformation and visualization methods.

VORWORT

Mit der vorliegenden Arbeit im Bereich der Chirurgiesimulation begann ich bereits Anfang 1997, wo ich am Institut für Medizinische Informatik der Technischen Universität Braunschweig in Zusammenarbeit mit der Universität Magdeburg ein Verfahren für die Deformationssimulation für den Einsatz in der Virtual Reality entwickelt habe, das auf der Basis von Neuro-Fuzzy Systemen arbeitet. Den Anstoß zu dieser Arbeit gab Herr Prof. Karl Heinz Höhne, der mir ein Buch zu der Konferenz „Visualization in Biomedical Computing“ [78] zur Verfügung stellte, das er als Editor bearbeitet hat. Mehrere Artikel in diesem Buch beschäftigten sich mit der Simulation von Gewebedeformationen in der virtuellen Chirurgie, empfahlen aber die Verwendung von finiten Elementen Modellen und beschrieben im Wesentlichen Lösungs- und Vereinfachungsverfahren für die anfallenden Differenzialgleichungssysteme. In diesem Zusammenhang entstand die damals noch vage Idee, in „irgendeiner Weise“ Neuronale Netze, die ja nichts anderes sind als universelle Approximatoren, für die Lösung der Differenzialgleichungssysteme zu verwenden. Aus den Neuronalen Netzen wurden dann Neuro-Fuzzy Systeme, um damit auch unscharfe Modellierungen zu berücksichtigen und das visuelle Ergebnis in den Vordergrund zu stellen – im Gegensatz zu einer physikalisch korrekten Modellierung. Ohne eine konkrete Vorstellung der Umsetzung wurde kurzfristig zusammen mit Andreas Nürnberger der Universität Magdeburg ein Abstract auf einer kleinen internationalen Konferenz eingereicht. Die Deadline zur Abgabe des vollständigen Papers sechs Wochen später erforderte somit ein „umfassendes Brainstorming“, um die versprochenen Ziele und Ergebnisse auch liefern zu können. Es waren intensive theoretische Betrachtungen und umfassende Programmiertätigkeiten erforderlich, aber das dabei entwickelte Verfahren erwies sich in dieser Anwendung als einzigartig.

Da das Verfahren für den Einsatz in der Chirurgiesimulation propagiert worden war, musste ein Simulator „drumherum“ entwickelt werden. Dieser Simulator diente anfänglich allein der Evaluation der Deformationsalgorithmen. Bereits in einem sehr frühen Stadium hat sich die Arbeit am Chirurgiesimulator verselbstständigt, nicht zuletzt, weil die Qualität und Geschwindigkeit der Deformationssimulation andere in dem Gebiet der Virtuellen Chirurgie angewendeten Algorithmen übertraf und somit der wissenschaftliche Erfolg schnell eintrat.

Mit meiner Mitwirkung an dem europäischen Forschungsprojekt ROBOSCOPE, das sich mit der robotergestützten minimal-invasiven Neurochirurgie beschäftigte und an dem 16 Institutionen und Firmen aus ganz Europa teilnahmen, entstand ein weiteres Simulationssystem für die Neurochirurgie, das im Gegensatz zu fast allen anderen Systemen reale Patientendatensätze, anstatt manuell erzeugte Modelle, verwenden kann.

Aus der reinen Transporthülle für die Deformationssimulation wurden somit eigenständige Forschungsprojekte, deren Gesamtumfang die Grenzen einer üblichen Dissertation von etwa fünf Personenjahren erheblich überschritten. Durchgeführt konnten die in

der vorliegenden Arbeit beschriebenen Projekte daher nur durch die Mitarbeit vieler Studenten, Diplomanden, medizinischer Promotionen und Projektmitarbeiter. Besonderer Dank gilt diesbezüglich Matthias Bruckner, Alexander Hartwig, Jan Hartwig, Helmut Kitzberger, Gerhard Kleinszig, Markus Krosche, Volker Leeb, Thomas Lison, Andreas Nürnberger, Ingo Paschke, Andreas Petersik, Maik Plischke, Michael Rudolph, Grisha Schneider, Florian Schröcker, Peter Sendtner, Stephen Starkie, Werner Stefanon, Mirko Streckenbach, Christian Wimmer und Klaus Hendrik Wolf. Im Speziellen Dr. Andreas Nürnberger, mit dem mich ein langjähriges intellektuelles und freundschaftliches Verhältnis verbindet, war stets bereit, sich für ein interessantes Problem zu begeistern und stundenlang Lösungsmöglichkeiten zu diskutieren.

Zu Dank verpflichtet bin ich auch Herrn Prof. Dr. Auer vom Institute of Applied Sciences in Medicine, Salzburg, Herrn Prof. Brian Davies vom Imperial College, London und Herrn Prof. Dr. Pretschner, die mir es wirtschaftlich ermöglicht haben, den Neurochirurgiesimulator ROBO-SIM zu entwickeln.

Ein ganz spezieller Dank gilt auch meinem Freund Herrn Dr. Friederich Peter Zeuner, der bestimmt und stetig auf eine Fertigstellung der Dissertation gedrängt und mich unermüdlich und uneingeschränkt bei der Beseitigung diverser, teils schwerwiegender Probleme unterstützt hat. Dies zu einer Zeit, in der ich den Sinn einer schriftlichen Ausarbeitung und der mit der Einreichung der Dissertation zusammenhängenden Unterwerfung unter organisatorische und bürokratische Prozesse nicht einsehen wollte.

Dass letztendlich die vorliegende Arbeit als Dissertation anerkannt worden ist, verdanke ich Herrn Prof. Dr. Rüdiger Westermann, der trotz meiner manchmal etwas zu direkten und undiplomatischen Vorgehensweise mir nicht „das Zeug vor die Füße geworfen hat“. Herr Prof. Dr. Bernhard Preim, der in kürzester Zeit ein Gutachten geschrieben hat, hat das Seine dazu beigetragen.

Schließlich darf ich natürlich nicht die Geduld meiner Frau vergessen, die trotz andauernder Phasen körperlicher und geistiger Abwesenheit meinerseits in den letzten Jahren nicht nur durchgehalten, sondern mich auch bedingungslos unterstützt hat.

Abschließend noch eine Warnung: Der Text ist konsequent unter Beachtung der neuen deutschen Rechtschreibung verfasst, sodass viele Schreibweisen stark gewöhnungsbedürftig sein dürften (z.B. Differenzial). An dieser Stelle ein Dank an meine Mutter, die sich die Mühe gemacht hat, die Einarbeitung der neuen Rechtschreibung vorzunehmen.

INHALTSVERZEICHNIS

1 EINLEITUNG

Die Entwicklungen in der Humanmedizin waren in den letzten Jahren unter anderem durch die Fortschritte der minimal-invasiven Chirurgie gekennzeichnet. Bei dieser Technik wird der Zugang zur eigentlichen Operationsstelle entweder durch natürliche Körperöffnungen oder durch sehr kleine, künstlich geschaffene vollzogen. Spezielle Operationswerkzeuge – so genannte Endoskope – werden über solche Zugänge an das eigentliche Operationsgebiet herangeführt. Endoskope ermöglichen einerseits über Glasfasern, Linsen- und Prismensysteme oder/und Minikameras den Blick in das Innere des Körpers, andererseits werden chirurgische Werkzeuge, wie Greifer, Schere oder Laser an der Spitze des Endoskops bereitgestellt oder können während des Eingriffes über einen Werkzeugkanal nachträglich eingeführt bzw. ausgetauscht werden.

Die minimal-invasive Chirurgie bietet für den Patienten und für das Gesundheitswesen als Ganzes erhebliche Vorteile. Eingriffe, die in konventioneller Art mehrtägige Krankenhausaufenthalte zur Folge hätten, können so teilweise ambulant durchgeführt werden – neben dem gesundheitlichen Aspekt im Zusammenhang der Patientenbelastung trägt dies auch zur massiven Kosteneinsparung in der Chirurgie bei.

Als Problem ist in diesem Zusammenhang zu erwähnen, dass chirurgische Eingriffe mit Hilfe von Endoskopen besondere Fertigkeiten und eine spezielle Ausbildung des Arztes verlangen. Zum einen sind die Operationswerkzeuge über Seilzüge oder auf elektronischem Weg fernzubedienen, zum anderen muss der Arzt das nach außen transportierte Bild, trotz etwaiger Verzerrungen (Fischauge[1]) und der nicht immer exakt bekannten Position und Ausrichtung der bildaufnehmenden Optik im Körper, richtig verstehen und deuten. Neben diesen mechanischen und geometrischen Problemen bei der Endoskopie unterliegt generell die medizinische Behandlung des menschlichen Körpers - und im speziellen auch die Chirurgie - stets besonderen rechtlichen und moralischen Zwängen, die zu einer außergewöhnlich hohen Verantwortung und Belastung des behandelnden Arztes führen. Fehldiagnosen und natürlich auch Fehler bei der Behandlung werden kaum toleriert und könnten zum Tod eines Patienten führen.

Daher ist die Ausbildung angehender Chirurgen sehr zeitaufwändig und erfordert viele Jahre Training. Eine Art des Trainings ist das risikoreiche Üben unter Aufsicht eines erfahrenen Kollegen direkt am Patienten.

Das Üben an Tieren, wie beispielsweise an Schweinen oder Kaninchen wird nur noch selten durchgeführt. Die hohen Kosten für Anästhesie und insbesondere die Wartung der chirurgischen Instrumente und der Tiere selbst sind für kleine und mittelgroße medizinische Einrichtungen kaum noch darstellbar [84]. Außerdem wird der Einsatz von Tieren

[1] *Um den einsehbaren Bereich durch das Endoskop zu erhöhen, werden in vielen Fällen extreme Weitwinkelobjektive eingesetzt.*

für die experimentelle Chirurgie in vielen Ländern kontrovers diskutiert oder ist sogar ganz verboten.

In einigen medizinischen Einrichtungen wird das chirurgische Training an Kadavern durchgeführt. Neben den hohen Kosten und der geringen Verfügbarkeit der Kadaver ist das Training allerdings nicht ausreichend realistisch [41].

Eine letzte Möglichkeit ist das Training an so genannten Pelvitrainern, welche manchmal auch unter dem Namen „Simulatoren" angeboten werden. Hierbei handelt es sich um eine Nachbildung des Operationsgebietes durch Kunststofforgane, an denen mit herkömmlichen Instrumenten manipuliert werden kann. Neben dem Nachteil, dass es sich nur um modellhafte Anatomie handelt, ist der geringe optische Realismus zu nennen. Mit *Realismus* ist im Rahmen dieser Arbeit die Wirklichkeitsnähe gemeint.

Wegen dieser Nachteile der herkömmlichen Trainingsmethoden sind bereits seit mehreren Jahren Chirurgiesimulatoren zumindest im Bereich der Wissenschaft auf dem Vormarsch. Die virtuelle Realität kann in diesen kritischen Bereichen hervorragende Unterstützung für die behandelnden Ärzte leisten. Durch die enormen Fortschritte in der Computertechnologie wurde es in den letzten Jahren möglich, mit vertretbarem Aufwand virtuelle Operationsumgebungen zu schaffen. Als eine mögliche Anwendung hat sich die Virtuelle Endoskopie und Chirurgie entwickelt, deren Zielsetzung wie folgt beschrieben werden kann:

- **Ausbildung und Training**
  Bei der Ausbildung von Ärzten sieht man sich mit dem besonderen Dilemma konfrontiert, dass es in vielen Bereichen aus verschiedensten Gründen nicht möglich ist, am eigentlichen Objekt - dem Menschen - Demonstrationen und Übungen durchzuführen. Neben der Verwendung von totem Gewebe bietet sich hier im Besonderen die Anwendung der Virtuellen Chirurgie in der Ausbildung an [138].
- **Vorbereitung von Eingriffen**
  Bei jedem Eingriff gibt es verschiedene Varianten der Durchführung, und selbst erfahrene Ärzte benötigen entsprechende Informationen, um den richtigen Weg ermitteln zu können. Jeder chirurgische Eingriff stellt für den Patienten eine große Belastung dar. Aus diesem Grund ist es wünschenswert, dass dieser möglichst kurz und optimal verläuft. Virtuelle Chirurgie kann hier in der Vorbereitung eine wesentliche Hilfe bieten [12; 89].
- **Evaluierung**
  Im Sinne einer Qualitätssicherung bei chirurgischen Eingriffen ist es prinzipiell notwendig, die Güte von Operationen entsprechend werten zu können. In der Virtuellen Chirurgie kann der Verlauf des simulierten Eingriffes überwacht und mit den Daten von optimalen Verfahren verglichen werden [170].
- **Roboterunterstützte Planung und Eingriffe**
  Selbst Ärzte mit langjähriger chirurgischer Praxiserfahrung können nie die Präzision in der Chirurgie erreichen, zu der Roboter im Stande wären. Bei einer Planung und Vorbereitung mit tatsächlichen Patientendaten ist es nur ein logischer Folgeschritt,

dass vorerst optimierte, simulierte Eingriffe später roboterunterstützt am Patienten real durchgeführt werden [8; 9].

1.1 Aktueller Stand der Chirurgiesimulation

Das Ziel der meisten Chirurgiesimulatoren ist ein möglichst realistisches Operationsszenario zu vermitteln. Hochqualitative Computerrekonstruktionen der menschlichen Anatomie wurden bereits verwirklicht und stellen exzellente Tools für die anatomische Visualisierung dar [79; 182]. Wenn jedoch Echtzeitinteraktionen möglich sein sollen, dann kann die Geschwindigkeit des verwendeten Computers ein beträchtlich limitierender Faktor sein, insbesondere, wenn virtuelle Gewebsdeformationen berechnet werden.

Gerade das ist auch der Grund, warum Chirurgiesimulatoren noch nicht zur täglichen ärztlichen Routine gehören, etwa vergleichbar mit den Flugsimulatoren für Piloten, bei denen eine rein statische Simulation ausreichend ist. Im Folgenden soll ein kurzer Überblick über einige der existierenden kommerziell erhältlichen Chirurgiesimulatoren gegeben werden, da diese die für einen chirurgischen Eingriff notwendigen Fertigkeiten vollständig vermitteln oder sogar den Eingriff selbst komplett simulieren. Reine Forschungssysteme, die nur dazu dienen, einen Teilaspekt eines Eingriffs zu simulieren, beispielsweise zur Demonstration neu entwickelter Algorithmen, bleiben dabei unberücksichtigt.

1.1.1 Chirurgische Planung und Navigation

Die kommerzielle Entwicklung chirurgischer Planungs- und Navigationssysteme ist bereits weit vorangeschritten. Laut der Millennium Research Group, die 2002 eine Marktanalyse durchführte, betrug im Jahr 2001 der Gesamtmarkt für chirurgische Navigation allein in den USA 160 Mio. US$ und jährlich wird eine Steigerung von 20% erwartet. Aufgrund dieses umfangreichen Marktes und der Ähnlichkeit der Basistechnologien der chirurgischen Navigationssysteme und der Chirurgiesimulatoren in Bezug auf die Visualisierung soll im Folgenden kurz auf diese Systeme eingegangen werden, auch wenn deren Einsatz nicht für das chirurgische Training, sondern operationsvorbereitend und während der Operation erfolgt. Chirurgische Navigation soll aber auch deswegen hier genannt werden, um eine klare Abgrenzung zu den Chirurgiesimulationen zu definieren.

Ziel der chirurgischen Navigation ist die Übertragung von Planungsdaten auf das Operationsgebiet eines aktuellen Patienten, um die Genauigkeit operativer Zugangswege und anderer Manipulationen zu erhöhen und damit mehr Sicherheit zu gewährleisten.

Die erhöhte Sicherheit besteht dann z.B. darin, dass anatomische Strukturen in der Umgebung des Operationsfeldes wie Blutgefäße und Nerven sowie andere Organe sicher geschont werden können, obwohl der Operateur sie nicht direkt sehen kann, da sie abseits seines Zugangsweges in der Tiefe des Gewebes liegen, also bereits im Moment ihrer Darstellung beschädigt oder zerstört sein können. Solche irrtümlichen Abweichungen vom beabsichtigten operativen Zugangsweg können zu schwerwiegenden und sogar tödlichen Komplikationen führen.

Erhöhte Sicherheit kann auch darin bestehen, dass die Vorausplanung von Zugangswegen und direkte Übertragung dieser Planungsdaten auf den Körper des Patienten die Positionierungspräzision von Implantaten erhöht – diese Vorgänge werden seit einigen Jahren zunehmend direkt an chirurgische Roboter übertragen z.B. System ROBODOC von Integrated Surgical Systems, CASPAR von Ortho-Maquet, beides Systeme für die Orthopädie; oder das Herzchirurgie-System von Intuitive Surgical sowie das Stereotaktische System NEUROMATE von Integrated Surgical Systems.

Zur Übertragung der Raumkoordinaten kommen derzeit entweder magnetische (I-SOTRAK, FASTRACK von Polhemus, Flock Of Birds von Ascension), Inertialtracker (drei Freiheitsgrade), passive optische Systeme mit Videokamera, die die Position per Mustererkennung bestimmen (VISLAN, Roke Manor Research oder optische Trackingsysteme unter Verwendung von LED´s (light emitting diodes) (z.B Optotrack, Solaris von Northern Digital) zum Einsatz.

Die Neuronavigation, also die exakte Planung von Eingriffen am Gehirn anhand von präoperativ gewonnener digitaler Bilddaten (CT, MRT), hat sich dank der technischen Neuerungen der letzten 15 Jahre aus dem Gebiet der stereotaktischen Neurochirurgie entwickelt:

Die Aufgabe ist eine exakte Definition eines Zielpunktes in der Tiefe des Gehirns durch Berechnung von Koordinaten an Bildern von Schädel und Gehirn, sowie der Übertragung dieser Koordinaten auf die reale Welt, also auf die Kopfoberfläche und das Gehirn eines Patienten. Diese sog. Registrierung erfolgt derzeit durch die Fixation von Landmarken, sog. fiducial markers, auf der Kopfoberfläche und zwar vor Durchführung einer Untersuchung mit bildgebenden Verfahren wie CT oder MRT. Diese fiducial markers sind auf den Bildern sichtbar. Daraufhin werden zu Beginn der Operation mit einem stiftförmigen Markierungsgerät die fiducial markers angefahren und damit die Raumkoordinaten zwischen Bilddaten und Körper abgestimmt.

Derzeit verwendete Systeme dienen entweder nur zur Übertragung von Planungsdaten auf das Operationsfeld oder zur Steuerung von Operationsinstrumenten im Operationsfeld, schließlich auch zur Führung des in der Neurochirurgie meist verwendeten Operationsmikroskops mit Hilfe eines Manipulatorarmes (z.B. Elekta, SurgiScope, Viewing Wand, Radionics, Optical Tracking System, Easiguide, Philips, VectorVision², BrainLAB)

Nachteil dieses Stands der Technik: Die Führung im Operationsfeld stützt sich ausschließlich auf die präoperativen Bilddaten. Somit kommen während des Eingriffes, nach erster Verlagerung von Organen, erhebliche Fehler in Betracht, welche die Weiterverwendung der Systeme während der Operation erschwert.

1.1.2 KISMET

Das Simulationspaket KISMET ist aus Arbeiten zur Überwachung und Steuerung von Fernhantierungsvorgängen in der Kerntechnik hervorgegangen [99]. Später wurde das System um die Simulation von flexiblen Objekten [97] und um ein Modul zum Volume-Rendering erweitert [98]. Die flexiblen Objekte werden verwendet, um eine Trainingsumgebung für minimal-invasive Chirurgie im Abdominalbereich zu realisieren. Die Mo-

delle der Organe werden als Oberflächenmodelle mittels nodaler Netze abgebildet, welche das physikalische Verhalten simulieren [99]. Die Modelle der Objekte sind aufgrund der Echtzeitanforderungen in ihrer Geometrie relativ einfach gehalten. Der Simulator enthält eine Vielzahl von chirurgischen Werkzeugen, u.a. es ist möglich, Objekte zu deformieren, zu schneiden und Klemmen anzubringen. Die Simulation der Anatomie beschränkte sich anfangs auf einfachste Modelle der Gallenblase und des Gallenganges.

Seit dem Jahr 2002 wird der Chirurgiesimulator von der Select-IT VEST System AG kommerziell angeboten. Mittlerweile hat sich auch die Qualität der Simulation verbessert. Durch die Einführung neuer Texturen und erweiterter Modelle ist der Realitätseindruck stark gestiegen, auch Blutungen werden dargestellt [37; 199]. Das System richtet sich durch die Einfachheit der simulierten Eingriffe und den integrierten Prozedurentrainer jedoch primär an den Anfänger, der noch nie in der Realität operiert hat.

1.1.3 Mentice

Die Firma Mentice AB aus Schweden stellt seit 1999 chirurgische Prozedurentrainer und Chirurgiesimulatoren her. Aufbauend auf dem System Procedicus MIST, einem einfachen Prozedurentrainer, der das Handling der Instrumente in einer virtuellen Welt, aufgebaut aus einfachen geometrischen Formen, trainiert, werden laparoskopische und arthroskopische Handlungstrainer vertrieben. Grundgedanke bei den Simulatoren ist nicht die vollständige Nachbildung eines chirurgischen Eingriffes, sondern vielmehr die Vermittlung der mechanischen Kenntnisse, um einen solchen Eingriff durchzuführen. Neben grundlegendem Training, beispielsweise dem Einfügen von Stiften in Löcher, wird trainiert, ein Gefäß mit einer Zange zu fassen und mit einem zweiten Instrument eine Klammer zu setzen. Die Schnelligkeit und Zielstrebigkeit der Vorgehensweise wird über eine Bahnverfolgung der Instrumentbewegungen evaluiert. Außerdem registriert das System, wie oft der Anwender andere Bereiche der Anatomie verletzt.

Aus ärztlicher Sicht sind solche Trainingssysteme ideal für die ersten Schritte in der minimal-invasiven chirurgischen Ausbildung. Komplette Eingriffe können damit allerdings nicht abgebildet werden, da die modellierte Anatomie sehr stark vereinfacht ist.

1.1.4 Immersion Medical

Immersion Medical aus den USA vertreibt die Simulatoren CathSim™ und PreOp™, welche mit der Übernahme von HT Medical Systems in ihr Portfolio übergegangen sind. Möglich wurde die Markteinführung durch die Verwendung von Standard-PCs und die damit verbundenen geringen Hardwarekosten.

CathSim™ [17; 203] ist ein Simulator mit dem das Einführen intravenöser Katheter trainiert werden kann, ein Verfahren, das bei 80% aller Krankenhauspatienten durchgeführt wird. Bedingt durch die PC-Hardware ist die Simulation auf das absolute Minimum beschränkt. Bei Kontakt der Nadel mit der Hautoberfläche wird diese bedingt durch den Nadeldruck lokal mittels eines stark vereinfachten Feder-Masse Systems verformt. Als Interfacehardware dienen eine reale Katheternadel und ein Plastikarm. Obwohl die Simulation und die verwendeten Modelle sehr einfach sind, haben erste Tests mit Studenten

sehr positive Resultate wegen des hohen Realitätscharakters von CathSim™ ergeben [17].

PreOp™ [33] ist ein Trainingssystem für Bronchoskopie und simuliert somit das Einführen eines Endoskops durch die Luftröhre. Nach Informationen des Herstellers war PreOp™ weltweit das erste komplette 3D Operationssimulationssystem mit realistischer kommerzieller und medizinischer Anwendung.

Ein flexibles Bronchoskop wird durch ein kinematisches Modell von verknüpften Verbindungen simuliert. Zur Reduktion der Rechenzeit werden Deformationen nicht berechnet, sondern mittels eines mitgelieferten Editors modelliert. Die Simulation von Deformationen ist limitiert auf lokale Deformation von Polypen. PreOp™ bietet eine Reihe von sehr aufwändigen grafischen Effekten wie z.B. das Husten des Patienten, welche den Effekten bei 3D Computerspielen ähneln. Erste Studien sollen gezeigt haben, dass die Motivation, mit dem System zu arbeiten, bei Studenten sehr hoch ist.

1.2 Motivation und Übersicht

Während es möglich ist, Endoskope mit allen notwendigen Werkzeugen direkt mit Hilfe von 3D-Tools am Computer zu generieren, ist es bei der Erzeugung des Modells für den menschlichen Körper (bzw. Teile von diesem) erheblich zweckmäßiger, direkt auf Bilder verschiedenster bildgebender Diagnoseverfahren zurückzugreifen. Eine besondere Rolle spielt hier, ob ein allgemeingültiges, anatomisch richtiges Modell eines Menschen, oder ob ein Abbild eines konkreten, speziellen Patienten gewünscht bzw. erforderlich ist.

- **Unspezifisch, anatomisch richtiges Modell**
  In diesem Fall kann auf besonders ausgiebige Datensätze des „Visible-Human"-Projects (U.S. National Library of Medicine) zurückgegriffen werden, für die Menschen nach ihrem Tod tiefgefroren, in Schichten zerteilt und fotografiert worden sind. Dadurch entsteht ein sehr exaktes Modell, das gut anwendbar ist, zumal manuelle Korrekturen und Optimierungen des 3D-Modells durchaus vertretbar sind, da sie ja nur einmalig vorzunehmen sind. Ein solcher Datensatz kann für Trainingszwecke gut verwendet werden [65; 170]. Nachteilig ist hier zu erwähnen, dass beim Visible-Human-Project gesunde menschliche Körper die Basis für die Schnittbilder waren – für die Virtuelle Chirurgie ist aber ein bestimmtes Krankheitsbild (z.B. ein raumfordernder Tumor) bei der Simulation wünschenswert bzw. notwendig.
- **Spezifisch, patientenorientiertes Modell**
  Die klassische Quelle für einen solchen Datensatz ist eine Serie von Schnittbildern bildgebender Diagnoseverfahren des zu modellierenden Körperabschnittes des Patienten. Durch spezielle Rechenverfahren (den sog. Rendering- und Segmentierungs-Verfahren) können diese 2D-Bilder als 3D-Modelle visualisiert werden.

Die ersten Schritte zum virtuellen Menschen bilden üblicherweise tomographische Bildgebungsverfahren, mit deren Hilfe sich so genannte Volumendatensätze erzeugen

lassen - es sei an dieser Stelle erwähnt, dass von der Qualität dieser Daten die Qualität aller weiteren Schritte abhängen.

Ein weiterer Schritt in der Modellerzeugung für die Virtuelle Chirurgie ist die Segmentierung. Betrachtet man medizinisches Bildmaterial als bloße Ansammlung von aufnahmeabhängigen Funktionswerten, die durch Umsetzung in zugehörige Farbwerte visualisiert werden können, so kann man unter der Segmentierung den zur Aufnahme inversen Prozess verstehen, bei dem einzelnen Funktionswerten, unter Kenntnis der ursprünglichen Bildinformation, wieder eine inhaltliche Bedeutung zugeordnet wird. Die Zuordnung aller Funktionswerte und deren Zusammenfassung zu logischen Einheiten (Teilobjekten) würde somit wieder zum Ausgangsvolumen, jedoch in diskretisierter Form, führen [28; 32; 121; 167].

Zur computergrafischen Visualisierung, Manipulation und Animation von Objekten ist eine geeignete Modellierung erforderlich. Dabei stellt sich je nach Problemstellung die Frage, ob nur die sichtbaren Teile (die Objektoberflächen) oder vollständige Objektvolumen in ein geometrisches Modell überführt werden müssen. Für eine realistische Darstellung computergrafischer Modelle muss zudem ein geeignetes Beleuchtungs- und Reflexionsmodell herangezogen werden. In der Computergrafik lassen sich grundsätzlich zwei verschiedene Verfahren zur Repräsentation von dreidimensionalen Objekten unterscheiden:

- **Volumenbasierte Verfahren**
  Hier werden die Daten von bildgebenden Diagnoseverfahren direkt verwendet und mit Hilfe des Volume-Renderings dargestellt (siehe Kapitel 3).
- **Oberflächenbasierte Verfahren**
  Die Oberfläche der Objekte wird hier als Menge aus Polygonen (meist Dreiecke) dargestellt (siehe Kapitel 4).

Die Visualisierung von statischer Anatomie allein ist jedoch noch nicht ausreichend für eine überzeugende Chirurgiesimulation. Auch die Physiologie muss zu einem gewissen Grad repräsentiert werden. Bei der Interaktion mit den virtuellen Instrumenten muss das modellierte Gewebe deformiert und manipuliert werden können. Da dies von der Berechnungszeit sehr aufwändige Algorithmen erfordert, wird im Kapitel 2 ein alternatives Deformationsverfahren dargestellt, das im Gegensatz zu allen anderen veröffentlichten Verfahren auf der Basis von kooperativen Neuro-Fuzzy Systemen arbeitet. Das Verfahren basiert, wie viele andere Verfahren auch, auf physikalischen Gesetzen. Dennoch können sowohl medizinisches Expertenwissen als auch Messdaten zur Optimierung des Realismus der Deformation eingesetzt werden.

Kapitel 3 und 4 beschreiben Verfahren für Volume-Rendering und Surface-Rendering im Hinblick auf die interaktive Darstellung von Deformationen und die Erhöhung des Realismus für die Chirurgiesimulation. Methoden zum Aufbringen von Oberflächendetails werden ebenso angesprochen wie Algorithmen für die Kollisionserkennung zwischen virtuellen Instrumenten und anatomischen Modellen.

Kapitel 5 enthält die wesentlichen Ergebnisse und Prototypen, die im Rahmen der Arbeit des Autors in der Virtuellen Chirurgie entstanden sind. Zum einen sind dies drei Pro-

totypen von Chirurgiesimulatoren in den Bereichen Gynäkologie, Handchirurgie und Neurochirurgie, zum anderen ein Qualitätsvergleich des für die Neurochirurgiesimulators verwendeten Volume-Rendering Verfahrens im Vergleich zu anderen und im Vergleich zu realen Aufnahmen.

Abschließend folgt nach einer eingehenden Diskussion der Ausblick auf zukünftige Erweiterungen und Entwicklungen.

2 DEFORMATIONSSIMULATION

Die reine Virtuelle Endoskopie beschränkt sich auf die Visualisierung von statischen Objekten. Wenn darüber hinaus auch die Simulation von chirurgischen Eingriffen realisiert werden soll, ist auch die Darstellung von Verformbarkeit und Destruktion der virtuellen Modelle erforderlich, und dies bedingt den Übergang von statischen zu deformierbaren Objekten.

Bei der Deformationssimulation gibt es eine grundsätzliche Abgrenzung zwischen Verfahren, die auf physikalischen Gesetzen beruhen und der Berechnung auf Basis einfacher Funktionen. Funktionale Berechnungen sind sehr schnell durchzuführen, daher wurden sie anfänglich für Simulation von Gewebedeformationen eingesetzt (siehe z.B. [16; 25; 38; 48; 150; 151; 181]). Aufgrund des geringen Realismus und der Schwierigkeit, die Funktionen mathematisch anzupassen, um die Realitätseindruck zu erhöhen, sollen diese Verfahren hier nicht berücksichtig werden.

Physikalisch basierte Methoden hingegen beruhen auf Gesetzen der newtonschen Mechanik, berechnen Positionen aus Geschwindigkeiten und Beschleunigungen von Massen. Generell sind diese Verfahren den funktionalen Berechnungen vorzuziehen, da sie sich, zumindest theoretisch, an der Realität orientieren und auf seit Jahrhunderten bekannten Gesetzen beruhen. Meist wird die Finite Elemente Methode (FEM) [227], eine seit Jahrzehnten bekannte Technik zur rechnerunterstützten Berechnung und Simulation von Belastung und Verformung in der mechanischen Festigkeitslehre, zur Deformationssimulation verwendet und bietet sich daher auch für die Modellierung von menschlichem Gewebe an. Erste Veröffentlichungen in diesem Zusammenhang stammen von Bro-Nielson [30-32], später folgten abgewandelte oder erweitere Verfahren [20; 45-47; 50; 90].

2.1 FEM-basierte Modelle

Meistens ist es nicht möglich, bei wissenschaftlichen Untersuchungen stark verflochtene Zusammenhänge in komplizierten Systemen unmittelbar und als Ganzes zu erfassen. Daher gehen Techniker und Wissenschaftler beim Behandeln solcher Sachverhalte üblicherweise so vor, dass sie ein zu untersuchendes System zunächst zweckdienlich in Elemente aufgliedern, sodann das Verhalten dieser Elemente betrachten, um schließlich im Ergebnis eines erneuten Zusammenfügens der Elemente Aufschlüsse über das Verhalten des Systems zu gewinnen. Die Modellierung eines Systems unter Verwendung einer endlichen Anzahl von definierten Elementen wird als *Diskretisierung* bezeichnet. In vielen Fällen ist für die mathematische Beschreibung einzelner Elemente eine infinitesimale Betrachtung erforderlich - die Elemente lassen sich in Form von Differenzialgleichungen oder äquivalenten Formulierungen beschreiben. Leider kann die Lösung der beschrei-

benden Differenzialgleichungen nur in sehr einfachen Fällen in mathematisch-analytischer Weise erfolgen. Im allgemeinen Fall sind hier Näherungsverfahren, wie beispielsweise die Überführung in ein Differenzen-Gleichungssystem, anzuwenden. Es handelt sich dabei in jedem Fall um Approximationen, von denen erwartet wird, dass sie mit zunehmender Verfeinerung der Diskretisierung gegen die exakte Lösung streben [227].

Bei der virtuellen Nachbildung von menschlichem Gewebe kann im Sinne der Verfahren der FEM von dem Problem des „elastischen Kontinuum" gesprochen und Verfahren der mechanischen Festigkeitslehre angewendet werden [19].

Bei der Strukturwahl finiter Elemente für die Modellierung von Volumenteilen ist die Entscheidung zwischen ein-, zwei- oder dreidimensionalen Objekten zu treffen. Für den Fall der eindimensionalen Elemente wird die Eigenschaft des Gesamtobjektes durch elastische Stäbe, bzw. Federn beschrieben, die über Knoten im Raum des Volumens in einer bestimmten Komplexität verbunden sind. Klassisches Beispiel dafür sind Feder-Masse-Systeme (siehe Kapitel 2.2). Ein anderes Beispiel für eindimensionale Elemente ist das Tensor-Masse-System – seine Erklärung und ein Vergleich wird in [46] gegeben.

Bei zweidimensionalen Elementen sind entsprechende Flächen die Grundlage der Modellierung. Hier werden die Elemente zur mathematischen Beschreibung der Oberfläche als Formflächen über Dreiecken betrachtet. Die Teilfläche über einem Dreieck wird dabei „Patch" genannt. Eine Oberfläche besteht aus einem Netz von Dreiecken und den darüber liegenden Patches. Die Form eines einzelnen Patch ist bestimmt durch eine gewichtete Formfunktion, wobei die Gewichtsvektoren den Patches die unterschiedliche Form verleihen. Die Gewichtsvektoren können durch Energieminimierung mittels der Finite Elemente Methode berechnet werden [40; 83; 144].

Bei dreidimensionalen Elementen handelt es sich um echte Teilvolumina des Gesamtobjektes, wobei meistens Tetraeder bevorzugt werden, da jeder Polyeder durch eine bestimmte Anzahl von Tetraedern dargestellt werden kann [71]. Der Einsatz von solchen FEM-Modellen in der Chirurgie-Simulation wird u.a. in [20; 30; 47; 50; 59; 90; 198] vorgeschlagen.

Generell hängt das Verhalten von FEM-Systemen im Wesentlichen von folgenden Eigenschaften ab:

- Die Art und die (innere) mathematisch Beschreibung der einzelnen FE
- Die Art der Komposition des Gesamtsystems durch die einzelnen FE

Von außen betrachtet sind Finite Elemente nur durch ihre Randpunkte bzw. ihre Randflächen und dem entsprechenden funktionalen Zusammenhang zueinander formuliert. Der genaue innere Verlauf der einzelnen Größen wird hierbei von so genannten Formfunktionen beschrieben und ist im einfacheren Fall als linear anzunehmen. Nichtlineares Verhalten führt aufgrund der Komplexität der Formfunktionen zu einer massiven Erhöhung des Rechenaufwandes und wird häufig dadurch vermieden, dass bei der Wahl der Modellierung durch FE die Diskretisierung so vorgegangen wird, dass eine dann angewendete Linearisierung möglichst geringe Fehler verursacht.

Die praktische Nutzung der FEM hängt ursächlich mit der Anwendung von Rechnersystemen zusammen. Der notwendige Rechenaufwand hängt hierbei einerseits von der

Komplexität der Formfunktion und andererseits von der Anzahl der finiten Elemente ab. Wenn die FEM als Grundlage für die Echtzeitdarstellung von deformierbaren Objekten dienen soll, ist eine entsprechend hohe Rechnerleistung erforderlich.

In der chirurgischen Simulation geht es nicht unbedingt um die exakte Modellierung des menschlichen Gewebes, sondern vielmehr um realitätsnahe Echtzeit-Darstellung des Verhaltens des Gewebes bei Deformation. In diesem Sinne können zur Verkleinerung des Rechenaufwandes verschiedenste Überlegungen angestellt werden:

- Die dynamische Deformations-Berechnung von Gewebe beschränkt sich auf die (sichtbaren) Randgebiete.
- Bei der Simulation wird Deformation durch Kollision von Operations-Tools mit dem Gewebe bestimmt bzw. eingeleitet. Aus diesem Grund bietet sich eine entsprechend räumlich begrenzte Berechnung an [137].
- Bei der Virtuellen Endoskopie werden in der Regel nur die inneren Oberflächen von Hohlräumen betrachtet. In diesem Sinne sind entsprechende Oberflächenmodelle völlig ausreichend bei der Modellbildung und können so komplexe (und rechenintensive) Volumenmodelle ersetzen.

2.2 Feder-Masse Systeme

Verschiedenste Realisationsansätze der FEM wurden bereits für Deformationsmodelle im Bereich der Chirurgiesimulation durchgeführt. Erste Verfahren zur interaktiven Deformationssimulation auf Basis von eindimensionalen FEM wurden von Terzopoulos vorgestellt [200-202]. Später wurden auch mehrdimensionale FEM in der interaktiven Chirurgiesimulation verwendet [20; 30; 47; 50; 59; 90; 225]. Diese aufwändiger zu berechnenden Modelle sind allerdings auch nicht in der Lage, medizinisches Gewebe überzeugend zu simulieren, vielfach wird nicht einmal eine subjektiv vom Chirurgen empfundene Verbesserung gegenüber einfachen Modellen deutlich [44; 180; 196]. Dies liegt nicht an der fehlenden Komplexität der Modelle, sondern an der Schwierigkeit der Bestimmung grundlegender, das Gewebe beschreibender, Parameter [160; 196]. Für medizinisches Gewebe ist dies deswegen so relevant, da für einfache Modelle bereits hunderttausende Parameter eingestellt werden müssen, die Parameterverteilung innerhalb des Gewebes nicht homogen ist und die Parameter sehr stark von physiologischen Aspekten abhängen.

Im Rahmen dieser Arbeit wird aus Performancegründen ein *Feder-Masse System* mit eindimensionalen finiten Elementen verwendet, da von der Prämisse ausgegangen wird, dass medizinisches Gewebe nicht durch komplexere Modelle realistischer simuliert werden kann, sondern primär durch die Bestimmung der das Gewebe beschreibenden Parameter. Eine solche Vorgehensweise birgt diverse Vorteile:

- Die Bearbeitung von Gewebe (z.B. beim Schneiden), welche letztendlich zur Destruktion vom Modell führt, kann bei Feder-Masse Systemen sehr gut implementieren werden und ist sogar während der Simulation durchführbar. Eine *Destruktion* ist definiert durch eine interne Modellveränderung durch Modifikation der Verknüpfungen der Elemente.
- Der Rechenaufwand, welcher für die Echtzeitdarstellung von Deformationen erforderlich ist, ist durchaus vertretbar.
- Feder-Masse Systeme benötigen keine Vorausberechnung, wie es bei einigen FEM der Fall ist.
- Im Rahmen dieser Arbeit wird gezeigt, dass Feder-Masse Systeme sich durch Neuro-Fuzzy Systeme realisieren lassen, mit denen medizinisches Expertenwissen genutzt und anhand realer Daten das physikalische Verhalten erlernt werden kann.

Feder-Masse Systeme sind bereits seit Jahrzehnten vorgeschlagene Modelle für die Visualisierung von Deformation virtueller Objekte. Erste bekannte Veröffentlichungen gehen auf Demetri Terzopoulos zurück [200-202]. Besondere Anforderungen bei der Modellierung von flächenförmigen Objekten wie Flaggen und Segel waren bei Veröffentlichungen von Provot im Mittelpunkt [149]. Aber auch neuere Simulationssysteme nutzen Feder-Masse Modelle [25; 46], die im Folgenden kurz beschrieben werden.

Bei *Feder-Masse Systemen* erfolgt eine Aufteilung in so genannte *Masseknoten* oder Knoten, die per Definition keine räumliche Ausdehnung besitzen, und in mechanische, masselose Verbindungseinheiten wie *Federn* oder Dämpfungselemente, die jeweils zwei Masseknoten verknüpfen. Die Summe aller Masseknoten entspricht also der Gesamtmasse des Objektes. Auf jeden Masseknoten *i* kann eine externe Kraft $\vec{F}_i$ einwirken, die beispielsweise durch eine Kollision mit einem anderen Objekt bewirkt wird. Durch Anwendung des Newtonschen Gesetzes $\vec{F}_i = m_i \cdot \vec{a}_i$ kann somit direkt die Beschleunigung $\vec{a}_i$ des Masseknotens berechnet werden. Zu beachten ist dabei, dass sowohl $\vec{F}_i$ als auch $\vec{a}_i$ dreidimensionale Vektoren sind. Die Geschwindigkeit $\vec{v}_i$ und Position $\vec{p}_i$ des Knotens kann durch Integrieren von $\vec{a}_i$ bzw. $\vec{v}_i$ nach der Zeit *t* berechnet werden.

$$\frac{\vec{F}_i}{m_i} = \vec{a}_i \quad , \quad \vec{v}_i = \int \vec{a}_i \; \mathrm{d}t \; , \quad \vec{p}_i = \int \vec{v}_i \; \mathrm{d}t \tag{2.1}$$

Die Kräfte, die die mechanischen Verbindungseinheiten ausüben, können ebenfalls durch physikalische Gesetze beschrieben werden: im einfachsten Fall für eine perfekt elastische Feder *s* mit dem *Hook'schen Gesetz* $\vec{F}_s = c_s \cdot \vec{e}_s$. Dabei wird durch $\vec{e}_s$ die gerichtete Streckung bzw. Stauchung der Feder und mittels der *Federkonstanten* c_s die Härte der Feder beschrieben. Die *Verbindungsmatrix V* eines Feder-Masse Systems ist eine $n \times n$ Matrix (wobei *n* die der Anzahl Knoten im System angibt) mit $v_{ij} = 1$, wenn die Knoten *i* und *j* verbunden sind, sonst 0.

Rein elastische Deformationen kommen in der Realität allerdings selten vor. Beispielsweise würde ein vollelastischer Gummiball, der senkrecht auf einen Tisch fällt, endlos weiterhüpfen. Jede Deformation würde sofort wieder in den Ursprungszustand zu-

rückschnellen. Um eine Dämpfung zu berücksichtigen, wird das Hook'sche Gesetz um eine Viskositätskomponente erweitert:

$$\vec{F}_s = c_s \cdot \vec{e}_s - d_s \cdot \vec{v}_s \tag{2.2}$$

Dabei gibt *d* die *Dämpfungs-* oder *Viskositätskonstante* und $\vec{v}_s$ die relative Geschwindigkeit der verknüpften Masseknoten zueinander an. Aus Geschwindigkeitsgründen sind die Gleichungen linear, was nur bei kleinen Deformationen physikalisch korrekt ist. Alle Gleichungen für die Masseknoten und Federn zusammengenommen ergeben ein Differenzialgleichungssystem, das mittels numerischer Lösungsverfahren per Computer berechnet werden kann [19].

Ein anderer, mehr intuitiver Ansatz wird durch iterative Algorithmen verwirklicht. Wird die Position eines Masseknotens verändert und somit das Modell verformt, werden die verbundenen Federn komprimiert bzw. gedehnt. Dabei wirken diese Elemente mit einer gewissen Kraft auf die umliegenden Masseknoten, die entsprechend ihrer Masse beschleunigt werden und somit andere Federn bzw. Masseknoten beeinflussen. Iterative Algorithmen berechnen genau diese Abfolge, bis sich ein Kräftegleichgewicht eingestellt hat. Zuerst werden die aktuellen Beschleunigungen der Masseknoten unter Berücksichtigung der internen Kräfte, die durch die Federn auftreten, berechnet. Aus der Integration der Beschleunigung über die Zeit wird die Geschwindigkeit berechnet. Leider kann dies mit numerischen Verfahren nicht kontinuierlich geschehen. Durch das Einführen einer Zeitschrittweite kann jedoch eine diskrete Approximation der Integration ermittelt werden.

Abbildung *2.1* zeigt ein einfaches Beispiel einer wie zuvor beschriebenen Simulation. Dargestellt sind zwei durch eine Feder verknüpfte Masseknoten, auf die externe Kräfte einwirken (gelbe Pfeile). Die Masseknoten werden dadurch in Richtung der hellblauen Pfeile beschleunigt. Durch die Trägheit der Masseknoten wird das System in eine Rotation versetzt, während die Feder abwechselnd gestreckt und gestaucht wird.

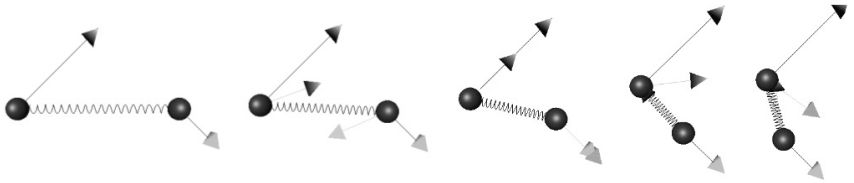

Abbildung 2.1*: Rotation zweier Masseknoten, die durch eine Feder verknüpft sind (aus* [153]*).*

Iterative Algorithmen sind sehr schnell in der Ausführung, weshalb sich der Einsatz in virtuellen Umgebungen anbietet. Allerdings gibt es ein grundsätzliches Problem durch die simple Approximation der Integrationen. Ist die Zeitschrittweite zu klein gewählt, so ist die Simulation zu langsam. Im entgegengesetzten Fall kann der Berechnungsfehler durch die Approximation zu groß werden; der Fehler akkumuliert sich, bis das System einen instabilen Zustand erreicht. Dies macht sich in der numerischen Simulation durch ein „Zerreißen" des Modells bemerkbar: Die Masseknoten driften innerhalb weniger

Zeitschritte auseinander und meist tritt ein interner Fließkommaüberlauf auf. Je fester die virtuellen Materialien sind, desto größer werden die internen Kräfte und desto kleiner muß die Zeitschrittweite sein. In [153] sind Anhaltspunkte für die Parameter des Feder-Masse Systems gegeben, die allerdings auch von der verwendeten Modellstruktur abhängen.

Für die Visualisierung in der Chirurgiesimulation ist insbesondere die Verknüpfung der Feder-Masse Systeme mit der darzustellenden Oberfläche oder dem entsprechenden Volumenmodell relevant. Der Aufbau von Feder-Masse Systemen für das deformierbare Volume-Rendering wird in Kapitel 3.3.1 erläutert. Die Verknüpfung von Oberflächenmodellen mit Feder-Masse Systemen wird in Kapitel 4.1 beschrieben.

Für eine realitätsnahe Simulation von Stoffen bzw. Körpergewebe durch Feder-Masse Systeme sind im Wesentlichen folgende Punkte von Bedeutung:

- **geometrische Anordnung von Knoten und Federn**
  Knoten und Federn haben nicht notwendigerweise etwas mit der dargestellten Oberflächenstruktur zu tun, wobei die sichtbaren Polygone eine Untermenge der Knoten als ihre Stützpunkte nutzen. Aus diesem Grund kann durch die Anordnung dieser Modellelemente die Eigenschaft des zu modellierenden Objektes in weiten Bereichen beeinflusst werden.
- **statische Parameter der Modellelemente**
  Jeder Knoten und jede Feder besitzt einen Parametersatz verschiedenster physikalischer Eigenschaften. Das gesamte Objekt wiederum besteht in der Regel aus einer großen Anzahl dieser Elemente. Durch Variation all dieser freien Konstanten kann das Verhalten des Objektes bei Deformation stark beeinflusst werden. Die einem realen Objekt zugrunde liegenden Parameter für die Simulation können grundsätzlich zwar aus physikalischen Überlegungen abgeleitet werden, dies stellt aber in der Regel aufgrund der komplexen Struktur, im speziellen auch bei menschlichem Gewebe, einen unüberschaubaren Aufwand dar.
- **Numerische Integration**
  Im Falle einer dynamischen Deformation in Echtzeit bedarf es aufgrund der großen Anzahl zu berechnender Elemente eines großen Rechenaufwands (numerische Integration). Auch im Sinne einer kontinuierlichen Rechenlastverteilung werden hier meist die Integrationen im konstanten Zeitabstand durchgeführt. Je kleiner dieses Zeitintervall, desto exakter die Berechnung, desto größer ist aber auch die Belastung für den Computer. In diesem Sinne ist ein entsprechender Kompromiss zwischen Performance des Systems und Genauigkeit der Integrationen zu finden. Eine Möglichkeit der Entlastung ist die begrenzt lokale Berechnung der Deformation. In vielen Fällen genügt es, wenn nur im unmittelbaren räumlichen Umfeld der Deformation, sprich der äußeren Krafteinwirkung, die entsprechenden Knoten und Federn berechnet werden, da durch den Aufbau des Objektes entferntere Teile keiner oder nur minimaler geometrischer Veränderung unterliegen würden [137]. Dieses Verhalten kann auch noch dadurch verbessert werden, dass der Berechnungsalgorithmus keinen fixen räumlichen Bereich als Grundlage verwendet, sondern sich an der tatsächlich

entstehenden Ausdehnung der Deformation orientiert. Als Nachteil ist hier zu erwähnen, dass in diesem Fall der Ort der lokalen Berechnung und der Ort des Einwirkens einer äußeren Kraft stets korreliert werden müssen.

Zu beachten ist, dass es bei der Simulation von Deformationen verschiedenster Objekte grundsätzlich unterschiedliche Forderungen geben kann. Während beispielsweise bei der Simulation eines Crash-Tests in der Automobilindustrie die exakte physikalische Modellierung im Vordergrund steht und es nicht zwingenderweise erforderlich ist, dass die zugrunde liegenden Berechnungen in Echtzeit durchzuführen sind, ist es bei der Virtuellen Chirurgie von wesentlicher Bedeutung, dass dem Arzt ein realitätsnaher Echtzeiteindruck vermittelt wird, ohne dass notwendigerweise alle physikalischen Gegebenheiten eingehalten werden müssen. Diese Unterschiede spielen bereits bei der Modellbildung und bei der Findung der notwendigen Parameter eine wesentliche Rolle.

Wenn Feder-Masse Systeme direkt durch Umsetzung der zugrunde liegenden physikalischen Bewegungs-Differenzialgleichungen realisiert werden, gibt es neben einem möglichen Geschwindigkeitsproblem bei der Berechnung auch die Schwierigkeit, die große Anzahl der Parameter für die Elemente (Knoten und Federn) festzulegen. Als möglicher Ausweg wird hier die Verwendung von Neuronalen Netzen in Kombination mit Fuzzy Systemen zur Initialisierung vorgeschlagen. Mit diesen Neuro-Fuzzy Systemen, die im folgenden Abschnitt vorgestellt werden, kann sowohl medizinisches Expertenwissen in Form von (Fuzzy-)Regeln als auch reale, visuell aufgenommene, Deformationen zur Parameterbestimmung verwendet werden.

2.3 Echtzeitsimulation deformierbarer Objekte mit Neuro-Fuzzy Systemen

Künstliche Neuronale Netze werden mittlerweile in einer Vielzahl von Anwendungen erfolgreich eingesetzt. Im Prinzip können mit Neuronalen Netzen – je nach Struktur und Aufbau - beliebige stetige mathematische Funktionen approximiert werden [131]. Durch die Lernfähigkeit sind Neuronale Netze in der Lage, die Approximation durch Vorgabe von Trainings-Datenmengen (z.B. Ein- und Ausgabetupel) selbstständig zu ermitteln. Im Zusammenhang mit der Deformationssimulation, die dem Lösen eines Differenzialgleichungssystems mit vielen Unbekannten entspricht, gilt es, eine für die Lösung geeignete Netzstruktur und deren Parameter zu ermitteln.

Durch die Interpretation eines beliebigen Feder-Masse Systems als Neuronales Netz ist es möglich, die meist unbekannten physikalischen Parameter des Feder-Masse Systems anhand realer Messdaten zu erlernen. Dies setzt allerdings das Vorhandensein von Trainingsdaten voraus. Gerade im Bereich der Virtuellen Chirurgie liegt aber Expertenwissen in Form von Erfahrung der Chirurgen vor, das für die Modellierung virtueller Organe genutzt werden sollte. Natürlich ist dieses Wissen nicht explizit in Form physikalischer Parameter vorhanden, sondern vielmehr als vage Einschätzungen in Form umgangssprachlicher Regeln oder medizinischer Fachtermini. Für Modellierungen vager Ausdrücke eignen sich Fuzzy Systeme in hervorragender Weise. Deshalb wurde im Fol-

genden ein Neuro-Fuzzy Ansatz gewählt, der die Vorteile der Neuronalen Netze mit denen der Fuzzy Systeme verbindet. Für die Deformationssimulation wurde ein so genannter kooperativer Neuro-Fuzzy Ansatz entwickelt [131], bei dem die Neuronalen Netze und die Fuzzy Systeme getrennt arbeiten. Nach einer kurzen Einführung in Neuronale Netze wird daher in den folgenden Kapiteln die für die Deformationssimulation geeignete Netzstruktur mit entsprechenden Lernverfahren beschrieben und die Berücksichtigung von Expertenwissen anhand von Fuzzy Systemen erklärt.

2.3.1 Neuronale Netze

Während Probleme, die durch einen effizienten Algorithmus beschrieben werden können, von einem Computer deutlich schneller gelöst werden als von einem Menschen, benötigt das menschliche Gehirn für viele Aufgaben, wie das Erkennen eines Gesichts, wesentlich weniger Zeit und vollbringt die Aufgabe in einer höheren Qualität und Genauigkeit. Ein weiterer Vorteil des menschlichen Gehirns ist, dass auch dann noch korrekte Ergebnisse geliefert werden, wenn es zu einem Ausfall einiger für die Problemlösung notwendiger Nervenzellen kommt. Selbst wenn die "Eingaben" ungenau sind, also beispielsweise ein Text durch Verschmutzung unleserlich geworden ist, kann das Gehirn eine Erkennung durchführen, während ein Computer in diesen Fällen oftmals fehlerhafte bzw. unbrauchbare Ergebnisse liefert. Die Idee ist daher, die Arbeitsweise des Gehirns auf Maschinen zu übertragen.

Künstliche Neuronale Netze[1] bestehen - wie das Gehirn von Säugetieren - aus einer großen Anzahl kleiner Elemente, den Neuronen. Information wird verarbeitet, indem sich die Neuronen mit Hilfe von gerichteten Verbindungen untereinander aktivieren. Dies geschieht im Prinzip analog zu den Vorgängen im Gehirn. Neuronale Netze zeichnen sich dabei durch ihre Lernfähigkeit aus. Sie können eine Aufgabe anhand von Trainingsbeispielen erlernen, ohne dazu explizit programmiert werden zu müssen. Weitere Vorteile sind die hohe Parallelität bei der Informationsverarbeitung, die hohe Fehlertoleranz und die verteilte Wissensrepräsentation, wodurch ein zerstörtes Neuron meist nur einen relativ kleinen Wissensausfall bedeutet.

Die Idee des Einsatzes Neuronaler Netze für die Deformationssimulation ist, die Lernfähigkeit zur Bestimmung der physikalischen Parameter des zugrunde liegenden Feder-Masse Systems zu nutzen. Wenn eine Repräsentation eines Neuronalen Netzes für ein Feder-Masse System gefunden wird, sind aber nicht nur die Lernverfahren, sondern auch alle anderen, gut untersuchten Möglichkeiten eines Neuronalen Netzes anwendbar.

Die einzelnen Neuronen eines Neuronalen Netzes übernehmen die Funktion einfacher Automaten oder Prozessoren, die aus ihrer aktuellen Eingabe und ihrem Zustand (Aktivierung) ihren neuen Zustand und ihre Ausgabe berechnen. Bei künstlichen Neuronalen Netzen wird der Zustand zur Berechnung der Ausgabe meist nicht betrachtet und man benutzt zur Bestimmung eine Ausgabe- und eine Propagierungsfunktion. Die Propagierungsfunktion berechnet aus den anliegenden Eingaben die Netzeingabe. Die Ausgabe-

---

[1] *Künstliche Neuronale Netze werden im Folgenden nur noch mit Neuronale Netze bezeichnet.*

funktion bestimmt aus der Netzeingabe schließlich mittels einer Aktivierungsfunktion die Ausgabe des Neurons.

Ein Neuronales Netz ist somit nach [131] definiert durch:

Definition 2.1

1. Eine endliche Menge von Verarbeitungseinheiten U (die Neuronen).
2. Eine Menge von externen Eingabefunktionen EXT, die jedem Neuron u_i eine externe Eingabe ext_i zuordnet.
3. Einer Gewichtungsmatrix W, wobei w_{ij} das Verbindungsgewicht zwischen Neuron u_i und u_j definiert.
4. Eine Menge von Eingabefunktionen NET, wobei net_i die Netzeingabe für Neuron u_i basierend auf den Verbindungsgewichten und der Ausgabe der verbundenen Neuronen berechnet.
5. Einer Menge von Aktivierungsfunktionen A, wobei a_i die Aktivierung des Neurons u_i basierend auf seinen aktuellen Eingaben net_i, der externen Eingabe ext_i und einem Biaswert berechnet.
6. Einer Menge von Ausgabefunktionen O, wobei o_i die Ausgabe von Neuron u_i basierend auf seiner aktuellen Aktivierung berechnet.

Während die Struktur innerhalb biologischer Neuronaler Netze sehr komplex sein kann, beschränkt man sich bei der Betrachtung künstlicher Neuronaler Netze meist auf Netzwerke mit einem hierarchischen Aufbau. Hierbei werden die Neuronen in Schichten zusammengefasst. Die Bestimmung der Ausgabe des Netzes erfolgt in einem vorwärtsbetriebenen, schichtweise aufgebauten Netz, einer so genannten *Vorwärtsarchitektur*, durch aufeinanderfolgende Berechnung der Ausgaben von der untersten zur obersten Schicht. Dieser Vorgang wird auch als *Propagation* bezeichnet.

Das Lernen in einem Neuronalen Netz erfolgt durch Veränderung der Netzgewichte und Schwellenwerte. Das Ziel des Lernvorganges ist dabei, das Netz so zu trainieren, dass es auf bestimmte Netzeingaben mit einer bestimmten Ausgabe reagiert. Auf diese Weise ist es dann auch in der Lage, auf neue unbekannte Eingaben mit einer geeigneten Ausgabe zu antworten. Das Prinzip der existierenden Lernverfahren basiert darauf, zunächst eine Eingabe durch das Netz zu propagieren und diese dann mit der gewünschten Ausgabe zu vergleichen. Anschließend werden die Gewichte und Schwellenwerte so verändert, dass das Netz bei nochmaliger Eingabe des selben Musters der gewünschten Ausgabe näher kommt. Beispiele von Lernalgorithmen sind z.B. der Lernalgorithmus des Perceptrons [176] oder der Backpropagation-Algorithmus des Multilayer-Perceptrons [127]. Ein genauere Beschreibung und eine allgemeine Einführung in Neuronale Netze findet man in [74; 174].

Allerdings sind die Vorwärtsarchitekturen nicht ohne externe Eingaben in der Lage, dynamische Systeme zu berechnen, da sie keine Informationen über den Zustand speichern können, sondern für jede Propagation immer nur die zur Verfügung gestellten Eingabevektoren verwenden [134]. Im folgenden Abschnitt werden daher spezielle Neuro-

nale Netze beschrieben, die zur Beschreibung eines dynamischen Systems herangezogen werden können.

2.3.1.1 Rückgekoppelte Neuronale Netze

Eine rückgekoppelte Neuronale Netzstruktur erlaubt Schleifen und Rückwärtsverbindungen zwischen Knoten, welche im Gegensatz zu reinen Vorwärtsarchitekturen die Beschreibung dynamischen Verhaltens gestatten. Die Rückkopplung ermöglicht es dem Netzwerk, frühere Zustände zu erinnern und es ist daher in der Lage diese Zustände zusammen mit den aktuellen Eingaben für die Berechnung neuer Zustände heranzuziehen. Abhängig von der Struktur und Gewichtung ist es möglich Neuronale Netze zu erstellen, die zu einem Fixpunkt konvergieren und begrenzte Zyklen und chaotisches oder dynamisches Verhalten beschreiben [110].

Im Folgenden werden rückgekoppelte Neuronale Netze formal kurz dargestellt, da dies für die weitere Erläuterung des Deformationsverfahrens erforderlich ist. Eine genauere Beschreibung findet sich u.a. in [124; 145].

Im Gegensatz zu Forwärtsarchitekturen muss in rückgekoppelten Neuronalen Netzen eine Zeitkomponente berücksichtigt werden. Formal wird für ein Neuron j die Ausgabe $o_j(t)$ für den Zeitschritt t unter Benutzung der externen Eingaben $ext_j(t - \mathrm{d}t)$, der Ausgaben der verbundenen Neuronen $o_i(t - \mathrm{d}t)$ und den Netzgewichten w_{ij} wie folgt berechnet [136]:

$$o_j(t) = a_j(net_j(t)) = a_i(\sum_i o_i(t - \mathrm{d}t)w_{ij} + ext_j(t - \mathrm{d}t)) \tag{2.3}$$

Für den kontinuierlichen Fall erhält man [145]:

$$\frac{\mathrm{d}o_j}{\mathrm{d}t} = -o_j + a_j(\sum_i o_i w_{ij} + ext_j) \tag{2.4}$$

Somit ist ein rückgekoppeltes Neuronales Netz in der Lage ein System von Differenzialgleichungen zu lösen und deren Parameter zu bestimmen [136].

Für die Deformationssimulation werden nicht nur skalare Werte, sondern Vektoren verwendet, da jede Bewegung in drei Dimensionen ausgeführt wird. Im nächsten Abschnitt wird daher eine Möglichkeit zum Vektorisieren Neuronaler Netzen beschrieben.

2.3.1.2 Vektorisierte (rückgekoppelte) Neuronale Netze

Eine *Vektorisierung* eines Neuronalen Netzes kann durch die Ersetzung der skalaren Verarbeitungselemente von konventionellen Neuronalen Netzen durch „vektorverarbeitende Einheiten“ und durch Vektorisieren der Verbindungen zwischen den Neuronen geschehen [136]. Die Gewichte des Neuronalen Netzes bleiben skalare Werte und der Propagationsalgorithmus kann auf die gleiche Weise wie in konventionellen Neuronalen Netzen durchgeführt werden. Da die prinzipielle Struktur, der Datenfluss und die Verarbeitung nicht verändert werden, können auch existierende Lernverfahren für das Training benutzt werden.

Vektorisierte Neuronale Netze können direkt in konventionelle Neuronale Netze umgewandelt werden, wenn sie keine speziellen Vektoroperationen benutzen. Dies kann durch das Kopieren eines Neurons und dessen Verbindungen, die dann alle das gleiche

Gewicht verwenden, geschehen. Die Anzahl der kopierten Neuronen inklusive des Originals entspricht dabei der Dimension der verwendeten Vektoren.

Durch ein vektorisiertes rückgekoppeltes Neuronales Netz können Lern- und Propagationsverfahren in ein System von Differenzialgleichungen transferiert werden, um dessen Parameter zu Lernen[1] und um das Gleichungssystem zu lösen. Damit ist das Gerüst für einen problemspezifischen Modellierungs- und Definitionsprozess und einen problemspezifischen Lösungs- und Propagationsalgorithmus gegeben. Für das Lernverfahren können herkömmliche Algorithmen angepasst werden. Ein Beispiel eines modifizierten Lernverfahrens für ein vektorisiertes rückgekoppeltes Neuronales Netz wird im Folgenden präsentiert.

Ein oft benutztes Lernverfahren für rückgekoppelte Neuronale Netze ist *backpropagation through time* (BPTT) [177; 178]. Die Hauptidee dieser Methode ist die Umwandlung eines rückgekoppelten Neuronales Netzes in eine Forwärtsarchitektur durch das Auffalten nach der Zeit, wobei dann ein konventionelles backpropagation Lernverfahren wie es beispielsweise für das Multilayer-Perceptron eingesetzt wird [127], verwendet werden kann. Wie beim Backpropagationalgorithmus ist die Verwendung einer nichtlinearen, monoton steigenden, differenzierbaren Aktivierungsfunktion Voraussetzung.

Die Propagationsfunktion eines vektorisierten Neuronalen Netzes lässt sich bei Verwenden von Vektoren in der Gleichung (2.3) wie folgt definieren:

$$\vec{o}_j(t) = \vec{a}_j(n\vec{e}t_j(t)) = \vec{a}_i(\sum_i \vec{o}_i(t) w_{ij} + e\vec{x}t_j(t)) \tag{2.5}$$

wobei $\vec{o}_j(t)$ die Ausgabe von Neuron j zurzeit t und $e\vec{x}t_j(t)$ die (externe) Eingabe des Neurons j ist (falls vorhanden).

Sei E der totale Fehler, beziehungsweise die zu minimierende Kostenfunktion, aller Ausgabeneuronen k über alle Zeitschritte $t = 0, ..., T$:

$$E = \sum_{t=0}^{T} E(t) = \sum_{t=0}^{T} \left[\frac{1}{2} \sum_k (\vec{E}_k(t))^2 \right] \tag{2.6}$$

mit dem *Fehlermaß* für den Knoten k:

$$\vec{E}_k(t) = \begin{cases} \vec{p}_k(t) - \vec{o}_k(t) & \text{wenn Knoten } k \text{ eine gewünschte Ausgabe } \vec{p}_k \text{ zur Zeit } t \text{ hat} \\ 0 & \text{ansonsten.} \end{cases} \tag{2.7}$$

Dann kann der *Fehlergradient* wie üblich abgeleitet werden durch ([136]):

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \sum_{t=1}^{T} \frac{\partial E(t)}{\partial n\vec{e}t_j(t)} \frac{\partial n\vec{e}t_j(t)}{\partial w_{ij}} . \tag{2.8}$$

[1] *Dies wird in der Literatur auch als das Lösen eines inversen Problems bezeichnet [145].*

Das Fehlersignal $\delta(t)$ für jedes Neuron wird hierbei wie folgt definiert:

$$\vec{\delta}_j(t) = -\frac{\partial E(t)}{\partial net_j(t)} = -\frac{\partial E(t)}{\partial \vec{o}_j(t)} \frac{\partial \vec{o}_j(t)}{\partial net_j(t)} \tag{2.9}$$

Mit

$$\vec{A}_j(t) = \vec{E}_j(t) + \sum_k \vec{\delta}_k(t+1) w_{ik}$$

erhält man daher die speziellen Fehlersignale:

$$\vec{\delta}_j(t) = \begin{cases} \vec{a}'_j(net_j(t))\vec{E}_j(t) & \text{für } t = T, \\ \vec{a}'_j(net_j(t))\vec{A}_j(t) & \text{für } 1 \le t < T. \end{cases} \tag{2.10}$$

Mit Gleichung (2.8) können die Gradienten nach der Propagation über alle Zeitschritte abgeleitet werden:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_{t=1}^{T} \vec{\delta}_j(t) \vec{o}_i(t-1) \tag{2.11}$$

Der einzige Unterschied zum herkömmlichen BPTT Algorithmus ist die Benutzung des Skalarprodukts in den Gleichungen (2.6), (2.10) und (2.11). Daher hängt beispielsweise die Fehlerrate $\delta(t)$ von der Vektorkomponente des lokalen Fehlervektors in Richtung des Vektors des lokalen Gradienten der Aktivierungsfunktion a ab.

Die Zeitkomplexität dieser Lernmethode ist $O(T \cdot n^2)$. Der benötigte Speicher wird durch $O(T \cdot n + n^2)$ beschrieben, da die Ausgabe jedes Neurons für jeden Zeitschritt gespeichert werden muss.

2.3.1.3 Modellierung physikalischer Systeme mit Neuronalen Netzen

Wenn ein (physikalisches) System mit einem Neuronalen Netz modelliert werden soll, ist es üblich, zuerst Trainingsmuster, die das Eingabe-Ausgabe-Verhalten beschreiben, zu ermitteln. Erst dann wird ein Neuronales Netz, üblicherweise eine Forwärtsarchitektur, erzeugt. Die Zahl der Ein- und Ausgabeneuronen muss dabei der Zahl der Ein- und Ausgabeeinheiten des zu modellierenden Systems entsprechen. Die Zahl der inneren Neuronen wird meistens anhand von Erfahrungswerten definiert oder eine Pruningmethode (Stutzen) wird benutzt [74; 174]. Trotzdem ist die Netzstruktur in vielen Fällen komplexer als benötigt, auch wenn das erhaltene Neuronale Netz erfolgreich trainiert und gestutzt worden ist. Darüber hinaus ist es nicht möglich, die Netzparameter zur Vereinfachung des Lernprozesses zu initialisieren, wenn bereits Expertenwissen über das (physikalische) System vorliegt. Die Einbeziehung von Expertenwissen kann jedoch notwenig sein, wenn nur eine kleine Anzahl von Trainingsmustern vorhanden ist, wie es oftmals bei realen Problemen der Fall ist.

Eine andere Methode für die Modellierung eines (physikalischen) Systems unter Benutzung Neuronaler Netze ist die möglichst genaue direkte Nachbildung des mathematischen Modells durch das Neuronale Netz. Dies scheint der Idee der Neuronalen Netze zu wiedersprechen, da sie üblicherweise als modellunabhängige Architekturen angesehen werden, aber auch in biologischen Neuronalen Netzen können vorstrukturierte Neuronale

Netze für spezielle Aufgaben gefunden werden [53]. Die Hauptvorteile dieses Ansatzes sind die Möglichkeit der Interpretation des Modells zu jeder Zeit, die Möglichkeit der Initialisierung des Neuronalen Netzes mit Vorwissen, beispielsweise unterstützt durch ein Fuzzy System (siehe Kapitel 2.3.6), und die problemangepasste Größe des Netzes. Darüber hinaus ist der Trainingsvorgang üblicherweise einfacher und schneller und die Approximation und Generalisierungsqualität ist gegenüber herkömmlichen Ansätzen erhöht [136]. Ein Nachteil dieses Ansatzes ist häufig die Schwierigkeit ein mathematisches Modell in eine Neuronale Netzstruktur zu transferieren, insbesondere bei der Modellierung von Differenzialgleichungssystemen mit rückgekoppelten Architekturen. Offensichtlich kann dieser Ansatz nicht verwendet werden, wenn keine mathematische Beschreibung des Systems vorhanden ist.

Im Folgenden wird die Möglichkeit zur Beschreibung eines Feder-Masse Modells anhand eines einfachen Beispiels, der Definition eines Integrationsneurons mit einem Eingang und einem Ausgang (Abbildung *2.2*), erläutert.

Abbildung 2.2*: Integrationsneuron: Ein einfaches rückgekoppeltes Neuron (aus* [136]*).*

Sei die Aktivierungsfunktion a := id, w_p := 1 und w_v := dt. Dann ist die Propagationsfunktion durch

$$\begin{aligned} p(t) = o(t) &= a\left(\sum_i o_i(t-\mathrm{d}t)w_i + ext_j(t-\mathrm{d}t)\right) \\ &= p(t-\mathrm{d}t)w_p + v(t-\mathrm{d}t)w_v \\ &= p(t-\mathrm{d}t) + v(t-\mathrm{d}t)\,\mathrm{d}t \end{aligned} \tag{2.12}$$

definiert. Damit ergibt sich

$$\frac{p(t)-p(t-\mathrm{d}t)}{\mathrm{d}t} = v(t-\mathrm{d}t) \tag{2.13}$$

Für d$t \to 0$ gilt

$$\frac{\mathrm{d}\,p}{\mathrm{d}\,t} = v(t) \tag{2.14}$$

beziehungsweise

$$p(t) = \int v(t)\,\mathrm{d}t\ . \tag{2.15}$$

Der Integrator ist die Basis für die Neuronale Netzstruktur, die für die Beschreibung eines Feder-Masse Systems benötigt wird, da zur Berechnung der Geschwindigkeit, bzw. der Position die Integration der Beschleunigung bzw. der Geschwindigkeit benötigt wird. Auf dieselbe Weise können somit komplexe Differenzialgleichungen beschrieben werden.

Bei Benutzung konventioneller Propagationstechniken wird das System von Differenzialgleichungen iterativ unter Benutzung einer Zeitkonstanten (üblicherweise 1) berechnet. Dies resultiert in einer ungenügenden Genauigkeit, besonders wenn Echtzeitsynchronisation benötigt wird. Nur Systeme mit diskreten Zuständen (z.B. endliche Automaten [118]) können genau berechnet werden. Eine Diskussion von numerischen Methoden für die Berechnung von (rückgekoppelten) Neuronalen Netzen wird beispielsweise in [119] gegeben. Eine Propagationsmethode für die Benutzung in Echtzeitanwendungen wird in Kapitel 2.3.4 beschrieben. Die Hauptidee dieses Ansatzes ist das Festsetzen der Aktivierung bestimmter Neuronen während einer Propagation des Neuronalen Netzes, bis ein Energieminimum für den Zeitschritt erreicht ist. Erst dann wird die Propagation für den nächsten Zeitschritt gestartet.

Beide in diesem Kapitel beschriebenen Modellierungsmethoden können dazu benutzt werden, komplexere Modelle durch modulare Komponenten zu erstellen. Zum Beispiel kann ein physikalisches System interagierender Einheiten durch das Verknüpfen individueller Komponenten erzeugt werden, die separat trainiert werden können. Dies kann das Design und den Trainingsprozess erheblich vereinfachen und in vielen Fällen kann das verbundene System noch nachträglich trainiert werden, wenn keine Daten der „Verbindungspunkte“ vorhanden sind.

Diese vorstrukturierten modularen Neuronalen Netze werden im Folgenden benutzt, um das in Kapitel 2.2 beschriebene Feder-Masse System nachzubilden.

2.3.2 Die neuronale Netzstruktur eines Feder-Masse Systems

2.3.2.1 Dynamik der Masseknoten

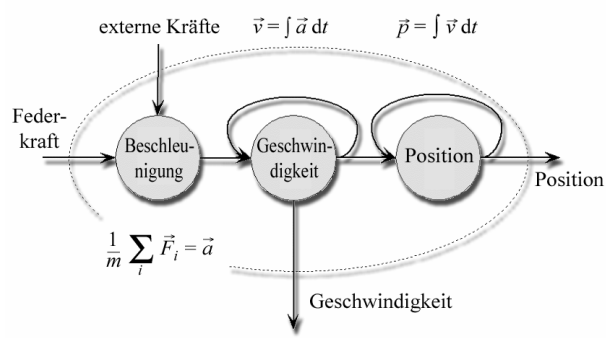

Abbildung 2.3*: Modulares Neuronales Netz für die Modellierung der Dynamik der Masseknoten (nach* [161]*).*

Um die Dynamik der Masseknoten zu simulieren, wird ein modulares, vektorisiertes, rückgekoppeltes Neuronales Netz verwendet, das aus drei Neuronen besteht, die die Position ($\vec{p}$, Positionsneuron), Geschwindigkeit ($\vec{v}$, Geschwindigkeitsneuron) und Beschleunigung ($\vec{a}$, Beschleunigungsneuron) der Masseknoten berechnen (siehe

Abbildung *2.3*). An das Beschleunigungsneuron, welches die aktuelle Beschleunigung berechnet, können externe sowie interne bzw. Federkräfte angelegt werden.

Die Geschwindigkeits- und Positionsneuronen sind rückgekoppelt und werden somit als Integratoren eingesetzt (siehe Kapitel 2.3.1.3).

Dieses modulare Neuronale Netz ist, bei geeigneter Wahl der Netzeingaben, -ausgaben und Gewichte, in der Lage, die in (2.1) (S 17) definierte Knotendynamik wie im Folgenden beschrieben zu berechnen[1].

Sei t_c die Zeitkonstante für die Simulation, s die Anzahl der Federn, n_{i1} und n_{i2} die Knotennummern, die durch Feder i verbunden sind, dann werden die Neuronen durch folgende Gleichungen beschrieben:

Die Netzeingaben, Netzausgaben und Aktivierungsfunktionen für alle Neuronen sind definiert als:

$$\begin{aligned} \vec{net}_j &= \sum_i (w_{ij} \cdot \vec{o}_i) \\ \vec{a}_j &= \vec{A}_j(\vec{net}_j) = \begin{cases} t_c \cdot \vec{net}_j & \text{für lokale Propagation} \\ \vec{a}_j & \text{für nächsten Zeitschritt} \end{cases} \\ \vec{o}_j &= \vec{O}_j(\vec{a}_j) = t_c \cdot \vec{a}_j \end{aligned} \tag{2.16}$$

Sei k die Nummer und m_k die Masse des Knotens, dann werden die Gewichte mit den folgenden Werten initialisiert:

$$\begin{aligned} &\text{Beschleunigungsneuron:} && w_{ij} = \begin{cases} 1 & \text{für } k = n_{i1} \\ -1 & \text{ansonsten} \end{cases} \\ &\text{Geschwindigkeitsneuron:} && w_{ij} = \frac{1}{m_k} \\ &\text{Positionsneuron:} && w_{ij} = 1 \end{aligned} \tag{2.17}$$

2.3.2.2 Dynamik der Federn

Zur Berechnung der Dynamik der Federn nach (2.2) (S 18) werden, wie in Abbildung *2.4* dargestellt, ebenfalls drei Neuronen benötigt. Das Positionskraftneuron berücksichtigt die relative Position und das Viskositätsneuron die Geschwindigkeit der verbundenen Knoten. Die Ausgaben dieser Neuronen werden im Gesamtkraftneuron zur Berechnung der Federkraft herangezogen. Die Netzeingabe- und Ausgabefunktionen sind für alle Neuronen definiert als:

$$\begin{aligned} \vec{net}_j &= \sum_{i=1}^{2} (w_{ij} \cdot \vec{o}_i) \\ \vec{o}_j &= \vec{O}_j(\vec{a}_j) = \vec{a}_j \end{aligned} \tag{2.18}$$

---

[1] *Um die einwirkenden Kräfte aller verbundenen Feder zu berücksichtigen, wird im Gegensatz zu (2.1) die Summe der einwirkenden Kräfte verwendet.*

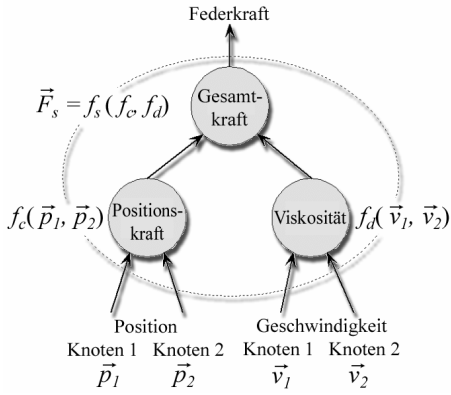

Abbildung 2.4*: Modulares Neuronales Netz für die Modellierung der Dynamik der Federn (nach* [161]*).*

Die Aktivierungsfunktion und die anfänglichen Gewichte der Neuronen sind wie folgt definiert:

$$\begin{aligned}
&\text{Positionskraftneuron:} && w_{1j} = 1,\ w_{2j} = -1 \\
& && \vec{a}_j = \vec{A}_j(n\vec{e}t_j) = \vec{f}_c(n\vec{e}t_j) \\
&\text{Viskositätsneuron:} && w_{1j} = 1,\ w_{2j} = -1 \\
& && \vec{a}_j = \vec{A}_j(n\vec{e}t_j) = \vec{f}_d(n\vec{e}t_j) \\
&\text{Gesamtkraftneuron:} && w_{1j} = c,\ w_{2j} = d \\
& && \vec{a}_j = \vec{A}_j(n\vec{e}t_j) = \vec{f}_s(n\vec{e}t_j)
\end{aligned} \tag{2.19}$$

wobei $\vec{f}_c(n\vec{e}t_j)$ die benutzte Federfunktion, $\vec{f}_d(n\vec{e}t_j)$ die Viskositätsfunktion und $\vec{f}_s(n\vec{e}t_j)$ die Gesamtkraftfunktion angeben. c und d beschreiben für lineare Federmodelle die Feder- und Viskositätskonstante. Ansonsten gilt $c = 1$ und $d = 1$.

Diese Neuronen berechnen das erweiterte Hook'sche Gesetz, das unter Benutzung der Gleichungen (2.2) und (2.19) wie folgt angegeben werden kann:

$$\vec{F}_s = \vec{f}_s\left(c \cdot \vec{f}_c(\vec{p}_1, \vec{p}_2) + d \cdot \vec{f}_d(\vec{v}_1, \vec{v}_2)\right) \tag{2.20}$$

Mittels der Funktionen $\vec{f}_c, \vec{f}_d$ und $\vec{f}_s$ kann ein beliebiges lineares oder nichtlineares Verhalten der Federn implementiert werden. Im Falle linearer Modelle gilt beispielsweise:

$$\vec{f}_c(\vec{p}_1, \vec{p}_2) = (\vec{p}_1 - \vec{p}_2)\left(\frac{l_{norm}}{|\vec{p}_1 - \vec{p}_2|} - 1\right)$$

$$\vec{f}_d(\vec{v}_1,\vec{v}_2) = \vec{v}_1 - \vec{v}_2 \tag{2.21}$$

$$\vec{f}_s(\vec{f}_c,\vec{f}_d) = c \cdot \vec{f}_c - d \cdot \vec{f}_d$$

wobei l_{norm} die initiale bzw. undeformierte Länge der Feder angibt. Die Funktion $\vec{f}_c$ kann dabei durch ein Neuronales Netz mit multiplikativen Verbindungen berechnet werden [146].

Durch Verwendung der modularen Neuronalen Netze der Masseknoten und Federn können wie bei einem Feder-Masse System größere Einheit zusammengesetzt werden, wofür im folgenden Abschnitt ein Beispiel gegeben wird.

2.3.3 Das modulare Neuronale Netz eines Feder-Masse Systems

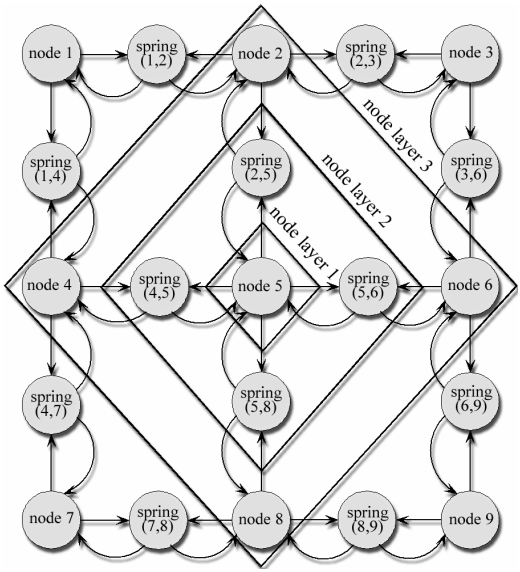

Abbildung 2.5*: 2D-Gitter aus modularen Neuronalen Netzen der Masseknoten (node) und Federn (spring) (nach* [161]*).*

Abbildung *2.5* zeigt ein Beispiel eines 2D-Gitters aus modularen Neuronalen Netzen der Masseknoten und Federn. Das Gitter ist strukturiert aus alternierenden Knoten und Federschichten. Das Differenzialgleichungssystem dieses Gitters kann aus den Gleichungen (2.1) und (2.20) wie folgt hergeleitet werden.

$$\vec{F}_s^{(i,j)} = \vec{f}_s^{(i,j)}\left(c^{(i,j)} \cdot \vec{f}_c^{(i,j)}(\vec{p}_i,\vec{p}_j) + d^{(i,j)} \cdot \vec{f}_d^{(i,j)}(\vec{v}_i,\vec{v}_j)\right)$$

$$\frac{\sum_{l=0}^{n} \vec{F}_l}{m_k} = \vec{a}_k = \frac{\mathrm{d}\,\vec{v}_k}{\mathrm{d}\,t} = \frac{\mathrm{d}^2\,\vec{p}_k}{\mathrm{d}\,t^2} \tag{2.22}$$

$$\forall\, i, j, k \in \{1, \ldots, 9\},\ j = i + 1 \wedge j = i + 3$$

wobei (i, j) die Feder zwischen den Knoten i und j beschreibt. Üblicherweise werden die Funktionen $\vec{f}_c^{(i,j)}$, $\vec{f}_d^{(i,j)}$ und $\vec{f}_s^{(i,j)}$ für alle Federn einheitlich gewählt.

Durch die Struktur unter Benutzung der modularen Neuronalen Netze, kann der Aufbau des gesamten Neuronalen Netzes direkt nach Vorgabe eines Feder-Masse Systems erfolgen. Die Modifikation der Netzwerkstruktur während der Simulation ist damit ebenfalls möglich, sodass beispielsweise Schnitte und Risse simuliert werden können, was insbesondere für die Chirurgiesimulation interessant ist.

2.3.4 Propagationsverfahren für die Deformationssimulation

Bei Benutzung konventioneller Propagationstechniken für rückgekoppelte Neuronale Netze, wie auch bei fast allen anderen numerischen Simulationsverfahren, wird das System von Differenzialgleichungen iterativ unter Benutzung einer Zeitkonstanten t_c berechnet. Um trotzdem eine genaue Approximation des dynamischen Verhaltens zu erreichen, ist es sinnvoll, den Propagationsalgorithmus in zwei Schritte aufzuteilen. Während des ersten Schritts, der lokalen Propagation, wird das Neuronale Netz propagiert, bis ein lokales Energieminimum erreicht wird. Die Masseknoten im Neuronalen Netz erreichen dann ihre Position für den aktuellen Zeitschritt (vgl. hierzu die Propagation in Hopfield Netzen [81]). Während dieses Propagationsschritts werden die Aktivierungen der Neuronen der Masseknoten nicht verändert (siehe Gleichung (2.16)).

Im zweiten Schritt werden die Aktivitäten a_i der Neuronen für den nächsten Zeitschritt berechnet. Am Ende dieses Propagationsschritts enthalten die Aktivitäten der Neuronen die Positionen, Geschwindigkeiten und Beschleunigungen der Masseknoten zur Zeit $t_0 + n \cdot t_c$, wobei n der Index des aktuellen Zeitschritts ist und t_0 die Startzeit angibt.

In beiden Propagationsschritten wird das Netz bei Beginn jedes Propagationsschritts im aktuellen Zustand eingefroren. Danach werden alle Federn berechnet. Die Eingaben jeder Feder sind die Positionen und Geschwindigkeitsvektoren der verbundenen Masseknoten, die Ausgabe ist ein Kraftvektor in Richtung der Feder. Wenn alle Federn aktualisiert sind, wird die Berechnung der Masseknoten gestartet.

Als Netzeingabe jedes Knotens wird die gewichtete Summe der externen und internen Kraftvektoren benutzt (Abbildung *2.3*). Die internen Kraftvektoren werden von den verbundenen Federn berechnet. Die Berechnung aller modularen Neuronalen Netze der Masseknoten und Federn kann parallel geschehen. Bei Verwendung üblicher Rechnerarchitekturen ist es effektiv, Masseknoten und Federn zu größeren Einheiten zu gruppieren und jeweils auf einem Prozessor zu berechnen. Allerdings ist die Reihenfolge der Berechnung während der Propagation wichtig für die Minimierung der benötigten Schritte. Diese Reihenfolge wird mit einer Heuristik bestimmt. Während des ersten Propagationsdurchlaufs wird der Knoten mit dem größten Kraftvektor identifiziert und der nächste

Durchlauf mit diesem Knoten gestartet. Während der folgenden Propagationsdurchläufe wird der Knoten mit der größten Kraftdifferenz zwischen der im letzten Durchlauf berechneten Kraft und der neuen Kraft ermittelt. Falls die Differenz größer als ein Schwellenwert ist, wird ein neuer Propagationsdurchlauf gestartet. Ansonsten bricht die aktuelle Propagation ab.

Das Neuronale Netz konvergiert damit zu einem durch das Energieminimum des dynamischen Systems definierten Fixpunkt, falls dieser existiert. Dies ist beispielsweise der Fall, wenn die Außenpunkte des Feder-Masse Systems festgesetzt sind und eine konstante Kraft an einen Knoten angelegt wird. Eine Objektrotation beispielsweise beschreibt dagegen eine periodische Oszillation.

Wenn die Parameter des dynamischen Systems im Verhältnis zur Zeitschrittweite ungünstig gewählt worden sind, tendiert das Neuronale Netz, wie alle numerischen Lösungsverfahren für Differenzialgleichungssysteme, zu einem instabilen oder chaotischen Verhalten. In [153] ist eine Tabelle für den Zusammenhang zwischen den Parametern eines Feder-Masse Systems und der Zeitschrittweite bei Verwendung iterativer Verfahren aufgeführt.

2.3.5 Automatisches Erlernen von Deformationseigenschaften realer Gewebe

Interessant wird die Verwendung von Neuronalen Netzen für die Beschreibung von Feder-Masse Systemen, insbesondere durch deren Lernfähigkeit. Die Gewichte im Neuronalen Netz können automatisch adaptiert werden, sodass sie bestimmte Lernaufgaben möglichst gut erfüllen. Da die Gewichte im oben präsentierten modularen Neuronalen Netz aber die physikalischen Parameter des zugrunde liegenden Feder-Masse Systems darstellen, ist es möglich, für Modelle unbekannter Materialen deren Deformationseigenschaften zu erlernen. In den folgenden Kapiteln wird ein dafür geeignetes Lernverfahren vorgestellt (siehe auch Anhang 0).

2.3.5.1 Das Lernverfahren

Der präsentierte Lernalgorithmus benutzt die Positionen von spezifischen Objektpunkten während diskreter Zeitschritte, um die benötigten physikalischen Parameter zu lernen oder zu optimieren. Die Lerndaten dieser Zeitreihendaten können beispielsweise von einem zeitabhängigen optischen Messverfahren für ein Objekt unter Einfluss externer Kräfte stammen.

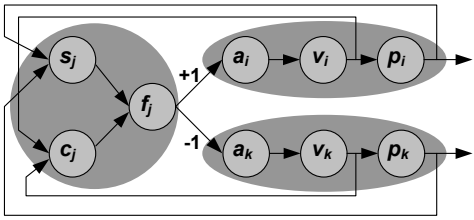

Abbildung 2.6*: Zwei Knoten (i und k), die durch eine Feder (j) verknüpft sind (nach* [136]*).*

Das Ziel dieses Lernverfahrens ist die Ableitung der Parameter des Feder-Masse Systems, die Massen der Knoten und die Federkonstanten c und d (siehe Gleichung (2.2)). Daher ist die Aufgabe des Algorithmus während des Lernens die Modifikation der Gewichte, die durch diese Konstanten definiert sind.

Um die Beschreibung des Algorithmus zu vereinfachen, wird im Folgenden ein einfaches Neuronales Netz benutzt, das aus zwei Knoten besteht, die durch eine Feder verknüpft sind (Abbildung *2.6*).

Die zwei modularen Neuronalen Netze der Knoten bestehen aus den drei Neuronen für die Berechnung der neuen Position p des Knotens basierend auf seiner aktuellen Position und seiner Geschwindigkeit, der neuen Geschwindigkeit v, basierend auf seiner aktuellen Geschwindigkeit und seiner Beschleunigung und der neuen Beschleunigung a, basierend auf den Kräften der verbundenen Federn und der externen Kräfte.

Mit der Schrittweite t_c für die Simulation sind diese Neuronen definiert durch:

$$\vec{o}_p(t) = \vec{a}_p(n\vec{e}t_p(t)) = t_c\vec{o}_v(t) + \vec{o}_p(t-1) \tag{2.23}$$

$$\vec{o}_v(t) = \vec{a}_v(n\vec{e}t_v(t)) = t_c\vec{o}_a(t) + \vec{o}_v(t-1) \tag{2.24}$$

$$\vec{o}_a(t) = \vec{a}_a(n\vec{e}t_a(t)) = w_m(\sum_i w_i\vec{o}_{f_1}(t) + e\vec{x}t_f(t)) \tag{2.25}$$

wobei $\vec{o}_{f_1}$ die Kraft der verbundenen Feder i ist. w_i definiert die Richtung der Kraft (Abbildung *2.6*), $w_m := m^{-1}$ und m ist die Masse des Knotens und $e\vec{x}t_f$ ist eine externe Kraft, die auf den Knoten einwirkt.

Das modulare Neuronale Netz der Feder besteht aus drei Neuronen: Das Positionskraftneuron berechnet die statische Federkraft f_s basierend auf den Positionen der verbundenen Knoten, das Viskositätsneuron berechnet die Dämpfungskraft f_d basierend auf den relativen Geschwindigkeit der verbundenen Knoten und dem Kraftneuron, das diese Kräfte zu einer einzigen Ausgabekraft f_f der Feder kombiniert (siehe auch Abbildung *2.4*). Diese Neuronen sind definiert als:

$$\vec{o}_f(t) = \vec{f}_f(n\vec{e}t_f(t)) = c_f\vec{o}_c(t) - d_f\vec{o}_s(t) \tag{2.26}$$

$$\vec{o}_s(t) = \vec{f}_s(n\vec{e}t_s(t)) = \vec{f}_s(\vec{o}_{p_1}(t) - \vec{o}_{p_2}(t)) \tag{2.27}$$

$$\vec{o}_d(t) = \vec{f}_d(n\vec{e}t_d(t)) = \vec{f}_d(\vec{o}_{v_1}(t) - \vec{o}_{v_2}(t)) \tag{2.28}$$

wobei c_f die physikalische Federkonstante, d_f die Viskositätskonstante und f_c, f_d die Feder- und Viskositätsfunktionen der Feder angeben.

Der präsentierte Lernalgorithmus führt nach der Propagation jedes Zeitschritts einen einzelnen Backpropagation Schritt (zeitlich) aus und modifiziert die Gewichte im Gegensatz zu dem BPTT Algorithmus sofort. Um eine große Abweichung von den benötigten Positionen zu vermeiden, wird eine teacher forcing Strategie benutzt [215]: Nach einer festgesetzten Anzahl von Zeitschritten werden die Ausgaben $\vec{o}_p(t)$ (beziehungsweise die Aktivierungen) der Positionsneuronen auf die benötigten Werte $\vec{p}(t)$ gesetzt.

Der Lernalgorithmus benutzt eine ähnliche Fehlerfunktion wie sie in den Gleichungen (2.6) und (2.7) definiert ist, mit der Ausnahme, das nur die Positionsneuronen für die

Kalkulation des Fehlers berücksichtigt werden, da die Geschwindigkeit, Beschleunigung oder die Federkräfte üblicherweise für das Training nicht verfügbar sind. Daher erhält man für den totalen Fehler $\vec{E}$:

$$\vec{E} = \sum_{t=0}^{T} \vec{E}(t) = \sum_{t=0}^{T} \left[\frac{1}{2} \sum_{p} (\vec{E}_p(t))^2 \right] \tag{2.29}$$

mit

$$\vec{E}_p(t) = \begin{cases} \vec{p}_p(t) - \vec{o}_p(t) & \text{wenn Knoten } p \text{ eine gewünschte} \\ & \text{Ausgabe } \vec{p}_p \text{ zur Zeit } t \text{ hat} \\ 0 & \text{sonst} \end{cases} \tag{2.30}$$

wobei $\vec{o}_p(t)$ die Position des Knotens *p* in dem physikalischen Modell ist und $\vec{p}_p(t)$ die Position des Knotens *p* im Zeitschritt *t*. Mittels des Gradientenabstiegsverfahren sind die Fehlerraten für die entsprechenden Neuronen wie folgt definiert (siehe auch Gleichungen (2.9) und (2.10)):

Sei $\vec{p}(t)$ die gewünschte Ausgabe zur Zeit *t* des Positionsneurons *p*, dann ist die Fehlerrate $\vec{\delta}_p(t)$ definiert als:

$$\vec{\delta}_p(t) = -\frac{\partial \vec{E}(t)}{\partial n\vec{e}t_p(t)} = \vec{p}(t) - \vec{o}_p(t) \tag{2.31}$$

Für die Fehlerraten des Geschwindigkeitsneurons *v* und des Beschleunigungsneurons *a* erhält man:

$$\vec{\delta}_v(t) = -\frac{\partial \vec{E}(t)}{\partial n\vec{e}t_v(t)} = \vec{\delta}_p(t) t_c = (\vec{p}(t) - \vec{o}_p(t)) t_c \tag{2.32}$$

$$\vec{\delta}_a(t) = -\frac{\partial \vec{E}(t)}{\partial n\vec{e}t_a(t)} = \vec{\delta}_v(t) t_c = (\vec{p}(t) - \vec{o}_p(t)) t_c^2 \tag{2.33}$$

Mit Gleichung (2.33) kann der Gewichtsgradient w_m definiert werden als:

$$\Delta w_m(t) = \eta \delta_a(t) \vec{o}_s(t) = \eta (\vec{p}(t) - \vec{o}_p(t)) \;\; t_c^2 \vec{o}_s(t) \tag{2.34}$$

wobei η die Lernrate beschreibt. Da t_c eine Konstante ist, folgt

$$\Delta w_m^*(t) = \eta_m (\vec{p}(t) - \vec{o}_p(t)) \vec{o}_f(t) \; . \tag{2.35}$$

mit der modifizierten Lernrate η_m. Die Fehlerraten für die Anpassung der Federkonstanten können auf die gleiche Weise erhalten werden:

$$\vec{\delta}_f(t) = -\frac{\partial \vec{E}(t)}{\partial n\vec{e}t_f(t)} = -\frac{\partial \vec{E}(t)}{\partial \vec{o}_f(t)} \frac{\partial \vec{o}_f(t)}{\partial n\vec{e}t_f(t)} \tag{2.36}$$

Mit Gleichung (2.25) erhält man

$$\frac{\partial \vec{E}(t)}{\partial \vec{o}_f(t)} = \sum_a \frac{\partial \vec{E}(t)}{\partial n\vec{e}t_a(t)} \frac{\partial n\vec{e}t_a(t)}{\partial \vec{o}_f(t)} = -\sum_a w_a \vec{\delta}_a(t) \tag{2.37}$$

Daher ist $\vec{\delta}_f(t)$ definiert als

$$\vec{\delta}_f(t) = \sum_a w_a \vec{\delta}_a(t) \tag{2.38}$$

Damit können die Gewichtsgradienten für die Konstanten c_f und d_f der Feder definiert werden als

$$\Delta w_c(t) = \eta \vec{\delta}_f(t) \vec{o}_s(t) \tag{2.39}$$

$$\Delta w_d(t) = -\eta \vec{\delta}_f(t) \vec{o}_d(t) \tag{2.40}$$

wobei η wieder die Lernrate angibt. Die Fehlerraten für die Gesamtkraftneuronen sind definiert durch

$$\vec{\delta}_s(t) = f_s'(net_s(t)) \vec{\delta}_f(t) w_1 = f_s'(net_s(t)) \sum_a \vec{\delta}_a(t) \tag{2.41}$$

Die Fehlerraten für die Viskositätsneuronen erhält man auf die gleiche Weise. Daher gilt

$$\vec{\delta}_d(t) = f_d'(net_d(t)) \sum_a \vec{\delta}_a(t) \tag{2.42}$$

Schließlich müssen die Fehlerraten für die Positionsneuronen abgeleitet werden.

$$\vec{\delta}_p(t) = -\frac{\partial \vec{E}(t)}{\partial n\vec{e}t_p(t)} = -\frac{\partial \vec{E}(t)}{\partial \vec{o}_p(t)} \frac{\partial \vec{o}_p(t)}{\partial n\vec{e}t_p(t)} \tag{2.43}$$

mit

$$\frac{\partial \vec{E}(t)}{\partial \vec{o}_p(t)} = \sum_{i \in \{s,c\}} \frac{\partial \vec{E}(t)}{\partial n\vec{e}t_i(t)} \frac{\partial n\vec{e}t_i(t)}{\partial \vec{o}_p(t)} = -\sum_s \vec{\delta}_s(t) - \sum_d \vec{\delta}_d(t) \tag{2.44}$$

$\vec{\delta}_f(t)$ ergibt sich zu

$$\vec{\delta}_f(t) = \sum_s \vec{\delta}_s(t) + \sum_d \vec{\delta}_d(t) \tag{2.45}$$

Der Lernalgorithmus passt die Gewichte jedes Knotens unter Benutzung der Gleichungen (2.35), (2.39) und (2.40) an. Wenn keine Zeitreihendaten $\vec{p}(t)$ für einen (inneren) Knoten verfügbar sind, wird der Fehler von den Neuronen der Federn unter Benutzung der Gleichungen (2.42) und (2.35) zurückpropagiert. Danach wir die Fehlerrate des Positionsneurons (Gleichung (2.31)) ersetzt durch

$$\vec{\delta}_p^*(t) = \sum_s \vec{\delta}_s(t) + \sum_d \vec{\delta}_d(t) \tag{2.46}$$

Wenn das zu simulierende Objekt aus homogenem Material besteht und die Netzwerkstruktur geeignet definiert worden ist, können alle Module die gleichen Gewichte benutzen. Daher können die Parameter (*c*, *m* und *s*) gemeinsam für alle zugehörigen Gewichte erlernt werden, womit die Performanz und die Stabilität des Lernalgorithmus stark verbessert werden kann. Durch leichte Modifikation des Algorithmus können sogar Bereiche gleichen Materials gemeinsam bearbeitet werden. Darüber hinaus kann in vielen Fällen die Masse m_j der Knoten durch Teilen der Gesamtmasse durch die Gesamtanzahl der Knoten vordefiniert werden.

2.3.5.2 Erzeugung von Lerndaten durch ein Feder-Masse System eines Würfels

Im Gegensatz zu dem im Anhang 0 (S 155) vorgestellten Beispiel für das Lernen von Parametern medizinischer Gewebe wird im Folgenden, um die generelle Einsatzfähigkeit des Lernalgorithmus zu überprüfen, ein sehr einfaches aber typisches Deformationsbeispiel benutzt: Ein Objekt wird über eine Zeitperiode unter Einfluss externer Kräfte deformiert [135]. Als Objekt wird ein einfacher Würfel, bestehend aus 27 Knoten und 70 Federn (siehe Abbildung *2.7*), mit vordefinierten Parametern gewählt. Die notwendigen Lerndaten für die Deformation des Objektes, die Positionsveränderungen über eine definierte Zeitperiode, werden über die Simulation künstlich erzeugt. Ziel ist es, mittels dieser Lerndaten die anfängliche Parameterverteilung neu zu erlernen.

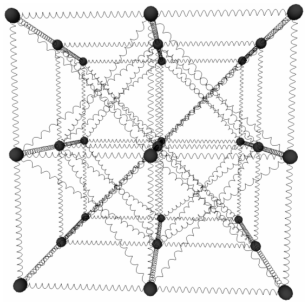

Abbildung 2.7*: Ein Feder-Masse Modell eines Würfels als Lernbeispiel (aus* [135]*)*

Während der Simulation sind die unteren neun Knoten des Würfels fixiert und eine konstante Kraft wird an den Knoten im Zentrum der oberen Schicht angelegt. Die Deformation des Objektes wird über eine Zeitperiode von 250 Zeitschritten simuliert. Nach dieser Periode haben die Knoten des Feder-Masse Systems ihre endgültige Position, also ein Energieminimum erreicht. Die Positionen der Knoten an den diskreten Zeitschritten werden als Lerndaten gespeichert.

Nach dieser Erzeugung der Testdaten wird der eigentliche Lernprozess gestartet. Für das erste Beispiel werden die Konstanten für die Masse der Knoten und die Feder- und Viskositätskonstanten zufällig aus einem Intervall definiert durch $[0{,}2x;\ 5x]$ um die ursprünglichen Werte x gewählt. Der Algorithmus wird initialisiert mit einer Lernrate $\eta = 0{,}01$ für die Federkonstanten c_i und d_i und $\eta_m = 0{,}1$ für die Massen m_j der Knoten. Eine teacher forcing Begrenzung auf zehn Iterationsschritte wird festgesetzt [215]. Nach 100 Lernzyklen, jeder Zyklus ist eine komplette Propagation nach der Zeit, wird der Lernalgorithmus mit einem Fehler von $E = 1{,}21$ beendet. Der Fehler nach jedem Lernzyklus ist aus Abbildung *2.8* (sample 1) ersichtlich. Durch die geringe Anzahl von unterschiedlichen Lerndaten ist der Algorithmus nicht in der Lage die vorherigen Parameter des Modells wieder zu ermitteln. Die resultierenden Konstanten c_i, s_i und m_i sind noch um ihre exakten, vorherigen Werte verteilt, was in untrainierten Situationen zu einem ungewollten Verhalten des Modells führen kann.

Ein zweites Lernbeispiel mit dem Würfelmodell wird wie oben initialisiert, nur dass eine Lernrate von $\eta = 0{,}1$ für die Federkonstanten benutzt wird. Außerdem werden die Massen vordefiniert und während des Lernens nicht modifiziert. Nach dem abermaligen Abbruch des Algorithmus nach 100 Lernzyklen kann ein Fehler von $E = 0{,}033$ erreicht werden (Abbildung *2.8*, sample 2). Allerdings konnte der Lernalgorithmus wiederum die vorherigen Parameter des Modells nicht ermitteln.

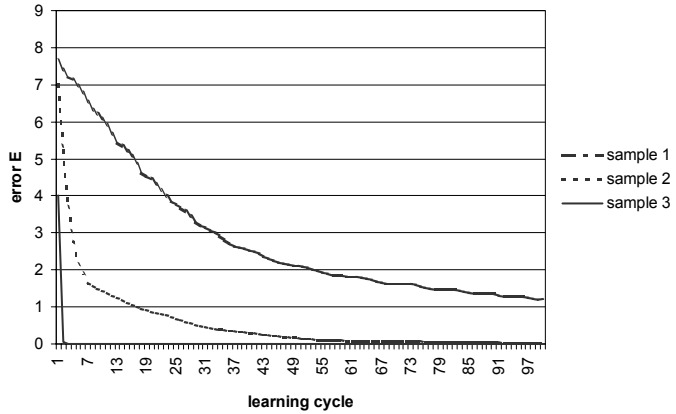

Abbildung 2.8*: Der Fehler E in Abhängigkeit von der Anzahl der Lernzyklen (learning cycle) für das Feder-Masse System des Würfels (aus* [135]*).*

In einem dritten Lernbeispiel werden die Konstanten *c* und *s* gemeinsam für alle Federn definiert. Die Massen sind vordefiniert und ein Wert wird zu den Konstanten addiert, um eine zufällige Übereinstimmung mit den initialen Parametern zu vermeiden. Nach dem Beenden des Algorithmus nach 100 Lernzyklen kann ein Fehler von $E = 0{,}138$ erreicht werden (Abbildung *2.8*, sample 3), was leicht höher als in dem letzten Beispiel ist. In diesem Fall ist der Lernalgorithmus aber in der Lage, die ursprünglichen Parameter des Modells mit einem Fehler kleiner als 0,1% zu ermitteln.

2.3.6 Fuzzy Systeme

In den letzten Abschnitten wurde ein Verfahren vorgestellt, mit dem man ein beliebiges Feder-Masse System als Neuronales Netz interpretieren kann. Expertenwissen, das meist in Form umgangssprachlicher Regeln und vagen oder unscharfen Einschätzungen vorliegt, kann dabei nicht berücksichtigt werden. Mit den im Folgenden beschriebenen Fuzzy Systemen kann dieses Fachwissen für die Deformationssimulation genutzt werden. Durch die Kombination mit den Neuronalen Netzen können sowohl die Vorteile der Fuzzy Systeme als auch die der Neuronalen Netze genutzt werden. Da beide Methoden in Form eines *kooperativen Neuro-Fuzzy Systems* getrennt genutzt werden, wird im Folgenden das Fuzzy System unabhängig von den Neuronalen Netzen beschrieben.

Um in komplexen Entscheidungssituationen eine sachgerechte Wahl zu treffen, ist es üblich, ein konkretes Problem in einem *Entscheidungsmodell* abzubilden. Eine generelle Schwierigkeit bei der Beschreibung von Modellen liegt darin, dass oftmals erhebliche Idealisierungen notwendig sind, um von einem realen Problem zu einem geeigneten mathematischen Modell zu gelangen. Zwar können durch Fortschritte in der Computertechnologie immer komplexere Modelle bearbeitet werden, aber deren Anwendung bleibt nach wie vor ein Problem, wenn die Vielfalt der Daten und die Form der Ergebnisse für den Benutzer unüberschaubar ist. Daher ist es teilweise notwendig, komplexe Modelle zu vereinfachen, ohne dabei die Aussagekraft der Ergebnisse zu verringern.

Eine Methode, komplexe Systeme zu vereinfachen, besteht in der kalkulierbaren Tolerierung eines Anteils an Impräzision, Vagheit und Unsicherheit bereits während der Modellierungsphase. Obwohl diese Systeme dann nicht perfekt sind, sind sie doch vielfach in der Lage das Problem geeignet zu lösen. Mit Fuzzy Systemen versucht man, den Vorteil einer solchen Komplexitätsreduzierung gegenüber anderen Systemen auszunutzen, die aufgrund der generellen Verwendung von präzisen Informationen wesentlich schwieriger zu handhaben sind.

Auch konventionelle mathematische Methoden bieten die Möglichkeit, Impräzision, Unsicherheit und Vagheit zu berücksichtigen.

Im Folgenden wird die Verwendung der Begriffe geklärt, sowie die bei konventionellen Methoden auftretenden Schwierigkeiten beschrieben (siehe [96]).

Impräzision von Informationen lässt sich darauf zurückführen, dass es oftmals unmöglich ist, bei einer Beobachtung oder Messung einen beliebigen Grad an Genauigkeit zu erreichen. Impräzision wird üblicherweise durch Intervalle beschrieben, etwa [10, 15] für den Durchmesser einer anatomischen Struktur in Millimetern. Dabei tritt jedoch die Schwierigkeit auf, dass zwar ein Durchmesser von 10 mm für möglich gehalten wird, jedoch ein Durchmesser von 9,9 mm bereits vollständig ausgeschlossen wird.

Unsicherheit dagegen ist mit Zufallsmechanismen verbunden, die üblicherweise durch stochastische Modelle ausgedrückt werden. Schwierig wird jedoch die Berücksichtigung von Unsicherheit aufgrund subjektiver Einschätzungen.

Vagheit schließlich wird im täglichen Leben bewusst in Kauf genommen, um dadurch die Kommunikation zu erleichtern und Informationen auf den für gezieltes Handeln in einer bestimmten Situation wesentlichen Anteil zu beschränken. So ist beispielsweise die Charakterisierung der Geschwindigkeit eines Autos durch das Adjektiv „schnell“ vollkommen ausreichend, um die Notwendigkeit eines Ausweichmanövers zu bestimmen. Die Grenze zwischen „schnell“ und „nicht schnell“ bezüglich einer gegebenen Geschwindigkeit eines Autos ist dabei nicht immer eindeutig zu ziehen, sondern es existiert vielmehr ein fließender Übergang zwischen den vagen Konzepten „schnell“ und „nicht schnell“. Diese Nuancierungen werden im Folgenden durch Fuzzy Mengen ausgedrückt, da insbesondere Vagheit mit konventionellen Methoden nur sehr schwierig zu beschreiben ist.

Nach der klassischen Definition einer Menge A gilt für ein beliebiges Objekt a entweder $a \in \mathrm{A}$ oder $a \notin \mathrm{A}$. Wie bereits erwähnt, bereitet dieser scharfe Übergang bei der Anwendung auf reale Problemstellungen oft große Schwierigkeiten.

Daraus entstand die Idee, jedem Element x einer Grundmenge X eine reelle Zahl $\mu(x)$ zuzuordnen, die den Grad der Zugehörigkeit zu der Menge X ausdrückt [226]. Dabei ist es üblich, den Wertebereich der *Bewertungsfunktion* oder *charakteristischen Funktion* μ auf das abgeschlossene Intervall [0,1] zu beschränken und den Funktionswert 0 den Objekten zuzuordnen, die nach Ansicht des Urteilenden die gewünschte Eigenschaft mit Sicherheit nicht aufweisen und den Wert 1 denjenigen Objekten, die die Eigenschaft mit Sicherheit aufweisen.

Anlehnend an [96] werden hier Fuzzy Mengen mit der sie charakterisierenden Funktion identifiziert.

Definition 2.2

Eine *Fuzzy Menge* μ von X ist eine Funktion von der Grundmenge bzw. der Domain X in das Einheitsintervall, d.h. $\mu : X \rightarrow [0,1]$.

Da Fuzzy Mengen meist benutzt werden um linguistische Ausdrücke oder vage Konzepte wie „ungefähr“, „langsam“ oder „klein“ zu beschreiben, werden sie üblicherweise mit einem Namen versehen. Im Folgenden werden Fuzzy Mengen daher auch durch ihren zugewiesenen Namen bezeichnet (z.B. „ungefähr“, „langsam“ oder „klein“).

Die Benutzung von Fuzzy Mengen zur formalen Darstellung vager Daten geschieht im allgemeinen auf rein intuitiver Basis, da in vielen Anwendungen kein Modell zugrunde gelegt wird, das den Zugehörigkeitsgraden eine eindeutige Interpretation zuordnet. Weil für den Einsatz in der Deformationssimulation real existierende physikalische Eigenschaften beschrieben werden, ist es sinnvoll, dass mindestens ein Zugehörigkeitswert der beschreibenden Fuzzy Menge den Wert 1 hat. Eine Fuzzy Menge wird dann als Charakterisierung der vagen Beobachtung einer existierenden, aber nicht direkt zugänglichen reellwertigen Größe benutzt.

Wenn, wie bei der Modellierung physikalischer Parameter, die Anzahl der Elemente der Grundmenge sehr groß oder unendlich ist, so benutzt man bei der Modellierung zumeist einfache Funktionsformen wie stückweise lineare Funktionen, bei denen festgelegte Punkte durch Geradenstücke verbunden werden oder Funktionen, die durch wenige Parameter beschrieben werden können [175]. In vielen Anwendungen werden daher trapezförmige Fuzzy Mengen verwendet, mit denen eine einfache Verarbeitung und Berechnung per Computer möglich ist [54].

$$\mu_{a,b,c,d}(x) = \begin{cases} \dfrac{x-a}{b-a} & \text{falls } x \in [a,b], \\ \dfrac{d-x}{d-c} & \text{falls } x \in [c,d], \\ 1 & \text{falls } x \in (b,c), \\ 0 & \text{ansonsten} \end{cases} \tag{2.47}$$

Trapezförmige Fuzzy Mengen sind definiert durch (2.47), wobei $a < b \leq c < d$ gilt.

Fuzzy Mengen werden im Allgemeinen nicht nur für die Definition eines einzelnen vagen Konzepts definiert, sondern für alle Konzepte, die im spezifischen Kontext ausrei-

chen, die Domain X abzudecken. Der Ausdruck Fuzzy Partition wird benutzt, um eine solche Menge von Fuzzy Mengen zu beschreiben. Beispiele für solche Fuzzy Partitionen finden sich im Anhang B und 0.

Wenn Relationen zwischen verschiedenen Domains beschreiben werden sollen, können linguistische Regeln oder *Fuzzy Regeln* verwendet werden. Eine konjunktive Fuzzy Regel hat die Form

$$R_r : \text{if } x_1 \text{ is } \mu_r^{(1)} \text{ and } \ldots \text{ and } x_n \text{ is } \mu_r^{(n)} \text{ then } y_1 \text{ is } \nu_r^{(1)} \text{ and } \ldots \text{ and } y_m \text{ is } \nu_r^{(m)},$$

wobei n die Anzahl der Eingabedomains, m die Anzahl der Ausgabedomains, $\mu_r^{(i)}$ und $\nu_r^{(j)}$ Fuzzy Mengen und x_i und y_j die Ein- und Ausgabewerte angeben.

Eine Menge von Fuzzy Regeln kombiniert mit einen Inferenzmechanismus für die Berechnung der Fuzzy Regel, z.B. einem Berechnungsschema, wie die Regeln ausgewertet und kombiniert werden müssen, um für jeden Eingabewert einen Ausgabewert zu berechnen, nennt man *Fuzzy System* [96].

Für den Einsatz zur Parameterinitialisierung im Rahmen der Chirurgiesimulation wurde der Ansatz des Fuzzy Systems von Mamdani gewählt, welcher erstmals in [117] beschrieben wurde. Weil diese Art des Fuzzy Systems zur Regelung eingesetzt wurde, wird das System auch *Mamdani-Regler* genannt. Der Mamdani-Regler approximiert eine Funktion durch Interpolation von vagen Kontrollpunkten im Datenraum. Damit kann die Berechnung einer Regelbasis auch als Interpolationsstrategie in einer vagen Umgebung interpretiert werden.

Der Mamdani-Regler benutzt die t-Normen und t-Conormen (siehe Anhang B):

$$\top\{a,b\} = \min\{a,b\} \text{ und } \perp\{a,b\} = \max\{a,b\} \tag{2.48}$$

und berechnet somit die Inferenz

$$\mu^{output}_{x_1,\ldots,x_n} : Y \to [0,1], \quad y \mapsto \max_{r\in\{1,\ldots,k\}} \{\min\{\mu^{(1)}_{i_{1r}}(x_i), \ldots, \mu^{(n)}_{i_{nr}}(x_n), \nu_{i_r}(y)\}\}. \tag{2.49}$$

Abbildung *2.9* zeigt den Inferenzprozess an einem Beispiel. Nach Definition der Fuzzy Partitionen und Fuzzy Regeln wird der Ausgabewert berechnet, indem zunächst der Erfüllungsgrad α_r der Prämisse jeder einzelnen Regel R_r zu den aktuellen Eingabewerten $x_1, \ldots, x_n$ ermittelt wird. Dazu werden zunächst die Zugehörigkeitsgrade der Eingabewerte x_i zu den Fuzzy Mengen der assoziierten Terme bestimmt. Anschließend werden diese Werte konjunktiv mit der oben beschriebenen t-Norm verknüpft.

Als Ausgabe jeder Regel R_r erhält man anschließend die Fuzzy Menge, die durch 'Abschneiden' der Fuzzy Menge μ_{i_r} bei α_r entsteht:

$$\mu^{output(R_r)}_{x_1,\ldots,x_n} : Y \to [0,1], \quad y \mapsto \min\{\mu^{(1)}_{i_{1,r}}(x_i), \ldots, \mu^{(n)}_{i_{n,r}}(x_n), \nu_{i_r}(y)\}.$$

Falls die Eingabewerte die Prämisse vollständig erfüllen, liefert die Regel R_r genau die Fuzzy Menge ν_r der Konklusion.

Die aus der Auswertung der einzelnen Regeln entstandenen Fuzzy Mengen werden anschließend mittels Maximumbildung (t-Conorm) zu einer Fuzzy Menge vereinigt und bilden das in Gleichung (2.48) beschriebene Ergebnis.

Die Entscheidungslogik liefert somit keinen scharfen Ausgabewert, sondern beschreibt ihn mittels einer Fuzzy Menge. Ein scharfer Ausgabewert wird dann meist durch die

Schwerpunktmethode (Center-of-Area) berechnet, bei der der Wert verwendet wird, der unter dem Schwerpunkt der durch die Fuzzy Menge definierten Fläche liegt.

Das im aktuellen Abschnitt vorgestellte Fuzzy System wurde in dem im nächsten Abschnitt beschriebenen Tool Elastodynamic Shape Modeler verwendet, um die Parameterinitialisierung für die Virtuelle Chirurgie zu ermöglichen.

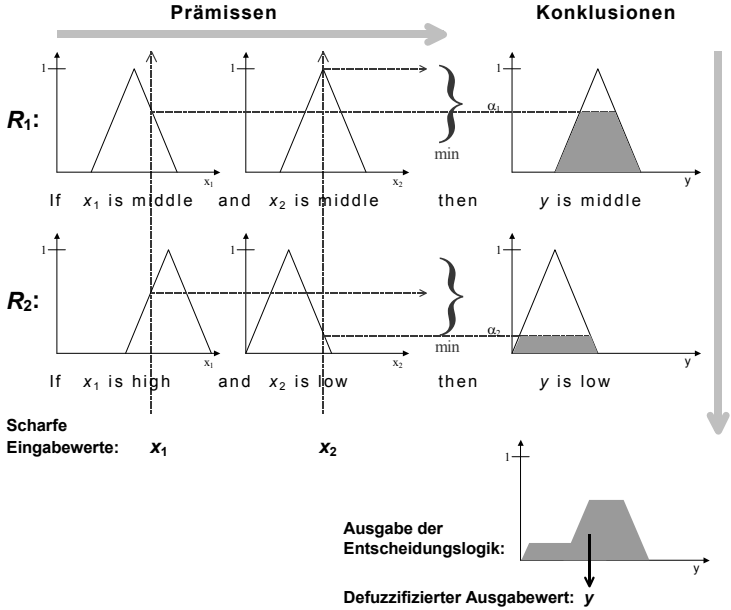

Abbildung 2.9*: Auswertung von zwei Regeln durch einen Mamdani-Regler.*

2.3.7 Parameterinitialisierung mit dem Elastodynamic Shape Modeler

Um den Modellierungsprozess von Feder-Masse Systemen für die Chirurgiesimulation zu vereinfachen, wurde der Elastodynamic Shape Modeler entwickelt [160]. Mit dieser Software ist es möglich, Expertenwissen in Form von linguistischen Regeln für die Beschreibung des Deformationsverhaltens von medizinischem Gewebe zu verwenden. In diesem Abschnitt werden die Erzeugung von deformierbaren Gewebemodellen und die Komponenten des Elastodynamic Shape Modeler beschrieben.

2.3.7.1 Die Erzeugung deformierbarer Gewebemodelle

Um deformierbare Modelle von Gewebe oder Organen zu erzeugen, werden geometrische Beschreibungen als Oberflächen- oder Volumenmodelle verwendet. Diese Modelle können entweder durch ein Tool zum dreidimensionalen Modellieren (z.B. Maya, Alias|Wavefront; Houdini, Side Effects Software ; 3DStudio,Autodesk) oder durch Segmen-

tierung von Daten bildgebender Verfahren in der Medizin entstanden sein (vgl. Kapitel 4.2) und direkt in den Elastodynamic Shape Modeler importiert werden, der die statische grafische Beschreibung in ein Deformationsmodell umwandelt. Dabei werden Kanten in Federn und Eckpunkte in Masseknoten umgewandelt. Dafür ist eine Datenstruktur entwickelt worden, die vollständig in die 3D Grafikbibliothek *Open Inventor* (Silicon Graphics Inc., Mountain View, USA) eingebunden ist [209]. Open Inventor ist für die gängigsten Computer Plattformen erhältlich und kann somit für viele Anwendungen genutzt werden.

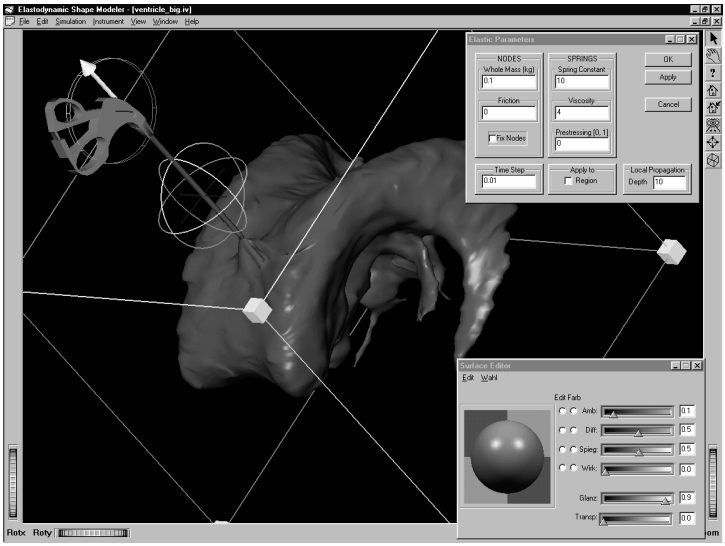

Abbildung 2.10*: Elastodynamic Shape Modeler: Deformation eines Oberflächenmodells. Rechts oben: Anzeige der Deformationsparameter für das gesamte Modell. Rechts unten: Editor zum Ändern der Oberflächendarstellung (aus* [160]*).*

Als Beispiel ist in Abbildung *2.10* das segmentierte Modell eines Ventrikels – das System flüssigkeitsgefüllter Kammern im Gehirn - importiert worden. In einem echten Simulator würde das Instrument innerhalb des Ventrikels und die Sicht transendoskopisch sein (vgl. Kapitel 5.2). Mit dem Elastodynamic Shape Modeler sollen allerdings im Gegensatz zu einem Chirurgiesimulator die Deformationseigenschaften eingestellt werden, wofür die Ansicht – innen oder außen – nicht relevant ist. Der Elastodynamic Shape Modeler ist selbst kein Simulator, sondern ein Werkzeug zum Erstellen von Deformationsmodellen, die in einem (Chirurgie-)simulator benutzt werden können. Dennoch sind grundsätzliche Techniken eines Simulators, wie die Kollisionserkennung, die Deformationssimulation und ein virtuelles Instrument implementiert. Mit dem virtuellen Instrument, das entweder durch eine Computermaus gesteuert werden kann, ist es möglich, das Modell zu deformieren. Zusätzlich kann das Instrument durch ein Force-Feedback Ein-

gabegerät (Laparoscopic Impulse Engine, Immersion Corp., San Jose, USA [152]) gesteuert werden, das es ermöglicht, Widerstände bei der Deformation zu erfühlen.

Der Elastodynamic Shape Modeler definiert durch die Initialisierung der notwendigen Parameter (Viskosität, Federkonstanten und Massen) anfänglich ein Standardverhalten für die importierten Deformationsmodelle. Zusätzlich ist die Möglichkeit zur Beschreibung und Auswertung einer Fuzzy Regelbasis implementiert worden, um die Parameter mittels Fuzzy Regeln einzustellen.

2.3.7.2 Die Komponenten des Elastodynamic Shape Modelers

Alle Komponenten des Elastodynamic Shape Modelers sind innerhalb einer grafischen Benutzerschnittstelle (Graphical User Interface, GUI) unter Microsoft Windows (Microsoft, Redmond, USA) implementiert worden.

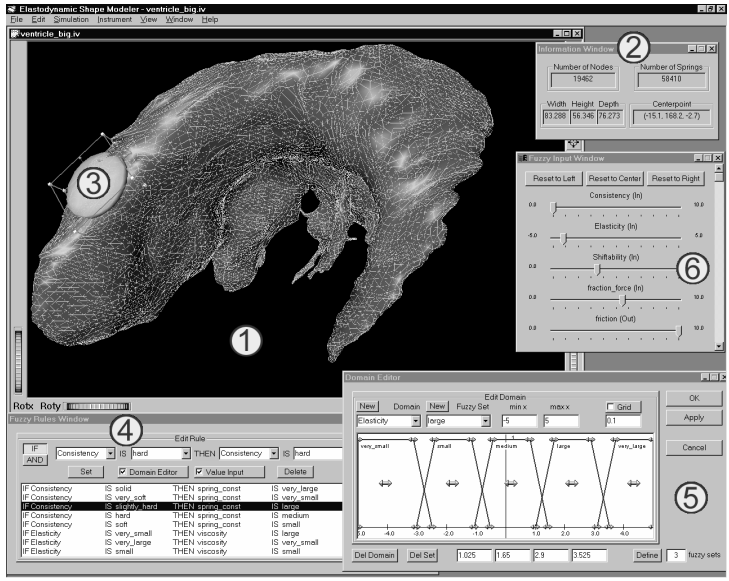

Abbildung 2.11*: Das Fuzzy System des Elastodynamic Shape Modelers: Das deformierbare Modell eines Ventrikels wird im Objektfenster (1) angezeigt. Das Informationsfenster (2) zeigt globale Daten (z.B. Anzahl der Knoten und Federn) über das Modell. Ein frei definierbares Ellipsoid (3) wird benutzt, um die Region für die Parameteränderung zu definieren. Verschiedene Eingabefenster (4)-(6) können benutzt werden, um das Fuzzy System zu definieren (aus* [160]*).*

Im Elastodynamic Shape Modeler sind folgende Hauptkomponenten implementiert (Abbildung *2.11*):

(1) Das Objektfenster zeigt das Deformationsmodell. Für die Visualisierung des zugrunde liegenden Feder-Masse Systems können verschiedene Visualisierungsformen gewählt werden. Hier sind die Federn in gelb eingezeichnet.

(2) Das Informationsfenster zeigt globale Daten über das Modell (die Anzahl der Knoten und Federn, die Größe und das Zentrum des Modells). Dies ist nützlich, um das Modell auf seine wirkliche (anatomische) Größe anzupassen.

(3) Unter Benutzung eines frei definierbaren Ellipsoids kann die Region eingestellt werden, in der die Parameter geändert werden sollen. Nur die Federn und Knoten innerhalb dieser Region werden neu definiert, wodurch unterschiedliches Deformationsverhalten in unterschiedlichen Bereichen definiert werden kann.

(4) In diesem Fenster kann die Menge der Fuzzy Regeln (Regelbasis) eingegeben werden. In den meisten Fällen kann eine der vordefinierten Regelbasen, die eingeladen und abgespeichert werden können, ohne Veränderung benutzt werden.

(5) Mit dem Domain Editor können die Fuzzy Partitionen definiert werden. Als Fuzzy Mengen werden trapezförmige Funktionen benutzt (Gleichung (2.47)). Die vier Werte, die eine trapezförmige Fuzzy Menge definieren, können durch Verschieben mit der Maus oder durch direkte Eingabe des Werts eingestellt werden.

(6) Das Eingabefenster erlaubt die Definition des Deformationsverhaltens des Modells durch die Verwendung von Schiebereglern. Die Regler werden automatisch nach den im Fuzzy System verwendeten linguistischen Ausdrücken erstellt und nach Ein- und Ausgabewerten sortiert. Auch die Ausgabewerte können verändert werden, wenn bestimmte Deformationsparameter bereits bekannt sind.

Nach der Definition des Fuzzy Systems können in Zusammenarbeit mit einem medizinischen Experten die Schieberegler im Eingabefenster verändert und die Auswirkung direkt im Objektfenster überprüft werden. Damit ist eine schnelle und intuitive Definition der Deformationsparameter möglich. Beispiele für die Definition und Auswertung von Fuzzy Systemen im Bereich der gynäkologischen und neurochirurgischen Simulation finden sich im Anhang B und 0.

Außerdem ist im Elastodynamic Shape Modeler das im Kapitel 2.3.5 beschriebene Lernverfahren implementiert, um die mit dem Fuzzy-System initialisierten Parameter durch die Berücksichtigung von Lerndaten weiter zu verbessern.

2.4 Diskussion

Visueller Realismus ist nicht gleich physikalischer Realismus. Zum Beispiel kann ein Deformationsmodell nach einem höchst genauen physikalischen Modell erstellt werden - in Erwartung eines sehr hohen visuellen Realismus. Leider ist dies in der Chirurgiesimulation nur selten der Fall, da alle physikalischen Modelle eine große Anzahl von Parametern benötigen, die meist unbekannt sind [160]. Fehlender Realismus bei den Deformationen wird von vielen Wissenschaftlern primär mit der Entwicklung aufwändigerer Simulationsmodelle begegnet, meist durch den Übergang von einfachen FEM zu komplexeren, dreidimensionalen FEM. Obwohl dieser Ansatz bei vielen Anwendungen, wie beispielsweise der Deformation einfacher Körper aus homogenen Stoffen wie Gummi, zum Erfolg führt, sind falsche oder ungenaue Parameter bei der Chirurgiesimulation der ei-

gentliche Grund, dass Simulationen nicht realistisch wirken [139; 180]. Dies liegt an der Schwierigkeit, diese Parameter für Gewebe und Organe zu bestimmen [196]. Daher versuchen viele Forschungsgruppen Wege zu finden, die Gewebeeigenschaften auf invasive und nicht-invasive Weise zu ermitteln, nur teilweise mit Erfolg [34; 102; 196].

Im Kontext dieser Arbeit wird daher entgegen anderen Veröffentlichungen, die komplexere Modelle für die Chirurgiesimulation empfehlen (siehe u.a. [20; 30; 47; 50; 59; 64; 80; 90; 198]) oder Arbeiten, die versuchen, die notwendigen Parameter zu bestimmen (siehe [34; 35; 102; 103; 187; 196]), empfohlen, die Parameter anhand realer Deformationen zu erlernen oder durch umgangssprachliche Ausdrücke anzupassen. Dazu wurde die in diesem Kapitel vorgestellte Architektur auf der Basis von Neuronalen Netzen entwickelt, die in der Lage ist, Feder-Masse Systeme bzw. eindimensionale FEM zu simulieren. Auf diese Weise ist es möglich, Deformationsparameter des physikalischen Modells zu erlernen und gleichzeitig die Simulationsgeschwindigkeit durch die Verwendung von problemangepassten Propagationsalgorithmen zu erhöhen. Alle für Neuronale Netze entwickelten Konzepte können somit für die Deformationssimulation verwendet werden. Im Speziellen gilt dies auch für die Möglichkeit der massiv parallelen Berechnung des Neuronalen Netzes. Beispiele für das Erlernen von Parametern wurden mit einem künstlich erzeugten Datensatz und durch ein Ultraschallphantom erarbeitet (Anhang 0).

Darüber hinaus wurde ein Fuzzy System entwickelt, das in Verbindung mit dem Neuronalen Netz verwendet werden kann, um die Deformationsparameter mittels einfacher linguistischer Ausdrücke oder per Schieberegler zu initialisieren. Damit kann ein medizinischer Experte sein meist implizit vorhandenes Wissen für die Optimierung der visuellen und haptischen Deformationssimulation zur Verfügung stellen. Um die Definition der Parameter zu erleichtern, wurde das Werkzeug Elastodynamic Shape Modeler entwickelt, mit dem jede Änderung der Parameter durch das Fuzzy System direkt am Deformationsmodell mit einem Force-Feedback Gerät überprüft werden kann. Jede Änderung der Eingaben des Fuzzy Systems hat direkte Auswirkungen auf die Simulation, die somit haptisch und visuell getestet werden kann. Anwendungen in der Chirurgiesimulation haben gezeigt, dass es so möglich ist, innerhalb kurzer Zeit überzeugende Gewebemodelle zu erstellen.

Die Definition, Modellierung und Simulation der Feder-Masse Systeme löst jedoch nur einen Teil der Probleme bei der Erstellung von Chirurgiesimulatoren. Visueller Realismus entsteht im Wesentlichen auch durch die Darstellungsqualität an sich. Insbesondere bei der Verwendung realer Patientendaten ist die technisch machbare Leistungsgrenze heutiger Computersysteme aufgrund der gewaltigen Datenmengen, die interaktiv visualisiert werden müssen, schnell erreicht. Im folgenden Kapitel werden daher Konzepte zur direkten interaktiven und deformierbaren Visualisierung von dreidimensionalen digitalen medizinischen Datensätzen beschrieben.

3 VISUALISIERUNG MITTELS VOLUME-RENDERING

Heutzutage werden standardmäßig bildgebende Verfahren wie die Computertomographie (CT) oder Magnetresonanztomographie (MRT) verwendet, die in der Lage sind, einen digitalen dreidimensionalen Datensatz der Anatomie des Patienten zu generieren [195]. Vielfach werden diese Datensätze bereits zur Planung insbesondere von minimalinvasiven Operationen eingesetzt. Heutzutage wird die Planung in den meisten Fällen auch am Computer noch anhand von Schichtbildern durchgeführt. Nur wenige Systeme nutzen darüber hinaus die Möglichkeit einer dreidimensionalen Darstellung beispielsweise durch ein simuliertes Endoskop, weil eine interaktive dreidimensionale Darstellung der Datensätze nur sehr schwierig und mit hohem technischen Aufwand möglich ist. Bei konsequenter Unterstützung durch spezielle Computerhardware kann heutzutage auch diese Darstellung, die unter dem Namen *Volume-Rendering* [61; 87] bekannt ist, interaktiv durchgeführt werden [142]. Mittlerweile ist dies auch mit handelsüblichen Grafikkarten auf dem PC möglich [39; 92; 169; 191].

Die Anwendung des Volume-Rendering scheint vor allem in der Neurochirurgie [10; 11; 13; 15] besonders geeignet zu sein, um eine möglichst realistische Darstellung virtueller Endoskopien zu liefern, da dort operationsvorbereitend immer ein MRT-Datensatz erstellt wird. Während erste Prototypen von Operationssimulatoren noch mit einer statischen Volumendarstellung arbeiten [98], gehen neuere Ansätze in die Richtung einer dynamischen Volumenvisualisierung, in der eine direkte Interaktion mit den Volumendaten möglich ist [168; 212; 223].

Ist die lokale Interaktion mit den Volumendaten, z.B. das Entfernen einzelner Volumenelemente immerhin technisch möglich, so stellt die Volumendeformation mittels Volume-Rendering eine wesentlich höhere Anforderung an die verwendete Hardware und ist nur in Ansätzen echtzeitfähig [57]. Ein bestehendes Verfahren benutzt zur Deformation der Volumendaten so genannte *Ray-Detector-Felder* [100], bei denen nicht das eigentliche Volumen deformiert, sondern die Sehstrahlen durch Detektorfelder abgelenkt werden. Nachteil des Verfahrens ist, dass die Deformation des Volumens keinerlei physikalische Eigenschaften simuliert und das Verfahren bei komplexer Verformung sehr langsam ist, da jede zusätzliche Veränderung durch ein weiteres Detektorfeld abgebildet werden muss.

Für den Einsatz in einem Simulator für minimal-invasive Neurochirurgie wurde daher im Rahmen der vorliegenden Arbeit eine Methode entwickelt, mit der physikalisch basierte Deformationen von Volumendaten in Echtzeit per Volume-Rendering visualisiert werden können [165]. Möglich wird dieses so genannte *dynamische Volume-Rendering* durch die konsequente Ausnutzung der Grafikhardware. Mittlerweile sind Weiterentwicklungen des grundlegenden Ansatzes durchgeführt und auf verschiedenen Hardwareplattformen, insbesondere auch auf PC-Basis, implementiert worden [39; 169; 191; 210].

Die ursprüngliche Methode des dynamischen Volume-Rendering wird, nach einer Einführung in die Darstellung dreidimensionaler Datensätze bildgebender Verfahren und in das hardwareunterstützte direkte Volume-Rendering, in Kapitel 3.3 beschrieben.

3.1 Darstellung dreidimensionaler Datensätze bildgebender Verfahren

Bildgebende Verfahren sind seit Konrad Röntgens Zeiten nicht mehr aus der Medizin wegzudenken. Zwar kommt das klassische Röntgenverfahren im Wesentlichen ohne den Einsatz von Rechnern und rechnergestützten Verfahren aus, doch sind diese beiden Elemente eine ganz entscheidende Komponente in der heutigen bildgebenden Diagnostik. Die Entwicklung, Implementierung und der Einsatz in der Routine vieler Verfahren sind ohne den Einsatz von Computern undenkbar.

Um dreidimensionale Datensätze bildgebender Verfahren im Computer zu visualisieren, müssen die in der Regel als Schichtbilder vorhandenen Daten als Volumendaten interpretiert werden. Die Volumendaten sind in einem regelmäßigen dreidimensionalen Gitter angeordnet, jedem Gitterpunkt ist ein skalarer Intensitäts- oder Farbwert zugeordnet. Dieses regelmäßige Gitter spannt das darzustellende Volumen auf.

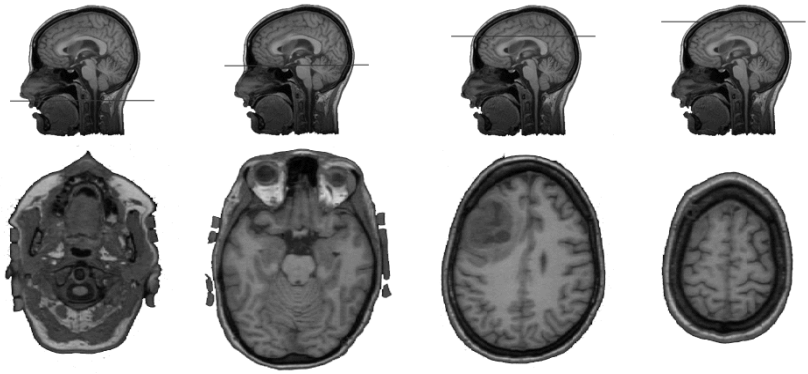

Abbildung 3.1*: Axiale MRT Schnittbilder eines menschlichen Kopfes in verschiedenen Tiefen jeweils angezeigt an einem zentralen, saggitalen Schnitt.*

Ein solches Gitter lässt sich im einfachsten Fall aus den bildgebenden Verfahren der Medizin rekonstruieren, indem die einzelnen Schichtbilder (z.B. MRT/CT) aufeinander gestapelt werden. Abbildung 3.1 zeigt drei solcher zweidimensionaler MRT Bilder in unterschiedlichen Schnitt-Tiefen. Das Stapeln der Schnittbilder ergibt ein Volumen.

Jedem Gitterpunkt wird dabei ein Intensitätswert des jeweiligen Schichtbildes zugeordnet. Eine *Voxelzelle* ist definiert durch die acht Gitterpunkte, die die Zelle umgeben [109]. Im Gegensatz dazu bezeichnet ein *Voxel* einen Würfel um den jeweiligen Abtastpunkt, in dem die Intensität einheitlich ist. Abbildung 3.2 zeigt ein solches Gitter, es sind vier Voxelzellen und ein Voxel dargestellt. Im Folgenden werden strukturierte Gitter

verwendet, die aus Voxelzellen aufgebaut sind, da diese eine bessere Datenstruktur als die Voxelrepräsentation für viele Algorithmen darstellen. Die Visualisierung der Volumenrepräsentation geschieht durch das so genannte Volume-Rendering [61].

Der Begriff Volume-Rendering beschreibt die Visualisierung von dreidimensionalen Daten. Die Aufgabe des Volume-Renderings besteht darin, ein semitransparentes Volumen als zweidimensionale Projektion darzustellen.

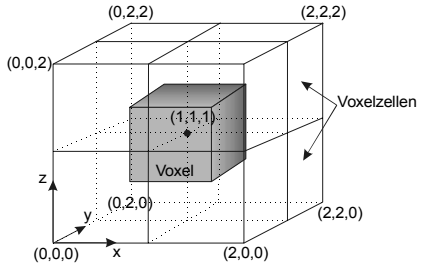

Abbildung 3.2*: Volumenrepräsentation durch strukturiertes Gitter, jeder Eckpunkt des Gitters besitzt einen Intensitätswert, eine Voxelzelle besteht aus 8 Intensitätswerten.*

Die verschiedenen Verfahren des Volume-Rendering lassen sich in indirekte und direkte Verfahren aufteilen [109]. Bei den indirekten Verfahren werden die Volumendaten nicht direkt visualisiert, sondern zuerst in eine andere geometrische Repräsentation umgewandelt. Diese Vorverarbeitung ermöglicht eine sehr schnelle Visualisierung der erzeugten Repräsentation durch auf Polygonnetze spezialisierte Hardware mittels Surface-Rendering (siehe Kapitel 4). Indirekte Verfahren besitzen aber den Nachteil, dass die vorherige Umwandlung Artefakte erzeugen kann, da sie auf der Reduktion der dreidimensionalen Daten auf Polygonnetze basiert.

Die direkten Volume-Rendering Verfahren arbeiten unmittelbar auf den Volumendaten, das heißt, die dreidimensionalen Daten werden direkt auf eine zweidimensionale Ansicht projiziert. Dies bietet den Vorteil, dass die Visualisierung ohne Datenreduktion arbeitet und somit das Volumen ohne Artefakte und mit allen Details angezeigt wird. Durch den Voxeln zugeordnete Transparenzwerte kann das gesamte Volumen semitransparent dargestellt werden, sodass der räumliche Eindruck durch durchscheinende Strukturen stark erhöht wird. Nachteil der direkten Verfahren sind die hohen Anforderungen an die Hardware; dies folgt aus den enormen Datenmengen, die verarbeitet werden müssen. Selbst das interaktive Volume-Rendering von statischen Volumendatensätzen erfordert einen hohen technischen Aufwand. In einem Chirurgiesimulator ist aber zusätzlich eine dynamische Volumendarstellung gefordert, die es dem Experten erlaubt, eine realitätsnahe Volumendarstellung interaktiv zu verändern, z.B. Gewebe zu verformen oder zu entfernen (*fragmentieren*). Eine statische Volumendarstellung unterstützt dies nicht, das Volumen kann zwar interaktiv betrachtet, nicht aber verändert werden. Durch die rasante Entwicklung moderner Texturhardware ist es seit einigen Jahren auch möglich gewor-

den, chirurgische Simulation an einer dynamisch verformbaren Volumendarstellung durchzuführen.

Der folgende Abschnitt enthält eine Übersicht über die verschiedenen direkten Volume-Rendering Verfahren im Hinblick auf ihre Fähigkeit zur Echtzeitdarstellung. Besonderer Wert wird dabei auf die Möglichkeit zur interaktiven Manipulation der Volumendaten gelegt, was die Auswahl der verwendbaren Volume-Rendering Verfahren stark einschränkt. Wegen der reinen Darstellung von Oberflächendaten werden die indirekten Volume-Rendering Verfahren im Kapitel 4 unter Surface-Rendering beschrieben. Im Folgenden meint Volume-Rendering ohne Zusatz immer das direkte Volume-Rendering.

3.2 Direktes Volume-Rendering

Das direkte Volume-Rendering lässt sich nach [109] in *Image-Order*, *Object-Order* und *Domain-basierte* Verfahren einteilen. Bei den Image-Order Verfahren wird für jeden Pixel auf der Bildebene ein Sehstrahl in das Volumen geschickt. Aus den getroffenen Voxeln wird dann der Farbwert des Pixels berechnet. Dieses Verfahren wird auch *Ray-Casting* genannt [107]. Ray-Casting Algorithmen überzeugen oft durch Ihre hervorragende Bildqualität, sind aber nicht echtzeitfähig und damit nicht interaktiv darstellbar [79; 182].

Bei den Object-Order Verfahren wird das Volumen in Schichten zerlegt, die einzeln auf die Bildebene projiziert werden. Durch das Übereinanderlegen der Schichten entsteht schließlich ein Abbild des Volumens. Es wird für jeden Voxel seine Projektion auf die Bildebene bestimmt und der Beitrag zu den Pixeln der Bildebene berechnet. Dieser Beitrag wird mit Hilfe des Compositing erzeugt, indem sein eigener Intensitätswert mit dem getroffenen Pixel auf der Bildebene verknüpft wird, um einen neuen Wert für den Pixel zu berechnen. Dies wird oft auch als das so genannte *Blending* bezeichnet. Ein Problem bei der Projektion der Voxel auf die Bildebene ist, dass die Voxel auf diskrete Bildpunkte abgebildet werden. Dies führte bei ersten Implementierungen zu Artefakten, wenn das Volumen vergrößert dargestellt werden sollte, da dann zwischen den projizierten Voxeln Löcher auf der Bildebene erschienen. Eine Verbesserung des grundlegenden Verfahrens stellte Westover mit seinem Splatting Algorithmus vor [211]. Das Splatting Verfahren legt über jeden Voxel eine Gaußsche Funktion mit endlicher Ausdehnung. Diese Funktion wird dann verwendet, um den Voxel auf die Bildebene zu projizieren. Hierdurch erhält jeder Voxel einen schneeballartigen Abdruck auf der Bildebene, wodurch die oben beschriebenen Artefakte verhindert werden. Die Bildqualität hängt dabei stark von der gewählten Funktion ab, das Verfahren führt leicht zu unscharfen Kanten und vermindert damit den Kontrast des dargestellten Volumens.

Durch ein neues Object-Order Verfahren, das texturbasierte Volume-Rendering, wurde eine Verbesserung der Bildqualität mit gleichzeitiger Geschwindigkeitssteigerung und Echtzeitfähigkeit erreicht. Dieses Verfahren wird im Folgenden erläutert.

3.2.1 Texturbasiertes Volume-Rendering

Das *texturbasierte Volume-Rendering* ist ein Object-Order Verfahren, welches oft auch als Volume-Slicing bezeichnet wird. Es ermöglicht die Visualisierung von Volumendaten in Echtzeit, indem es die technischen Möglichkeiten moderner Grafik-Hardware ausnutzt. Das Verfahren basiert darauf, die vorhandene Textur- und Blending-Funktionalität zu verwenden, um die Visualisierung wesentlich zu beschleunigen. Das Volumen wird in Schnittebenen zerlegt, und die daraus resultierenden Polygone werden dann durch Blending übereinander projiziert. Diese Methode wurde erstmals 1993 in [4] erwähnt. Im darauf folgenden Jahr erschienen erste Arbeiten, welche das texturbasierte Volume-Rendering beschreiben [36; 218]. Die erste Referenzimplementierung VolRen entstand 1996 bei der Firma Silicon Graphics. Im Jahr 1998 veröffentlichte Silicon Graphics schließlich die OpenGL Volumizer API[1], eine C++ Klassenbibliothek, welche eine standardisierte Schnittstelle für das Programmieren von Volume-Rendering bietet.

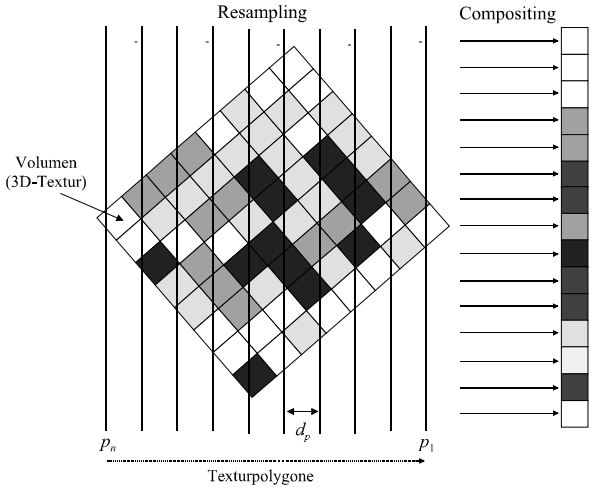

Abbildung 3.3*: Grundlegendes Verfahren des texturbasierten Volume-Renderings (nach* [185]*).*

Das texturbasierte Volume-Rendering ist in den Resampling- und den Compositing-Schritt aufgeteilt. Im Resampling-Schritt wird das Volumen in zur Bildebene parallel liegende Schnittebenen (engl. slices) zerlegt. Jede dieser Schnittebenen wird als semitransparente Textur auf ein rechteckiges Polygon gelegt. Im Compositing-Schritt werden diese Polygone nun von hinten nach vorne (engl. back-to-front) gezeichnet, wobei jeder Bildpunkt durch die kumulative Überlagerung der Polygone seinen endgültigen Intensitätswert erhält. Beide Schritte werden hierbei vollständig in Hardware ausgeführt, was im Gegensatz zu den klassischen Verfahren steht, welche diese Schritte in Software durchführen. Das Verfahren benötigt deshalb nur wenig Prozessorleistung

[1] *Im Folgenden nur noch Volumizer oder OpenGL Volumizer genannt.*

durchführen. Das Verfahren benötigt deshalb nur wenig Prozessorleistung und Hauptspeicher, allerdings erfordert es einen ausreichend großen Texturspeicher. Es wird wesentlich dadurch beschleunigt, dass die nötigen Funktionen wie Texturinterpolation und α-Blending bereits in der Hardware implementiert sind. Abbildung 3.3 zeigt das grundlegende Verfahren mit seinen beiden Schritten. Das Volumen wird dabei in den Texturspeicher geladen und die Texturkoordinaten der Polygone p_1 bis p_n werden so gesetzt, dass sie als Textur die Schnitte aus dem Volumen enthalten. Die so texturierten Polygone werden nun von hinten nach vorne gezeichnet und die Blendingfunktion bestimmt dabei, wie aus den Daten der Textur die Bildpixel erzeugt werden.

Als Eingabewerte erfordert das Verfahren ein Volumen V als regelmäßiges dreidimensionales Feld von skalaren Werten und eine feste Anzahl von rechteckigen Polygonen n, deren Texturkoordinaten zu berechnen sind. Des Weiteren wird die Position und Orientierung von Kamera und Volumen in Weltkoordinaten benötigt. Wie in Abbildung 3.3 für den zweidimensionalen Fall gezeigt, sollen die n Polygone mit einem Abstand von d_p zueinander und parallel zur Bildebene von hinten nach vorne in den Bildspeicher gezeichnet werden. Das Verfahren gliedert sich in die Initialisierung und drei Schritte, die für jedes der n Polygone ausgeführt werden (Algorithmus 3.1).

```
Lade die Volumendaten als 3D-Textur in den Texturspeicher
Für alle Polygone p_1 bis p_n:
Berechne Weltkoordinaten für Polygon p_i parallel zur Bildebene
Versehe p_i mit Schnittbild aus 3D Textur
Zeichne p_i mit Blending-Funktion in den Bildspeicher
```

Algorithmus 3.1: Texturbasiertes Volume-Rendering.

Die einzelnen Schritte des Algorithmus 3.1 werden im Folgenden näher erläutert.

3.2.1.1 Laden der Textur

Zur Initialisierung des Verfahrens müssen die Volumendaten als dreidimensionale Textur in den Texturspeicher geladen werden. Dies bietet den Vorteil, dass nicht mehr auf den vergleichsweise langsamen Hauptspeicher zugegriffen werden muss und somit die Hardwarefunktionalität des dreidimensionalen Texturemappings voll ausgenutzt werden kann. Dreidimensionales Texturemapping ist eine Erweiterung des weit verbreiteten zweidimensionalen *Texturemapping*s. Beim Texturemapping [61] wird ein zweidimensionales Bild als Textur auf ein dreidimensionales Polygonmodell gelegt. Hierbei wird die Textur auf Polygone abgebildet, indem jedem Eckpunkt eines Polygons eine Texturkoordinate zugeordnet wird. Die Texturkoordinaten werden als zweidimensionale, auf die Texturgröße normierte Koordinaten (u,v) angegeben. Ein Pixel dieser Textur wird als *Texel* bezeichnet.

Das dreidimensionale Texturemapping erweitert die Texturfunktionalität um eine weitere Dimension, sodass die Texturkoordinaten (u,v,w) als Position innerhalb eines drei-

dimensionalen Texturwürfels angegeben sind. Dadurch ist es möglich, eine beliebige Schnittebene durch den Texturwürfel auf ein Polygon abzubilden.

3.2.1.2 Berechnung der Polygonkoordinaten

Zur Berechnung der Polygonkoordinaten wird das Volumen *V* achsenparallel zu den Koordinatenachsen und um den Nullpunkt zentriert in das Weltkoordinatensystem gelegt. Es wird ein Würfel *Q* ermittelt, dessen Seitenlänge *d* die Raumdiagonale des Volumens ist (siehe Abbildung 3.4).

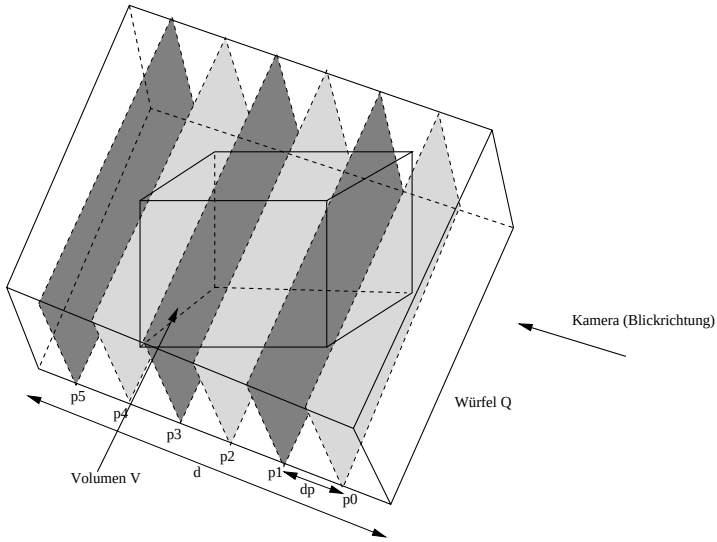

Abbildung 3.4*: Zerlegung des Würfels Q in Schnittpolygone, der Würfel ist in Richtung der Kamera rotiert, damit die Schnittpolygone parallel zum Betrachter liegen (aus* [185]*).*

Dieser Würfel wird ebenfalls um den Nullpunkt zentriert, er beinhaltet somit das gesamte Volumen. Durch die Wahl der Raumdiagonale als Seitenlänge beinhaltet der Würfel das Volumen auch nach einer beliebigen Rotation um den Nullpunkt. Dies ist wichtig um zu verhindern, dass Teile des Volumens nicht durch Polygone geschnitten werden. Der Würfel *Q* wird in Polygone aufgeteilt, indem der Würfel in Richtung der Z-Achse gleichmäßig in *n* parallele Polygone aufgeteilt wird. Der Abstand der Polygone beträgt hierbei $d_p = d/n$.

Der so bestimmte Würfel muss nun je nach Kameraorientierung rotiert werden, um zu gewährleisten, dass die Schnittebenen parallel entlang der Blickrichtung liegen. Hierzu wird die Rotationsmatrix R_K der Kamera aus der Transformationsmatrix der Kamera isoliert. Nach der Rotation wird der Würfel mittels einer Transformation T_V positioniert und skaliert, hierdurch erscheint das dargestellte Volumen ebenfalls verschoben und skaliert,

ohne dass es effektiv verändert wurde. Die gesamte Transformation des Würfels T_Q ergibt sich also aus $T_Q = T_V R_K T_V^{-1}$.

3.2.1.3 Zuordnung der Schnittbilder zu den Polygonen

In diesem Schritt werden den erzeugten Polygonen die jeweiligen Schnittbilder zugeordnet. Dies erfolgt über das dreidimensionale Texturemapping. Durch die Verwendung der dreidimensionalen Textur-Hardware wird die Darstellung beliebiger Schnitte durch ein Volumen zu einer trivialen und schnellen Operation.

Die Texturkoordinaten werden berechnet, indem die Polygonkoordinaten der den Würfel aufteilenden Polygone aus dem Weltkoordinatensystem in das Texturkoordinatensystem übertragen werden. Hierzu wird eine Skalierung S benötigt, welche die Koordinaten auf den Texturraum skaliert. Dies ist dadurch bedingt, dass das Volumen im Texturspeicher auf 1 normiert ist. Des Weiteren ist eine Translation zum Mittelpunkt des Volumens innerhalb der Textur nötig, was über die Translation T_0 zum Punkt (0,5; 0,5; 0,5) erfolgt. Somit ergibt sich die Transformation $T_T = T_0 S R_K T_V^{-1}$, welche auf die Polygonkoordinaten angewendet die entsprechenden Texturkoordinaten liefert.

Die Volumendaten bestehen aus diskreten Werten, woraus folgt, dass eine geeignete Interpolationsmethode gesucht ist, um bei einer vergrößerten Ansicht Darstellungsfehler (sogenannte *Artefakte*) zu vermeiden. Die einfachste Methode ist hierbei die Nearest-Neighbour-Interpolation, bei dem jedem Raumpunkt der nächstliegende Texel zugeordnet wird. Alle Punkte innerhalb eines Voxels erhalten so denselben Wert und es entstehen leicht rechteckige Treppenstufeneffekte. Eine wesentlich bessere Interpolationsmethode ist die trilineare Interpolation [77]. Bei dieser Methode werden für jeden Raumpunkt die acht benachbarten Werte der Eckpunkte der Voxelzelle betrachtet, in welcher sich der Raumpunkt befindet. Hierbei werden die einzelnen Werte je nach Abstand zum Raumpunkt gewichtet addiert.

3.2.1.4 Zeichnen mit Alpha-Blending in den Bildspeicher

Die in den vorherigen Schritten berechneten und mit Schnittbildern versehenen Polygone werden nun nacheinander in den Bildspeicher gezeichnet. Die Art des Zeichnens lässt sich durch die Wahl einer *Alpha-Blending* Funktion festlegen, wodurch bestimmt wird, wie einzelne Pixel der Polygone, die auf denselben Bildpunkt fallen, kombiniert werden. Hierzu benötigt man eine allgemeine $RGBA_1$ Textur, in der jedem Voxel sowohl ein RGB Farbwert, als auch eine die Transparenz des Voxels bestimmende α-Komponente zugeordnet ist. Ein α-Wert von 1 bedeutet, dass der Voxel opak ist, 0 stellt die vollständige Transparenz dar. Mit Werten im Intervall [0, 1] kann eine Semitransparenz erreicht werden, wodurch die Darstellung von semitransparenten Volumendaten möglich wird. Um semitransparente Polygone miteinander zu mischen, muss also eine Funktion bestimmt werden, welche aus dem momentanen Bildpunkt und dem Pixel des abgebildeten Polygons einen neuen Farb- und Transparenzwert berechnet. Hierbei müssen die Farbwerte der beiden Punkte nach ihren jeweiligen α-Werten gewichtet werden,

---

[1] RGBA, Red Green Blue Alpha. Die Farben Rot, Grün, Blau für additives Mischen der Farben, Alpha gibt zusätzlich die Transparenz an.

um den neuen Farbwert zu berechnen. Die bekannteste Transparenzfunktion, die zugleich die besten Ergebnisse liefert, ist wie folgt definiert [206]:

$$C_{out} = (1 - \alpha)\, C_{in} + \alpha C \tag{3.1}$$

wobei C_{out} der neu berechnete Farbwert des Bildpunktes ist, C_{in} den alten Wert des Bildpunktes angibt und C und α den Farb- und Alphawert des aktuellen Pixels des Polygons bestimmen.

Die wohl wichtigste Bedingung für den Einsatz des Volume-Renderings für die Virtuelle Endoskopie und chirurgische Simulation ist die Echtzeitfähigkeit des Verfahrens, verknüpft mit der Möglichkeit der interaktiven Veränderung des Volumens im Falle der chirurgischen Simulation. Das günstigste Verfahren in Bezug auf die Echtzeitfähigkeit stellt das texturbasierte Volume-Rendering dar. Dies folgt aus der intensiven Verwendung der Grafikhardware, was zu einer enormen Performanzsteigerung gegenüber den herkömmlichen Verfahren führt. Das bisher vorgestellte Verfahren des texturbasierten Volume-Rendering enthält allerdings außer Translation, Rotation und Skalierung keine Möglichkeiten der interaktiven Veränderung des Volumens. Im folgenden Kapitel wird daher OpenGL Volumizer beschrieben, welches das bereits vorgestellte einfache Verfahren um diverse Möglichkeiten der Interaktion erweitert und somit ein ideales Visualisierungsverfahren für interaktive medizinische Simulatoren bildet.

3.2.2 OpenGL Volumizer

OpenGL Volumizer [55] ist ein texturbasiertes Volume-Rendering Verfahren, das die Firma Silicon Graphics, Mountain View, USA 1998 entwickelt hat. OpenGL Volumizer besitzt eine standardisierte Schnittstelle für das Programmieren mit Volume-Rendering und wird als C++ Klassenbibliothek ausgeliefert. Dadurch bietet Volumizer eine hohe Abstraktion, die es dem Programmierer ermöglicht, portierbare Applikation zu schreiben, die dennoch effizientes, hardware-beschleunigtes Volume-Rendering beinhalten. Volumizer kapselt hierzu die plattform-spezifischen Details, sodass der Programmierer sich nicht mit der konkreten Grafikhardware des verwendeten Rechners auseinandersetzen muss. OpenGL Volumizer bietet neben einer einfachen Visualisierung auch die Möglichkeit der dynamischen Veränderung der dargestellten Volumendaten. Hierzu wird die Repräsentation des darzustellenden Volumens in die beiden Grundkomponenten *Appearance* und *Geometry*, also Erscheinung und Geometrie, aufgeteilt. In der dreidimensionalen Computergrafik werden graphische Objekte häufig auf diese Art definiert. Zum Beispiel kann ein dreidimensionaler Würfel durch die Angabe seiner sechs Seitenflächen (Geometry) und das Aufbringen einer zweidimensionalen Textur (Appearance) auf jeder dieser Seiten beschrieben werden. Die Appearance enthält die eigentlichen Volumendaten als dreidimensionales Feld skalarer Werte, entspricht also der in Kapitel 3.1 beschriebenen Volumendefinition. Mit der Geometry wird die visuelle Darstellung der Volumendaten als eine auf einer Menge von Tetraedern aufgebaute Geometrie beschrieben. Hierbei wird das Tetraeder als volumetrisches Grundprimitiv eingeführt. Eine Geometry ist also eine aus Tetraedern aufgebaute Geometrie, welche die Darstellung der Volumendaten bestimmt. Durch Veränderung der Geometrie kann nun die Darstellung des Volu-

mens manipuliert werden, ohne dass dabei die eigentlichen Volumendaten verändert werden müssen. Damit steht OpenGL Volumizer im Kontrast zu herkömmlichen Volume-Rendering Verfahren, welche keine solche Trennung beinhalten.

3.2.2.1 Geometrie

Zur Definition eines volumetrischen Körpers mittels einer Geometrie wird ein geometrisches Grundprimitiv benötigt. Das Dreieck ist das einfachste und flexibelste Primitiv das zur Darstellung von polygonaler Geometrie verwendet werden kann. Jedes beliebige Polygon kann durch Triangulation in Dreiecke zerlegt werden [147]. Analog hierzu ist das Tetraeder das einfachste und effizienteste Primitiv, welches zur Repräsentation einer volumetrischen Geometrie verwendet werden kann. Jeder Polyeder kann in Tetraeder zerlegt werden [71]. Ein Würfel kann z.B. in fünf Tetraeder aufgeteilt werden. Aus diesem Grund verwendet OpenGL Volumizer das Tetraeder als grundlegendes *volumetrisches Primitiv*. Zusätzlich bietet Volumizer auch volumetrische Primitive höherer Ordnung, allerdings werden diese Primitive intern ebenfalls in Tetraeder aufgeteilt. Beispiele für volumetrische Primitive höherer Ordnung sind Pyramiden, Prismen und Hexaeder.

Jedes Tetraeder der Geometrie ist definiert durch die Koordinaten seiner vier Eckpunkte, welche in Weltkoordinaten angegeben werden. Durch diese Eckpunkte ist die räumliche Lage des Tetraeders bestimmt. Hierdurch ist allerdings noch nicht festgelegt, welcher Ausschnitt der Volumendaten (also der Appearance) innerhalb des Tetraeders dargestellt werden soll. Analog zu dem in Kapitel 3.2.1 vorgestellten dreidimensionalen Texturemapping müssen den Eckpunkten des Tetraeders Texturkoordinaten zugeordnet werden. Hierdurch wird festgelegt, welchen Bereich des Volumens der Tetraeder enthält. Mittels linearer Interpolation kann so für jeden Raumpunkt innerhalb des Tetraeders eine eindeutige Zuordnung zu einer Voxelzelle des Volumens erfolgen. Die Texturkoordinaten sind auf 1 normiert, für jeden Eckpunkt eines Tetraeders wird eine (u,v,w) - Texturkoordinate festgelegt. Im einfachsten Fall ist die Geometrie ebenfalls auf 1 normiert und zeigt die Volumendaten eins-zu-eins an. In diesem Fall müssen die Texturkoordinaten nicht explizit angegeben werden, sondern es können direkt die geometrischen Koordinaten als Texturkoordinaten verwendet werden.

3.2.2.2 Appearance

Die Appearance enthält die eigentlichen Volumendaten. Volumendaten können leicht eine Größe erreichen, die nicht in den Texturspeicher der Grafikhardware passt. Aus diesem Grund besitzt Volumizer eine virtuelle Voxel-Speicherverwaltung. Hierzu wird das gesamte Volumen in *Blöcke* (Bricks) aufgeteilt. Diese Blöcke enthalten jeweils einen dreidimensionalen Würfel der Volumendaten, der so gewählt werden sollte, dass er in den Texturspeicher der Hardware passt. Um das Volumen anzuzeigen, müssen die darzustellenden Blöcke in den Texturspeicher geladen werden. Blöcke, die außerhalb des aktuell sichtbaren Bereiches bzw. der Kameraperspektive liegen, müssen nicht berücksichtigt werden. Hierdurch können Ausschnitte eines sehr großen Volumens, welches nur durch Hauptspeicher und Festplattenkapazität beschränkt ist, dargestellt werden. Die Appearance besteht demnach aus einer Menge von Blöcken. Volumizer verlangt aus Optimierungsgründen eine Blockgröße in Zweierpotenzen; ein Volumen von 256x256x128

Voxel lässt sich z.B. in 4 Blöcke der Größe 128^3 einteilen. Die Blöcke müssen sich hierbei an ihren Rändern überlappen, um Artefakte an den Blockübergängen zu vermeiden. Die Artefakte entstehen aufgrund der Interpolation, welche am Rand eines Blocks keinen Zugriff auf die Voxel des benachbarten Blocks besitzt. Um diese Interpolationsfehler zu vermeiden, muss ein Block jeweils den Rand des benachbarten Blocks enthalten. Die Interpolation erhält so die Voxelwerte des benachbarten Blocks, ohne auf diesen zugreifen zu müssen. Für die trilineare Interpolation (siehe Kapitel 3.2.1) beträgt dieser Rand einen Voxel, sodass aus der Überlappung ein zu vernachlässigender zusätzlicher Speicherverbrauch entsteht, sofern keine extrem kleinen Blöcke verwendet werden.

3.2.2.3 Der Rendering-Algorithmus

Volumizer verarbeitet ein darzustellendes Volumen in zwei getrennten Schritten, der Polygonisierung und dem Compositing. Das Volumen sei hierbei gegeben durch seine Appearance und Geometry. Weitere Eingabewerte für die Darstellung sind die Projektionsmatrix und die Transformationsmatrix. Volumizer basiert auf dem texturbasierten Volume-Rendering (siehe Kapitel 3.2.1), erweitert es aber um die frei definierbare Tetraedergeometrie, welche die Darstellung beliebiger volumetrischer Körper ermöglicht. Durch die frei wählbare Geometrie erhöht sich die Komplexität des Resampling-Schritts wesentlich. Im Folgenden wird dieser Schritt als Polygonisierung bezeichnet. Auf den Compositing-Schritt wird hier nicht weiter eingegangen, da er dem in Kapitel 3.2.1.4 beschriebenen Schritt entspricht.

Für jede Schnittebene s_i, $i \in \{s_1, \ldots, s_n\}$:
    Berechne Weltkoordinaten für Schnittebene s_i parallel zur Bildebene
    Für jedes Tetraeder t_j, $j \in \{1, \ldots, k\}$ der Geometrie:
        Teste auf Schnitt zwischen Tetraeder und Polygon
        Falls Schnitt vorhanden, berechne Schnittpolygon, sonst nächstes Tetraeder
        Berechne Texturkoordinaten für das Schnittpolygon
        Füge Schnittpolygon in die Polygonliste der aktuellen Schnittebene ein

---

Algorithmus 3.2: Polygonisierungsverfahren von OpenGL Volumizer.

Um ein Volumen darzustellen, unterteilt es Volumizer in parallele Schnittebenen, was dem Verfahren entspricht, welches in Abbildung 3.4 dargestellt ist. Im Unterschied zu herkömmlichen texturbasierten Verfahren müssen diese Schnittebenen allerdings nicht mit einem Volumenwürfel geschnitten werden, sondern mit der Tetraedergeometrie. Die Tetraedergeometrie ist hierbei gegeben durch eine Menge von Tetraedern $\{t_1, \ldots, t_k\}$, die durch n Schnittebenen $\{s_1, \ldots, s_n\}$ geschnitten werden. Das Polygonisierungsverfahren erzeugt für jede Schnittebene eine Liste l von Polygonen mit entsprechenden Texturkoordinaten, die aus dem Schnitt zwischen den Tetraedern und der Schnittebene berechnet werden (Algorithmus 3.2).

Aus dem Schnitt eines Tetraeders und einer Ebene ergibt sich ein Drei- bzw. Viereck, je nachdem wie der Schnitt durchgeführt wird [55]. Die Texturkoordinaten für dieses erzeugte Polygon lassen sich durch lineare Interpolation aus den Texturkoordinaten der Eckpunkte des Tetraeders berechnen. Während bei dem herkömmlichen Verfahren pro Schnittebene nur ein viereckiges Polygon gezeichnet werden muss, enthält nun jede Schnittebene eine Liste von Drei- und Vierecken. Abbildung 3.5 zeigt einen bereits polygonisierten Tetraeder.

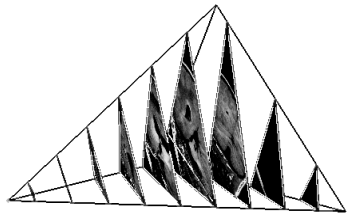

Abbildung 3.5*: Ein in 9 Ebenen geschnittenes Tetraeder. Die entstandenen Dreiecke sind mit den entsprechenden Schnitt-Texturen belegt (aus* [185]*).*

Die so erzeugten Polygone könnten nun von hinten nach vorne gezeichnet werden, angefangen mit der Polygonliste der hintersten Schnittebene. Dies würde allerdings voraussetzen, dass das gesamte Volumen in einen Block (siehe 3.2.2.2) passt. Da dies im Allgemeinen nicht der Fall ist, müssen in einem weiteren Verarbeitungsschritt alle erzeugten Polygone an den Blöcken geschnitten werden. Hierzu wird für jeden Block eine zusätzliche Liste von Polygonlisten angelegt, bei n Schnittebenen also n Polygonlisten pro Block. Jedes Polygon der ursprünglichen Schnittlisten l wird nun mit jedem Block geschnitten. Falls das geschnittene Polygon innerhalb des Blocks liegt, wird es als neues Polygon in die entsprechende Polygonliste des Blocks eingetragen. Aus dem Schnitt eines n-eckiges Polygons mit einem Würfel entsteht maximal ein n+6-eckiges Polygon, d.h. insgesamt kann bei dem Clipping ein 10-eckiges Polygon entstehen. Nach diesem Clipping müssen die Blöcke nach ihrer Raumlage sortiert werden, um sie aus dem Blickwinkel der Kamera von hinten nach vorne zu zeichnen. Danach kann die Darstellung des Volumens durch das Zeichnen der einzelnen Polygonlisten stattfinden.

Abbildung 3.6 zeigt ein rechteckiges Volumen, welches in fünf Tetraeder zerlegt wurde. Links ist die Polygonisierung zu sehen, die einzelnen Texturen sind opak dargestellt, um den Aufbau der Polygone zu verdeutlichen. Das mittlere Bild zeigt das Volumen mittels Alpha-Blending, wobei die semitransparenten Texturen übereinander gezeichnet werden. Die Abbildungen wurden mit einer geringen Anzahl von Schnittebenen berechnet und nach der Polygonisierung rotiert, damit die einzelnen Polygone sichtbar sind. Ansonsten wären die Polygone senkrecht auf dem Sichtstrahl und dann wäre nur das erste Polygon sichtbar. Die Darstellung mit Alpha-Blending und einer hohen Anzahl an Schnitten (Nyquist-Rate [55]) auf der rechten Seite zeigt die maximale Bildqualität, einzelne Schnittebenen werden nicht mehr wahrgenommen.

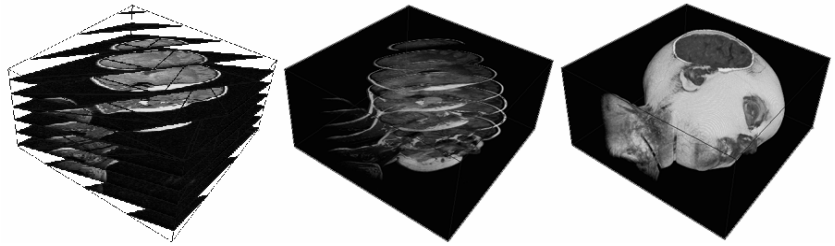

Abbildung 3.6*: Von links nach rechts: Polygonisierung eines in 5 Tetraeder aufgeteilten Volumens (opak), mit Alpha-Blending, hochaufgelöste Darstellung des Volumens durch hohe Anzahl an Schnittebenen (aus* [185]*).*

3.2.2.4 Qualität der Darstellung

Die Qualität der Darstellung, die mit Volumizer erreicht werden kann, hängt von der Anzahl der gewählten Schnittebenen (engl. Samples) ab. Eine ideale Anzahl der Schnittebenen ist erreicht, wenn die Nyquist-Rate eingehalten wird, d.h. wenn die Anzahl der Schnittebenen der doppelten Auflösung des Volumens entspricht [61]. Die zu erzeugende Anzahl der Schnittebenen wird in Volumizer durch den SamplingPeriod-Vektor angegeben, hierbei wird jeweils für die X, Y und Z-Achse eine Sampling-Periode angegeben. Volumizer verwendet dann den Wert der Achse, welche den kleinsten Winkel zur Kameraorientierung besitzt, um die Anzahl der Schnittebenen festzulegen.

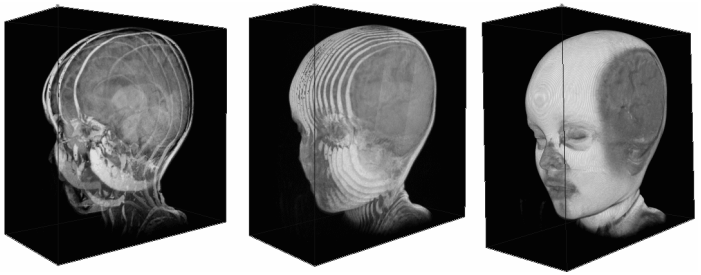

Abbildung 3.7*: Darstellung eines Volumens mit steigender Anzahl von Schnittebenen (aus* [185]*).*

Abbildung 3.7 zeigt ein Volumen der Größe 256x256x128, welches mit unterschiedlicher Sampling-Periode dargestellt wurde, das rechte Volumen ist mit der größtmöglichen Schnittebenenanzahl dargestellt. Eine zu niedrige Schnittebenenzahl erzeugt Artefakte, da so nur ein Teil der vorhandenen Volumendaten zur Visualisierung verwendet wird und der Abstand zwischen den einzelnen Ebenen zu groß gerät.

Die Sampling-Rate kann dynamisch verändert werden. Dadurch ist es beispielsweise möglich, das Volumen mit niedriger Auflösung darzustellen, wenn es interaktiv rotiert oder transformiert wird, und es in höherer Detailtreue darzustellen, wenn es nicht mehr bewegt wird.

Volumizer bietet neben der Echtzeitfähigkeit der visuellen Darstellung von Volumendaten eine wesentliche Erweiterung der Möglichkeiten gegenüber herkömmlichen texturbasierten Volume-Rendering Verfahren. Durch die Trennung in Geometry und Appearance und die Visualisierung durch die Texturhardware ist die Darstellung von deformierbaren volumetrischen Körpern, welche aus Tetraedern aufgebaut sind, möglich geworden. Des Weiteren bietet Volumizer eine standardisierte, im Prinzip plattformunabhängige Schnittstelle für das Volume-Rendering, die eine Programmierung des Volume-Renderings auf abstrakter Ebene ermöglicht. Dies erhöht die Wiederverwendbarkeit der erstellten Applikationen, da diese nicht an hardwarenahe Funktionen gebunden sind. Ein wichtiger Vorteil von Volumizer, aber auch allgemein von texturbasierten Verfahren, ist die einfache Integration in bestehende polygonbasierte 3D-Bibliotheken. Da Volumizer ein Volumen als eine Anzahl von semitransparenten texturierten Polygonen darstellt, können gleichzeitig beliebige polygonbasierte opake Objekte angezeigt werden [71]. Dies ist gerade in einem interaktiven Simulator sehr wichtig, da chirurgische Instrumente, wie z.B. ein Endoskop, normalerweise als Polygonmodelle vorliegen. Volumizer bietet also eine ideale Möglichkeit zum Mischen traditioneller polygonbasierter 3D-Grafik mit der Volumenvisualisierung.

Problematisch ist jedoch der Aufwand für die Berechnung der Schnittpolygone bei einer hohen Anzahl von Tetraedern. Um diesen Aufwand zu reduzieren, kann die neu zu berechnende Region – wie im folgenden Kapitel beschrieben - durch ein so genanntes *Volume of Interest* eingeschränkt werden.

3.2.3 Geschwindigkeitsgewinn durch das Volume of Interest

Im letzten Kapitel wurde ein Verfahren vorgestellt, das aus Oberflächenmodellen eine aus Tetraedern aufgebaute Geometrie erzeugt. Ein Problem bei der Visualisierung dieser Geometrie ist die hohe Anzahl der erzeugten Tetraeder bei wachsender Komplexität des zugrunde liegenden Oberflächenmodells. Eine wesentliche Beschleunigung der Visualisierung bietet ein so genanntes *Volume of Interest* (VOI). Hierbei wird die Tatsache ausgenutzt, dass insbesondere bei der Virtuellen Endoskopie aus der Blickrichtung und Position des Betrachters (bzw. der Kamera) selten das gesamte Volumen zu sehen ist. Nicht sichtbare Ausschnitte des Volumens müssen daher nicht berechnet werden.

In Abbildung *3.8* ist das VOI durch einen Kegel visualisiert. Der Sichtwinkel der Kameraoptik des Endoskops beträgt 100°. Das aus Endoskopperspektive sichtbare Volumen stellt nur ca. 5% des gesamten Volumendatensatzes dar. Bei jeder Änderung von Kameraparametern wird dieses VOI neu berechnet, sei es bei Positionsveränderungen oder Rotationsbewegungen.

Bei Verwendung von OpenGL ist das Sichtfeld der Kamera meist rechteckig und kann somit den gesamten Bildschirm ausfüllen. Runde Sichtfelder sind bei der Endoskopie üblicher und können durch Überlagern der visualisierten Szene mit einer kreisförmigen Blende erzeugt werden [99].

Das VOI berechnet sich durch Projektion des sichtbaren Sichtfeldes der Kamera aus den Kamerakoordinaten in Weltkoordinaten, d.h. in Koordinaten innerhalb des Volumens. Die vier Eckpunkte des Sichtfeldes können per Matrixmultiplikation in die Welt-

koordinaten transformiert werden. Aus dem so berechneten VOI wird eine Tetraedergeometrie aufgebaut.

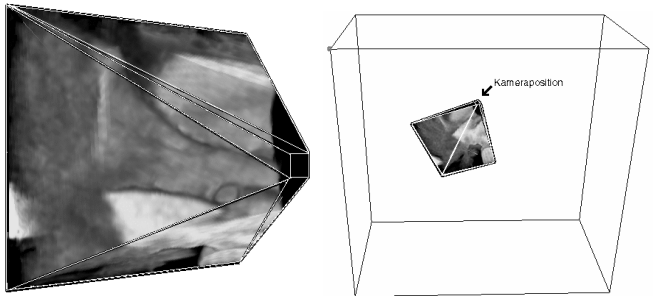

Abbildung 3.8*: Ein aus fünf Tetraedern aufgebautes VOI (links) und die Position des VOI im Volumendatensatz (rechts) (aus* [185]*).*

Im einfachsten Fall wird dieses Sichtfeld in fünf Tetraeder zerlegt, welche dann als Volumizer Geometrie verwendet werden (siehe Abbildung 3.8 (links)). Abbildung 3.8 (rechts) zeigt die Position und Orientierung des Sichtfeldes relativ zum gesamten Volumendatensatz. Dieser Aufbau des VOI eignet sich für die statische Visualisierung, beispielsweise für die Virtuelle Endoskopie. Das VOI kann aber auch dazu verwendet werden, die Darstellung komplexer Tetraedergeometrien zu beschleunigen. Dies geschieht, indem nur die Tetraeder dargestellt werden, welche sich im Sichtfeld befinden. Dies bedeutet einen enormen Geschwindigkeitsvorteil bei der Visualisierung, was daran liegt, dass eine aus einem Oberflächenmodell erzeugte Tetraedergeometrie aus einigen tausend Tetraedern bestehen kann, allerdings wesentlich weniger Tetraeder aus der Kamerasicht zu sehen sind.

Zur Berechnung der darzustellenden Tetraeder muss für jedes Tetraeder getestet werden, ob es im aktuellen VOI enthalten ist. Dazu genügt ein einfacher Test, der für alle Tetraeder überprüft, ob einer der Eckpunkte innerhalb des Sichtfeldes liegt, wofür beispielsweise der in [86] präsentierte Algorithmus verwendet werden kann. Im Falle eines rechteckigen VOI wie in Abbildung 3.8 angegeben, kann mit einem Verfahren gearbeitet werden, das für jeden Punkt überprüft, ob er innerhalb der durch die begrenzenden Flächen aufgespannten Ebenen liegt [1].

Neben der Verwendung von OpenGL Volumizer zum Geschwindigkeitsgewinn bei der Visualisierung durch das Volume of Interest existiert die noch interessantere Möglichkeit der Volumenmanipulation zur Simulation von Deformationen oder Fragmentierungen, die im folgenden Kapitel beschrieben wird.

3.3 Volumenmanipulation

In diesem Abschnitt werden Verfahren vorgestellt, die dazu geeignet sind, eine visuell realistische Verformung von volumetrischen Körpern zu simulieren und darzustellen. Grundsätzliche Voraussetzung für eine Interaktion ist die Erkennung von Kollisionen des virtuellen Instruments mit dem Volumen. Die entsprechenden Verfahren sind in Anhang A beschrieben.

Wie bereits im letzen Kapitel aufgezeigt wurde, bietet OpenGL Volumizer durch die frei wählbare Volumengeometrie die grundlegende Möglichkeit der Visualisierung von deformierten und fragmentierten Volumen. Es werden zwei Methoden zur Erzeugung einer geeigneten Repräsentation eines volumetrischen Körpers im Feder-Masse-Modell vorgestellt. Die erste Methode basiert darauf, den Körper durch eine volumetrische Diamantgitterstruktur anzunähern, die zweite verwendet ein Oberflächenmodell des Körpers. Für diese beiden Methoden wird schließlich die Erzeugung einer Tetraedergeometrie beschrieben, welche zur Visualisierung durch OpenGL Volumizer geeignet ist.

3.3.1 Geometrische Zerlegung des Volumens

3.3.1.1 Volumen als Diamantgitter

Für die Darstellung deformierbarer Volumen, deren Dynamik durch die in Kapitel 2 beschriebene Neuro-Fuzzy Deformation simuliert wird, benötigt man eine Aufteilung des Volumens in ein Feder-Masse System. Dabei können grundsätzlich zwei verschiedene Ansätze verwendet werden. Zum einen gibt es die Möglichkeit, ein volumetrisches Netz zu erzeugen, welches das gesamte Volumen ausfüllt. Hierzu wird im Folgenden die Erzeugung einer den darzustellenden Körper ausfüllenden Diamantgitterstruktur vorgestellt. Dieser Ansatz erzeugt eine nahezu physikalisch korrekte Repräsentation, da das gesamte Volumen des Objektes mit Masseknoten ausgefüllt wird und die Kraftverteilung im inneren des Objektes optimiert ist. Des Weiteren wird eine Zerlegung des Diamantgitters in Tetraeder beschrieben, was die Verwendung von OpenGL Volumizer beschleunigt.

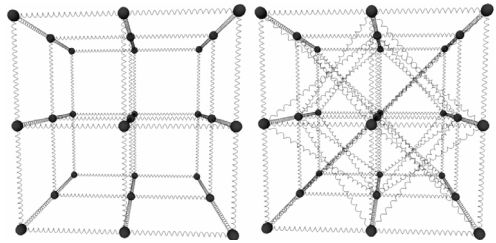

Abbildung 3.9*: Feder-Masse Systeme motiviert durch Kristallgitter. Links: einfaches Würfelgitter (Simple Cubic). Rechts: Body Centered Würfelgitter (Body Centered Cubic) (aus* [166]*).*

Im Folgenden wird ein Verfahren beschrieben, welches ein beliebiges Volumen in ein volumetrisches Netz aufteilt. Abbildung 3.9 zeigt zwei Beispiele von volumetrischen Netzen die durch Kristallgitter motiviert sind: links ein einfaches Würfelgitter und rechts ein Body Centered Würfelgitter (z.B. Natrium oder Kalium) [5; 221]. Dabei wird das Volumen in Würfel aufgeteilt, die Atome im Kristallgitter werden durch Masseknoten repräsentiert und die Ionenverbindungen durch Federn. Die Idee dabei ist die Stabilität und Regelmäßigkeit von Kristallgittern für Volumengitter für die Volumendeformation zu nutzen. Bei der Simulation von Volumengittern in der Form von einfachen Würfelgittern kann es aufgrund der fehlenden diagonalen Stabilität zu einem seitlichen Ausscheren von Masseknoten kommen [158]. Die Stabilität kann durch die Verwendung von zusätzlichen diagonalen Verbindungen im Body Centered Würfelgitter[1] erhöht werden. Dies ermöglicht eine bessere Kraftverteilung bei Deformationen, die nicht entlang der Hauptachsen erfolgen.

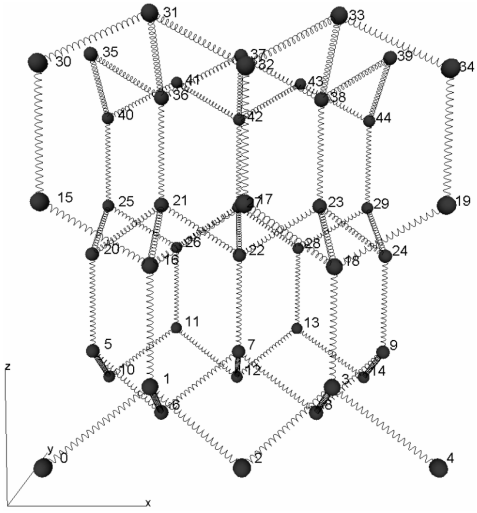

Abbildung 3.10*: Feder-Masse System nach einem Diamantgitter (aus* [166]*).*

Diese diagonalen Federn führen allerdings dazu, dass jeder interne Knoten durch 14 Federn mit seinen Nachbarn verbunden ist. Hierdurch wird das Modell unnötig komplex und die Simulationsgeschwindigkeit sinkt. Es ist sinnvoll, das Modell zu vereinfachen und damit die Performanz der Simulation zu steigern, ohne die Stabilität der Struktur zu reduzieren. Aus diesem Grund wurden Kristallstrukturen untersucht, die eine minimale Anzahl von Federn besitzen, ohne dass dabei die Stabilität beeinträchtigt wird [159].

---

[1] *Im Gegensatz zum Body Centered Würfelgitter ist in Abbildung* 3.9 *(rechts) das Atom im Zentrum der diagonalen Verbindungen aus Geschwindigkeitsgründen nicht als Masseknoten modelliert worden.*

In Diamantgittern (siehe Abbildung 3.10) besitzt jedes Atom nur vier Verbindungen zu anderen Atomen [116]. Dennoch ist die Gitterstruktur des Diamanten stabiler als die der anderen vorgestellten Strukturen [94]. Bei einem analog zum Diamantgitter erzeugten Feder-Masse System wird eine anliegende Kraft gleichmäßig an drei benachbarte Knoten weitergegeben, da der Winkel zwischen allen beteiligten Federn 120° beträgt. Das folgende Kapitel beschreibt den Aufbau eines solchen Diamantgitters und wie daraus eine Tetraedergeometrie für Volumizer erzeugt werden kann.

3.3.1.2 Erstellung des Diamantgitters

Eine Möglichkeit, ein Diamantgitter zu erzeugen, besteht in der Wiederholung der in Abbildung 3.11 gezeigten grundlegenden Tetraederstruktur in x-,y-, und z-Richtung. Ein so erzeugtes Gitter besitzt allerdings eine Schräglage in allen Richtungen. Dies ist für die Kombination mit einer Volumendarstellung ungeeignet, da die Volumendaten ein rechtwinkliges Volumen ausfüllen.

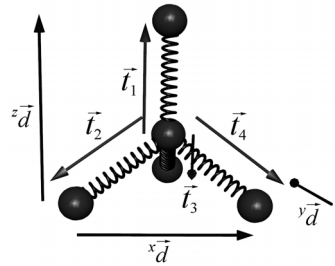

Abbildung 3.11*: Struktur zur Erzeugung eines Diamantgitters (aus* [166]*).*

Daher wird in Algorithmus 3.3 eine Prozedur vorgestellt, die benutzt werden kann, um ein Diamantgitter zu erzeugen, das an ein quaderförmiges Objekt angepasst ist. Ein solches Diamantgitter wird im Folgenden rechtwinklig genannt.

Die Knotenindizes sind in Abbildung 3.10 am Beispiel eines Diamantgitters mit 5×3×3 Knoten gezeigt. Für eine einfache Indizierung der Knoten sind diese aufsteigend in *x*-, *y*- und *z*-Richtung nummeriert.

Der Vektor $\vec{S}$ definiert die Größe des Gitters angegeben in Knotenindizes. Die Initialisierung der Vektoren $\vec{d}$ und $\vec{t}_1$ bis $\vec{t}_4$ ist aus Abbildung 3.11 ersichtlich. Δ_y und Δ_z enthalten die Differenzen der Knotenindizes zwischen den Schichten in *y*- und *z*-Richtung. Die Prozedur iteriert mittels dreier Schleifen über das gesamte Volumen und erzeugt ein Netz aus Masseknoten und Federn, die in einer Diamantstruktur angeordnet sind. Die Schleifenvariablen *p*, *q*, *r* zeigen auf die Schicht in *x*-, *y*- und *z*-Richtung. Die Vektoren $\vec{v}_1$ und $\vec{v}_2$ enthalten die Position von zwei aufeinander folgenden Knoten in *x*-Richtung. Wenn *q* gerade ist, dann wird $\vec{v}_1$ direkt durch Multiplizieren der Schleifenindizes mit den Komponenten des Differenzvektors $\vec{d}$ berechnet. $\vec{v}_2$ ist die Position des nächsten Knoten in $\vec{t}_2$ -Richtung (siehe Abbildung 3.11).

Procedure BuildDiamondMesh ($\vec{S}$: Vector3d of Integer);
  Var
    p, q, r, Δ_x, Δ_y, a, b : Integer;
    $\vec{d}, \vec{v}_1, \vec{v}_2, \vec{t}_1, \vec{t}_2, \vec{t}_3, \vec{t}_4$: Vector3d of Real;

  Begin
    Initialize $\vec{t}_1, \vec{t}_2, \vec{t}_3, \vec{t}_4$; // see Abbildung 3.11

    $\vec{d} := \begin{pmatrix} {}^x\vec{t}_4 \\ {}^y\vec{t}_3 \\ {}^z\vec{t}_1 \end{pmatrix} - \vec{t}_2$; // difference vector (see Abbildung 3.11)

    $\Delta_y := {}^x\vec{S}$; // difference of indices in y-direction

    $\Delta_z := {}^y\vec{S} \cdot \Delta_y$; // difference of indices in z-direction

    for r := 0 to ${}^z\vec{S} - 1$ do
      for q := 0 to ${}^y\vec{S} - 1$ do
        for p := 0 to $\frac{{}^x\vec{S}+1}{2} - 1$ do
          if q is even then
            $\vec{v}_1 := \begin{pmatrix} {}^x\vec{d} \cdot p \\ {}^y\vec{d} \cdot q \\ {}^z\vec{d} \cdot r \end{pmatrix}$; // start- and endpoint

            $\vec{v}_2 := \vec{v}_1 - \vec{t}_2$; // for the actual spring
          else
            $\vec{v}_1 := \begin{pmatrix} {}^x\vec{d} \cdot p \\ {}^y\vec{d} \cdot q - {}^y\vec{t}_4 \\ {}^z\vec{d} \cdot r - {}^z\vec{t}_4 \end{pmatrix}$; $\vec{v}_2 := \vec{v}_1 + \vec{t}_4$;
          end;
          if r is odd then swap ${}^z\vec{v}_1$ and ${}^z\vec{v}_2$;
          a := AddNode($\vec{v}_1$);
          b := AddNode($\vec{v}_2$);
          AddSpring(a, b);
          AddSpring(a, $a - 1$);
          if q is odd then
            AddSpring(b, $b - \Delta_y$);
          else
            AddSpring(a, $a - \Delta_y$);
          end;
          if (r and 1) xor (q and 1) = 1 **then**
            AddSpring(b, $b - \Delta_z$);
          else

```
                    AddSpring(a, a − Δz);
                end
            end
        end
    end
End;
```

Algorithmus 3.3: Prozedur zur Erzeugung eines rechtwinkligen Diamantgitters (aus [166]*).*

Wenn q ungerade ist, dann ist die Berechnung von $\vec{v}_1$ und $\vec{v}_2$ leicht unterschiedlich, weil $\vec{v}_2 = \vec{v}_1 + \vec{t}_4$ (siehe Abbildung 3.10, Knotenindizes 5 ($\vec{v}_1$) und 6 ($\vec{v}_2$), $p = 0$, $q = 1$). Wenn r ungerade ist, dann wird die z-Komponente von $\vec{v}_1$ und $\vec{v}_2$ getauscht (siehe Abbildung 3.10, Knotenindizes 0, 1 und 15, 16).

Mit der Funktion `AddNode` wird ein Knoten zum Gitter hinzugefügt. Der Rückgabewert enthält den Knotenindex. `AddNode` fügt nur Knoten hinzu, wenn p kleiner als ${}^x\vec{S} - 1$ ist. Daher enthält das Beispiel in Abbildung 3.10 fünf, anstatt sechs Knoten in x-Richtung. Die Prozedur `AddSpring` fügt eine Feder zwischen den angegebenen Knoten ein, falls beide existieren und $p > 0$ ist.

3.3.1.3 Anpassen einer Tetraedergeometrie an das Diamantgitter

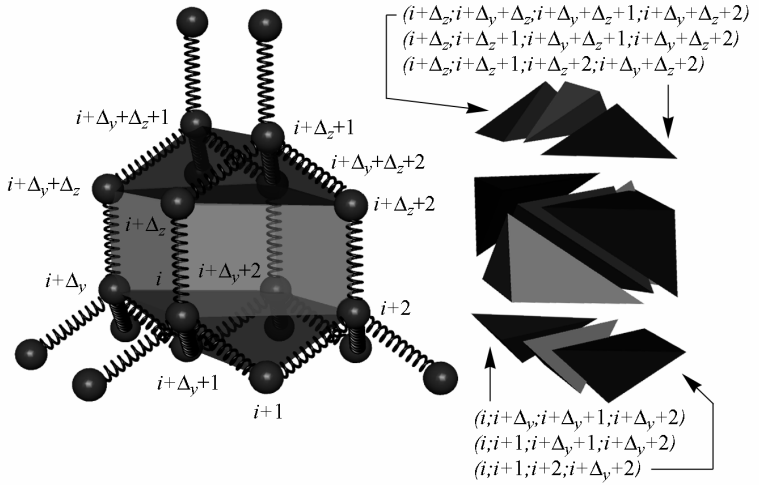

Abbildung 3.12*: Zerlegung der Diamantstruktur in Tetraeder. Links ist ein Feder-Masse System abgebildet, das eine Menge von Tetraedern enthält. Die rechte Seite zeigt eine Partition des Diamantgitters und die Knotenindizes der Tetraederecken (aus* [166]*).*

Mit der Erzeugung eines Diamantgitters existiert ein Feder-Masse System, das verwendet werden kann, um Volumendatensätze zu deformieren. Eine Voraussetzung hier-

für ist die Integration der Geometrie von OpenGL Volumizer in die Struktur des Diamantgitters. Die Struktur der Volumizer Geometrie ist, wie in Kapitel 3.2.2.1 beschrieben, aus Tetraedern aufgebaut, weshalb auch das Diamantgitter in eine Tetraedergeometrie zerlegt werden muss. Hierzu ist es sinnvoll, eine gleichmäßige sich wiederholende Geometrie im Diamantgitter zu bestimmen und jede dieser Geometrien in Tetraeder aufzuteilen. Eine solche sich wiederholende Geometrie ist in Abbildung 3.12 gezeigt, sie besteht aus einem Würfel und zwei Prismen. Der Index *i* zeigt auf den ersten Knoten des Diamantgitters, an dem die Geometrie erzeugt werden soll. Die restlichen Indizes werden über die Δ_y und Δ_z Offsets berechnet. Eine mögliche Zerlegung der gezeigten Geometrie in 11 Tetraeder ist auf der rechten Seite der Abbildung zu sehen.

Um Objekte wie z.B. Organe innerhalb des Volumendatensatzes anzunähern, muss das rechtwinklige Diamantgitter mit dem Objekt geschnitten werden (Clipping), damit das Volumen nur innerhalb des Objektes dargestellt wird. Dies kann über die Festlegung eines Schwellwertes stattfinden, welcher die Oberfläche des Objektes innerhalb der Volumendaten angibt. Algorithmen zur Generierung einer solchen Oberfläche werden in Kapitel 4.2 genauer diskutiert.

Jeder Knoten des Gitters, der nur Eckpunkt von Tetraedern ist, die außerhalb des Objektes liegen, kann aus dem Gitter gelöscht werden, ebenso die dazugehörigen Tetraeder. Die resultierenden Tetraeder schließen das gesamte Volumen des Objektes ein. Abbildung 3.13 (links) zeigt eine Kugel, die durch ein Diamantgitter approximiert wurde. Die Tetraeder sind transparent dargestellt, sodass die ursprüngliche Form der Kugel zu sehen ist.

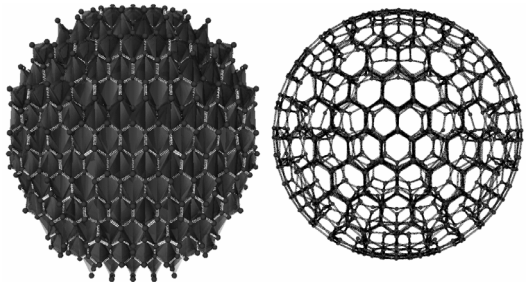

Abbildung 3.13*: Durch Diamantgitter angenäherte Kugel, links: nicht optimiert mit transparent gezeichneten Tetraedern, rechts: optimiert (aus* [166]*).*

Ein Problem hierbei ist allerdings, dass das geschnittene Diamantgitter ein größeres Volumen einschließt als das Volumen der ursprünglichen Kugel. Dies wirft sowohl bei einer Kollisionserkennung als auch bei dem späteren Entfernen von Tetraedern Probleme auf. Deshalb kann es notwendig sein, das Diamantgitter zu optimieren, damit es das exakte Volumen des Objektes erhält. Diese Optimierung wird erreicht, indem die Knoten, die außerhalb des Objektes liegen, auf seine Oberfläche verschoben werden. Die Positionen der übrigen Knoten müssen ebenfalls angepasst werden, sodass alle durch die gespannten Federn auf die Knoten wirkenden Kräfte im Gleichgewicht sind. Diese Opti-

mierung kann mittels des Propagationsalgorithmus der Neuro-Fuzzy Deformation aus Kapitel 2 gelöst werden, der auch bei der Simulation zum Einsatz kommt. Hierbei wird das Feder-Masse System so lange berechnet, bis die Positionen aller Knoten sich nicht mehr verändern. Das Neuronale Netz wird damit als Lösungsmethode für das Optimierungssystem zum Ausgleich der internen Kräfte im Modell benutzt.

Das Feder-Masse System kann nun mit den neuen Knotenpositionen fixiert werden, indem die anfänglichen, ungespannten Längen aller viskoelastischen Elemente durch die aktuellen Längen ersetzt werden. Ein optimiertes Diamantgitter ist in Abbildung 3.13 (rechts) abgebildet. Ein Problem der Optimierung ist jedoch, dass die Struktur etwas Stabilität verliert. Dies liegt daran, dass der Winkel zwischen benachbarten Federn, insbesondere der Außenelemente, nicht mehr exakt 120° beträgt. Trotzdem bleibt das Diamantgitter nahezu intakt [166].

3.3.2 Oberflächenbasierte Volumenmanipulation

Ein Problem des Diamantgittermodells ist, dass eine große Anzahl von Tetraedern benötigt wird, die das Objekt ausfüllen. Dies wirkt sich aber negativ auf die Geschwindigkeit von OpenGL Volumizer aus (vgl. Kapitel 3.2.2.3). Außerdem ist es in vielen Anwendungen schwierig, das Zielvolumen als geschlossene Oberfläche zu segmentieren. Zusätzlich sind die inneren Tetraeder bei relativ opaken Objekten nicht zu sehen, ihre Darstellung ist insofern überflüssig.

Aus diesen Gründen wird im Folgenden ein vereinfachtes Verfahren zur Erzeugung eines Feder-Masse Systems vorgestellt. Es basiert darauf, dass die Simulation des physikalischen Verhaltens der Oberfläche des Objektes für eine elastodynamische lokale Deformierung ausreichend ist [97]. Damit repräsentiert das Feder-Masse System zwar nur die Oberfläche des volumetrischen Körpers, aber diese zweidimensionale Repräsentation lässt sich durch die vorgestellte Prismenerzeugung zu einem Volumen erweitern. Das Verfahren erzeugt aus einem Oberflächenmodell (siehe auch Kapitel 4.1) eine Tetraedergeometrie, die zur Darstellung in OpenGL Volumizer geeignet ist. Die erzeugte Geometrie kann dann durch Veränderung der Tetraederkoordinaten manipuliert werden, was zu einer Deformierung oder Fragmentierung des dargestellten Volumens führt. Das hier vorgestellte Verfahren kombiniert also die Detailtreue des direkten Volume-Rendering mit der aus dem indirekten Volume-Rendering bekannten Zwischenrepräsentation durch Polygonnetze, um eine interaktive Deformation der Volumendaten zu erreichen.

3.3.2.1 Erzeugung einer volumetrischen Geometrie

Es gibt verschiedene Verfahren, um ein Oberflächenmodell eines Objektes innerhalb der Volumendaten zu berechnen. Im medizinischen Bereich handelt es sich bei diesen Objekten um verschiedene Körperorgane bzw. Hohlräume, wie z.B. die Gallenblase oder das Ventrikelsystem des Gehirns. Das grundlegende Verfahren bei der Bestimmung eines Oberflächenmodells aus Volumendaten besteht darin, die Ränder des gesuchten Objektes zu finden und ein das Objekt umschließendes Polygonnetz zu erzeugen. Diese Annäherung der dreidimensionalen Volumendaten durch ein das Objekt umschließendes

Polygonnetz wird Triangulation genannt und entspricht dem Verfahren des indirekten Volume-Rendering (siehe Kapitel 4.2).

Das vorgestellte Verfahren zur oberflächenbasierten Volumendeformation benötigt als Eingabe ein Oberflächenmodell des Volumens, welches in Form einer Dreiecksmenge vorliegen muss (vgl. Kapitel 4.2). Das Verfahren erzeugt aus der Dreiecksmenge eine Prismengeometrie, wofür das Oberflächenmodell geometrisch aufgeblasen oder geschrumpft wird, je nachdem, ob das Volumen innerhalb oder außerhalb der Oberfläche angezeigt werden soll.

Das Verfahren gliedert sich in die folgenden drei Schritte:

1. Normalenberechnung
2. Erzeugung von Prismen aus Dreiecken
3. Zerlegung der Prismen in Tetraeder

Die Berechnung der Flächennormalen erfolgt dabei entweder durch das Kreuzprodukt oder durch die genauere, in [132] (S 499), beschriebene Methode. Die Punktnormalen werden über Mittelung der Flächennormalen der angrenzenden Dreiecke berechnet.

3.3.2.1.1 Prismenerzeugung

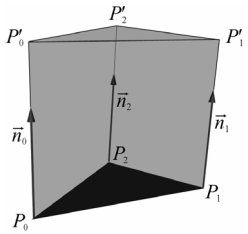

Abbildung 3.14*: Erzeugung eines Prismas aus einem Dreieck unter Benutzung der Punktnormalen.*

In diesem Schritt wird zu jedem Dreieck des Oberflächenmodells ein Prisma erzeugt. Dies ergibt als Gesamtheit eine Prismengeometrie, deren Volumen sich an das Oberflächenmodell anschmiegt. Die Punkte P'_0, P'_1, P'_2 ergeben sich zu (siehe auch Abbildung *3.14*):

$$\begin{aligned} P'_0 &= P_0 + s \cdot \vec{n}_0 \\ P'_1 &= P_1 + s \cdot \vec{n}_1 \\ P'_2 &= P_2 + s \cdot \vec{n}_2 \end{aligned} \tag{3.2}$$

Der Faktor *s* gibt die Länge der zu erzeugenden Prismen an. Dieser Faktor muss vom Benutzer bestimmt werden und hängt von der Skalierung und dem vom Oberflächenmodell eingeschlossenen Volumen ab. Das Vorzeichen des Faktors bestimmt, ob die Oberfläche aufgeblasen oder geschrumpft wird. Hiermit kann bestimmt werden, ob das Volumen außerhalb oder innerhalb der Oberfläche dargestellt werden soll. Dies ist nötig, da Strukturen von außen oder von innen deformiert werden können. Beispielsweise ist der

Gehirnventrikel ein Hohlraum und somit liegt die gewünschte Volumendarstellung bei der Endoskopie außerhalb der Oberfläche.

3.3.2.1.2 Zerlegung in Tetraeder

Anschließend werden die Prismen jeweils in Tetraeder zerlegt, um eine Volumizer-Geometrie zu erhalten. Abbildung 3.15 (links) zeigt eine mögliche Zerlegung eines Prismas und Abbildung 3.15 (rechts) eine aus vier Prismen aufgebaute Geometrie, in der Volumendaten angezeigt werden.

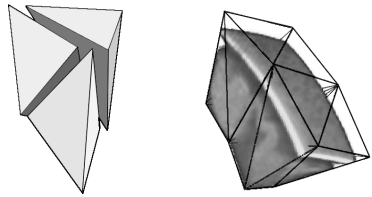

Abbildung 3.15*: Links: Zerlegung eines Prismas in drei Tetraeder. Rechts: Vier zu Prismen erweiterte Dreiecke der Oberfläche mit angezeigten Volumendaten (aus* [185]*).*

Abbildung 3.16 stellt einen Ausschnitt der Schädeldecke eines Menschen und die dazugehörige Tetraedergeometrie dar. Um die visuelle Darstellung zu beschleunigen wird in Kapitel 3.2.3 ein Clipping-Verfahren vorgestellt, welches nur die Tetraeder anzeigt, die aus der Position und Blickrichtung des Betrachters nötig sind.

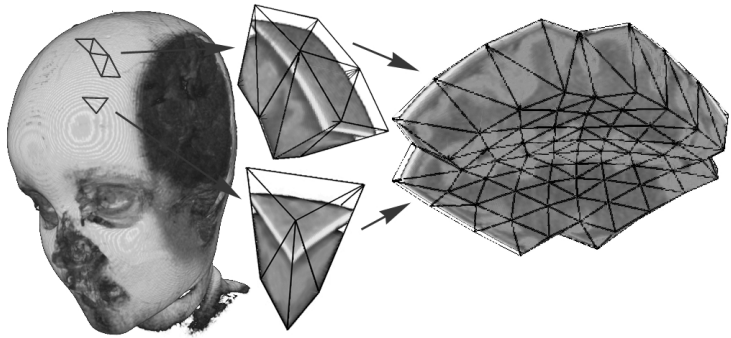

Abbildung 3.16*: Aufbau einer Tetraedergeometrie anhand eines groben Oberflächenmodells des Volumendatensatzes eines Kopfes (aus* [165]*).*

3.3.2.2 Fragmentierung und Deformation

Die vorgestellte Erzeugung von Tetraedergeometrien aus Oberflächenmodellen eignet sich für die Deformation und Fragmentierung von Volumendaten. Dabei ist zu beachten, dass die Deformation und Fragmentierung nicht auf den eigentlichen Volumendaten ausgeführt wird, sondern nur auf der Geometrie. Der visuelle Effekt ist der einer Deformati-

on, die eigentlichen Volumendaten werden nicht verändert, wodurch ein enormer Geschwindigkeitsvorteil gegenüber einer direkten Deformation von Voxelmengen erreicht wird. Hierzu wird die Tatsache ausgenutzt, dass die Deformation des Oberflächenmodells durch ein Feder-Masse System berechnet werden kann. Die Deformation des Oberflächenmodells lässt sich auf die Tetraedergeometrie übertragen, da die Grundflächen der erzeugten Prismen eine eindeutige Zuordnung zu den Dreiecken des Oberflächenmodells besitzen. Ist die Tetraedergeometrie einmal erzeugt, so kann die Geometrie dynamisch an Veränderungen im Oberflächenmodell angepasst werden, ohne dass die Geometrie neu erzeugt werden muss. Eine Deformierung bzw. Fragmentierung des Oberflächenmodells drückt sich in einer Positionsveränderung der Eckpunkte der Dreiecksmenge aus. Diese Veränderung muss in der Tetraedergeometrie ebenfalls durchgeführt werden, wodurch auch das enthaltene Volumen deformiert bzw. fragmentiert wird.

Wie in Kapitel 3.2.2 beschrieben, ist die Zuordnung von Texturkoordinaten zu Tetraederkoordinaten beliebig. Damit kann jedem Tetraedereckpunkt eine beliebige Texturkoordinate zugeordnet werden. Jedes Tetraeder besitzt also zusätzlich zu seinen vier geometrischen Koordinaten der Eckpunkte noch vier Texturkoordinaten, die dem Tetraeder die anzuzeigenden Volumendaten zuordnen. Diese Zuordnung wird im Folgenden dazu verwendet, das dargestellte Volumen entweder zu deformieren oder zu fragmentieren.

3.3.2.2.1 Fragmentierung

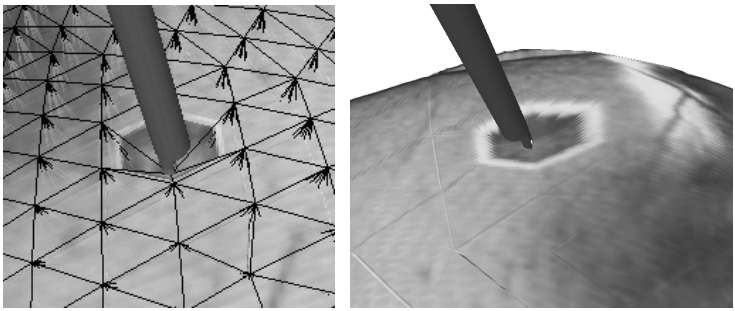

Abbildung 3.17*: Fragmentierung durch Anpassung der Texturkoordinaten am Beispiel eines MRT-Datensatzes. Links mit angezeigter Prismengeometrie, rechts ohne (nach* [165]*).*

Die Fragmentierung basiert auf dem visuellen Eindruck, dass Teile des Volumens herausgeschnitten werden. Dieser visuelle Effekt ist dadurch zu erreichen, dass jede Veränderung von Tetraederkoordinaten auch zu einer Veränderung der Texturkoordinaten führt. Bei der Fragmentierung sind Tetraederkoordinaten und Texturkoordinaten immer identisch zu halten. Das Eindrücken eines Dreiecks des Oberflächenmodells führt dazu, dass die Grundfläche des jeweiligen Prismas ebenfalls eingedrückt wird. Dadurch wird ein vorher sichtbarer Teil des Volumens nicht mehr angezeigt, was zu dem visuellen Eindruck des Fragmentierens führt. Abbildung 3.17 zeigt eine solche Fragmentierung am Beispiel eines MRT-Datensatzes.

3.3.2.2.2 Deformation

Die Deformation basiert im Gegensatz zur Fragmentierung darauf, dass die Texturkoordinaten der Tetraeder nach der Generierung der Tetraedergeometrie nicht mehr verändert werden. Bei der Generierung der Geometrie werden die Texturkoordinaten den geometrischen Koordinaten gleichgesetzt, was eine gleichförmige Zuordnung von Volumendaten zur Geometrie darstellt. Wird nun die Lage eines Dreiecks des Oberflächenmodells verändert, so werden lediglich die geometrischen Koordinaten der betroffenen Prismen (und der zugrunde liegenden Tetraeder) verändert. Damit enthält die dargestellte Geometrie nach wie vor die gleichen Volumendaten, welche aber auf die neue Geometrie skaliert werden. Dadurch entsteht der visuelle Eindruck der Deformation der Volumendaten.

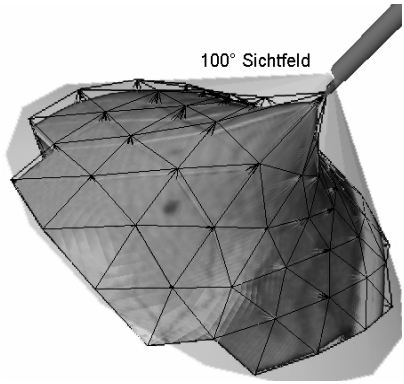

Abbildung 3.18*: Deformation durch gleichbleibende Texturkoordinaten bei veränderter Geometrie am Beispiel eines MRT-Datensatzes. Das sichtbare Volumen aus der Kameraperspektive des Endoskops ist als Kegel eingeblendet. Die Kamera hat eine Objektiv mit 100° Sichtwinkel (nach* [165]*).*

Abbildung 3.18 zeigt eine solche Deformation, es ist sowohl die Tetraedergeometrie als auch das deformierte Volumen zu sehen.

3.4 Ergebnisse

Mittels OpenGL Volumizer und einer geeigneten Geometrie, verknüpft mit einem Deformationsmodell, ist es prinzipiell möglich, Deformationen und Fragmentierungen durchzuführen. Für den Einsatz in einem Chirurgiesimulator ist jedoch die Interaktivität und damit die Geschwindigkeit des Renderings von erheblicher Bedeutung. Relevant ist auch die Qualität der Darstellung, um Operationen zu simulieren. Daher enthält das aktuelle Kapitel Geschwindigkeitstests und weist auf Darstellungsprobleme hin.

3.4.1 Geschwindigkeitsmessungen

Die Geschwindigkeitsmessungen wurden auf einer Silicon Graphics Onyx2 IR mit 512 MB Hauptspeicher und 16 MB Texturspeicher durchgeführt. Das gewählte Volumen hat eine Größe von 256x256x128 Voxeln (bei 2 Byte Speicherbedarf pro Voxel) und passt somit vollständig in den Texturspeicher der ONYX. Dies ist ein wichtiges Kriterium, da das Volumen sonst in Blöcke (siehe Kapitel 3.2.2.2) aufgeteilt wird, was sich durch das notwendige Nachladen der Volumendaten aus dem Hauptspeicher negativ auf die Geschwindigkeit auswirken würde.

Die Darstellung wurde in einem 700x600 Bildpunkte großen Bildschirmfenster ausgegeben. Die Geschwindigkeitsmessungen wurden auf verschiedenen Ansichten durchgeführt. Das gesamte Volumen wurde aus der Gesamtübersicht und aus der endoskopischen Sicht dargestellt. In der endoskopischen Sicht wurden Messungen mit und ohne die Optimierung durch ein „Volume of Interest“ durchgeführt. Das gewählte Kamerasichtfeld hat eine Tiefe von 64 Voxeln. Des Weiteren wurde die Performanz der Volumendeformation durch das in Kapitel 3.3.2.2.2 beschriebene Verfahren gemessen. Tabelle 3.1 zeigt die Ergebnisse der Messungen.

Darstellung	*Texturebenen*	*Anzahl der Tetraeder*	*Bilder pro Sekunde (fps)*
gesamtes Volumen (Übersicht)	1024	5	15
endoskopische Sicht (kein VOI)	1024	5	8
endoskopische Sicht (VOI)	512	5	10
Volumendeformation (Übersicht)	512	900	8
Volumendeformation (endoskopische Sicht)	512	900	3

Tabelle 3.1*: Geschwindigkeitsmessungen von OpenGL Volumizer auf einer SGI Onyx2 IR mit zwei Prozessoren und einem Raster Manager*

Mittlerweile wurden unter konsequenter Ausnutzung der Grafikhardware auf verschiedenen Computerplattformen weitere Verfahren für das dynamische Volume-Rendering entwickelt [39; 92; 169; 191; 210; 212]. Diese Verfahren sind dem in dieser Arbeit vorgestellten Verfahren prinzipiell ähnlich – über die eigentliche Visualisierung hinaus werden allerdings Teile der Deformationsberechnung ebenfalls auf der Grafikhardware ausgeführt. Dadurch ergibt sich eine höhere Geschwindigkeit von etwa 15 fps für ein Volumen von 256x256x256 Voxeln mit 1 Byte pro Voxel[1]. Direkt vergleichbar ist die Geschwindigkeit allerdings nicht, da die in Tabelle 3.1 angegebenen Zeitdauern bereits den Aufwand für die Deformationsberechnung beinhalten. Außerdem liegt dem in der vorliegenden Arbeit beschriebenen Verfahren im Gegensatz zu [39; 92; 169; 191] ein physika-

[1] *Dies entspricht 256x256x128 Voxeln mit jeweils 2 Byte Speicherbedarf. Leider wurde in [169] keine Angabe über die Größe des Darstellungsfensters gemacht, was ebenfalls einen Einfluss auf die Darstellungsgeschwindigkeit hat.*

lisches Deformationsmodell zugrunde, das aufgrund seiner Komplexität eine generell höhere Berechnungszeit erfordert, aber auch realistischere Ergebnisse liefert.

3.4.2 Qualität der Darstellung

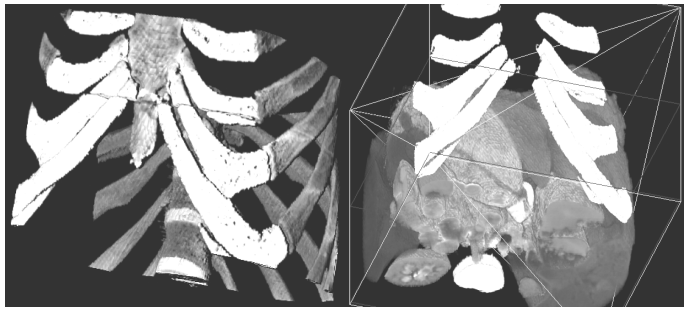

Abbildung 3.19*: Visualisierung eines Ausschnittes des Brustkorbes von Daten des Visible Human (links ohne Organe, rechts mit Organen). Auf der rechten Seite sind die Tetraederkanten der Geometrie von OpenGL Volumizer mit eingezeichnet.*

Die Darstellungsqualität für die statische Visualisierung mit OpenGL Volumizer ist generell sehr gut (siehe Abbildung *3.19*). Die Einsetzbarkeit von OpenGL Volumizer für die Virtuelle Endoskopie muss jedoch im Detail überprüft werden. Dies wurde in einem direkten Vergleich verschiedener Methoden für die Virtuelle Endoskopie im Rahmen dieser Arbeit durchgeführt (siehe Kapitel 5.3).

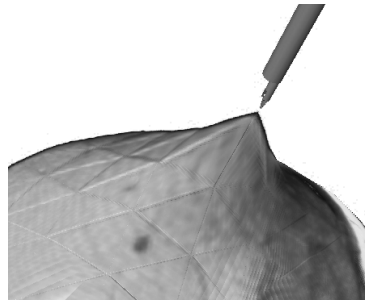

Abbildung 3.20*: Volumendeformation in der Übersicht. Leichte Darstellungsprobleme sind als Tetraedergrenzen beim Rendering sichtbar.*

Bei der Qualität der Darstellung der Volumendeformation hingegen kommt es sehr auf die Kameraposition an. Während die Qualität der Darstellung in der Gesamtansicht des Volumens noch gut ist (siehe Abbildung 3.20), gibt es bei der endoskopischen Sicht erhebliche Probleme. Dies liegt darin begründet, dass die Auflösung der Volumendaten für Nahaufnahmen zu gering ist. Das Endoskop erreicht bei der Deformierung des Volumens

eine Nähe von wenigen Millimetern zum deformierten Gewebe, was bei einer Auflösung der Voxel von ca. 0,5×0,5×1,0 mm^3 dazu führt, dass nur ein sehr verschwommenes Bild dargestellt wird, welches größtenteils aus Artefakten der Interpolationsmethode stammt.

Abbildung 3.21 zeigt die Ansicht eines etwa 13 Voxel durchmessenden Tumors mittels statischem (Mitte) und dynamischen Volume-Rendering aus einer Entfernung von etwa einem Zentimeter. In dieser Ansicht sind an den treppenstufenartigen Strukturen eindeutig einzelne interpolierte Voxel zu erkennen. Die Breite und Höhe des Sichtfeldes beträgt nur etwa 15 × 15 Voxel. Damit ist die Auflösung zu gering für Nahaufnahmen.

Yagel et al. erwähnen in [223] das gleiche Problem für ihren Simulator. Es handelt sich hierbei um kein grundsätzliches Problem der Darstellung, sondern der bildgebenden Verfahren der Medizin. Die geringe Auflösung resultiert je nach Interpolationsverfahren in folgenden Darstellungsfehlern: Treppenstufeneffekte, diffuse Objektgrenzen oder es werden - wie bei dem indirekten Volume-Rendering - vermutete bzw. approximierte Grenzen gezogen. Allerdings werden im Bereich der Magnetresonanztomographie Fortschritte gemacht, die darauf hoffen lassen, dass Patientendaten in höherer Auflösung in naher Zukunft zur Verfügung stehen werden [183; 184].

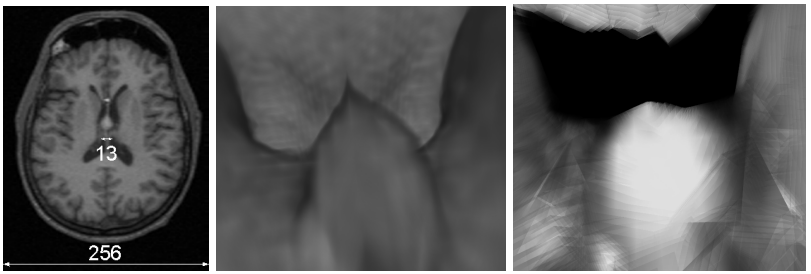

Abbildung 3.21*: Darstellung eines Tumors im Ventrikelbereich des menschlichen Hirns. Links: Axialer Schnitt mit Angabe der Pixelauflösung des Bildes und des Tumors. Mitte: Statische virtuelle endoskopische Ansicht des Tumors durch OpenGL Volumizer. Rechts: Volumendeformation des Tumors mittels dynamischen Volume-Renderings. Die Bildqualität ist wegen der geringen Auflösung des Tumors bei der Deformation sehr schlecht.*

3.5 Diskussion

Ist die lokale Interaktion mit den Volumendaten, z.B. das Entfernen einzelner Volumenelemente immerhin technisch möglich, so stellt die interaktive Volumendeformation mittels Volume-Rendering eine wesentlich höhere Anforderung an die verwendete Hardware. Im aktuellen Kapitel wurde daher eine Methode vorgestellt, die unter konsequenter Ausnutzung der Hardware eines high-end Grafiksystems eine Volumendeformation mittels direktem Volume-Rendering ermöglicht. Unter Benutzung von OpenGL Volumizer in Verknüpfung mit einer geeigneten Tetraeder-Geometrie und einem Deformationsmodell können visuelle Deformationen und Fragmentierungen durch einfaches Än-

dern der Texturkoordinaten durchgeführt werden – eine Veränderung der Volumendaten ist nicht notwendig. Allerdings ist die Anwendung des dynamischen Volume-Renderings auf Bereiche beschränkt, die eine große Übersicht bieten, bzw. bei denen eine große Anzahl von Voxel im Sichtfeld liegen. Für Deformationssimulationen aus großer Nähe, wie beispielsweise bei der Virtuellen Endoskopie im Ventrikelbereich des Gehirns, ist eine Anwendung aufgrund der schlechten Bildqualität nicht möglich. Zudem ist die Geschwindigkeit der Volumendeformation bei der virtuellen endoskopischen Ansicht mit drei Bildern pro Sekunde zu gering, um realistisch Operationen zu simulieren. Die Anwendung von OpenGL Volumizer sollte sich daher auf die statische Darstellung der Virtuellen Endoskopie beschränken, bei der mit 10 Bildern pro Sekunde ausreichende Geschwindigkeiten erreicht werden.

Aktuelle Veröffentlichungen nutzen konsequent die in den letzten Jahren stark weiterentwickelte PC-Grafikhardware, um Voxel-Volumen interaktiv zu visualisieren [56; 93; 95; 125] und auch zu deformieren [39; 92; 169; 191; 210]. Ermöglicht wird dies einerseits durch frei programmierbare Prozessoren auf der Grafikhardware, andererseits durch immer größere Texturspeicher. In Kombination mit einem physikalisch-basiertem Ansatz für die Volumendeformation wird in naher Zukunft eine ausreichende Geschwindigkeit und Qualität des dynamischen Volume-Renderings auf low-cost Hardware möglich sein. Um das dynamische Volume-Rendering auch für die Chirurgiesimulation einsetzen zu können, muss allerdings auch die Auflösung der dreidimensionalen medizinischen Datensätze steigen. Dies ist jedoch noch nicht in Sicht, da hochaufgelöste Datensätze nur in kleinen Teilvolumina medizinisch erforderlich sind [82].

Für die Simulation von Deformationen kann aber auch eine andere Möglichkeit der Darstellung der Patientendatensätze, die Verwendung einer polygonalen Repräsentation der Anatomie, angewendet werden. Hierbei werden die (Organ-)Oberflächen als Menge von Polygonen approximiert. Im Gegensatz zu den direkten Volume-Rendering Verfahren handelt es sich dabei um indirektes Volume-Rendering.

Das indirekte Volume-Rendering ist mit heutiger Grafikhardware sehr effizient und ermöglicht eine Echtzeitdarstellung [61]. Im Gegensatz zum direkten Volume-Rendering wird das Organ allerdings stark vereinfacht und durch einen „hohlen“ Körper dargestellt, selbst wenn das reale Organ kein Hohlkörper ist. Dies macht es schwierig, Eingriffe wie Schnitte realistisch darzustellen, da das Innere des Organs in der Simulation leer erscheint. Dennoch können Hohlorgane oder Membrane durchaus realistisch mit Polygonnetzen visualisiert werden. Hinzu kommt, dass Polygonnetze bislang auf herkömmlichen Computern die einzige Möglichkeit bieten, dynamisches Verhalten, wie Deformationen in ausreichender Qualität und Geschwindigkeit zu simulieren (siehe unter anderem [16; 18; 20; 25; 25; 38; 46; 52; 99; 140; 159-161; 180; 192; 194; 197; 224; 225]).

Das nächste Kapitel enthält daher nach einer kurzen Einführung über Oberflächendarstellungen Triangulierungsmethoden, mit denen die Oberflächen für das indirekte Volume-Rendering erstellt werden können. Außerdem wird eine Texture-Placement Methode vorgestellt, mit der der visuelle Realismus bei der Chirurgiesimulation erheblich verbessert werden kann. Das Kapitelende bildet ein Abschnitt über die Möglichkeiten der Manipulation von Oberflächen.

4 VISUALISIERUNG MITTELS SURFACE-RENDERING

Im Gegensatz zum Volume-Rendering wird das Surface-Rendering bereits länger hardwareunterstützt mit interaktiver Darstellungsgeschwindigkeit durchgeführt. Zeigten 1981 noch erste Vektordisplays Objekte als Linienmodelle (wire-frame) interaktiv an, gab es bereits 1990 Raster Displays, die Objekte aufgebaut aus mehreren hunderttausend Dreiecken, mittels Surface-Rendering und Schattierung darstellten. Heutzutage enthält jeder PC eine Grafikkarte, die dreidimensionales Surface-Rendering mit Shading, Texturemapping, Antialiasing und atmosphärischen Effekten in hohen Bildwiederholfrequenzen ermöglicht. Die Preis-Leistungsentwicklung vollzieht sich so rasant, dass heute einfache Spielekonsolen dasselbe leisten wie noch vor fünf Jahren Grafikrechner der obersten Leistungs- und auch Preisklasse. Optimiert sind diese Systeme jedoch fast ausschließlich für das Surface-Rendering und nicht für das in der Medizin viel eher benötigte direkte Volume-Rendering, obwohl immer mehr Anwendungen auch texturbasiertes Volume-Rendering auf Grafikhardware durchführen, allerdings in einer optisch geringeren Qualität. Durch das indirekte Volume-Rendering, bei dem die zu visualisierenden Strukturen durch Oberflächen angenähert werden, gibt es aber die Möglichkeit, heutige Standardgrafikhardware für die qualitativ hochwertige Darstellung von dreidimensionalen medizinischen Datensätzen zu nutzen. Im folgenden Kapitel werden daher nach einer Einführung über parametrische Oberflächen und Polygongitter Verfahren zum indirekten Volume-Rendering vorgestellt. Zudem bieten Grafikkarten mit dem Texturemapping die Möglichkeit, feine Oberflächenstrukturen wie mit einer Tapete auf sehr viel gröbere Oberflächen aufzubringen. Um diese Technik für die Chirurgiesimulation zu nutzen, wird im aktuellen Kapitel ein Texture-Placement Verfahren entwickelt, mit dem der visuelle Realismus für die Chirurgiesimulation erheblich verbessert werden kann.

4.1 Parametrische Oberflächen und Polygongitter

Glatte Oberflächen werden in vielen Anwendungen der Computergrafik benötigt. Viele Objekte der realen Welt besitzen glatte Oberflächen, was insbesondere auf Organe und andere anatomische Bereiche zutrifft. Bei der Darstellung dieser realen Objekte mit dem Computer kann meist keine mathematische Beschreibung herangezogen werden. Deshalb werden diese Objekte durch Ebenenteile, Kugeln oder andere Formen approximiert, die leicht mathematisch zu beschreiben sind. In der Computergrafik haben sich für die Approximation *parametrische Oberflächen* und *Polygongitter* als besonders geeignet durchgesetzt [61].

Bei parametrischen Oberflächen kann eine glatte Oberfläche durch wenige Kontrollpunkte beschrieben werden. Zwischen den Kontrollpunkten wird die Oberfläche durch parametrische Funktionen höherer Ordnung definiert. Die am häufigsten verwendeten

parametrischen Oberflächen sind die *NURBS* (nonuniform, rational B-splines) [58]. NURBS eignen sich sehr gut für den Einsatz in der Computergrafik, da sie invariant für homogene Transformationen der Kontrollpunkte sind und einen stetigen Übergang zweiter Ableitung an den Kontrollpunkten besitzen. Oberflächenmodelle können mittels NURBS-Flächen durch die Angabe von wenigen Kontrollpunkten definiert werden, ohne kantig zu wirken.

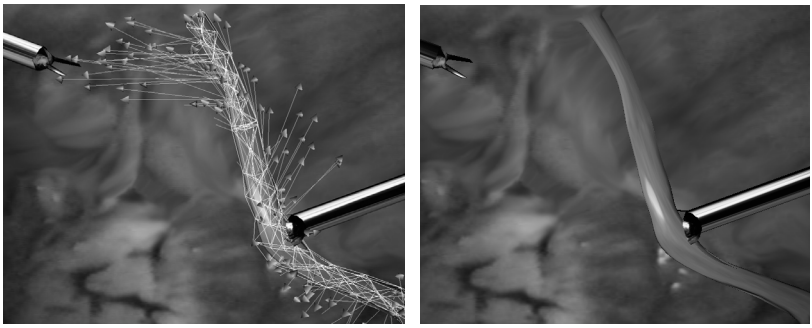

Abbildung 4.1*: Grafische Repräsentation eines Eileiters durch NURBS-Flächen. Links: die zugrundeliegende Gitterstruktur des Feder-Masse Systems ist eingeblendet (aus* [159]*).*

Abbildung *4.1* (links) zeigt ein Modell eines Eileiters, dargestellt als NURBS-Fläche. Das umliegende Gitternetz zeigt das zugrunde liegende Feder-Masse System. Die Positionen der Masseknoten werden gleichzeitig als Kontrollpunkte der NURBS-Fläche benutzt, sodass trotz eines recht groben Feder-Masse Systems Deformationen des virtuellen Eileiters sehr glatt aussehen (Abbildung *4.1* (rechts)).

Für die Simulation von komplexen deformierbaren Oberflächen eignen sich NURBS-Flächen allerdings nur bedingt, da der Vorteil einer glatten Oberfläche durch zusätzliche Berechnungen und einem damit einhergehenden Geschwindigkeitsverlust erkauft wird. Außerdem sind nicht alle dreidimensionalen Oberflächen problemlos in NURBS zu konvertieren, vielfach – insbesondere bei segmentierten Objekten in der Medizin - sind umfangreiche Anpassungen an den Objekten notwendig.

In den meisten Fällen werden aus diesem Grund für den Einsatz in der Chirurgiesimulation Polygongitter verwendet. Nach [61] ist ein Polygongitter wie folgt definiert:

Definition 4.1

Ein Polygongitter ist eine Menge von Punkten, Kanten und Polygonen, sodass jede Kante zu mindestens einem Polygon gehört und von höchstens zwei Polygonen geteilt wird. Ein Punkt gehört zu mindestens zwei Kanten.

Die Zusammensetzung aus einer Menge von Dreiecken stellt den allgemeinsten Fall dar, weil jedes Polygon in einzelne Dreiecke zerlegt werden kann [147]. In weiterer Folge soll nur dieser allgemeine Fall der Dreiecksmengen betrachtet werden.

Die Verknüpfung von Polygongittern mit Feder-Masse Systemen geschieht durch die Zuordnung von Federn zu den Kanten und von Masseknoten zu den Eckpunkten der Polygone. Die Eckpunkte der Polygone werden dabei in regelmäßigen Zeitabständen (meist 20-mal pro Sekunde) mit den Masseknoten des Feder-Masse Systems abgeglichen. Zur Reduktion des Berechnungsaufwands können regelmäßige Dreiecksmengen auch wie in Abbildung *4.2* angegeben zugeordnet werden – die diagonale Federverbindung entfällt dabei.

Polygongitter haben sich mittlerweile als Standardverfahren für Oberflächen durchgesetzt, da deren Darstellung direkt durch günstige Grafikbeschleuniger sehr schnell durchgeführt werden kann. Kanten zwischen den Polygonen können durch Shadingtechniken [68; 143] unterdrückt werden. Dadurch kann auch bei einer begrenzten Anzahl von Polygonen der Eindruck einer runden Oberfläche erzeugt werden. Trotzdem bleiben im Gegensatz zu NURBS-Flächen Kanten an den Grenzen des Objektes sichtbar. Dies wird meistens durch eine hohe Anzahl dargestellter Polygone kompensiert. Dennoch sind Polygongitter mit heutiger Grafikhardware mit einer höheren Bildwiederholfrequenz darstellbar als NURBS-Flächen, die intern fast immer auch durch Polygongitter dargestellt werden.

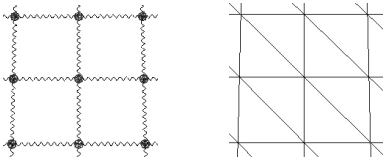

Abbildung 4.2*: Links: Ausschnitt eines ebenen Netzes als Feder-Masse System. Rechts: gleiches Netz mit den Kanten der zugeordneten Dreiecke (in der eigentlichen Modellansicht sind die Polygone ausgefüllt und das Netz als geschlossene Fläche dargestellt).*

Polygongitter haben neben einer schnelleren Darstellung gegenüber NURBS den Vorteil, dass sie für die Visualisierung von Volumendatensätzen in Form von indirektem Volume-Rendering herangezogen werden können. Dafür wird die Oberfläche der darzustellenden Objekte vor der Visualisierung als Polygongitter approximiert. Dieser Prozess wird auch *Triangulation* genannt.

4.2 Indirektes Volume-Rendering mittels Triangulation

Bei der *Triangulation* für das *indirekte Volume-Rendering* werden Objektgrenzen im Zweidimensionalen durch Linienzüge und im Dreidimensionalen durch Polygongitter angenähert. Die Objektgrenzen werden durch die *Segmentierung* bestimmt. In der Bildverarbeitung werden medizinische Volumendatensätze meist zur Reduzierung der Komplexität Schicht für Schicht interaktiv oder halbautomatisch als Linienzüge segmentiert. Schwerwiegernster Nachteil dieser Art der Segmentierung ist das Problem, die einzeln segmentierten Linienzüge zu einer glatten dreidimensionalen Struktur zusammenzufü-

gen. Demgegenüber stellt die Segmentierung von dreidimensionalen Strukturen aus medizinischen Volumendatensätzen und die Erzeugung einer kompakten geometrischen Struktur schon allein wegen des Umfangs der zu bearbeitenden Daten und der Komplexität und Variabilität der zu segmentierenden anatomischen Strukturen ein Problem dar. Darüber hinaus bereiten medizinische Volumendatensätze zusätzliche Probleme wegen fehlender Daten und damit unterbrochener Strukturen durch Sampling-Artefakte, Aliasing und Rauschen.

Die Anwendungsgebiete der Segmentierung in der Medizin liegen allerdings nicht nur bei der Visualisierung anatomischer Strukturen und der chirurgische Planung und Navigation, sondern insbesondere auch bei der Diagnose von Krankheiten [123]. Aus diesen Gründen ist die Segmentierung gut erforscht und eine Vielzahl unterschiedlicher Algorithmen wurden bereits vorgestellt. Eine vollautomatische Segmentierung wird allerdings meist über so genannte Schwellwertverfahren durchgeführt. Es wird dabei angenommen, dass die Grenzen von anatomischen Bereichen sich durch einen gleichmäßigen Intensitätswert auszeichnen. Für viele anatomische Objekte ist dies auch der Fall, z.B. Knochenstrukturen im CT oder der Ventrikelbereich im menschlichen Gehirn, sichtbar im MRT. Viele Details sind allerdings nur durch einen medizinischen Experten interaktiv zu segmentieren. Im Folgenden soll nicht so sehr auf die Segmentierungsalgorithmen eingegangen werden – hier wird eine Segmentierung aufgrund von Schwellwerten angenommen – sondern auf Triangulationsalgorithmen, mit denen aus dem segmentierten Voxelvolumen eine Oberfläche erzeugt werden kann.

4.2.1 Isosurface- und Triangulationsverfahren

Eines der ersten indirekten Verfahren zur Approximation von volumetrischen Objekten war das *Marching-Cubes Verfahren* [114], bei dem vorausgesetzt wird, dass das Volumen als strukturiertes Gitter vorliegt. Das Verfahren besteht darin, eine so genannte *Isosurface* innerhalb der Volumendaten zu berechnen. Eine Isosurface ist eine Oberfläche, die dadurch definiert ist, dass alle Punkte des zugehörigen Volumens, die auf der Oberfläche liegen, denselben Intensitätswert besitzen. Diese Oberfläche wird durch ein Polygongitter angenähert.

Die Oberfläche eines gesuchten Objektes innerhalb der Volumendaten entspricht der Isosurface mit einem korrekt gewählten Schwellwert für die Voxelintensität. Ein Voxel, dessen Intensität größer als die des Schwellwertes ist, liegt innerhalb des Objektes, ein Voxel mit kleinerer Intensität außerhalb. Das Marching-Cubes Verfahren berücksichtigt dabei jede Voxelzelle des gesamten Gitters. Ein Schnitt mit der Objektgrenze innerhalb einer Voxelzelle wird anhand der Intensitäten ihrer Eckpunkte erkannt. Ein Randpunkt ist zwischen zwei Gitterpunkten vorhanden, wenn jeweils ein Punkt den Schwellwert überschreitet und der andere Punkt kleiner oder gleich dem Schwellwert ist. Der Schnittpunkt an der jeweiligen Kante der Voxelzelle kann durch lineare Interpolation ermittelt werden. Jeder Voxel des Gitters enthält für jede seiner Kanten Informationen, ob ein Schnitt aufgetreten ist. Damit ergeben sich $2^8 = 256$ Kombinationsmöglichkeiten für jeden Würfel. Für jeden dieser Fälle wird nun eine Polygonisierung gewählt. Die Gesamtheit der Polygonisierung aller Würfel ergibt das gesuchte Oberflächenmodell.

Der Vorteil des Marching-Cubes Verfahrens liegt in seiner Einfachheit. Ein Problem des Verfahrens ist, dass der gesamte Volumenraum iterativ bearbeitet wird, damit ist sein Laufzeitverhalten $O(n^3)$, wobei n der Seitenlänge des Volumens entspricht. Ein weiteres Problem besteht darin, dass bei medizinischen Daten oft keine klaren Schwellwerte für Organe sichtbar sind, sodass die Oberfläche nicht durch einen einfachen Schwellwert definiert werden kann.

Es gibt verschiedene Ansätze, das Marching-Cubes Verfahren zu optimieren. Ein wichtiger Ansatz zur Beschleunigung ist, nicht den ganzen Volumenraum zu berechnen, sondern nur die Würfel entlang der gefundenen Oberfläche zu untersuchen. Hierzu wird es nötig, einen oder mehrere Startwürfel (engl. seed cube) festzulegen, die sich auf der Oberfläche befinden. Das optimierte Verfahren verfolgt ausgehend von den Startwürfeln die Würfel entlang der Oberfläche, bis die gesamte Oberfläche abgearbeitet ist [24; 222].

Eine weitere Optimierung ist die adaptive Anpassung der Würfelgröße. Das einfache Marching-Cubes Verfahren basiert auf der Würfelgröße einer Voxelzelle, hierdurch werden in relativ gleichförmigen Bereichen der Oberfläche genauso viele Polygone erzeugt wie in stark gekrümmten Bereichen. Es ist sinnvoll, gleichförmige Bereiche mit einer reduzierten Anzahl von Polygonen anzunähern, da dies zu einer höheren Geschwindigkeit der Visualisierung führt. Der einfachste Ansatz führt das normale Marching-Cubes Verfahren durch und reduziert die Polygone des erzeugten Oberflächenmodells in einem zweiten Verarbeitungsschritt [63; 186]. Dies hat allerdings den Nachteil, dass zuerst eine große Anzahl von Polygonen erzeugt wird, von denen viele im zweiten Schritt wieder gelöscht werden, je nach Prozentsatz der Reduktion und Form der Oberfläche.

Ein Verfahren, welches dies umgeht, basiert darauf, dass das Marching-Cubes Verfahren mit einer Würfelgröße von mehreren Voxeln gestartet wird. Diese Würfel werden dann je nach Charakteristik der Oberfläche polygonisiert oder rekursiv verkleinert, bis zu einer minimalen Würfelgröße einer Voxelzelle [189; 213].

Ein weiteres Verfahren, welches in [222] vorgestellt wird, startet mit der Würfelgröße einer Voxelzelle. Es fasst dann Würfel, die sich auf einem gleichförmigen Teil der Oberfläche befinden, zu größeren Würfeln zusammen. Diese Zusammenfassung wird rekursiv durchgeführt, bis sich aufgrund der Krümmung der Oberfläche keine Würfel mehr zusammenfassen lassen. Nach dieser Reduktion erfolgt die Polygonisierung, wobei beachtet werden muss, dass die Ränder unterschiedlich großer Würfel speziell betrachtet werden, da es sonst zu Löchern in der Polygonisierung kommen würde.

Marching-Cubes Verfahren haben sich allerdings generell als nicht besonders geeignet für die Segmentierung und Triangulation medizinischer Datensätze erwiesen, obwohl sie auch heute noch in vielen Applikationen verwendet werden [123]. Das schwerwiegendste Problem ist, dass Marching-Cubes Verfahren schlecht mit fehlenden oder verrauschten Daten umgehen können und die triangulierten Oberflächenmodelle somit in der Regel nicht zusammenhängend sind. Dies ist einerseits problematisch für eine qualitativ hochwertige Visualisierung, andererseits können solche Modelle auch nicht für die Simulation von Deformationen herangezogen werden, weil dafür der Zusammenhang der Polygone, deren Kanten als Federn benutzt werden, von besonderer Bedeutung ist.

Wegen dieser Nachteile wird häufig die *Delaunay Triangulation* eingesetzt [21; 26]. Die dreidimensionale Delaunay Triangulation einer Punktmenge besteht aus allen Tetraedern mit Eckpunkten aus der Punktmenge, die umschreibende Kugeln besitzen, die keine Punkte aus der Punktmenge enthalten. Für die Triangulation medizinischer Volumendatensätze werden die zu segmentierenden Konturen in Zellen aufgeteilt, die die Basis der zu triangulierenden Punktmenge bilden.

Für die Oberflächenrekonstruktion von medizinischen Volumendaten ist allerdings die Delaunay Triangulation auch nicht optimal geeignet, weil sie fehlende Daten nicht extrapolieren kann [51]. Für die spätere Anwendung der Oberflächenmodelle für die Chirurgiesimulation scheinen daher Deformationsmodelle besonders geeignet zu sein [123].

4.2.2 Triangulation mit Deformationsmodellen

Im Gegensatz zu anderen Triangulationsalgorithmen haben *Deformationsmodelle* die Fähigkeit, anatomische Strukturen unter Berücksichtigung von Randbedingungen zu segmentieren und triangulieren, abgeleitet aus den Bildern und der zusätzlichen Verwendung von a priori Wissen über den Ort, die Größe und die Form der Strukturen. Darüber hinaus bieten Deformationsmodelle intuitive Interaktionsmechanismen, die einem Mediziner erlauben, sein Spezialwissen in den Segmentierungsprozess mit einzubringen. Die Segmentierung selbst wird wieder durch Intensitätsschwellwerte durchgeführt.

Deformationsmodelle werden beschrieben durch das Zusammenspiel von Geometrie, Physik und Approximationstheorie: Die Geometrie repräsentiert das zu triangulierende Objekt, die Physik liefert Randbedingungen, wie die Form über den Raum und die Zeit variiert und die Approximationstheorie sorgt für Verfahren, die die Form möglichst gut an den Datensatz anpassen. Oft gehen Deformationsmodelle von einfachen Geometrien aus, deren Form durch Deformation verändert wird, bis sie möglichst gut an die zu segmentierende Region mit konsistenten Intensitätswerten angepasst ist [42; 122; 207]. Die für die Verwendung in der Chirurgiesimulation ausschlaggebenden Vorteile sind, dass nur zusammenhängende Oberflächen erzeugt werden, dass fehlende Daten extrapoliert werden und dass verrauschte Daten keine isolierten Artefakte erzeugen. Hinzu kommt, dass durch die Anwendung der Deformationsmodelle Oberflächenrepräsentationen erstellt werden, die besonders für die spätere Deformationssimulation geeignet sind, da die Verfahren für die Triangulation und Simulation grundsätzlich gleich funktionieren.

Die am häufigsten verwendeten Rekonstruktionsverfahren mit Deformationsmodellen basieren auf parametrischen Repräsentationen wie NURBS oder Finiten Elementen. Eine parametrische Repräsentation stellt eine kontinuierliche Transformation zwischen dem Parameterraum, eingebettet in die Euklidische Ebene $\mathbb{R}^2$, und einer dreidimensionalen Oberfläche dar. Durch die kontinuierliche Repräsentation können geometrische Größen wie Normalenvektoren oder Informationen über die Krümmung überall auf der Oberfläche angegeben werden.

Parametrische Repräsentationen haben mehrere Nachteile, insbesondere kommt es zu Problemen, wenn die Form keine planare, zylindrische oder toroide Topologie besitzt. Beispielsweise wird für Kugeln zumindest ein degenerierter Punkt am Pol erzeugt, wenn eine Ebene auf die Kugel abgebildet wird. Normalenvektoren und Krümmung können in

diesem Punkt nicht stabil berechnet werden. Auch weiterführende Arbeiten konnten dieses Problem nur teilweise lösen, indem sie mehrere parametrische Teilstücke unter Berücksichtigung geeigneter geometrischer Kontinuität zusammensetzen [51].

Um die Nachteile parametrischer Deformationsmodelle zu vermeiden, benutzen andere Deformationsmodelle unstrukturierte Gitter ohne zusätzliche Parametrisierung. Vasilescu und Terzopoulus [204] benutzen beispielsweise nichtlineare Feder-Masse Modelle für die Rekonstruktion. Ein Problem bei diesen Modellen ist, dass die numerische Stabilität stark abhängig von der Topologie des Gitters ist. Wenn nicht genug Federn mit einem Knoten verbunden sind, ist das System unterbestimmt und verschiedene Ruheformen des zu rekonstruierenden Objektes sind möglich. Wenn hingegen zu viele Federn mit einem Knoten verbunden sind, ist das mathematische System überbestimmt und die Deformation des Feder-Masse Modells ist sehr eingeschränkt.

Dennoch ist die Verwendung von Feder-Masse Modellen für die Rekonstruktion sehr interessant, da die Deformationssimulation für die Virtuelle Chirurgie ebenfalls durch Feder-Masse Modelle realisiert wird. Im Folgenden wird daher die Verwendung von angepassten Feder-Masse Modellen in strukturierter Form vorgeschlagen. Eine Methode, die sich als besonders geeignet für die Triangulation von medizinischen Datensätzen erwiesen hat, ist die Verwendung von *Simplex-Gittern*.

Simplex-Gitter sind wie Dreiecksgitter in der Lage, beliebige Topologien zu repräsentieren. Darüber hinaus kann mittels regulierenden Kräften eine hohe Ordnung von geometrischer Kontinuität einfach und effizient gewährleistet werden. Nach [51] ist ein dreidimensionales Simplex-Gitter wie folgt definiert:

Definition 4.1

Ein k-Simplex-Gitter M aus $\mathbb{R}^n$ ist eine $(k + 1)$-Zelle aus $\mathbb{R}^n$.

Die Definition einer k-Zelle kann rekursiv wie folgt angegeben werden:

Definition 4.2

Eine 0-Zelle des $\mathbb{R}^n$ ist definiert als ein Punkt $P \in \mathbb{R}^n$ und einer 1-Zelle des $\mathbb{R}^n$ als eine Kante des $\mathbb{R}^n$, z.B. ein ungeordnetes Paar (P, M) unterschiedlicher Knoten. Eine p-Zelle $(p \geq 2)$ C des $\mathbb{R}^n$ ist rekursiv definiert als eine Vereinigung von $(p - 1)$-Zellen, sodass:

1. jeder Knoten, der zu C gehört, zu p unterschiedlichen $(p - 1)$-Zellen gehört,
2. der Schnitt zweier $(p - 1)$-Zellen entweder leer oder eine $(p - 2)$-Zelle ist.

Eine 2-Zelle ist somit eine Menge von Kanten, die nur einen einzigen Knoten gemeinsam haben und damit ein geschlossenes Polygon im $\mathbb{R}^n$ darstellt. Wichtigste Eigenschaft eines k-Simplex-Gitters ist, dass jeder Eckpunkt $(k + 1)$ Eckpunkten benachbart ist [51].

Prinzipiell ist die Triangulation mittels Simplex-Gittern auf generellen n-dimensionalen Daten möglich. Für die Triangulation von Volumendatensätzen gilt $n = 3$.

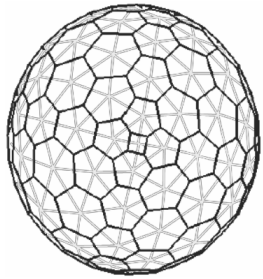

Abbildung 4.3*: Zusammenhang zwischen einem 2-Simplex-Gitter (schwarz) und einer Triangulation.*

Simplex-Gitter selbst sind eigentlich keine Triangulationen, aber sie sind dual zu Triangulationen und können daher leicht in ein Dreiecksgitter zur hardwarenahen Visualisierung transformiert werden (Abbildung *4.3*). Außerdem sind sie lokal leicht zu verfeinern, um beispielsweise starke Krümmungen in dem zu segmentierenden Objekt besser zu approximieren.

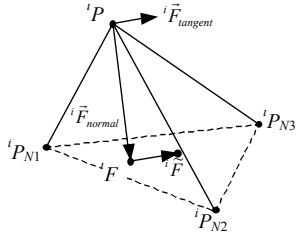

Abbildung 4.4*: Geometrische Interpretation der Tangential- und Normalkraft ausgehend von einem Punkt* ${}^iP$ *und seinen drei Nachbarn* ${}^iP_{N1}$, ${}^iP_{N2}$, ${}^iP_{N3}$ *(nach* [51]*).*

Für die Segmentierung und Triangulation anatomischer Strukturen in dieser Arbeit wird die Deformation der Simplex-Gitter im Gegensatz zu der Arbeit von Delingette [51] durch das in Kapitel 2.3 vorgestellte Modell verwirklicht, wobei die 1-Zellen den Federn und die Eckpunkte den Masseknoten entsprechen. Außerdem werden im Folgenden nur 2-Simplex-Gitter verwendet, die sich als sehr geeignet zur Triangulation medizinischer Volumendatensätze erwiesen haben [14]. Um das Modell zu stabilisieren, werden die internen Kräfte analog zu [51] mit

$$\vec{F}_{\text{int}} = \vec{F}_{tangent} + \vec{F}_{normal} \tag{4.1}$$

definiert. Das Ziel der Tangentialkraft $\vec{F}_{tangent}$ ist die Knotenposition relativ zu seinen drei Nachbarn (bei 2-Simplex-Gittern) zu kontrollieren.

$${}^i\vec{F}_{tangent} = {}^i\tilde{F} - {}^iF = ({}^i\tilde{\varepsilon}_1 - {}^i\varepsilon_1)\,{}^iP_{N1} + ({}^i\tilde{\varepsilon}_2 - {}^i\varepsilon_2)\,{}^iP_{N2} + ({}^i\tilde{\varepsilon}_3 - {}^i\varepsilon_3)\,{}^iP_{N3} \tag{4.2}$$

Dabei geben die metrischen Parameter ε_1^i, ε_2^i und ε_3^i ($\varepsilon_1^i + \varepsilon_2^i + \varepsilon_3^i = 1$) die baryzentrischen Koordinaten von ${}^iF$ bezüglich des Dreiecks (${}^iP_{N1}$, ${}^iP_{N2}$, ${}^iP_{N3}$) an, mit denen

die lokale Form um einen Knoten definiert werden kann. Die metrischen Parameter kontrollieren die Knotenposition bezüglich seiner drei Nachbarn in der Tangentialebene des Knotens (siehe auch Abbildung *4.4*).

Die metrischen Referenzparameter ${}^i\tilde{\varepsilon}_1$, ${}^i\tilde{\varepsilon}_2$ und ${}^i\tilde{\varepsilon}_3$ $({}^i\tilde{\varepsilon}_1 + {}^i\tilde{\varepsilon}_2 + {}^i\tilde{\varepsilon}_3 = 1)$ sind als die entsprechenden baryzentrischen Koordinaten von ${}^i\tilde{F}$ definiert und geben die Knotenverteilung des Simplex-Gitters an. Damit können beispielsweise an Stellen mit starker Oberflächenkrümmung mehr Knoten gesetzt werden. Für regelmäßig verteilte Knoten sollten alle Parameter gleich 1/3 sein.

Die Kraft in Normalenrichtung ${}^i\vec{F}_{normal}$ schränkt die mittlere Krümmung der Oberfläche durch den Simplex-Winkel ein:

$${}^i\vec{F}_{normal} = (L({}^ir, {}^id, {}^i\tilde{\varphi}) - (L({}^ir, {}^id, {}^i\varphi))\ \ {}^i\vec{n}\,, \tag{4.3}$$

wobei der Simplex-Winkel ${}^i\varphi$ definiert ist durch

$${}^i\varphi \in [-\pi, \pi]:$$
$$\cos({}^i\varphi) = \frac{\left\| {}^iC - {}^iO \right\|}{{}^iR} \quad \text{sign}(({}^iC - {}^iO)\cdot {}^i\vec{n})\,, \tag{4.4}$$

$L({}^ir, {}^id, {}^i\varphi)$ definiert ist als

$$L({}^ir, {}^id, {}^i\varphi) = \frac{({}^ir^2 - {}^id^2)\tan({}^i\varphi)}{\sqrt[\varepsilon]{{}^ir^2 + ({}^ir^2 - {}^id^2)\tan^2({}^i\varphi)} + {}^ir} \tag{4.5}$$

mit

$$\begin{aligned} \varepsilon &= 1 \qquad if \quad \left|{}^i\varphi\right| \le \pi/2 \\ \varepsilon &= -1 \qquad if \quad \left|{}^i\varphi\right| > \pi/2 \end{aligned}\,. \tag{4.6}$$

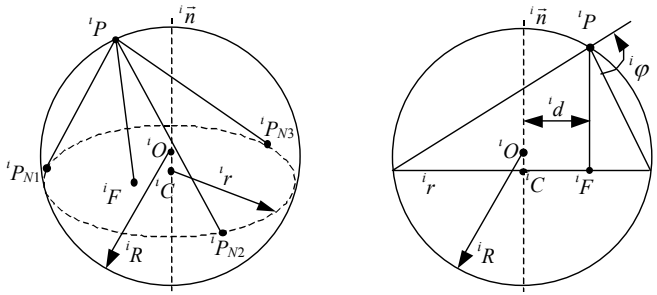

Abbildung 4.5*: Parameterdefinition zur Berechnung des Simplex-Winkels: Links: Die umschreibende Kugel mit dem Radius* ${}^iR$ *und dem umschreibenden Kreis des Radius* ${}^ir$*. Rechts: Projektion der linken Grafik auf die Ebene (*${}^iO$*,* ${}^iC$*,* ${}^iP$*) (nach* [51]*).*

Die Parameter ${}^ir$, ${}^id$, ${}^iC$, ${}^iO$, ${}^iR$ und ${}^i\vec{n}$ sind laut Abbildung *4.5* beschrieben und wie folgt definiert: ${}^ir$ ist der Radius der umschreibenden Kugel des Tetraeders (${}^iP$, ${}^iP_{N1}$, ${}^iP_{N2}$,

${}^iP_{N3}$), ${}^iO$ ist der Mittelpunkt der Kugel und ${}^id$ ist der Abstand zwischen ${}^iF$ und dem Mittelpunkt ${}^iC$ des umschreibenden Kreises des Dreiecks (${}^iP_{N1}$, ${}^iP_{N2}$, ${}^iP_{N3}$).

Der Referenz-Simplex-Winkel ${}^i\tilde{\varphi}$ wird wie in [51] vorgeschlagen durch die C^2 Bedingung als Durchschnitt der Simplex-Winkel der Nachbarknoten eingeschränkt:

$$ {}^i\tilde{\varphi} = \sum_{j \in Q_{iP}^{is}} {}^j\varphi \,, \tag{4.7}$$

wobei Q_{iP}^{is} rekursiv als die Vereinigung von Q_{iP}^{is-1} mit dessen Nachbarn definiert wird (es gilt $Q_{iP}^{0} = \{{}^iP\}$). Die Größe der Nachbarschaft ${}^is$ korrespondiert intuitiv mit der Festigkeit des Deformationsmodells: Je größer der Wert von ${}^is$ ist, umso fester ist das Modell.

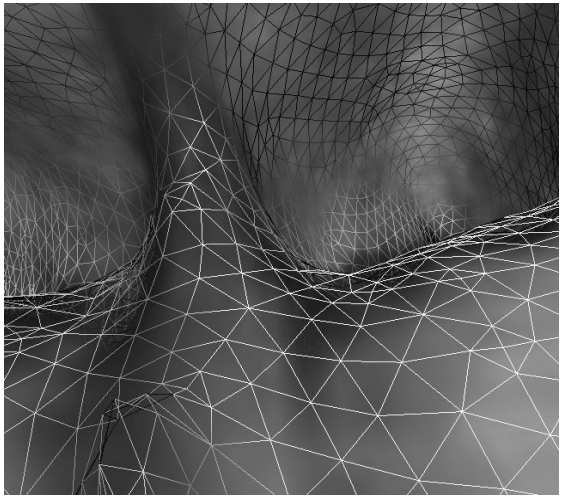

Abbildung 4.6*: Segmentierung der Gehirnventrikel durch ein Simplex-Gitter. Dargestellt ist eine Dreiecksmenge, die aus einer initialen Kugel transformiert worden ist.*

Die Deformation selbst wird unter Anlegen der internen Kraft, beschrieben in Gleichung (4.1), mit dem in Kapitel 2.3 vorgestellten Verfahren simuliert.

Die Segmentierung der Gehirnventrikel im Rahmen der Simulation minimal-invasiver neurochirurgischer Eingriffe wurde mit 2-Simplex-Gittern in der anfänglichen Form einer Kugel unter Berücksichtigung der C^2 Bedingung durchgeführt. Die Größe der Nachbarschaft wurde dabei interaktiv je nach Komplexität des Simplex-Gitters angepasst. Zur hardwarenahen Visualisierung wurde das Simplex-Gitter anschließend in ein Dreiecksgitter umgewandelt. Die Intensitätswerte für die Segmentierung sind durch einen Neurochirurgen interaktiv festgelegt worden. Abbildung 4.6 zeigt die Überlagerung einer Virtuellen Endoskopie mit dem direkten Volume-Rendering und das Gittermodell eines segmentierten Ventrikels. In der linken Bildhälfte kann man in der Nähe des Septum Pel-

lucidums sehen, dass das Verfahren fehlende Daten sehr gut ausgleicht und keine Artefakte entstehen.

Wegen der besseren Übersicht wurde der Ventrikel in Abbildung 4.6 mit nur einer geringen Anzahl von Dreiecken trianguliert. Abbildung *4.7* zeigt das Oberflächenmodell einer Leber, dass mit einer hohen Anzahl von Dreiecken segmentiert wurde. Zu beachten ist, dass die Dichte der Dreiecke von der Krümmung der Oberfläche abhängt.

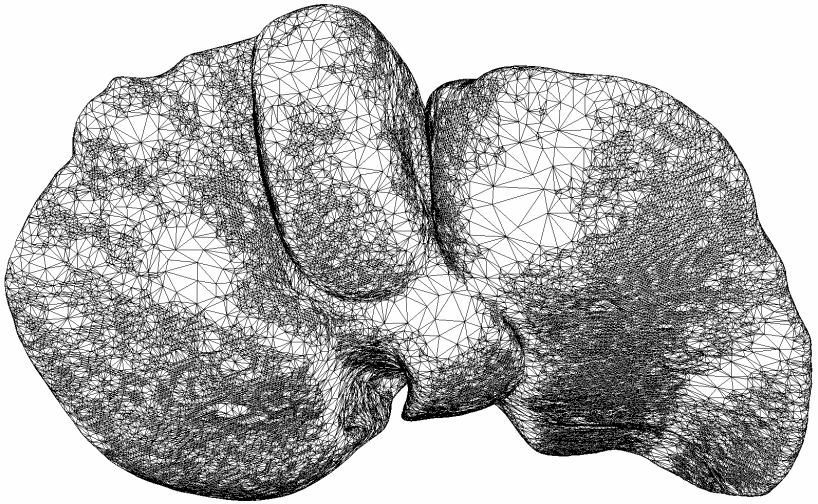

Abbildung 4.7*: Segmentierung einer Leber und Triangulation mit mehr als 200.000 Dreiecken.*

Das Surface-Rendering bietet durch die Visualisierung als Dreiecksgitter im Gegensatz zum hardwareunterstütztem Volume-Rendering die Möglichkeit, den visuellen Realismus durch das Aufbringen von Texturen zu erhöhen. Da herkömmliche Verfahren zwar in vielen Anwendungen durch die Darstellung zusätzlicher Details den Realismus erhöhen können, aber für den Einsatz in Chirurgiesimulatoren eher ungeeignet sind, wird im nächsten Kapitel nach einer kurzen Einführung ein im Rahmen der Implementierung des Neurochirurgiesimulators entwickeltes Texture-Placement-Verfahren vorgestellt.

4.3 Texturemapping

Für die Virtuelle Chirurgie ist der Einsatz qualitativ hochwertiger Texturen von besonderer Bedeutung. Erst dadurch können die Farbe des Gewebes, sowie Oberflächenstrukturen und Blutgefäße in dem Modell sichtbar gemacht werden. *Texturemapping* heißt die Technik, die es ermöglicht, Bildinformation auf ein dreidimensionales Modell zu übertragen [61].

Die interaktive Darstellung komplexer dreidimensionaler Modelle mit qualitativ hochwertigen Texturen kann nur mit moderner Grafikhardware durchgeführt werden. Mittels Texturemapping wird ein zweidimensionales Bild so weit gedehnt oder gestaucht, bis es die Form des dreidimensionalen Modells annimmt. Während dieses Verfahren für einfache Objekte noch gute Ergebnisse liefert, können bei komplexen Modellen teilweise extreme Verzerrungen auftreten.

Heckbert schreibt in einer Zusammenfassung über Texturemapping [75]: „Ein sehr häufiger Kritikpunkt in dem Bestreben nach möglichst wirklichkeitsgetreuen Computermodellen war der Mangel an Oberflächendetails wie zum Beispiel Maserung, Unebenheiten, Kratzer, Schmutz oder Fingerabdrücke. Wirklichkeitstreue braucht Komplexität oder zumindest den Anschein davon. Texturemapping ist eine relativ kostengünstige Möglichkeit, um den Anschein von Komplexität zu erzeugen ohne die Notwendigkeit, jedes kleinste Detail im Modell nachzubilden". Heckbert definiert Texturemapping als eine Shading-Technik zur künstlichen Erzeugung von Bildern, die ein digitalisiertes Bild auf die Oberfläche eines dreidimensionalen Modells legt, vergleichbar mit einer gemusterten Tapete, die an die Wand geklebt wird.

Texturemapping bietet somit die Möglichkeit, die Farbinformation eines zweidimensionalen Bildes auf eine dreidimensionale Oberfläche zu transferieren. Erst durch die Verwendung von Texturen ist es möglich Computermodelle zu schaffen, die kaum von der Realität zu unterscheiden sind. Der Vorteil von Texturemapping besteht darin, dass es die Darstellung von vielen kleinen Details erlaubt, ohne dabei die Zeit, die für die Darstellung des endgültigen Bildes benötigt wird, wesentlich zu verlängern.

In den Anfängen wurde Texturemapping dazu verwendet, um anhand der Intensität eines zweidimensionalen Bildes die Farbe der Oberfläche eines dreidimensionalen Objektes festzulegen. Diese Methode wurde aber im Laufe der Zeit auf so gut wie alle Eigenschaften erweitert, die eine reale Oberfläche aufweisen können. Solche Erweiterungen zum ursprünglichen Algorithmus beinhalten: Modifikation von Oberflächennormalen zur Vortäuschung von Unebenheiten (bump mapping) [23; 173], Veränderung der Transparenz, um Lichtdurchlässigkeit zu erzeugen [62; 88], Festlegung der Rauheit einer Oberfläche, um spiegelnde oder glatte Flächen zu simulieren (u.a. [68; 69; 88; 126]) und die Verschiebung von Teilen der Oberfläche, um auch feine Strukturen in dem Modell wiederzugeben (displacement mapping) [43].

Obwohl das Texturemapping schon seit den Siebzigerjahren bekannt ist und mittlerweile auch von Standardgrafikhardware im unteren Preisbereich unterstützt wird, gibt es bis heute noch kein zufrieden stellendes Verfahren mit dem es möglich ist, ein Bild so auf einem Modell zu positionieren, dass die Merkmale des Bildes mit den entsprechenden Oberflächenstrukturen des Modells übereinstimmen [106]. Besonders bei komplexen Modellen wie sie in der Operationssimulation verwendet werden, war Texturemapping bisher nur mit starken Verzerrungen und einem begrenzten Auflösungsvermögen zu realisieren.

4.3.1 Interaktives Texture-Placement

Durch die Entwicklung eines neuen Verfahrens könnten diese Verzerrungen vermieden werden. Es ist aber zu beachten, dass eine interaktive Darstellung mit hohen Bildwiederholfrequenzen nur durch die Verwendung jener Techniken möglich ist, welche auch hardwaremäßig unterstützt werden. Da ohne Grafikhardwareunterstützung keine interaktive Darstellung möglich ist, muss ein Weg gefunden werden, der durch Verwendung der vorhandenen Möglichkeiten in der Lage ist, die Verzerrungen zu vermeiden.

Ein weiteres Problem betrifft die exakte Positionierung von Bildinformation auf den Modellen. Bei der Erstellung von Modellen für die Virtuelle Chirurgie ist es besonders wichtig, Bilder so zu positionieren, dass die Merkmale und Details der Bilder genau mit denen der Oberfläche übereinstimmen. Gerade bei endoskopischen Operationen ist es dem Chirurgen nur durch besondere Landmarken möglich, sich zu orientieren.

Für die Simulation, die zur Ausbildung von Ärzten und zum Üben besonders schwieriger Operationen herangezogen werden kann, ist es selbstverständlich, dass sich diese Landmarken auch im Modell an der richtigen Position befinden müssen. Da dafür in der Literatur keine zufrieden stellende Methode dargestellt wird [106], wird im Folgenden ein neues Verfahren beschrieben, welches in der Lage ist, die genannten Probleme durch ein verzerrungsfreies Texture-Placement zu beseitigen. Durch die Implementierung des neuen Verfahrens in einem Textur-Editor wird es möglich, Modelle der menschlichen Anatomie zu erstellen, die optisch kaum von der Realität zu unterscheiden sind.

4.3.2 Verzerrungsfreies Texture-Placement

Alle bekannten Texturemapping Verfahren halten sich streng an den Lösungsansatz, der durch die Implementierung von Texturemapping in gängigen Grafikbibliotheken wie zum Beispiel Open Inventor [208] oder OpenGL [219] nahe gelegt wird. Dieser sieht vor, dass ein digitalisiertes Bild aus dem Texturbereich auf die Oberfläche eines dreidimensionalen Objektes transferiert wird. Sobald die Oberfläche des Objektes aber gekrümmt ist, kann diese Zuordnung nur durch die Einführung von Verzerrungen bewerkstelligt werden. Die vorhandenen Verfahren unterscheiden sich im Wesentlichen nur durch die Art und Weise, wie sie die notwendigen Verzerrungen minimieren. Eine optimale Lösung kann jedoch nur darin bestehen, die Verzerrungen erst gar nicht durchzuführen.

Daher wird bei 3D-Paint Programmen das Texturbild erst bei der Bearbeitung des Objektes erstellt. Zum Beispiel schlugen Hanrahan et al. [72] bereits 1990 vor, Farben und Oberflächendetail direkt auf die Polygone aufzutragen und somit jedes Polygon einzeln einzufärben. Der Nachteil des Verfahrens liegt auf der Hand: Um eine hohe Auflösung der Oberflächendetails zu erhalten, müssen die Modelle aus sehr vielen Polygonen bestehen. Der Aufwand, der dadurch bei der Darstellung entsteht, lässt ein interaktives Rendering nicht zu.

Weil komplexer Detailreichtum ohne ein echtes Texturemapping nicht interaktiv darstellbar ist, wurde 1999 ein erweitertes Verfahren entwickelt, bei dem ein Texturbild durch Zeichnen auf der Oberfläche des Modells erzeugt wird [85]. Die Veränderung des

Pinsels im Texturraum (inverse Verzerrung) führt zu einer verzerrungsfreien Darstellung auf der Modelloberfläche. Eine erweiterte Version des Verfahrens, bei dem das Zeichnen auf polygonalen 3D-Modellen nur eine zusätzliche Funktion eines 3D-Modellierungssystem mit einem haptischen Eingabegerät darstellt, wird in [70] beschrieben. Hier kann eine unverzerrte Texturierung des Objektes erreicht werden, auch wenn eine verzerrte Parametrisierung vorliegt. Bei jedem Pinselstrich wird die an der jeweiligen Stelle herrschende Verzerrung berechnet und dann die Farbinformation invers verzerrt in den Texturbereich übertragen. Damit kann ein scheinbar unverzerrtes Texturemapping erzeugt werden.

Bei sehr feiner Musterung mit starkem Kontrast wird aber die verzerrte Parametrisierung durch starke Unterschiede in der Texelgröße sichtbar, eine qualitativ hochwertige Darstellung ist damit unmöglich.

Um ein wirklich verzerrungsfreies Texturemapping zu erreichen, muss ein gänzlich anderer Lösungsweg beschritten werden, ohne dabei auf eine Hardwarebeschleunigung zu verzichten. Anhand eines dreidimensionalen Puzzles kann gezeigt werden, dass es möglich ist, genau diese Forderung zu erfüllen.

Bei einem 3D-Puzzle ist es nicht möglich, alle notwendigen Teile aus einem einzigen Foto so auszuschneiden, dass daraus das gewünschte Objekt aufgebaut werden kann, auf dem das ursprüngliche Bild zu erkennen ist [105]. Geht man allerdings von dem fertig zusammengesetzten dreidimensionalen Puzzle aus, ist es hingegen möglich, es zuerst zu bemalen und es dann wieder in seine Einzelteile zu zerlegen. Legt man die Teile in einer Ebene nebeneinander, kann man diese fotografieren und man erhält damit einen Puzzlebausatz in Fotoform.

Für die Modellierung realer Objekte reicht es allerdings nicht aus, das Puzzle nur zu bemalen, es muss vielmehr möglich sein, auch Fotos auf die Oberfläche zu übertragen. Dies kann durch eine Projektion der Farbinformation erreicht werden, vergleichbar mit einem Diaprojektor, der ein Bild auf eine Statue wirft.

Diese Idee stellt den Hindergrund des neu entwickelten Verfahrens dar, das in den folgenden Abschnitten in seinen Einzelheiten erläutert wird.

4.3.2.1 Zerlegung des Objektes in seine elementaren Dreiecke

Verzerrungsfreies Texturemapping ist dadurch definiert, dass jedem zweidimensionalen Oberflächenelement des dreidimensionalen Objektes ein Texturbereich zugeordnet wird, der die gleiche geometrische Form aufweist. Um eine konstante Texeldichte und somit auch eine konstante Texelgröße über die gesamte Oberfläche zu erreichen, muss auch sichergestellt werden, dass die Zuordnung aller Oberflächenelemente mit demselben Skalierungsfaktor erfolgt.

Der Texturbereich kann in einfacher Weise als Farbinformationsspeicher verwendet werden, indem man das Objekt in seine elementaren Dreiecke zerlegt und diesen dann einen Texturbereich zuordnet, der die gleiche geometrische Form aufweist. Solange der Texturbereich nicht für die Aufnahme eines Bildes, sondern nur als Speicher verwendet wird, kann dieser Vorgang für jedes Dreieck unabhängig von seinen Nachbarn erfolgen.

Zu beachten ist aber, dass jeder zugeordnete Texturbereich einzigartig sein muss, es darf also keine Überlappungen geben.

In den Spezifikationen von OpenGL [188] ist festgelegt, dass für jedes Objekt nur eine einzige Textur verwendet werden darf, deren Größe

$$\begin{aligned} w_s &= 2^n + 2b_s \\ h_s &= 2^m + 2b_s \\ 0 &\le b_s \le 1 \end{aligned} \tag{4.8}$$

betragen muss, wobei w_s die Breite, h_s die Höhe und b_s die Randbreite angibt. Die maximale Größe ist implementierungsabhängig, beträgt aber üblicherweise 1024 × 1024 Pixel. Moderne Grafikhardware, wie sie heutzutage schon in jedem PC vorzufinden ist, besitzt einen Grafikspeicher von mindesten vier Megabyte und wäre damit in der Lage, für ein einziges Objekt insgesamt 16 Texturen mit der Maximalgröße zu verwenden, womit die Darstellungsqualität drastisch verbessert werden könnte. Durch die Zerlegung des Polygongitters wird es möglich, jedes Dreieck als eigenständiges Objekt zu betrachten, womit jedem Dreieck seine eigene Textur zugeordnet werden könnte. Dies ist allerdings nur eine theoretische Möglichkeit, da hochaufgelöste dreidimensionale Modelle mehrere zehntausend Dreiecke aufweisen können und der Texturspeicher auch bei den High-End-Grafikworkstations begrenzt ist. Ausgehend von der verfügbaren Grafikhardware können die Dreiecke aber in mehreren Gruppen zusammengefasst und diese als neue Objekte betrachtet werden. Durch die Zuordnung eigener Texturen zu den Gruppen von Dreiecken kann jeweils die beste Darstellungsqualität durch die Ausnutzung aller verfügbaren Ressourcen gewährleistet werden.

Mit der Zuweisung von Teilen des Texturbereichs zu den Dreiecken des Objektes wird der Texturbereich zu einem Speicher für Farbinformation, der die Darstellung von Bildern auf der Oberfläche des Objektes ermöglicht. Die Zuordnung für alle Dreiecke, welche die unterschiedlichsten Abmessungen aufweisen können, erfolgt unabhängig voneinander. Dabei lässt sich nicht vermeiden, dass Teile des Texturbereichs ungenützt bleiben. Da dies aber zu einer Verschlechterung der Darstellungsqualität führen würde, wird im folgenden Abschnitt eine Methode beschrieben, die in der Lage ist, durch spezielle Anordnung der Dreiecke den ungenützten Texturbereich zu minimieren.

4.3.2.2 Verschnittminimierung

Im letzten Abschnitt wurde gefordert, dass jedem Dreieck des Objektes ein eindeutiger Texturbereich zugeordnet werden muss. Bei der nicht überlappenden Verteilung einer Vielzahl von Dreiecken mit unterschiedlichen Abmessungen in dem quadratischen Texturbereich lässt es sich nicht vermeiden, dass Teile von diesem Bereich ungenützt bleiben. Der Texturbereich ist eine begrenzte Systemressource, welche von der Größe des Speichers der verwendeten Grafikhardware abhängig ist. Die Qualität der grafischen Darstellung wiederum ist von der Größe des verwendeten Texturbereiches abhängig. Werden Teile von dem vorgegebenen Texturbereich durch unzureichende Anordnung der Dreiecke vergeudet, wird dadurch die maximal mögliche Darstellungsqualität herabgesetzt. Es ist daher notwendig, die Dreiecke so anzuordnen, dass die ungenützte Fläche

minimiert wird. Dies ist eine typische Anwendung für Algorithmen, die unter dem Begriff *Verschnittminimierung* bekannt sind [108].

Im Gegensatz zu den üblichen Anwendungsgebieten der Verschnittminimierung besteht die Besonderheit dieser Aufgabenstellung darin, dass zwar alle Dreiecke um den gleichen Skalierungsfaktor vergrößert oder verkleinert werden dürfen, aber die Größe des Texturbereichs genau vorgegeben ist und nicht verändert werden darf. Daher wurde für das Texture-Placement Verfahren ein neuer Verschnittminimierungsalgorithmus entwickelt, der im Folgenden beschrieben wird. Für die Funktionsweise des Algorithmus wird eine einheitliche Beschriftung nach Abbildung *4.8* für jedes einzelne Dreieck festgelegt.

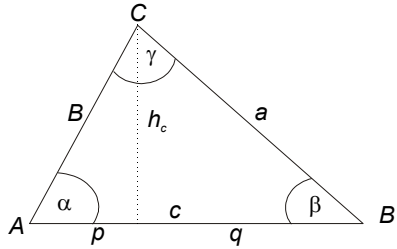

Abbildung 4.8*: Einheitliche Beschriftung für alle Dreiecke. c bezeichnet die längste Seite des Dreiecks, der Fußpunkt der Höhe hc teilt die Seite c in die zwei Abschnitte p und q (nach* [104]*).*

Es ist besonders darauf zu achten, dass die längste Seite jedes Dreiecks immer mit *c* beschriftet wird, dadurch wird erreicht, dass die Winkel α und β immer kleiner als 90° sind, was für die weitere Vorgehensweise von besonderer Bedeutung ist. Jedes Dreieck wird durch die dreidimensionalen Koordinaten der Eckpunkte *A, B* und *C* beschrieben, die anderen Abmessungen errechnen sich durch elementare Vektorrechnung:

$$\vec{BC} = \vec{a} = C - B \qquad a = |\vec{a}| \qquad \alpha = \arccos\left(\frac{\vec{b}\cdot\vec{c}}{b\cdot c}\right) \qquad h_c = b\cdot\sin\alpha$$

$$\vec{AC} = \vec{b} = C - A \qquad b = |\vec{b}| \qquad \beta = \arccos\left(-\frac{\vec{a}\cdot\vec{c}}{a\cdot c}\right) \qquad p = b\cdot\cos\alpha$$

$$\vec{AB} = \vec{c} = B - A \qquad c = |\vec{c}| \qquad \gamma = \arccos\left(\frac{\vec{a}\cdot\vec{b}}{a\cdot b}\right) \qquad q = c - p$$

Die Berechnung der Seitenlängen *a, b* und *c* ist notwendig, um die längste herauszufinden. In weiterer Folge wird nur mehr die Länge *c* sowie die Höhe h_c benötigt. Zusammen mit den Winkeln α und β ist das Dreieck damit vollständig bestimmt.

Im ersten Schritt der Verschnittminimierung werden die Dreiecke nach der Höhe h_c sortiert. Daraufhin wird der Flächeninhalt aller Dreiecke A_i berechnet und zu der Gesamtfläche A_S aufsummiert. Die Relation zu der Fläche des Texturbereichs A_T ergibt eine erste Näherung für den Skalierungsfaktor *f*:

$$A_i = \frac{c \cdot h_c}{2}$$
$$A_s = \sum A_i = \sum \frac{c \cdot h_c}{2} \tag{4.9}$$
$$f = \sqrt{\frac{A_T}{A_s}} = \sqrt{\frac{2 \cdot w_s \cdot h_s}{\sum c_i \cdot h_{ci}}}$$

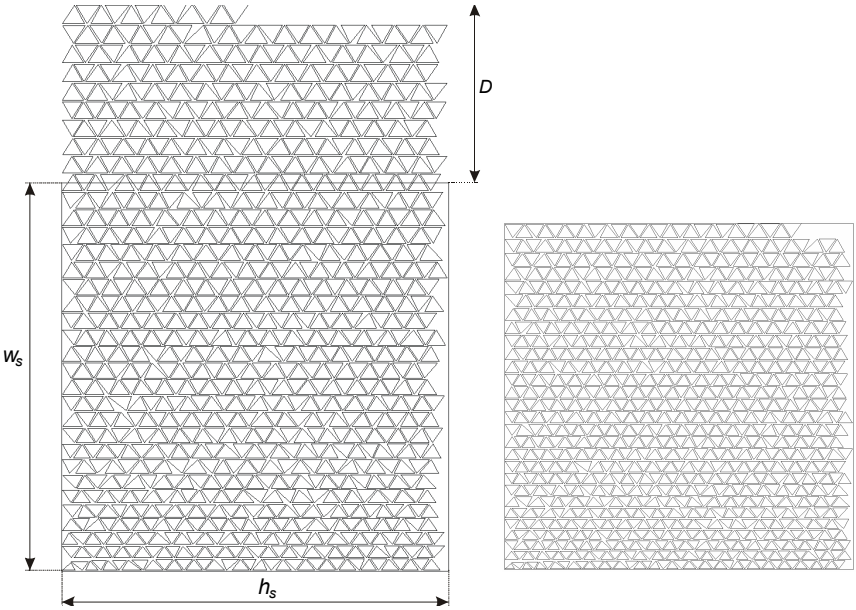

Abbildung 4.9*: Initiale Verteilung der Dreiecke bei der Verschnittminimierung (links). Die Differenz D der vertikalen Ausdehnung der Dreiecke mit der vertikalen Texturgröße ws wird zur Berechnung eines Skalierungsfaktors für die Größe der Dreiecke verwendet. Endgültige Verteilung der Dreiecke nach der Verschnittminimierung (rechts) (nach* [104]*).*

Die Skalierung der Dreiecke geschieht durch Multiplikation der Seite c und der Höhe h_c mit dem Faktor f. Die Fläche des Texturbereiches A_T berechnet sich aus der Breite w_s und der Höhe h_s des verwendeten Texturbitmaps (siehe Abbildung *4.9*). Dabei wird die Anzahl der Pixel in horizontaler und vertikaler Richtung direkt als geometrisches Längenmaß verwendet.

Ausgehend von dem Dreieck mit der kleinsten Höhe h_c werden die Dreiecke zeilenweise so nebeneinander angeordnet, dass der Eckpunkt C abwechselnd entweder oben oder unten liegt. Dabei muss ein minimaler Abstand d_{min} zwischen den Dreiecken eingehalten werden (siehe Abbildung *4.10*), der auf die Rasterung des Bitmaps zurückzuführen ist. Sobald die horizontale Ausdehnung einer Dreieckszeile die vorgegebene Tex-

turbreite w_s überschreitet, wird eine neue Zeile begonnen. Die vertikale Position dieser Zeile wird durch die Höhe des letzten Dreiecks der vorigen Zeile bestimmt, vergrößert um den minimalen Abstand d_{min}.

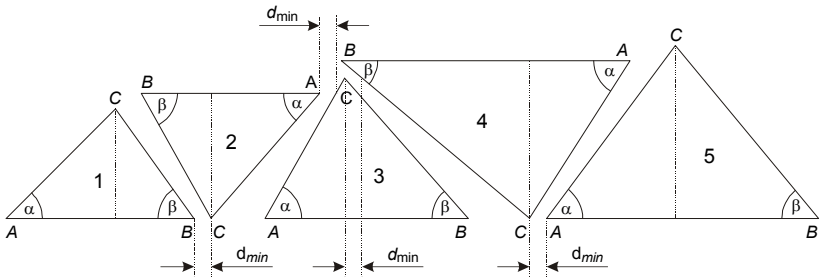

Abbildung 4.10*: Zeilenweise Anordnung der Dreiecke bei der Verschnittminimierung. Die Dreiecke werden so angeordnet, dass der Eckpunkt C abwechselnd oben beziehungsweise unten liegt. Zwischen den Dreiecken muss in horizontaler Richtung immer ein minimaler Abstand dmin eingehalten werden (nach* [104]*).*

Nach der Wiederholung dieses Vorgangs für alle Dreiecke wird sich eine mit Abbildung *4.9* (links) vergleichbare Verteilung ergeben. Die Differenz D zwischen der vertikalen Ausdehnung der Dreiecke und der Texturhöhe h_s wird zur weiteren Berechnung des Skalierungsfaktors f für die Größe der Dreiecke verwendet:

$$f = \sqrt{\left(1 + \frac{D - h_s}{5 \cdot h_s}\right)^{-1}} \tag{4.10}$$

Nach der Skalierung aller Dreiecke mit dem Faktor f muss der gesamte Vorgang der Anordnung der Dreiecke wiederholt werden. Die vertikale Ausdehnung der Dreiecke nähert sich bei jeder Wiederholung schrittweise der Texturhöhe an. Das Abbruchkriterium ist erreicht, wenn die vertikale Ausdehnung mindestens um die halbe minimale Distanz d_{min} kleiner ist als die Texturhöhe h_s. Ein Beispiel der dadurch entstehenden Verteilung der Dreiecke ist rechts in Abbildung *4.9* zu sehen.

Durch die Verteilung der Dreiecke im Texturbereich wird dieser zu einem Speicher für Farbinformation. Um auf dem Objekt damit aber auch Bilder anzeigen zu können, muss eine Möglichkeit geschaffen werden, diesen Speicher mit Farbinformation zu füllen. Im folgenden Abschnitt wird daher ein Verfahren beschrieben, welches hierzu die Projektion von Bildinformation auf die dreidimensionale Oberfläche verwendet.

4.3.2.3 Bildprojektion auf die Oberfläche

Durch die Verteilung der Dreiecke in der Texturebene wurde erreicht, dass der Texturbereich als Speicher für Bildinformation genützt werden kann. Dieser Umstand kann aber erst durch eine Methode nutzbar gemacht werden, welche in der Lage ist, den Speicher für Farbinformation auch in sinnvoller Weise zu füllen. Die Projektion von Bildern auf die Oberfläche liefert hierfür die besten Voraussetzungen. Neben der geforderten

Möglichkeit zur Speicherung von Farbinformation stellt die Projektion auch eine ausgezeichnete Möglichkeit zur exakten Positionierung von Bildern auf dem Objekt dar.

Die Bildpositionierung auf dreidimensionalen Oberflächen wurde bisher meist durch die manuelle Festlegung von Korrespondenzpunkten zwischen dem dreidimensionalen Objekt und der zweidimensionalen Textur gelöst [112; 115; 141; 179]. Die Textur wird dann so auf das Objekt gelegt, dass durch zusätzliche Verzerrungen die festgelegten Korrespondenzpunkte genau übereinander liegen. Bei dem Bestreben nach möglichst verzerrungsfreiem Texturemapping ist diese Vorgehensweise inakzeptabel.

Ein dreidimensionales Malprogramm sollte daher die Möglichkeit nutzen, das Texturbild erst bei der Bearbeitung des Objektes zu erstellen. Wird ein Pinsel auf das Objekt gesetzt, so wird seine Farbe an die entsprechende Stelle in den Texturbereich übertragen, was bewirkt, dass sie sofort am Objekt angezeigt wird. Bilder können auf die gleiche Weise auf das Objekt übertragen werden. Als Quelle für die Farbinformation dient in diesem Fall die Bildprojektion auf die Oberfläche. Die nötige Voraussetzung für die Speicherung der Farbinformation wurde bereits im vorigen Abschnitt geschaffen.

Ein zweidimensionales Bild kann durch Translation und Rotation im dreidimensionalen Raum genau an die gewünschte Position über dem Objekt gebracht werden. Zusätzlich muss es möglich sein, das Bild in seiner Breite und Höhe zu skalieren, um es genau an die lokalen Gegebenheiten des Objektes anzupassen. Mathematisch können diese Transformationen nach Foley et al. [61] als homogene Transformationsmatrix angegeben werden.

Eine Projektion der Farbinformation vom Bild auf die Oberfläche könnte so erfolgen, dass für jeden Bildpunkt eine Gerade in Projektionsrichtung aufgestellt wird, deren Schnittpunkt mit der Oberfläche dazu verwendet wird, dem entsprechenden Texel die Farbe des Bildpunktes zuzuweisen. Durch Wiederholung dieses Vorganges für alle Bildpunkte könnte das Bild über die Oberfläche direkt in die Texturebene übertragen werden. Die Farbinformation des Texturbereiches wird aber umgehend auf der Oberfläche angezeigt. Man hätte damit das Bild auf die Oberfläche projiziert.

Allerdings ist die Berechnung des Schnittpunktes einer Geraden mit einem Objekt aus mehreren tausend Dreiecken sehr aufwändig. Da diese Berechnung für jeden Bildpunkt wiederholt werden müsste, ist es sinnvoller, den Projektionsvorgang umzukehren und somit das Bild von der Oberfläche aus abzutasten. Damit müssen nur mehr Schnittpunkte der Scannerstrahlen mit dem rechteckigen Bild, also Schnittpunkte von Geraden mit einer einzigen Ebene, berechnet werden. Dieser Vorgang kann noch weiter vereinfacht werden, indem dem Bild ein lokales dreidimensionales Koordinatensystem zuordnet wird. Mit Hilfe einer Transformationsmatrix M kann jeder Punkt des Objektes, gegeben im globalen Koordinatensystem, in lokale Bildkoordinaten umgerechnet werden.

$$P_{lokal} = M^{-1} \cdot P_{global} \tag{4.11}$$

Wählt man die Projektionsrichtung genau so, dass sie mit der Flächennormalen des Bildes und somit auch mit der z-Achse des lokalen Koordinatensystems übereinstimmt, dann erhält man den Schnittpunkt eines Scannerstrahls, ausgehend von einem Punkt der

Oberfläche des Objektes, ohne weitere Berechnung durch die *x*- und *y*-Koordinate des transformierten Punktes.

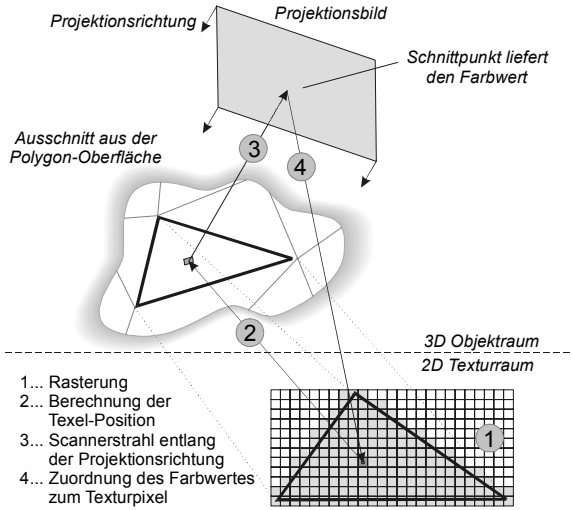

Abbildung 4.11*: Bildprojektion durch Abtastung des Bildes von der Oberfläche. Ausgehend von dem Texturbitmap (1) wird für jedes betroffene Pixel die Position auf der Oberfläche berechnet (2). Von dort wird entlang der Projektionsrichtung der Schnittpunkt mit dem Projektionsbild ermittelt (3) und dessen Farbwert in das Texturpixel übertragen (4) (nach* [106]*).*

Wird das Bild so in das lokale Koordinatensystem gelegt, dass die Breite und Höhe genau der Anzahl der Bildpunkte in der jeweiligen Richtung entspricht, erhält man durch die beschriebene Transformation nicht nur den Schnittpunkt des Scannerstrahls mit dem Bild, sondern auch den Index des betroffenen Pixels. Dieser kann einfach durch Rundung der lokalen Koordinaten *x* und *y* auf die nächst kleinere ganze Zahl berechnet werden.

Als Vorbereitung für den Projektionsvorgang müssen jene Dreiecke ermittelt werden, die innerhalb des Wirkungsbereichs des Projektionsbildes liegen, also jene Dreiecke, deren Scannerstrahlen auch tatsächlich auf das Projektionsbild treffen. Dieses Problem kann auf einfache Weise gelöst werden, indem alle Eckpunkte der Dreiecke getestet werden, ob ihr Scannerstrahl auch wirklich auf das Projektionsbild treffen würde. All jene Dreiecke, deren Ecken keinen einzigen Schnittpunkt mit dem Projektionsbild bewirken, werden während des Scannens nicht berücksichtigt.

Der erste Schritt des Projektionsvorganges (siehe Abbildung *4.11*) besteht darin, alle betroffenen Dreiecke im Texturbereich zu rastern. Bei der Verteilung der Dreiecke im Texturbereich (siehe Kapitel 4.3.2.2) findet nur eine geometrische Zuordnung der Eckpunkte statt. Der Texturbereich entspricht aber einem digitalisierten Bild, bestehend aus einer diskreten Anzahl von Pixel.

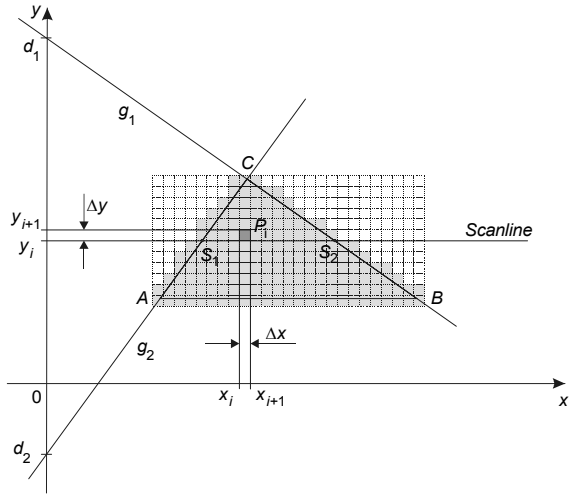

Abbildung 4.12*: Inkrementelle Rasterung eines Dreiecks. Eine Scanline wird in diskreten Abständen Δy vertikal über das Dreieck verschoben. An jeder Stelle yi werden die Schnittpunkte S1 und S2 berechnet. Ausgehend von S1 ergeben sich die horizontalen Pixel durch wiederholte Addition der diskreten Schrittweite Δx (nach* [104]*).*

Für jedes von der Projektion betroffene Dreieck muss die genaue Position der betroffenen Pixel ermittelt werden. Diesen Vorgang nennt man Rasterung. Ein inkrementeller *Scanline-Algorithmus* [61] ist eine schnelle und effiziente Methode, um diese Berechnung durchzuführen (siehe dazu Abbildung *4.12*).

Ein Bitmap besteht aus einer Menge von Bildpunkten, deren Positionen durch die Indizes ihrer Zeilen und Spalten festgelegt wird. Die Koordinaten der Punkte *A, B* und *C* werden durch den Verschnittminimierungsalgorithmus festgelegt und liegen in dem Bereich zwischen (0,0) und (1,1).

Für die Rasterung ist es notwendig, ein neues Koordinatensystem einzuführen, in dem eine Korrespondenz zwischen den Koordinaten der Eckpunkte und den Indizes der einzelnen Bildpunkte hergestellt werden kann. Dies geschieht durch eine einfache Skalierung der Texturkoordinaten auf die Größe des verwendeten Texturbitmaps. Die Positionen der Pixel werden in diesem Koordinatensystem durch die ganzen Zahlen repräsentiert, wobei die Eckpunkte nicht auf diese Zahlenmenge beschränkt sind.

Durch die spezielle Anordnung der Dreiecke kann davon ausgegangen werden, dass die Seite *c* des Dreiecks immer parallel zur *x*-Achse ist. Als Vorbereitung für die Rasterung ist es notwendig, für die Seiten *a* und *b* jeweils eine Geradengleichung aufzustellen, die eine einfache Berechung der Schnittpunkte mit der Scanline ermöglichen.

Zum Beginn der Rasterung wird von dem Dreieck der Punkt mit der kleinsten *y*-Koordinate ermittelt. Durch Rundung dieses Wertes auf die nächst kleinere ganze Zahl erhält man die *y*-Position der ersten Pixel-Reihe. In dieser Höhe wird eine Scanlinie er-

zeugt, deren Schnittpunkte mit den Geraden g_1 und g_2 die horizontale Ausdehnung der Pixelreihe bestimmen. Ausgehend von Schnittpunkt S_1 wird die x-Koordinate des ersten Pixels wiederum durch Rundung auf die nächst kleinere ganze Zahl errechnet. Die weiteren Pixel in dieser Zeile ergeben sich durch wiederholte Addition der horizontalen Schrittweite Δx. Das Ende der Pixelzeile ist erreicht, wenn der x-Wert des nächsten Pixels größer wäre als die x-Koordinate des zweiten Schnittpunktes S_2.

Sobald alle Pixel einer Zeile ermittelt wurden, ergibt sich die nächste Scanline durch Addition der vertikalen Schrittweite Δy. Die Rasterung eines Dreiecks wird abgebrochen, wenn die vertikale Position der nächsten Scanline über den höchsten Punkt des Dreiecks reichen würde. Durch spezielle Wahl des Koordinatensystems ergibt sich für die horizontale und vertikale Schrittweite: $\Delta x = \Delta y = 1$.

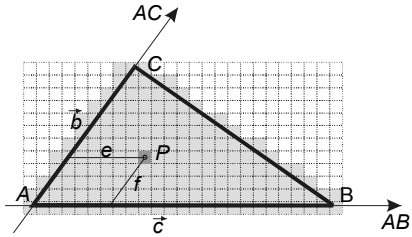

Abbildung 4.13*: Lokales schiefwinkeliges Koordinatensystem für die Berechnung der Position der Texel auf der dreidimensionalen Oberfläche. Die Skizze ist sowohl für den zweidimensionalen Texturbereich als auch für den dreidimensionalen Objektraum gültig, da die Dreiecke in beiden Bereichen die gleiche geometrische Form aufweisen und sich nur durch einen konstanten Skalierungsfaktor in der Größe unterscheiden (nach* [104]*).*

Die Berechnung der relativen Position eines Texels zum Dreieck ist für die Berechnung der entsprechenden Position auf der dreidimensionalen Oberfläche notwendig. Durch die Einführung eines lokalen schiefwinkeligen Koordinatensystems für jedes Dreieck (siehe Abbildung 4.13) lässt sich diese Berechnung auf eine einfache Koordinatentransformation zurückführen.

Die globalen Texturkoordinaten der Punkte *A, B* und *C* werden durch den Verschnittminimierungsalgorithmus (siehe Kapitel 4.3.2.2) festgelegt und sind bekannt. Die Rasterung liefert die globalen Koordinaten der Texel, damit können die lokalen Koordinaten e und f berechnet werden:

$$\begin{aligned} A_{Tex} &= \begin{pmatrix} x_{ATex} & y_{ATex} & 1 \end{pmatrix}^T \\ B_{Tex} &= \begin{pmatrix} x_{BTex} & y_{BTex} & 1 \end{pmatrix}^T \\ C_{Tex} &= \begin{pmatrix} x_{CTex} & y_{CTex} & 1 \end{pmatrix}^T \end{aligned}, \quad \begin{aligned} \vec{b}_{Tex} &= \begin{pmatrix} x_{bTex} & y_{bTex} & 1 \end{pmatrix}^T = C_{Tex} - A_{Tex} \\ \vec{c}_{Tex} &= \begin{pmatrix} x_{cTex} & y_{cTex} & 1 \end{pmatrix}^T = B_{Tex} - A_{Tex} \end{aligned}$$

$$\begin{aligned} P_{Tex} &= \begin{pmatrix} x_{PTex} & y_{PTex} & 1 \end{pmatrix}^T \\ P_{lokal} &= \begin{pmatrix} e & f & 1 \end{pmatrix}^T \end{aligned}, \quad \begin{aligned} P_{Tex} &= M \cdot P_{lokal} \\ P_{lokal} &= M^{-1} \cdot P_{Tex} \end{aligned} \tag{4.12}$$

$$\begin{bmatrix} e \\ f \\ 1 \end{bmatrix} = \begin{bmatrix} x_{cTex} & x_{bTex} & x_{ATex} \\ y_{cTex} & y_{bTex} & y_{ATex} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x_{PTex} \\ y_{PTex} \\ 1 \end{bmatrix}$$

Durch die Berechnung von *e* und *f* aus dem Texturbereich ist es besonders einfach, die entsprechende Position des Texels im dreidimensionalen Objektraum zu berechnen. Die verwendete Transformationsmatrix setzt sich aus den Richtungsvektoren *b* und *c* sowie der Position des Punktes *A* zusammen:

$$\begin{aligned} A_{Obj} &= \begin{pmatrix} x_{AObj} & y_{AObj} & z_{AObj} & 1 \end{pmatrix}^T \\ B_{Obj} &= \begin{pmatrix} x_{BObj} & y_{BObj} & z_{BObj} & 1 \end{pmatrix}^T \\ C_{Obj} &= \begin{pmatrix} x_{CObj} & y_{CObj} & z_{CObj} & 1 \end{pmatrix}^T \\ \vec{b}_{Obj} &= \begin{pmatrix} x_{bObj} & y_{bObj} & z_{bObj} & 1 \end{pmatrix}^T = C_{Obj} - A_{Obj} \\ \vec{c}_{Obj} &= \begin{pmatrix} x_{cObj} & y_{cObj} & z_{cObj} & 1 \end{pmatrix}^T = B_{Obj} - A_{Obj} \end{aligned} \tag{4.13}$$

$$P_{Obj} = \begin{pmatrix} x_{PObj} & y_{PObj} & z_{PObj} & 1 \end{pmatrix}^T, \; P_{Obj} = M_{Tex \to Obj} \cdot P_{lokal}$$

$$\begin{bmatrix} x_{PObj} \\ y_{PObj} \\ z_{PObj} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{cObj} & x_{bObj} & x_{AObj} \\ y_{cObj} & y_{bObj} & y_{AObj} \\ z_{cObj} & z_{bObj} & z_{AObj} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} e \\ f \\ 1 \end{bmatrix}$$

Sobald die Position eines Texels auf der dreidimensionalen Oberfläche berechnet wurde, kann mit der anfangs beschriebenen Methode der Schnittpunkt des zugehörigen Scannerstrahls mit dem Projektionsbild berechnet werden (siehe Schritt 3 in Abbildung 4.11). Daraus ergeben sich auch umgehend die Indizes des betroffenen Bildpunktes und damit auch dessen Farbwert. Durch Speicherung dieses Farbwertes im Texturbereich an der Stelle des Ausgangspunktes wird das Projektionsbild punktweise in den Texturbereich und somit auch auf die Oberfläche übertragen.

Das Projektionsbild wird durch Abtastung von der Oberfläche her auf das Objekt übertragen. Im Prinzip entspricht dieser Vorgang einer diskreten Abtastung, wie sie zum Beispiel auch für das Scannen von Fotos verwendet wird. Die Auswirkungen diskreter Abtastung kontinuierlicher Signale auf die digitale Rekonstruktion werden durch die Informationstheorie als *Aliasing* zusammengefasst und machen sich durch „Treppenstufeneffekte“ bemerkbar. Aus diesem Grund wurde für die Übertragung des Projektionsbildes ein Anti-Aliasing-Filter implementiert [61], mit dem die Bildqualität weiter erhöht werden kann.

4.3.2.4 Darstellungsprobleme bei Verwendung von Mip-Mapping

Bei der Darstellung mittels moderner Grafikhardware können Probleme mit dem Texture-Placement-Verfahren auftreten, wenn eine Filterung, wie heutzutage allgemein üb-

lich, durch *Mip-Mapping* [214] durchgeführt wird. Mip-Mapping wird hauptsächlich verwendet, um störendes Pixelflimmern bei bewegten Objekten zu beseitigen. Sobald ein einziger Bildpunkt des Monitors einen größeren Bereich der Oberfläche des Objektes abdeckt, wird sein Farbwert durch jenes Texel bestimmt, das sich gerade zufällig in der Mitte dieses Bereiches befindet. Durch eine kleine Bewegung des Objektes kann kurze Zeit später bereits ein anderes Texel im Zentrum liegen und somit verändert sich auch die Farbe des angezeigten Bildpunktes sprunghaft. Bei einer kontinuierlichen Bewegung entsteht dadurch Pixelflimmern.

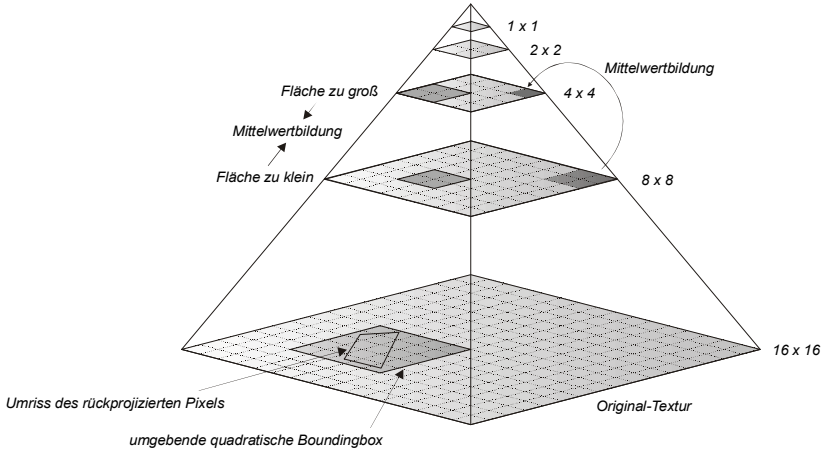

Abbildung 4.14*: Mip-Mapping Pyramide: Ausgehend von der Original-Textur errechnen sich die darüber liegenden Schichten durch wiederholte Mittelwertbildung, dabei werden immer vier auf ein Pixel reduziert. Der Farbwert des rückprojizierten Pixels errechnet sich durch Mittelwertbildung der angenäherten Bounding-Box der darüber liegenden Schichten (aus* [104]*).*

Beim Mip-Mapping wird vor der eigentlichen Anwendung eine Texturpyramide durch wiederholte Mittelwertbildung berechnet (Abbildung *4.14*). Ausgehend von der originalen Textur mit einer Größe von $2^N \times 2^N$ Pixel in der untersten Ebene, wird für jede weitere Schicht die Seitenlänge halbiert, bis in der obersten Schicht nur mehr ein einziges Pixel verbleibt. Der Farbwert eines Pixels in einer der oberen Schichten entspricht genau dem Mittelwert von 2×2 Pixel der darunter liegenden Schicht. Der Farbwert des Pixels in der obersten Schicht wird damit zum Mittelwert über die gesamte Original-Textur.

Zur Filterung während der interaktiven Darstellung ist zuerst für jeden Bildpunkt eine Rückprojektion in die Texturebene notwendig. Hochwertige Filter würden den Farbwert des Pixels durch Mittelwertbildung des betroffenen Texturbereiches errechnen. Bei hochauflösenden Texturen kann dies aber dazu führen, dass für jedes Pixel der Anzeige Hunderte Pixel der Textur gemittelt werden müssen. Beim Mip-Mapping hingegen wird eine Näherung für den Farbwert errechnet, indem in den darüber liegenden Schichten nach einem Bereich gesucht wird, der seiner Bounding-Box am ehesten entspricht. Er-

streckt sich die Bounding-Box beispielsweise über den gesamten Texturbereich, so wird das Pixel aus der obersten Schicht der Pyramide herangezogen. Reicht die Bounding-Box in der Basisschicht nur über ein einziges Pixel, so ist es nicht notwendig, die darüber liegenden Schichten zu betrachten, es wird dann nur der Farbwert des betroffenen Pixels verwendet. Wenn kein Bereich gefunden werden kann, der die Bounding-Box genau abdeckt, wird eine Mittelwertbildung zwischen den Bereichen jener Schichten durchgeführt, deren Teilflächen am ehesten der Bounding-Box entsprechen.

In diesem Fall werden für die Mittelwertbildung auch Teile der Textur herangezogen, die zwar in der Nachbarschaft liegen, aber nicht direkt dem Oberflächenelement zugewiesen sind. Für herkömmliche Texturemapping Verfahren stellt dieser Umstand kein Problem dar, da benachbarte Bereiche der Textur auch zwangsläufig im Objekt benachbart sind. Das im Kapitel 4.3.2 vorgestellte Verfahren ordnet allerdings die Elemente der Oberfläche unabhängig voneinander in dem Texturbereich an und die Nachbarschaftsbeziehung ist daher nicht mehr gewährleistet. Die Filterung mit Mip-Mapping wirkt sich bei der Darstellung durch unterschiedliche Färbung der Randgebiete der Dreiecke aus. Am gesamten Objekt wird dadurch die Gitterstruktur der zugrunde liegenden Dreiecke sichtbar. Abhilfe hierfür kann nur durch die Speicherung redundanter Farbinformation in der direkten Umgebung der einzelnen Dreiecke geschaffen werden. Die Farbe sollte dabei ein Mittelwert der Pixel der angrenzenden Dreiecke darstellen.

Nachdem die theoretischen Grundlagen des Verfahrens für völlig verzerrungsfreies Texturemapping beschrieben worden sind, wird im nächsten Abschnitt deren Anwendung innerhalb eines Textur-Editors dargelegt.

4.3.3 Ein interaktiver Textur-Editor

Für die Erstellung von Computermodellen als Abbild realer Objekte ist die farbliche Darstellung besonders wichtig. Eine nahe liegende Vorgehensweise für die Modellierung besteht darin, ein Objekt aus unterschiedlichen Perspektiven zu fotografieren, um diese Fotos für die Erstellung der Textur zu verwenden. Eine ideale Reproduktion der Farbinformation eines Objektes könnte erreicht werden, wenn es hierbei möglich wäre, den Vorgang des Fotografierens umzudrehen. Dies würde einer Rückprojektion der Bildinformation auf die Oberfläche entsprechen. Diese Umkehrung wäre aber nur möglich, wenn für jedes Foto die genaue Position und Ausrichtung der Kamera sowie die gesamten Abmessungen und Eigenschaften des optischen Systems bekannt wären. Obwohl die technische Umsetzung einer idealen Rückprojektion sicher realisierbar ist, stellt sie für die praktische Anwendung einen viel zu hohen Aufwand dar. Oftmals ist es auch gar nicht möglich, ein Objekt aus beliebiger Perspektive zu fotografieren.

Eine gute Näherung an die ideale Rückprojektion stellt die Parallelprojektion dar. In Kombination mit einer manuellen Positionierung der Projektionsbilder kann damit ebenfalls eine sehr gute Reproduktion erreicht werden. Da hierfür überhaupt keine Information über die Relation der Kamera zu dem Objekt vorhanden sein muss, wird es auch möglich, Bilder zu verwenden, die nicht von dem zu modellierenden Objekt stammen. Dies stellt eine wesentliche Voraussetzung für das geplante Einsatzgebiet des Textur-Editors, die Erstellung qualitativ hochwertiger Texturen für die Operationssimulation, dar. Die

Form eines Organs kann jederzeit durch nicht-invasive bildgebende Verfahren modelliert werden, die Bildinformation kann jedoch nur während einer realen Operation gesammelt werden, sodass für die Erstellung der Texturen nur Bilder von früheren Operationen anderer Patienten verwendet werden können.

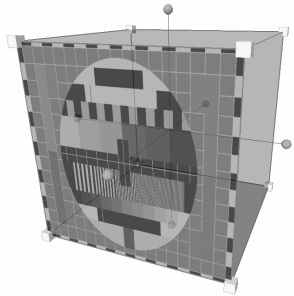

Abbildung 4.15*: Bildprojektor zur Positionierung, Rotation und Skalierung des Projektionsbildes (aus* [104]*).*

Bei der Erstellung von realistischen Modellen ist es besonders wichtig, Bildinformation so auf einem Modell anzuordnen, dass besondere Merkmale des Bildes mit den entsprechenden Oberflächenstrukturen des Modells übereinstimmen. Für die Bildprojektion bedeutet dies, dass es möglich sein muss, das Projektionsbild in beliebiger Lage und Orientierung über dem Objekt anzuordnen. Für eine genaue Anpassung der Größe des Bildes an die Gegebenheiten des Objektes muss eine interaktive Skalierung des Bildes sowohl in der Breite als auch in der Höhe möglich sein. Der Textur-Editor bietet dafür einen frei transformierbaren Bildprojektor.

Grafisch wird der Bildprojektor durch einen halbtransparenten Quader dargestellt, wobei eine seiner Seitenflächen dazu verwendet wird, um das Projektionsbild in Form einer ebenfalls halbtransparenten Textur darzustellen (siehe Abbildung 4.15).

Wie angesprochen, kann die Textur eines Objektes normalerweise nur aus mehreren Bildern unterschiedlicher Kameraperspektiven zusammengesetzt werden. Durch die Überlagerung mehrerer Projektionen auf der Modelloberfläche ergeben sich aber deutlich sichtbare Kanten an den Rändern einer Projektion. Dies wird durch eine Überblendung des Projektionsbildes mit den Farben des betroffenen Bereiches der Oberfläche verhindert. Implementiert ist diese Überblendung durch eine gewichtete Mittelwertbildung, wobei die Gewichtung des Beitrages des Projektionsbildes ausgehend von 0% am Rand des Bildes linear zunimmt, bis sie in einem vorgegebenen Abstand vom Rand 100% erreicht. Der Beitrag der Farbe der Oberfläche verläuft genau entgegengesetzt. Die relative Ausdehnung des Randbereiches, der zur Überblendung verwendet werden soll, kann neben anderen Einstellungen in einem Dialogfenster festgelegt werden.

Ein normales Farbbild wird üblicherweise durch drei Byte pro Bildpunkt kodiert, ein Byte wird dabei für jeweils eine der drei Grundfarben Rot, Grün und Blau verwendet. Moderne Grafikformate, wie zum Beispiel das Portable-Network-Grafik-Format, sind in

der Lage, für jeden Bildpunkt ein weiteres Byte zu speichern. Zusätzlich zu den drei Farbkanälen entsteht dadurch ein weiterer Kanal, der als Alpha-Layer bezeichnet und üblicherweise für die Speicherung der Transparenzinformation verwendet wird. In Bildbearbeitungsprogrammen wird dieser Kanal auch als Maske bezeichnet und kann dazu verwendet werden, um unerwünschte Teile aus Bildern auszublenden oder dem Bild eine Form zu geben, die sich von einem Rechteck unterscheidet. Für die Kodierung der Transparenz steht ein Byte zur Verfügung, das heißt, es können zwischen völliger Durchsichtigkeit und Undurchsichtigkeit insgesamt 256 verschiedene Transparenzstufen durch entsprechende Grauwertgebung festgelegt werden.

Der Bildprojektor des Textur Editors ist in der Lage Bilder zu verwenden, die einen Alpha-Layer beinhalten (siehe Abbildung 4.16). Der Alpha-Wert wird als Gewichtungsfaktor für die Mittelwertbildung zwischen der Farbe der Oberfläche und der des Projektors verwendet. An Stellen völliger Transparenz des Projektors bleibt die Farbe der Oberfläche unverändert, wohingegen bei vollkommener Undurchsichtigkeit die Farbe der Oberfläche gänzlich durch die des Projektors ersetzt wird. Durch die Verwendung des Alpha-Layers wird es dem Anwender ermöglicht, mit einem Bildbearbeitungsprogramm beliebige Überblendungsmasken zu erstellen. Vergleichbar mit der Überblendung im Randgebiet können dadurch sichtbare Kanten durch die Projektion verhindert werden. Zusätzlich ist es möglich, unerwünschte Bildteile auszublenden oder die rechteckige Form des Bildes beliebig zu beschneiden.

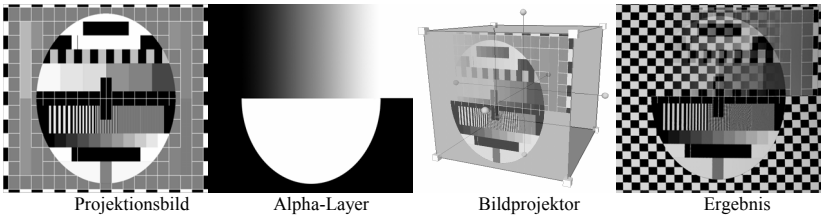

Abbildung 4.16 *Verwendung des Alpha-Layers bei der Bildprojektion. Durch Grauwerte kann im Alpha-Layer die Transparenz eines Bildes festgelegt werden, die im Bildprojektor berücksichtig wird. Das Ergebnis der Projektion auf eine karierte Oberfläche ist im rechten Bild zu sehen (aus* [104]*).*

In Abbildung 4.16 ist eine Maske zu sehen, die mit einem linearen Grauwertgradienten in der oberen Hälfte eine Überblendungsfunktion veranschaulichen soll. In der unteren Hälfte ist durch den Alpha-Layer eine Beschneidungsmaske realisiert, die das zwangsläufig rechteckige Foto auf einen Halbkreis beschneidet.

Zusätzlich zu der Verwendung des Alpha-Layers kann bei der Projektion optional ein Antialiasing-Filter benutzt werden, der die Qualität der Projektion deutlich verbessert (siehe Abbildung *4.17*).

Abbildung 4.17*: Verbesserung in der Darstellungsqualität durch die Verwendung eines Antialiasing-Filters. Bedingt durch die diskrete Abtastung ergeben sich bei der Projektion ohne Filterung teilweise starke Aliasingeffekte (links), welche bereits durch die Verwendung eines einfachen Aliasingfilters deutlich reduziert werden können (rechts) (aus* [105]*).*

Mit Hilfe der Maus kann der Bildprojektor beliebig im dreidimensionalen Raum verschoben, gedreht und skaliert werden. Da durch die Maus aber nur eine zweidimensionale Eingabe möglich ist, muss die Positionierung und die Skalierung in mehreren Schritten erfolgen. Durch die Verwendung eines dreidimensionalen Eingabegerätes wie zum Beispiel das Phantom [120] der Firma Sensable (siehe Abbildung 4.18) mit sechs Freiheitsgraden zur Positionierung, kann dieser Vorgang wesentlich beschleunigt und vereinfacht werden. Der Textur Editor unterstützt die Verwendung des Phantoms als Eingabegerät zur Positionierung des Projektors.

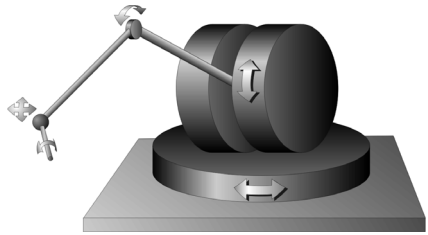

Abbildung 4.18*: Schematische Darstellung des Phantoms der Firma Sensable. Durch die Messung von sechs Freiheitsgraden (durch Pfeile dargestellt) ist es in der Lage, die dreidimensionale Position und Orientierung des Eingabestiftes zu ermitteln (aus* [104]*).*

Sobald ein Projektor in die Szene eingefügt wird, muss das Phantom kalibriert werden. Dabei wird der Anwender aufgefordert, den Eingabearm des Phantoms in die Normalposition zu bringen. Diese wird als relative Nullposition verwendet und kann theoretisch überall sein. Es hat sich aber als sinnvoll erwiesen, den Eingabearm so zu positionieren, dass alle Gelenke im rechten Winkel angeordnet sind. Am Haltestift des Eingabearms befindet sich ein Taster, der als Aktivierungsschalter programmiert ist. Durch Drücken dieses Schalters wird das Phantom als Eingabegerät aktiviert. Der Projektor wird solange synchron mit dem Phantom bewegt, bis der Taster wieder losgelassen wird. In der passi-

ven Phase kann die Maus dazu verwendet werden, die Größe des Projektors zu verändern oder eine Feinjustierung der Position vorzunehmen. Solange das Phantom aktiviert ist, bewegt sich der Projektor immer relativ zur Bildschirmebene. Eine Veränderung der Kameraperspektive bewirkt dann nur eine Drehung des Objektes, nicht aber des Projektors. Solange das Phantom nicht aktiviert ist, wird durch eine Veränderung der Kameraperspektive die gesamte Szene, also auch der Projektor gedreht.

Neben der Bestimmung der Stiftposition in sechs Freiheitsgraden ist das Phantom in der Lage, durch eingebaute Motoren eine Kraftrückkopplung (Force-Feedback) auf den translatorischen Freiheitsgraden durchzuführen. Damit können Widerstände, feste Grenzen und Viskositäten realistisch simuliert werden. Neben dem Phantom, das am weitesten verbreitete Force-Feedback-Gerät für die Chirurgiesimulation, gibt es noch eine Reihe anderer Geräte, die sich für die Chirurgiesimulation eigenen: allen voran die Laparoscopic Impulse Engine der Firma Immersion (San Jose, USA), die u.a. für die Implementierung des gynäkologischen Simulators SUSILAP-G eingesetzt worden ist (siehe Kapitel 5.1). Eine Übersicht über die für die Chirurgiesimulation geeigneten Geräte findet sich in [152].

Unter Benutzung der Kraftrückkopplung ist mit dem Phantom ein weiteres Feature implementiert: die Grenzen des zu texturierenden Objektes können ertastet werden. Dadurch ist es dem Anwender möglich, sich an der Oberfläche durch Fühlen entlang zu bewegen. Komplexe Objekte können damit leichter mit einer Textur versehen werden, als wenn nur die zweidimensionale Repräsentation des Objektes am Bildschirm betrachtet wird.

Für die Implementierung der Kraftrückkopplung ist eine Kollisionserkennung erforderlich, die testet, wann der virtuelle Stift in das Objektmodell eindringt. Ein Eindringen führt zu der aktiven Bewegung des Endeffektors (Stifts) des Phantoms in Richtung der Außengrenzen des Modells. Auch für die im nächsten Abschnitt beschriebene Manipulation von Oberflächenmodellen, beispielsweise Deformieren oder Schneiden mit einem chirurgischen Instrument, wird eine Kollisionserkennung benötigt. Die in dieser Arbeit verwendeten Verfahren sind im Anhang A enthalten.

4.4 Oberflächenmanipulation

Für eine realistische Simulation eines chirurgischen Eingriffs sind grundlegende Interaktionsmöglichkeiten mit den Oberflächenmodellen zu implementieren. Neben der einfachsten Reaktion auf eine Kollision, dem Deformieren des Modells, sind insbesondere das Greifen, Schneiden und Koagulieren relevant, da diese Techniken in einer Vielzahl von chirurgischen Interventionen eingesetzt werden.

4.4.1 Deformieren

Ein möglicher Ansatz für die Erkennung von Berührungen ist die permanente Überwachung aller beteiligten Objekte auf gegenseitiges Eindringen. Für die Virtuelle Chirurgie ist die Erkennung von Kollisionen jedoch nur ein Teil der Problemlösung. Genau-

so wichtig ist die Reaktion auf die Kollision, die bei der Chirurgiesimulation in der Regel in einer Deformation des berührten Gewebes resultiert. Wenn ein Eindringen festgestellt wird, wird dies durch Verschieben der betroffenen Oberflächenelemente des deformierbaren Objektes verhindert, bzw. rückgängig gemacht.

Die Problematik der Kollisionserkennung kann daher in zwei Aufgaben unterteilt werden:

1. Erkennung der Kollision als solches
2. Reaktion auf diese Kollision (z.B. Deformation des berührten Objektes)

Die Erkennung einer Berührung kann durch den Test auf Kollision zwischen den Oberflächenpolygonen und einfachen geometrischen Figuren angenähert werden. Im einfachsten Fall ist die Oberfläche durch eine ungeordnete Liste aller Dreiecke repräsentiert. Hier müssen bei der Prüfung entsprechend alle Elemente getestet werden. Wenn eine Kollision erkannt wird, sind entsprechend die beteiligten Polygone (Dreiecke) des deformierbaren Objektes so zu verschieben, dass die Eindringung rückgängig gemacht wird (Abbildung *4.19*). Neben der Analyse der Position und Ausrichtung der beteiligten Objekte ist dabei auch die relative Bewegungsrichtung zueinander zu berücksichtigen.

Bei der Simulation laufen permanent die Kollisionserkennung und die Simulation des deformierbaren Objektes parallel ab. Die Rate der Kollisionserkennung ist dabei so hoch zu wählen, dass in Abhängigkeit der möglichen Bewegungsgeschwindigkeit die Eindringtiefe beteiligter Objekte nur einen Bruchteil der Größe dieser Geometrien erreichen darf.

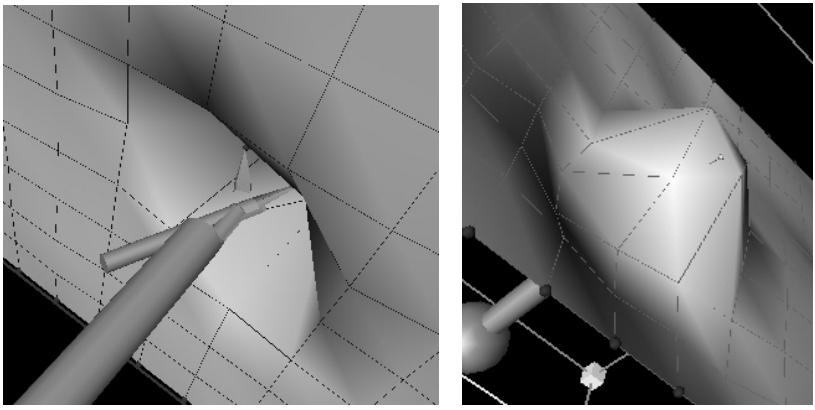

Abbildung 4.19*: links: Kollision eines Op-Tools mit einer elastodynamischen Fläche im Raum. Betroffene Polygone werden dynamisch an die Spitze der Blätter verschoben; rechts: gleiche Deformation, von der Rückseite betrachtet (aus* [91]*).*

Bei (konstanter) Druckausübung eines Objektes auf ein deformierbares, arbeiten grundsätzlich Simulation und die Kollisionserkennung gegeneinander – dies kann zu ei-

nem entsprechenden „Zittern“ der Oberfläche führen, welches verhindert oder zumindest gemildert werden kann durch:

- Temporäre Fixierung der aktuell berührten Oberflächenelemente (Unterdrückung der Simulation): Dabei besteht die größte Schwierigkeit zu erkennen, wann die Fixierung wieder aufgehoben werden soll. Eine Analyse der Richtungen der Kraftvektoren in den festgesetzten Knoten ist dafür sinnvoll. Zeigen die Kraftvektoren in die entgegengesetzte Richtung des virtuellen Instruments, sind die Knoten wieder freizusetzen.
- Totale Synchronisation zwischen Simulation und Kollisionserkennung: Simulation und Kollisionserkennung funktionieren mit der gleichen Frequenz. Dies führt in der Regel wegen des Aufwands der Kollisionserkennung zu Problemen mit der Ausführungsgeschwindigkeit.
- Große Dämpfung des deformierbaren Modells: Die Knoten sind sehr träge, hohe Beschleunigungen sind ausgeschlossen. Dies ist meist bei Membranen in flüssigkeitsgefüllten Räumen gegeben.

Neben der einfachen Deformation des Modells ist das Greifen mit einer Zange von Bedeutung. Oftmals geschieht dies in Zusammenhang mit dem Reißen des virtuellen Gewebes bei Einsatz einer Biopsiezange.

4.4.2 Greifen und Reißen

Das Greifen des virtuellen Gewebes geschieht prinzipiell ebenso wie die einfache Deformation mit temporärer Fixierung der an der Kollision beteiligten Knoten. Solange die Zange geschlossen ist, werden die Knoten permanent fixiert und deren Position analog der Position der Zange geändert, wobei nicht die absolute Position gesetzt, sondern nur die Positionsdifferenz zwischen zwei diskreten aufeinander folgenden Zeitschritten berücksichtigt wird. Beim Öffnen der Zange wird die permanente Fixierung aufgehoben. Das Greifen wird behandelt wie eine einfache Deformation.

Bei Oberflächenmodellen kann aber über das Greifen noch eine weitere chirurgische Funktion simuliert werden: das Reißen. Da Oberflächenmodelle kein Inneres besitzen, die Modelle also hohl sind, können nicht wie beim Volume-Rendering einfach graphische Elemente abgetragen werden. Beim Herausreißen von Polygonen aus dem Oberflächenmodell würde ein Loch zurückbleiben, durch das man in das Innere des Modells sehen könnte. Unter Benutzung der Funktionen des Greifens und der Deformation kann jedoch ein Reißen simuliert werden: Falls beim Greifen die internen Kräfte der beteiligten Federn eine Grenzkraft überschreiten, werden die beteiligten Knoten wie beim Öffnen der Zange freigesetzt. Allerdings werden die Ruhepositionen, also die Positionen der Masseknoten im Ruhezustand, der beteiligten Knoten entgegengesetzt zur Oberflächennormalen verschoben. Bei der Simulation werden die Knoten dann zu der neuen Position hingezogen und es verbleibt ein sichtbares Loch. Zusätzlich wird die Textur im Bereich des Loches abgedunkelt und in der geschlossenen Zange ein Polyeder visualisiert, das die Größe des so entstandenen Lochs besitzt (Abbildung *4.20*). Damit wird der Eindruck erweckt, Gewebe aus der Oberfläche herausgetrennt zu haben.

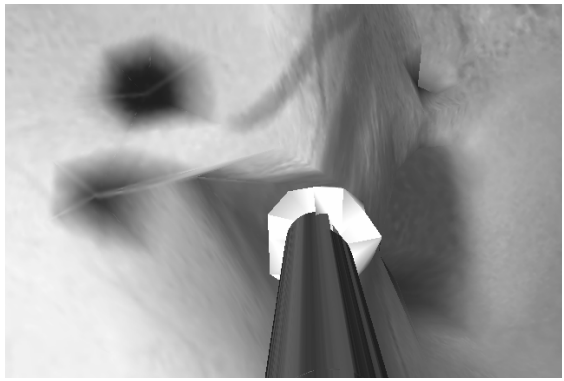

Abbildung 4.20*: Virtuelle Fragmentierung von Oberflächenmodellen: Simulation eines Loches durch Verändern der Ruhepositionen der Masseknoten und Visualisierung eines Polyeders in der geschlossenen Biopsiezange.*

Obwohl Organe, die durch Oberflächenmodelle simuliert werden, hohl sind und daher ein Schnitt nicht realistisch wirken würde, besteht auch Bedarf für die Simulation von Schnitten in Oberflächenmodellen, beispielsweise bei der Simulation von dünnen Membranen oder Kapseln. Daher werden im nächsten Kapitel die grundlegenden Abläufe beim Schnitt von Oberflächenmodellen beschrieben.

4.4.3 Schneiden

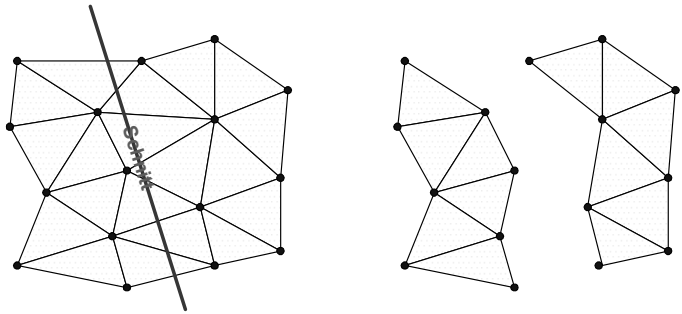

Abbildung 4.21*: Oberflächenschnitt durch Entfernung von Elementen (aus* [91]*).*

Ein Problem beim Schneiden von Oberflächenmodellen ist der Realismus bzw. die Analogie zum realen Eingriff. Organe, die durch Oberflächenmodelle erzeugt werden, sind hohl: Bei einem Schnitt würde man in das hohle Innere sehen und der Realismus des simulierten Schnittes wäre zerstört. Deshalb ist das Schneiden von Oberflächenmodellen insbesondere bei der Simulation von Membranen relevant. Membrane sind auch in der

Realität sehr dünn und eine Simulation dieser dünnen Struktur ist sehr realistisch durch Oberflächenmodelle möglich.

Eine Möglichkeit für das Schneiden von Oberflächenmodellen ist die Entfernung einer Schicht ganzer Elemente im zu trennenden Bereich (Abbildung *4.21*). Für Volumenmodelle wurde Entsprechendes in [46] vorgeschlagen.

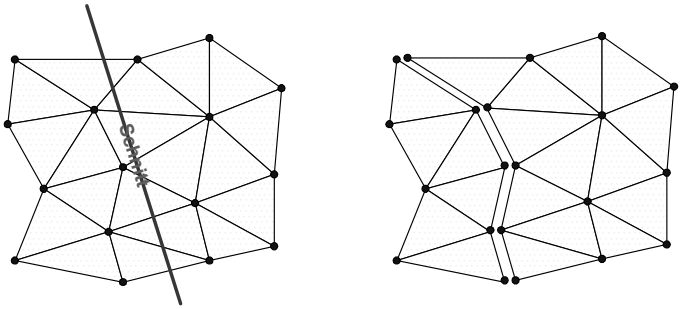

Abbildung 4.22*: Oberflächenschnitt durch Auftrennen von Verbindungen entlang der Kanten (aus* [91]*).*

Bei einer geringen Größe der Teilelemente im Verhältnis zur Ausdehnung des Schnittes ist dieses Verfahren aufgrund der einfachen Implementierbarkeit als durchaus zweckmäßig zu betrachten. Für kurze Schnitte im Verhältnis zur mittleren Polygongröße, wie sie durch eine Schere oder ein Skalpell entstehen, ist dieses Verfahren aufgrund der unrealistisch unmittelbar entstehenden Lücke im Objekt jedoch ungeeignet.

Eine zweite Möglichkeit zum Trennen virtueller Modelle ist das Unterbinden des Zusammenhaltens aneinander angrenzender Objekt-Elemente (Abbildung *4.22*). Nachteilig ist hier zu erwähnen, dass der ausgeführte Schnittverlauf sich bei dieser Methode zwangsläufig an den Kanten (bzw. Oberflächen) der Elemente orientieren wird.

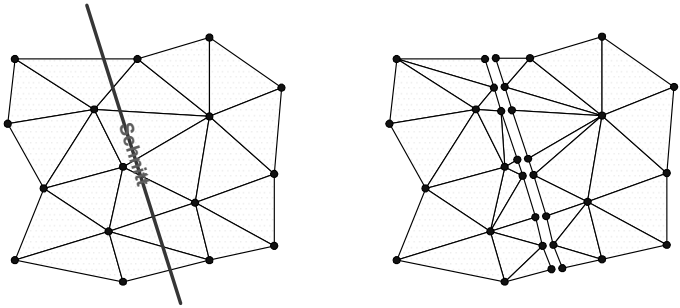

Abbildung 4.23*: Oberflächenschnitt durch Teilung der Elemente (aus* [91]*).*

Die dritte, und am besten geeignete Methode, ist die echte Teilung der Elemente entlang der Schnittlinie [153; 154; 162]. Für Volumenmodelle werden entsprechende Verfahren in [22] vorgestellt. Im ersten Schritt werden hier alle Elemente, die vom Schnitt berührt werden, aus dem Objekt entfernt. Im zweiten Schritt werden neue kleinere Elemente entlang der Schnittlinie im Sinne einer Objektgrenze wieder eingefügt (Abbildung *4.23*).

Die Zerteilung von Modellen bei der Chirurgiesimulation kann als Ursache die Anwendung von entsprechenden Werkzeugen haben oder eine Destruktion kann auch mit dem Überschreiten einer mechanischen Grenzspannung innerhalb des Objektes durch Deformation begründbar sein. Die Implementierung und Realisierung der Möglichkeit eines entsprechend provozierten Risses unterscheidet sich nicht wesentlich von Schnitten, die durch entsprechende Werkzeuge dem Modell beigebracht werden [28].

Wichtig ist in jedem Fall, dass die Summe der Knotenmasse vor und nach dem Schnitt gleich ist. Dies erreicht man durch eine anteilige Reduktion der Masse der benachbarten Knoten:

$$m_{neu} = \frac{\sum_{i \in \mathrm{N}(neu)} m_i}{\left|\mathrm{N}(neu)\right| + 1} \tag{4.14}$$

$$m_i = m_i \frac{\left|\mathrm{N}(neu)\right|}{\left|\mathrm{N}(neu)\right| + 1} \qquad \forall \qquad i \in N(neu)\ ,$$

wobei m_{neu} die Masse des neuen Knotens angibt und N(*neu*) die Menge der Nachbarn des neuen Knotens darstellt. Die Parameter der Federn sollten analog der benachbarten Federn gesetzt werden. Sinnvoll ist in diesem Zusammenhang das Setzen einer Vorspannung der Federn durch Reduktion der Ruhelänge. Damit wird erreicht, dass der Schnitt aufklafft, sich also langsam erweitert.

Im nachfolgenden werden spezielle Schneideverfahren, die für die Anwendung in der Virtuellen Chirurgie besonders geeignet sind, vorgestellt. Um die Komplexität des Oberflächenmodells (d.h. die Anzahl der Polygone) möglichst gering zu halten und dennoch den ausgeführten Schnittverlauf nicht allzu sehr von dem vorgegebenen abweichen zu lassen, ist das Schneiden durch die Teilung der Polygone das optimale Verfahren. Bei der Realisierung sind dabei folgende Vorgehensweisen denkbar:

sukzessiv geführter Schnitt entlang der Oberfläche

In diesem Fall wird permanent die Position eines Zeigegeräts (z.B. ein Laser) dazu verwendet, um entlang der Bewegungslinie den Schnitt stetig voranzutreiben. Bei rascher Bewegung des Zeigegeräts wird durch Geradeninterpolation sichergestellt, dass keine Unterbrechungen entstehen. Ein Beispiel für einen sukzessiv geführten Schnitt ist in Abbildung *4.24* dargestellt. Der Start des Schnitts wird registriert und anschließend wird synchron zur Bewegung des Schneidewerkzeuges der Schnitt durch die Oberfläche vollzogen.

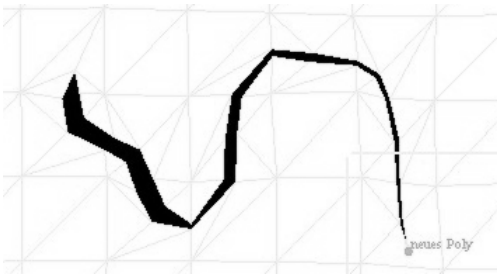

Abbildung 4.24*: Beispiel eines sukzessiv geführten Schnitts (aus* [91]*).*

Geradliniger Schnitt von einem definierten Anfangs- zu einem vorgegebenen Endpunkt

In diesem Fall wird zwischen den beiden Extrempunkten geradeninterpoliert und geschnitten.

Beim Schneidevorgang ist generell zu beachten, dass nicht nur Polygone getrennt werden, sondern das Schneiden auf der Basis von Deformationsmodellen, im Folgenden *Meshes* genannt, bestehend aus Masseknoten, Federn und Polygonen, durchgeführt wird. Die Masseknoten und Federn bestimmen die dynamischen Eigenschaften der Geometrie und die Polygone, die unmittelbar mit den Knoten verbunden sind, die Flächen und damit das Aussehen des Meshes.

Die nachfolgenden Erklärungen beschränken sich auf die Veränderungen des Netzes durch Schneidevorgänge aus der Sicht der Knoten und Polygone. Die Auswirkungen auf die Federn sind unmittelbar durch die Auftrennung von Seitenkanten der Polygone gegeben und werden bei der Implementierung berücksichtigt.

4.4.3.1 Schnitt durch Vorantreiben entlang der Oberfläche

Interpretation des Anfangspunktes:

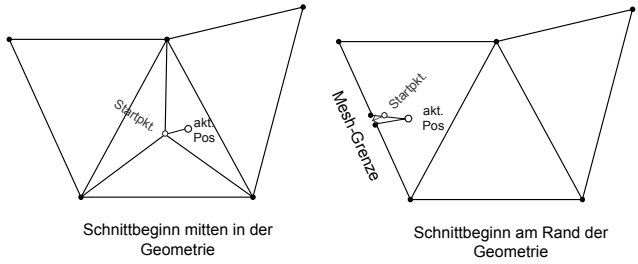

Abbildung 4.25*: Varianten des Schnittbeginns in Abhängigkeit des Abstands zum Mesh-Rand (aus* [91]*).*

Wenn der Beginn des Schnittes innerhalb der Oberflächengeometrie liegt, kommt es zur Zerschlagung des betroffenen Dreiecks – der Schnitt beginnt dann exakt an der vorgegebenen Stelle. Falls sich der Startpunkt sehr nah an der Grenze des Meshes befindet (durch einen einstellbaren Schwellwert bestimmt), dann wird der Schnittpunkt direkt durch eine Normalenprojektion an die Dreieckskante verlegt und diese wird aufgetrennt (Abbildung *4.25*).

Vorantreiben des Schnitts:

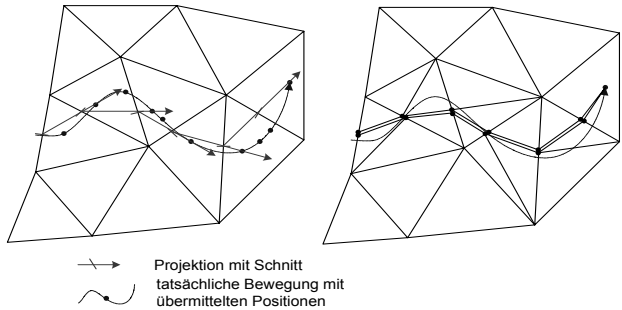

Abbildung 4.26*: Der erste Stützpunkt nach dem Dreieckswechsel bestimmt die Projektionsrichtung für den Schnitt (aus* [91]*).*

Durch die Überwachung der Bewegung des Schneidewerkzeugs, werden permanent neue Koordinatenpunkte entlang der Meshoberfläche geliefert. Der Schnitt wird von Dreieck zu Dreieck geradlinig vorangetrieben (Abbildung *4.26*).

Das Oberflächenmodell kann dabei durchaus gekrümmt im Raum liegen. Es wird nicht vorausgesetzt, dass sich die Werkzeugbewegung exakt auf der Oberfläche vollzieht – es wird auf jeden Fall der Koordinatenpunkt des Schneidewerkzeuges normal auf die jeweils betroffene Dreiecksfläche projiziert.

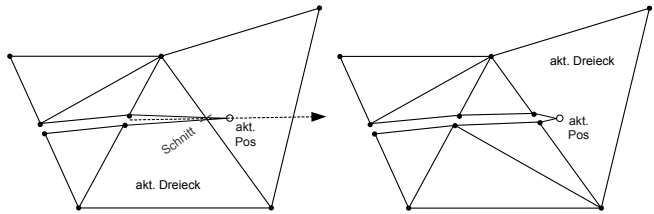

Abbildung 4.27*: Ausführung des Dreieckschnittes. Die übermittelte aktuelle Position (akt. Pos.) liegt außerhalb des aktuellen Dreiecks. Dies führt dazu, dass der Schnitt vollzogen und das Dreieck gewechselt wird (aus* [91]*).*

Der Schnitt wird geradlinig durch das Dreieck immer erst dann vollzogen, wenn dieses verlassen wird. Es werden zwei neue Stützknoten eingefügt, die Feder der Seitenkan-

te wird unterbrochen und vorgespannt, weitere Federn werden entlang des Schnittes ergänzt. Zusätzlich wird noch das im Allgemeinen entstandene Viereck in zwei Dreiecke zerschlagen (Abbildung *4.27*). Die Vorspannung der Federn führt zu einem Aufklaffen des Schnitts, ähnlich wie beim Auftrennen von Gewebe.

Wenn die aktuelle Position nicht im benachbarten Dreieck liegt, werden interpolierte Zwischenpunkte in jedem überstrichenen Dreieck entlang einer gedachten geraden Verbindung berechnet.

Beim geradlinigen Schnitt zwischen vorgegebenen Start- und Endpunkt wird diese Interpolation für alle notwendigen Zwischenpunkte vollzogen.

Abschluss des Schnittes:
Für den Abschluss des Schnittes existieren mehrere Möglichkeiten:

- Automatisches Ende: wenn eine Position außerhalb des Meshs übergeben wird, wird der Schneidevorgang automatisch beendet – der Rand wird aufgetrennt (freigeschnitten). Dies gilt natürlich auch dann, wenn der Schneidevorgang in einem bereits vollzogenen Schnitt mündet.
- Manuelles Ende: hier wird analog zum Schneidebeginn das Dreieck zerschlagen und exakt am Endpunkt ein einzelner Knoten eingefügt, bei welchem der Schnitt schließlich endet (vgl. Beginn des Schnittes...).
- Abbruch: Bei kritischen Situationen während des Schnittes (z.B. Positionen entfernen sich zu sehr von der Oberfläche, Koordinatenpunkte-Pfad kann nicht ermittelt werden,...) wird dieser abgebrochen – es wird versucht „noch alles zu retten, was zu retten möglich ist“, d.h. es wird versucht, den Schnitt an der letzten gültigen Position regulär abzuschließen.

4.4.3.2 Schnitt mittels Dreieck

Das klassische Schneidewerkzeug in der Chirurgie ist die Schere, welche sich näherungsweise als schneidendes Dreieck darstellen lässt. Bei dieser Schneidetechnik werden alle Polygone des Oberflächenmodells geschnitten, die mit dem Schneidedreieck kollidieren.

Wenn ein Schnitt nahe an der Grenze des Oberflächenmodells begonnen wird - dies gilt auch für bereits existierende Schnittgrenzen - so kommt es zu einer vollständigen Durchtrennung. Vergleichbares gilt auch für das Schnittende: wenn der Schnitt über den Rand des Oberflächenmodells hinausgeht, dann wird dieser vollständig getrennt und das Schneiden beendet.
Für die Schnittberechnung werden folgende Voraussetzungen angenommen:

- Die zu schneidende Fläche muss als Dreiecksmenge vorliegen. Neben dem allgemeinen Problem, dass sonst ein Netzelement keine ebene Fläche mehr darstellt, kommt es sonst - vor allem beim Schneiden durch ein Polygon - zu einer wesentlichen Aufwandserhöhung. Da jedes Polygon in eine Menge von Dreiecken zerlegt werden kann, stellt dies keine Einschränkung dar [147].

- Der Schnitt kann beim sukzessiven Schneiden nicht in das unmittelbar zuvor geschnittene Dreieck zurückgeführt werden, was ebenfalls zur Reduktion des Berechnungsaufwands beiträgt.
- Der Schnitt ist innerhalb eines Dreiecks auf jeden Fall geradlinig. Dies stellt in der Virtuellen Chirurgie bei den im Vergleich zum Gesamtobjekt sehr kleinen Dreiecken kein Nachteil dar.
- Interpolationen werden immer geradlinig durchgeführt. In der Praxis ist auch dies kein Nachteil, da beim Führen des virtuellen Instruments eine hohe Update-Frequenz angestrebt wird. Dies führt gerade bei linearer Interpolation zu einem realistischen Schnittverlauf.

Für den Schnitt mittels eines Dreiecks – z.B. die Schere eines virtuellen Instruments – spielt die Erkennung der Berührung zweier Dreiecke eine wesentliche Rolle. Ein effizienter Algorithmus für die Kollisionserkennung zwischen zwei Dreiecken im Raum ist in [128] beschrieben.

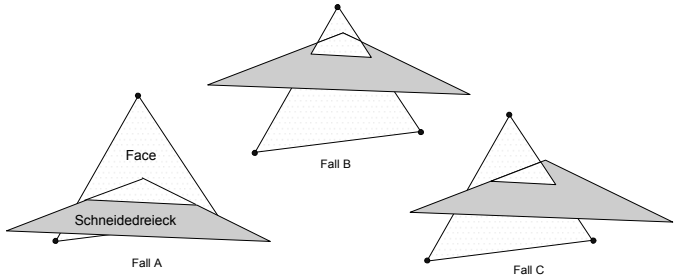

Abbildung 4.28*: Drei Fälle, wie eine Dreiecks-Dreiecksberührung aussehen kann. Fall A: zwei Kanten des Schneidedreiecks durchstoßen das Oberflächendreieck; Fall B: keine, oder Fall C: eine Kante durchstößt das Oberflächendreieck (aus* [91]*).*

Beim Schneiden mit dieser Technik werden alle Dreiecke des Meshs dahingehend getestet, ob sie durch ein definiertes Schneidedreieck berührt werden. Abgesehen von den Extremfällen, bei denen nur ein Punkt oder eine Kante eines Dreiecks das andere berührt, gibt es die in Abbildung *4.28* angegebenen Kollisionsmöglichkeiten.

Abbildung *4.29* zeigt die durch die Kollisionen verursachten Schnitte und die entsprechenden neu gebildeten Dreiecke, Knoten und Federn.

Im Fall A werden zusätzlich zwei Knoten eingefügt, die ein Aufklaffen des Schnittes ermöglichen – es entstehen sieben neue Dreiecke. Im Fall B kommt es zur Auftrennung der beiden geschnittenen Dreieckskanten und somit zur Zerteilung des Dreiecks in drei kleinere Dreiecke. Im Fall C wird ein Knoten beim Durchstoßpunkt der Schneidedreieckskante eingefügt, was in insgesamt vier Dreiecken resultiert.

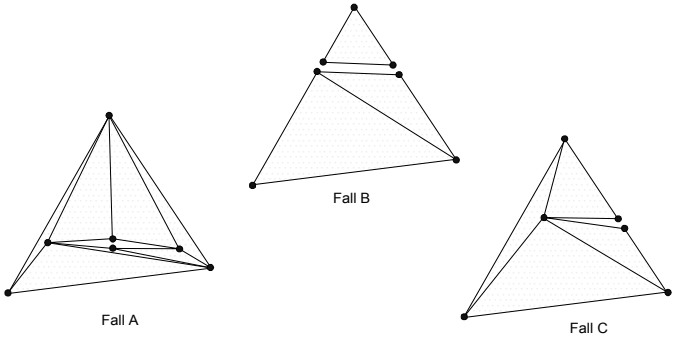

Abbildung 4.29*: Die neue entstandenen Dreiecke, Knoten und Federn nach dem Schnitt mittels Schneidedreieck in Abhängigkeit der drei Berührungsarten (aus* [91]*).*

Der Schneidealgorithmus testet konsequent alle Oberflächenelemente des Objektes auf Kollision mit dem Schneidedreieck – bei einer Berührung wird das betroffene Dreieck unabhängig von den benachbarten in der oben beschriebenen Art geschnitten. Damit bei angrenzenden Dreiecken ein entsprechend geschlossener Schnitt vollzogen wird, ist folgender Vorgang zweckmäßig: Während des Schneidens werden grundsätzlich alle geteilten Dreieckskanten mit den zugehörigen neu erzeugten Knoten protokolliert. Wenn eine Kante geschnitten werden soll, wird vorerst geprüft, ob sie bereits (im Zusammenhang mit einem benachbarten Dreieck) geteilt wurde – ist dies der Fall, werden die bereits eingefügten Stützknoten direkt verwendet und die Kante nicht erneut geteilt (Abbildung *4.30*).

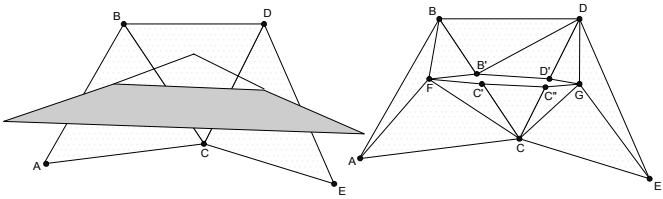

Abbildung 4.30*: Schnitt mittels Dreieck durch drei benachbarte Faces. links: Position des Schneidedreiecks; rechts: die Oberfläche nach dem vollzogenen Schnitt (aus* [91]*).*

Angenommen, das Dreieck BDC ist das erste getestete Polygon und entsprechend werden die Kanten BC und DC geteilt und die Knoten B', C', D' und D" eingefügt. Beim Schneiden von Dreieck ABC wird erkannt, das die Kante BC schon geteilt ist, und es werden entsprechend die vorhandenen Knoten B' und C' für die neu eingefügten Dreiecke verwendet. Vergleichbares vollzieht sich beim Dreieck CDE.

Die Federn liegen in den Membranen stets entlang der Kanten der Oberflächenelemente. Beim Durchtrennen von Seitenkanten der Polygone wird geprüft, ob eine entspre-

chende Feder vorhanden ist – falls dies der Fall ist, wird diese entfernt und zwei neue werden eingefügt. Im Beispiel von Abbildung *4.30* sei eine Feder zwischen B und C gespannt. Beim Durchtrennen dieser Kante wird diese entfernt und durch zwei neue, welche zwischen B und B' bzw. C und C' gespannt werden, ersetzt.

4.4.3.3 Bewertung der Schneideverfahren

Durch die gute Annäherbarkeit einer Schere durch ein Dreieck, ist bei der Verwendung dieses Operationstools zweifellos der Schneide-Algorithmus mittels Dreieck die Methode der Wahl, zumal dieser auch bei zusammengeklappten und verwundenen Gewebehäuten – wie sie beispielsweise während des Schneidens von Membranen auftreten können – verlässlich arbeitet.

Die Möglichkeit des Schneidens von Oberflächen beschränkt sich nicht auf ebene Strukturen, sondern ist prinzipiell für jedes Oberflächenmodell gegeben und kann – wenn notwendig – sogar das gesamte virtuelle Objekt betreffen. Bei der Definition von Schnitten durch zwei Punkte, die direkt auf der Oberfläche angegeben werden, ist es nicht in allen Fällen einfach, ein passendes Schneidedreieck zu bestimmen, welches für den gewünschten Schnitt sorgt – taucht das Dreieck zu gering in die Fläche (bzw. Objekt) ein, können bei gekrümmten Flächen Unterbrechungen im Schnitt die Folge sein, taucht es zu tief ein, so ist die Gefahr gegeben, dass auch darunter liegende Teile der Oberfläche ungewollt in Mitleidenschaft gezogen werden. In diesem Fall ist das Vorantreiben des Schnittes entlang der Oberfläche die Methode der Wahl.

Neben den Manipulationsmöglichkeiten, die direkt das elastodynamische Netz und die polygonale Struktur des virtuellen Objektes betreffen, gibt es weitere Manipulationen, die lediglich die Farbe der Oberfläche verändern. Bei der Koagulation, der Verödung von Gewebe, verändert sich in der Regel lediglich das Aussehen der Oberfläche, was durch Verändern der Textur simuliert werden kann. Dies ist insbesondere mit dem in Kapitel 4.3.1 vorgestellten Texture-Placement-Verfahren durchzuführen. Zusätzlich kann die Viskosität der beteiligten Federn verändert werden, um eine Verhärtung des Gewebes zu simulieren.

4.5 Ergebnisse

Im aktuellen Kapitel wurden alle für die Oberflächenvisualisierung im Rahmen der Chirurgiesimulation relevanten Methoden erläutert. Neben der grundsätzlichen Repräsentation von Oberflächenmodellen als Polygongitter und NURBS-Flächen wurde mit der Triangulation durch Deformationsmodelle ein geeignetes Verfahren für das indirekte Volume-Rendering vorgestellt, mit dem reale Patientendaten, gewonnen aus MRT und CT, als Oberflächenmodelle visualisiert werden können. Das Verfahren, das mit Simplex-Gittern arbeitet, wurde erfolgreich in dem Simulationssystem ROBO-SIM für virtuelle, minimal-invasive neurochirurgische Operationen eingesetzt (vgl. Kapitel 5.2).

Oberflächendetails werden mit dem Texturemapping auf die anatomischen Modelle aufgetragen. Da herkömmliche Texturemapping Verfahren für die Virtuelle Chirurgie

wegen der beim Mapping auftretenden Verzerrungen nicht geeignet sind, wurde ein neuartiges Texture-Placement Verfahren entwickelt, mit dem Oberflächenstrukturen verzerrungsfrei und in hoher Qualität auf die Modelle aufgetragen werden können. Auch dieses Verfahren hat sich beim Einsatz in ROBO-SIM bewährt, bei dem das Ventrikelsystem im menschlichen Gehirn virtuell dargestellt wird.

Für realistische Oberflächenmanipulationen werden außerdem verschiedene Kollisionserkennungsverfahren benötigt. Eine Übersicht über diese Verfahren wird im Anhang A gegeben.

Im Rahmen der in Kapitel 5.2 genannten Chirurgiesimulatoren wurden diverse Oberflächenmanipulationen implementiert: Deformieren, Greifen, Reißen, Schneiden und Koagulieren von Gewebe. Die dazu erforderlichen Verfahren wurden hier ebenfalls beschrieben.

Bei der Auswahl und Implementierung aller Algorithmen wurde besonders auf Effizienz und Ausführungsgeschwindigkeit geachtet. Auf einem Zwei-Prozessor-System läuft die Deformationssimulation exklusiv auf einer CPU, die Kollisionserkennung und die Oberflächenmanipulation laufen sequenziell auf der anderen CPU. Grafische Ausgabe und Texturemapping werden exklusiv durch die Grafikhardware berechnet und dargestellt.

Ausführende Systeme waren zum einen eine Onyx2 IR von Silicon Graphics mit zwei MIPS10000, 180 MHz CPUs, zum anderen ein PC der Firma Dell mit zwei Pentium XEON 800 MHz Prozessoren und einer GForce2 Grafikkarte. Beide Systeme waren in der Lage, die Simulation ruckelfrei mit einer minimalen Bildwiederholungsfrequenz von 20 Hz durchzuführen. Allerdings hatte die wesentlich teurere Onyx2 IR Probleme mit der Anwendung des Mip-Mappings beim Verfärben der Textur während des Fragmentierens, was sich in einer fünftel Sekunde Pause der gesamten Simulation äußerte. Um diese störende Pause zu vermeiden musste das Mip-Mapping ausgeschaltet werden, wodurch die Bildqualität reduziert wurde.

4.6 Diskussion

Die Visualisierung mittels Oberflächenmodellen hat gegenüber dem direkten Volume-Rendering einige Vorteile: schnelle Deformationsalgorithmen werden schon seit etwa 15 Jahren veröffentlicht, durch den Einsatz von Texturemapping kann die Darstellung wesentlich realistischer gestaltet werden und Kollisionserkennung und Oberflächenmanipulationen sind einfacher und mit einer höheren Bildwiederholfrequenz zu berechnen. Echte dreidimensionale Patientendaten können über Segmentier- und Triangulationsalgorithmen ebenfalls verwendet werden, was durch das so genannte indirekte Volume-Rendering genutzt wird.

Der größte Vorteil der Oberflächenmodelle ist die Geschwindigkeit der reinen Darstellung auf Low-Cost-Hardware. Schon seit Jahren werden einfache PC-Grafikkarten immer mehr auf die schnelle Darstellung von texturierten Oberflächenmodellen optimiert. Mittlerweile stellen moderne Grafikkarten zu einem Preis unter 400 € bereits effektiv

mehr als 50 Mio. texturierte Dreiecke pro Sekunde dar – ein Wert, den noch vor vier Jahren nur Hochleistungsgrafikrechner in einer Preisklasse über 1 Mio. € erreichten. Im Vergleich zu texturbasiertem Volume-Rendering ist auch die Qualität der Darstellung nicht zuletzt durch das Verwenden von Texturen realer endoskopischer Aufnahmen wesentlich höher. Die Entwicklung eines Chirurgiesimulators auf PC-Basis kann daher heute nur bei einer Visualisierung durch Oberflächenmodelle realisiert werden, obwohl bereits im Entertainmentmarkt Volume-Rendering für Effekte wie Nebel und Feuer verwendet wird. Die Entwicklung einer interaktiven Deformation von Volumendaten auf PC-Basis in der gleichen Qualität wie das Surface-Rendering ist jedoch noch nicht abzusehen, wenn auch dort viel versprechende Ansätze vorliegen [39; 92; 169; 191; 210].

Demgegenüber ist es jedoch erforderlich, dass für die Planung chirurgischer Eingriffe nur die direkten Patientendaten benutzt werden. Jede Konvertierung, jede Vereinfachung der im Computer gespeicherten Modelle der menschlichen Anatomie stellt eine potentielle Verfälschung der realen Gegebenheiten dar, die im Allgemeinen nicht toleriert werden kann. Nur bei der direkten Visualisierung der Patientendaten ist sichergestellt, dass so wenig wie möglich verändert wird.

Allerdings wird auch dieser Grundsatz von den Chirurgen kontrovers diskutiert. Bei der Virtuellen Endoskopie beispielsweise ist allein schon die Angabe eines Schwellenwertes, der sichtbares (z.B. Gewebe) von durchsichtigem Material (z.B. klare Flüssigkeiten, Luft) trennt, eine Manipulation der realen Daten. Es bleibt daher immer dem medizinischem Experten überlassen, ob und wie er die Daten verändert. In diesem Zusammenhang wäre auch eine Visualisierung durch indirektes Volume-Rendering zulässig, wenn der Chirurg die Segmentierung interaktiv durchführt oder zumindest das Ergebnis überprüft.

Um die Qualität der Virtuellen Endoskopie allgemein zu untersuchen, wurden daher am Institute of Applied Sciences in Medicine, Salzburg, Österreich, zwei medizinische Doktorarbeiten initiiert, die zum einen unterschiedliche Methoden der Virtuellen Endoskopie und zum anderen die Virtuelle Endoskopie mit realen Aufnahmen qualitativ und quantitativ vergleichen. Die Ergebnisse dieser Untersuchungen werden im nächsten Kapitel beschrieben. Außerdem werden Chirurgiesimulatoren in drei verschiedenen medizinischen Bereichen vorgestellt, die alle die in Kapitel 2 beschriebene Deformationssimulation auf Basis von kooperativen Neuro-Fuzzy Systemen verwenden.

5 EINSATZBEISPIELE IN DER CHIRURGIESIMULATION

Die in den vorhergehenden Kapiteln beschriebenen Methoden und Verfahren sind zwar generell für die Anwendung im Bereich der Virtuellen Realität einsetzbar, wurden aber speziell für den Einsatz in der Chirurgiesimulation entwickelt. Dies gilt insbesondere für das Deformationsverfahren auf Basis der Neuro-Fuzzy Systeme, das in der Lage ist, durch umgangssprachliche Ausdrücke und einfache Regler das Deformationsverhalten virtueller Organe zu bestimmen. Aber auch die Darstellungs- und Kollisionsverfahren sind gemäß den Besonderheiten der Virtuellen Chirurgie optimiert. Im aktuellen Kapitel werden daher zwei Chirurgiesimulatoren vorgestellt, für die die beschriebenen Verfahren und Methoden implementiert und evaluiert worden sind. Darüber hinaus enthält das Kapitel noch eine Evaluationsstudie der Virtuellen Endoskopie allgemein und den Vergleich verschiedener Verfahren der Virtuellen Endoskopie, um die Darstellungsqualität von OpenGL Volumizer mit anderen Methoden im Hinblick auf ihre Verwendbarkeit für die Virtuelle Endoskopie zu vergleichen.

5.1 Simulation laparoskopischer Eingriffe in der Gynäkologie

Der erste Chirurgiesimulator, der zur Deformationssimulation Neuro-Fuzzy Systeme benutzte, war das System SUSILAP-G (**SU**rgical **SI**mulator for **LAP**aroscopy in **G**ynaecology) [159; 161], mit dem es möglich ist, eine virtuelle Sterilisation durchzuführen.

Es gibt zwei übliche Methoden zur Sterilisation: *Minilaparotomie* und *Laparoskopie* [217]. Bei der Minilaparotomie werden die Eileiter, die die Eier von den Ovarien zum Uterus leiten, durch einen schmalen Schnitt im Abdomen nach außen gezogen. Ein kleiner Teil jedes Eileiters wird entfernt und die Enden werden abgebunden.

Bei der Laparoskopie wird der Abdominalraum mit Kohlendioxyd gefüllt, um einen Manipulationsraum für die operativen Bewegungen zu schaffen. Nach Punktion des Abdominalraums wird ein Endoskop und ein Koagulationsinstrument, mit dem die Eileiter unterbrochen werden, eingeführt [130]. Die Sterilisation ist der häufigste laparoskopische Eingriff, da dieser insbesondere für die Familienplanung eingesetzt wird [60].

Obwohl die Laparoskopie wegen der Verkürzung des Krankenhausaufenthaltes des Patienten und der reduzierten Operationsdauer [190] meist vorgezogen wird, birgt sie auch Risiken, insbesondere, wenn der operierende Chirurg nicht genügend Erfahrung besitzt [73]. Das schwerwiegendste Risiko ist eine Perforation des Darmes, was zu einer massiven Infektion führen kann. Dann ist es notwendig, die Operation offen weiterzuführen, was bei 3,8% aller laparoskopischen Operationen erforderlich ist [190]. Auch bei der Sterilisation, die einen Standardeingriff darstellt, treten durchschnittlich 0,5 schwerwiegende [73] und 1-2 einfache Komplikationen [217] auf 1.000 Operationen auf.

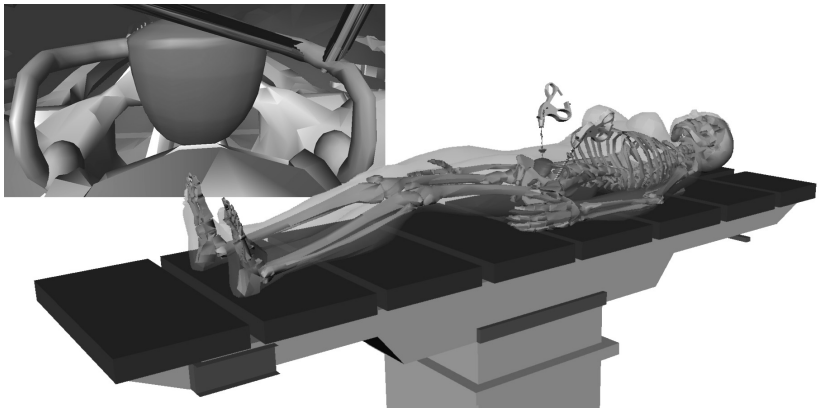

Abbildung 5.1*: Die virtuelle Patientin als Oberflächengeometrie. Zwei virtuelle laparoskopische Instrumente sind in das Abdomen eingeführt. Links oben: virtuelle endoskopische Sicht auf den Uterus mit Adnexen.*

Wegen der Schwierigkeit des laparoskopischen Eingriffes gegenüber der offenen Chirurgie und der Vielzahl der jährlich durchgeführten Sterilisationen, ist es für angehende Chirurgen daher sinnvoll, den Eingriff am Simulator virtuell zu üben. Weil sich die Simulation nicht an fertig ausgebildete Chirurgen richtet, wurde eine sehr grobe Modellierung der weiblichen Anatomie, bestehend aus einem Außenkörper, einem Skelett und dem Uterus mit den Eileitern, im Rahmen einer ersten Implementierung als ausreichend empfunden (Abbildung *5.1*). Die Modelle wurden nicht aus realen dreidimensionalen Patientendaten segmentiert, sondern mit einem Modellierungstool erstellt [160].

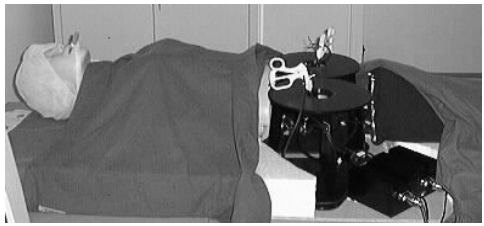

Abbildung 5.2*: Simulationspuppe mit integrierten Laparoscopic Impulse Engines (aus* [154]*).*

Als Schnittstelle zum Anwender wird eine Puppe benutzt, die als Patientin dient (Abbildung *5.2*), einem Monitor, auf dem die endoskopische Sicht simuliert wird und zwei Laparoscopic Impulse Engines, da diese den Bewegungsraum laparoskopischer Instrumente hervorragend simulieren und außerdem eine haptische Rückkoppelung bieten [152].

Bei SUSILAP-G wurden die Eileiter als Feder-Masse System modelliert (Abbildung *5.3*). Um eine Verbindung des Eileiters mit der Bauchhöhle zu simulieren, wurde jeder

dritte Knoten durch eine Feder mit seiner Ursprungsposition verbunden. Dadurch werden zusätzliche Elemente zur internen Stabilisierung des Eileiters überflüssig. Für die graphische Darstellung dieser Feder-Masse Systeme wird die Oberfläche geschlossen gezeichnet und mit einer Textur versehen. Die Textur wurde nach Vorlage realer endoskopischer Aufnahmen erstellt.

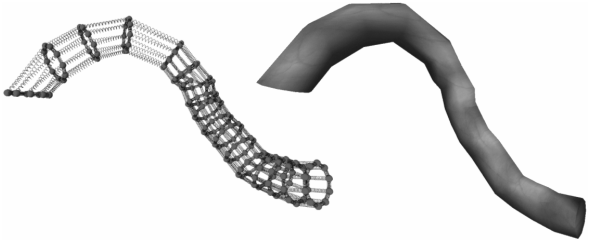

Abbildung 5.3*: Ein Eileiter als Feder-Masse System (links) und als graphisches Oberflächenmodell versehen mit Texturen (rechts) (aus* [154]*).*

In Verbindung mit einer dynamischen Kollisionserkennung Strahl ⇔ Dreieck (Kapitel A.2), die einen Kontakt zwischen den Eileitern und dem virtuellen Instrument feststellt, wird das Modell verformt, indem die entsprechenden Masseknoten entlang der Bewegungsrichtung des Instruments verschoben werden. Da das Modell an jeder beliebigen Stelle verformt werden kann, werden immer alle Federn und Knoten berechnet. Das in Abbildung *5.3* dargestellte Modell des Eileiters besteht aus 170 Masseknoten und 340 Federn und ist somit im Gegensatz zu vergleichbaren starren Oberflächenmodellen, die mit bis zu 500.000 Linien[1] noch in Echtzeit dargestellt werden können, recht einfach aufgebaut.

Für den Uterus ist eine Modellierung als Feder-Masse System zu aufwendig und unnötig, da der Uterus aus sehr festem Gewebe besteht und während chirurgischer Eingriffe am Eileiter als ganzes verschoben wird. Der Uterus kann so in alle Richtungen bewegt und gedreht werden. Erreicht wird dies durch ein einfaches Feder-Masse System innerhalb der anatomischen Struktur (siehe Abbildung *5.4*) bestehend aus acht Masseknoten und sieben Federn.

Die sechs äußeren Knoten haben eine feste Position und sind damit nicht beweglich. Die zwei inneren Knoten bestimmen die Bewegung des Uterus, dessen Ursprung immer in der Mitte des Elements zwischen den Knoten liegt. Die Orientierung des Uterus wird ebenfalls mit Hilfe dieses Elements bestimmt. Die Positionsveränderung der beweglichen Masseknoten erfolgt relativ zum Kontaktpunkt mit dem virtuellen Instrument. Tritt eine Kollision auf, so wird der gesamte Uterus in Bewegungsrichtung des virtuellen Instruments verschoben. Diese Art der Deformation wird globale Deformation genannt [154]. Zur Verdeutlichung der Rotation sind in diesem Beispiel die Adnexe starre Modelle. Bei der Verwendung in der Operationssimulation werden diese mittels Feder-Masse Syste-

[1] *In einem starren Modell sind die Linien das Äquivalent zu den Federn im Feder-Masse System.*

men simuliert. Die Parameterbestimmung für die Feder-Masse Systeme wurde durch das im Kapitel 2.3.7 beschriebene Tool „Elastodynamic Shape Modeler“ durchgeführt. Das verwendete Fuzzy System und die eingestellten Parameter sind im Anhang B aufgeführt.

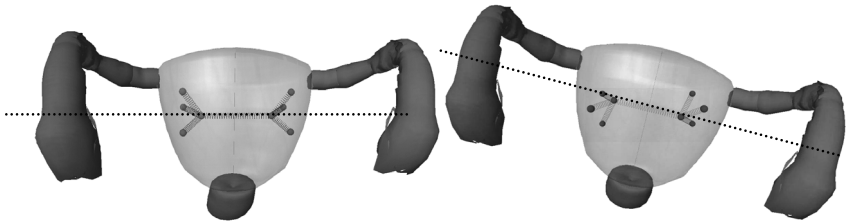

Abbildung 5.4*: Ein einfaches Feder-Masse System innerhalb des Uterus für die globale Deformation. (nach* [154]*).*

Die Simulation von Deformationen lässt die Operationssimulation realistischer erscheinen. Ein sinnvolles Training mit einem solchen System wird allerdings erst möglich, wenn die Modelle veränderbar sind. Für eine Sterilisation muss die Möglichkeit bestehen, die Prozeduren Greifen und Koagulieren durchzuführen. In SUSILAP-G werden bei einer Kollision des Eileiters mit den Backen der Zange die betreffenden Masseknoten in Bewegungsrichtung der jeweiligen Backe verschoben. Ähnlich funktioniert die Koagulation, mit dem Unterschied, dass die Verformung des Eileiters durch Entspannen der verformten Federn dauerhaft ist, womit eine inelastische Deformation simuliert wird. Darüber hinaus verfärbt sich das koagulierte Gewebe. Dies wird durch Einfärben der Textur auf der Oberfläche des virtuellen Organs erreicht.

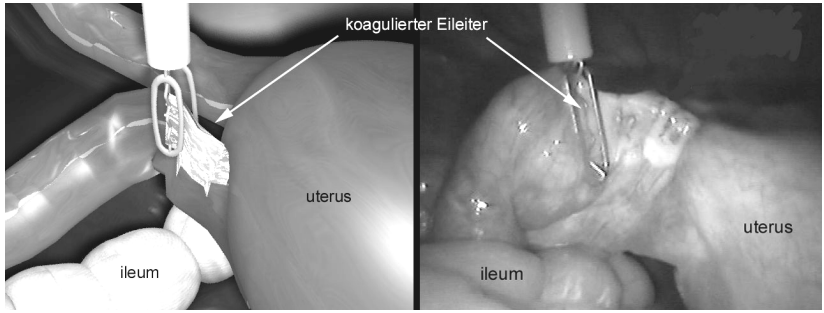

Abbildung 5.5*: Gegenüberstellung einer virtuellen Sterilisation mit SUSILAP-G (links) und einer realen laparoskopischen Sterilisation (rechts). Modelliert sind die Eileiter, der Darm (ileum), der Uterus und das Abdomen im Hintergrund.*

SUSILAP-G wurde Mitte 1998 auf der 114. Tagung der Norddeutschen Gesellschaft für Gynäkologie und Geburtshilfe erstmals einem Fachpublikum vorgestellt [155]. Obwohl viele Chirurgen den Simulator für allgemeines laparoskopisches Training geeignet hielten, da die Navigation mit den virtuellen Instrumenten als sehr realistisch empfunden

wurde, galt die hauptsächliche Kritik den einfachen anatomischen Modellen. Weil das schwerwiegendste Risiko bei der Laparoskopie eine Perforation des Darms ist, wurde insbesondere das Fehlen einer Darmsimulation bemängelt. Aus diesem Grund wurde die bestehende Version von SUSILAP-G erweitert und graphisch aufwendiger gestaltet. Abbildung *5.5* zeigt eine Gegenüberstellung dieser neuen Version mit einer realen laparoskopischen Sterilisation, bei der komplexere Modelle des Uterus und der Eileiter modelliert worden sind. Außerdem ist nun ein Modell des Darms vorhanden, das aus einzelnen, rigiden Ellipsoiden besteht, die mittels eines einfachen Feder-Masse Systems simuliert werden, das aus Masseknoten jeweils in den Schwerpunkten der Ellipsoide besteht, die durch Federn verknüpft sind.

Um die Kollisionserkennung mit dem Darm zu vereinfachen, werden nur zwischen den festen Geometrien des Instruments und den einzelnen Ellipsoiden Kollisionstests durchgeführt. Dies hat den Vorteil, dass für die Kollisionserkennung ein schnelles OBB-Tree Verfahren verwendet werden kann (vgl. Kapitel A.1). Für das Training mit SUSILAP-G ist ein Ziel die Vermeidung einer Kollision mit dem Darm. Deswegen ist es in diesem Zusammenhang sinnvoll, die Kollisionen zu registrieren und dem Trainierenden, der in sukzessiven Trainingsdurchläufen versucht, die Anzahl der Kollisionen mit dem Darm zu reduzieren, als eine Art Evaluationsstatistik für den Eingriff zu präsentieren.

Als Hardwareplattform für SUSILAP-G dient eine Onyx2 Infinite Reality von SGI, ausgerüstet mit zwei Prozessoren und einem Rastermanager, der für die Grafikdarstellung benötigt wird. Da ein solcher Computer in der Anschaffung um ein Vielfaches höhere Kosten als die eines PC verursacht, ist eine Anwendung als Trainingssystem für Einzelpersonen oder auch kleinere Ausbildungsstätten nicht zu erwarten. Deshalb wurde auf der TELEMEDIZIN ′98 in Zusammenarbeit mit der Telekom Braunschweig eine telemedizinische Erweiterung von SUSILAP-G getestet [154]. Ziel dieses Feldversuchs war zu ermitteln, ob ein Training mit SUSILAP-G ferngesteuert über Standard-ISDN-Leitungen möglich ist. Damit wäre ein Üben mit SUSILAP-G möglich, ohne dass sich der Anwender in der Nähe der Hardwareplattform befinden müsste. Er würde dann lediglich die Simulationspuppe mit integrierten Laparoscopic Impulse Engines (Abbildung *5.2*) und einen PC mit Videokonferenzhardware benötigen. Diese Interfacehardware ist auch wesentlich leichter zu transportieren als eine Onyx2, die etwa 100 kg wiegt.

Die Kommunikation des Personalcomputers mit der Onyx2 geschieht dabei über geeignete Netzwerkverbindungen wie z.B. ISDN. Abbildung *5.6* zeigt den Aufbau der Komponenten und deren Vernetzung. Die Positionsdaten der Impulse Engines werden über eine ISDN-Leitung mit einer Übertragungsrate von 64kBit/s an einen ISDN-Server gesendet. Dieser ISDN-Server - ebenfalls ein Standard-PC - ist über ein Local Area Network (LAN) mit dem Grafikserver verbunden, was voraussetzt, dass Grafik- und ISDN-Server nicht weit voneinander entfernt sind. Der Grafikserver, die Onxy2 von SGI, berechnet anhand der Positionsdaten die neue endoskopische Ansicht inklusive virtueller Instrumente und Deformationen. Die bei der Deformation auftretenden Kräfte werden über die gleiche ISDN-Leitung an den PC mit den Impulse Engines weitergeleitet. Der Videoausgang des Grafikservers wird über ein FBAS bzw. SVHS-Signal an ein handelsübliches Videokonferenzsystem angeschlossen. Dieses System ist wiederum mittels

mehrerer ISDN-Leitungen mit einem entsprechenden System am Ort der Simulationspuppe verbunden. Dort kann das Videobild der virtuellen Operation, das durch sechs ISDN-Kanäle übertragen wurde, auf einem Monitor angezeigt werden.

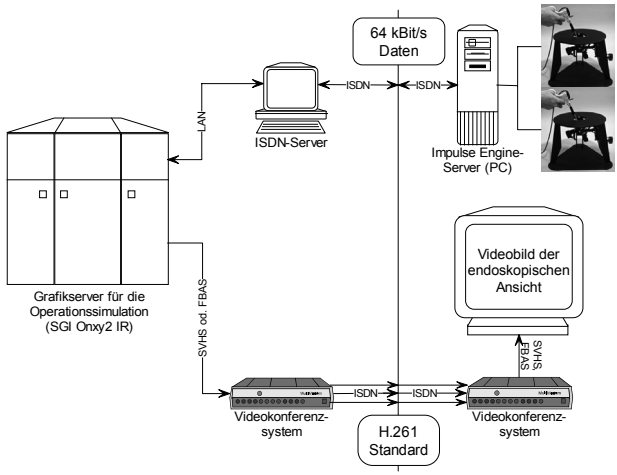

Abbildung 5.6*: Datenübertragung per Videokonferenz. Die beiden Impulse Engines sind mit einem handelsüblichen PC gekoppelt, der mit einer ISDN-Karte ausgestattet ist (aus* [154]*).*

Auf der TELEMEDIZIN'98 testeten fünf Chirurgen, acht Ärzte und eine Reihe von interessierten Fachbesuchern die telemedizinische Erweiterung von SUSILAP-G. Während alle Testpersonen die Darstellungsqualität des übertragenen Videobildes für ausreichend hielten, bemängelten sie jedoch die hohe Latenzzeit des Systems. Durch das verwendete Kompressionsverfahren des Videokonferenzsystems H.261 [27] entstand eine Verzögerung von fast einer halben Sekunde. Eine Bewegung des Instruments erschien also erst eine halbe Sekunde später auf dem Monitor. Alle testenden Chirurgen empfanden ein Training unter diesen Bedingungen als vollkommen unrealistisch und deshalb nicht verwendbar. Der bei der direkten Verwendung von SUSILAP-G gelobte Realismus der virtuellen Navigation mit den laparoskopischen Instrumenten wurde durch die telemedizinische Erweiterung vollkommen zerstört. Ein Einsatz von SUSILAP-G als telemedizinisches Trainingssystem ist mit den technischen Übertragungsraten von ISDN selbst dann nicht möglich, wenn man mehrere Kanäle verwendet [154].

5.2 Simulation minimal-invasiver neurochirurgischer Eingriffe

Nahezu alle Chirurgiesimulatoren in Forschung und Entwicklung, wie auch das Simulationssystem SUSILAP-G, haben allerdings einen bedeutenden Mangel, sodass sie für das Training erfahrener Chirurgen nur bedingt einsatzfähig sind: Sie arbeiten nur mit der

modellhaften Anatomie des Menschen und benutzen keine realen Patientendaten, wie sie beispielsweise durch Anwendung von MRT oder CT vorliegen. Demgegenüber wird aber von Chirurgen eine immer größer werdende Automatisierung insbesondere für die Planung von Operationen eingesetzt [11]. Für den Arzt ändern sich daher die Arbeitsbedingungen zusehends in Richtung Planung, Simulation und Supervision. Mehr denn je zuvor spielen Einrichtungen für Ausbildung und Training, für Planung und Simulation eine Rolle [9]. Unter „Planung“ wird in diesem Zusammenhang die millimetergenaue Vorplanung des operativen Zugangswegs zu einem Zielpunkt in der Tiefe des Körpers unter Zuhilfenahme präoperativer Patientendaten verstanden. Die hierfür erforderlichen technischen Einrichtungen wurden wegen des hohen chirurgischen Komplikationsgrades vor allem in der Neurochirurgie entwickelt, wo sie heute unter dem Begriff „Neuronavigation“ bereits zum Standard zählen. Inzwischen wurden auch andere Gebiete der Chirurgie, z.B. die Wirbelsäulenchirurgie und einige Bereiche der Orthopädie, der Urologie und der Hals-Nasen-Ohren Heilkunde sowie der Kieferchirurgie mit solchen Entwicklungen ausgestattet (Chirurgische Navigation).

Mit der Entwicklung von bildgesteuerten Planungseinrichtungen ist auch die chirurgische Simulation auf der Basis realer Patientendaten in den Bereich der Anwendbarkeit gerückt. Die Schwierigkeit, realitätsnahe Darstellungen der menschlichen Anatomie realer Patienten im Computer zu simulieren und zu visualisieren, liegt insbesondere an dem großen Umfang der Datensätze, die von bildgebenden Verfahren in der Medizin generiert werden. Um diese Datenmengen zu bewältigen, müssen – trotz konsequenter Ausnutzung der Computerhardware - teilweise erhebliche Vereinfachungen der physikalischen Modelle und der Darstellungsqualität in Kauf genommen werden (vgl. Kapitel 3 und 4).

Je kritischer der chirurgische Eingriff für die Gesundheit und das Leben des Patienten ist, desto notwendiger ist es, dass dieser möglichst optimiert und perfekt durchgeführt wird. Neben der guten Ausbildung des Arztes, ist auch die Qualität der Operationsvorbereitung verantwortlich für den Erfolg. Eingriffe am Gehirn stellen in diesem Sinne eine besondere Herausforderung für alle Beteiligten dar. Bei Tumoren im Ventrikelbereich des Gehirns kann die Virtuelle Endoskopie und Chirurgie im Sinne der Ausbildung, der Evaluierung und der Simulation wertvolle Unterstützung leisten.

Die *Ventrikel* sind flüssigkeitsgefüllte Hohlräume im Gehirnareal des Kopfes und dienen physiologisch unter anderem der mechanischen Dämpfung. Einsatzgebiete der minimal-invasiven Neurochirurgie sind aufgrund der Lage der Ventrikel, Tumoroperationen tief im Inneren des Gehirns (bis zu 15 cm). Minimal-invasive chirurgische Interventionen am menschlichen Gehirn sind besonders schwierig, da lebenswichtige Strukturen die unmittelbare Nachbarschaft eines kleinen Operationsfeldes von wenigen Millimetern bilden können. Über ein Bohrloch (*Trepanation*), wird ein Endoskop eingeführt, das gleichzeitig zum Beobachten, Spülen und als Kanal für mehrere chirurgische Mikroinstrumente dient. Die Planung erfolgt dabei anhand digitaler medizinischer Datensätze des Patienten, die meist mittels MRT oder CT dreidimensional aufgenommen werden. Trotz der dreidimensionalen Datensätze wird die traditionelle Planung meist anhand einer be-

grenzten Anzahl von Schichtbildern in Form von Folien durchgeführt. Eine dreidimensionale Darstellung wird bisher lediglich zur Planung der operativen Freilegung genutzt.

Daher wurde 1998 die Entwicklung von ROBO-SIM im Rahmen des von der EU finanzierten Projekts 4018 (ROBOSCOPE) des „Telematics Program“ [2] für robotergestützte, bildgesteuerte minimal-invasive Neurochirurgie begonnen. ROBO-SIM ist ein Planungs- und Simulationssystem für minimal-invasive neurochirurgische Eingriffe, das anhand realer Patientendatensätze arbeitet.

ROBO-SIM erlaubt als Eingabeschnittstellen wahlweise den Betrieb mit einer Laparoscopic Impulse Engine, einem herkömmlichen Neuroendoskop, das mit einem Positionsbestimmungsgerät ausgestattet ist und mit dem aktiven Manipulator NEUROBOT.

5.2.1 Eingabeschnittstellen

5.2.1.1 NEUROBOT

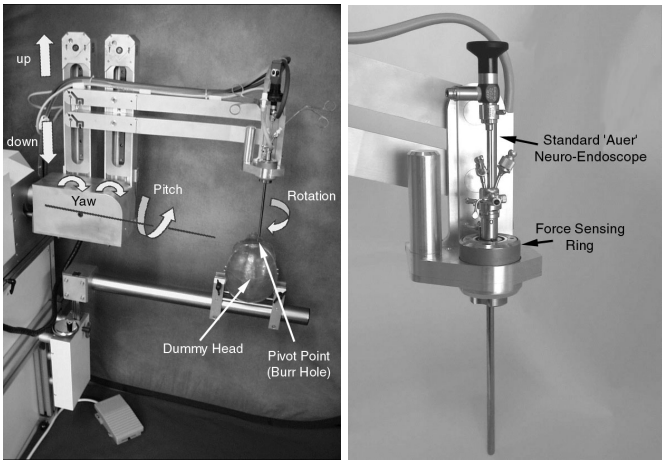

Abbildung 5.7*: Links: Der aktive Manipulator NEUROBOT besitzt vier Freiheitsgrade für die Bewegung um einen Pivotpunkt (Trepanationspunkt). Ein künstlicher Kopf wird am Roboter zur Simulation von Eingriffen befestigt. Rechts: Ein herkömmliches Neuro-Endoskop, das an einem Kraftsensor-Ring befestigt ist, wird benutzt, um den Roboter zu führen (aus* [164]*).*

Eine von drei möglichen Benutzerschnittstellen für ROBO-SIM ist ein aktiver Manipulator, NEUROBOT, entwickelt von Fokker Control Systems b.v., Niederlande (siehe Abbildung *5.7* links). NEUROBOT besitzt eingebaute robotische Fähigkeiten wie eine Restriktion seiner Bewegungen auf erlaubte Gebiete und die Möglichkeit, automatisch Konturen zu folgen, wenn sie sich deformieren oder verschieben. Wie bei üblichen endoskopischen Instrumenten ist die Bewegung auf vier Freiheitsgrade um einen Pivotpunkt - das Bohrloch beziehungsweise den Punkt der Trepanation - beschränkt. Damit können, wie mit einem handgeführten Neuroendoskop, alle Positionen im Inneren des Gehirns,

die durch die Trepanation erreichbar sind, angefahren werden. Das Zielvolumen bildet damit einen Kegel mit der Spitze im Zentrum der Trepanation. Andere Bewegungen sind bereits durch die Mechanik des Roboters ausgeschlossen.

Am Endeffektor des Roboters, auf den ein Standard (Neuro-)Endoskop mit Kamera aufgesteckt werden kann, ist ein Kraftsensor-Ring befestigt. Jede Kraft, die in einer bestimmten Richtung auf den Kraftring ausgeübt wird, resultiert in einer aktiven Bewegung des Roboters, wodurch das Endoskop leichtgängig und fast schwerelos bewegt werden kann. Zusätzlich können so genannte aktive Grenzen definiert werden, mit denen die Bewegung auf bestimmte (anatomische) Bereiche beschränkt wird. Dadurch ist man in der Lage, feste Grenzen zu simulieren, die z.B. den Bereich eines Gehirntumors beschreiben. Ungewollte Bewegungen außerhalb des so markierten Gebiets werden damit unmöglich.

Durch einen Fußschalter, der während der Bewegung des Endoskops gedrückt sein muss, kann in einen Positionsmodus umgeschaltet werden, in dem der Roboter das Endoskop fest an der gewählten Position hält und somit keine Bewegung mehr zulässt. Der Chirurg hat damit die Möglichkeit zwei Instrumente, die er durch den Endoskopkanal einführt, gleichzeitig zu bedienen, ohne dass sich das Endoskop bewegt.

Intention im Rahmen des Projekts ROBOSCOPE war, NEUROBOT sowohl für reale als auch für simulierte minimal-invasive neurochirurgische Operationen zu verwenden, was ermöglicht, in dem Simulator dieselbe Ausrüstung wie für reale Operationen zu benutzen. Bei der Simulation mit ROBO-SIM ist es möglich, einen künstlichen Kopf an einer Halterung des Roboters zu befestigen, der für unterschiedliche Trepanationspunkte eingerichtet werden kann.

5.2.1.2 Modifizierte Laparoscopic Impulse Engine

Eine wesentlich preisgünstigere Variante ROBO-SIM zu bedienen, ist die Verwendung einer modifizierten Laparoscopic Impulse Engine (IEngine, siehe Abbildung *5.8*). Anstatt des Griffes wird ein Neuroendoskop an der IEngine befestigt. Der Griff wird zur Simulation des transendoskopischen Instruments mit zwei zusätzlichen Freiheitsgraden versehen, mit denen das transendoskopische Instrument gesteuert werden kann: Die Rotation des Instruments um seine Achse und die translatorische Bewegung des Instruments, beides unabhängig einer Bewegung des Endoskops. Diese zusätzlichen Freiheitsgrade sind schematisch in Abbildung *5.8*, rechts dargestellt.

Eine modifizierte IEngine wäre wegen der zusätzlichen zwei Freiheitsgrade ein ideales Eingabemedium gewesen. Leider ließ sich dies aus Kostengründen im Rahmen des ROBOSCOPE-Projekts nicht realisieren, sodass die zusätzlichen zwei Freiheitsgrade über einen Schalter emuliert werden. Mit dem Schalter kann zwischen der Bewegung des transendoskopischen Instruments und dem Endoskop selbst umgeschaltet werden. Damit ist allerdings keine gleichzeitige Bewegung des Instruments und des Endoskops simulierbar. Um aber eine Bewegungsbegrenzung wie bei einer Operation mit NEUROBOT durchzuführen, wird bei Umschalten auf das transendoskopische Instrument die IEngine über ihren Force-Feedback Mechanismus fixiert. Dies entspricht bei der realen Operation mittels NEUROBOT dem Festsetzen des Endoskops mit dem Fußschalter.

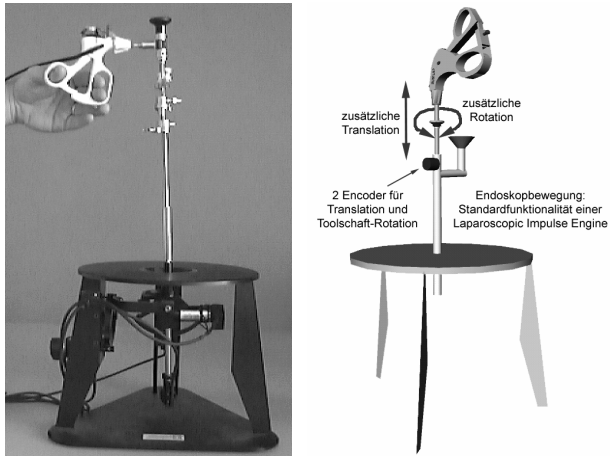

Abbildung 5.8*: Links: modifizierte Laparoscopic Impulse Engine. Schematische Ansicht. Rechts: Um zwei zusätzliche Freiheitsgrade hinzuzufügen, die die Bewegung eines transendoskopischen Instruments simulieren, sind zwei zusätzliche Encoder erforderlich.*

5.2.1.3 Manuelles Neuroendoskop

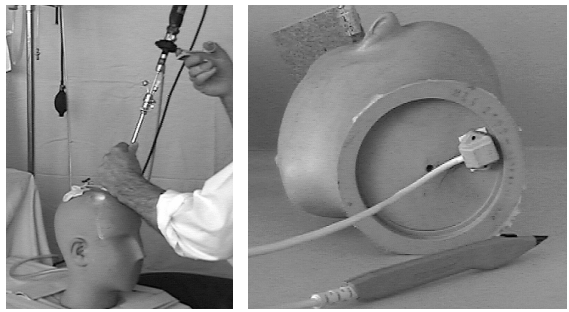

Abbildung 5.9*: Simulation eines herkömmlichen Eingriffes mit einem handgeführten Neuroendoskop (links). Am Endoskop und an dem künstlichen Kopf ist ein Positionsbestimmungssystem befestigt (rechts).*

Eine dritte Möglichkeit, ROBOSIM zu bedienen, ist die Benutzung eines herkömmlichen Neuroendoskops, das mit einem magnetischen Positionsbestimmungssystem (Polhemus Fasttrack) versehen ist (Abbildung *5.9*, links). Damit können ständig die Position und Orientierung (sechs Freiheitsgrade) des Endoskops ermittelt werden. Für jedes transendoskopische Instrument werden ebenfalls Position und Orientierung bestimmt. Da die Instrumente flexibel sind, wird deren Krümmung bei der Positionsberechnung mit berücksichtigt: Die Krümmung wird mit einer einfachen Spline-Kurve angenähert. Diese

Näherung ist ausreichend, da nur die relative Bewegung des Instruments bezüglich des Endoskops relevant ist.

Zusätzlich zu der Bewegung des Endoskops und der transendoskopischen Instrumente wird auch die Bewegung des künstlichen Kopfes durch ein weiteres Positionsbestimmungssystem berücksichtigt (Abbildung *5.9*, rechts). Damit ist eine relative Bewegung des virtuellen Endoskops und des Patientendatensatzes wie bei einer realen Operation möglich.

5.2.2 Die Benutzeroberfläche

Abbildung 5.10*: Das Ladefenster für die Patientendaten. Eine Ansicht per Volume-Rendering des Patienten wird zusammen mit den Patientendaten, die hier anonymisiert wurden, angezeigt. Der Datensatz kann direkt per Klick mit der Computermaus geladen werden (aus* [163]*).*

User-Interface und Bedienungssoftware nutzen derzeit als Plattform eine Onyx 2 Infinite Reality von Silicon Graphics.

Sowohl für die Planung als auch für die Simulation werden die Datensätze bildgebender Verfahren von realen Patienten benutzt, wofür eine DICOM-Schnittstelle[1] implementiert wurde. Verwendet wurden unter anderem Datensätze von 3D-Ultraschallgeräten (Kretz Voluson), MRT Datensätze, fMRT Datensätze und MRT Angiogramme. Die Datensätze werden über ein Dialogfenster geladen, in dem jeder Datensatz als Volume-Rendering zusammen mit den relevanten Patientendaten angezeigt wird (Abbildung *5.10*).

Nachdem der Datensatz geladen worden ist, kann die Planung des Eingriffs beginnen.

[1] *DICOM: Digital Imaging and Communication in Medicine, Standard für die Modellierung und die Übertragung von Bilddaten in der Medizin [3].*

5.2.3 Planung

Die Planung eines Eingriffs wird in folgenden Schritten, die in den nächsten Abschnitten beschrieben werden, durchgeführt:

- Definition eines Start- und Zielpunktes für den geplanten Eingriff,
- Überprüfen und Ändern der Trajektorie vom Start- zum Zielpunkt,
- Virtuelle Kraniotomie: Simulation einer Trepanation (Öffnen des Schädels),
- Definition von aktiven Grenzen,

Die Planung erfolgt in der Regel in dieser Reihenfolge. Über ein Menüfenster können die Schritte aber auch beliebig aufgerufen werden.

5.2.3.1 Definition des Zugangs- und Zielpunktes

Üblicherweise beginnt die präoperative Planung mit der Definition eines Zugangspunktes in den Schädel für die Trepanation, sowie mit der Festlegung eines Zielpunktes, der in den zu operierenden Bereich gelegt wird. Mehrere verschiedene Planungsfenster, bestehend aus verschiedenen Komponenten, sind in ROBO-SIM integriert. Diese Komponenten sind zwei Editoren mit den axialen, koronalen und sagittalen Schnitten des Patienten für die Planung des Zugangs- und Zielpunktes, eine virtuelle endoskopische Ansicht wahlweise in Volume-Rendering oder Surface-Rendering, eine Planungsansicht mit einer beliebig positionierbaren Kamera, einer globalen Ansicht (Außenansicht, Position des Endoskops relativ zum Schädel des Patienten) und mehrere Ansichten mit Schnitten entlang des Endoskops und eine pseudo-dreidimensionale Ansicht der Schnitte mit Einblendung der Trajektorie vom Start zum Zielpunkt. Alle Ansichten sind interaktiv und veränderbar in Größe und Position.

Eine bestimmte Anzahl von Planungsfenstern sind vordefiniert und neue Fenster können ebenfalls unter Zuhilfenahme der Komponenten erstellt und abgespeichert werden. Abbildung *5.11* zeigt ein Beispiel eines solchen Planungsfensters. Bewegung der Positionsmarker für den Zugangs- und Zielpunkt resultiert in der Veränderung der Schnitte in den anderen Fenstern der Komponente, der virtuellen endoskopischen Ansicht und der Position des Endoskops.

Bei dem dargestellten Datensatz sieht man bei der virtuellen Außenkamera innerhalb des Ventrikels im zweiten Fenster von links unten, dass das Septum pellucidum, das die beiden Hälften des Ventrikels trennt, nicht abgebildet ist. Dies liegt an der mangelnden Auflösung des MRT Datensatzes, der die unter einem Millimeter dünne Membran nicht darstellt.

Eine solche Planung von Trepanationspunkt und Zielpunkt des Eingriffes anhand der Schichtbilder ist auch mit am Markt etablierten, modernen Neuronavigationssystemen (z.B. VectorVision[2], BrainLAB AG, Deutschland) durchführbar. Allerdings wird keine Ansicht einer Virtuellen Endoskopie berechnet und wegen der fehlenden Simulation entfällt damit auch die Möglichkeit zur Planung der Ausrichtung des Endoskops (Rotation um die Längsachse).

Darüber hinaus bietet ROBO-SIM weitere Planungsmöglichkeiten, welche auch die Überprüfung der Trajektorie vom Start- zum Zielpunkt mit einschließt.

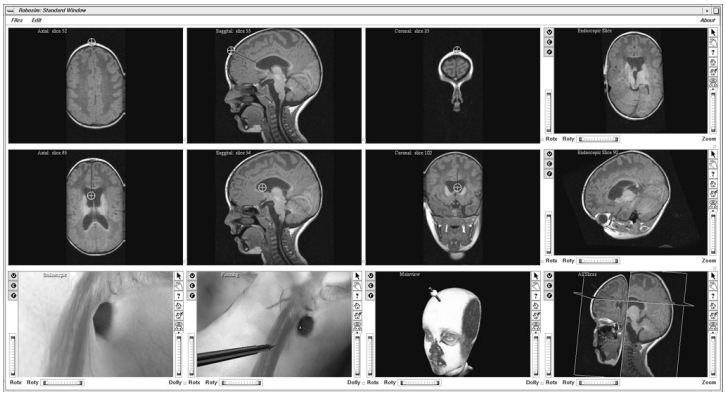

Abbildung 5.11*: Ein Beispielfenster von ROBO-SIM: Die obere Reihe von Bildern erlaubt die Definition des Trepanationspunktes (Zugang in den Schädel) durch Verschieben der Positionsmarker mit der Computermaus. Mit der mittleren Reihe ist die entsprechende Definition des Zielpunktes möglich. Das Bild links unten zeigt einen Blick durch ein virtuelle Endoskop (indirektes Volume-Rendering) von einem frontalen Zugang in den linken lateralen Ventrikel. Rechts neben diesem Bild ist ein Blick durch eine virtuelle Kamera außerhalb des Endoskops entlang des ventrikulären Lumens zu sehen, der die Position des virtuellen, transendoskopischen Instruments zeigt. Das Bild rechts davon zeigt den virtuellen Kopf des Patienten mit dem virtuellen Endoskop. Die rechte Spalte zeigt berechnete Schnitte entlang des Endoskops und eine pseudo-3D-Ansicht mit eingeblendeter Trajektorie (aus* [163]*).*

5.2.3.2 Überprüfen der Trajektorie vom Start- zum Zielpunkt

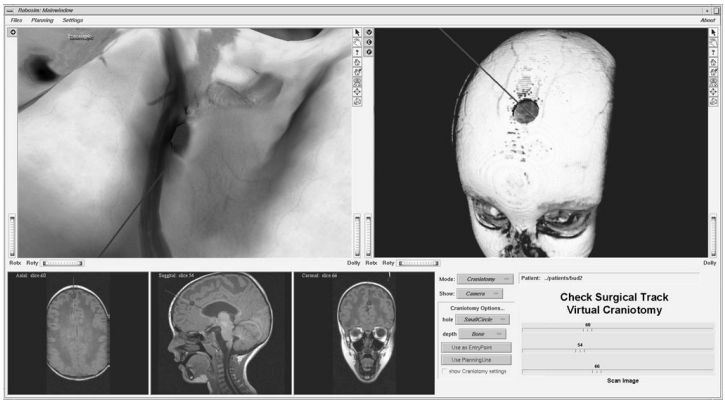

Abbildung 5.12*: Die oberen Bilder zeigen eine Außenansicht des Patienten mittels Volume-Rendering und eine virtuelle endoskopische Ansicht des Zielpunktes mittels Surface-Rendering. Die Line vom Start- zum Endpunkt kann mit der Computermaus verschoben werden. Die untere Reihe*

von Bildern zeigt die axialen, sagittalen und koronalen Schnitte des Patienten jeweils überlagert mit der Trajektorie und dem Schnittpunkt dieser mit der gewählten Schicht (aus [163]*).*

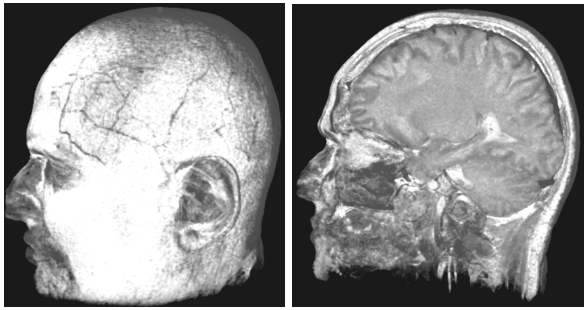

Abbildung 5.13*: MRT Angiogramm, bei dem die Gefäßverläufe rot eingefärbt worden sind. Links: Außenansicht eines Datensatz, der mit direktem Volume-Rendering visualisiert worden ist, rechts: Schnitt durch den Datensatz.*

Der nächste Schritt ist zu überprüfen, ob die Trajektorie vom Start- zum Zielpunkt funktionell wichtige Areale des Gehirns verletzt. Das Standardfenster zur Überprüfung der Trajektorie besteht aus einer globalen Ansicht, einer virtuellen endoskopischen Ansicht und den axialen, koronalen und sagittalen Schichten des Patienten, in denen die Trajektorie durch rote Linien und einem roten Punkt, in dem die Trajektorie die Schicht schneidet, angezeigt ist (Abbildung *5.12*). Mit der Computermaus können alle Schichten durch Schieberegler oder direkt durch eine Bewegung mit gedrückter Maustaste interaktiv verändert werden, während ständig die Linien und Punkte neu angezeigt werden. Die Trajektorie kann gegebenenfalls durch ein Verschieben der Kugeln am Anfang und Ende der Trajektorie verändert werden.

Besonders interessant wird diese Art der Planung, wenn als Basisdatensatz ein funktioneller MRT Datensatz (fMRT) verwendet wird, bei dem die funktionell wichtigen Areale des Gehirns als Farbinformation angezeigt werden. Eine andere Möglichkeit bietet die Verwendung von MRT Angiogrammen, die Gefäßverläufe mit in der Planung berücksichtigen (Abbildung *5.13*).

5.2.3.3 Virtuelle Kraniotomie

Nachdem die Trajektorie vom Start- zum Zielpunkt überprüft worden ist, kann die virtuelle Trepanation durchgeführt und die Größe, Tiefe und Position der Trepanation geplant werden.

Das Fenster für die virtuelle Kraniotomie hat ein ähnliches Layout wie das Fenster zur Überprüfung der Trajektorie, jedoch wird in der Außenansicht die virtuelle Trepanation gezeigt. Außerdem gibt es Bildschirmfunktionen zur Variation der Trepanation. Jede Änderung der Einstellung wird sofort angezeigt.

Ebenfalls von erhöhtem Planungsinteresse ist hier die Verwendung von funktionellen MRT Datensätzen oder MRT Angiogrammen, da durch die Einführung des Endoskops funktionelle Bereiche oder Gefäße verletzt werden könnten (Abbildung *5.14*).

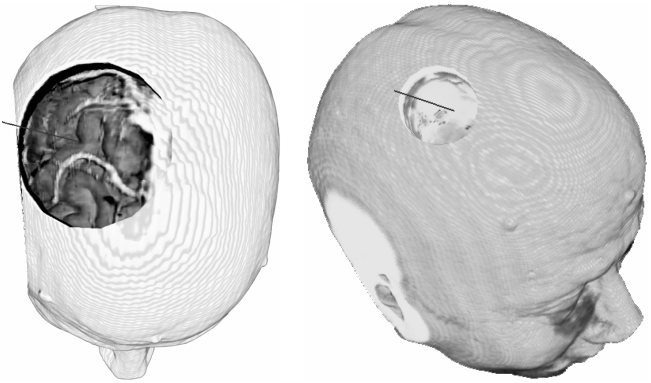

Abbildung 5.14*: Virtuelle Kraniotomie. Virtuelle Öffnung des Schädels und Definiton der Größe, Position und Tiefe der Trepanation. Die Bilder zeigen einen per direktem Volume-Rendering visualisierten MRT-Datensatz eines Patienten, der mit dem entsprechenden fMRT-Datensatz (funktionelle Bereiche im Gehirn sind farbig gekennzeichnet) überlagert worden ist.*

Mit der Planung des Zugangs- und Zielpunktes unter Überprüfung der Trajektorie und der virtuellen Kraniotomie wäre die Planung für einen herkömmlichen Eingriff bereits abgeschlossen. Die Verwendung von NEUROBOT für einen Eingriff bietet allerdings zusätzliche Funktionen, die eine erweiterte Planung benötigen: Das Definieren von aktiven Grenzen.

5.2.3.4 Definition von aktiven Grenzen

Ein weiterer Schritt bei der Planung minimal-invasiver neurochirurgischer Eingriffe mittels ROBO-SIM besteht in der Definition der „Active-Constraints“ oder aktiven Grenzen (siehe Abbildung *5.15*). Damit wird ein Bereich (die Go-Area) definiert, der bei einer realen Operation, durchgeführt mit NEUROBOT, nicht verlassen werden kann. Eine solche Go-Area kann z.B. ein Gehirntumor sein. Damit wird erreicht, dass ein Chirurg nicht versehentlich gesunde Gehirnbereiche verletzt. In der vorhandenen Version können beliebige Ellipsoide als Go-Area definiert werden.

Darüber hinaus wird ein Zylinder als Go-Area benutzt, der von der Trepanationsstelle bis zum Zielpunkt führt und den Durchmesser der Trepanation besitzt. Beim realen Eingriff, wie auch bei der Simulation mit NEUROBOT kann dieser Zylinder für die Annäherungsphase, dem Vordringen des Endoskops in die Nähe des Zielpunktes, ebenfalls nicht verlassen werden.

Zu beachten ist, dass die Active Constraints natürlich verändert werden müssen, wenn bei der realen Operation Fragmentierungen des Gewebes durchgeführt werden. Schon allein beim Öffnen des Schädels verändern sich die präoperativen Bedingungen durch das

so genannte „Brain Shift“ wegen der Druckveränderungen im Gehirn [171]. Um diese Fehler auszugleichen, werden im Rahmen des ROBOSCOPE-Projekts die präoperativen MRT Daten mit intraoperativen dreidimensionalen Ultraschalldaten elastisch koregistriert [172]. Das Ergebnis, die Koregistrierungsmatrix, die beschreibt, wie die einzelnen Voxel des präoperativen Datensatzes dem intraoperativen Datensatz zugeordnet werden, wird benutzt, um die Active Constraints entsprechend der intraoperativen Situation anzupassen.

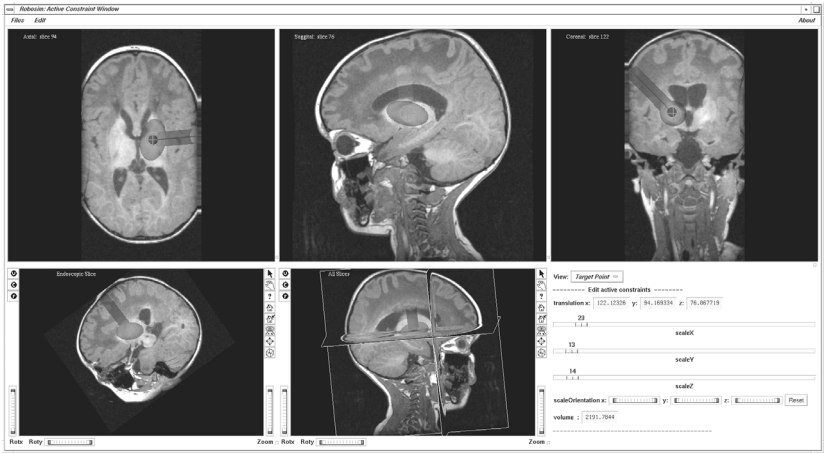

Abbildung 5.15*: Das ‚Active Constraint' Fenster wird zur Definition eines Ellipsoids benutzt, das die Go-Area (erlaubter Bewegungsraum) für den aktiven Manipulator NEUROBOT angibt. NEUROBOT begrenzt dann den Bewegungsraum bei der Durchführung der realen Operation auf das vordefinierte Gebiet. Das Ellipsoid kann durch die Regler unten rechts in Größe, Position und Orientierung beliebig verändert werden (aus* [164]*).*

5.2.4 Simulation

Nachdem die Planungsschritte durchgeführt sind, kann die Simulation gestartet werden. Es ist möglich, eine Ansicht mittels direktem Volume-Rendering für die virtuelle, interaktive Endoskopie zu benutzen [14]. Darüber hinaus ist ein Simulationstool auf der Basis von deformierbaren Oberflächenmodellen integriert. Diese Oberflächenmodelle sind über die Verwendung von Simplex-Gittern (Kapitel 4.2.2) für eine Anzahl von Patienten semiautomatisch erzeugt worden.

Die Simulation der Deformationen basiert auf Feder-Masse Systemen, die durch Neuro-Fuzzy Systeme berechnet werden (Kapitel 2). Mit den Neuro-Fuzzy Systemen ist es möglich, die Eigenschaften realer Gewebe zu erlernen. Außerdem können diese Eigenschaften durch die Angabe einfacher, umgangssprachlicher Regeln (z.B. „Das Gewebe ist weich.“) beschrieben werden. Ein Chirurg mit langjähriger Erfahrung hat so die Möglichkeit, die Gewebeeigenschaften zu charakterisieren, ohne Wissen über die verwende-

ten Verfahren zu besitzen. Die elastodynamische Struktur wird mit einem Feder-Masse System beschrieben, das direkt als Neuronales Netz interpretiert werden kann. Die notwendigen Parameter wurden sowohl durch das Tool „Elastodynamic Shape Modeler“ (Anhang 0), als auch durch einen Lernaufbau mittels Ultraschallphantom bestimmt (Anhang 0).

Eine statische Kollision zwischen dem Ventrikel und dem Endoskop wird unter der Verwendung von OBB-Trees und der RAPID-Bibliothek [66] durchgeführt (siehe auch Kapitel A.1). Dabei wird die Form des Objektes hierarchisch in kleinere Einheiten zerlegt, die einzeln bei Bedarf auf Kollision überprüft werden. Der Aufwand für die Berechnung ist dadurch nur noch logarithmisch von der Komplexität des Objektes abhängig. Um das statische Modell nach jeder Fragmentierung neu zu berechnen, werden zurzeit zwei OBB-Trees des Oberflächenmodells erzeugt. Während ein Modell auf Kollision getestet wird, wird das andere im Hintergrund auf den neuesten Stand gebracht. Leider ist diese Art der Kollisionserkennung bei Modellveränderungen recht aufwendig, sodass die Entwicklung eines zusätzlichen dynamischen Kollisionserkennungsalgorithmus erforderlich war. Die dynamische Kollisionserkennung, eine Erkennung Strahl⇔Dreieck (Kapitel A.2), wird nur bei einer Kollisionsmeldung der statischen Kollisionserkennung durchgeführt. Dadurch kann der Berechnungsaufwand sehr gering gehalten werden.

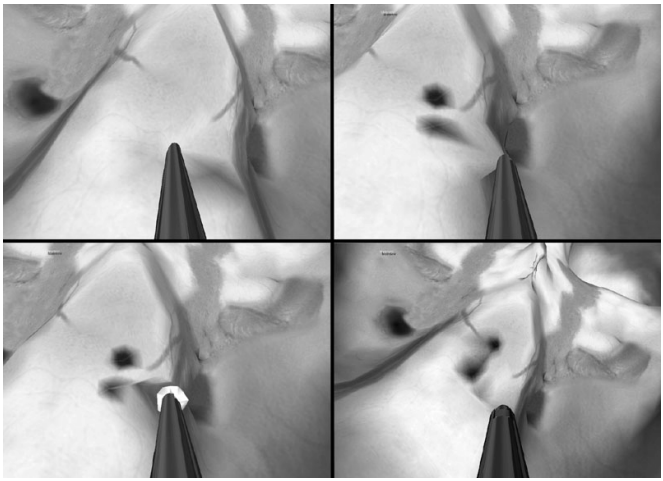

Abbildung 5.16*: Virtuelle, lokale Gewebeentfernung innerhalb des Ventrikels. Von links oben nach rechts unten: Virtuelle Deformation bei Kontakt mit dem transendoskopischen Instrument, virtuelle Gewebeentfernung durch Ziehen mit der Biopsiezange und virtuelle Fragmentierung simuliert durch Verschiebung einiger Punkte normal zum Oberflächengradienten. Zusätzlich verbleibt das entfernte Gewebe in der Biopsiezange. Rechts unten: Blick in den Ventrikel nach der Fragmentierung.*

Abbildung *5.16* zeigt eine simulierte lokale Gewebeentfernung innerhalb des Ventrikels. Zur Beurteilung der Qualität der Simulation zeigt die Abbildung *5.17* eine simulier-

te (links) und eine reale (rechts) neurochirurgische Endoskopie im Vergleich. Gefäße kleiner als 100 µm werden noch dargestellt, wodurch sich eine Darstellungsqualität ergibt, die der einer modernen, digitalen Endoskopie entspricht.

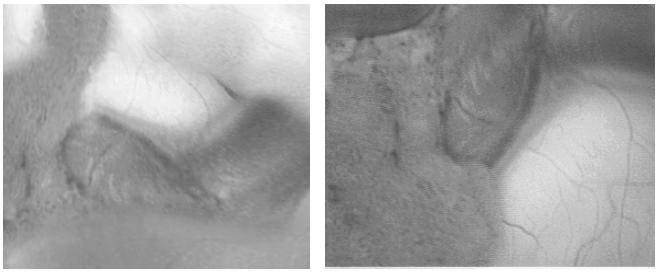

Abbildung 5.17*: Neurochirurgische Endoskopie im Vergleich. Links: simuliert, rechts: real.*

5.2.5 Evaluation von ROBO-SIM

Neben einer generellen Validierung der Virtuellen Endoskopie zum Trainieren und Planen von minimal-invasiven neurochirurgischen Operationen (siehe Kapitel 5.3) wurde ein qualitativer Test von ROBO-SIM durch einen erfahrenen Neurochirurgen durchgeführt [163]. Getestet wurde die vollständige Entfernung eines ventrikulären Tumors (Abbildung *5.18*).

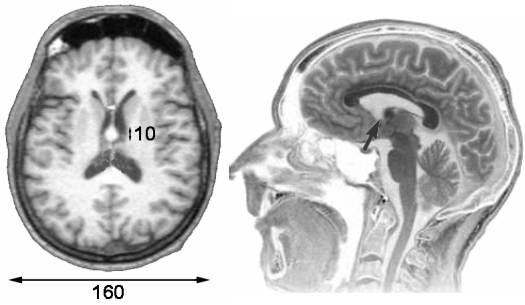

Abbildung 5.18*: Testdatensatz mit einem ventrikulären Tumor (nach* [166]*).*

Das Ergebnis der Evaluation ist in Tabelle *D.1* im Anhang D (S 158) wiedergegeben. Die Funktionalitäten von ROBO-SIM wurden von dem Neurochirurgen mit Punkten bewertet, wobei eine höhere Punktzahl eine bessere Bewertung bedeutet. Die Funktionalitäten sind nach Absprache mit dem evaluierenden Neurochirurgen aufgeführt worden. Zusammenfassend war das Ergebnis des Tests, dass die Planungskomponente zumindest ebenso gut ist, wie aktuelle am Markt befindliche Systeme, aber in der Qualität nicht deutlich darüber hinausgeht. Eine Zugangsplanung mit Hilfe der virtuellen Kraniotomie

ist sehr gut umgesetzt und sinnvoll, um die eigene Planung zu kontrollieren, würde aber von einem erfahrenen Neurochirurgen nicht unbedingt benötigt werden. Die zusätzliche Planung der Ausrichtung des Endoskops, die bei einem herkömmlichen System nicht vorhanden ist, macht eine Verwendung der Virtuellen Endoskopie erforderlich. Die Virtuelle Endoskopie unter Verwendung von Volume-Rendering wird als nicht so gut angesehen wie die mittels Surface-Rendering. Dies liegt insbesondere an der Visualisierung der notwendigen anatomischen Landmarken beim Surface-Rendering. Ganz wesentlich im Planungssystem wurde die Überprüfung der Trajektorie angesehen, da dadurch Schicht für Schicht der Schaden am Gewebe kontrolliert werden kann. Die Pseudo-3D-Ansicht hingegen wurde als „schönes Spielzeug, aber nicht relevant" abgetan. Positiv bewertet wurde auch die Anzahl der vordefinierten Planungsfenster, während die Möglichkeit, sich selbst Planungsfenster zu erstellen mit 0 Punkten abgetan wurde. Dies liegt daran, dass im Prototypenstadium von ROBO-SIM die Planungsfenster über eine Skriptsprache zu definieren sind, die nur von einem Computerfachmann sinnvoll anzuwenden ist.

Der Realismus der Simulation insgesamt wurde mit 3 Punkten als nahezu optimal angesehen. Bemängelt wurde die fehlende Simulation von Blutungen, Spülungen und Membranen. Als sehr gut wurde die Instrumentensteuerung bewertet, die von realen Eingriffen nicht zu unterscheiden ist. Somit ist ROBO-SIM für Studenten ein ideales Werkzeug zum Trainieren von minimal-invasiven Eingriffen in der Neurochirurgie. Für erfahrene Chirurgen ist ROBO-SIM allerdings nur dann sinnvoll einsetzbar, wenn zusätzlich weitere Schritte wie Blutungen, Spülungen, Membrane und Komplikationen simuliert werden.

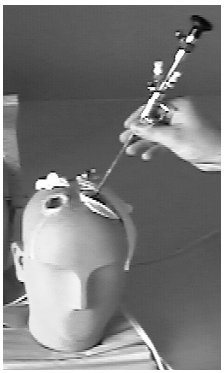

Abbildung 5.19*: MRT-Phantom (aus* [157]*).*

Neben dem rein qualitativen Test an einem Beispielsdatensatz eines realen Patienten wurde ein Test mit MRT-Phantomen durchgeführt, die ein intrakraniales Hämatom und einen Ventrikeltumor simulieren [163]. Das Phantom hat die Form eines Kopfes und besteht aus drei separierten Bestandteilen, die das Gehirngewebe, ein intrakraniales Häma-

tom und einen ventrikulären Tumor simulieren (Abbildung *5.19*). Die Konsistenz der Bestandteile ist ähnlich der realen Gegebenheiten und alle Bestandteile sind sowohl mittels MRT als auch Ultraschall klar sichtbar. Der Tumor im Phantom ist umgeben von einer künstlichen Tumorkapsel und in einer mit klarer Flüssigkeit angefüllten Kammer angebracht.

Sowohl das künstliche Hämatom als auch der Tumor können mit minimal-invasiven neurochirurgischen Instrumenten entfernt werden. Das Gehirngewebe selbst kann nicht entfernt werden, aber es ist möglich, einen spezifischen Druck auszuüben, um normalen bzw. pathologischen interkraniellen Druck zu simulieren. Bei Entfernung des künstlichen Tumors bzw. des Hämatoms wird das künstliche Gehirngewebe verschoben, was den „Brain Shift" simuliert.

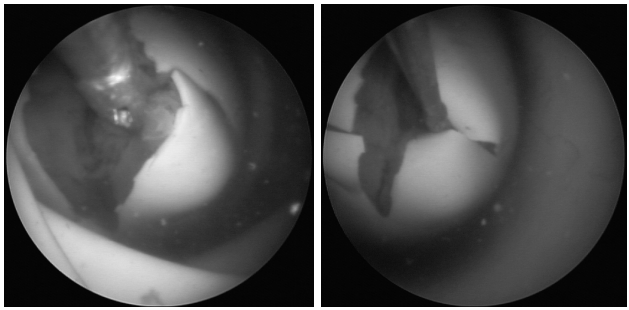

Abbildung 5.20: Entfernung eines ventrikulären Tumors am MRT-Phantom. Innerhalb der Tumorkapsel befindet sich ein Ball aus Schweinefleisch. Links: Entfernung des Fleisches mittels einer Biopsiezange. Rechts: Künstlicher Tumor nach dem Entfernen eines Teiles des Fleisches.

Von dem Phantom wurde ein MRT Datensatz aufgenommen, der als Basis für die Planung und Simulation mit ROBO-SIM verwendet wurde. Sowohl mit ROBO-SIM als auch am Phantom wurde anschließend die Entfernung des künstlichen Tumors durch einen erfahrenen Neurochirurgen durchgeführt (Abbildung 5.20). Die Qualität beider Eingriffe wurde wie oben beschrieben miteinander verglichen. Das Resultat dieses qualitativen Vergleichs war, dass die Simulation mittels ROBO-SIM insgesamt visuell realistischer wirkte. Bei der Entfernung des Tumors im Phantom war allerdings der haptische Realismus höher als im Simulator. Dies liegt insbesondere in der schlechten Qualität der Laparoscopic Impulse Engines, die für die Force-Feedback Funktion Seilzüge verwenden, deren interne Reibung gerade bei der Simulation kleiner Kräfte Probleme bereitet. Ein „Brain Shift" wurde beim Simulieren per ROBO-SIM nicht berücksichtigt, weil bei einer Operation unter ständiger Sicht eine solche Simulation auch nicht so relevant wäre, dass sie den dafür erforderlichen Programmieraufwand rechtfertigt.

Neben einer Qualitätseinschätzung von ROBO-SIM stellt sich die grundsätzliche Frage der Genauigkeit der Virtuellen Endoskopie, also auch der bei der Simulation verwendeten Visualisierung, da man bei der Planung und Simulation vorzüglich Datensätze von

Magnet Resonanz Tomographen benutzt. Von der Güte der Virtuellen Endoskopie, also der Qualität und Sichtbarkeit real vorhandener anatomischer Details in der Simulation, hängt die Einsetzbarkeit eines Simulationssystems, mit dem man reale Eingriffe trainieren kann, wesentlich ab. Aus diesem Grund wurde ein Qualitätsvergleich von verschiedenen Verfahren für die Virtuelle Endoskopie durchgeführt, der im folgenden Abschnitt beschrieben ist.

5.3 Qualitätsvergleiche der Virtuellen Endoskopie

Viele potentielle Anwendungen von der Verbesserung der radiologischen Diagnostik bis zum Erlernen der Neuroanatomie und dem Planen und Trainieren von chirurgischen Eingriffen sind durch den Einsatz der Virtuellen Endoskopie entstanden [6; 10; 13]. Allerdings blieb für klinische Anwendungen die Frage offen, ob die Qualität der Virtuellen Endoskopie ausreichend für die medizinische Entscheidungsfindung bei realen Operationen ist. Daher wurden im Rahmen der Evaluation von ROBO-SIM verschiedene verfügbare Tools für Virtuelle Endoskopie analysiert und deren Qualität im Hinblick auf die Visualisierung von kleinen anatomischen Bereichen des Gehirns verglichen [7]. Zusätzlich fand ein Vergleich der Entfernungen zwischen verschiedenen anatomischen Landmarken in der Virtuellen und realen Endoskopie statt. Abschließend wurde die Genauigkeit von Objektrepräsentationen in der Virtuellen Endoskopie durch den Vergleich mit realen Strukturen gemessen.

5.3.1 Vergleich von Tools für Virtuelle Endoskopie

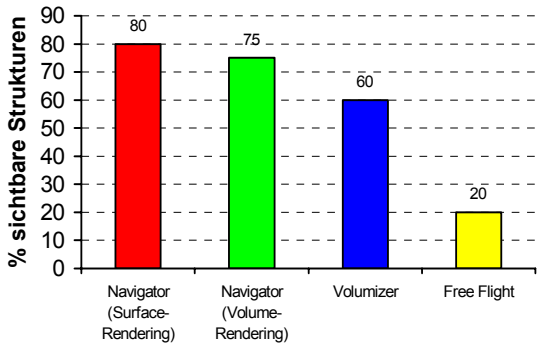

Abbildung 5.21*: Qualität der Visualisierung mit verschiedenen Tools für Virtuelle Endoskopie: mittlere sichtbare anatomische Strukturen pro Standardblickrichtung.*

Die getesteten Tools waren: Navigator (GE Medical Systems, Buc, France) als Surface-Rendering (indirektes Volume-Rendering) und Volume-Rendering (Ray Casting)[49],

OpenGL Volumizer und Free Flight (Wake Forest Univ., indirektes Volume-Rendering) [205]. Als Datensätze wurden MRT-Datensätze von 38 Patienten und Freiwilligen verwendet. Zusätzlich wurden 30 so genannte anatomische „Standardblickrichtungen" definiert, die in [6] beschrieben sind. Die Zusammensetzung der Datensätze ist in Tabelle *D.2* im Anhang D wiedergegeben.

Für die semi-quantitative Analyse wurden die anatomischen Strukturen definiert, die in jeder Standardblickrichtung theoretisch sichtbar sein sollten. Die Prozentzahl der tatsächlich sichtbaren Strukturen wurde als Maß für die Qualität der Virtuellen Endoskopie verwendet. Die statistische Analyse wurde mit dem Tool ANOVA und dem Scheffé-Test durchgeführt [29]. Beispiele für die ersten drei Standardblickrichtungen zeigen Abbildung *D.12* bis Abbildung *D.14* im Anhang D.

Das Gesamtergebnis der Evaluation ist in Abbildung *5.21* wiedergegeben. Die besten Ergebnisse zeigt das Surface-Rendering mit Navigator, gefolgt vom Volume-Rendering mit Navigator. Bei OpenGL Volumizer sind immerhin noch ca. 60% der anatomischen Strukturen sichtbar. Surface-Rendering mit Free Flight ist hingegen von wesentlich schlechterer Qualität.

5.3.2 Vergleich realer und virtueller Endoskopie

Neben dem Vergleich verschiedener Methoden zur Virtuellen Endoskopie wurde zusätzlich ein quantitativer Vergleich zwischen Bildern realer und virtueller Endoskopie durchgeführt [157].

Dafür wurden dreidimensionale MRT-Datensätze von sieben mit formaldehydfixierten Gehirnen aufgenommen, die in Agar-Gel eingebettet wurden. Danach wurden mittels eines herkömmlichen Neuroendoskops (Storz-Auer-Neuroendoskop) und einer digitalen Kamera Fotos aus den 30 Standardblickrichtungen aufgenommen. Zum Vergleich wurden die identischen Blickrichtungen der Tools für Virtuelle Endoskopie herangezogen (Abbildung *5.22*).

Danach wurden die Gehirne axial auf Höhe der Cella Media der lateralen Ventrikel durchgeschnitten und durch keilförmige Einschnitte künstliche Landmarken erstellt. Die Entfernungen zwischen diesen Landmarken in den realen Gehirnen und den virtuellen Ansichten wurden statistisch verglichen.

Der gleiche Test wurde mit einem MRT-Phantom durchgeführt, dass mit exakt definierten Landmarken für die Messung von Distanzen aus Plexiglas konstruiert worden ist. Diese Distanzen wurden wie vorstehend beschrieben mit den Distanzen in der Virtuellen Endoskopie verglichen.

Das Ergebnis dieser Untersuchung zeigt, dass die Genauigkeit Virtueller Endoskopie generell, unabhängig von den Verfahren, größenabhängig ist. Strukturen kleiner einem Millimeter erzeugen Fehler im Bereich von 2-5 Prozent (Relation der Größe der realen Struktur zur Größe der virtuellen Struktur). Die räumliche Genauigkeit ist größer als 98 Prozent für Strukturen bis 10 Millimeter Größe. Strukturen größer als 10 Millimeter werden in der Virtuellen Endoskopie sehr genau wiedergegeben.

Sehr schlecht aufgelöst werden daher dünne Strukturen wie Membrane, die für den Einsatz in der Virtuellen Endoskopie rekonstruiert werden sollten.

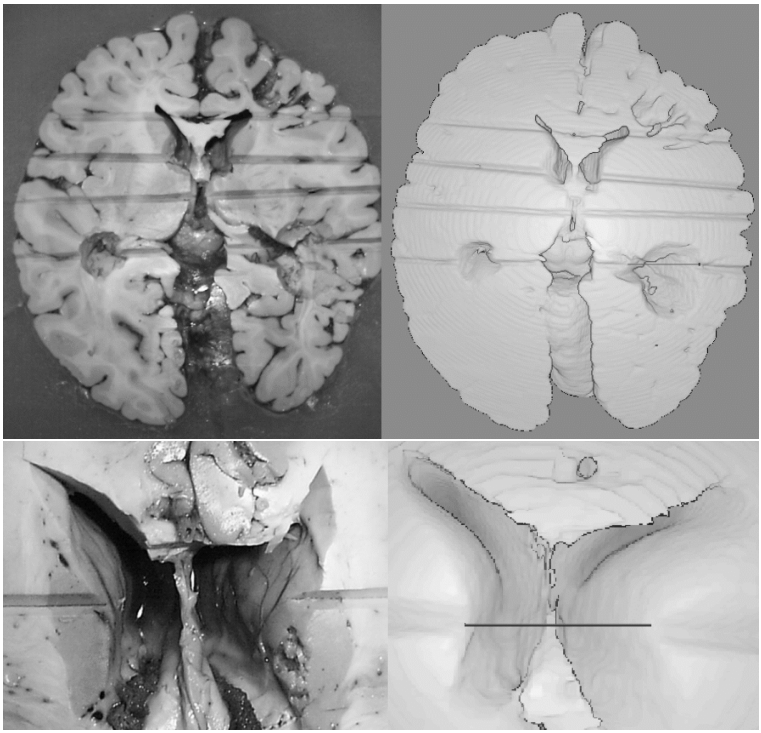

Abbildung 5.22*: Quantitativer Vergleich zwischen einem realen und einem virtuellen Bild eines präparierten Gehirns.*

5.4 Diskussion

Im vorliegenden Kapitel wurden zwei Simulatoren aus den Bereichen Gynäkologie (SUSILAP-G) und Neurochirurgie (ROBO-SIM) präsentiert. Bereits bei dem Simulator SUSILAP-G, der schon 1998 implementiert wurde, zeigt sich die grundlegende Einsatzfähigkeit von Neuro-Fuzzy Systemen für die Simulation von Deformationen. Die Simulation eines einfachen Feder-Masse Systems, modelliert als virtueller Eileiter, genügt, um grundsätzliche Probleme der Auge-Hand-Koordination beim Blick durch ein Endoskop und die Navigation in der menschlichen Anatomie zumindest für einen chirurgischen Studenten begreifbar zu machen. Trotzdem ist der Simulator recht einfach gehalten und genügt nicht den Ansprüchen, die ein angehender Chirurg an ein Simulationssystem stellt. Als Medium für die grundsätzliche Eignung der in dieser Arbeit beschriebenen Methoden und Verfahren hat es jedoch seinen Zweck erfüllt.

Während SUSILAP-G nur eine einfache, modellhafte Anatomie des Menschen beinhaltet, arbeitet ROBO-SIM hingegen mit den vorher aufgenommenen Patientendaten. Dies ermöglicht nicht nur eine Simulation eines beliebigen Eingriffes, sondern auch die Planung dieser Operation anhand eines realen Falles. ROBO-SIM umfasst dabei mehr Planungsschritte als mit heutigen Neuronavigationssystemen möglich wäre und verbessert dadurch die Planungsqualität. Zusätzlich kann der geplante Eingriff simuliert werden. Dies nicht nur zum Trainieren der Operation, sondern als Erweiterung der Planung: Ist der Tumor mit dem geplanten Trepanationspunkt erreichbar? Wie viel gesundes Gewebe wird beim Eindringen des Endoskops zerstört? Diese Daten können dann zu einer Veränderung der Planung im Hinblick auf die Position des Trepanationspunktes führen und somit die Qualität einer nachfolgenden realen Operation verbessern.

ROBO-SIM ist von einem Neurochirurgen mit dem Ergebnis evaluiert worden, dass grundsätzliche Probleme gut zu trainieren sind, für eine professionelle Anwendung aber zu wenig Details simuliert werden. Insbesondere wurde bemängelt, dass Membrane, Blutungen und Spülungen nicht berücksichtigt sind. Im Hinblick auf das Prototypenstadium von ROBO-SIM und darauf, dass dies das erste System weltweit darstellt, das minimalinvasive neurochirurgische Interventionen simulieren kann, ist dessen Qualität aber bemerkenswert. Insbesondere die Qualität der Virtuellen Endoskopie anhand von Oberflächenmodellen ist wegen des Einsatzes des speziell für ROBO-SIM entwickelten und in Kapitel 4.3.2 beschriebenen Texture-Placement Verfahrens von einer realen Endoskopie kaum noch unterscheidbar.

Allerdings war bezüglich der Virtuellen Endoskopie eine generelle Validierung erforderlich: Wie gut entsprechen Darstellungen der Virtuellen Endoskopie der realen Anatomie? Daher wurden verschiedene Tools für Virtuelle Endoskopie mit dem in ROBO-SIM verwendeten Volume-Rendering Verfahren verglichen. Das Ergebnis dieser Studie war, dass kleine Details mit einem Größenfehler (Durchmesser) bis 5% dargestellt werden. Bei größeren Strukturen ist die Genauigkeit noch höher. Im Vergleich verschiedener Verfahren schnitt das für ROBO-SIM verwendete Volume-Rendering Verfahren mit OpenGL Volumizer recht gut ab, wurde aber von den im Navigator verwendeten Ray Casting und Surface-Rendering Verfahren übertroffen. Insbesondere bei der Verwendung moderner Grafikhardware sind hier sowohl Geschwindigkeits- als auch Qualitätsverbesserungen möglich. Eine Implementierung dieser Verfahren für ROBO-SIM erscheint daher sinnvoll.

Die Evaluierung von ROBO-SIM wurde nur von einem Neurochirurgen nach von ihm willkürlichen gewählten Funktionalitäten getestet. Dies liefert eine erste Einschätzung der Verwendbarkeit des Systems. Um aber wirklich aussagekräftige Daten für die Untersuchung der Qualität der Simulation zu erhalten, sind Evaluierungen mit anderen Neurochirurgen nach einer komplexeren Skala erforderlich.

6 Zusammenfassung und Ausblick

Die minimal-invasive Chirurgie bietet gegenüber der offenen Chirurgie für den Patienten und für das Gesundheitswesen als Ganzes erhebliche Vorteile. Eingriffe, die in konventioneller Art mehrtägige Krankenhausaufenthalte zur Folge hätten, können so teilweise ambulant durchgeführt werden – neben dem gesundheitlichen Aspekt im Zusammenhang der Patientenbelastung trägt dies auch zur massiven Kosteneinsparung in der Chirurgie bei.

Allerdings wirft die minimal-invasive Chirurgie das Problem auf, dass chirurgische Eingriffe mit Hilfe von Endoskopen besondere Fertigkeiten und somit eine besondere Ausbildung des Arztes verlangen, die sehr zeitaufwändig ist und viele Jahre Training erfordert. Wegen der Nachteile der herkömmlichen Trainingsmethoden ist sich die Fachwelt einig, dass die Einführung von Chirurgiesimulatoren zur Verbesserung der Trainingsqualität - ähnlich wie die Flugsimulatoren bei der Ausbildung zum Piloten - unabdingbar ist. Durch die enormen Fortschritte in der Computertechnologie wurde es in den letzten Jahren möglich, mit vertretbarem Aufwand virtuelle Operationsumgebungen samt Patienten zu schaffen.

Ein wesentlicher Aspekt, der die Chirurgiesimulatoren von den Flugsimulatoren abhebt und zugleich den technischen Aufwand erheblich erhöht, ist die Simulation der Physiologie und insbesondere der Deformation von Geweben und Organen. Die Deformationssimulation stellt höchste Anforderungen in Bezug auf den Realismus und den Berechnungsaufwand virtueller Interaktionen. Der Berechnungsaufwand ist deshalb so hoch, weil Deformationen interaktiv, also vielmals pro Sekunde ausgeführt werden müssen. Der geringe Realismus bei den üblichen Deformationsmethoden hängt im Wesentlichen von den unbekannten Deformationsparametern medizinischer Gewebe ab. Um diese Probleme zu adressieren, wurde in dieser Arbeit ein Deformationsverfahren auf der Basis von kooperativen Neuro-Fuzzy Systemen vorgestellt, das interaktiv zu berechnen ist und mit dem die Deformationsparameter des physikalischen Modells anhand realer Daten erlernt werden können. Außerdem ist es möglich, die Deformationsparameter mittels einfacher, umgangssprachlicher oder medizinischer Ausdrücke zu initialisieren. Damit kann ein medizinischer Experte sein meist implizit vorhandenes Wissen für die Optimierung der Deformationssimulation zur Verfügung stellen.

Die Möglichkeit zum automatischen Erlernen der Deformationsparameter wurde an zwei Beispielen überprüft: anhand eines konstruierten Experiments und anhand der dreidimensionalen Ultraschallaufnahmen eines speziell entwickelten Phantoms. Durch die fehlende technische Möglichkeit dreidimensionale Aufnahmen deformierten Gewebes in kurzen, für die Interaktivität benötigten Zeitabfolgen zu erstellen, war bisher allerdings eine Überprüfung des automatischen Erlernens des dynamischen Echtzeitverhaltens nur an konstruierten Beispielen möglich. Im Zuge der Entwicklung von Ultraschallgeräten, die dreidimensionale Datensätze in schnellen Abfolgen erzeugen können, wird daher ei-

ne weitere Überprüfung der Lernverfahren erforderlich und auch möglich. Die Firma Kretztechnik, Zipf, Österreich besitzt in dieser Hinsicht die mit am weitesten entwickelten Ultraschallgeräte weltweit.

Die Verwendung eindimensionaler FEM, nämlich Feder-Masse Systeme, ist ein recht einfacher physikalischer Ansatz, der eine theoretisch geringere Realitätsnähe als dreidimensionale FEM besitzt. Mit dem rasanten Fortschritt der Computerhardware sind jetzt auch dreidimensionale FEM interaktiv berechenbar (siehe u.a. [50; 64; 80; 90; 133]), haben aber beim Einsatz in der Chirurgiesimulation nach wie vor den Nachteil, dass durch die fehlende Parameterbestimmung ein geringer Realismus erreicht wird [80; 180; 196]. Zukünftige Arbeiten sollten daher die Anpassung des Neuro-Fuzzy Ansatzes auf dreidimensionale FEM zum Ziel haben.

Eine qualitativ hochwertige Deformationssimulation löst jedoch nur einen Teil der Probleme bei der Erstellung von Chirurgiesimulatoren. Visueller Realismus entsteht im Wesentlichen auch durch die Darstellungsqualität an sich. Insbesondere bei der Verwendung realer Patientendaten ist die technisch machbare Leistungsgrenze heutiger Computersysteme aufgrund der gewaltigen Datenmengen, die interaktiv visualisiert werden müssen, schnell erreicht. Es wurden daher zwei grundsätzliche Konzepte zur Visualisierung dreidimensionaler medizinischer Datensätze vorgestellt: volumenbasierte Verfahren, bei denen die Daten von bildgebenden Diagnoseverfahren direkt verwendet und mit Hilfe des Volume-Renderings dargestellt werden und oberflächenbasierte Verfahren, bei denen nur die Objekthülle als eine Menge von Polygonen visualisiert wird.

Die Anwendung des Volume-Renderings scheint vor allem im Bereich der Neurochirurgie besonders geeignet zu sein, um eine möglichst realistische Darstellung virtueller Endoskopien zu liefern. Während erste Prototypen von Operationssimulatoren noch mit einer statischen Volumendarstellung arbeiten, gehen neuere Ansätze in Richtung einer dynamischen Volumenvisualisierung, in der eine direkte Interaktion mit den Volumendaten möglich ist [39; 169; 191; 210; 212]. Im Rahmen dieser Arbeit wurde eines der ersten Verfahren für das dynamische Volume-Rendering entwickelt [166]. Bei diesem Verfahren wird die eigentliche Deformation durch Neuro-Fuzzy Systeme berechnet und per direktem Volume-Rendering mittels Grafikhardware interaktiv visualisiert. Während die Deformation umfangreicher Teile des Datensatzes in guter Qualität möglich ist, ist eine Deformation in den kleinen Bereichen, die durch ein virtuelles Endoskop sichtbar sind, wegen der schlechten Darstellungsqualität nicht einsetzbar. Dies liegt an der geringen Auflösung der Volumendatensätze, was bei einer Betrachtung aus nächster Nähe zu Artefakten in der Darstellung führt. Für die Simulation von Deformationen in der minimal-invasiven Chirurgie scheint das dynamische Volume-Rendering daher nicht geeignet zu sein. Diese Aussage kann sich in nächster Zeit jedoch durch hochaufgelöstere medizinische Datensätze und schnellere Volume Rendering Verfahren auf moderner Grafikhardware relativieren [95].

Bei der Simulation von Deformationen kann aber auch eine andere Möglichkeit der Darstellung der Patientendatensätze, die Verwendung einer polygonalen Repräsentation der Anatomie, verwendet werden. Hierbei werden die (Organ-)Oberflächen als Polygongitter segmentiert. Im Gegensatz zu dem direkten Volume-Rendering Verfahren handelt

es sich dabei um das so genannte indirekte Volume-Rendering. Das indirekte Volume-Rendering ist mit heutiger Grafikhardware sehr effizient und ermöglicht problemlos eine Echtzeitdarstellung. Geeignete Triangulationsverfahren für das indirekte Volume-Rendering wurden in Kapitel 4 beschrieben. Mit dem indirekten Volume-Rendering sind auch Manipulationen wie Greifen und Schneiden leicht und elegant zu implementieren – entsprechende Verfahren sind ebenfalls im Kapitel 4 formuliert worden.

Im Gegensatz zum direkten Volume-Rendering besteht beim indirekten Volume-Rendering zusätzlich die Möglichkeit, den Realismus der Darstellung erheblich zu erhöhen, indem Aufnahmen von realen Endoskopien verwendet werden, um diese auf die Oberfläche des Modells aufzubringen. Dieses Verfahren – das Texturemapping – ist bereits in der überwiegenden Mehrzahl aller auf dem Markt befindlicher Grafikhardware implementiert und kann daher ohne Geschwindigkeitsverlust für die Darstellung verwendet werden. Für die Chirurgiesimulation ist die Anwendung herkömmlicher Texturemapping Verfahren aber nicht sinnvoll, da diese die Oberflächendetails nur mit Verzerrungen aufbringen können. Im Rahmen dieser Arbeit wurde daher ein neues Texture-Placement Verfahren entwickelt, mit dem anatomische Details korrekt, verzerrungsfrei und in sehr hoher Qualität platziert werden können. Die dadurch erreichbare Grafikqualität macht eine Unterscheidung zwischen realem und virtuellem Bild selbst für den medizinischen Experten beinahe unmöglich.

Die in den ersten Kapiteln beschriebenen Verfahren für die Deformationssimulation und Visualisierung sind zwar generell für die Anwendung im Bereich der Virtuellen Realität einsetzbar, wurden aber speziell für den Einsatz in der Chirurgiesimulation entwickelt und optimiert. Kapitel 5 beschreibt die zwei Chirurgiesimulatoren aus den Bereichen Gynäkologie (SUSILAP-G) und Neurochirurgie (ROBO-SIM), für die die im Rahmen dieser Arbeit entwickelten Verfahren und Methoden implementiert worden sind. Beide Simulatoren richten sich, von den verwendeten Methoden und von der Qualität der Simulation, an grundsätzlich unterschiedliche Anwender. SUSILAP-G ist für den medizinischen Anfänger gedacht, der sich grundsätzlich mit der Navigation mittels endoskopischer Instrumente vertraut machen will. Die Anatomie ist künstlich und nicht anhand realer Datensätze erzeugt.

ROBO-SIM hingegen arbeitet mit den vorher aufgenommenen Patientendaten und richtet sich daher an den medizinischen Experten. Dies ermöglicht nicht nur eine Simulation eines beliebigen Eingriffes, sondern auch die Planung dieser Operation anhand eines realen Falles. ROBO-SIM umfasst dabei mehr Planungsschritte als mit heutigen Neuronavigationssystemen möglich wäre und verbessert dadurch die Planungsqualität. Zusätzlich kann der geplante Eingriff simuliert werden.

ROBO-SIM ist von einem Neurochirurgen mit dem Ergebnis evaluiert worden, dass grundsätzliche Probleme gut zu trainieren sind, für eine professionelle Anwendung aber zu wenig Details simuliert werden. Insbesondere wurde bemängelt, dass Membrane, Blutungen und Spülungen nicht berücksichtigt sind. Im Hinblick auf das Prototypenstadium von ROBO-SIM und darauf, dass dies das erste System weltweit darstellt, das minimalinvasive neurochirurgische Interventionen simulieren kann, ist dessen Qualität aber be-

merkenswert. Die Qualität der Virtuellen Endoskopie mittels indirektem Volume-Rendering ist von einer realen Endoskopie kaum noch unterscheidbar.

Obwohl eine erste Evaluierung des Systems durch einen erfahrenen Neurochirurgen durchgeführt worden ist, sind weitere Tests mit anderen Neurochirurgen erforderlich, um eine repräsentative Aussage über die Güte des Systems zu treffen. Bevor diese Tests durchgeführt werden sollten, ist eine Neuimplementierung des Systems auf PC-Hardware sinnvoll, um die neuen, in diesem Gebiet entwickelten Technologien, beispielsweise frei-programmierbare Grafikprozessoren, zur Qualitätsverbesserung und Geschwindigkeits-erhöhung zu verwenden.

Anhang A KOLLISIONSERKENNUNG

Wenn in einem interaktiven Simulationssystem mehrere Objekte existieren, die relativ zueinander bewegt werden können, ist es wichtig, Kollisionen zwischen diesen Objekten zu testen und darauf entsprechend zu reagieren. Eine mögliche Reaktion ist in diesem Fall die Verhinderung der Weiterbewegung. Im schwierigeren Fall ist auch eine Deformation bis hin zu einer Destruktion zu simulieren.

Ein Unterscheidungsmerkmal ist auch die Kollisionserkennung unter Benutzung von Volume-Rendering und Surface-Rendering. Beim Surface-Rendering kann die Kollision direkt mit den Oberflächenelementen getestet werden, beim Volume-Rendering muss eine geeignete Kollisionsoberfläche gefunden werden. Dies können z.B. Isosurfaces oder triangulierte Oberflächenstrukturen sein, die aber nicht unbedingt grafisch dargestellt werden müssen (siehe auch Kapitel 3.3.2). Ist das Oberflächenmodell gefunden, so kann mit diesem eine Kollisionserkennung wie beim Surface-Rendering durchgeführt werden.

Kollisionserkennung kann in Abhängigkeit der Art der beteiligten virtuellen Modellteile in unterschiedlichster Weise realisiert werden. Das geht von der exakten Berechnung der Berührungen unter Berücksichtigung gesamter Objektoberflächen bis hin zur teilweisen Vereinfachung der zu überwachenden Objekte durch einfache geometrische Primitive, wie Linien oder Zylinder. Für die Anwendung in der minimal-invasiven Virtuellen Chirurgie kommen aufgrund der häufigsten Art der Kollision – Endoskop und Gewebe - in der Regel nur sehr wenige Verfahren in Frage, die im Folgenden kurz beschrieben werden:

Bei der Kollisionserkennung kann zwischen einer rein statischen und einer dynamischen unterschieden werden. Bei der statischen Kollisionserkennung geht es im Wesentlichen um die Verhinderung des Eindringens eines Objektes in ein anderes, ohne dass sich dabei direkt etwas am Aussehen und der Form ändert. Diese Art der Kollisionserkennung hat somit keinen Einfluss auf die Geometrie der beteiligten Objekte, sondern nur auf die Bewegungsfreiheit [111].

Bei der dynamischen Kollisionserkennung ist zumindest ein deformierbares Objekt beteiligt. Bei einer Berührung kommt es vorerst zu einer Verformung des Objektes, welches das Eindringen des Berührenden verhindert. Erst bei entsprechend hohen Kräften zwischen den Objekten wird auch die Weiterbewegung und somit ein echtes Eindringen als solches unterbunden, oder es kann auch zur Zerstörung (z.B. Reißen) des Objektes kommen. Für hochflexible Objekte wie Kleidungsstücke sind aufgrund der komplexen Gestalt bei Deformierung spezielle Ansätze notwendig, da solche Objekte nicht nur mit anderen, sondern durch die hohe Flexibilität auch massiv mit sich selbst kollidieren können [101; 148].

A.1 Statische Kollisionserkennung

Bei Oberflächenmodellen werden bei der *statischen Kollisionserkennung* zur Performance-Steigerung die Oberflächenelemente in Form von hierarchischen Bäumen verwaltet - ein Beispiel dafür ist der so genannte „Oriented-Bounded-Box-Tree“ (OBB-Tree) [67]. Beim Testen auf Kollision kann durch die Position und Ausrichtung der beteiligten Objekte dann unmittelbar ein Teilbaum (Ast) ermittelt werden und somit reduziert sich drastisch die Anzahl der zu testenden Polygone. Weitere Ansätze zur statischen Kollisionserkennung bei großen Objekten finden sich in [216].

Die Erzeugung eines solchen Baumes stellt einen nicht zu vernachlässigenden Rechenaufwand dar. Bei strukturell unveränderlichen Objekten ist dieser ein einziges Mal vorauszuberechnen. Wenn sich allerdings das Objekt in seinem Aufbau oder durch Deformation in seiner Gestalt verändert, ist diese Baumerzeugung u.U. zu wiederholen, wodurch eine Berechnung in Echtzeit zumindest schwierig wird. Oftmals wird versucht, den Berechnungsaufwand zu reduzieren, indem eine Neuberechnung des OBB-Tree’s nur in diskreten Zeitabständen, etwa zehn mal pro Sekunde, durchgeführt wird [164]. Die Diskrepanz zwischen dem Deformationsobjekt und dem Kollisionsobjekt, die innerhalb dieser Zehntelsekunde auftreten kann, wird dabei vernachlässigt.

Bei Oberflächenmanipulationen, wie beispielsweise ein Schneidevorgang, werden Oberflächenelemente gelöscht und neue hinzugefügt – dementsprechend verliert ein berechneter OBB-Tree seine Gültigkeit, eine Aktualisierung wäre erforderlich. Der notwendige zusätzliche Rechenaufwand bei dynamisch veränderlichen Objekten ist meist nicht akzeptabel, weswegen bei der Virtuellen Chirurgie größtenteils von einer hierarchischen Polygonverwaltung Abstand genommen wird.

A.2 Dynamische Kollisionserkennung

Bei der *dynamischen Kollisionserkennung* wird auf die Erzeugung von hierarchischen Polygonstrukturen verzichtet. Aus diesem Grund sind bei einer Kollisionsprüfung grundsätzlich alle Oberflächenpolygone auf Berührung zu testen. Als Vereinfachung wird angenommen, dass bei der Kollisionserkennung die Berührung von einem komplexen Oberflächenmodell (welches aus einer bestimmten Anzahl von Polygonen besteht) und einem simplen geometrischen Objekt zu testen ist. Diese Vereinfachung eignet sich insbesondere in der Virtuellen Chirurgie: Das komplexe Objekt entspricht dem Organ und das einfache geometrische Modell dem chirurgischen Instrument. Durch die hohe Frequenz der Kollisionserkennung wird trotzdem ein hoher Rechenaufwand, welcher proportional mit der Polygonanzahl korreliert, benötigt. Wenn die Kollisionserkennung im Zusammenhang mit Force-Feedback Verwendung eingesetzt wird, muss die Testfrequenz sogar im kHz-Bereich liegen [152; 193; 220].

Somit ergibt sich die Notwendigkeit, erhebliche Vereinfachungen durchzuführen und sehr effiziente Algorithmen für folgende geometrische Kollisionen zu entwickeln:

Strahl (oder begrenzte Linie) ⇔ Dreieck,
Dreieck ⇔ Dreieck,
Kugel ⇔ Dreieck oder
Zylinder ⇔ Dreieck

Mittlerweile wurden bereits viele optimierte Algorithmen zur Realisierung solcher Aufgaben veröffentlicht (u.a. [76; 86; 128; 129]).

Im Zusammenhang mit der Virtuellen Endoskopie ist ein wesentliches Kollisionsobjekt das Endoskop. Geometrisch betrachtet ließe sich dieses näherungsweise sehr gut durch einen Zylinder darstellen, allerdings sind die dafür erforderlichen Berechnungen für die Berührungserkennung sehr aufwändig [86]. Besser geeignet ist für die Virtuelle Chirurgie die Strahl⇔Dreiecks-Erkennung. Eine besonders schnelle Implementierung wird in [129] beschrieben. Wesentlicher Unterschied bei der Strahl⇔Dreiecks-Erkennung ist, dass an der Kollision immer nur ein Dreieck beteiligt ist (siehe Abbildung *A.1*).

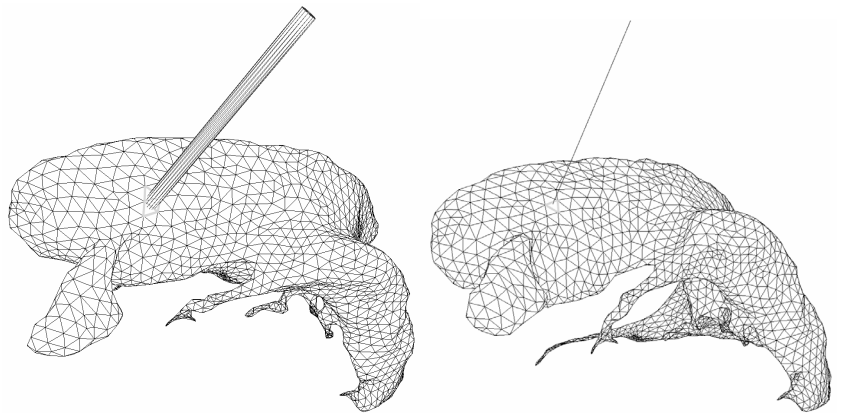

Abbildung A.1*: Unterschied zwischen einer Kollisionserkennung Zylinder⇔Dreieck (links) und Strahl⇔Dreieck (rechts).*

Ebenfalls gut geeignet für die Anwendung in der Chirurgiesimulation ist der in [128] veröffentlichte schnelle Dreieck⇔Dreieck-Kollisionstest. Dieser wird insbesondere für die Simulation von Schnitten benötigt (siehe Kapitel 4.4.3.2).

Ein außergewöhnliches Verfahren zur dynamischen Kollisionserkennung in der Virtuellen Chirurgie wurde von Lombardo entwickelt [113]. Bei diesem Verfahren, welches wegen seiner Sonderstellung im Folgenden beschrieben wird, werden Eigenschaften der Grafikhardware unter OpenGL genutzt, die eigentlich nur zur Darstellung entwickelt worden sind. Ein starres Endoskop wird hier sozusagen als gedachte Kamera mit definiertem Öffnungswinkel und festgelegter Darstellungstiefe betrachtet. Wird das Endoskop bewegt, so wird durch jeden inkrementellen Bewegungsschritt ein Volumenbereich

aufgebaut, der durch eine Sichtpyramide (View-Frustum) von OpenGL beschrieben werden kann (siehe Abbildung *A.2* und auch Kapitel 3.2.3).

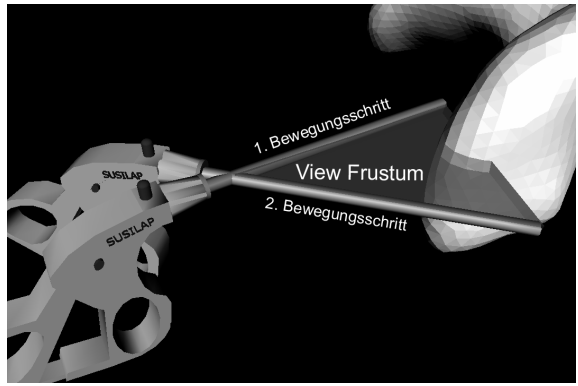

Abbildung A.2*: Verwendung einer Sichtpyramide (View Frustum) für die Kollisionserkennung. Alle Polygone innerhalb der Sichtpyramide werden auf Kollision getestet (aus* [153]*).*

OpenGL hat Funktionen, um die Flächen zu ermitteln, die sich innerhalb der Sichtpyramide befinden. Falls dies der Fall ist, so ist eine Kollision aufgetreten, an denen die entsprechenden Flächen beteiligt waren. Diese Art der Kollisionserkennung ist in der Theorie besonders schnell (ca. 100 mal schneller als Softwarelösungen), da sie direkt auf der Grafikhardware ausgeführt werden kann [113].

Allerdings hat sich das Verfahren im praktischen Einsatz als nicht sehr stabil in der Ausführung erwiesen. Dies liegt insbesondere an der unterschiedlichen Implementierung von OpenGL auf verschiedener Hardware. Auf Computern von Silicon Graphics führte die Anwendung des Verfahrens wegen der expliziten Nummerierung (Markierung mit OpenGL-Befehlen) jedes Polygons zu einer wesentlich verlangsamten Grafikdarstellung, sodass der Geschwindigkeitsvorteil zunichte gemacht wurde. Bei diversen Grafikkarten auf PC-Basis unter Microsoft Windows NT und Windows 2000 führte das Verfahren sogar zu einem Absturz des gesamten Systems.

Anhang B OPERATOREN FÜR FUZZY MENGEN

Um mit Fuzzy Mengen arbeiten zu können, müssen klassische Mengenoperatoren und erweiterte Operatoren für die Grundrechenarten vorhanden sein. Diese Operatoren werden im Folgenden beschrieben:

Um geeignete Aggregationsmöglichkeiten für Zugehörigkeitsfunktionen zu finden, ist es nahe liegend, zunächst die klassischen Mengenoperatoren: Schnitt und Vereinigung auf Fuzzy Mengen zu übertragen.

Der Schnittoperator für Fuzzy Mengen sollte folgende Eigenschaften aufweisen:

1. Der Schnitt einer Fuzzy Menge mit einer gewöhnlichen Menge darf nur zum Ausschluss von Elementen oder zum Erhalt des vorliegenden Zugehörigkeitsgrades führen.
2. Monoton wachsende Zugehörigkeitsgrade in den Argumenten müssen auch monoton wachsende Zugehörigkeitsgrade im Schnitt zweier Fuzzy Mengen nach sich ziehen.
3. Außerdem sind Kommutativität und Assoziativität als selbstverständlich vorauszusetzen.

Diese vier Punkte werden durch die Definition der *t-Norm* erfüllt:

Definition B.1:

Eine Funktion $\mathsf{T}: [0,1]^2 \to [0,1]$ heißt *t-Norm*, wenn die Bedingungen

$\mathsf{T}(a,1) = a$	(Einselement)
$a \le b \Rightarrow \mathsf{T}(a,c) \le \mathsf{T}(b,c)$	(Monotonie)
$\mathsf{T}(a,b) = \mathsf{T}(b,a)$	(Kommutativität)
$\mathsf{T}(a, \mathsf{T}(b,c)) = \mathsf{T}(\mathsf{T}(a,b),c)$	(Assoziativität)

erfüllt sind.

Dual zum Begriff der t-Norm, mit dem generalisierte Schnittoperatoren realisiert werden können, wird der Begriff der *t-Conorm* benutzt, der für die Definition verschiedener generalisierter Vereinigungsoperatoren herangezogen werden kann.

Definition B.2:

Eine Funktion $\perp: [0,1]^2 \to [0,1]$ heißt genau dann *t-Conorm*, wenn $\perp$ kommutativ, assoziativ, monoton nicht-fallend in beiden Argumenten ist und 0 als Einheit besitzt.

t-Normen und t-Conormen induzieren Konnektive für Fuzzy Mengen vermittels

$$(\mu \cap_{\mathsf{T}} \mu')(x) := \mathsf{T}(\mu(x), \mu'(x)) \text{ und}$$

$$(\mu \cup_{\perp} \mu')(x) := \perp(\mu(x), \mu'(x)).$$

Mit diesen Definitionen lassen sich als Spezialfälle verschiedene Durchschnitts- und Vereinigungsoperatoren definieren, indem die Funktionen elementweise für alle $\mu, \mu' \in \mathrm{F}(X), x \in X$ ausgeführt werden.

Anhang C PARAMETERADAPTION VIRTUELLER GEWEBE

C.1 SUSILAP-G: Fuzzy System

Das Fuzzy System, das zur Bestimmung der Deformationsparameter der Gewebe im Chirurgiesimulator SUSILAP-G verwendet wurde, ist in Abbildung *C.1* bis Abbildung *C.3* sowie Tabelle *C.1* angegeben. Eingabedomain ist consistency (Konsistenz), Ausgabedomains sind spring_const (Federkonstante) und viscosity (Viskosität). Die Eingabewerte wurden interaktiv mit dem Elastodynamic Shape Modeler auf 5,5 für das Feder-Masse System des Eileiters, auf 2,5 für die Verknüpfung des Eileiters mit dem Hintergrund und auf 10,1 für die globale Deformierung des Uterus. Nach Berechnung durch das Fuzzy System wurden die Parameter jeweils homogen für alle Federn der Modelle mit 16,2606 (Federkonstante), 2,87713 (Viskosität) für den Eileiter, 2,41327 (Federkonstante), 10,6875 (Viskosität) für die Hintergrundverknüpfung und auf 81,3091 (Federkonstante), 6 (Viskosität) für den Uterus eingestellt. Die Massen wurden mit 10 g für die Eileiter, 500 g für den Uterus und jeweils 100 g für die Ellipsoide des Darmes initialisiert. Das Gewicht ist gleichmäßig auf die Masseknoten der Feder-Masse Systeme aufgeteilt worden.

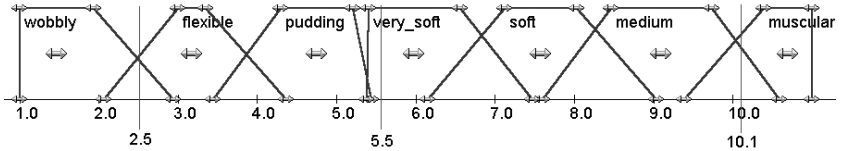

Abbildung C.1*: Fuzzy Mengen der Domain consistency. Die Eingabewerte für die Federn des Eileiters (Mitte, Wert 5,5), der Verknüpfung des Eileiters mit dem Hintergrund (links, Wert 2,5) und des Feder-Masse Systems für den Uterus (rechts, Wert 10,1) sind durch gestrichelte Linien markiert.*

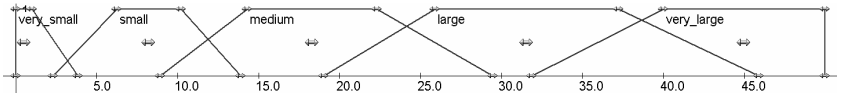

Abbildung C.2*: Fuzzy Mengen der Domain spring_const.*

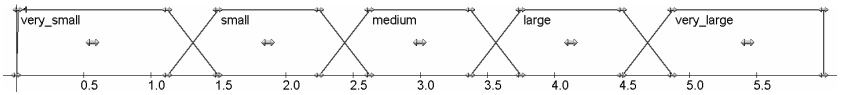

Abbildung C.3*: Fuzzy Mengen der Domain viscosity.*

IF consistency IS flexible	THEN	spring_const IS small	AND	viscosity IS small
IF consistency IS muscular	THEN	spring_const IS very_large	AND	viscosity IS medium
IF consistency IS pudding	THEN	spring_const IS small	AND	viscosityIS very_large
IF consistency IS wobbly	THEN	spring_const IS small	AND	viscosity IS very_small
IF consistency IS very_soft	THEN	spring_const IS very_small	AND	viscosity IS very_large
IF consistency IS soft	THEN	spring_const IS small	AND	viscosity IS large
IF consistency IS medium	THEN	spring_const IS medium	AND	viscosity IS medium

Tabelle C.1*: Regelbasis zur Bestimmung der Deformationsparameter für die Feder-Masse Modelle von SUSILAP-G.*

Die Parameter für die Simulation des Darmes wurden ebenfalls durch das oben beschriebene Fuzzy System ermittelt. Eingabewert für die Domain consistency (siehe Abbildung *C.1*) war 7,0 und die berechneten Parameter 11,3536 (Federkonstante) und 9,13015 (Viskosität). Durch diese Einstellung konnte erreicht werden, dass die Darmbewegungen sehr viskos wirken und die einzelnen Segmente des Darms auch bei stärkeren Kollisionen und dadurch resultierenden Auslenkungen noch eine sichtbare Einheit bilden.

C.2 ROBO-SIM: Fuzzy System

Das Fuzzy System, das zur Bestimmung der Deformationsparameter verwendet wurde, ist in Abbildung *C.4* bis Abbildung *C.8* sowie Tabelle *C.2* angegeben. Eingabedomains sind consistency (Konsistenz), shiftability (Verschieblichkeit) und sensitiveness (Sensitivität), Ausgabedomains sind spring_const (Federkonstante), viscosity (Viskosität).

Die Eingabewerte wurden interaktiv mit dem Elastodynamic Shape Modeler auf 7,3 für consistency, 4,9 für sensitiveness und 3,0 für shiftability. Nach Berechnung durch das Fuzzy System wurden die Parameter jeweils homogen für alle Federn auf 45,4871 für spring_const, und 72,8125 für viscosity eingestellt. Als Gewicht des Ventrikels wurde 1 kg verwendet, was eher ein fiktiver Wert ist, da sich die deformierbare Masse in der Realität außerhalb des Ventrikels befindet. Das Gewicht wurde gleichmäßig auf die Masseknoten des Ventrikelmodells aufgeteilt.

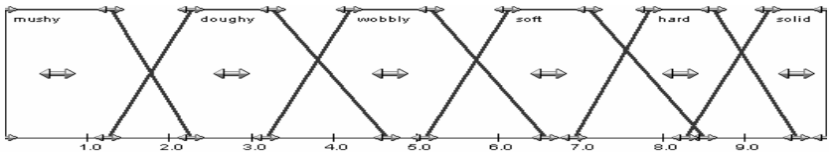

Abbildung C.4*: Fuzzy Mengen der Domain consistency.*

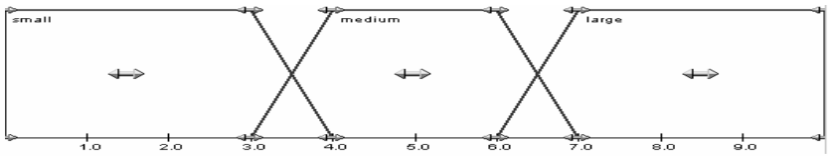

Abbildung C.5*: Fuzzy Mengen der Domain shiftability (aus* [160]*).*

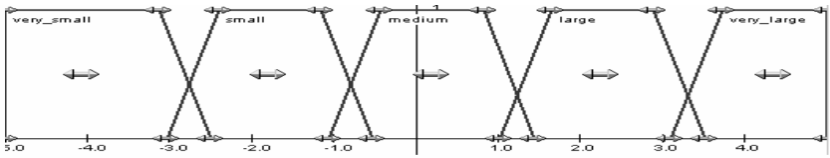

Abbildung C.6*: Fuzzy Mengen der Domain sensitiveness (aus* [160]*).*

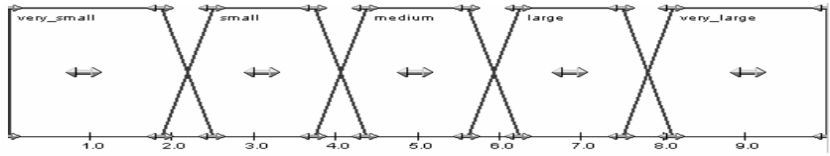

Abbildung C.7*: Fuzzy Mengen der Domain viscosity (aus* [160]*).*

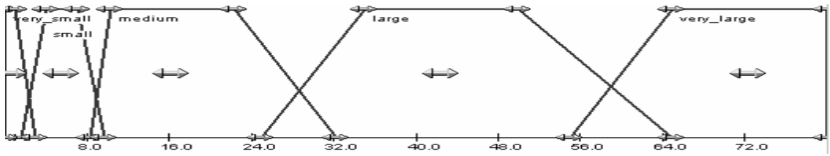

Abbildung C.8*: Fuzzy Mengen der Domain spring_const. (aus* [160]*).*

IF Consistency IS solid	THEN spring_const IS very_large	
IF Consistency IS hard	THEN spring_const IS large	
IF Consistency IS soft	THEN spring_const IS small	
IF Consistency IS wobbly	THEN spring_const IS medium	AND viscosity IS small
IF Consistency IS doughy	THEN spring_const IS very_small	AND viscosity IS very_large
IF Consistency IS mushy	THEN spring_const IS large	AND viscosity IS large
IF Sensitiveness IS very_large	THEN viscosity IS very_small	AND spring_const IS very_small
IF Sensitiveness IS large	THEN viscosity IS small	AND spring_const IS small
IF Sensitiveness IS medium	THEN viscosity IS medium	AND spring_const IS medium
IF Sensitiveness IS small	THEN viscosity IS large	AND spring_const IS small
IF Sensitiveness IS very_small	THEN viscosity IS very_large	AND spring_const IS very_small
IF Shiftability IS large	THEN viscosity IS very_large	AND spring_const IS medium
IF Shiftability IS medium	THEN viscosity IS medium	AND spring_const IS small
IF Shiftability IS small	THEN viscosity IS very_small	AND spring_const IS small

Tabelle C.2*: Regelbasis zur Bestimmung der Deformationsparameter für das Feder-Masse System des Ventrikels (aus* [160]*).*

Neben der Parameterbestimmung durch das Fuzzy System, die sich bereits für den Simulator SUSILAP-G bewährt hat, wurde für ROBO-SIM auch das automatische Lernen von Parametern getestet, was im nächsten Kapitel beschrieben wird.

C.3 ROBO-SIM: Lernexperiment

Um das in Kapitel 2.3.5.1 beschriebene Lernverfahren für die automatische Parameterbestimmung eines anatomischen Feder-Masse Modells zu benutzen, werden digitale medizinische Datensätze benötigt, die die dreidimensionale Deformation anatomischer Strukturen abbilden können. MRT kommt dafür in der Regel nicht in Frage, da durch die lange Untersuchungsdauer dynamische Aufnahmen nicht möglich sind. Für einen ersten Test wurde daher ein Ultraschallphantom entwickelt, das einen Ballon enthält, der mit einem Kontrastmedium für Ultraschall gefüllt ist (Abbildung *C.9*). Der Druck in dem Ballon kann verändert werden, um sowohl dessen Größe als auch seine Form zu variieren.

Mit dem Ultraschallgerät Voluson 530D (Kretztechnik, Zipf, Österreich), das in der Lage ist, dreidimensionale Datensätze mit einer Auflösung von 184^3 Pixel in schneller Abfolge zu generieren, sind fünf Datensätze des Phantoms zu unterschiedlichen Zeiten aufgenommen worden, während der Druck im Ballon erhöht worden ist. Insgesamt wurden fünf Sekunden Deformation registriert. Mit dem in Kapitel 4.2.2 beschriebenen Triangulationsverfahren wurde die Form des Ballons durch ein Feder-Masse System angenähert, welches aus 102 Masseknoten und 300 Federn besteht.

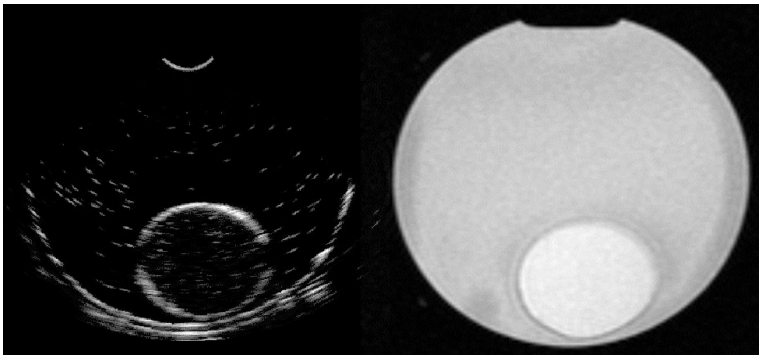

Abbildung C.9*: Links: Ultraschallbild des Phantoms bestehend aus einem flüssigkeitsgefüllten Ballon in einem Glasbehälter. Rechts: MRT-Bild des gleichen Phantoms.*

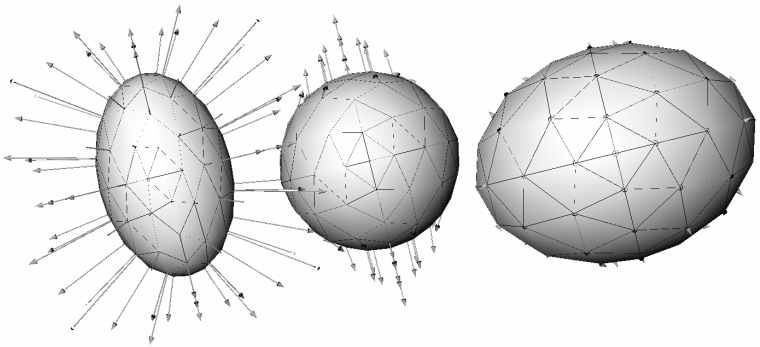

Abbildung C.10*: Externe Kräfte zur Simulation des Drucks im Ballon. Links: Anlegen externer Kräfte an jeden Masseknoten (skaliert mit 0,5). Mitte: Interne Kräfte nach einem Drittel der Zeitschritte (skaliert mit 5, die externen Kräfte wurden ausgeblendet). Rechts: Form des Ballons im Kräftegleichgewicht.*

Die fünf generierten Datensätze wurden benutzt, um die Federkonstanten des Feder-Masse Systems zu lernen, wobei die Positionen der Masseknoten zu den diskreten Zeitschritten als Lerndaten benutzt wurden. Externe Kräfte, die proportional zum Druck und parallel zu den Punktnormalen liegen, wurden an jeden Masseknoten angelegt (Abbildung *C.10*, links) und die Masse jedes Knotens wurde vordefiniert durch die Masse des Ballons geteilt durch die Anzahl der Knoten.

Leider ist mit dem verwendeten Ultraschallgerät, obwohl es sich dabei um das zum Zeitpunkt des Tests schnellste Gerät am Markt handelte, keine Echtzeitmessung möglich. Daher wurde das Lernen der Massen und der Viskositäten abgeschaltet, weil dynamisches Echtzeitverhalten nicht durch die Verwendung von Datensätzen statischer Modelle erlernt werden kann. Die Anzahl der aufgenommenen Volumina – eines pro Sekunde –

genügt nicht für die Aufstellung eines dynamischen Modells. Die Federkonstanten wurden zufällig initialisiert.

Die Zeitkonstante für die Propagation wurde mit 0,02s definiert, sodass der Lernalgorithmus fehlende Positionen während des Lernens berücksichtigen musste. Die teacher forcing Begrenzung wurde auf eins gesetzt, damit die gemessenen und simulierten Knotenpositionen für jeden verfügbare Datensatz synchronisiert wurden. Der Lernalgorithmus terminierte nach 100 Lernzyklen mit einem Fehler $E \approx 9.11$ (Gleichung (2.29)), wobei für jeden Lernzyklus eine komplette Propagation nach der Zeit durchgeführt wurde. Das Ergebnis des Lernprozesses ist in Abbildung *C.11* abgebildet (Learning c).

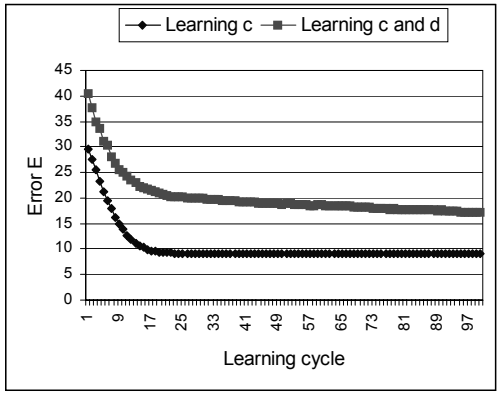

Abbildung C.11*: Lernergebnis für den elastischen Ballon: Fehler E.*

Außerdem wurde das Ergebnis der Simulation (Abbildung *C.10*, rechts) verglichen mit dem realen Volumen des Phantoms. Die Werte variierten um weniger als 5% im Kräftegleichgewicht.

Anhang D EVALUATION VON ROBO-SIM

D.1 Qualitative Evaluation von ROBO-SIM

Funktionalität	4	3	2	1	0
Dialog zum Laden von Patientendatensätzen	X				
Planung des Zugangs			X		
Planung des Zielpunktes			X		
Virtuelle Endoskopie (Volume-Rendering)			X		
Virtuelle Endoskopie (Surface-Rendering)		X			
Pseudo-3D-Ansicht mit Trajektorie				X	
Überprüfung der Trajektorie	X				
Anzahl und Qualität der vorhandenen vordefinierten Planungsfenster		X			
Erstellen neuer Planungsfenster					X
Virtuelle Kraniotomie		X			
Realismus Simulation (Gesamteindruck)		X			
Realismus Tumorentfernung		X			
Realismus Instrumentensteuerung	X				
Eignung zum Trainieren durch Studenten	X				
Eignung zum Trainieren durch erfahrene Chirurgen				X	

Tabelle D.1*: Ergebnis der qualitativen Evaluation von ROBO-SIM (höhere Punktzahlen bedeuten ein besseres Ergebnis) (nach* [157]*).*

D.2 Validierung der Virtuellen Endoskopie: Tabellen und Abbildungen

Indikation	Anzahl	Indikation	Anzahl
Normal brain anatomy	1	Glioblastoma	1
Hydrocephalus	7	Glioma	2
Hydrocephalus occlusus	2	Prolactinoma	1
Aquaeductal stenosis	2	Meningioma	3
NPH	4	Meduloblastoma	1
Brain Atrophy	5	Empty sella	1
Temporal atrophy	2	Astrocytoma	3
Intraventricular tumor	1	Choroid plexus cyst	1
Acoustic neurinoma	1		

Tabelle D.2*: Indikation der MRT-Testdatensätze zur Einschätzung der Darstellungsqualität verschiedener Volume-Rendering Verfahren.*

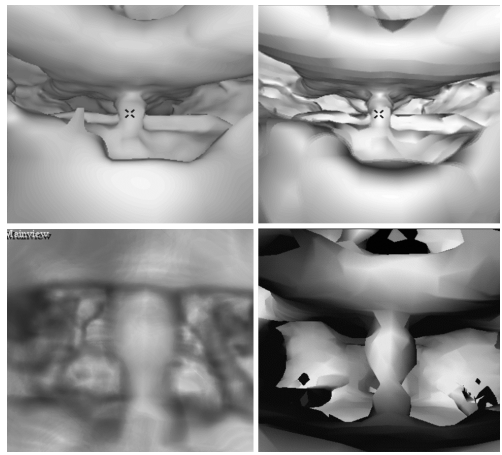

Abbildung D.12*: Vergleich virtueller Endoskopie, von links oben nach rechts unten: Navigator Surface-Rendering, Navigator Volume-Rendering, OpenGL Volumizer, Free Flight. Blickrichtung Tuberculum sellae - Fossa interpeduncularis.*

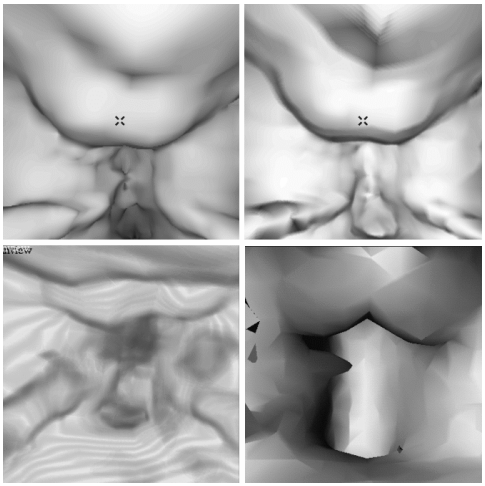

Abbildung D.13*: Vergleich virtueller Endoskopie, von links oben nach rechts unten: Navigator Surface-Rendering, Navigator Volume-Rendering, OpenGL Volumizer, Free Flight. Blickrichtung Median position of dorsum sellae - Premamillar membran.*

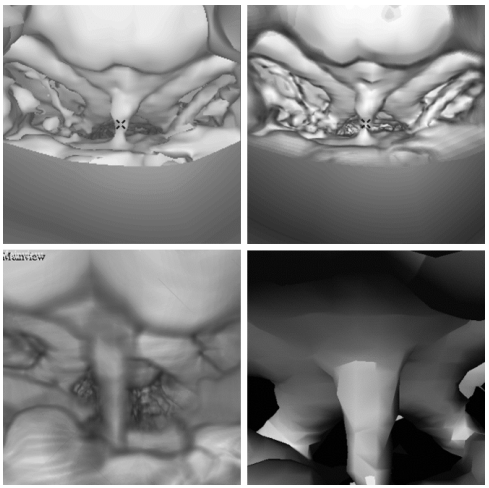

Abbildung D.14*: Vergleich virtueller Endoskopie, von links nach rechts: Navigator Surface-Rendering, Navigator Volume-Rendering, OpenGL Volumizer, Free Flight. Blickrichtung Fossa interpeduncularis – Infundibulum.*

INDEX

GLOSSAR[1]

Abdomen
Nom.: PNA der Bauch ; der Rumpfabschnitt zwischen Brustkorb u. Becken, bestehend aus Bauchwand, -höhle (Cavitas abdominalis) u. -eingeweiden .

Adnexa, Adnexe
Anhangsgebilde. Männliche u. weibliche Hoden, Nebenhoden, Samenleiter, Samenbläschen, Prostata bzw. - klinisch als Adnexe i.e.S. - Eileiter, Eierstock, Epoophoron einschließlich ihrer Bänder u. Peritoneumanteile

Agar
(malaiisch) getrockneter, in Fäden geschnittener Schleim aus roten Meeresalgen; löst sich nach Quellen unter Erwärmen in Wasser u. erstarrt nach Abkühlen transparent noch in 0,5%iger Lösung zu einem Gel

Agar-Gel
erstarrte 0,5-2%ige Agaraufkochung. Agar: (malaiisch) getrockneter, in Fäden geschnittener Schleim aus roten Meeresalgen; löst sich nach Quellen unter Erwärmen in Wasser u. erstarrt nach Abkühlen transparent noch in 0,5%iger Lsg. zu einem Gel

Angiogramm
das bei der Angiographie erstellte Röntgenkontrastbild

Angiographie
Gefäßdarstellung durch Injektion eines Röntgenkontrastmittels u. anschließende Anfertigung schneller, programmierter Aufnahmeserien (Angiogramme). Bei der MRT Angiographie entfällt die Injektion eines Kontrastmittels

Biopsiezange
Zange zur Entnahme einer Gewebeprobe von einem Lebenden.

Brain Shift
Volumenverschiebungen während einer chirurgischen Operation oder durch Druckveränderungen beim Öffnen des Schädels (Trepanation).

Cella Media
Mittlere Kammer im Ventrikel.

Computer Tomographie
Die Röntgen-Computer Tomographie oder einfach Computer Tomographie (CT) ist ein Verfahren, das eine zweidimensionale Verteilung der Röntgenabsorptionskoeffizienten in einer Schicht für ein aufgenommenes Objekt liefert (tomos ist griechisch für Scheibe). Damit ist

[1] *Quellen: Roche Lexikon Medizin, Urban & Fischer, München / DUDEN Rechtschreibung – 22. Auflage, Dudenverlag*

die Zuordnung von Orten in der aufgenommenen Schicht und Orten des realen Objektes möglich. Im Vergleich zur konventionellen Röntgenaufnahme, die ja nur eine Projektion der Röntgenabsorptionskoeffizienten eines 3D Objektes auf eine 2D Fläche liefert, erhält man signifikant mehr diagnostisch relevante Informationen. Röntgen CTs zeigen besonders gut harte Bestandteile, wie z.B. Knochen.

Endoskopie
diagnostische Betrachtung (Spiegelung) von Körperhöhlen u. Hohlorganen mit einem Endoskop; z.T. kombiniert mit operativen Eingriffen

Endoskop
röhrenförmiges, mit Lichtquelle u. optischem System ausgestattetes Instrument für die Endoskopie

Hämatom, Haematoma
Bluterguss. Traumatisch bedingte Blutansammlung außerhalb der Gefäße

intrakranial, intrakraniell
innerhalb der Schädelhöhle bzw. in die Schädelhöhle hinein

Koagulation
(auch Elektrokoagulation) sog. »Kaltkaustik« zur operativen Zerstörung umschriebener Gewebsbezirke durch Hochfrequenzstrom, wobei durch Wasserdampfbildung (Gewebsaustrocknung) u. Eiweißkoagulation in der Folge örtlicher Wärmeentwicklung ein tiefgreifender Schorfkegel entsteht.

Kraniotomie
operative Schädeleröffnung (Trepanation)

Laparotomie
op. Eröffnung der Bauchhöhle; meist von vorn durch einen typ. Bauchdeckenschnitt

Laparoskopie
Endoskopie des Bauchraumes u. seiner Organe meist knapp li.-oberhalb des Nabels unter asept. Kautelen. Zur Klärung von Lage, Größe, Farbe u. Beschaffenheit v.a. von Leber, Gallenblase, Milz, Magen, großem Netz, Beckenorganen

lateral
seitlich, von der Mitte(llinie) abgewandt

Lumen
Lichtung eines Hohlorgans

Magnet Resonanz Tomographie (MRT)
Durch die Magnetresonanz Tomographie wird die Verteilung und chemische Bindung von Wasserstoffatomen im Körper gemessen. Dazu werden elektromagnetische Felder verwendet, die die Wasserstoffprotonen aus ihrer Ruhelage auslenken. MRT Bilder zeigen einen höheren Kontrast bei der Darstellung unterschiedlicher Weichgewebe als Röntgen CTs.

OpenGL
3D Grafikbibliothek für interaktive Computergrafik, welche mittlerweile zum Industriestandard geworden ist

Open Inventor
3D Grafikbibliothek mit einer objektorientierten Schnittstelle. Basis für Open Inventor ist OpenGL

Ovarium
Nom.: *PNA* Syn.: Ovar, Oophoron der sich in der Genitalleiste aus eingewanderten Urkeimzellen etwa in der 10. Embryonal-Wo. differenzierende, circa pflaumengroße, paarige »Eierstock« bds. an der hinteren-seitl. Beckenwand (im Mesovarium)

Pixel
Kunstwort aus engl. **pic**ture **el**ement: kleinstes Element bei der gerasterten, digitalisierten Darstellung eines Bildes; Bildpunkt.

präoperativ
vor einer Operation

Septum Pellucidum
dünne „durchscheinende“ Membran als Trennwand zwischen den Vorderhörnern der Seitenventrikel

Uterus
Nom.: *PNA* Syn.: Metra Nom.: *PNA* die als Fruchthalter fungierende Gebärmutter; ein in Anteversio-Anteflexio-Stellung etwa in der Mitte des kleinen Beckens der Frau zwischen Harnblase u. Mastdarm gelegenes birnenförmiges, muskelstarkes Hohlorgan.

Trepanantion (chirurgisch)
operative Eröffnung einer Mark- oder der Schädelhöhle

Ventrikel, Hirnventrikel
Mit glasklarer Flüssigkeit gefüllter Hohlraum im Gehirn. Dient u.a. zur mechanischen Dämpfung

SYMBOLE UND ABKÜRZUNGEN

m_i	Masse des Masseknotens i
$\vec{F}_i$	Krafteinwirkung (Vektor) auf den Masseknoten i
$\vec{a}_i$	Beschleunigungsvektor des Masseknotens i
$\vec{v}_i$	Geschwindigkeitsvektor des Masseknotens i
$\vec{p}_i$	Positionsvektor des Masseknotens i
$\vec{F}_s$	Federkraftvektor der Feder *s*
c_s	Federkonstante der Feder *s*
$\vec{e}_s$	gerichtete Streckung bzw. Stauchung der Feder *s*
d_s	Viskositätskonstante der Feder *s*
$\vec{v}_s$	relative Geschwindigkeit der mit der Feder *s* verknüpften Masseknoten zueinander
U	Menge von Neuronen in einem Neuronalen Netz
u_i	Neuron mit der Nummer *i*
ext_i	externe Eingabe des Neurons u_i
w_{ij}	Verbindungsgewicht zwischen Neuron u_i und u_j
net_i	Netzeingabe für Neuron u_i
A	Menge von Aktivierungsfunktionen im Neuronalen Netz
a_i	Aktivierung des Neurons u_i
O	Menge von Ausgabefunktionen im Neuronalen Netz
o_i	Ausgabe von Neuron u_i
E	totaler Fehler der zu minimierenden Kostenfunktion im Neuronalen Netz
O	Berechnungskomplexität
t_c	Zeitkonstante für einen Zeitschritt
η	Lernrate
μ, ν	Fuzzy Menge
X,Y	Domains bzw. Grundmengen
R	eine Fuzzy Regel
$\top$	t-Norm
$\perp$	t-Conorm
$\mathbb{R}$	Menge der reellen Zahlen
${}^i\varphi$	Simplex-Winkel
${}^i\tilde{\varphi}$	Referenz-Simplex-Winkel

C++	Hochsprache zur objektorientierten Programmierung
CT	Computer Tomographie
GUI	Graphical User Interface (Grafische Benutzerschnittstelle)
DICOM	Digital Imaging and Communication in Medicine
FBAS	Farb-Bild-Austast-Synchron
FE	Finite Elemente
FEM	Finite Elemente Methode
FMRT	funktionelle Magnet Resonanz Tomographie
ISDN	Integrated Services Digital Network
LAN	Local Area Network
MRT	Magnet Resonanz Tomographie
OBB	Oriented Bounding Box
PC	Personal Computer
RAPID	Robust and Accurate Polygon Interference Detection
SGI	Silicon Graphics Inc.
SUSILAP-G	SUrgical SImulator for LAParoscopy in Gynaecology
SVHS	Super Video Home System

LITERATURVERZEICHNIS

[1] *Graphics Gems III*. In: Kirk,D (Hrsg.), Academic Press Inc., **1994**.

[2] *Compendium of Health Telematics Projects 94-98, 4th R&D Framework Programme, Telematics Applications Programme '97 Healthcare Telematics*. Compendium of Projects, European Commission Directorate General XIII, Telecommunications, Information Market and Exploitation of Research, Directorate C: Telematics Applications (networks and services), **1999**.

[3] Digital Imaging and Communications in Medicine (DICOM). http://medical.nema.org, National Electrical Manufacturers Association, **2001**.

[4] Akeley K. *RealityEngine Graphics*. Computer Graphics, Proceedings of SIGGRAPH, **1993**, 109-116.

[5] Ashcroft NW und Mermin ND. *Solid State Physics*. Saunders College Publishing, New York, **1976**.

[6] Auer DP und Auer LM. *Virtual Endoscopy: A new Tool for Teaching and Training in Neuroimaging*. Neuroradiol 4, **1998**, 3-14.

[7] Auer DP, Sendtner P, Schneider G und Auer LM. *Evaluation of Virtual Endoscopy for Application in Clinical Neurosciences*. Lecture Notes in Computer Science, MICCAI, Springer, **2001**.

[8] Auer LM. *Political and Medical Legal Issues of Medical Robotics*. Society for Minimally Invasive Therapy. 9th. Annual International Meeting, **1997**.

[9] Auer LM. *Robots for Neurosurgery*. In: Hellwig,DH, Bauer,BL (Hrsg.), Minimally Invasive Techniques for Neurosurgery - Current Status and Future Perspectives, Springer, **1998**, 243-249.

[10] Auer LM und Auer DP. *Virtual Endoscopic Views of the Brain: First Experience*. Proceed.ASNR, American Society of Head and Neck Radiology, **1997**, 303.

[11] Auer LM und Auer DP. *Virtual Endoscopy for Planning and Simulation of Minimally Invasive Neurosurgery*. Neurosurgery 43, **1998**, 529-548.

[12] Auer LM, Auer DP und Knoplioch JF. *Virtual Endoscopy for Planning and Simulation of Minimally Invasive Neurosurgery*. Lecture Notes in Computer Science 1205, Springer, **1997**, 315-318.

[13] Auer LM, Auer DP und Lipinski B. *Virtual Endoscopy for Clinical Neurosciences*. 2nd International Congress Beijing China; Syllabus Book of Abstracts, European Chinese Society for Clinical Magnetic Resonance, **1997**.

[14] Auer LM, Radetzky A, Wimmer C, Kleinszig G, Schröcker F, Auer DP, Delingette H, Davies B und Pretschner DP. *Visualization for Planning and Simulation of Minimally Invasive Neurosurgical Procedures*. Proc. MICCAI; Lecture Notes in Computer Science, Springer, **1999**, 1199-1209.

[15] Auer LM, Wimmer CA, Radetzky A und Pretschner DP. *Simulation and Planning of Minimally Invasive Neurosurgical Procedures*. ESEM 99, **1999**.

[16] Avis J, Briggs NM und Kleinermann F. *Anatomical and Physiological Models for Surgical Simulation*. Proc. of Medicine Meets Virtual Reality 7, IOS Press, **1999**, 23-29.

[17] Barker VL. *CathSim*^{TM}. In: Westwood,JD et al. (Hrsg.), Medicine Meets Virtual Reality; Studies in Health Technology and Informatics 62, IOS Press, **1999**, 36-37.

[18] Basdogan C, Ho C-H und Srinivasan MA. *Simulation of tissue cutting and bleeding for laparoscopic surgery using auxiliary surfaces*. Medicine Meets Virtual Reality, IOS Press, **1999**.

[19] Beitz W und Küttner KH. *Methode der finiten Elemente und der Randelemente*. Springer Verlag Berlin, Heidelberg, **1995**,

[20] Berkley J, Weghorst S, Gladstone H, Raugi G, Berg D und Ganter M. *Fast Finite Element Modeling for Surgical Simulation*. In: Westwood,JD et al. (Hrsg.), Medicine Meets Virtual Reality; Studies in Health Technology and Informatics 62, IOS Press, **1999**, 55-60.

[21] Bern M und Eppstein D. *Mesh Generation and Optimal Triangulation*. In: Du,D-Z, Hwang,F (Hrsg.), Computing in Euclidean Geometry, Lecture Notes Series on Computing , World Scientific, **1992**, 23-90.

[22] Bielser D, Maiwald VA und Gröller E. *Interactive Cuts through 3-Dimensional Soft Tissue*. In: Brunet et al. (Hrsg.), Computer Graphics Forum 18(3), Blackwell Publisher, **1999**, 31-38.

[23] Blinn JF. *Simulation of wrinkled surfaces*. Proceedings of the SIGGRAPH Conference, **1978**, 286-292.

[24] Bloomenthal J. *An Implicit Surface Polygonizer*. Graphic Gems IV, Academic Press, **1994**,

[25] Bockholt U, Ecke U, Müller W und Voss G. *Realtime Simulation of Tissue Deformation for the Nasal Endoscopy Simulator (NES)*. In: Westwood,JD et al. (Hrsg.), Medicine Meets Virtual Reality; Studies in Health Technology and Informatics 62, Amsterdam:IOS Press, **1999**, 74-75.

[26] Boissonnat JD und Geiger B. *Three dimensional reconstruction of complex shapes based on the delaunay triangulation*. In: Acharya,RS, Goldgof,DB (Hrsg.), Biomedical Image Processing and Biomedical Visualization, SPIE, **1993**, 964-975.

[27] Bonneau W, Read CJ und Shirali G. *Selective Visual Region of Interest To Enhance Medical Video Conferencing*. In: Kim,Y, Mun,SK (Hrsg.), Medical Imaging 1998: Image Display. Proc. of SPIE, **1998**, 2-7.

[28] Bowden R, Mitchel TA und Sahardi M. *Real-time Dynamic Deformable Meshes for Volumetric Segmentation and Visualisation*. Proc. British Machine Vision Conference, British Machine Vision Association, **1997**, 310-319.

[29] Box GEP, Hunter W.G. und Hunter J.S. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. John Wiley & Sons, **1978**.

[30] Bro-Nielson M. *Surgery Simulation Using Fast Finite Elements*. In: Höhne,KH, Kikinis,R (Hrsg.), Lecture Notes in Computer Science 1131, Springer, **1996**, 529-534.

[31] Bro-Nielson M. *Finite Element Modeling in Surgery Simulation*. Proceeding of IEEE, **1998**, 490-503.

[32] Bro-Nielson M und Cotin S. *Real-Time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condesation*. Computer Graphics Forum 15(3), Blackwell Publisher, **1996**, 57-66.

[33] Bro-Nielson M, Tasto JL, Cunningham R und Merril GL. *PreOpTM Endoscopic Simulator: A PC-Based Immersive Training System for Bronchoscopy*. In: Westwood,JD et al. (Hrsg.), Medicine Meets Virtual Reality; Studies in Health Technology and Informatics 62, IOS Press, **1999**, 76-82.

[34] Brouwer I, Ustin J, Bentley L, Sherman A, Dhruv N und Tendick F. *Measuring In Vivo Animal Soft Tissue Properties for Haptic Modeling in Surgical Simulation*. In: Westwood,JD (Hrsg.), Medicine Meets Virtual Reality, IOS Press, **2001**, 69-74.

[35] Bruyns C und Ottensmeyer M. *Measurements of Soft-Tissue Mechanical Properties to Support Development of a Physically Based Virtual Animal Model*. In: Dohi,T, Kikinis,R (Hrsg.), Lecture Notes in Computer Science, MICCAI, Springer, **2002**, 282-289.

[36] Cabral B, Cam N und Foran J. *Accelerated volume rendering and tomographic reconstruction using texture hardware*. Symposium on Volume Visualization, Proceedings of ACM SIGGRAPH, **1994**, 91-98.

[37] Çakmak H.K., Maaß H, Strauss G, Trantakis C, Nowatius E und Kühnapfel U. *Modellierung chirurgischer Simulationsszenarien für das Virtuelle Endoskopie Trainingssystem (VEST)*. CURAC 2002, electronic publication, **2003**.

[38] Cameron BM und Robb RA. *An Axial Skeleton Based Surface Deformation Algorithm for Patient Specific Anatomical Modeling*. In: Westwood,JD (Hrsg.), Medicine Meets Virtual Reality, IOS Press, **2000**, 53-58.

[39] Chen M, Silver D, Winter AS, Singh V und Cornea N. *Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation*. Eurographics, ACM Press, **2003**, 35-44.

[40] Cirak F, Scott MJ, Antonsson EK, Ortiz M und Schröder P. *Integrated Modeling, Finite-Element Analysis, and Engineering Design for Thin-Shell Structures using Subdivision*. California Institute of Technology, Pasadena, CA 91125, **1999**.

[41] Clerici T, Lange J, Zerz A, Beller S, Szinicz G, Losert UO und Siegl H. *Educational Opportunities in Minimally Invasive Surgery*. Klin Worhenschr 107(2), **1995**, 43-48.

[42] Cohen I, Cohen LD und Ayache N. *Using Deformable Surfaces to Segment 3-d Images and Infer Differential Structures*. In: Sandini,G (Hrsg.), Second European Conference on Computer Vision, Lecture Notes in Computer Science, Springer, **1992**, 648-652.

[43] Cook RL. *Shade trees*. Proceedings of the SIGGRAPH Conference, **1984**, 623-629.

[44] Cosman PH, Cregan PC, Martin CJ und Cartmill JA. *Virtual reality simulators: current status in acquisition and assessment of surgical skills*. ANZ J Surg 72(1), **2002**, 30-34.

[45] Cotin S, Delingette H und Ayache N. *Real-time elastic deformations of soft tissues for surgery simulation*. **1999**, 62-73.

[46] Cotin S, Delingette H und Ayache N. *Efficient Linear Elastic Models of Soft Tissues for real-time surgery simulation*. INRIA, **1998**.

[47] Cotin S, Delingette H und Ayache N. *Real-time elastic deformations of soft tissues for surgery simulation*. INRIA Sophia Antipolis France, **1998**.

[48] Cover SA, Ezquerra NF, O'Brien JF, Rowe R, Gadacz T und Palm E. *Interactively Deformable Models for Surgery Simulation*. IEEE Computer Graphics & Applications Nov, **1993**, 68-75.

[49] Davis C, Ladd M, Romanowski R, Wildermuth S, Knoplioch J und Debatin J. *Human Aorta: Preliminary Results with Virtual Endoscopy Based on Three-dimensional MR Imaging Data Sets*. Radiology 199, **1996**, 37-40.

[50] De S. und Bathe K.J. *The Method of Finite Spheres*. Computational Mechanics 25, **2000**, 329-345.

[51] Delingette H. *General Object Reconstruction Based on Simplex Meshes*. International Journal of Computer Vision 32(2), **1999**, 111-146.

[52] Delp SL, Loan P und Basdogan C. *Surgical Simulation: An Emerging Technology for Training in Emergency Medicine*. Teleoperators and Virtual Environments 6(4), **1997**, 147-159.

[53] Domany E, van Hemmen JL und Schulten K. *Models of Neural Networks II*. Springer, New York, **1994**.

[54] Duboi D und Prade H. *Fuzzy Sets and Systems. Theory and Applications*. Addison-Wesley, New York, **1980**.

[55] Eckel G. *OpenGL Volumizer Programmer's Guide*. Silicon Graphics, Inc., **1998**,

[56] Engel K, Kraus M und Ertl T. *High-quality pre-integrated volume rendering using hardware-accelerated pixel shading*. SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware , ACM Press, **2001**, 9-16.

[57] Fang S, Srinivasan R, Raghavan R und Richtsmeier J. *Volume Morphing and Rendering -- An Integrated Approach*. Journal of Computer Aided Geometric Design 17(1), **2000**, 59-81.

[58] Farin GE. *NURB Curves and Surfaces*. A. K. Peters Ltd, Wellesley, **1995**.

[59] Ferrant M, Warfield SK und Guttmann CRG. *3D Image Matching Using a Finite Element Based Elastic Deformation Model*. **1999**, 202-209.

[60] Filshie M. *Laparoscopic sterilization*. Seminar Laparoscopic Surgery, Academic Department of Obstetrics and Gynaecology, University Hospital, **1999**, 112-117.

[61] Foley J, van Dam A, Feiner S und Hughes J. *Computer Graphics, Principles and Practice*. Addison-Wesley, New York, **1990**.

[62] Gardener GY. *Visual simulation of clouds*. Proceedings of the SIGGRAPH Conference, **1985**, 297-303.

[63] Garland M und Heckbert PS. *Surface Simplification Using Quadric Error Metrics*. SIGGRAPH'97, **1997**.

[64] GAsser TC und Holzapfel GA. *A rate-independent elastoplastic constitutive model for (biological) fiber-reinforced composites at finite strains: Continuum basis, algorithmic formulation and finite element implementation*. Computational Mechanics 29(4-5), **2003**, 340-360.

[65] Gingins P, Beylot P, Kalra P und Thalmann NM. *Modeling using the Visible Human Dataset*. MIRALab, University of Geneva, **1996**.

[66] Gottschalk S. *RAPID Collision Library*. University of N. Carolina, Chapel Hill, USA, **1998**.

[67] Gottschalk S, Lin M und Manocha D. *OBB-Tree: A Hierarchical Structure for Rapid Interference Detection*. Computer Graphics Proceedings, Annual Conference Series ACM SIGGRAPH, **1996**, 171-180.

[68] Gouraud H. *Continuous shading of curved surfaces*. Proceedings of the IEEE Transactions on Computers, **1971**, 623-639.

[69] Green N. *Enviroment mapping and other applications of world projections*. Proceedings of the IEEE Computer Graphics and Applications, **1986**, 108-114.

[70] Gregory A, Ehmann S und Lin M. *inTouch: Interactive Multiresolution Modeling and 3D Painting*. Proc. IEEE Virtual Reality Conference, **2000**, 45-54.

[71] Grzeszczuk R, Henn C und Yagel R. *Advanced Geometric Techniques for Ray Casting Volumes*. Proc. of ACM SIGGRAPH 98, **1998**.

[72] Hanrahan P und Haeberli PE. *Direct wysiwyg painting and texturing on 3d shapes*. Computer Graphics 24(4), **1990**, 215-223.

[73] Harkki-Siren P, Sjoberg J und Kurki T. *Major complications of laparoscopy: a follow-up Finnish study*. Obstet Gynecol 94(1), **1999**, 94-98.

[74] Hayken S. *Neural Networks*. Prentice-Hall Inc, New Jersey, **1994**.

[75] Heckbert PS. *Survey of texture mapping*. Proceedings of the IEEE Computer Graphics and Applications, **1986**, 56-67.

[76] Held M. *ERIT: A Collection of Efficient and Reliable Intersection Tests*. Graphics Tools (97) 2(4), **1997**, 25-44.

[77] Hill S. *Trilinear Interpolation*. Academic Press, **1994**, 521-525.

[78] Höhne KH und Kikinis R. *Visualization in Biomedical Computing*. Lecture Notes in Computer Science 1131, Springer, Berlin, **1996**.

[79] Höhne KH, Schiemann T und Tiede U. *High Quality Rendering of Attributed Volume Data*. IEEE Visulization '98, **1998**, 255-262.

[80] Holzapfel GA, Stadler M und Schulze-Bauer ChAJ. *Soft Tissue Biomechanics: A necessity for future directions in engineering and medicine*. 13th conference of the European Society of Biomechanics, **2002**.

[81] Hopfield JJ. *Neural networks and physical systems with emergent collective computational abilities*. Proc. Nat. Acad. Sci., **1982**, 2554-2558.

[82] Hormann M, Traxler H, Ba-Ssalamah A, Mlynarik V, Shodaaj-Baghini M, Kubiena H und Trattnig S. *Correlative high-resolution MR-anatomic study of sciatic, ulnar, and proper palmar digital nerve*. Magn Reson Imaging 21(8), **2003**, 879-885.

[83] Hughes T. *The Finite Element Method*. Prentice-Hall, New Jersey, **1987**.

[84] Jambon AC, Dubois P und Karpf S. *A Low-Cost Training Simulator for Initial Formation in Gynecologic Laparoscopy*. In: Troccaz,J, Grimson,E (Hrsg.), Proc. of CVRMed-MRCAS'97, Lecture Notes in Computer Science, Springer, **1997**, 347-355.

[85] Johnson DE, Thompson II TV, Kaplan M, Nelson D und Cohen E. *Painting Textures with a Haptic Interface*. Virtual Reality '99, **1990**, 282-285.

[86] Karabassi E-A, Papaioannou G, Theoharis T und Boehm A. *Intersection test for collision detection in particle systems*. Journal of Graphics Tools 4(1), **1999**, 25-37.

[87] Kaufman A, Cohen D und Yagel R. *Volumetric Graphics*. IEEE Computer 26(7), **1993**, 51-64.

[88] Kay DS und Greenber DP. *Transparency for computer sythesized images*. Proceedings of the SIGGRAPH Conference, **1979**, 158-164.

[89] Kikinis R, Gleason PL, Moriarty TM und Moore MR. *Computer Assisted Interactive Three-Dimensional Planning for Neurosurgical Procedures*. Harvard Medical School, Boston MA, **1996**, 640-651.

[90] Kim J, De S. und Srinivasan MA. *Computationally Efficient Techniques for Real Time Surgical Simulation with Force Feedback*. 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, **2002**, 51.

[91] Kitzberger H. Darstellung und Schneiden von dünnen Membranen in der virtuellen Chirurgie. Institut für Elektro- und Biomedizinische Technik, Technische Universität Graz, **2000**.

[92] Kniss J, Kindlmann G und Hansen C. *Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets*. Visualization, IEEE Computer Society, **2001**, 255-262.

[93] Kniss J, Premoze S, Hansen C und Ebert D. *Interactive translucent volume rendering and procedural modeling*. IEEE Visualization, IEEE Computer Society, **2002**, 109-116.

[94] Kossevich AM. *The Crystal Lattice*. Wiley-VCH, Weinheim, **1999**.

[95] Krüger J und Westermann R. *Acceleration Techniques for GPU-based Volume Rendering*. IEEE Visualization , **2003**.

[96] Kruse R, Gebhardt J und Klawonn F. *Fuzzy-Systeme*. B.G. Teubner, Stuttgard, **1993**.

[97] Kuhn C. Modellbildung und Echtzeitsimulation deformierbarer Objekte zur Entwicklung einer interaktiven Trainingsumgebung fr die Minimal-Invasive Chirurgie. Wissenschaftliche Berichte FZKA 5872, Forschungszentrum Karlsruhe, **1997**.

[98] Kühnapfel U und Hübner M. *Echtzeit Volumenvisualisierung medizinischer Bilddaten für Diagnose, Navigationsunterstützung und Instrumen-*

tenplanung. 4. Workshop Digitale Bildverarbeitung in der Medizin, **1996**.

[99] Kühnapfel U, Krumm H, Kuhn C, Hübner M und Nelsius B. *Endosurgery Simulation with KISMET: A flexible tool for Surgical Instrument Design, Operation Room Planning and VR Technology based Abdominal Surgery Training*. Proceedings Modeling - Virtual Worlds - Distributed Graphics MVD '95, **1995**, 165.

[100] Kurzion Y und Yagel R. *Volume Deformation using Ray Detectors*. 6th Eurographics Workshop on Rendering, **1995**, 21-32.

[101] Lafleur B, Thalmann NM und Thalmann D. *Cloth Animation with Self-Collision Detection*. Springer, **1991**, 179-187.

[102] Lang J. Deformable Model Acquisition and Validation. The University of British Columbia, **2001**.

[103] Lang J, Pai D.K. und Woodham RJ. *Robotic Acquisition of Deformable Models*. International Conference on Robotics and Automation, IEEE, **2002**, 933-938.

[104] Leeb V. Textur Editor zur Erstellung qualitativ hochwertiger Texturen für den Einsatz in der virtuellen Chirurgie. Institut für Elektro- und Biomedizinische Technik, Technische Universität Graz, **2000**.

[105] Leeb V und Radetzky A. 3D-Puzzle; Texture-Mapping ohne Verzerrungen; Computergrafik. iX 8/2000, Heise Verlag, **2000,** 139-143.

[106] Leeb V und Radetzky A. *Interactive Texturing by Polyhedron Decomposition*. Int. Conf. on Virtual Reality, IEEE Computer Society Press, **2001**, 165-171.

[107] Levoy M und Totsuka T. *Frequency Domain Volume Rendering*. Computer Graphics, Proceedings of ACM SIGGRAPH '93, **1993**, 271-278.

[108] Li Z und Milenkovic V. *Compaction and separation algorithms for non-convex polygons and their applications*. European Journal of Operations Research 84, **1995**, 539-561.

[109] Lichtenbelt B, Csiszar P und Naqvi S. *Introduction to Volume Rendering*. Prentice Hall, New Jersey, **1998**.

[110] Lin CT und Lee CSG. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall Inc, New Jersey, **1996**.

[111] Lin MC und Manocha D. *Efficient Contact Determination Between Geometric Models*. Department of Computer Science, University of North Carolina - Capel Hill, **1994**.

[112] Litwinowicz P und Miller G. *Efficient techniques for interactive texture placement*. Proceedings of the SIGGRAPH Conference, ACM, **1994**, 119-122.

[113] Lombardo J-C, Cani M-P und Neyret F. *Real-time Collision Detection for Virtual Surgery*. **1999**.

[114] Lorensen WE und Cline HE. *Marching Cubes: A High Resolution 3-D Surface Construction Algorithm*. Computer Graphics 21(4), **1987**, 163-169.

[115] Maillot J, Yahia H und Verroust A. *Interactive texture mapping*. Proceedings of the SIGGRAPH Conference, **1993**, 27-34.

[116] Mak TCW und Zhou GD. *Crystallography in modern chemistry: a resource book of crystal structures*. John Wiley & Sons Inc., New York, **1992**.

[117] Mamdani EH und Assilian S. *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. Int.J.Man Machine Studies 7, **1975**, 1-13.

[118] Manolios P und Fanelli R. *First-Order Recurrent Neural Networks and Deterministic Finite State Automata*. Neural Computation 6(6), **1994**, 1155-1173.

[119] Mascagni MV und Sherman AS. *Numerical Methods for Neuronal Modeling*. In: Koch,C, Segev,I (Hrsg.), Methods in Neuronal Modeling, MIT Press, **1998**, 439-484.

[120] Massie TH und Salisbury JK. *The PHANTOM Haptic Interface: A Device for Probing Virtual Objects*. Dynamic Systems and Control (v.1 of 2) 55(1), **1994**, 295-299.

[121] McInerney T und Terzopoulos D. *A Dynamic Finite Element Surface Model for Segmentation and Tracking in Multidimensional Medical Images with Application to Cardiac 4D Image Analysis*. Computerized Medical Imaging and Graphics 19(1), **1995**, 69-83.

[122] McInerney T und Terzopoulos D. *A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking*. Proc. of the Fourth Int. Conf. on Computer Vision, IEEE Computer Society Press, **1993**, 518-523.

[123] McInerney T und Terzopoulos D. *Deformable models in medical image analysis*. Journal of Medical Image Analysis 1(2), **1996**, 91-108.

[124] Medsker LR und Jain LC. *Recurrent Neural Networks: Design and Applications*. CRC Press, **1999**.

[125] Meissner M, Hoffmann U und Strasser W. *Enabling classification and shading for 3d texture mapping based volume rendering using OpenGL and extensions*. IEEE Visualization, **1999**, 110-119.

[126] Miller GS und Hoffman RC. *Illumination and reflectance maps: Simulated objects in simulated and real enviroments*. Advanced Computer Graphics Seminar of the Siggraph Conference, **1984**, 1-12.

[127] Minsky ML und Papert SA. *Perceptrons*. MIT Press, Cambridge, **1988**.

[128] Möller T. *A Fast Triangle-Triangle Intersection Test*. Graphics Tools 2(2), **1997**, 25-30.

[129] Möller T und Trumbore B. *Fast, Minimum Storage Ray-Triangle Intersection*. Prosolvia Clarus AB, Chalmers University of Technology, **2000**.

[130] Moltz L und Tischer A. *Laparoscopic tubal sterilization: methodologic progress by the use of a so-called coagulotome*. **1985**, 901-905.

[131] Nauck D, Klawonn F und Kruse R. *Neuronale Netze und Fuzzy-Systeme*. Vieweg, Wiesbaden, **1996**.

[132] Newman WM und Sproull RF. *Principals of interactive Computer Graphics*. McGraw Hill, New York, **1979**.

[133] Nienhuys H-W und van der Stappen AF. *A surgery simulation supporting cuts and finite element deformation*. In: Niessen,WJ, Viergever,MA (Hrsg.), Lecture Notes in Computer Science, MICCAI, Springer, **2001**, 153-160.

[134] Nürnberger A. Recurrent Neuro-Fuzzy Systems for the Analysis of Dynamic Systems. Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, **2001**.

[135] Nürnberger A, Radetzky A und Kruse R. *Determination of Elastodynamic Model Parameters using a Recurrent Neuro-Fuzzy System*. Proc. 7th Europ. Congress on Intelligent Techniques and Software Computing (EUFIT 99), **1999**, 8.

[136] Nürnberger A, Radetzky A und Kruse R. *Using Recurrent Neuro-Fuzzy Techiques for the Identification and Simulation of Dynamic Systems*. Neurocomputing 36(1-4), **2001**, 123-147.

[137] Opalch A und Cani M-P. *Local Deformations for Animation of Implicit Surfaces*. **1997**.

[138] Ota D, Loftin B, Saito T, Lea R und Keller James. Virtual Reality in Surgical Education. Computers in Biology and Medicine 25(2)/1995, Division of Surgical Oncology, University of Missouri, **1995,**

[139] Paisley AM, Baldwin PJ und Paterson-Brown S. *Validity of surgical simulation for the assessment of operative skill*. Br J Surg 88(11), **2001**, 1525-1532.

[140] Park J, Metaxas D und Young A. *Deformable Models with Parameter Functions: Application To Heart-Wall Modeling*. Proc. of Computer Vision and Pattern Recognition, IEEE Comp. Soc. Press, **1994**, 437-442.

[141] Pedersen HK. *A Framework for interactive texturing on curved surfaces*. Proceedings of the SIGGRAPH Conference, **1996**.

[142] Pfister H, Hardenberg J, Knittel J, Lauer H und Seiler L. *The VolumePro Real-Time Ray-Casting System*. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, **1999**.

[143] Phong B-T. *Illumination for computer-generated pictures*. Communication of the ACM, **1975**.

[144] Pilloud M. FEM-basierte Oberflächenmodelle menschlicher Gesichter. Diplomarbeit an der Eidgenössische Technische Hochschule Zürich, Institut für Informationssysteme, **1995**.

[145] Pineda FJ. *Recurrent Backpropagation Networks*. In: Chauvin,Y, Rumelhart,DE (Hrsg.), **1995**, 99-135.

[146] Pollak JB. *Cascaded backpropagation on dynamical connectionist networks*. Proc. of the Ninth Conf. Cognitive Sci. Soc, **1987**, 391-404.

[147] Preparata FP und Shamos MI. *Computational Geometry*. In: Gries,D (Hrsg.), Springer, New York, **1985**.

[148] Provot X. *Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior*. Canadian Human-Computer Communications Society, **1995**, 147-154.

[149] Provot X. *Collision and self-collision handling in cloth modle dedicated to design garments*. Eurographics, **1997**, 177-190.

[150] Qin H und Terzopoulos D. *Dynamic Manipulation of Triangular B-Splines*. Solid Modeling, **1995**, 351-360.

[151] Qin H und Terzopoulos D. *D-NURBS: A Physics-Based Framework for Geometric Design*. IEEE Transactions on Visualization and Computer Graphics 2(1), **1996**, 85-96.

[152] Radetzky A. Virtual Reality; Mit Gefühl; Computer Haptik - die Zukunft der VR. iX 9/1999, Heise Verlag, **1999,** 116-121.

[153] Radetzky A. Virtual Reality; Virtueller Crash; Simulation deformierbarer Körper. iX 3/2000, Heise Verlag, **2000,** 178-183.

[154] Radetzky A, Bartsch W, Grospietsch G und Pretschner DP. *SUSILAP-G: Ein Operationssimulator zum Training minimal-invasiver Eingriffe in der Gynäkologie*. Zentralblatt für Gynäkologie 2, **1999**, 110-117.

[155] Radetzky A und Grospietsch G. *Virtuelle gynäkologische Laparaoskopie - neue Entwicklungen*. 114. Tagung der Norddeutschen Gesellschaft für Gynäkologie und Geburtshilfe, Alete Wissenschaftlicher Dienst, **1998**, 40.

[156] Radetzky A und Leeb V. *Der PC als Konkurrent von High-End-Simulationssystemen*. Wehrtechnik 34(1), **2002**, 98-99.

[157] Radetzky A und Nürnberger A. *Visualization and Simulation Techniques for Surgical Simulators Using Actual Patient's Data*. Int.J.Artificial Intelligence in Medicine 26, **2002**, 255-279.

[158] Radetzky A, Nürnberger A und Pretschner DP. *A Neuro-Fuzzy Approach for the Description and Simulation of Elastic Tissues*. In: Baumeister,M et al. (Hrsg.), Proc. of the European Workshop Multimedia Technology in Medical Training, Augustinus, **1997**, 30-40.

[159] Radetzky A, Nürnberger A und Pretschner DP. *Simulation of elastic tissues in virtual medicine using neuro-fuzzy systems*. In: Kim,Y, Mun,SK (Hrsg.), Medical Imaging 1998: Image Display. Proc. of SPIE, **1998**, 399-409.

[160] Radetzky A, Nürnberger A und Pretschner DP. *Elastodynamic Shape Modeler: A Tool for Defining the Deformation Behavior of Virtual Tissues*. RadioGraphics 20(3), **2000**, 865-881.

[161] Radetzky A, Nürnberger A, Pretschner DP und Kruse R. *The Simulation of Elastic Tissues in Virtual Laparoscopy using Neural Networks*. In: Nauck,D et al. (Hrsg.), Proc. of Neural Networks in Applications (NN'98), **1998**, 167-174.

[162] Radetzky A, Nürnberger A, Teistler M und Pretschner DP. *Elastodynamic shape modeling in virtual medicine*. Int. Conf. on Shape Modeling and Applications, IEEE Computer Society Press, **1999**, 172-178.

[163] Radetzky A und Rudolph M. *Simulating Tumour Removal in Neurosurgery*. International Journal of Medical Informatics 64, **2001**, 461-472.

[164] Radetzky A, Rudolph M, Stefanon W, Starkie S, Davies B und Auer LM. *Simulating Minimally Invasive Neurosurgical Interventions using an Active Manipulator*. In: Delp,SL et al. (Hrsg.), Lecture Notes in Computer Sciences, Proc. of MICCAI, Springer, **2000**, 578-587.

[165] Radetzky A, Schröcker F und Auer LM. *Improvement of Surgical Simulation Using Dynamic Volume Rendering*. In: Westwood,JD (Hrsg.), Studies in Health Technology and Informatics, Medicine Meets Virtual Reality 2000, IOS Press, **2000**, 272-278.

[166] Radetzky A, Wimmer C, Kleinszig G, Brukner M, Auer LM und Pretschner DP. *Interactive Deformable Volume Graphics in Surgical Simulation*. International Workshop on Volume Graphics, **1999**, 221-237.

[167] Rey D, Subsol G, Delingette H und Ayache N. *Automatic Detection and Segmentation of Envolving Precesses in 3D Medical Images: Application to Multiple Sclerosis*. INRIA Sophia Antipolis, **1988**.

[168] Rezk-Salama C, Engel K, Bauer M, Greiner G und Ertl T. *Interactive volume on standard PC graphics hardware using multi-textures and multi-stage rasterization*. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS , ACM Press, **2000**, 109-118.

[169] Rezk-Salama C, Scheuering M, Soza G und Greiner G. *Fast Volumetric Deformation on General Purpose Hardware*. Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware, **2001**.

[170] Robb RA. *Virtual (computed) Endoscopy: Development and Evaluation using the visible human datasets*. National Institutes of Health, Bethesda, Maryland, **1996**.

[171] Roberts DW, Hartov A, Kennedy FE, Miga MI und Paulsen KD. *Intraoperative Brain Shift and Deformation: A Quantitative Analysis of Cortical Displacement in 28 Cases*. Neurosurgery 43(4), **1998**, 749-760.

[172] Roche A, Malandain G, Ayache N und Prima S. *Towards a Better Comprehension of Similarity Measures Used in Medical Image Registration*.

In: Taylor,R, Colchester,A (Hrsg.), Proc. MICCAI; Lecture Notes in Computer Science, Springer, **1999**, 555-566.

[173] Rogers DF. *Procedural Elements for Computer Graphics*. McGraw-Hill, New York, **1985**.

[174] Rojas R. *Neural Networks - A Systematic Introduction*. Springer, Berlin, **1996**.

[175] Rommelfanger H. *Fuzzy Decision Support-Systeme*. Springer, Berlin, **1994**.

[176] Rosenblatt F. *A Probabilistic Model for Information Storage and Organization in the Brain*. Psychological Review 65, **1958**, 386-408.

[177] Rumelhart DE, Hinton GE und Williams RJ. *Learning internal representations by error propagation*. In: Rumelhart,DE, McClelland,JL (Hrsg.), MIT Press, **1986**,

[178] Rumelhart DE, Hinton GE und Williams RJ. *Learning representations by backpropagation errors*. Nature 323, **1986**, 533-536.

[179] Sannier G und Magnenat-Thalmann N. *A user-friendly texture-fitting methodology for virtual humans*. Proceedings of Computer Graphics International, **1997**, 167-176.

[180] Satava RM. *Accomplishments and challenges of surgical simulation*. Surg Endosc 15(3), **2001**, 232-241.

[181] Schiemann T. *Interaktive Verfahren für deformierende Eingriffe an volumenbasierten digitalen Körpermodellen*. Shaker Verlag, Aachen, **1998**.

[182] Schiemann T, Nuthmann J, Tiede U und Höhne KH. *Segmentation of the Visible Human for High Quality Volume Based Visualization*. In: Höhne,KH, Kikinis,R (Hrsg.), Lecture Notes in Computer Science, Springer, **1996**, 13-22.

[183] Schmalbrock P, Pruski J, Sun L, Rao A und Monroe J. *Phased Array RF Coils for High Resolution Imaging of the Inner Ear and the Brain Stem*. J.Comp.Assist.Tom 19, **1995**, 8-14.

[184] Schmalbrock P, Ying K und Clymer BD. *Echo-Time Reduction for Submillimeter Resolution Imaging with a Phase Encode Time Reduce Acquisition Method*. Magn.Reson.Med. 33, **1995**, 82-87.

[185] Schröcker F. Dynamisches Volume-Rendering als Methode zur Gewebeverformung in der virtuellen Chirurgie. Institut für medizinische Informatik, Technische Universität Braunschweig, **1999**.

[186] Schroeder W, Zarge J und Lorensen WE. *Decimation of Triangle Meshes*. Computer Graphics, Proceedings ACM SIGGRAPH '92, **1992**.

[187] Schulze-Bauer ChAJ, Mörth Ch und Holzapfel GA. *Passive Biaxial Mechanical Response of Aged Human Iliac Arteries*. Biomechanical Engineering 125, **2003**, 395-406.

[188] Segal M und Akeley K. *The OpenGL Graphics System: A Specification*. In: Frazier,C (Hrsg.), Silicon Graphics Inc., **1997**.

[189] Shu R, Chen Z und Kankanhalli MS. *Adaptive marching cubes*. The Visual Computer 11, **1995**, 202-217.

[190] Shushan A, Mohamed H und Magos AL. *How long does laparoscopic surgery really take? Lessons learned from 1000 operative laparoscopies*. Hum Reprod 14(1), **1999**, 39-43.

[191] Singh V, Silver D und Cornea N. *Real-time volume manipulation*. Eurographics, ACM Press, **2003**, 45-51.

[192] Song GJ und Reddy NP. *Tissue Cutting in Virtual Environments*. Interactive Technology and the New Paradigm for Healthcare, IOS Press, **1995**, 359-364.

[193] Srinivasan MA, Basdogan C und Ho C-H. *Haptic Interactions in the Real and Virtual Worlds*. In: Duke,DJ, Puerta,A (Hrsg.), Eurographics Workshop (June 2-4, 1999, Braga, Portugal), Springer-Verlag Wien, **1999**, 1-16.

[194] Staib LH und Duncan JS. *Parametrically Deformable Contour Models*. Proc. of Computer Vision and Pattern Recognition, IEEE Comp. Soc. Press, **1989**, 98-103.

[195] Stetter und Oppelt. *Magnetresonanztomographie (MRT)*. In: Hutten,H (Hrsg.), Springer-Verlag Heidelberg, New York, **1992**, 243-282.

[196] Suga M, Matsuda T, Okamoto J, Takizawa O, Oshiro O, Minato K, Tsutsumi S, Nagata I, Sakai N und Takahashi T. *Sensible human projects: haptic modeling and surgical simulation based on measurements of practical patients with MR elastography--measurement of elastic modulus*. Stud Health Technol Inform 70, **2000**, 334-340.

[197] Suzuki N, Hattori A und Kai S. *Surgical Planning System for Soft Tissues Using Virtual Reality*. In: Morgan,KS (Hrsg.), Medicine Meets Virtual Reality: Global Healthcare Grid, IOS Press, **1997**, 39-163.

[198] Székely G, Bajka M, Bechbühler C, Dual J und Enzler R. *Virtual Reality Based Surgery Simulation for Endoscopic Gynaecology*. In: Westwood,JD et al. (Hrsg.), Medicine Meets Virtual Reality; Studies in Health Technology and Informatics 62, IOS Press, **1999**, 351-357.

[199] Tait RJ, Schaefer G, Kühnapfel U und Çakmak H.K. *Interactive Spline Modelling of Human Organs for Surgical Simulators*. ESM 2003, 17th European Simulation Multiconference, **2003**, 355-359.

[200] Terzopoulos D und Fleischer K. *Deformable Models*. The Visual Computer 4, **1988**, 306-331.

[201] Terzopoulos D und Fleischer K. *Modeling inelastic deformation: Viscoelasticity, plasticity, fracture*. Computer Graphics 22(4), **1988**, 269-278.

[202] Terzopoulos D, Platt J, Barr A und Fleischer K. *Elastically deformable models*. Computer Graphics 21(4), **1987**, 205-214.

[203] Ursino M, Tasto JL, Nguyen BH, Cunningham R und Merril GL. *CathSim^{TM}: An Intravascular Catheterization Simulator on a PC*. In: Westwood,JD et al. (Hrsg.), Medicine Meets Virtual Reality; Studies in Health Technology and Informatics 62, IOS Press, **1999**, 360-366.

[204] Vasilescu M und Terzopoulos D. *Adaptive Meshes and Shells*. Proc. of Int. Conf. on Computer Vision and Pattern Recognition, **1992**, 829-832.

[205] Vining D, Shifrin R, Haponik E und Choplin R. *Virtual Bronchoscopy*. Radiology 193, **2001**, 261.

[206] Watt A und Watt M. *Advanced Animation and Rendering Techniques*. Addison-Wesley, **1992**.

[207] Welch W und Witkin A. *Constructing Intrinsic Parameters with Active Models for Invariant Surface Reconstruction*. IEEE Transactions on Pattern Analysis and Machine Intelligence, **1993**, 668-681.

[208] Wernecke J. *The Inventor Mentor*. Addison-Wesley, New York, **1994**.

[209] Wernecke J. *The Inventor Toolmaker*. Addison-Wesley, **1994**.

[210] Westermann R und Rezk-Salama C. *Real-Time Volume Deformation*. Computer Graphics Forum (EG2001 Proceedings), Eurographics Assiciation and Blackwell Publishers, **2001**, 443-452.

[211] Westover L. *Footprint evaluation for volume rendering*. Computer Graphics, Proceedings of ACM SIGGHRAPH, **1990**, 367-376.

[212] Wiet GJ, Bryan J, Dodson E, Sessanna D, Stredney D, Schmalbrock P und Welling B. *Virtual Temporal Bone Dissection Simulation*. In: Westwood,JD (Hrsg.), Medicine Meets Virtual Reality, IOS Press, **2000**, 378-384.

[213] Wilhelms J und van Gelder A. *Octree for faster isosurface generation*. ACM Trans Graph 11, **1992**, 201-227.

[214] Williams L. *Pyradmial Parametrics*. Computer Graphics 17(3), **1983**, 1-11.

[215] Williams RJ und Zipser D. *A learning algorithm for continually running fully recurrent neural networks*. Neural Computation 1, **1989**, 270-280.

[216] Wilson A, Larsen E, Manocha D und Lin MC. *Partitioning and Handling Massive Models for Interactive Collision Detection*. Computer Graphics Forum 18(3), Blackwell Publisher, **1999**, 319-330.

[217] Wilson EW. *Sterilization*. Baillieres Clin Obstet Gynaecol 10(1), **1996**, 103-119.

[218] Wilson O, Gelder AV und Wilhelms J. *Direct volume rendering via 3D textures*. University of California, Baskin Center for Computer Engineering and Information Sciences, **1994**.

[219] Wright RS, Jr. und Sweet M. *OpenGL SuperBible*. Waite Group Press, Indianapolis, **2000**.

[220] Wu W-C, Basdogan C und Srinivasan MA. *Visual, Haptic and Biomodal Perception of Size and Stiffness in Virtual Environments*. **1999**, 19-26.

[221] Wyckoff RWG. *Crystal Structures*. John Wiley & Sons, New York, London, **1963**.

[222] Yagel R, Cornhill JF, Shekhar R und Fayyad E. *Octree-Based Decimation of Marching Cubes Surfaces*. Proc. Visualization '96, **1996**, 65-72.

[223] Yagel R, Wiet G, Stredney D und Schmalbrock P. *A Volumetric Approach to Virtual Simulation of Functional Endoscopic Sinus Surgery*. Medicine meets Virtual Reality 5, IOS Press, **1997**.

[224] Yeh TP und Vance JM. *Combining sensitive methods, finite element analysis, and free-form deformation to facilitate structural shape design in a virtual environment*. Proc. 23rd ASME Design Automation Conference, **1997**.

[225] Yoshida H, Tsutsumi S, Mizunuma M und Yanai A. *A surgical simulation system of skin sutures using a three-dimensional finite element method*. Clin Biomech 16(7), **2001**, 621-626.

[226] Zadeh LA. *Fuzzy Sets*. Information and Control 8, **1965**, 338-353.

[227] Zienkiewicz OC. *Methode der finiten Elemente*. Carl Hanser Verlag München, **1984**.