Simulation einer Schädeltrepanation mit Hilfe der Java3D-Technologie

Kai Annacker*♯, Hans-Gerd Lipinski*, Dietrich H. W. Grönemeyer♯

*Abt. Medizinische Informatik
Fachhochschule Dortmund, 44139 Dortmund
♯Institut für MikroTherapie
Universität Witten/Herdecke, 44799 Bochum
Email: annacker@microtherapy.de

Zusammenfassung. Die Trepanation der menschlichen Schädelkalotte, wie sie im Rahmen eines stereotaktischen Eingriffs notwendig ist, wird simuliert. Dazu werden verschiedene anatomische Strukturen (Haut, Muskeln, Nerven, Knochen) segmentiert und dreidimensional rekonstruiert. Im Rahmen der Simulation werden Gewebeareale interaktiv durchbohrt, wobei der Operateur mittels eines Force-Feedback-Joysticks eine spürbare Rückmeldung über den Status des virtuellen Bohrers erhält. Unter Nutzung der modernen Java 3D Technologie und entsprechender Graphik-Hardware wird das 3D-Modell in Echtzeit dem Stand der Trepanation angepaßt.

1 Einleitung

Um einen stereotaktischen Eingriff in das menschliche Gehirn durchführen zu können, ist eine Öffnung der Schädelkalotte notwendig. Dabei findet zumeist das Verfahren der *osteoplastischen Trepanation* Anwendung, bei der der ausgesägte Knochendeckel nach Abschluß des eigentlichen diagnostischen bzw. therapeutischen Eingriffs replantiert wird. Durch das erzeugte Bohrloch kann dann eine Sonde, welche an einem am Kopf des Patienten montierten Führungsrahmen befestigt ist, entlang eines zu definierenden Trajektes auf einen Zielpunkt innerhalb des Gehirns zubewegt werden. An diesem Zielpunkt erfolgt dann die diagnostische oder therapeutische Maßnahme, z. B. eine Biopsie, die Einsetzung einer Stimulationselektrode oder eine Koagulation des umliegenden Nervengewebes.

Im Zuge der Entwicklung eines stereotaktischen Operations-Simulators wurde in einem ersten Teilprojekt das virtuelle Trepanieren einer Schädelkalotte eingeführt. Für den Operateur sind dabei auf der einen Seite eine komfortable, interaktive Festlegung der Lage des Trajektes, und damit auch des Bohrloches, anhand präoperativ angefertigter Computertomogramme, auf der anderen Seite aber auch die differenzierte Steuerung des (virtuellen) Bohrwerkzeuges von Interesse. Es wurde ein System entwickelt, das diese Forderungen aufgreift und in Form einer Kombination von Soft- und Hardware implementiert.

2 Methoden und Ergebnisse

2.1 Basisdaten, Soft- und Hardware

Als Grundlage für die eingesetzten Segmentierungs- und Rekonstruktionsverfahren wurden kraniale Computertomogramme mit einer Auflösung von 256×256 Bildpunkten (horizontaler und vertikaler Pixelabstand: 1mm) und einem Schichtabstand von zwei bis vier Millimetern verwendet.

Die Software wurde für ein 800MHz-Dual-Prozessor-System inklusive Hochleistungs-3D-Graphikkarte auf Intel-/Windows-Basis in der system- und plattformunabhängigen Programmiersprache Java 2 der Firma Sun Microsystems in der Version 1.3 entwickelt. Java 2 wurde trotz geringfügiger Performance-Nachteile gegenüber Sprachen wie C oder Pascal gewählt, da es die Perspektive bietet, den Simulator auch auf anderen Computersystemen, für die ebenso eine *Java Virtual Machine*, die notwendige Laufzeitumgebung, erhältlich ist bzw. sein wird, einsetzen zu können [1, 2]. Ferner bietet die fortschrittliche Java 3D Technologie über die OpenGL-Schnittstelle der Graphikkarte Zugriff auf deren 3D-Fähigkeiten [3, 4].

2.2 Segmentierung und dreidimensionale Rekonstruktion

In einem ersten Schritt werden die zu rekonstruierenden Strukturen des Patientenschädels anhand ihrer Dichte, d. h. anhand des durch sie erzeugten Grauwertes innerhalb des Voxelquaders, interaktiv definiert. Dabei können nicht miteinander verbundene Strukturen auch getrennt segmentiert werden.

Die Oberfläche jedes einzelnen der so definierten Objekte wird nun mit Hilfe des *Marching-Cube-Algorithmus* trianguliert [5]. Hierzu wird der gesamte Voxelquader auf Übergänge von Objekt zu Umgebung zwischen zwei benachbarten Voxeln durchsucht. Durch diese detektierten Übergänge werden die zu konstruierenden Grenzflächen definiert, die sich jeweils nur aus bis zu fünf Dreiecken zusammensetzen, in ihrer Gesamtheit jedoch schließlich die vollständige Objektoberfläche bilden. Zur besseren Orientierung werden die einzelnen, den unterschiedlichen Gewebestrukturen (z. B. Haut, Muskeln, Nerven, Knochen) entsprechenden Objekte verschiedenfarbig dargestellt.

In einem weiteren Schritt wird versucht, die hohe Anzahl erzeugter Dreiecke durch Betrachtung ihrer Größen- und Lageverhältnisse zu reduzieren [6]. Dies erscheint notwendig, da auch modernste 3D-Graphik-Hardware beim Umgang mit derart vielen Polygonen an Grenzen stößt, was sich in sinkenden Frameraten und damit verlängerten Reaktionszeiten bemerkbar macht. Dabei verschlechtert sich der visuelle Eindruck des dreidimensionalen Oberflächenmodells bis zu einer Reduktion um etwa 50% der Dreiecke nicht, sondern gewinnt eher noch an Qualität. Hinzu kommt, daß dieses 3D-Modell schneller visualisiert wird, und sowohl Rotation als auch Zoom ohne relevante Wartezeiten bewältigt werden können. Dies ermöglicht, auch intraoperativ ohne Verzögerung dem virtuellen Operateur eine gewünschte Ansicht des Schädels und des Bohrareals zu liefern.

2.3 Trajekt und Simulation der Trepanation

Um während der anschließenden Simulation nicht für jeden Zeitpunkt das gesamte dreidimensionale Modell erneut berechnen zu müssen, wird der Voxelquader, und damit auch die zu rekonstruierende Oberfläche, in zwei unterschiedlich zu behandelnde Volumen geteilt: ein sehr großes, das sich während des Bohrvorganges nicht verändern wird, und ein deutlich kleineres, das durch die Bohrung potentiell entfernt wird. Hierzu werden zuvor Lage und Winkel des Bohrtrajektes, speziell der Zielpunkt innerhalb des virtuellen Schädels, interaktiv festgelegt.

Da der simulierte Vortrieb des Bohrwerkzeuges nur Auswirkungen auf das kleinere der beiden Volumen hat, kann der größte, *statische* Teil des Modells einmalig vor Beginn der Simulation dreidimensional rekonstruiert werden. Nur die Oberfläche des kleineren, *dynamischen* Anteils muß bei Vortrieb des Bohrers — reduziert um das an- oder durchbohrte Gewebeareal — erneut berechnet werden. Dadurch gelingt es nahezu verzögerungsfrei, einen visuellen Eindruck des Bohrergebnisses zu vermitteln, zumal während der Simulation das sich gerade entwickelnde Bohrloch eingesehen werden kann, ohne den Bohrer selbst zurückführen zu müssen.

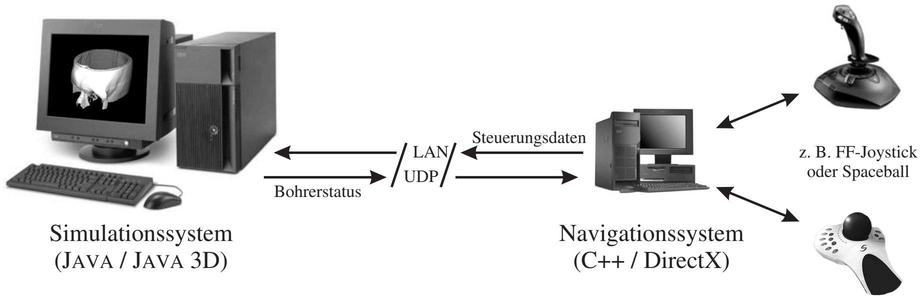
2.4 Navigation und Rückkopplung

Die Segmentierung, Festlegung des Trajektes und die dreidimensionale Ansicht werden per Tastatur und Standardmaus gesteuert; auch die Simulation läßt sich auf diese Weise durchführen. Um dem virtuellen Operateur jedoch ein realitätsnahes Bohren zu ermöglichen, kann ein *Force-Feedback-Joystick* benutzt werden, der mit Hilfe eingebauter Motoren softwareseitig steuerbare Widerstände und Bewegungen gegen Auslenkungen erzeugen kann. So wirkt der Joystick in Abhängigkeit von den zu durchdringenden Strukturen dem Vortrieb entgegen und meldet die aktuelle Drehzahl des virtuellen Bohrers durch angepaßte Vibration zurück.

Da derzeit keine direkte Joystick-Steuerung in JAVA möglich ist, wird zur kontinuierlichen Registrierung der Joystickhaltung und gleichzeitigen Steuerung der Force-Feedback-Effekte auf die *Microsoft DirectX* Technologie (insbesondere das DirectInput API) zurückgegriffen, die fester Bestandteil aktueller Microsoft Windows Systeme ist [7]. Zu diesem Zweck empfahl sich die Verwendung der Entwicklungsumgebung *Microsoft Visual C++*, die bereits auf die Nutzung des DirectX API vorbereitet ist.

Um Informationen zwischen der Joystick-Steuerung und dem Simulator auszutauschen, werden *Datagramme* per *User Datagram Protocol* (UDP) über ein lokales Netzwerk (local area network, LAN) versendet [8]. Wie in Abbildung 1 dargestellt sendet das Simulationssystem speziell die für die Force-Feedback-Steuerung notwendigen Daten, während gleichzeitig das Navigationssystem Informationen über den aktuellen Status des Eingabegeräts übermittelt. Diese lose Kopplung — UDP ist ein verbindungsloses Netzwerkprotokoll — der beiden Programmteile hat den Vorteil, daß durch die systemabhängige Joystick-Steuerung

Abb. 1. Schematische Darstellung der Funktionsweise des Operations-Simulators.



nicht die Plattformunabhängigkeit des Simulators eingeschränkt wird. Ferner wird die Simulations-Workstation nicht durch das Joystick-Programm zusätzlich belastet, auch wenn es möglich wäre, beide Programme auf einem einzigen Rechner zu betreiben. Der Nachteil von UDP, keine feste Verbindung zu etablieren, wird dadurch kompensiert, daß die Datagramme, also die Informationspakete, numeriert und mit Prüfsummen abgesichert werden. So kann sichergestellt werden, daß fehlende oder veränderte Pakete erkannt werden.

3 Diskussion

Die virtuelle Schädeltrepanation als Teil einer Operations-Simulation wurde entwickelt, um einem virtuellen Operateur die Möglichkeit zu geben, anhand von Patienten-CTs ein Trajekt festzulegen, diesem folgend eine Bohrung durchzuführen und sie visuell zu beurteilen. Dazu werden verschiedene anatomische Strukturen farblich differenziert dargestellt und bei Vortrieb eines virtuellen Bohrwerkzeuges in Echtzeit um durchbohrte Strukturen reduziert.

Die Steuerung der Simulation mit Hilfe eines bislang vornehmlich in der PC-Spiele-Szene anzutreffenden Force-Feedback-Joysticks bietet sich durch die gelungene Kombination von Steuermöglichkeit und Bohr-Feedback an. Verzichtet man beim virtuellen Bohrvorgang auf diese Art der Rückkopplung, ist auch der Einsatz anderer Eingabegeräte möglich, die die Microsoft DirectInput Technologie unterstützen, z. B. eines *Spaceballs*, einer Art Trackball mit sechs Freiheitsgraden.

Probleme bestehen derzeit noch bei der Echtzeit-Simulation des Oberflächen-Modells während der Trepanation auf nicht hinreichend performanten (preiswerteren) PC-Systemen, also bei für die jeweilige Graphik-Hardware zu großer Anzahl darzustellender Polygone. Im Zuge stetig zunehmender Leistungsfähigkeit von Standard-PCs und weiterer Optimierung der Software ist jedoch zu erwarten, daß das Simulations-System zukünftig auch hier sinnvoll einsetzbar sein wird.

Danksagung

Diese Arbeit wurde mit Mitteln des Ministeriums für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen (Assistentenprogramm NRW) und der Fachhochschule Dortmund (Rektorat II / Forschung und Entwicklung) gefördert.

Literatur

1. Sun Microsystems Inc.: The JAVA Technology, http://java.sun.com/.
2. Sun Microsystems Inc.: The JAVA 2 Standard Edition 1.3, http://java.sun.com/j2se/1.3/.
3. Sun Microsystems Inc.: The JAVA 3D API, http://java.sun.com/products/java-media/3D/.
4. Silicon Graphics Inc.: OpenGL — The Industry's Foundation for High Performance Graphics, http://www.opengl.org/.
5. Bourke, Paul: Polygonizing a scalar field, 1997, http://www.swin.edu.au/astronomy/pbourke/modelling/polygonise/.
6. Bourke, Paul: Surface (polygonal) simplification, 1997, http://www.swin.edu.au/astronomy/pbourke/modelling/surfsimp/
7. Microsoft Corporation: DirectX 8.0, http://www.microsoft.com/directx/.
8. Internet RFC/STD/FYI/BCP Archives: Request For Comments 768, The User Datagram Protocol (UDP), http://www.faqs.org/rfcs/rfc768.html.