Die Integration des Heidelberger Raytracers in OpenGL zur integrierten Segmentierung und Visualisierung dreidimensionaler medizinischer Volumendaten

K. Wolsiffer, A. Mayer, R. Niebsch, H. Evers, H.P. Meinzer

Deutsches Krebsforschungszentrum Heidelberg, Abteilung Medizinische und Biologische Informatik, Im Neuenheimer Feld 280, 69120 Heidelberg
K.Wolsiffer@DKFZ-Heidelberg.de

In der medizinischen Bildverarbeitung ist die Segmentierung von Volumendatensätzen nur in wenigen Fällen automatisch möglich. Daher muß häufig auf interaktive Segmentierungsalgorithmen und -methoden zurückgegriffen werden, welche je nach verwendetem Verfahren mehr oder weniger einer Steuerung durch den Benutzer bedürfen. Da es sich bei medizinischen Volumendaten um sehr speicherintensive Daten handelt, die meist als eine Serie von Schichtbildern vorliegen, ist die interaktive Segmentierung ein recht zeitaufwendiger Prozeß. Im Folgenden wird ein System beschrieben, mit dem der interaktive Segmenterierungsvorgang durch Integration von Volumenvisualisierungs- und Segmentierungsmethoden sowie durch die Ermöglichung einer Interaktion direkt im 3D-Raum erheblich verkürzt und vereinfacht wird.

Keywords: Interaktive 3D-Segmentierung, Heidelberger Raytracer, 3D-I/O-Devices, OpenGL

1 Einleitung

Die Computertomographie und die Magnetresonanztomographie erlangen als bildgebende Verfahren in der Medizin eine ständig wachsende Bedeutung. Aufgrund der dadurch bedingten Daten- und Informationsflut wächst gleichzeitig der Bedarf an rechnerunterstützter Bildverarbeitung. Dabei kommt insbesondere den Segmentierungsmethoden eine wichtige Bedeutung zu, da mit ihrer Hilfe eine Filterung und Verdichtung der Bildinformation erreicht werden kann. Die Segmentierung medizinischer Volumendaten ist jedoch nur in wenigen Fällen voll automatisch möglich. Meistens kann die Trennung des Zielvolumens vom umliegenden Gewebe lediglich durch einen *interaktiven* Segmentierungsprozeß realisiert werden, welcher im allgemeinen einen iterativen Charakter besitzt:

Zunächst wird ein Segmentierungsverfahren oder eine Gruppe von Verfahren parametrisiert und ausgeführt und anschließend das Ergebnis überprüft. Wurde das Zielvolumen nicht vollständig in der segmentierten Region eingeschlossen, oder enthält es noch Teile des umliegenden Gewebes, so wird entweder dasselbe Verfahren mit veränderten Parametern ein zweites Mal durchgeführt oder das Ergebnis mit einem anderen Segmentierungsalgorithmus weiterverarbeitet.

Bei Volumendaten, insbesondere im medizinischen Bereich, ist dieser Prozeß aufgrund der großen Datenmengen sehr zeitaufwendig. Dies liegt daran, daß man sich sowohl bei der Parametrierung der Verfahren als auch bei der anschließenden Ergebniskontrolle Schicht für Schicht durch das Volumen arbeiten muß. Eine schnellere und intuitivere Beurteilung schichtübergreifender Strukturen wird durch eine 3D-Darstellung des Ergebnisses ermöglicht.
Ziel beim Entwurf des vorliegenden Systems war die Beschleunigung des interaktiven Segmentierungsprozesses durch Schaffung einer *integrierten Segmentierungsumgebung*, in der gleichzeitig sowohl die Parametrierung der Algorithmen und die Echtzeit-Navigation im 3D-Raum als auch die Ergebniskontrolle im volumenvisualisierten Bild stattfinden können.
Der *Heidelberger Raytracer* [1] erlaubt ein volumenbasiertes Rendering von 3D-Datensätzen. Aufgrund der hohen Rechenintensität können jedoch noch keine interaktiven Frameraten erzielt werden. Um dennoch die Navigation und Interaktion im 3D-Raum zu ermöglichen, wurde der Heidelberger Raytracer in *OpenGL* integriert. Der Heidelberger Raytracer erzeugt ein statisches Bild des Volumens, während die Navigation und Interaktion im 3D-Raum durch die Bereitstellung von beweglichen Hilfselementen, wie beispielsweise einer Schnittebene oder eines Pointers, ermöglicht wird. Diese werden in *OpenGL* als Polygonmodelle definiert, in *Echtzeit* visualisiert und anschließend dem statischen Bild aus dem Raytracer überlagert. Die Überlagerung wird dabei so durchgeführt, daß der Eindruck ensteht, die Hilfselemente würden sich tatsächlich durch das Volumen bewegen, d.h. daß sie auch Teile des Volumens überdecken bzw. selbst von Volumenteilen überdeckt werden können.

2 Methoden

2.1 Der Bildverarbeitungssserver

In der Abteilung für Medizinische und Biologische Informatik (MBI) des Deutschen Krebsforschungszentrums in Heidelberg wurde eine Client/Server-Umgebung für (parallele) medizinische Bildverarbeitung geschaffen [2]. Auf dem Server werden sowohl Funktionen für die Segmentierung als auch für die Volumenvisualisierung mit dem Heidelberger Raytracer zur Verfügung gestellt. Das Segmentierungssystem erfüllt die Rolle des Clients und dient als „Viewing-Station" zur Darstellung, Steuerung und Interaktion. Durch die so erzielte Trennung der Interaktion und der visuellen Kontrolle von der Implementierung der Algorithmen wird eine größtmögliche Flexibilität und Erweiterbarkeit des Segmentierungssystems gewährleistet.
Der Bildverarbeitungsserver arbeitet nach einem *Session-Konzept*: Jeder Client baut zu Beginn der Kommunikation eine logische Session auf, deren Datenbereich von allen anderen Clients isoliert ist und innerhalb derer die gesamte nachfolgende Kommunikation abgewickelt wird. Um die Übertragung der Volumendaten von der eigentlichen Verarbeitung zu trennen, lädt der Client das zu bearbeitende Volumen nur einmal auf den Server. Dieser generiert ein abstraktes *„Handle"* und liefert es an den Client zurück, welcher das Handle bei zukünftigen Operationen zur Identifikation der Daten benutzt. Auf diese Art wird die wiederholte Übertragung von Volumendaten über das Netz vermieden und der Datenverkehr über das Netz wesentlich reduziert, so daß zufriedenstellende Antwortzeiten erzielt werden können.

2.2 Volumenrendering mit dem Heidelberger Raytracer

Zur Evaluation des Segmentierungsprozesses ist eine Visualisierung des Ergebnis- und des Originalbildes notwendig. Die einfachste Möglichkeit zur Ergebniskontrolle bietet hierbei die Ansicht der 2D-Schichten. Zur effizienten Beurteilung dreidimensionaler Strukturen sollte jedoch zusätzlich eine 3D-Renderingkomponente zur Verfügung stehen. Hierfür gibt es grundsätzlich zwei Lösungsansätze: Volumenrendering und Oberflächenrendering [3]. Beim Volumenrendering werden Lichtstrahlen durch das Volumen verfolgt und das Bild durch Berechnung von Intensitätswerten entlang der Strahlen erzeugt. Beim Oberflächenrendering wird zunächst eine Oberflächenrepräsentation für das zu betrachtende Objekt erstellt und diese anschließend mit einer virtuellen Lichtquelle beleuchtet.
In medizinischen Anwendungen wird der volumenbasierte Renderingansatz aufgrund seiner größeren Detailtreue favorisiert, obwohl er einen höheren Rechenaufwand erfordert.

Moderne Grafikworkstations verfügen über leistungsfähige Texture-Engines, mit welchen eine sehr schnelle Berechnung eines Volumenrenderings möglich ist. Einen Engpaß bildet hier jedoch die Übertragung des Volumens in den Texturspeicher, ein Vorgang, der bei einem interaktiven Segmentierungssystem häufig durchgeführt werden muß. Außerdem erlaubt diese Technik keine Definition von Lichtquellen und erfordert somit den Verzicht auf einen Schattenwurf. Aus diesen Gründen wurde für das Volumenrendering der Heidelberger Raytracer verwendet, der an das von Kajiya [4] vorgeschlagene Verfahren angelehnt ist und die Definition zweier Lichtquellen in den Winkeln 0 und 45 Grad erlaubt [1]. Diese Wahl bietet gegenüber den Texturing-Ansätzen außerdem den Vorteil, daß das System auch auf Workstations mit relativ wenig Grafikhardware-Unterstützung lauffähig ist.

Das *Viewing-Modell* des Heidelberger Raytracers erlaubt die freie Navigation des Beobachters im 3D-Raum und ermöglicht so alle sechs Freiheitsgrade. Der Betrachter kann seine Position im Raum beliebig verändern. Er kann sozusagen durch den Raum „fliegen“ und dabei in jede Richtung blicken. Das betrachtete Volumen bleibt hingegen fest in einem globalen Koordinatensystem verankert. Desweiteren ist die Definition einer perspektivischen Projektion möglich, so daß Objekte, die vom Betrachter weiter entfernt liegen, kleiner erscheinen als Objekte, die sich näher beim Betrachter befinden.

2.3 Interaktive Visualisierung und Navigation

Da mit dem Heidelberger Raytracer keine interaktiven Frameraten erreicht werden können, werden zur Navigation im 3D-Raum *grafische Hilfselemente*, die mit OpenGL in Echtzeit gerendert werden, in das mit dem Heidelberger Raytracer visualisierte Volumen eingeblendet. Die Orientierung des Volumens wird mit Hilfe eines um das Volumen gelegten Quaders realisiert, dessen Frontseite zur besseren Orientierung in roter Farbe hervorgehoben ist. Die Orientierung dieses Quaders kann nun interaktiv verändert werden. Befindet sich der Quader in der gewünschten Position, wird ein neuer Rendervorgang angestoßen, so daß die neue Aus-richtung des Volumens mit der des Quaders übereinstimmt. Mit Hilfe einer verschiebbaren Schnittebene kann die Schicht, die im Schnittbildbereich erscheinen soll, ausgewählt werden. Auf diese Art wird gleichzeitig ein Bezug zwischen Schichtbild- und 3D-Darstellung hergestellt. Die Drehung des Quaders kann entweder mit dem Spacemaster, einem 3D-Eingabegerät mit sechs Freiheitsgraden, oder mit der Maus nach dem Prinzip einer Virtual-Trackball-Steuerung erfolgen.

2.4 OpenGL

OpenGL ist ein Client/Server-fähiges und Hardware-unabhängiges Softwareinterface für die Grafikhardware. Um die Hardware-Unabhängigkeit zu gewährleisten, stellt OpenGL keine High-Level-Befehle zur Beschreibung dreidimensionaler Objektmodelle zur Verfügung. Stattdessen müssen die Objekte aus einer kleinen Menge geometrischer Primitiven aufgebaut werden. Zur Visualisierung werden die Objekt-Koordinaten der Primitiven durch die OpenGL Viewing-Pipeline transformiert und so in Window-Koordinaten umgerechnet. Intern werden diese Transformationen durch die Multiplikation homogener Vektoren mit Transformationsmatrizen realisiert. Die Gesamtheit aller Transformationen wird als *Viewing-Pipeline* bezeichnet. Diese besteht aus insgesamt fünf Stationen: Zunächst werden die Objektkoordinaten durch die *Modelview-Transformation* in Eye-Koordinaten umgerechnet. Diese Transformation entspricht dem Positionieren der Objekte relativ zum Viewpoint. Anschließend werden Objektteile, die durch optional definierte *Clipping-Planes* aus dem Viewing-Volumen herausfallen, entfernt. Im nächsten Schritt wird durch Anwendung einer *Projektionsmatrix* eine perspektivische oder eine orthogonale Projektion erstellt. Bei der perspektivischen Projektion hat das Viewing-Volumen die Form eines Pyramidenstumpfes, so daß weiter entfernt liegende Objekte kleiner erscheinen. Die aus der Projektionstransformation resultierenden Clipping-Koordinaten werden durch eine *perspektivische Division* in normalisierte Device-Koordinaten umgerechnet. Schließlich erhält man nach Durchführung der *Viewport-Transformation*, welche die Koordinaten auf einen bestimmten Viewport abbildet, die Window-Koordinaten. Diese werden schließlich im OpenGL-Framebuffer gespeichert [5].

2.5 Integration des Heidelberger Raytracers in OpenGL

Die Integration des Raytracers in OpenGL erfogt nach einem *Z-Merging-Algorithmus* unter Ausnutzung des OpenGL Tiefenpuffers [5]. In diesem können die Tiefen- oder Z-Werte gespeichert werden, welche die Distanz eines im Framebuffer abgebildeten Objektpunktes vom Betrachter angeben. Ein Pixelwert kann einen bereits im Framebuffer befindlichen Pixelwert nur dann überschreiben, wenn er den sogenannten „*Depth-Test*“ besteht, d.h. wenn sein Z-Wert kleiner als der des bereits gespeicherten Pixelwertes ist.

Der Raytracer liefert nach jedem Rendervorgang jeweils ein Grauwert- und ein Tiefenbild. Diese werden zunächst in den Framebuffer kopiert und anschließend mit den von OpenGL gerenderten Hilfselementen nach dem Z-Merging-Algorithmus überlagert. Dadurch entsteht der Eindruck, daß sich die Hilfselemente und das Volumen tatsächlich in demselben Viewing-Volumen befinden. Werden die Hilfselemente bewegt, so müssen zunächst der Tiefenpuffer und der Farbpuffer restauriert und anschließend die Hilfselemente in der neuen Position gerendert werden.

Zur Realisierung dieses Ansatzes muß das Viewing-Modell von OpenGL auf das des Raytracers abgebildet und das Viewing-Volumen so initialisert werden, daß sich sämtliche Objekte im Innern des Viewing-Volumens befinden und mit einer adäquaten Auflösung und Größe angezeigt werden.

Vor dem Starten eines neuen Raytracing-Vorgangs muß die Gesamt-Rotationsmatrix, welche die Summe der Rotationen des das Volumen umgebenden Quaders beschreibt, in drei Winkel *alpha, beta* und *gamma* zerlegt werden. Diese Winkel geben die Rotationsrichtung um die X- Y- und Z-Achse an und werden dem Raytracer als Parameter übergeben. Sie müssen das Volumen aus der Ausgangsposition in die durch die Gesamtrotationsmatrix beschriebene Orientierung versetzen.

Außerdem müssen die vom Raytracer gelieferten Z-Koordinaten, die den Eye-Koordinaten in der OpenGL-Viewing-Pipeline entsprechen, transformiert werden, so daß sie zu den im OpenGL-Tiefenpuffer gespeicherten Z-Werten „passen". Sei *Zwin* die im Tiefenpuffer gespeicherte Z-Koordinate und *f, n* die vordere und hintere Grenze eines Viewing-Volumens bei einer perspektivischen Projektion, dann erhält man für die Z-Koordinate *Zwin:*

$$Zwin = 0{,}5 + \frac{f+n}{2 \cdot (n-f)} + \frac{f \cdot n}{Zray \cdot (f-n)} \tag{1}$$

3 Ergebnisse und Diskussion

Bei dem vorliegenden System wird die Segmentierung und die Visualisierung medizinischer Volumendaten in *einer* grafischen Benutzerschnittstelle integriert (Abb. [3.1]). Sowohl die Parametrierung der Verfahren als auch die Ergebniskontrolle kann nun direkt im 3D-Raum erfolgen, was eine erhebliche Vereinfachung und Verkürzung des Segmentierungsprozesses bewirkt. Soll beispielsweise ein Region-Growing durchgeführt werden, wird zunächst durch Verschiebung der Schnittebene im 3D-Raum eine Schicht ausgewählt und auf dieser eine rechteckige oder kreisförmige Saatregion, sowie weitere Region-Growing-Parameter definiert. Nach Beendigung des Segmentierungsprozesses kann das Ergebnis anhand der Schichten kontrolliert werden. Es ist jedoch auch möglich „auf Knopfdruck" ein 3D-Rendering des Ergebnisvolumens zu erzeugen, wodurch die Ergebniskontrolle insbesondere bei der Beurteilung schichtübergreifender Strukturen erleichtert wird. Zur Steuerung der Navigation stehen ein Spacemaster, als ein „echtes" 3D-Eingabegerät, und eine Virtual-Trackball-Steuerung mit der Maus zur Verfügung. Um den Tiefeneindruck bei der Interaktion im 3D-Raum zu verstärken, können auch Stereo-Views erzeugt werden.

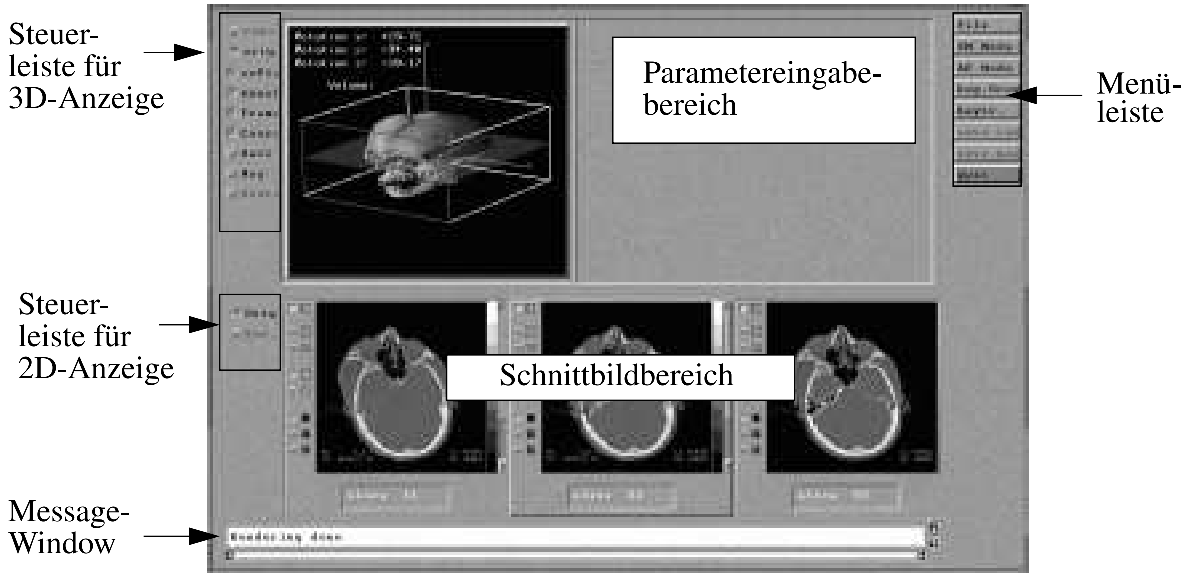

Abbildung 3.1: Überblick über die grafische Benutzerschnittstelle

Die Integration von Volumenvisualisierung und -segmentierung und die Interaktionsmöglichkeiten direkt im 3D-Raum eröffnen Perspektiven, die zur Erforschung und Entwicklung weiterer interaktiver Segmentiermethoden und neuer Techniken zur Mensch-Maschine-Interaktion ermutigen. Für das vorliegende System ist die Integration und Evaluation sowohl neuer als auch bereits bestehender interaktiver Segmentierverfahren und Interaktionsmethoden, beispielsweise unter Ausnutzung von Force-Feedback-Devices, geplant.

Literatur

[1] Meinzer H.P., Meetz K., Scheppelmann D., Engelmann U., Baur H.J.: „The Heidelberg Ray Tracing Model“ IEEE Computer Graphics and Applications, pp. 34-43, November 1991

[2] A. Mayer, R. Niebsch, B. Kochner, K. Wolsiffer, M.H. Makabe, U. Baur, A. Schröter, H.P. Meinzer: „SIPS: Scalable Image Processing Server. Ein paralleler Segmentierungs- und Visualisierungsserver“, Tagungsband 4. Freiburger Workshop f. Digitale Bildverarbeitung in der Medizin, March 1996, pp. 114-119, 1996.

[3] David G., Heath, PhD, P.A. Soyer, MD, B.S. Kuszyk, MD, D.F. Bliss, MA, P.S. Calhoun, BFA, D.A.Bluemke, MD, PhD, M.A. Choti, MD, E.K. Fishman, MD: “Three-dimensional Spiral CT during Arterial Portography: Comparison of three Rendering Techniques”, Radiographics, Vol. 15, Iss 4, pp. 1001-1011, 1995.

[4] Kajiya, James, Von Herzen, Brian: “Ray Tracing Volume Densities”, ACM SIGGRAPH Computing Group, Vol 18, No. 4, pp. 165-174, July 1984.

[5] Jackie Neider, Tom Davis, Mason Woo: “OpenGL Programming Guide, The Official Guide to Learning OpenGL”, Release 1, Addison Wesley, 1993.