Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft

Wissenschaftliche Berichte
FZKA 7190

Echtzeitfähige Gewebemodellierung für chirurgische VR-Trainingssimulatoren

Abschlussbericht zum Projekt „Gewebemodellierung“ der Landesinitiative Baden-Württemberg 01.10.2003 – 31.03.2006

H. Çakmak, U. Kühnapfel, G. Bretthauer

Institut für Angewandte Informatik

Juni 2006

Forschungszentrum Karlsruhe

in der Helmholtz-Gemeinschaft

Wissenschaftliche Berichte

FZKA 7190

Echtzeitfähige Gewebemodellierung für chirurgische VR-Trainingssimulatoren

Abschlussbericht zum Projekt „Gewebemodellierung“ der

Landesinitiative Baden-Württemberg

01.10.2003 – 31.03.2006

H. Çakmak, U. Kühnapfel, G. Bretthauer

Institut für Angewandte Informatik

Forschungszentrum Karlsruhe GmbH, Karlsruhe
2006

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

Mitglied der Hermann von Helmholtz-Gemeinschaft
Deutscher Forschungszentren (HGF)

ISSN 0947-8620

urn:nbn:de:0005-071908

Echtzeitfähige Gewebemodellierung für chirurgische VR-Trainingssimulatoren

Zusammenfassung

Chirurgische Eingriffe am Menschen müssen effizient, sicher und schonend durchgeführt werden. Oft fehlt es jedoch an Möglichkeiten, operative Eingriffe unter realen Bedingungen zu erlernen und zu trainieren. Chirurgen werden deshalb neben den klassischen Trainingsmethoden – bei der die realistischen Gegebenheiten nur begrenzt nachgebildet werden können – verstärkt mit VR-basierten Trainingssystemen aus- und weitergebildet. Diese ermöglichen eine beliebig oft wiederholbare, objektiv bewertbare Trainingmöglichkeit in einem nahezu realistisch nachgebildeten virtuellen Operationsbereich, bestehend aus Organmodellen und chirurgischen Instrumenten mit einer breiten Palette an Interaktionsmöglichkeiten. Der Grad der Realitätsnähe eines VR-Trainingssystems wird zum großen Teil durch die Güte der Gewebemodellierung bestimmt.

Gewebemodellierung fassen wir als einen Oberbegriff für alle Prozeduren von der geometrischen Modellerstellung anhand radiologischer Bilddaten bis hin zur physikalisch-basierten Gewebsdeformationssimulation sowie die realistische Visualisierung und die Möglichkeiten der Interaktion mit dem virtuellen Gewebe auf.

In diesem Projekt werden drei Fragestellungen aus dem Bereich der Gewebemodellierung näher untersucht: die Modellierung, die Visualisierung und die Deformationssimulation. Dabei werden konkrete Lösungen für das am Forschungszentrum Karlsruhe entwickelte VR-Trainingssystem für die minimal invasive Chirurgie „VS*One*“ geliefert.

Für die effiziente Modellierung von deformierbaren Organmodellen wurde die Methode der impliziten Oberflächen implementiert. Diese eignen sich optimal für den unerfahrenen Modellierer, da keinerlei geometrische Vorkenntnisse notwendig sind. Auch lassen sich mit dieser Methode bereits vorhandene beliebige 3D-Organmodelle nachträglich modifizieren. Für die Integration der 3D-Modelle in den VR-Simulator wurden entsprechende Softwaremodule für das Einlesen, die Darstellung und die Simulation der Deformation erstellt. Die Unterstützung deformierbarer Objekte mit beliebiger Geometrie ermöglicht den Modellaustausch in einem weltweiten Computernetzwerk.

Im Bereich der realitätsnahen Visualisierung von Gewebe wurden mehrere Methoden zur Texturkomposition aus Basis-, Struktur- und Beleuchtungstexturen untersucht. Die entwickelten Algorithmen nutzen die Fähigkeiten moderner Grafikkarten aus, unterstützen aber auch ältere Grafikkarten mit einfacher Textureinheit.

Eine weitere Fragestellung in diesem Projekt ist die Optimierung der Deformationssimulation mit der *Fast-Finite-Element*-Methode (FFE). Ausgehend von einer rudimentären Implementierung wird ein objekt-orientierter Ansatz mit adaptiver Berechnung der Steifigkeitsmatrizen vorgestellt. Dadurch ist eine hybride Modellhaltung und Deformationssimulation möglich. Untersucht wurde auch die Tauglichkeit eines zweistufigen FFE-Verfahrens in einem VR basierten chirurgischen Trainingssimulator.

Real-time Tissue Modelling for Virtual Reality based Surgical Simulators

Abstract

Surgical interventions on humans need to be performed with a high grade of efficiency, security and care. A major problem is the absence of possibilities for learning and training of surgical operations under realistic conditions. Virtual Reality (VR) based training systems seem to be a good alternative to classical training methods, which allow unlimited repeatable training with objective assessment. VR based training systems mostly offer a realistic environment with haptic input devices and virtual surgical interactions with deformable or static organ models. The level of realism of a VR based surgical training system depends mostly on the quality of the tissue model.

Tissue modelling covers the topics of geometrical model creation using CT and MR data, the physical simulation of object deformation, the realistic visualisation and the surgical model interactions.

In this project, we developed new methods for modelling, visualisation and the simulation of tissue deformation. As a development environment we used "VS*One*", a VR based training system for minimally invasive surgery developed at Forschungszentrum Karlsruhe.

We developed a software module for modelling deformable objects with implicit surfaces, which are most suitable for inexperienced modellers without any geometrical knowledge. This module supports also the geometrical modification of existing 3d models. Special model IO-procedures enable to convert static models into a deformable model representation and to import them into the simulation kernel of "VS*One*". The support of deformable models with arbitrary triangle meshes within our simulation software allows the model exchange with other developers and helps to build up a world-wide network of model databases.

Further development was done in the area of realistic visualisation of virtual tissue for real-time applications. A real-time texture composition algorithm enables to map basic, structure and lighting textures onto the 3d objects utilizing modern graphics card's multi-texturing option, but also supports older hardware with single texture-units.

Real-time simulation of soft tissue deformation is mostly done using mass-spring models. As an alternative we use the *Fast-Finite-Element*-Method (FFE). We extended the existing implementation with several new features like object-oriented and adaptive calculation of the stiffness matrix and the possibility for hybrid deformation simulation. A two-step FFE-simulation was examined for suitability in a VR based surgical training system.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1. Einleitung

Die realitätsnahe Modellierung und Simulation von biologischem Gewebe ist in den letzten Jahren aufgrund neuartiger Entwicklungen im Bereich der computerunterstützten Operationsplanungs- und Trainingssystemen verstärkt in den Vordergrund der aktuellen Forschung gerückt. Eine komparative Studie zum Stand der Technik ist in [SJ03] zu finden.
Nach unserem Verständnis ist „Gewebemodellierung“ nicht nur auf die computertechnische Nachahmung des Verhaltens von biologischem Weichgewebe beschränkt, sondern umfasst die Themengebiete der effizienten dreidimensionalen Gewebsmodellierung, der realistischen Echtzeit-Visualisierung und der echtzeitfähigen, physikalischen Deformations- und Interaktionssimulation von biologischem Weichgewebe während chirurgischer Interaktionen in einem Trainingssimulator. Ziel dieses Projekts ist die wissenschaftliche Untersuchung ausgewählter Problemstellungen sowie die Erarbeitung praktischer Lösungen für das am Forschungszentrum Karlsruhe entwickelte VR-Trainingssystem für die Chirurgie „VS*One“*, die jedoch ohne Einschränkung auf andere Systeme übertragbar sind.
Die Arbeitspakete wurden wie folgt festgelegt:

- *Effiziente Modellierung von deformierbaren Organmodellen*: In diesem Arbeitspaket sollen neue Verfahren entwickelt und bekannte Methoden kombiniert werden, um die Erstellung von Simulationsszenarien mit deformierbaren Objekten effizient und einfach zu gestalten. Basierend auf der Modellierung mit impliziten Oberflächen werden neue Objekttypen und deren Integration in den VR-Chirurgietrainer vorgestellt.
- *Visualisierungstechniken zur Steigerung der Darstellungsechtheit*: Die Möglichkeiten moderner PC-Grafikkarten zur Darstellung photorealistischer Simulationsbilder in Echtzeit sollen ausgeschöpft werden, wobei auch Software-Lösungen für ältere Graphik-Hardware zur Verfügung gestellt werden sollen. Hierzu sollen Multi-Texturierungsmethoden implementiert und die Resultate diskutiert werden.
- *Optimierung der Deformationssimulation*: Vorhandene Simulationstechniken für die Deformation von Weichgewebe sollen optimiert werden. Insbesondere soll die *Fast-Finite-Element*-Methode optimiert und deren Tauglichkeit in einem VR-Chirurgie-Simulator untersucht werden. Hierzu sind die neuen Simulationstechniken in das Endoskopie-Trainingssystem VS*One* zu integrieren.

Im folgenden Kapitel wird zunächst ein kurzer Überblick über die eigenen Vorarbeiten im Bereich der chirurgischen VR-Simulatoren gegeben. Insbesondere wird hier der am Forschungszentrum Karlsruhe entwickelte Laparoskopietrainer „VS*One“* und die entsprechenden Softwarepakete zur Gewebemodellierung vorgestellt. In Kapitel 3 werden die angewandten Methoden für die oben definierten Arbeitspakete vorgestellt, in Kapitel 4 werden Ergebnisse vorgestellt und diskutiert. In Kapitel 5 wird eine Zusammenfassung und ein Ausblick gegeben.

2. Eigene Vorarbeiten

Im Rahmen eines Technologietransfer-Projekts zwischen dem Institut für Angewandte Informatik und der Fa. „*Select IT VEST Systems AG*", Bremen wurde „VS*One*“, ein VEST-

System (**V**irtual **E**ndoscopic **S**urgery **T**raining) für die minimal invasive Chirurgie (siehe Abbildung 2-1) entwickelt [CMK05].

Abb. 2-1 Einsatz des VEST-Trainingssystems „VS*One*“ in einem Fortbildungskurs für Chirurgen an der Uniklinik Kiel, Prof. L. Mettler

Das System ermöglicht das Erlernen und Trainieren chirurgischer Interaktionen in Echtzeit, wobei dem Benutzer visuelle und haptische Informationen für die Orientierung und Interaktion innerhalb eines virtuellen Simulationsszenarios zur Verfügung stehen. Eine Bedienbox mit integrierten haptischen Eingabegeräten dient zur Positionserfassung für die Modellinteraktion und zur haptischen Ausgabe der aus der Simulation resultierenden Kräfte. Die folgende Abbildung zeigt schematisch den Datenfluss, sowie den Zusammenhang der einzelnen Systemkomponenten.

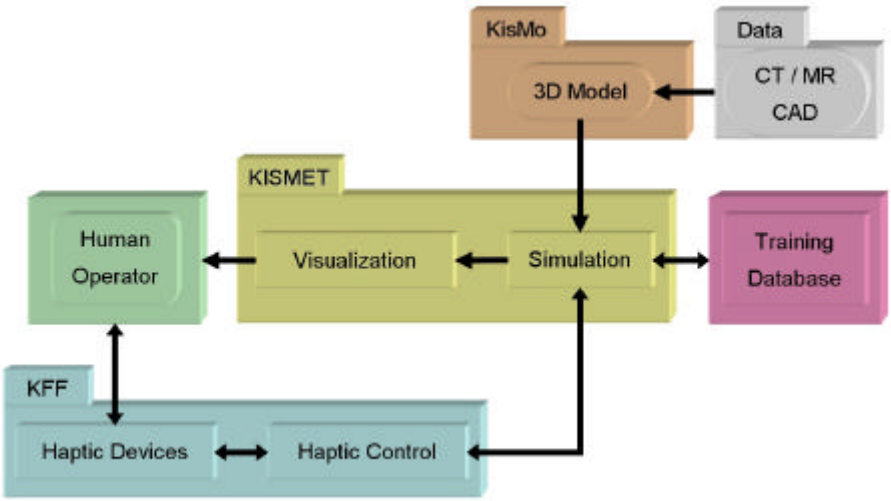

Abb. 2-2 Datenfluss und Systemdesign

Den Kern des Chirurgiesimulators bildet die im Forschungszentrum Karlsruhe entwickelte Echtzeit-Simulationssoftware KISMET, die *OpenGL* als Graphikschnittstelle verwendet und für die Betriebssysteme UNIX und Windows verfügbar ist. Die Software enthält eine

umfangreiche Bibliothek an Algorithmen für die Strukturmechanik, Robotik und Visualisierung, wobei die enge Kopplung zwischen Strukturdynamik-Berechnung und Visualisierung eine hohe Effizienz (Echtzeitverhalten) ermöglicht. Durch die langjährige, kontinuierliche Forschung und Anwendung in verschiedenen Bereichen hat sich KISMET zu einem flexiblen, stabilen und universell einsetzbaren Simulationssystem entwickelt.
Die Modellerstellung erfolgt mit den Software-Werkzeugen KisMo und VESUV. KisMo ist als Autorenwerkzeug zur effizienten Erstellung von Simulationsszenarien konzipiert. Wichtigste Software-Komponente ist die Freiformflächenmodellierung anhand radiologischer Bilddaten, wodurch eine optimale Anpassung an Form und Lage der Zielobjekte erfolgt. Neben der rein geometrischen Modellierung werden auch biomechanische Struktureigenschaften für die Gewebesimulation definiert. VESUV ist eine Software für die Visualisierung, interaktive Segmentation und semi-automatische Polygonnetzerstellung aus computertomographischen Bilddaten (CT, MR). Alle Softwarepakete sind optimal aufeinander abgestimmt, so dass ausgehend von radiologischen Bilddaten patientenspezifische Modelle erstellt und mit dem Trainingssystem evaluiert werden können [CMS02].
Zudem werden alle benutzerspezifische Simulationsereignisse (Interaktionen, Fehler) in einer Trainingsdatenbank protokolliert und für eine weitere Evaluierung zur Verfügung gestellt. Eine ausführliche Beschreibung des VS*One* und dessen Einsatz zum Training der Cholezystektomie und der gynäkologischen Myomektomie findet sich in [CMK05].

3. Angewandte Methoden

Im folgenden werden die neu entwickelten Methoden für die Themenbereiche Modellierung, Visualisierung und die physikalische Deformationssimulation vorgestellt.

3.1. Effiziente Modellierung deformierbarer Objekte

3.1.1. Problemstellung und Motivation

Die geometrische Modellierung deformierbarer Objekte für chirurgische Simulationsszenarien orientiert sich an den Vorgaben der Simulationssoftware KISMET. Historisch gewachsen sind bestimmte Datenstrukturen zur Speicherung deformierbarer Objekte. So auch die standardmäßig verwendete ELA_NURBS-Modellspezifikation, die ursprünglich für die direkte Übernahme von NURBS-Geometrien (**N**on-**U**niform **R**ational **B**-**S**plines) aus externen 3D-Softwarepaketen konzipiert war. Durch eine Erweiterung mit zusätzlichen Elastodynamik-Parametern sollten starre NURBS-Modelle als deformierbare Objekte im Simulationssystem eingesetzt werden können. Da die Rechengeschwindigkeit für die echtzeitfähige Darstellung als Freiformfläche jedoch unzureichend war, wurde stets eine polynomiale Approximation der NURBS-Modelle durchgeführt, bei der die Anordnung der Kontrollpunkte zwingend in einer Matrix vorgeschrieben ist. Zur Erhöhung der Rechengeschwindigkeit wurde zudem oftmals auf die Approximation des Oberflächennetzes aus den Kontrollpunkten verzichtet, so dass die Kontrollpunkte identisch mit den Knoten des Feder-Masse-Netzes verwendet wurden. KISMET unterstützt drei Grundprimitive für deformierbare Objekte (Ebene, Rohr, Kugel), die an eine Matrix-Struktur gebunden sind und eine hieraus resultierende Anzahl und Topologie von Federelementen enthalten müssen. Eine weitere Modellspezifikation ist ELA_POLY, eine reduzierte Variante der ELA_NURBS Spezifikation, bei der die Freiformflächeninformationen durch Polygonrepräsentation ersetzt

ist, die sich jedoch weiterhin an einer vorgeschriebenen Knoten- und Federelemente-Topologie orientiert. Die Nachteile der bisherigen Modell-Definitionen in KISMET sind:

- Keine Flexibilität in der Objektdefinition, da nur drei Grundobjektformen existieren und komplexe Objekte aus diesen Grundprimitiven zusammengesetzt werden müssen.
- Fest vorgeschriebene Topologie der Knoten und Federelemente zur Definition eines Feder- Masse-Netzes.
- Keine Unterstützung beliebiger Dreiecksnetze, dadurch u.a. kein Modellaustausch mit anderen Entwicklern möglich.

Die Modellierung deformierbarer Objekte mit KisMo ist an die Grundobjekte (Ebene, Rohr, Kugel) angepasst, dabei wird nach Angabe der Dimension, Typ und Auflösung das entsprechende Grundobjekt erstellt. Der Benutzer verformt interaktiv das Modell, bis es eine Ähnlichkeit mit dem Zielobjekt erreicht. Eine weitere Methode ist die direkte Konstruktion der Objekte mittels radiologischer Datensätze. Dabei werden tomographische Schichtbilder visualisiert, der Umriss des zu modellierenden Organs in einigen Volumenschichten als Freiformkurven nachgebildet, fehlende Kurven interpoliert und zu einer Freiformfläche zusammengesetzt. Vorteil dieser Methode ist die direkte Orientierung an Radiologiedaten und somit korrekte Lage und Größe der Organmodelle. Nachteil ist die Beibehaltung der Matrixstruktur während der Objektkonstruktion, wodurch stets ein Kompromiss aus Modellgenauigkeit und Modellgröße berücksichtigt werden muss. Zudem ist die Modellierung mit Freiformflächen für Mediziner als sog. „Modellautoren“ trotz benutzerfreundlicher Programmführung meist zu komplex. Unser Ziel ist daher die Einführung einfacherer Modelliermethoden für deformierbare Objekte.

3.1.2. Modellierung mit impliziten Oberflächen

Implizite Oberflächen werden durch mathematische Gleichungen definiert und sind aufgrund ihrer einfachen Handhabbarkeit ein ideales Modellierwerkzeug. Die bekannteste implizite Oberfläche ist die Definition einer Kugeloberfläche, die durch alle Punkte bestimmt ist, deren Abstand zum Kugelmittelpunkt gleich dem Kugelradius sind.
Für die Modellierung mit impliziten Oberflächen verwenden wir eine optimierte Version der von Wyvill vorgestellten *SoftObjects* [WMW86]. Der Benutzer erstellt und positioniert Partikel innerhalb eines Volumens, daraus wird eine abstandsabhängige Dichteverteilung *D* für jedes Voxel berechnet und die Isofläche erstellt. Der Wert eines Voxels $\underline{x}$ innerhalb des Volumens berechnet sich aus der Summe der Dichteverteilungen für alle Partikel, wobei r_i den Abstand zwischen Partikelmittelpunkt und $\underline{x}$ darstellt und R_i den Wirkungsbereich eines Partikels wiedergibt.

$$D(r_i, R_i) = \sum_i \left(1 - \frac{4r_i^6}{9R_i^6} + \frac{17r_i^4}{9R_i^4} - \frac{22r_i^2}{9R_i^2} \right) \tag{3.1}$$

Die realisierte Modellierungspipeline beinhaltet Module zur Verwaltung der Partikel, Berechnung der Dichteverteilung, Volumenvisualisierung und 3D-Mesh-Generierung. Daneben wurden zusätzliche Exportfunktionalitäten zur Weiterverarbeitung der Modelle implementiert. Die folgende Abbildung gibt den Datenfluss bei der Modellierung mit impliziten Oberflächen wieder.

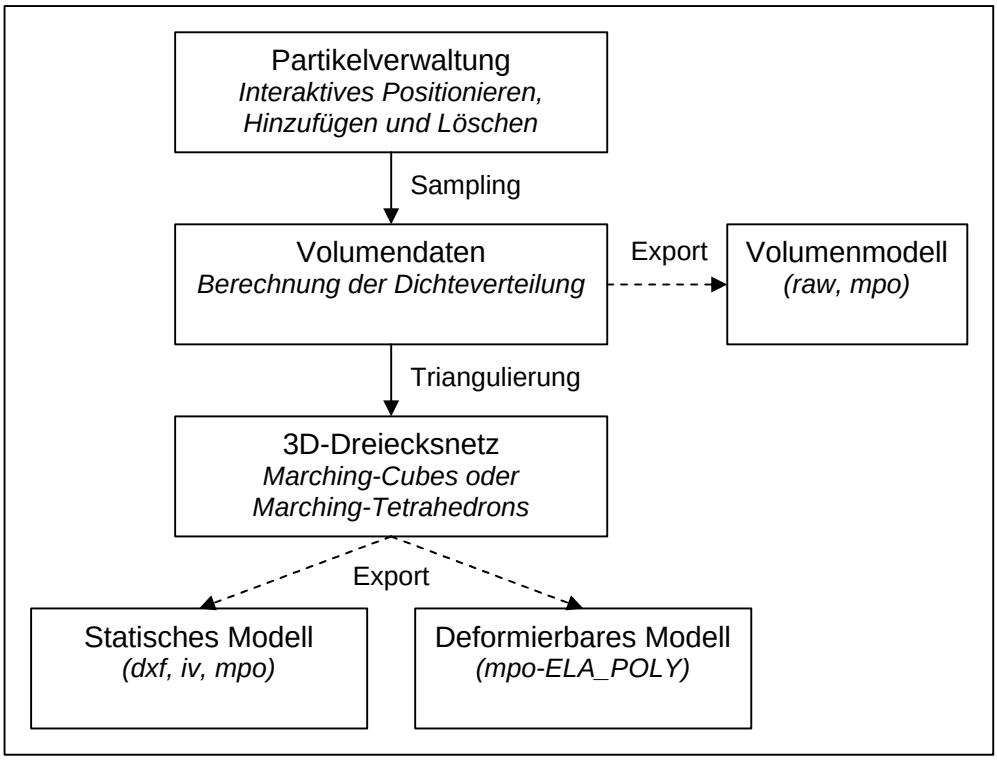

Abb. 3-1 Datenfluss für die Modellierung mit impliziten Oberflächen

Die Modellierung erfolgt durch Positionierung von Partikel innerhalb eines Volumens beliebiger Größe, wobei neue Partikel interaktiv erzeugt, gelöscht oder in ihren Eigenschaften verändert werden. Die Abtastung der diskreten Voxelwerte erfolgt in Abhängigkeit von den Partikeleigenschaften und einem Iso-wert kontinuierlich während der Modellierphase oder auf Anforderung des Benutzers. Die erzeugte Dichteverteilung kann als Volumenmodell oder nach einer Triangulation als Dreiecksnetz exportiert werden. Dabei steht eine Auswahl gängiger 3D-Dateiformate für reine Geometriedaten zur Verfügung, für die Verwendung als deformierbare Objektmodelle wurde eine Erweiterung der KISMET-Datentypen inklusive einer Importfunktionalität implementiert.

3.1.3. Der neue Objekttyp ELA_EXPL für deformierbare Objekte

Die Verwendung der impliziten Oberflächenmodelle als deformierbare Objekte in *KISMET* ist nicht ohne weiteres möglich. Hierzu wurde das ELA_POLY Objekttyp derart erweitert, dass beliebige Dreiecksnetze eingelesen, visualisiert und elastodynamisch verformt werden können. Der neue Datentyp für die explizite Objektdefinition ELA_EXPL als Untertyp des ELA_POLY Datentyps erwartet die explizite Angabe folgender Informationen für ein deformierbares Objekt:

- Anzahl und Positionskoordinaten der Modellknoten
- Physikalische Deformationsparameter (Masse, Federkonstante, Dämpfung) Texturkoordinaten, Normalenvektoren und Beschränkungen der Knotenbewegungen
- Zuordnung von Oberflächenknoten an Zentralknoten für die Simulation globaler Objektbewegungen
- Anzahl und Topologie der Federelemente
- Materialindex.

Die Einführung des neuen Objekttyps ELA_EXPL ist für die Simulationsplattform KISMET von enormer Bedeutung. Es ermöglicht die Übernahme von statischen Modellen aus externen Modelliersoftwarepaketen und deren direkte Transformation in deformierbare Modelle. Zudem ist mit diesem neuen Objekttyp ein Austausch verformbarer Modelle mit anderen Entwicklergruppen möglich, was für den Aufbau einer globalen Modelldatenbank in einer GRID-Umgebung für chirurgische Simulatoren von großer Bedeutung ist.

3.1.4. Optimierung von Dreiecksnetzen

Während der Modellierphase mit impliziten Oberflächen wird die Objektgeometrie kontinuierlich für eine intuitive Art der Modellierung visualisiert. Das wird dadurch gewährleistet, dass die Abtastrate bei der Dichteverteilungsberechnung gering gehalten wird und die Dichteverteilung nur innerhalb der Wirkungsbereiche der Partikel berechnet wird. Eine vollständige Abtastung des Volumendatensatzes würde eine echtzeitfähige Modellerstellung gerade bei hoher Volumenauflösung ($>64^3$) und einer Vielzahl von Partikeln unmöglich machen, da hierfür der Beitrag aller Partikel auf alle Voxel berechnet werden müsste. Durch die Einschränkung auf einen bestimmten Wirkungsbereich werden Volumenabtastwerte selektiv nur in Abhängigkeit von der Partikelposition berechnet.
Eine weitere Optimierung betrifft das generierte Dreiecksnetz für die Weiterverwendung als deformierbares Modell. Die Deformationssimulation benötigt im Idealfall ein Dreiecksnetz mit nahezu gleichgroßen Dreiecken, um numerische Instabilitäten während der Deformationssimulation zu vermeiden. Während der Triangulierung mit dem *Marching-Cubes*-Algorithmus [LC87] werden in unserer Implementierung kleine Dreiecke und deren benachbarte degenerierte Dreiecke eliminiert. Ein Schwellwert wird verwendet, um die Lage des Schnittpunktes der Isofläche mit den Kanten eines Voxels zu klassifizieren. Ist der Abstand eines Schnittpunkts zu einer Voxelecke kleiner als ein benutzerdefinierter Schwellwert e, so kollabiert der Schnittpunkt zu diesem Eckpunkt. Bei der Triangulierung werden anschließend alle Dreiecke mit Nullfläche eliminiert, diese sind gekennzeichnet durch mindestens zwei identische Eckpunkte. Die folgende Abbildung zeigt ein Beispiel für die Optimierung des Dreiecksnetzes während der Triangulation. Auf dem linken Bild sieht man einen Teil des Originaldreiecksnetzes, in der kleine Dreiecke mit jeweils drei benachbarten degenerierten Dreiecken gut zu erkennen sind. Mit der Schwellwertmethode können diese effizient eliminiert und ein regelmäßigeres Dreiecksnetz erstellt werden (Abbildung 3-2 *rechts*).

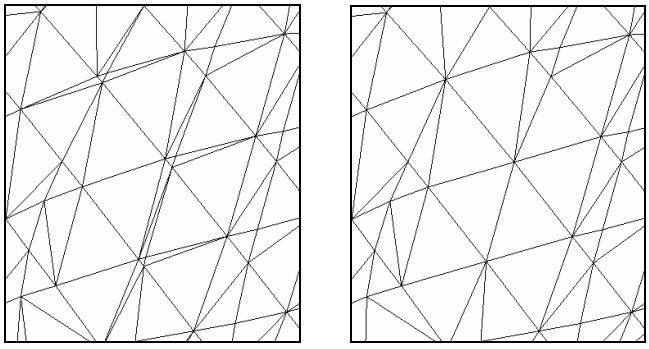

Abb. 3-2 Optimierung des Dreiecksnetzes während der Triangulierung

Die Reduktionsrate definieren wir als die Anzahl eliminierter Punkte und Dreiecke im Verhältnis zur Gesamtzahl im nicht optimierten Zustand, sie hängt somit direkt von dem gewählten Schwellwert ab. Wird der Schwellwert zu klein gewählt, so ist die Optimierung

nicht ausreichend, wird sie zu groß gewählt, weist das Dreiecksnetz Stufen (*Aliasing*-Effekte) auf, bedingt durch die Abbildung aller Schnittpunkte (der Isofläche mit den Voxeln) auf die Voxelecken. Die folgenden Diagramme zeigen die mittlere Reduktionsrate für Punkte und Dreiecke eines 3D Modells generiert aus fünf Partikeln für unterschiedliche Auflösungen des Abtastvolumens (8^3-128^3) in Abhängigkeit von dem Schwellwert e.

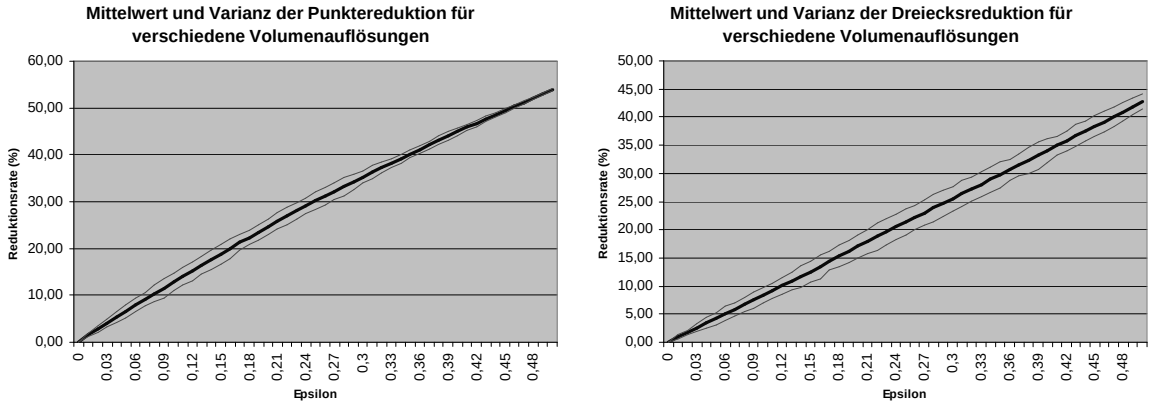

Abb. 3-3 Reduktionsrate der Punkte und Dreiecke für verschiedene Auflösungen des Volumendatensatzes und unterschiedliche Schwellwerte e

Es ist ein nahezu linearer Zusammenhang zwischen der Wahl des Schwellwertes und der Reduktionsrate für Punkte und Dreiecke des Modells zu erkennen. Ein zu großer Schwellwert e bewirkt eine Vergröberung des Modells durch die Abbildung der Dreieckspunkte auf die Voxelecken wodurch sog. „Stufen" entstehen, ein zu kleiner Wert belässt zu viele degenerierte und kleine Dreiecke im Modell. Als guten Kompromiss hat sich der Schwellwert $e = 0.15$ bewährt.

3.1.5. Modellmodifikation mit impliziten Oberflächen

Ein weiteres Anwendungsgebiet für implizite Oberflächen ist die Modifikation bestehender 3D-Modelle. Hierzu gibt es eine ähnliche Methode in [ABG04], wir verfolgen im Gegensatz dazu jedoch das Ziel deformierbare Modelle nach der Modellmodifikation zu erzeugen. Abbildung 3-4 verdeutlicht unsere Vorgehensweise.
Zunächst wird ausgehend von einem beliebigen 3D-Geometriemodell eine Voxelisierung durchgeführt. Dabei wird ein Dreiecksnetz in ein Volumenmodell umgewandelt, wobei die Abtastrate und somit die Größe des Volumenmodells frei wählbar sind. Ergebnis ist ein Volumendatensatz, in der alle Voxel mit einem binären Wert als „*zum Objekt gehörig*" oder „*unbesetzt*" markiert sind. Hierfür benutzen wir die Methode nach Karabassi [KPT99]. Es werden drei orthogonale z-Buffer Paare um das entsprechende Objekt mit den entsprechenden Blickrichtungen gelegt, die Tiefeninformation im z-Buffer erstellt und ausgewertet. Ein Voxel gilt als zum Objekt gehörig, falls es zwischen den im z-Buffer gespeicherten Grenzwerten liegt.
Die Modifikation des Volumenmodells erfolgt mit der Methode der impliziten Oberflächen. Eine beliebige Anzahl an Partikel wird im Volumendatensatz platziert und mit den zuvor erzeugten Voxelwerten unter Verwendung von CSG-Operationen kombiniert. Dies erfolgt durch Addition bzw. Subtraktion der Dichtewerte zu bzw. von den Voxelwerten. Der *Marching-Cubes*-Algorithmus erzeugt aus dem modifizierten Volumenmodell ein Dreiecks-

netz, das als statisches Modell in gängigen 3D-Dateiformaten oder als deformierbares Modell im KISMET ELA_EXPL-Format exportiert wird.

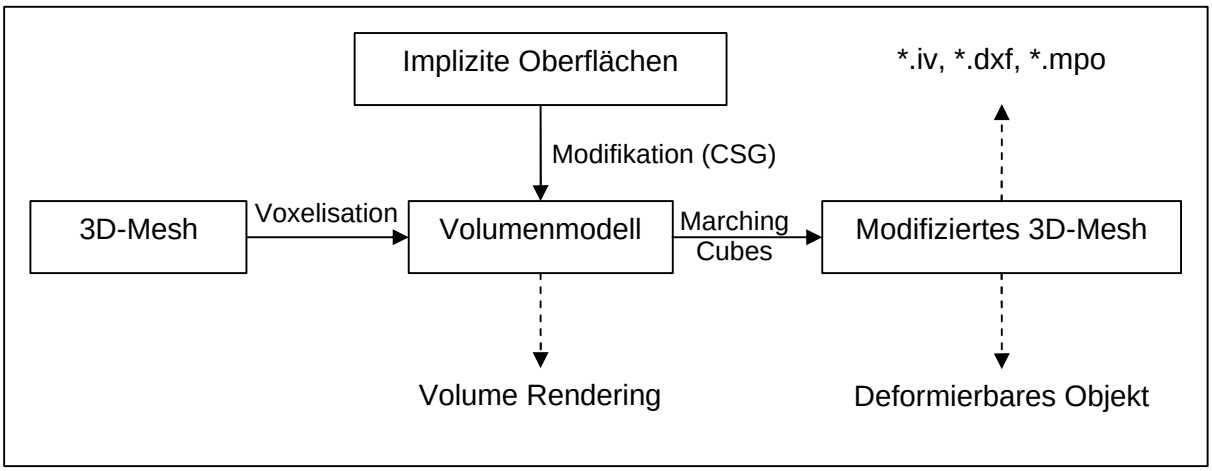

Abb. 3-4 Datenfluss für die Modellmodifikation mit impliziten Oberflächen

3.2. Erweiterung der Darstellungsechtheit für chirurgische Simulatoren

3.2.1. Problemstellung und Motivation

Die Simulationsplattform KISMET beinhaltet eine umfangreiche Bibliothek an graphischen Darstellungsarten für polygonale Objekte. In einer Datenbank werden Materialparameter und Texturen verwaltet, die über eine Indexadressierung einzelnen Objekten zugewiesen werden.
Die bisher unterstützte Abbildung einer einzigen Textur auf eine Objektoberfläche ist jedoch nicht Stand der Technik. Aktuelle PC-Grafikkarten im mittleren Preissegment besitzen bereits mehrere Textureinheiten, die eine hardwaremäßig beschleunigte Mehrfachtexturierung (*Multi-Texturing*) erlauben. Dies ermöglicht die Abbildung und Darstellung mehrerer Texturen auf einem Objekt in einem Zeichendurchlauf (*render pass*). Ältere Grafikkarten mit einer Textureinheit müssen je nach Anzahl der abzubildenden Texturen das gesamte Modell mehrfach darstellen. Die Technik der Mehrfachtexturierung ist im Bereich der Computerspiele und 3D-Rendering-Software etabliert und dient zur Steigerung der Echtheit der computergraphischen Darstellung. Sie ermöglicht die Überblendung (*Compositing*) mehrerer Texturen z.B. zur Hervorhebung von visuellen Details oder zur Darstellung textur-basierter Spezialeffekte. Zum letzteren gehört die *Bump-Mapping*-Technik, wobei neben einer Grundtextur eine weitere Textur die Rauhigkeit der Oberfläche angibt. Im folgenden bezeichnen wir diesen Texturtyp als Strukturtextur. Ziel ist es Algorithmen zu implementieren, die die Möglichkeiten moderner Graphikkarten hinsichtlich Mehrfach-texturierung ausschöpfen, aber auch ältere Graphikkarten unterstützen. Entsprechende Bedienschnittstellen sollen - zumindest im Modelliersystem *KisMo* - eine interaktive Auswahl und Manipulation von Texturen ermöglichen.

3.2.2. Hardwareunterstützte Multi-Texturierung

Die Graphikbibliothek OpenGL besitzt je nach Graphikkartenhersteller diverse Erweiterung, so auch *GL_ARB_multitexture* für die hardwareunterstützte Multi-Texturierung. Ist dies gegeben, so kann mit Hilfe des OpenGL-Befehls *glActiveTextureARB(GL_TEXTURE**x**_ARB)*

die Textureinheit mit dem Index *x* aktiviert und die entsprechende Textur mit *glBindTexture( )* in den Graphikkartenspeicher geladen werden. Die Festlegung der Texturkoordinaten erfolgt mit dem Befehl *glMultiTexCoord2fARB* und wird nacheinander für alle benutzten Textureinheiten festgelegt, bevor eine geometrische Primitive in einem *render-pass* dargestellt wird.

3.2.3. Die Embossed-Bump-Mapping-Technik

Eine visuelle Aufwertung von Simulationsmodellen erfolgt durch die Anwendung der sog. *Embossed-Bump-Mapping*-Technik (EBM) für die Darstellung einer gewissen Oberflächenrauhigkeit ohne die Modellkomplexität zu erhöhen [Len03][Nvi]. Die Idee hinter EBM ist in Abbildung 3-5 verdeutlicht: Man nimmt ein Grauwertbild *B* (im folgenden als Strukturtextur bezeichnet), berechnet die Inverse *B'* und verschiebt es auf der Objektoberfläche um eine benutzerdefinierte Anzahl an Pixel bzw. Texel in Richtung der Lichtquelle; das endgültige Bild *r(B+B')* ergibt das sog. *Embossed Texture* (ET).

Sei das Grauwertbild *B* definiert als eine Höhenfunktion *f(s,t)* Î *{0..1}*, dann ist das Reliefbild *r(s,t)* die Summe aus dem Originalbild und einer invertierten Kopie, die gerichtet verschoben ist; sie lässt sich berechnen als:
r(s,t) = 0.5+0.5**(f(s,t) – f(s+ds,t+dt))*
Der Verschiebungsvektor *(ds,dt)* gibt die Anzahl der Texel an, um die auf der Objektoberfläche in Richtung der Lichtquelle verschoben wird; *ds* und *dt* sind parametrisierbar.

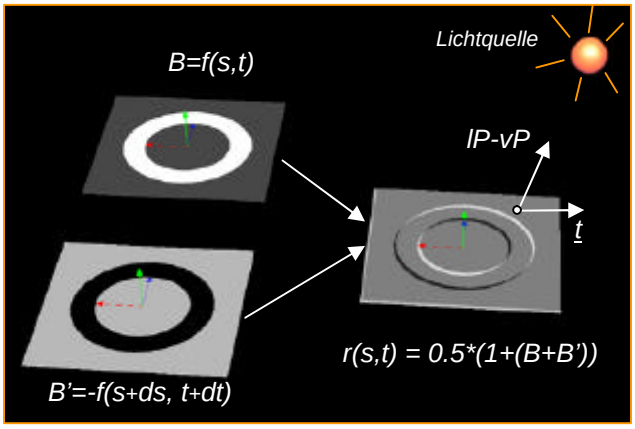

Abb. 3-5 *Embossed Texture* Berechnung

Für die praktische Umsetzung ist zu beachten, dass die Position der Lichtquelle (*lP*) in das Objektkoordinatensystem transformiert werden muss. Zudem muss für jedes Vertex des Objekts (*vP*) die sog. *TBN*-Matrix berechnet werden. Diese beinhaltet **T**angenten-, **B**inormalen- und **N**ormalenvektor [Len03] und muss nur bei jeder Änderung der Objektgeometrie aktualisiert werden. Der Verschiebungsvektor $\underline{v}$ wird folgendermaßen berechnet:

$$v = \left(\frac{lP - vP}{\|lP - vP\|} \cdot TBN \right) \cdot offset \qquad (3.2)$$

Dabei ist *offset* ein benutzerdefinierter skalarer Wert, der den Grad der Texturverschiebung angibt. Die Texturkoordinaten werden für die Darstellung des *Embossed Texture* um *ds*=$\underline{v}$[0] und *dt*=$\underline{v}$[1] verschoben.

3.2.4. Implementierung unter OpenGL

Da die Implementierung universell gestaltet sein soll, aber auch Möglichkeiten der hardwareunterstützten Multi-Texturierung ausschöpfen soll, wurden drei Algorithmen implementiert: *render-pass-3*, *render-pass-2* und *render-pass-1*. Entsprechend den Funktionsnamen brauchen sie je nach Hardwarebeschaffenheit drei, zwei oder einen Durchlauf für die

Zeichenroutinen pro Bildausgabe. Die OpenGL-Funktionsaufrufe für die Darstellung eines Objekts mit Struktur- und Farbtextur sind im folgenden für alle drei Algorithmen angegeben.

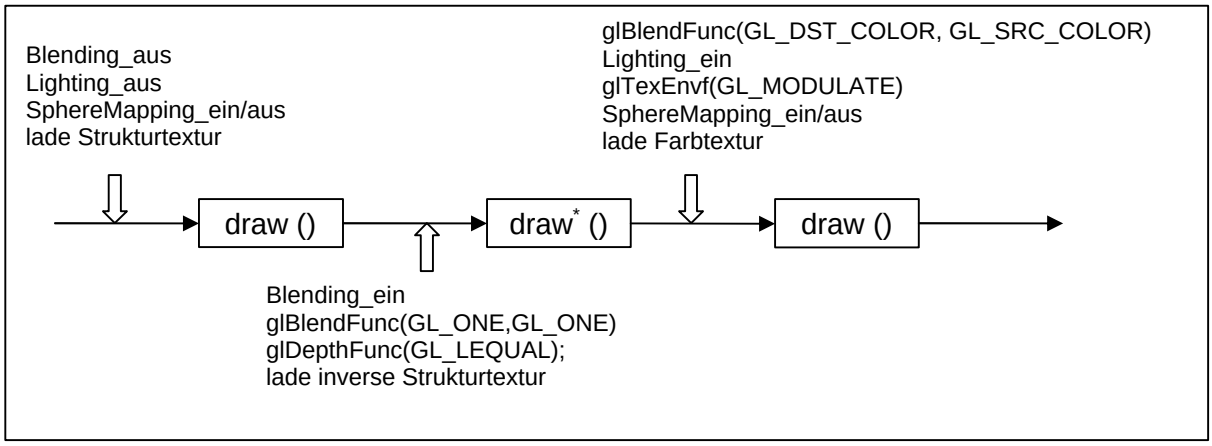

Abb. 3-6 Der *render-pass-3*-Algorithmus für das *Embossed Texturing*

Die einfachste und eine in Bezug auf die Zielhardware universelle Methode ist *render-pass-3* (siehe Abbildung 3-6), da stets nur eine Textureinheit verwendet wird. Man beachte, dass die Verschiebung der Texturkoordinaten für die Erstellung des *Embossed Texture* nur in *draw*( )* durchgeführt wird. Das sog. *Sphere-Mapping* (Umgebungstexturierung) kann vom Benutzer über die Benutzerschnittstelle für beide Texturtypen gesondert kontrolliert werden.

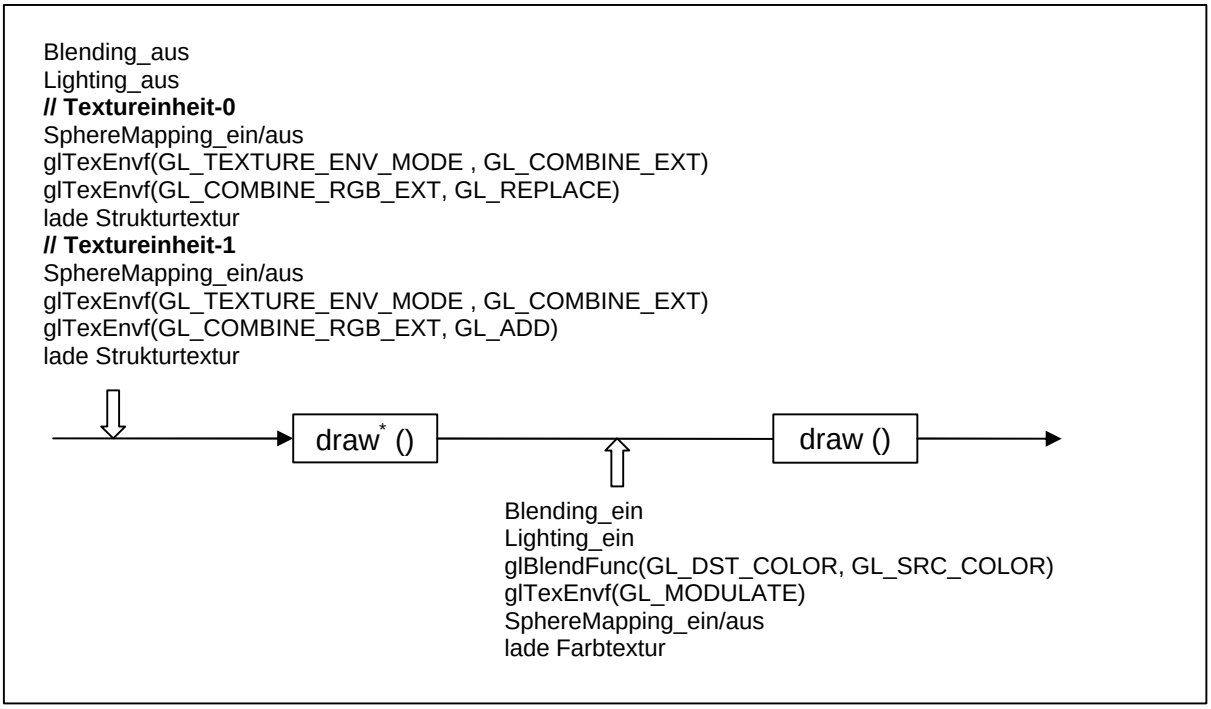

Abb. 3-7 Der *render-pass-2*-Algorithmus für das *Embossed Texturing*

Die Funktion *draw*( )* für die *render-pass-2* Implementierung (siehe Abbildung 3-7) setzt zwei Textureinheiten der Graphikkarte voraus, um die normale und inverse Strukturtextur gleichzeitig auf das Objekt abzubilden. Die notwendige Texturverschiebung für die inverse Textur wird mit *glMultiTexCoord2fAR* für die entsprechende Textureinheit angegeben.

Blending_ein
// Vermische Alpha-Kanal mit Farb-Kanälen: Überlagerung Bump mit Grundtextur
glBlendFunc(GL_SRC_ALPHA, GL_ZERO)
Lighting_ein
// Textureinheit -0
SphereMapping_ein/aus
Lade kombinierte Farb-Struktur-Textur
glTexEnvf(GL_TEXTURE_ENV_MODE , GL_COMBINE_EXT) ***//*** **Aktiviere Texture-Kombinierer**
glTexEnvf(GL_COMBINE_RGB_EXT, GL_MODULATE) ***//*** **Mische Texturfarben mit Materialfarben**
glTexEnvf (GL_COMBINE_ALPHA_EXT, GL_REPLACE) ***//*** **Lade Alpha-Kanal (A)**
// Textureinheit -1
SphereMapping_ein/aus
Lade kombinierte Farb-Struktur-Textur
glTexEnvf(GL_TEXTURE_ENV_MODE,GL_COMBINE_EXT) ***//*** **Aktiviere Texture-Combiner**
glTexEnvf(GL_COMBINE_RGB_EXT,GL_REPLACE) ***//*** **Lade TexturfarbenRGB**
glTexEnvf(GL_SOURCE0_RGB_EXT,GL_PREVIOUS_EXT); ***//*** **Übernehme Farben aus dem TU-0**
glTexEnvfGL_OPERAND0_RGB_EXT,GL_SRC_COLOR); ***//*** **Benutze RGB-Werte**
glTexEnvfGL_SOURCE0_ALPHA_EXT,GL_TEXTURE) ***// Bearbeite Alpha-Kanal***
glTexEnvf(GL_COMBINE_ALPHA_EXT,GL_ADD_SIGNED_EXT) ***// Alpha =A + B – 0.5***
glTexEnvf(GL_OPERAND0_ALPHA_EXT,GL_ONE_MINUS_SRC_ALPHA) ***// B = 1-A***
glTexEnvf(GL_SOURCE1_ALPHA_EXT,GL_PREVIOUS_EXT) ***// Hole A (=Transparenzkanal aus TU-0)***
glTexEnvf(GL_OPERAND1_ALPHA_EXT,GL_SRC_ALPHA) ***// Benutze alpha-Werte***

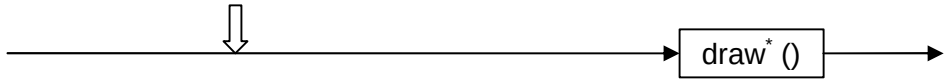

Abb. 3-8 Der *render-pass-1*-Algorithmus für das *Embossed Texturing*

Bei *render-pass-1* (siehe Abbildung 3-8) wird eine einzige 4-Kanal Textur verwendet, die RGB-Farbkanäle entsprechen denen der Farbtextur, der Alphakanal beinhaltet die Strukturtextur. Mit Hilfe der Erweiterung *GL_EXT_texture_env_combine* lassen sich die Textureinheiten programmieren, d.h. Operationen auf Texturen anwenden und miteinander kombinieren. Die Strukturtextur, die bei diesem Algorithmus im Wertebereich [0;0.5] definiert sein muss, wird in einer Textureinheit invertiert und mit der normalen Strukturtextur durch Addition (*GL_ADD_SIGNED_EXT)* wieder auf den Bereich [0;1] abgebildet. Bei dieser Methode erscheinen die Farben subjektiv blasser, da sie mit den Werten aus dem Alpha-Kanal skaliert werden. In der Literatur wird daher meist eine Verdopplung des RGB-Anteils empfohlen, was jedoch zu einer verfälschten Farbwiedergabe führt. Für die praktische Implementierung im obigen Algorithmus sei vermerkt, dass die Skalierung mit *glTexEnvf(GL_TEXTURE_ENV, GL_RGB_SCALE_EXT, 2)* für Textureinheit-0 durchgeführt werden muss.

3.2.5. Vergleich und Diskussion der Methoden

Der Vergleich der drei Methoden erfolgt zum einen über die erzeugte Bildqualität, zum anderen über die Bildrate (Abbildung 3-9). Für den Vergleich wurde ein *Pentium-III 1GHz* Rechner mit einer *nVidia GeForce 6600GT* Graphikkarte verwendet. Die Implementierung wurde in das mit dem *FOX-Toolkit* [FOX] erstellte GUI integriert, das nicht für hohe Bildraten optimiert ist, jedoch eine hohe Flexibilität während der Programmierung und einen großen Bedienkomfort ermöglicht. Wie oben bereits erwähnt, liefert *render-pass-1* eine schlechtere Bildqualität, da die Farben nur mit der halben Intensität dargestellt werden und die Darstellung subjektiv „zu dunkel“ wirkt. Die Bildrate ist im Vergleich zu den anderen Methoden etwas höher, da nur ein Zeichendurchlauf benötigt wird, jedoch die Textur-

operationen auch Rechenzeit in Anspruch nehmen. Diese Methode ist bedingt empfehlenswert, da:

- die Bildqualität nicht optimal ist
- eine separate Darstellung der Strukturtextur ohne Farbtextur nicht möglich ist (die Grund- und Strukturtextur müssen kombiniert als eine RGBA-Textur vorliegen)
- nur auf Graphikkarten mit mehreren Textureinheiten lauffähig ist.

Die Methode *render-pass-2* liefert qualitativ sehr gute Ergebnisse, die Bildwiederholrate ist akzeptabel. Der Vorteil dieser Methode ist, dass die Strukturtextur von der Farbtextur getrennt bearbeitet wird und daher die Farbtextur mit weiteren Texturen (z.B. *Spotlight*-Textur) unter Ausnutzung mehrerer Textureinheiten dargestellt werden kann. Sie ist standardmäßig anzuwenden, wenn mehrere Textureinheiten benutzt werden können.

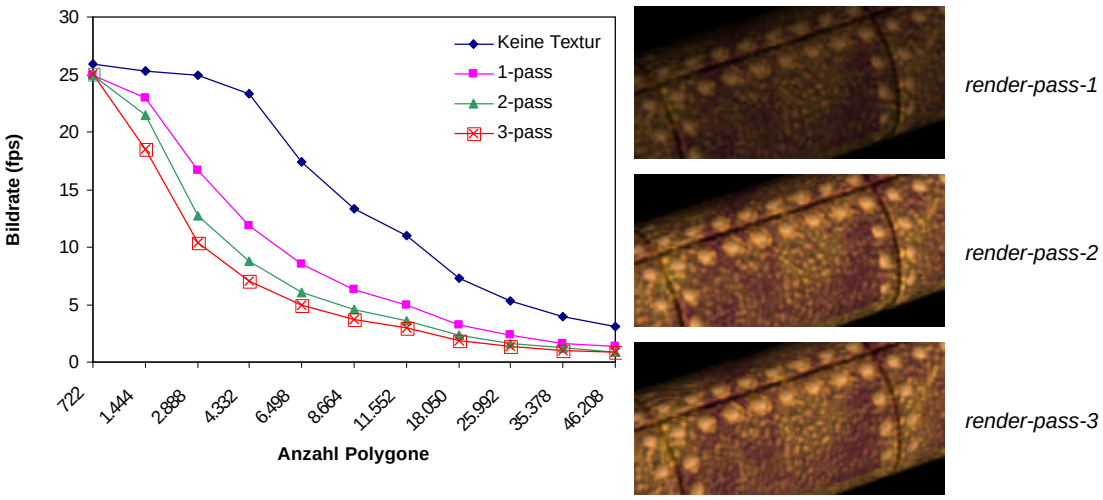

Abb. 3-9 Vergleich der drei Algorithmen für das *Embossed Texturing*

Die Methode *render-pass-3* liefert qualitativ sehr gute Ergebnisse. Der Vorteil dieser Methode ist ihre universelle Einsetzbarkeit, da eine einzige Textureinheit benötigt wird, was bei älteren Graphikkarten der Fall ist. Nachteil ist die niedrige Bildwiederholrate bei niedriger und mittlerer Modellkomplexität, bei sehr hoher Modellkomplexität sind die Bildraten aller drei Methoden nahezu gleich.

3.3. Optimierung der Deformationssimulation mit der FFE-Methode

Neben den Themen Modellierung und echtzeitfähige photorealistische Modelldarstellung wurde im Rahmen des Projekts als dritter Themenkomplex die Simulation deformierbarer Objekte näher untersucht.

3.3.1. Problemstellung und Motivation

KISMET benutzt für die Deformationssimulation vorwiegend Feder-Masse-Netze aufgrund ihrer einfachen Handhabbarkeit. Eine Implementierung der *Fast-Finite-Element*-Methode (FFE) [Bro98] wurde im Rahmen einer Studie zur Deformationsuntersuchung von harten Materialien am Institut für Angewandte Informatik durchgeführt [KÇ98]. Die Methode konnte bisher jedoch nicht problemlos in einem Chirurgiesimulator eingesetzt werden, da die notwendige Einbindung in die Modellstruktur und die Möglichkeiten der Modellinteraktion ausstanden.

3.3.2. Implementierung der FFE-Methode in KISMET

Bei der FFE-Methode wird ein linearer Zusammenhang zwischen einer wirkenden Kraft *f* und einer Objektverformung durch Knotenverschiebung *u* durch ein Gleichungssystem der Form *Ku=f* ausgedrückt. Die Steifigkeitsmatrix *K* lässt sich durch Blockmatrizen beschreiben, wobei die Objektknoten nach ihrer Lage als Oberflächenknoten oder innere Knoten geordnet vorliegen.

$$\begin{bmatrix} K_{oo} & K_{oi} \\ K_{io} & K_{ii} \end{bmatrix} \begin{bmatrix} u_o \\ u_i \end{bmatrix} = \begin{bmatrix} f_o \\ f_i \end{bmatrix} \tag{3.3}$$

Die Steifigkeitsmatrix beinhaltet Teilmatrizen für die Oberflächenknoten K_{oo} und für die inneren Knoten K_{ii}, auf die jedoch im folgenden aufgrund der verwendeten Modellstruktur verzichtet wird. Das reduzierte System bestehend aus Oberflächenknoten lässt sich mathematisch umformen zu $K'_{oo}u_o=f'_o$, wobei wir im folgenden die Indizes weglassen und vereinfacht als *Ku=f* schreiben. Die Berechnung der Steifigkeitsmatrix *K* erfolgt unter Ausnutzung der Elastodynamiksimulation für Feder-Masse-Netze. Dabei wird sequentiell jeder Objektknoten mit einem vorgegebenen Kraftvektor *f* ausgelenkt, die dabei entstehenden Positionsänderungen u_i aller restlichen Knoten registriert und die Werte f_i/u_i in der inversen Steifigkeitsmatrix abgelegt. Die Dauer dieses Vorverarbeitungsschritts ist modellabhängig und kann je nach Prozessorleistung und Anzahl der Integrationsschritte mehrere Minuten bis Stunden in Anspruch nehmen. Es wird dabei eine einzige globale Steifigkeitsmatrix für alle Modelle in der Simulationsszene erstellt.
Für die Deformationssimulation wird die Steifigkeitsmatrix zunächst eingelesen und zur Simulationszeit aus den benutzerdefinierten externen Kräften durch selektive Matrix-Vektor-Multiplikation die Auslenkung $u=S_j(K_j^{-1}f_j)$ der restlichen Knoten berechnet. Diese Art der Deformationssimulation nennen wir im folgenden die elastostatische Deformationssimulation (synonym mit FFE verwendet), die im Gegensatz zur elastodynamischen Deformations-simulation mit einem Feder-Masse-Netzwerk unabhängig von der Simulationszeit ist.

3.3.3. Erweiterungen der FFE-Implementierung für den Einsatz im Chirurgiesimulator

Die ursprüngliche Implementierung der FFE-Methode ist in dem VEST Simulator für chirurgische Applikationen nicht ohne weiteres möglich. Hierzu wurden folgende Erweiterungen und Optimierungen durchgeführt:

- Adaptive Berechnung der inversen Steifigkeitsmatrix
- Objektorientierte Aufteilung der globalen Steifigkeitsmatrix
- Hybride Modellhaltung und -simulation
- Möglichkeit der Interaktion mit den Modellen
- Nicht-lineare Objektdeformation durch zweistufigen FFE-Ansatz.

Die ursprüngliche Implementierung der Berechnung der inversen Steifigkeitsmatrix liefert bei ungünstiger Parameterwahl (Anzahl der Auslenkungsschritte) teilweise falsche Ergebnisse, da die Berechnung bei zu wenigen Auslenkungsschritten entweder vorzeitig abgebrochen wird oder Berechnungen unnötig lang dauern, falls die Anzahl der Integrationsschritte zu groß gewählt wird. Die neue adaptive Berechnung der Steifigkeitsmatrix liefert exakte Ergebnisse ohne Zeitverlust. Dabei wird jeder Objektknoten sequentiell durch eine externe Kraft ausgelenkt, was sich aufgrund des Feder-Masse-Netzes positionsverändernd auf die

benachbarten Knoten auswirkt. Ist die Positionsänderung aller Objektknoten nicht mehr signifikant, werden die Teilergebnisse registriert, die Objektform zurückgesetzt und die Bearbeitung mit dem nächsten Objektknoten fortgesetzt. Abbruchkriterium für einen Auslenkungsschritt ist der Vergleich der Summe der Geschwindigkeitsvektoren aller Objektknoten während der Auslenkung mit einem vorgegebenen Schwellwert. Vorteil der adaptiven Methode ist eine garantiert vollständige Auslenkung der Objektknoten und somit exakte Werte in der Steifigkeitsmatrix.

Die Größe der Steifigkeitsmatrix K für n Objektknoten beträgt $72n^2$, wobei jedes Matrixelement K_{ij} eine 3x3 Untermatrix ist und als *double*-Wert mit 8 Byte im Speicher bzw. Festplatte gehalten wird. Angenommen, die Größe eines Simulationsszenarios bewegt sich im Rahmen von ca. 10.000 Objektknoten verteilt auf mehrere Objekte. Dafür sind in der bisherigen Implementierung 7.2GB bei einer einzigen globalen Steifigkeitsmatrix notwendig. Unterteilt man die Steifigkeitsmatrix jedoch objekt-orientiert, so dass jedem Objekt eine eigene Steifigkeitsmatrix zugeordnet ist, so reduziert sich der benötigte Speicherplatz erheblich. Die beste Reduktionsrate wird erzielt, wenn in der Simulationsszene alle Objekte die gleiche Anzahl von Knoten besitzen (Beweis der „Minimierung der Summe der Quadrate“ mit vollständiger Induktion). Seien k Objekte mit insgesamt n Knoten in der Simulationsszene wobei jedem Objekt n/k Knoten zufallen, so beträgt der benötigte Speicherplatz nur noch $72k(n/k)^2 = 72n^2/k$. Bei der Modellierung sollte daher versucht werden, das Simulationsszenario aus möglichst vielen Teilmodellen mit nahezu gleicher Knotenmächtigkeit zu erstellen

Da die ursprüngliche Implementierung der FFE-Berechnung in KISMET eine Alles-oder-Nichts-Lösung war, konnte eine elastodynamische und elastostatische Berechnung nicht gleichzeitig durchgeführt werden. Nach Umschaltung in das FFE-Berechnungsmodus wurde eine globale Steifigkeitsmatrix eingelesen und die Deformationssimulation aller Objekte mit der elastostatischen Methode berechnet. Eine objekt-orientierte Aufteilung der Steifigkeitsmatrix ermöglicht nun die selektive Zuweisung der Berechnungsmethode zu einzelnen Objekten und dadurch eine hybride Modellhaltung und -berechnung. Die Berechnungsmethode ist abhängig von der Art, wie Objekte in der Simulationsszene interagieren. Sollen sie nur verformbar sein und keine Topologieänderung erfahren, so ist eine FFE-Dynamikberechnung ausreichend, ansonsten sollte eine elastodynamische Dynamikberechnung gewählt werden. Die Festlegung der Berechnungsmethode erfolgt mittels neuer Skript-Befehle in der KISMET-Skriptbibliothek.

Die Möglichkeiten der Modellinteraktion sind beschränkt auf solche, die keine dauerhafte Änderung der Modelltopologie herbeiführen wie etwa Greifen, Zug und Druck. Eine Modifikation der Topologie z.B. durch virtuelle chirurgische Schnitte bedarf der Neuberechnung der Steifigkeitsmatrix des entsprechenden Objekts, was jedoch nicht echtzeitfähig durchgeführt werden kann, außer es wird eine Parallelisierung der Steifigkeitsmatrixberechnung durchgeführt. Prinzipiell setzt die FFE-Methode eine externe Krafteinwirkung auf Objektknoten voraus, um daraus die Knotenverschiebungen zu ermitteln. Da in einem Chirurgiesimulator jedoch die Knotenverschiebung aufgrund der Kollisionsbehandlung mit den Eingabeinstrumenten vorgegeben wird, müssen für alle interagierenden Knoten k_i in einem Vorverarbeitungsschritt die externen Kräfte ermittelt werden, um in der eigentlichen FFE-Berechnung eine Auslenkung der interagierenden Knoten k_i von ihrer Ruheposition zur Zielposition und damit eine Objektverformung zu erreichen.

Eine Erweiterung der bestehenden FFE-Algorithmen ist der Versuch einer Approximation eines nichtlinearen Deformationsverhaltens. Nicht-lineare Elastizitätseigenschaften von Organen werden gewöhnlich mittels Polynome vierten Grades beschrieben und basieren auf Messungen an organischen Schlachtabfällen und in-vivo Messungen an tierischen Organen [MK99]. Untersucht wurde hier wie mit Hilfe des FFE-Ansatzes das typische Verhalten von Gewebe bei Zug nachgebildet werden kann. Bei biologischem Gewebe ist zunächst ein linearer Zusammenhang zwischen Zugkraft und Verformung in einem gewebespezifischen Bereich vorhanden, für eine weitere Dehnung ist ein erhöhter Kraftaufwand notwendig. Wir bilden dieses Phänomen durch die Kombination von zwei Steifigkeitsmatrizen K_1 und K_2 nach. Für die Berechnung von K_1 werden die Gewebeparameter derart eingestellt, dass das Gewebe elastodynamisch leicht verformbar ist, für K_2 werden Gewebeparameter mit höherer Steifigkeit gewählt. Die folgende Formel beschreibt die Kombination von zwei Steifigkeitsmatrizen zur Berechnung der Objektverformung u bei einer Kraft f. Die Berechnung der Knotenauslenkungen erfolgt in Abhängigkeit von einem Kraftschwellwert f_0, indem zwischen den Steifigkeitsmatrizen K_1 und K_2 umgeschaltet wird.

$$u(K_1, K_2, f) = K_1^{-1} f - \mathrm{sgn}(\| f - f_0 \|) \cdot K_2^{-1}(f - f_0) \qquad (3.4)$$

Eine graphische Darstellung dieser Funktion ist in der folgenden Abbildung gegeben. Ab einem vorgegebenen Schwellwertpaar (u_0, f_0) ist eine erhöhte Zugkraft notwendig um das Gewebe weiter zu dehnen. Dies wird in der FFE-Simulation ebenfalls deutlich, dabei zeigt $u_{K1}(f_0)$ den Zustand an, ab der die zweite Steifigkeitsmatrix K_2 aktiviert wird. Bei gleicher Krafteinwirkung ist bei der neuen Funktion mit der Kombination von zwei Steifigkeitsmatrizen $u_{K1,K2}$ eine kleinere Deformation zu erkennen als bei der Berechnung mit nur einer Steifigkeitsmatrix u_{K1}.

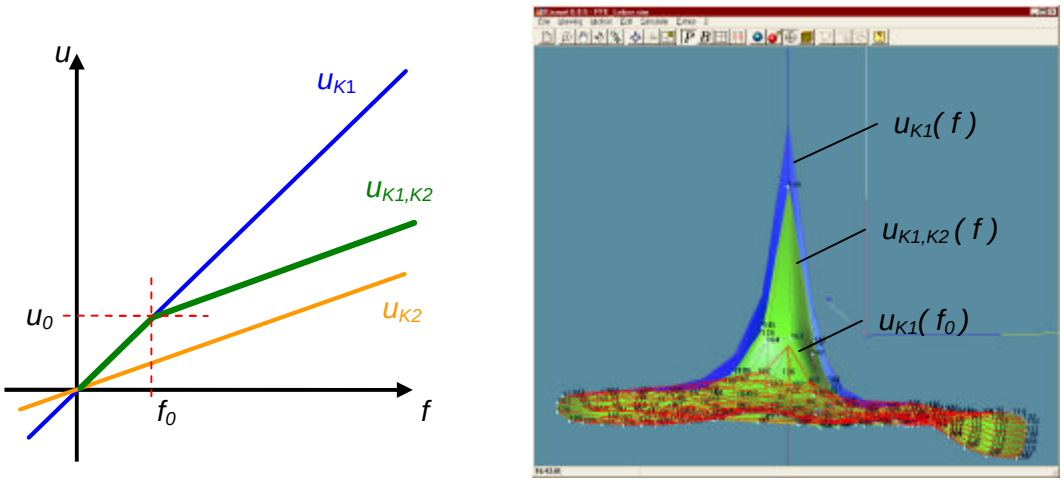

Abb. 3-10 Zweistufige Deformationssimulation mit der FFE-Methode

4. Ergebnisse

4.1. Neue Modellierungsmethoden für deformierbarer Objekte

Die Modelliersoftware KisMo wurde um das Modul *MetaBalls* erweitert. Dieser ermöglicht interaktiv das Editieren von Partikeln zur Erstellung einer impliziten Oberfläche. Die Dichteverteilung wird in einem Volumendatensatz gespeichert und mittels Volumen-

visualisierung dargestellt. Abbildung 4-1-*links* zeigt ein eingefärbtes Dichtefeld. Durch Wahl eines Isowerts werden entsprechende Voxel markiert und mit Hilfe des optimierten Marching-Cubes-Algorithmus in ein Dreiecksnetz konvertiert. Abbildung 4-1-*rechts* zeigt die generierte implizite Oberfläche mit den in KisMo verfügbaren Darstellungsmodi für polygonale Modelle.

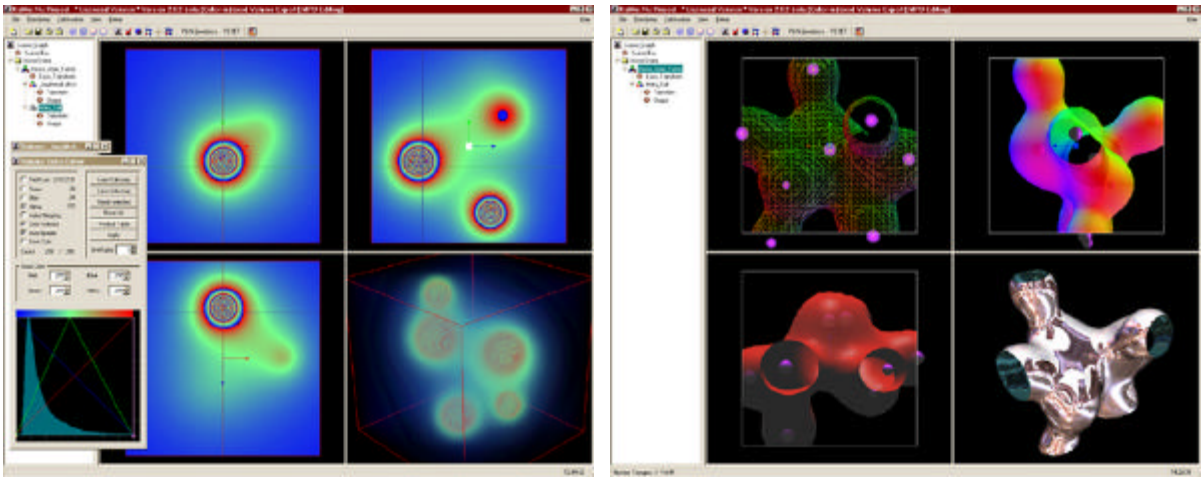

Abb. 4-1 Modellierung und Darstellung impliziter Oberflächen in KisMo

Das erzeugte Dreiecksnetz kann als statisches 3D-Geometriemodell oder als deformierbares Simulationsmodell zur Verwendung mit KISMET exportiert werden. Der neue Exportfilter für das KISMET ELA_EXPL-Format erzeugt ausgehend von der Objektgeometrie das Feder-Masse-Netz; für das obige Modell beträgt die Anzahl der Knoten 920 und für die Federelemente 2.658. Die Zuweisung der korrekten Deformationsparameter erfolgt je nach gewünschtem Materialverhalten und Applikation im Modelliersystem. Hier wurde ein Standard-Parametersatz aus KisMo gewählt, welches eine Simulationsinstabilität ausschließt. Die Deformationssimulation in KISMET läuft auf einem *Pentium-III-1GHz* Rechner mit 12 Elastodynamik-Zyklen, was bei dem verwendeten Einprozessor-Rechner der Bildrate entspricht. Die elastodynamische Echtzeitdeformation des modellierten Objekts in KISMET ist in der folgenden Abbildung gezeigt, bei der eine externe Kraft auf eine ausgewählte Gruppe von Objektknoten wirkt.

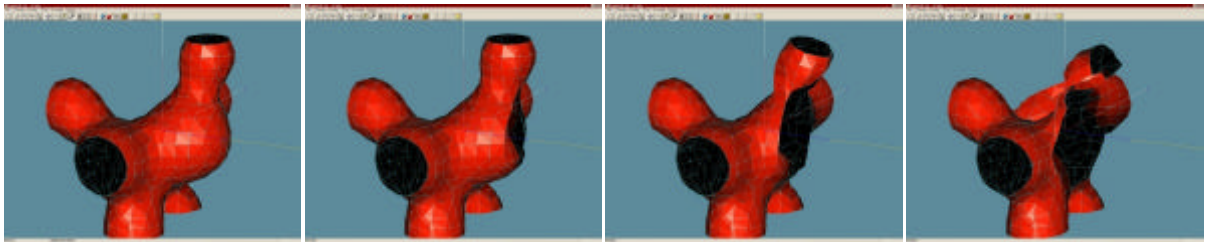

Abb. 4-2 Deformationssimulation von beliebigen Dreiecksnetzen mit KISMET

Das neue Verfahren für den Modelldatenaustausch unter Verwendung des ELA_EXPL-Datentyps wurde u.a. an einem im World-Wide-Web frei verfügbaren 3D-Modell (Leber, Gallenblase, Gallengang) getestet, wobei die einzelnen Objekte in einem zusammenhängenden, irregulären Dreiecksnetz zusammengefasst sind. Das Modell, das im AutoCAD dxf-Format (v.13, 3D-Faces) vorliegt, besteht aus 4.533 Knoten und 13.558 Federelementen. Es hat im Gegensatz zu bisherigen Simulationsmodellen in KISMET keine regelmäßige

Anordnung der Objektpunkte in einer Matrixstruktur. In KisMo wurde zunächst das dxf-Modell in das ELA_EXPL-Format für KISMET konvertiert und in ein Simulationsmodell eingebunden. Es werden mehrere Objektknoten gepickt und die Deformation durch Kraftdefinition an den gewählten Knoten herbeigeführt. Gut erkennbar ist die Modelldeformation an der Gallenblase und an der linken Seite der Leber (Abbildung 4-3). Die Elastodynamik wird auf einem *Pentium-III 1GHz* Rechner drei mal pro Sekunde berechnet, auf einem Mehrprozessor-rechner werden interaktive Bild- und Berechnungsraten erreicht.

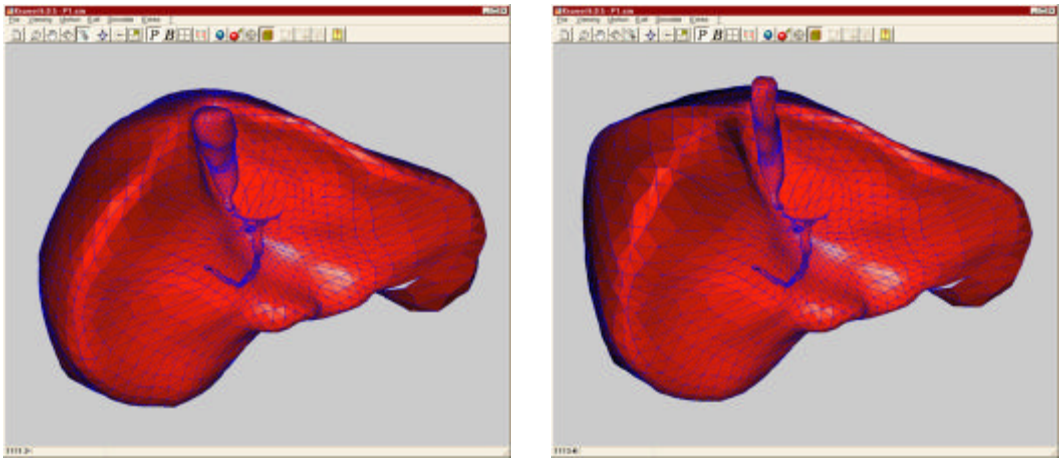

Abb. 4-3 Deformationssimulation eines Gallenblase-Lebermodells mit irregulärem Dreiecksnetz in KISMET

Als ein weiteres Anwendungsgebiet für implizite Oberflächen wurde die 3D-Modellmodifikation vorgestellt. Exemplarisch wurde ein polygonales 3D Modell der weiblichen Reproduktionsorgane für chirurgische OP-Simulation in der Gynäkologie ausgewählt. Das Modell besteht aus einem einzigen Dreiecksnetz, welches den Uterus, Ovarien und die Eileiter beinhaltet. Als Simulationsmodell ist eine derartige Repräsentation eher ungeeignet, da Sie den modularen Austausch der Teilmodelle nicht ermöglicht. Für die Demonstration der Modellmodifikation mit unserer Methode ist die Modellauswahl vertretbar. Ausgehend von einem beliebigen Dreiecksnetz wird – wie in Abbildung 4-4 dargestellt – eine Voxelrepräsentation des polygonalen Modells mit einer Auflösung von 256^3 Voxel erstellt. In einem weiteren Schritt werden fünf Partikel zur Generierung der Myome geeignet platziert und mit dem Volumenmodell zu einer impliziten Oberfläche verschmolzen. Bei der Triangulierung wurde eine niedrigere Auflösung gewählt, um die Echtzeitfähigkeit der Deformationssimulation zu gewährleisten. Das fertige Modell kann für die Simulation der Entfernung von Myomen am Uterus eingesetzt werden.

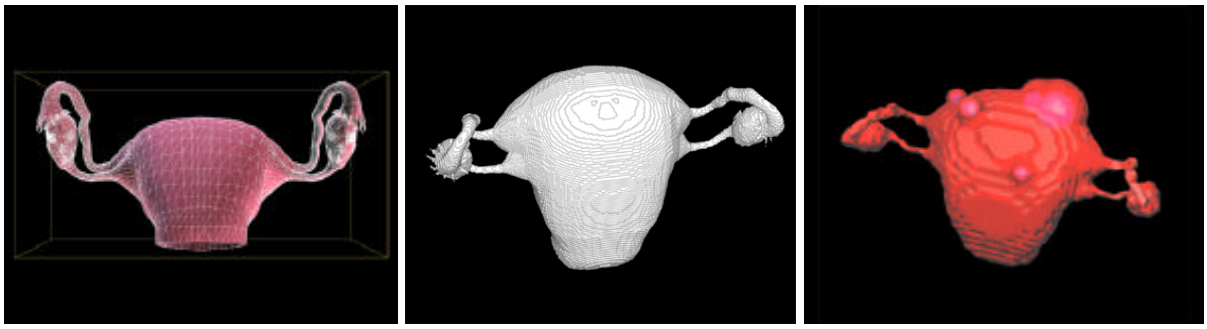

Abb. 4-4 Modellmodifikation mit impliziten Oberflächen: „Myome am Uterus“

Die Verwendung binärer Werte bei der Voxelisierung macht sich in der Polygonaldarstellung negativ bemerkbar. Vergleicht man das Originalmodell mit dem modifizierten Modell, so fallen starke *Aliasing*-Effekte (Stufen) auf. Diese können eliminiert werden, falls anstatt einer binären Voxelisierung eine höhere Wertigkeit und Filteralgorithmen eingesetzt werden. Wir verwenden einen dreidimensionalen Glättungsfilter. Bei der Voxelisierung eines polygonalen Datensatzes wird ein binärer Volumendatensatz mit sehr großer Auflösung erzeugt und anschließend auf die kleinere Zielgröße skaliert, wobei Graustufen durch die Bildung des arithmetischen Mittels der Voxelwerte entstehen. In den folgenden Abbildungen ist die Wirkung der Glättungsoperation sichtbar: Links die direkte Voxelisierung auf die Zielgröße 128^3, rechts eine Voxelisierung mit 768^3 und anschließender Skalierung auf 128^3 jeweils mit Volumenvisualisierung und polygonaler Darstellung nach einer Triangulation.

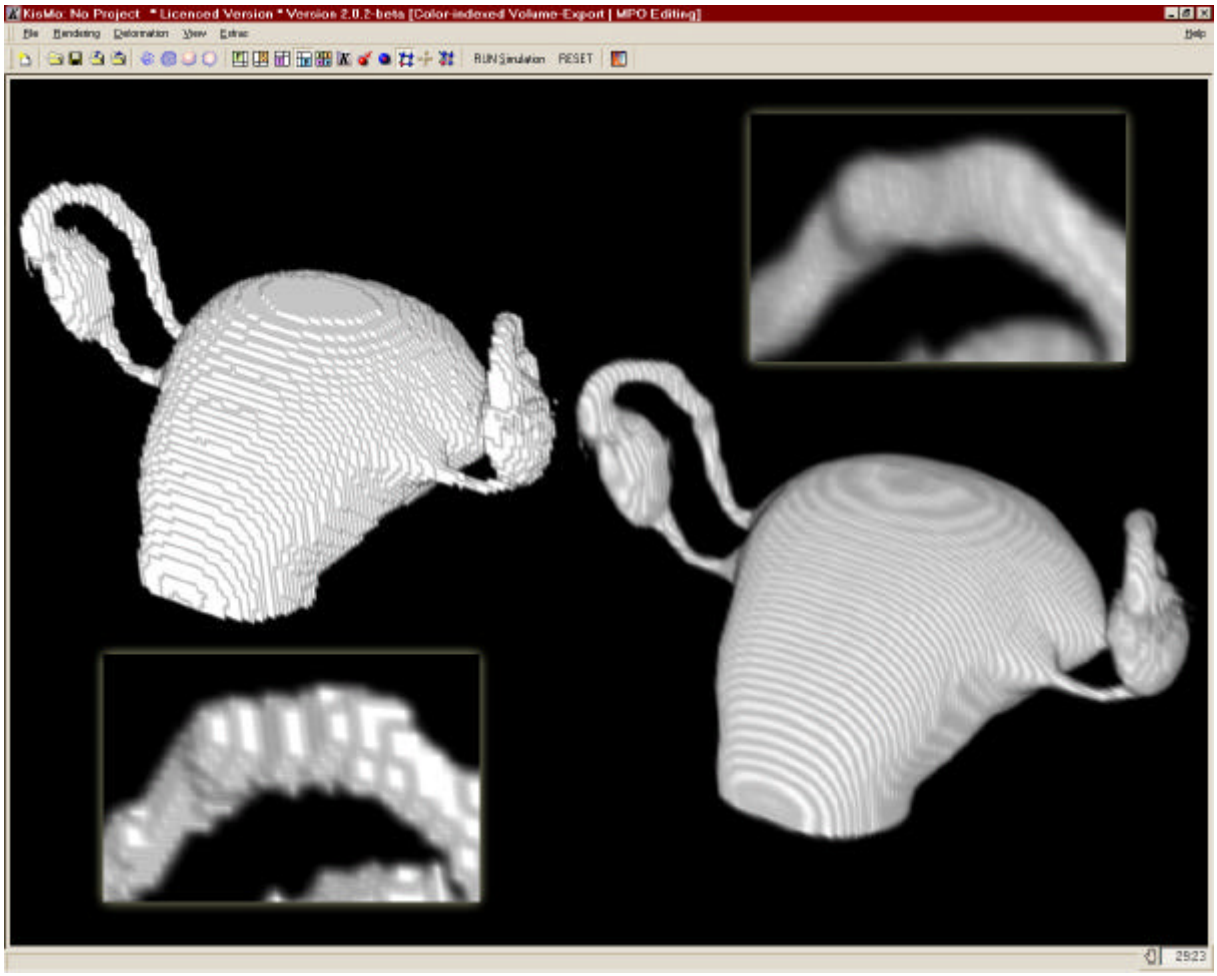

Abb. 4-5 Glättung bei der Voxelisierung dargestellt mit Volumenvisualisierung

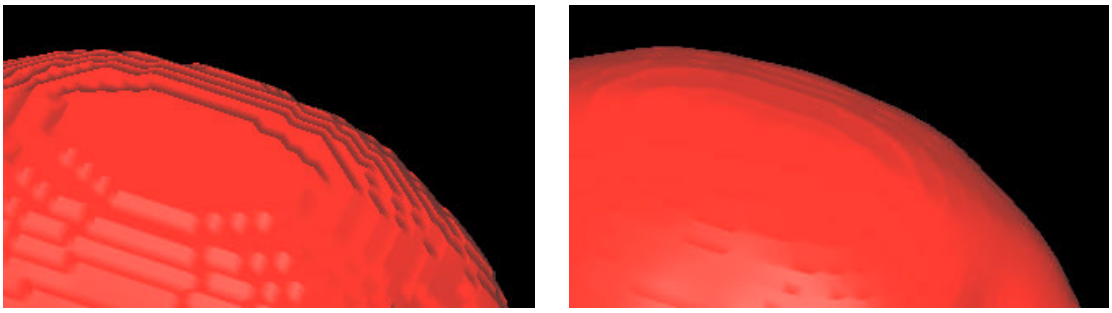

Abb. 4-6 Vergleich der Dreiecksnetze bei direkter und gefilterter Voxelisierung

4.2. Echtzeitfähige Texturierungsmethoden

Oberflächentexturierung ist eine gängige Methode zur Steigerung des visuellen Realismus. In der Praxis reicht jedoch eine einzige Textur hierfür nicht aus. Eine Mehrfachtexturierung in Kombination mit speziellen Darstellungstechniken, wie die vorgestellte echtzeitfähige *Embossed Texturing* ermöglicht nahezu photorealistische Darstellungsmöglichkeiten in Echtzeitanwendungen. Exemplarisch für die vorgestellten Methoden soll die Texturkomposition für ein 3D-Modell der Lunge, die auf dem *Visible Human* Datensatz basiert, vorgestellt werden. Das patientenspezifische Modell wurde aus einem DICOM-Datensatz gemäß der in [CMS02] beschriebenen Methode erstellt.
Wir verwenden vier Texturen (Abbildung 4-7), von denen zunächst die Grundfarbe des Modells durch eine Basistextur (*a*) definiert wird. Eine weitere Textur (*b*) wird additiv mit der Grundtextur überlagert und erlaubt die Darstellung weiterer Details. Die visuelle Simulation der Reflektion einer Lichtquelle auf einer feuchten Oberfläche wird mit einer weiteren Textur (*c*) dargestellt, hierfür setzen wir additive Texturierung und *Sphere-Mapping* ein. Die vierte Textur wird für die Strukturierung der Oberfläche mit der *Embossed-Texture-Mapping*-Methode (*d*) benutzt. Diese ist ähnlich der Textur (*b*), die eine visuell verstärkende Wirkung der Strukturen besitzt. Die folgende Abbildung zeigt alle Texturen separat, die Texturkomposition erfolgt mit der *render-pass-2*-Methode (siehe Abbildung 4-8). Eine Darstellung mit der *render-pass-1*-Methode liefert ein besseres Echtzeitverhalten, da mehrere Textureinheiten gleichzeitig angesprochen werden, jedoch ist die Darstellungsqualität aufgrund der im letzten Kapitel beschriebenen Tatsache im Vergleich zu den anderen beiden Methoden schlechter.
Es sollte kurz darauf hingewiesen werden, dass die *Embossed-Bump-Mapping*-Technik eine Visualisierung von sehr feinen Oberflächenstrukturen ermöglicht ohne die Objektgeometrie sehr detailliert modellieren zu müssen, was bei echtzeitkritischen Anwendungen besonders vorteilhaft ist.

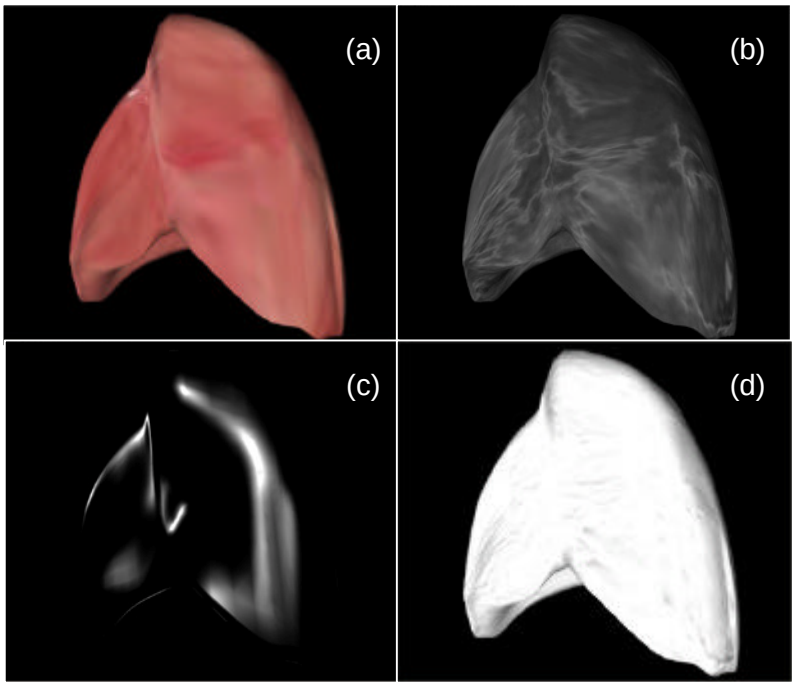

Abb. 4-7 Ausgangstexturen für die Texturkomposition

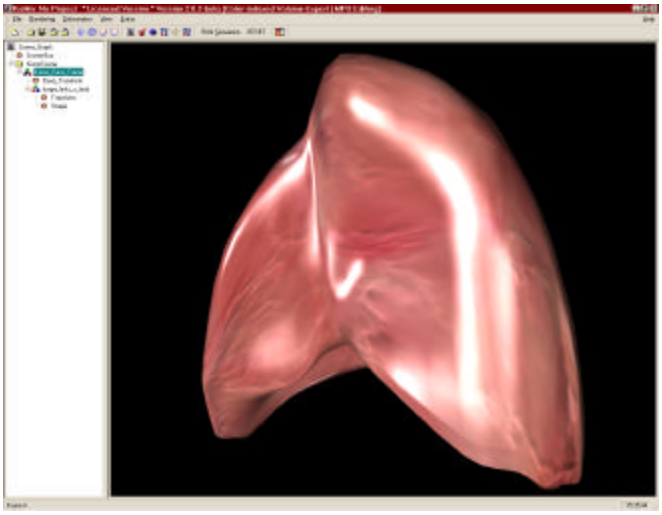

Abb. 4-8 Photorealistische Modelldarstellung mit Texturkomposition in Echtzeit

4.3. Hybride Simulationsmodelle für chirurgische Anwendungen

Es wurde die bestehende Simulationsszene für die Cholezystektomie bearbeitet, in dem alle Organmodelle, die primär nicht im Fokus der Operation stehen und daher nicht Topologie verändernd interagieren sollten, als FFE-Objekte definiert. Im Cholezystektomie Modell sind dies die Modelle für die Leber, Darm, periphere Arterien und Gallengänge (ca. 66,2% der Objektknoten des Simulationsmodells), für die jeweils eine separate Steifigkeitsmatrix in einem Vorverarbeitungsschritt berechnet wurde. Alle restlichen Organmodelle werden mit der elastodynamischen Methode berechnet.
In einer Zeitmessung auf einem *Pentium-III-500-Dual-Prozessor*-Rechner wurde ermittelt, wie sich die Berechnungsrate für die Deformationssimulation eines hybriden Modells mit einem steigendem Anteil an FFE-Objekten verhält. In der Abbildung 4-9-*links* ist erkennbar, dass die Steigerung der Rechenzyklen proportional zum Anteil von FFE-Objekten ist. Eine Verdopplung der Dynamik-Rechenzyklen für die Deformationssimulation wird bei einem Anteil von 66,2% FFE-Objekte erreicht.

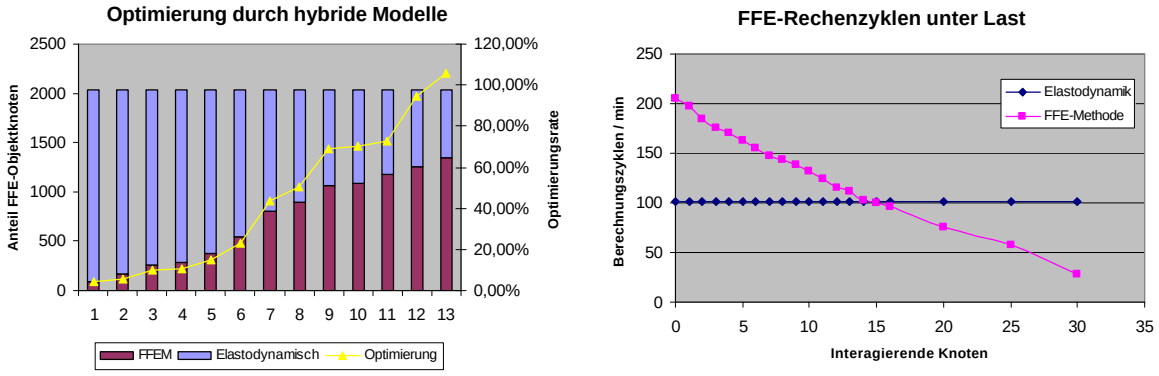

Abb. 4-9 Beschleunigung der Deformationsberechnung bei einem hybriden Cholezystektomie-Simulationsmodell

Es sollte bemerkt werden, dass die FFE-Methode Last abhängig ist, d.h. solange keine Objektknoten mit virtuellen Eingabegeräten interagieren, ist die Berechnung sehr schnell,

ansonsten muss durch Matrixmultiplikation die Positionsänderung der entsprechenden Knoten bei Krafteinwirkung berechnet werden. Abbildung 4-9-*rechts* zeigt das Verhältnis der Anzahl der Dynamikrechenzyklen pro Minute bei steigender Anzahl interagierender Knoten eines FFE-Objekts. Es wurde das hybride Cholezystektomie-Modell mit 66,2% FFE-Knotenanteil verwendet. Liegt keine Last an, so sind 205 Rechenzyklen pro Minute möglich, das ist mehr als das Doppelte wie bei der voll-elastodynamischen Methode. Findet eine Interaktion mit 14 Objektknoten gleichzeitig statt, was während einer interaktiven chirurgischen Simulation aufgrund der vielen Objekt-Instrument-Kollisionen durchaus möglich ist, so halbiert sich die Zahl der Rechenzyklen und entspricht nur noch dem eines voll-elastodynamischen Cholezystektomie-Modells. Bei mehr als 14 gleichzeitigen Knoten-interaktionen wird die Berechnung sogar langsamer als beim voll-elastodynamischen Modell. Je nach Simulationsmodell und benötigten Interaktionmöglichkeiten muss daher der Einsatz eines hybriden Simulationsmodells erwogen werden, da die FFE-Berechnung im jetzigen Entwicklungsstand keine Modellmodifikation erlaubt und bei Kontakt mit mehreren Objektknoten die Zahl der Rechenzyklen stark heruntersetzen kann.

5. Zusammenfassung und Ausblick

Im Rahmen der Forschungsarbeiten zur Gewebemodellierung wurden in den Bereichen Modellierung, graphische Echtzeitdarstellung und Deformationssimulation neue Algorithmen implementiert. So ermöglicht die neue Softwarekomponente von KisMo eine intuitive und effiziente Modellierung mit impliziten Oberflächen. Im Rahmen dieses Projekts wurden auch Erweiterungen für die Simulationssoftware KISMET implementiert, hervorzuheben ist dabei die Möglichkeit deformierbare Objekte mit beliebigem Dreiecksnetz zu bearbeiten, wodurch der Austausch von Simulationsmodellen zwischen verschiedenen Entwicklergruppen möglich ist. Weiterhin wurden echtzeitfähige Algorithmen für die Mehrfachtexturierung entwickelt, die den Realitätsgrad der computergraphischen Darstellungen erhöht. Als dritter Themenkomplex wurde die Deformationssimulation mit der *Fast-Finite-Element*-Methode eingehend untersucht. Hierbei wurde ausgehend von einer rudimentären Implementierung eine optimierte Berechnung und objektorientierte Aufteilung der Steifigkeitsmatrix erreicht, wodurch auch hybride Simulationsmodelle mit Interaktionsmöglichkeiten für das VEST-System zur Verfügung stehen.

Weitere Forschungsarbeit hinsichtlich Optimierung und Anpassung bestehender Methoden ist in allen drei Themenbereichen sinnvoll. Im Rahmen der Modellierung ist die Möglichkeit der Texturierung von impliziten Oberflächen zu untersuchen, wobei insbesondere die prozedurale Erstellung von 3D-Texturen sowie die Entwicklung von geeigneten Texturabbildungsmethoden von großer Bedeutung ist. Die bei der Modifikation von Dreiecks-modellen mit impliziten Oberflächen entstehenden *Aliasing*-Effekte wurden mit einem 3D-Glättungsfilter weitgehend reduziert. Weitere Methoden sind in [Jon96] [SK99] zu finden, die mit unserem Verfahren qualitativ verglichen werden sollten. Einige Simulationsmethoden des Chirurgiesimulators wie etwa die Pulssimulation oder bestimmte Partikelsimulationen setzen historisch bedingt eine vorgegebene Anordnung der Objektknoten voraus, um Kollisionsbehandlungen zu optimieren. Im Zuge der Verwendung beliebiger Dreiecksnetze müssen diese bestehenden Algorithmen nun überarbeitet werden. Die Interaktionssimulation deformierbarer Objekte mit der FFE-Methode bei der eine Topologieänderung auftritt, muss eingehend untersucht werden. Wichtige Fragestellung hierbei ist die echtzeitfähige Neuberechnung der Steifigkeitsmatrizen. Eine Lösung ist die Parallelisierung der

Berechnung, bei der jedoch die echtzeitfähige Lastverteilung auf Rechner in einem Rechnerverbund und die Zusammenführung der Teilergebnisse eine große Herausforderung darstellt.

6. Literatur

[ABG04] Allegre R., Barbier A., Galin E., Akkouche S.: “A Hybrid Shape Representation for Free-form Modeling", International Conference on Shape Modeling and Applications, SMI 2004, pp. 7-18, 2004

[Bro98] Bro-Nielsen M.: “Finite element modeling in medical VR”, Journal of the IEEE 86(3):490-503, 1998

[CMK05] Çakmak H. Maass H., Kühnapfel U.: “VS*One* - a virtual reality simulator for laparoscopic surgery”, Official journal of ‘The Society for Medical Innovation and Technology’, Minimally Invasive Therapy and Allied Technologies (MITAT) 14:3; 134-144, ISSN 1365-2931, 2005

[CMS02] Çakmak H. K., Maass H., Strauss G., Trantakis C., Nowatius E., Kühnapfel U.: “Modellierung chirurgischer Simulationsszenarien für das Virtuelle Endoskopie Trainingssystem (VEST)”, CURAC 2002, 1. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V. (http://curac.org/journal_curac2002.htm) , Leipzig/Germany 4.-5.10.2002

[FOX] http://www.fox-toolkit.org

[Jon96] Jones M.: “The Production of Volume Data from Triangular Meshes Using Voxelisation”, Computer Graphics Forum, Vol. 15, no. 5 pp. 311-318, 1996

[KÇ98] Kühnapfel U., Çakmak H.: *Unveröffentlichter Bericht*, Institut für Angewandte Informatik, Forschungszentrum Karlsruhe, 1998

[KPT99] Karabassi E.-A., Papaioannou G., Theoharis T.: “A fast depth-buffer-based voxelization algorithm”, Journal of Graphics Tools 4(4), pp.5-10, 1999

[LC87] Lorensen W., Cline H.: “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, Computer Graphics (Proceedings of SIGGRAPH '87), Vol. 21, No. 4, pp. 163-169, 1987

[Len03] E. Lengyel: “Mathematics for 3D Game Programming & Computer Graphics”, Charles River Media, ISBN: 1584502770, 2003

[MK99] Maaß, H., Kühnapfel, U.: “Noninvasive Measurement of Elastic Properties of Living Tissue”, Computer Assisted Radiology and Surgery (CARS '99), pp 865-870, Elsevier Science, 1999

[Nvi] www.nvidia.com

[SJ03] Schijven M, Jakimowicz J.: “Virtual Reality Surgical Laparoscopic Simulators”, Surgical Endoscopy, 17(12): 1978-1984, ISSN 0930-2794, Springer Verlag, 2003

[SK99] Sramek M., Kaufman A. “Alias-free voxelization of geometric objects“, IEEE Transactions on Visualization and Computer Graphics 3(5), pp. 251-266, 1999

[WMW86] Wyvill G., McPheeters C., Wyvill B.: “Data Structure for Soft Objects”, The Visual Computer, vol.2, no.4, pp. 227-234, 1986