Ernst-Mach-Gymnasium Haar Qualifikantenjahrgang 2010/2011

Seminararbeit im Leitfach Physik

3D-Tumorvisualisation

Verfasser: Ulrich Köhler
W-Seminar: Medizinische Bildgebungsverfahren
Kursleiter: Ernst Bartels

	schriftlich, einfache Wertung	Präsentation
Erzielte Punkte		

Datum der Abgabe im Kollegstufensekretariat Unterschrift des Kursleiters

Inhaltsverzeichnis

1. Einleitung

In dieser Arbeit sollen Methoden und Algorithmen diskutiert und verglichen werden, die zur dreidimensionalen Visualisierung von Tumoren in Echtzeit dienen. Hierzu werden zuerst in Kapitel 2.1 nach einer Definition dieses Begriffs die allgemeinen Anforderungen sowie in Kapitel 2.2 auf Seite 5 Anwendungsmöglichkeiten für solche Systeme dargestellt. Darauf aufbauend werden in Kapitel 3.1 auf Seite 6 reine Softwarealgorithmen[1] mit auf spezieller Hardware basierenden Algorithmen verglichen. Kapitel 3.2 auf Seite 7 stellt kurz die in dieser Arbeit beschriebenen Volumenrenderingverfahren[2] Oberflächenrenderingalgorithmen[2] gegenüber. Danach zeigt Kapitel 3.3 auf Seite 8 einen Überblick der verfügbaren Plattformen zur Implementierung von Visualisationsalgorithmen. Kapitel 3.4 auf Seite 11 führt einen beispielhaften Algorithmus zur Hervorhebung von Tumoren sowie dessen im Rahmen dieser Arbeit entstandene Implementation („VERTEBRA[3]“) ein.

Anschließend wird in Kapitel 4 auf Seite 16 das Konzept der „Augmented Reality“ diskutiert, das bereits jetzt für viele medizinische Visualisationssysteme eingesetzt wird. Mit ARION™ wird in Kapitel 5 auf Seite 18 ein Beispiel für ein solches System diskutiert. Mit dem Fazit und einem kurzen Ausblick in Kapitel 6 auf Seite 19, das in Kurzform die wichtigsten Thesen, Konzepte und Resultate rekapituliert, endet der Hauptteil dieser Arbeit.

2. Echtzeit-Tumorvisualisationssysteme

2.1. Anforderungen

Um die Anforderungen für Echtzeit-Tumorvisualisationssysteme zu erläutern, muss der Begriff zuerst mit allen zu erfüllenden Anforderungen definiert werden. Die folgende Definition basiert auf [Bru04, Kapitel 3.1.1, Seite 17] sowie auf [KAB+08]:

Definition 1 (*Echtzeit-Tumorvisualisationssystem*)

Als Echtzeit-Tumorvisualisationssystem sei ein System definiert, das in Echtzeit in der Lage ist, aus dreidimensionalen medizinischen Eingabedatensätzen hochqualitative Ausgabedaten zu generieren, die auf Displays in einer Weise dargestellt werden können, die menschlichen Benutzern ermöglicht, Informationen über anatomische Strukturen zu erkennen, selbige zu unterscheiden sowie interaktiv im Datensatz zu navigieren. Der primäre Einsatzzweck des Systems besteht hierbei in der Visualisation von Tumoren, die durch spezielle Klassifizierungsverfahren visuell hervorgehoben werden.

Ein Tumor im Sinne dieser Definition sei eine Raumforderung oder Gewebeanomalie, die sich beim verwendeten Verfahren, die medizinischen Eingabedaten zu gewinnen, vom umgebenden Gewebe oder Raum abzeichnet.

[1] Algorithmus – Verarbeitungsvorschrift
[2] Siehe Kapitel 3.2 für eine Definition
[3] VERTEBRA – engl. *vertebra* = Wirbel – Volumetric Examiner for Radiological/Tomographical Experimental Basic Realtime Analysis

Im Einzelnen sind die Anforderungen wie folgt zu beschreiben:

Echtzeitfähigkeit und Interaktionsfähigkeit Das System muss in der Lage sein, Eingabedaten in Echtzeit zu verarbeiten, sodass der menschliche Benutzer eine Änderung der Ausgabedaten nicht als Einzelbildsequenz sondern als dynamischen Prozess wahrnimmt (vgl [AMHH08, Kapitel 1, Seite 1]). Eine Vergleichsgröße ist die Einheit FPS[4], die die Anzahl der Bilder pro Sekunde angibt. Möller et al. definieren in [AMHH08], dass ab 6 FPS für das menschliche Auge der Übergang von Einzelbildern zu einer Bewegung zunimmt und ab 15 FPS ein Programm als sicher echtzeitfähig gelten kann – ab 72 FPS kann das menschliche Auge keinen Unterschied mehr erkennen (vgl. [AMHH08, Kapitel 1, Seite 1]). Als weiteres Kriterium führen Möller et al. an, dass ein derartiges System fähig sein muss, auf Benutzerinteraktionen mit geringer Latenzzeit[5] zu reagieren (vgl. [AMHH08, Kapitel 1, Seite 1]). Beispiele für solche Interaktionen sind die freie Verschiebung des Blickpunktes bzw. Blickwinkels sowie die Vergrößerung bzw. Verkleinerung des dargestellten Ausschnittes („zoomen“). Möglichkeiten der Verarbeitung von Nutzerinteraktionen werden in [Bru04, Kapitel 3.6, Seite 62-66] beschrieben.

Medizinische 3D-Eingabedaten Ein Echtzeit-Tumorvisualisationssystem dient im Sinne der obigen Definition zur Verarbeitung medizinischer Eingabedaten. Die Eingabedaten müssen sich über drei Dimensionen erstrecken. CT- und MRT-Scanner liefern mehrere zweidimensionale Datensätze, die zusammengenommen einen dreidimensionalen Eingabedatensatz bilden. Obwohl in der Praxis nicht üblich, kann auch ein einzelner zweidimensionaler Datensatz als dreidimensionaler Eingabedatensatz mit der Tiefe 1 gelten. Gemäß obiger Definition müssen die Eingabedaten nicht konstant bleiben – dadurch wird der Einsatz des Systems in intraoperativen Verfahren mit wiederholten tomografischen Scans wie bei [OKKS94] ermöglicht - in dieser Publikation beschreiben die Autoren den Einsatz eines solchen Verfahrens bei der chirurgischen Entfernung von Gliazellentumoren exemplarisch. Um die Eingabedaten für die Software lesbar zu machen, müssen die Datensätze entweder in einem speziellen Format (beispielsweise dem DICOM-Standard, zu dem [MEM02] eine Einführung bietet) gespeichert werden, oder aber es findet eine direkte Anbindung an die Scannerhardware statt.

Hochqualitative Ausgabedaten Das System muss in der Lage sein, die Daten in einer Art und Weise zu visualisieren, die eine für die Anwendung und das verwendete Display[6] angemessene Qualität liefert – da hochqualitative Rendertechniken eine höhere Rechenleistung erfordern, muss dafür die Bildrate reduziert werden. Da ab einer bestimmten Qualitätsstufe die subjektiv wahrgenommene Qualität nicht mehr steigt, macht es keinen Sinn, die Qualität ab dieser Stufe weiter auf Kosten der Bildrate zu steigern (vgl. [KAB+08, Kapitel 3.3, Seite 5]).

[4] FPS – Frames per second
[5] Latenzzeit – Zeit zwischen Aktion und Reaktion
[6] Ein Display sei in diesem Kontext ein Gerät, das dazu dient, Informationen visuell anzuzeigen

Klassifizierungsverfahren für Tumoren Vom System wird gefordert, Klassifizierungsalgorithmen[7,8] ausführen zu können, die zur visuellen Hervorhebung von Tumoren dienen.

Vergleich zu [Bru04, Kapitel 3.1.1, Seite 17f.] Diese Definition unterscheidet sich von den in [Bru04, Kapitel 3.1.1, Seite 17] beschriebenen Anforderungen durch die Spezialisierung auf Tumoren, die explizit geforderte Echtzeitfähigkeit sowie die nicht vorhandene Anforderung, das System in reiner Software zu implementieren (siehe dazu auch Kapitel 3.1 auf der nächsten Seite). Die obige Definition ist zudem in Bezug auf das eingesetzte Verfahren zur Gewinnung der Eingabedaten allgemein gehalten, während [Bru04] sich primär auf CTs und MRTs konzentriert.

Lokalisierung bei Visualisationssystemen Besonders bei intraoperativen und radiotherapeutischen Verfahren wird an Tumorvisualisationssysteme, speziell an solche, die Konzepte der in Kapitel 4 vorgestellten Augmented Reality benutzen, die Anforderung gestellt, die Bewegungen des Körpers des Patienten, des Betrachters und des Displays verfolgen zu können und diese Daten in die Berechnung einfließen zu lassen. Vor allem für das Mitverfolgen von Bewegungen des Patienten und des Betrachters sind nach aktuellem Stand der Technik ein oder mehrere am sich bewegenden Ziel angebrachte Marker vonnöten, die je nach System beispielsweise magnetisch arbeiten (siehe [SVH+02b]) oder infrarotes Licht reflektieren und somit von mehreren Kameras erfasst und im dreidimensionalen Raum lokalisiert werden können[9]. Zudem kann je nach Anwendungsbereich eine Registrierung nötig sein, die versucht, zwei dreidimensionale Datensätze durch Rotation, Translation[10] und Skalierung[11] zu überlagern. Das in Kapitel 5 auf Seite 18 und [SVH+02b] vorgestellte System ARION™ benutzt beispielsweise die Registrierung, um präoperativ und intraoperativ generierte Bilder zu überlagern.

2.2. Anwendungsmöglichkeiten

Schon heute werden medizinische Visualisationssysteme in den verschiedensten Bereichen der Medizin eingesetzt – daher kann an dieser Stelle lediglich eine exemplarische Auswahl präsentiert werden:

- **Intraoperative Verfahren**: Die Visualisationssysteme werden während chirurgischen Verfahren eingesetzt. Wie bereits in Kapitel 2.1 erwähnt setzten Okudera et al. in [OKKS94] ein medizinisches Visualisationssystem zur chirurgischen Entfernung von Gliazellentumoren ein; Abgesehen von den bereits erwähnten Verfahren, die sich auf onkologische Verfahren beschränken, existieren auch Anwendungsmöglichkeiten

[7] Klassifizierung – Prozess, um einem einzelnen Datenelement eine Farbe und Transparenz zuzuordnen (vgl. [Bru04, Kapitel 3.2.2, Seite 28])

[8] Anmerkung: Die hier behandelte Klassifizierung ist nicht mit der taxonomischen Klassifikation von Tumoren zu verwechseln

[9] Ein Problem stellt hierbei die Verdeckung der freien Sicht der Kameras auf die Marker dar

[10] Translation – Bewegung/Verschieben

[11] Skalierung – Vergrößerung bzw. Verkleinerung

in anderen Gebieten der Chirurgie: Beispielsweise nutzen Maupu et al. in [MVHWB05] ein stereoskopisches Visualisationsverfahren, um den Chirurgen während endovaskulärer Eingriffe zu leiten und erhöhen dadurch die Sicherheit dieser Verfahren (vgl. [MVHWB05, Abstract]). Auf intraoperative Verfahren zielen besonders Forschungsarbeiten ab, die die in Kapitel 4 auf Seite 16 vorgestellten Konzepte der Augmented Reality benutzen. Beispielsweise beschreiben Suthau et al. in [SVH+02b] Konzepte und Einsatzmöglichkeiten des Augmented Reality-basierten Visualisationssytems ARION™ in der Leberchirurgie – in diesem Kontext wird vor allem die Resektion eines Lebertumors behandelt. ARION™ wird in Kapitel 5 auf Seite 18 genauer behandelt.

- **Radiotherapeutische Verfahren**: Auch in der Radiotherapie werden bereits jetzt medizinische Visualisationsverfahren eingesetzt. Einen der Anwendungsbereiche stellt hier die Vorraussage von so genannten Portalen dar: Um die Strahlendosis, die auf gesundes Gewebe einwirkt, möglichst gering zu halten, muss aufgrund der Bewegung des menschlichen Körpers (zum Beispiel Atmung) vorhergesagt werden können, wann sich ein Tumor wo befindet, um den Therapiestrahl nur dann einzuschalten, wenn möglichst viele Schäden am Tumor und gleichzeitig möglichst wenige Schäden am gesunden, umgebenden Gewebe zu erwarten sind. Eine Methode zur kontinuerlichen Visualisation dieser Portale wurde bereits 1987 in [LS87] vorgestellt. Auch in der Verifikation des Erfolges von radiotherapeutischen Verfahren werden Visualisationssysteme eingesetzt. Einen weiteren Anwendungsbereich innerhalb der Radiotherapie, für den sich speziell Tumorvisualisationssysteme eignen, bietet die Brachytherapie[12] – hier ist vor allem die genaue Planung von Bedeutung, um Schäden an gesundem Gewebe zu vermeiden. Pötter et al. beschreiben in ihrem zweiteiligen Artikel [HMPL+05] sowie [PHML+06] Empfehlungen der ESTRO[13] für Konzepte, Terminologien und Volumen-Dosis-Beziehungen für die brachytherapeutische Behandlung von cervicalen[14] Tumoren. Robert Krempien beschreibt schliesslich in [KDH+03] einen Ansatz, durch die Kombination von MRT- und CT-Datensätzen das zu bestrahlende Volumen bei der Planung einer brachytherapeutischen Behandlung besser zu definieren.

3. Konzepte zur Tumorvisualisation

3.1. Vergleich von Hardware- und Softwarealgorithmen

Bei Betrachtung der Hardware, auf der ein System die Berechnungen zur Umwandlung der Eingabedaten in Ausgabedaten durchführt, lassen sich zwei grundlegende Konzepte unterscheiden:

- **Hardwarealgorithmen**: Diese Methode benutzt für einen oder mehrere Berechnungsschritte Hardware, deren Vorhandensein nicht auf allen Plattformen garantiert

[12] Brachytherapie – Radiotherapeutisches Verfahren, bei dem Strahlungsquellen in der Nähe des zu bestrahlenden Tumors platziert werden
[13] ESTRO – [The] European Society for Therapeutic Radiology and Oncology
[14] Cervical – Auf den Gebärmutterhals (Cervix) bezogen

ist[15,16]. Durch diese Hardware steigt meist die Geschwindigkeit des Systems. Beispiele für Hardware im Sinne dieser Definition sind:

  - GPUs, siehe Kapitel 3.3.2 auf Seite 9
  - FPGAs, siehe Kapitel 3.3.3 auf Seite 10
  - Spezialhardware, siehe Kapitel 3.3.4 auf Seite 11

- **Softwarealgorithmen**: Diese Methode benutzt für keinen Berechnungsschritt derartige Hardware, sondern lediglich die auf allen üblichen Plattformen vorhandenen CPUs. Bevor die Daten zur Anzeige auf einem Display an die Grafikeinheit übergeben werden, ist die Transformation von dreidimensionalen Eingabedaten in zweidimensionale Ausgabedaten bereits abgeschlossen. Ein Beispiel für einen reinen Softwarealgorithmus ist das in [Bru04] behandelte Raycasting.

Bruckner vergleicht in [Bru04] Software- und Hardwaremethoden wie folgt:
Obwohl Grafikhardware genutzt werden könnte, um die Geschwindigkeit eines Volumenrenderingsystems zu erhöhen, entstehen dadurch Probleme wie veraltete Treiber[17] oder der Unterschied von unterstützten Funktionen von System zu System (Übersetzt aus [Bru04, Kapitel 3.1.1, Seite 17, Paragraph „Pure Software “]). Bruckner stellt daher an die in seiner Arbeit verwendeten Verfahren den Anspruch, Softwarealgorithmen im Sinne der obigen Definition zu sein. Diese Arbeit behandelt dagegen auch Hardwarealgorithmen, da der Performancegewinn mitunter nicht unerheblich ist und aktive Forschung über beide Methoden betrieben wird (für Details bezüglich der Hardwareplattformen siehe 3.3 auf der nächsten Seite).

3.2. Vergleich von Volumen- und Oberflächenverfahren

Abgesehen von der diskutierten Unterscheidung zwischen Software- und Hardwarealgorithmen können 3D-Visualisationsalgorithmen in Volumen- und Oberflächenverfahren unterteilt werden.

Volumenrendern[18] ist ein Verfahren innerhalb der Visualisationstechnik, das sich mit Volumendaten beschäftigt – also Daten, die möglicherweise Informationen in ihrem Inneren haben, jedoch nicht aus Oberflächen oder Kanten bestehen (vgl. [Bru04, Kapitel 1, Seite 2]).

Als Alternative bietet sich das so genannte Surface Rendering („Oberflächenrendern“) an, bei dem durch mathematische Verfahren Flächen und Kanten im Datensatz gesucht werden, die dann zusammenhängend gerendert werden.(vgl. [Bru04, Kapitel 1, Seite 2f.]).

[15] Anmerkung: Im Sinne dieser Definition gelten CPUs nicht als Spezialhardware, da sie notwendigerweise zur Datenverarbeitung auf Systemen, auf denen die hier beschriebenen Algorithmen lauffähig wären, vorhanden sein müssen.

[16] Anmerkung: Diese Hardware ist nicht zwangsläufig mit der in Kapitel 3.3.4 beschriebenen Spezialhardware gleichzusetzen

[17] Anmerkung: Ein Treiberupdate würde eine erneute Zertifizierung nach dem Medizinproduktegesetz erfordern und wäre daher zeitaufwändig und damit kostenintensiv

[18] Anmerkung: Die Begriffe Volumenrendern und Volumenrendering werden in dieser Arbeit synonym verwendet

Kuszyk et al. stellen in [KHBF96] die Nachteile des Surface Renderings gegenüber dem Volumenrendern dar. Als Beispiel für einen Nachteil des Surface Rendering wird dargestellt, dass Läsionen unter Knochen verborgen werden während Volume Rendering-Algorithmen diese korrekt darstellen (Quelle: [KHBF96], Abstract). Da die Publikation jedoch schon 1996 veröffentlicht wurde, gelten die Resultate unter Umständen für die heutigen Algorithmen und hochauflösenden Tomographen nicht mehr. Eine aktuellere Darstellung liefern Udupa et al. in [UHC91]. Die Autoren kommen zu dem Schluss, dass Surface Rendering-Algorithmen im Bereich der Informationsdarstellung einen kleinen Vorteil gegenüber volumenbasierten Algorithmen haben, aber vor allem durch den geringeren Bedarf an Rechenzeit und Speicherplatz überlegen sind. Auch [Bru04] thematisiert den Vergleich von Oberflächen- und Volumenrendering und kommt zum Schluss, dass der Vorteil an gewonnener Bildqualität den Mehraufwand an Rechenzeit und Speicher gegenüber Surfacerendering-Algorithmen überwiegt – tatsächlich würde beim Rendern der Oberflächen eine Dimension verloren gehen, da das Innere von Objekten verborgen bleibt (vgl. [Bru04, Seite 2f.]).

3.3. Plattformen zur performanten Berechnung von Visualisationsdaten

3.3.1. Berechnung auf dem Hauptprozessor

Die naheliegendste Möglichkeit, Operationen auf großen Mengen volumetrischer Daten[19] durchzuführen, ist, die Berechnungen vom Hauptprozessor des Computers durchführen zu lassen. Da diese Recheneinheit auf allen heute üblichen Computerplattformen vorhanden ist, bleibt die Applikation unabhängig von spezieller Hardware.

Obwohl moderne Prozessoren mehrere Rechenkerne besitzen und dadurch eine Parallelisierung möglich ist, bietet dennoch spezialisierte Hardware eine höhere Parallelisierbarkeit. Zudem muss zur Optimierung der Geschwindigkeit einer Applikation oft der Code auf bestimmte CPUs eines bestimmten Herstellers optimiert werden. Beispielsweise unterstützen moderne Prozessoren so genannte SIMD[20]-Operationen, die dem Programm erlauben, eine Operation auf mehrere Variablen gleicher Länge und gleichen Typs auf einmal anzuwenden (vgl. [Fog10, Kapitel 12, Seite 103]. Sofern eine Applikation solche Befehle benutzt[21], entsteht dadurch eine Abhängigkeit von Plattformen, welche Prozessoren verbaut haben, die die jeweiligen SIMD-Instruction sets[22] unterstützen.

Es gibt zwar Ansätze, erst zur Laufzeit einen speziell auf die vorhandene Plattform abgestimmten Code zu benutzen – allerdings muss dafür der Code entweder manuell oder automatisch für die jeweilige Plattform angepasst werden. Bei einer manuellen Anpassung würde dies in einem stark erhöhten Zeit- und damit Kostenaufwand resultieren (vgl. auch [Fog10][23]). Dagegen besteht auch die Möglichkeit, die Optimierung des Quellcodes

[19] Volumetrische Daten – Daten, die dreidimensionale Informationen enthalten

[20] SIMD – Single Instruction Multiple Data

[21] Anmerkung: Für C/C++ bieten die so genannten Intrinsics ein API zur Benutzung dieser Operationen, siehe auch [Kapitel 12.3, Seite 107]Fog2010

[22] Instruction Set – Befehlssatz (eines Prozessors) – (Unter)menge der vom Prozessor unterstützten Maschinencode-Befehle

[23] Anmerkung: Fog behandelt nicht die Optimierung für spezielle Echtzeitbetriebsssyteme, die im medizi-

von einem Compiler[24] vornehmen zu lassen. Zum Zeitpunkt des Verfassens dieser Arbeit ist die Compilertechnologie jedoch noch nicht so weit fortgeschritten, dass die automatische Optimierung einen menschlichen Entwickler mit Erfahrung in diesem Bereich ersetzen könnte (siehe auch [Fog10]). Außerdem bestehen hier starke Unterschiede zwischen den einzelnen Compilern – beispielsweise läuft vom Intel C++ Compiler generierter Maschinencode, der auf CPUs desselben Herstellers hochperformant läuft, oft nur langsam oder gar nicht auf Prozessoren des Konkurrenten AMD (vgl. [Fog10, Kapitel 2.5, Seite 10]).

3.3.2. Berechnung auf GPUs

Da die Rechenleistung der Hauptprozessoren moderner Computer für viele der heutigen 3D-Anwendungen nicht mehr ausreicht, enthält ein Großteil der heutigen Computer Grafikhardware, die in der Lage ist, 3D-Anwendungen zu beschleunigen. Hierbei berechnet der Hauptprozessor die darzustellenden Daten und gibt sie an die Grafikhardware weiter, um aus den Daten mithilfe der GPU[25] ein zweidimensionales Bild zu erzeugen, das beispielsweise auf einem Display angezeigt werden kann.

Um der GPU mitzuteilen, welche Objekte wo im dreidimensionalen Koordinatensystem auf welche Art gerendert werden sollen, müssen Programme ein Grafik-API[26] wie OpenGL oder DirectX benutzen. Im konkreten Fall von VERTEBRA fiel die Entscheidung aufgrund der Plattformunabhängigkeit auf OpenGL.

Neuere GPUs können außerdem so genannte Shader ausführen – kleine Programme, die für bestimmte Untereinheiten der zu visualisierenden Objekte ausgeführt werden. OpenGL stellt hierfür die Shaderprogrammiersprache GLSL[27] zur Verfügung, die in [Ros06] für die OpenGL-Version 2.0 ausführlich beschrieben wird. Nach [Ros06, Seite 38-47] können in dieser Sprachversion die folgenden Typen von Shadern unterschieden werden[28]:

- Vertex[29]-Shader
- Fragment[30]-Shader

Die Shader-Prozessoren (Untereinheiten der GPU, die die Shader ausführen), arbeiten massiv parallel, führen also das Shaderprogamm für eine große Zahl an Vertices bzw. Fragments gleichzeitig aus. Dies wird durch die spezielle Hardwarearchitektur von GPUs ermöglicht. Daraus resultiert ein großer Zuwachs an Geschwindigkeit, was speziell für Echtzeit-Tumorvisualisationssysteme wichtig ist, da die Operationen ständig über neuen Datensätzen berechnet werden müssen. Das in 2.1 auf Seite 3 dargestellte Echtzeitkriterium

nischen Bereich teilweise zum Einsatz kommen

[24] Compiler – Programm, das menschenlesbaren Quellcode in maschinenlesbare Befehle umsetzt

[25] GPU – Graphics Processing Unit – Grafikprozessor

[26] API – Application Programming Interface – Programmierschnittstelle

[27] GLSL – [Open]Graphics Layer Shading Language – OpenGL-Shadersprache

[28] Geometry-Shader sind im in [Ros06] benutzten Standard OpenGL 2.0 nicht enthalten und werden daher an dieser Stelle nicht aufgeführt

[29] Vertex – Ein einzelner Punkt im Raum, der einen Eckpunkt eines Polygons bzw. einen Punkt oder eines der Enden einer Linie darstellt. – Vgl. [WS00, Seite 664] bzw. [Ros06, Seite 685]

[30] Fragment – Datensatz bestehend aus der Information, die nötig ist, um ein Pixel zu zeichnen Vgl. [Ros06, Seite 675]

wäre nicht erfüllt, falls die Zeit, die benötigt wird, um die Berechnungen einschließlich Rendering auf einem Datensatz auszuführen, größer ist als die Zeitdifferenz zum Eintreffen des nächsten Datensatzes. Durch die massive Parallelisierung kann dieses Kriterium somit für größere Datensätze erfüllt werden. Den Ansatz, Fragment-Shader für GPU-beschleunigtes Volumenrendering zu benutzen, verfolgen unter anderem Kruger und Westermann in [KW03].

In den letzten Jahren tritt abgesehen von Shadern vor allem das GPGPU[31]-Konzept in den Vordergrund, das die Möglichkeit bereitstellt, GPUs frei programmieren zu können, im Gegensatz zu Shadern aber nicht auf Grafikdatenmengen wie Vertices oder Fragmente sondern auf beliebigen Eingabedatensätzen ausgeführt wird. Wie auch bei Shadern kann die Anwendung von der massiven Parallelisierung profitieren. Auch für GPGPU-Programmierung müssen bestimmte APIs definiert werden, um der Applikation den nötigen Zugriff auf die Grafikhardware zu gewähren – [SK10] führt in das CUDA[32]-API des GPU-Herstellers NVIDIA ein, während [KHKH10] neben CUDA auch das herstellerunabhängige OpenCL[33] thematisiert. Marsalek et al. wenden schließlich das GPGPU-Konzept auf das Volumenrendering an und stellen ihre CrUDA-basierte Implementation in [MHS08] vor.

3.3.3. Berechnung auf FPGAs

Abgesehen von den bereits diskutierten Möglichkeiten ist mit so genannten FPGAs[34] eine weitere Form der Hardwarebeschleunigung für die medizinische Visualisationstechnik verfügbar. FPGAs sind ICs[35], bei denen nach der Herstellung eine anwendungsspezifische Programmierung und Konfiguration möglich ist (vgl. [Kib09], Kapitel 1.7.1, Seite 16). Daher sind die Produktionskosten für ein auf FPGAs basierendes Visualisationssystem wesentlich geringer als bei dedizierter Hardware, die für eine bestimmte Aufgabe gefertigt wurde und nicht nachträglich programmierbar ist. Wie bereits in Kapitel 3.1 auf Seite 6 diskutiert, stellt Bruckner in [Bru04] Nachteile von hardwarebasiertem Rendering gegenüber softwarebasierten Darstellungen im Bezug auf große volumetrische Datensätze dar. FPGAs erlauben, wie Leeser et al. in [LCM+05] mithilfe einer Implementation des Parallel-Beam Backprojection-Algorithmus zeigen, eine performante Implementation dieser Softwarealgorithmen. In [THL09] findet sich ein Vergleich der Performanz von CPU-, GPU- und FPGA-basierten Systemen anhand von PRNGs[36] – wie aus Tabelle 6 in dieser Publikation ersichlich liefert das im Experiment benutzt FPGA-basierte System bei zwei der drei dargestellen Verteilungen[37], die jeweils unterschiedliche Algorithmen benötigen, bessere Resultate (also eine größere Zahl generierter Zufallszahlen pro Zeiteinheit) als die anderen getesteten Plattformen.

[31] GPGPU – General Purpose Graphics Processing Unit
[32] CUDA – Compute Unified Device Architecture
[33] OpenCL – Open Computing Language
[34] FPGA – Field Programmable Gate Array
[35] IC – Integrated Circuit – Integrierter Schaltkreis
[36] PRNG – Pseudo-random number generator – Pseudozufallszahlengenerator
[37] Anmerkung: Gemeint ist hier die statistische Verteilung der Zufallszahlen, zum Beispiel Normalverteilung oder Cauchy-Verteilung

3.3.4. Berechnung auf Spezialhardware

Neben den dargestellten Visualisationsmethoden gibt es durchaus Ansätze, die für die Visualisation nötigen Berechnungen auf spezialisierter Hardware auszuführen. Aufgrund des geringen Umfangs dieser Arbeit soll an dieser Stelle lediglich ein Überblick über die bereits bestehenden Hardwareplattformen gegeben werden. Diese Zusammenstellung basiert auf [Bru04, Kapitel 2.5.5, Seite 14].

- Günter Knittel beschreibt in [Kni95] mit VOGUE eine skalierbare Architektur für Volumenrendering
- VIRIM ist eine massiv-parallele, echtzeitfähige Volumenrenderingarchitektur, die in [GPR+95] beschrieben wird
- In [MKW+02] beschreiben Meißner et al. VIZARD II, eine interaktive und rekonfigurierbare Volumenrenderingarchitektur
- Eine kostenoptimierte, echtzeitfähige Architektur namens EM-Cube beschreiben Osborne et al. in [OPL+97]
- EM-Cube dient zudem als Basis für das kommerziell verfügbare VolumePro-Board, das in [PHK+99] beschrieben wird (vgl. [Bru04, Kapitel 2.2.5, Seite 14])

Insbesondere bei [Kni95], [GPR+95] sowie [OPL+97] ist zu beachten, dass auch hier das Publikationsdatum sehr lange zurückliegt und daher die Systeme unter Umständen nicht ausreichend mit aktueller Hardware zusammenarbeiten – beispielsweise könnte die Auflösung und Geschwindigkeit der heutigen Tomographen für ehemals echtzeitfähige Systeme ein Problem darstellen. Das kommerziell verfügbare VolumePro-Board aus [PHK+99] ist jedoch laut [Bru04, Kapitel 2.2.5, Seite 14] fähig, einen Datensatz mit einer Auflösung von 512 Bildpunkten in allen drei Raumdimensionen mit einer Geschwindigkeit von 30 FPS zu rendern.

3.4. VERTEBRA – Beispiel für einen Klassifizierungsalgorithmus

Um einen Klassifikationsalgorithmus und die Verarbeitung von Eingabedaten in einem Echtzeit-Tumorvisualisationssystem zu demonstrieren, wurde mit VERTEBRA im Rahmen dieser Arbeit ein Proof-of-Concept-Projekt geschrieben. Diese Implementation fällt unter die Kategorie der Hardwarealgorithmen (in diesem Fall wird ein Teil der Berechnung durch Shader auf einer GPU durchgeführt) und benutzt die folgende Verarbeitungsvorschrift für Eingabedaten:

1. Import als Satz von Einzelbildern aus dem Eingabedatenformat – Unterstützt wird hier das DICOM-Format sowie diverse, weit verbreitete Bildformate, darunter JPEG[38] und PNG[39]. Davon eignet sich prizipiell nur DICOM zur Verarbeitung von medizinischen Daten, da nur bei diesem Format medizinische Metadaten standardisiert

[38] JPEG – Joint Pictures Expert Group
[39] PNG – Portable Network Graphics

gespeichert werden können und zum Beispiel Hounsfield-Fenster direkt auf die Daten angewandt werden können[40].

2. Für CT-Eingabedaten: Anwendung eines Hounsfield-Fensters auf die Eingabedaten. Da zum Beispiel MRT-Eingabedaten durch dimensionslose Einheiten repräsentiert werden, muss dieser Schritt hier entfallen.

3. Umrechnung der Daten in ein eigenes Bildformat, das einen performanten Zugriff auf die Graustufendaten erlaubt.

4. Übergabe der Daten an das Grafik-API OpenGL, wobei jeder Bildpunkt durch einen Punkt mit einer bestimmten Farbe repräsentiert wird[41]

5. Klassifizierung von Tumoren durch einen GLSL-Vertexshader (siehe auch Kapitel 3.3.2 auf Seite 9). Hierbei wird der Graustufenverlauf der Vertices auf einen Farb- und Transparenzverlauf abgebildet. Ausgehend von der Beobachtung, dass Tumoren auf den Eingabedaten einen höheren Graustufenwert (zum Beispiel Wert auf der Hounsfield-Skala) haben als das umgebende Gewebe, werden Vertices umso transparenter und umso türkiser eingefärbt, je geringer ihr Graustufenwert ist. Je höher dagegen ihr Graustufenwert ist, desto weniger transparent und desto röter ist die Farbe, die ihnen zugeordnet wird. Durch dieses Verfahren geben Gewebestrukturen mit einem niedrigen Graustufenwert (zum Beispiel weniger dichte Strukturen in CT-Scans) den Blick auf hinter ihnen liegende Strukturen mit einem hohen Graustufenwert frei, die in der Signalfarbe rot gerendert werden und nicht oder nur wenig transparent sind. Nicht alle Tumoren haben bei Scans mit unterschiedlichen bildgebenden Verfahren einen höheren Graustufenwert als das umgebende Gewebe. Verfahren, die einem Tumor denselben Wert zuordnen wie dem umgebenden Gewebe machen eine Unterscheidung für das Visualisationssystem prinzipbedingt unmöglich – daher hängt die ordnungsgemäße Funktion eines Visualisationssystems in erster Linie von der Qualität der Eingabedaten ab. Bei Tumoren, die sich im Eingabedatensatz durch einen geringeren Graustufenwert als das umgebende Gewebe auszeichnen, ist eine Umkehr des Farb- und Transparenzverlaufes, auf den die Graustufenskala abgebildet wird, erforderlich.

Exemplarisch wird die Verarbeitung von CT-Eingabedaten anschaulich in Abbildung 1 auf Seite 14 dargestellt, wobei die Umsetzung von Eingabedaten in Objekte im dreidimensionalen Koordinatensystem der Einfachheit halber unberücksichtigt bleibt.

Das Verfahren, das VERTEBRA implementiert, ist wie bereits zuvor erwähnt als Hardwarealgorithmus einzustufen, da für Schritt 4 ein Shaderprogramm verwendet wird, das auf

[40] Anmerkung: Die angesprochenen Bildformate eignen sich zwar durch die so genannten EXIF-Daten ebenfalls zur Speicherung beliebiger Metadaten, jedoch sind diese für medizinische Daten nicht standardisiert

[41] Anmerkung: Es gibt diverse Möglichkeiten, die Eingabedaten in eine Menge von dreidimensionalen Objekten umzurechnen. Alle Möglichkeiten zu beschreiben und zu diskutieren, würde den Umfang dieser Arbeit sprengen. Aufgrund der geringen Anzahl von Vertices, der Unterscheidbarkeit der einzelnen Datenelemente und der Einfachheit der Implementierung wurde für VERTEBRA die Punkte-Methode gewählt.

der GPU ausgeführt wird – zudem findet der gesamte Renderingprozess auf der GPU statt. Dadurch kann VERTEBRA die Geschwindigkeitsvorteile, die durch die Verwendung der GPU entstehen, ausnutzen – allerdings gelten auch hier die in 3.1 auf Seite 6 dargestellten Nachteile von Hardwarealgorithmen. Abgesehen von der intermediären, performanten Datenstruktur zur Speicherung der Bilddaten[42] wurden keine weiteren Optimierungen implementiert – zum einen, um die Implementierung einfach und wartbar zu halten, zum anderen, da die meisten Optimierungen die Resultate bestimmter Berechnungen zwischenspeichern und so diese Schritte selbst bei einer Navigation im 3D-Koordinatensystem nicht mehr erneut durchführen müssen, daher aber von konstanten Eingabedaten abhängig sind, was speziell bei intraoperativen Systemen nicht der Fall ist (zum Beispiel wenn eine intraoperative Verifikation der Resektion vorgenommen wird) und einen Teil der in 2.1 auf Seite 3 dargestellten Anforderungen darstellt. Ein Beispiel für solche Optimierungen wäre, die Klassifizierung bereits vor der Übergabe der Daten an OpenGL durchzuführen und die bereits klassifizierten Daten im Speicher vorzuhalten[43]. Daher geben die im Folgenden angegebenen Geschwindigkeitsmesswerte lediglich einen Überblick über die zu erwartenden Bildraten bei sich andauernd ändernden Eingabedaten.

Geschwindigkeit VERTEBRA erreicht durch seine Implementierung des zuvor dargestellten Verfahrens auf der in Anhang B auf Seite 20 beschriebenen Plattform mit Eingabedaten von 15 Einzelbildern mit 512 mal 512 Bildpunkten aus dem Visible Human Project eine Bildrate von 7 Bildern pro Sekunde, was für diese Auflösung gerade noch ausreichend für eine Einteilung als Echtzeitsystem nach Kapitel 2.1 auf Seite 4 ist. Bei einem Einsatz als intraoperatives Echtzeitsystem ist zudem die Auflösung der Eingabedaten stark von der verwendeten Bildgebungshardware abhängig. Je nach Einsatzszenario könnten bei einem Produktivsystem die oben beschriebenen Kriterien für eine Einhaltung des Echtzeitkriteriums zwischen Änderungen der Eingabedaten sorgen, während für die komplette Neuberechnung bei Änderung der Eingabedaten eine höhere Latenzzeit akzeptabel wäre. Für Softwarealgorithmen nach der Definition in Kapitel 3.1 müssen wiederum andere Optimierungsverfahren zum Einsatz kommen, da der Prozess des Renderns hier viel Rechenzeit in Anspruch nimmt.

VERTEBRA kann bei Datensätzen in der Größenordnung des VolumePro-Boards (vgl. 3.3.4) das Echtzeitkriterium nicht erfüllen, was jedoch auch am auf der Testplattform nur begrenzt zur Verfügung stehenden Grafikkartenspeicher liegen dürfte - dieser wird im beschriebenen Verfahren stark ausgelastet.

Abbildung 2 auf Seite 15 zeigt VERTEBRAs Darstellung eines gesunden Gewebes[44]; deutlich zu sehen sind teils volltransparente Gebiete (durch die der schwarze Hintergrund zu sehen ist) sowie die in der Signalfarbe rot angezeigten Knochen. Ein gut auszumachender,

[42] In dieser Struktur wird jedes einzelne 2D-Eingabebild zusammenhängend im Arbeitsspeicher als Array von Gleitkommazahlen mit doppelter Präzision gespeichert

[43] Auch eine Speicherung auf der Grafikkarte ist möglich – Dies könnte in OpenGL beispielsweise durch ein Vertex Buffer Object implementiert werden

[44] Quelle der DICOM-Daten: Visible Human Project; Männlicher Pelvis – 1mm CT-Schnittbilder; VHM.501.DCM bis VHM.515.SCM; Verwendet wurde ein Weichteil-Hounsfield-Fenster mit einer Breite von 350 HU bei einem Zentrum bei 50 HU

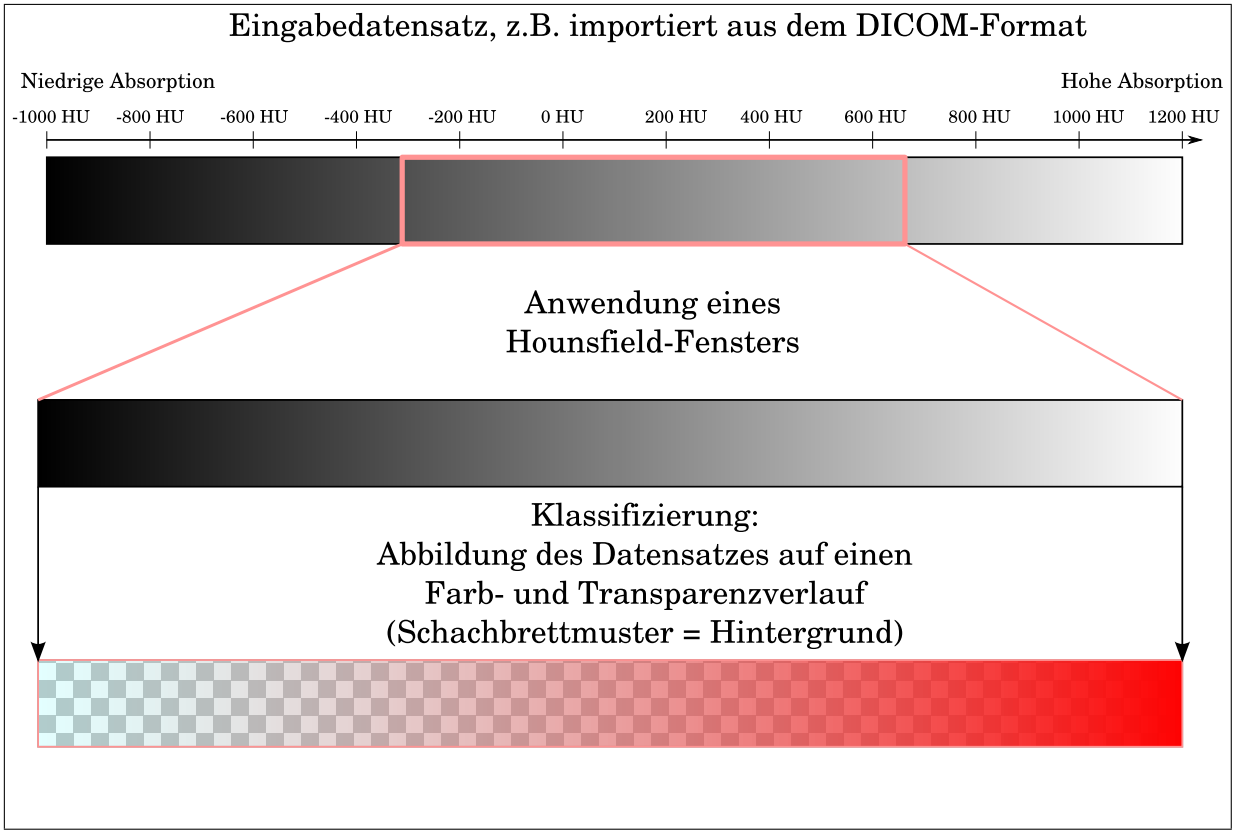

Abbildung 1: Verarbeitung von CT-Eingabedaten in VERTEBRA

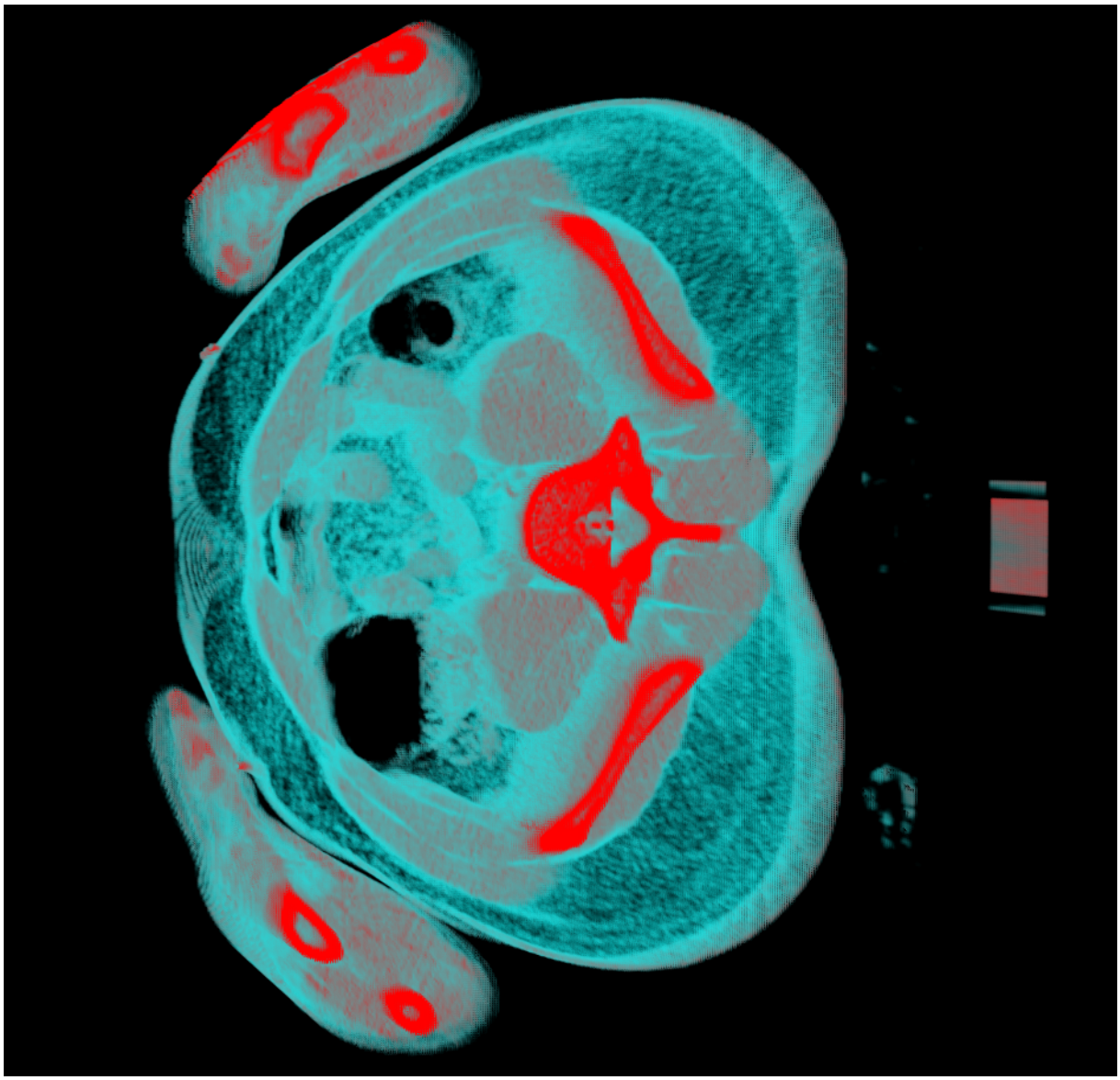

Abbildung 2: Beispielrendering aus VERTEBRA

weil stark absorbierender Tumor, würde ebenfalls rot angezeigt werden.

4. Augmented Reality – Zukunft von Visualisationssystemen?

In den letzten Jahren wird neben den herkömmlichen Displays intensiv an der so genannten „Augmented Reality“ geforscht. Dieser Begriff, der sich mit 'Erweiterte Realität' (Quelle: [Tö10, Seite 1]) übersetzen lässt, beschreibt laut [SVH+02b] (Seite 1; Englische Publikation: [SVH+02a]) das Konzept, reale Bilder mit zusätzlichen Informationen zu ergänzen.

4.1. Displays für Augmented Reality-Applikationen

In [Tö10, Kapitel 2.2, Seite 21] werden zwei grundsätzliche Methoden unterschieden, eine Kombination von realen und virtuellen Daten in Displays umzusetzen:

- **Optical See-Through Displays**: Diese Art von Displays ermöglicht den „direkten Blick auf die umgebende Welt“[45]. Technisch kann dieses Konzept beispielsweise durch einen halbdurchlässigen Spiegel, der Combiner genannt wird, realisiert werden. Dadurch bleibt die „direkte Sicht auf die Umgebung“[45] erhalten und die Darstellungsqualität der realen Informationen wird kaum verringert – lediglich der Transmissionsgrad des halbdurchlässigen Spiegels von mitunter weniger als 50% führt zu einer verringerten Lichtstärke des realen Bildes (vgl. [SVH+02b, Kapitel 3.1.3, Seite 7]). Problematisch ist hierbei die je nach Geschwindigkeit des Systems unterschiedlich große Verzögerung der virtuellen Komponente gegenüber dem realen Bild, die je nach Verarbeitung der Daten durch die Applikation unterschiedlich viel Zeit in Anspruch nehmen kann. Im Englischen wird diese Verzögerung mit *lag* bezeichnet. Zur reinen Datenverarbeitung kommt hier noch hinzu, dass Position und Ausrichtung derjenigen Person, aus deren Sicht die Daten dargestellt werden sollen, von Sensoren erfasst und in Koordinaten und Winkel umgerechnet werden müssen, um den Teil der Applikation, die die Visualisationsdaten berechnet, den zu rendernden Blickwinkel sowie die Position des Betrachters mitteilen zu können. (vgl. [Tö10, Kapitel 2.2, Seite 21f.]). Eine zentrale Anforderung dieser Arbeit an die vorgestellten Konzepte ist, diesen *lag* durch den Einsatz von echtzeitfähigen Algorithmen und Techniken gering zu halten – optimalerweise unter der Wahrnehmbarkeitsschwelle. Technisch bedingt ist diese Verzögerung jedoch immer vorhanden. Ein weiteres Problem von *Optical See-Through Displays* stellt die Tatsache dar, dass durch die Projektion des virtuellen Bildes über den halbdurchlässigen Spiegel in jedem Fall das reale Umfeld des Betrachters durchscheint.

- **Video See-Through Displays**: Entgegen dem Ansatz von *Optical See-Through Displays* wird bei diesen Displays das Bild komplett über einen bzw. (häufiger, siehe auch „Head-Mounted Displays“) zwei Monitore angezeigt. Der reale Anteil des Bildes wird hier über eine bzw. zwei Videokameras (eine für jedes Auge) aufgenommen und um die gleiche Zeitspanne verzögert, die benötigt wird, um den virtuellen Anteil des

[45] Zitiert aus [Tö10, Kapitel 2.2, Seite 21f.]

Bildes soweit zu verarbeiten, dass er auf dem Monitor angezeigt werden kann. Durch das Fehlen eines halbdurchlässigen Spiegels kann außerdem für Teile des Bildes nur der virtuelle Anteil angezeigt werden. Obwohl bei dieser Art von Displays kein Unterschied zwischen dem *lag* des realen und des virtuellen Bildes besteht, ist die Anzeige des Gesamtbildes verzögert. Zudem wird je nach Auflösung und Qualität der Kamera(s) beziehungsweise der Monitore der reale Anteil des Bildes in schlechterer Qualität als bei *Optical See-Through Displays* dargestellt (vgl. [Tö10, Kapitel 2.2, Seite 22]).

Diese Konzepte werden zudem in [RF00] verglichen, wobei auch hier die technische Entwicklung seit des Erscheinens der Arbeit im Jahre 2000 berücksichtigt werden muss.

Basierend auf den dargestellten Grundkonzepten lassen sich laut [Tö10, Kapitel 2.2, Seite 22-30] sowie [SVH+02b, Kapitel 3.1, Seiten 4-7] die folgenden Typen von Displays unterscheiden, die in [Tö10] sowie in der einschlägigen Fachliteratur genauer erörtert werden:

- Head-Mounted Displays (HMDs); siehe [Tö10, Kapitel 2.2.1, Seite 23-25]
  - Monokulare Displays mit einem Bildschirm nur für ein Auge, darunter auch Mikrodisplays (siehe [SVH+02b, Kapitel 3.1, Seite 6]
  - Binokulare Displays mit Bildschirmen für beide Augen
- Raum- und umgebungsfixierte Displays; siehe [Tö10, Kapitel 2.2.2, Seite 25-28], darunter:
  - Feststehende Displays
  - Relativ feststehende Displays
- Bewegliche Displays; siehe [Tö10, Kapitel 2.2.3, Seite 28]
- Handheld-Displays; siehe [Tö10, Kapitel 2.2.4, Seite 28f.]
- VRD (Virtual Retinal Display): Hierbei wird das Bild von Laserstrahlen direkt auf die Netzhaut projiziert; siehe [SVH+02b, Kapitel 3.1.2, Seite 5f.]

4.1.1. Stereoskopische und autostereoskopische Displays

Unter den bereits genannten Displays lassen sich Displays, die nur eine zweidimensionale Sicht erlauben, von stereoskopischen Displays abgrenzen, die entweder durch spezielle Brillen (zum Beispiel Shutterbrillen) oder durch autostereoskopische Verfahren, die keine solche Brille benötigen, dreidimensionale Bilder anzeigen. Binokulare HMDs sind prinzipbedingt stereoskopisch (vgl. [Tö10, Kapitel 2.2.1, Seite 23-25]).

4.1.2. Weitere Darstellungsmethoden

Schließlich sei noch darauf hingewiesen, dass aufgrund der obigen Definition von Augmented Reality abgesehen von den bereits genannten Darstellungen durch visuelle Mittel auch die

folgenden nicht visuellen Erweiterungen der Realität sowie Kombinationen möglich sind, die in [Tö10, Kapitel 2.4, Seite 36-41] erörtert werden.

- Akustische Darstellung; siehe [Tö10, Seite 37]
- Taktile bzw. haptische Darstellung; siehe [Tö10, Seite 38f.]
- Gustatorische[46] bzw. olfaktorische[47] Darstellung; siehe [Tö10, Seite 39-41]

Diese Konzepte finden jedoch in der Medizintechnik bisher kaum Anwendung.

5. ARION^{TM} – Beispiel für ein existierendes Visualisationssystem

Wie bereits in Kapitel 2.2 auf Seite 5 beschrieben, existieren bereits jetzt für verschiedene Anwendungsbereiche innerhalb der Medizin verschiedene Visualisationssysteme. Da die Zahl der bestehenden Systeme unüberschaubar groß ist, soll an dieser Stelle exemplarisch das System ARION^{TM} diskutiert werden, das sich auch für den Einsatz als Tumorvisualisationssystem eignet.

Das System ARION^{TM},[48] wurde von der Abteilung für Medizinische und Biologische Informatik sowie der Chirurgischen und Radiologischen Klinik der Universität Heidelberg als Augmented Reality-basierender Prototyp eines intraoperativen Visualisationssystems (IGSS[49]) für die onkologische Leberchirurgie entwickelt. Suthau et al stellen in [SVH+02b] den Einsatz dieses Systems bei der Resektion eines Lebertumors dar. Die Autoren unterscheiden in [SVH+02b] fünf Module des beschriebenen Einsatzes von ARION^{TM}:

- *Modul 1*: Es wurden präoperativ kontrastmittelgestützte CT-Aufnahmen der betroffenen Leber angefertigt, anhand derer mithilfe des Operationsplanungssystem LENA des DKFZ[50] der chirurgische Eingriff geplant wurde. Zudem wurde hier ein mathematisches Modell (Graph) der Gefäße des betroffenen Organs erstellt.
- *Modul 2*: Während des Eingriffs wurden mithilfe von Ultraschallaufnahmen die Gefäßbäume erfasst und ebenfalls dreidimensional als Graph modelliert. Während die Module 2 bis 4 ausgeführt werden, wird die betroffene Leber fixiert.
- *Modul 3*: Die prä- und intraoperativen Gefäßgraphen, werden zueinander registriert[51] – daher können mithilfe eines magnetischen Trackingsystems die in der Planung festgelegten Schritte exakt ausgeführt werden.

[46] Gustatorisch – Den Geschmackssinn betreffend
[47] Olfaktorisch – den Geruchssinn betreffend
[48] ARION^{TM} – Augmented Reality for Intra-Operative Navigation
[49] IGSS – Image-guided surgery system – Bildgeführtes Operationssystem
[50] DKFZ – Deutsches KrebsForschungsZentrum
[51] Registrierung – Prozess, der versucht, die beste Überlagerung zwischen zwei Datensätzen – in diesem Fall Gefäßgraphen – zu finden

- *Modul 4*: Navigationshilfen werden in der Nähe des Tumors in die Leber eingesetzt, um während des Eingriffes Lageveränderungen mitverfolgen zu können und somit die geplanten Schritte auch weiterhin exakt ausführen zu können.
- *Modul 5*: Während der Resektion werden auch die chirurgischen Instrumente mithilfe des zuvor beschriebenen Trackingsystems lokalisiert – diese Informationen werden „in Beziehung zu den transparenten intrahepatischen[52] Strukturen“[53] gesetzt und auf einem Flachbildschirm angezeigt.

ARION erfüllt die in Kapitel 2.1 dargestellten Anforderungen an ein Tumorvisualisationssystem: Die Interaktionsfähigkeit ist beispielsweise durch die Reaktion des Systems bei Änderung der chirurgischen Instrumente gegeben; der intraoperative Teil von ARION™ arbeitet soweit aus [SVH+02b] ersichtlich in Echtzeit. Sowohl der präoperative als auch der intraoperative Teil generieren aus medizinischen, dreidimensionalen Eingabedaten Ausgabedaten, die in ihrer Qualität ausreichend sind, um während einer realen Tumorresektion eingesetzt zu werden. Obwohl in [SVH+02b] nicht explizit erwähnt, legt der Einssatzzweck nahe, dass ARION™ ein Klassifizierungsverfahren für Tumoren implementiert, da der Chirurg für eine erfolgreiche Resektion den Tumor auf dem Display erkennen können muss.

6. Fazit und Ausblick

In dieser Arbeit wurden Konzepte und Methoden zur dreidimensionalen Visualisierung von Tumoren vorgestellt. Der Vergleich von Software- und Hardwarealgorithmen in Kapitel 3.1 sowie die behandelten Plattformen in Kapitel 3.3 zeigten, dass bei der Auswahl der Plattform letztlich immer ein Kompromiss zwischen Kompatibilität und Geschwindigkeit gefunden werden muss, jedoch in der Forschung verschiedenste Ansätze verfolgt werden. Das vorgestellte exemplarische Konzept zur Tumorklassifizierung sowie die Proof-of-Concept-Implementierung VERTEBRA verdeutlichen, dass die Implementierung von einfachen, performanten Klassifizierungsverfahren ohne großen Aufwand möglich ist, jedoch alle Visualisationssysteme von der Qualität der Eingabedaten abhängen sowie nicht beliebig viele Daten in vertretbarer Zeit verarbeiten können. Das in Kapitel 4 vorgestellte Konzept der Augmented Reality zeigt einen der Einsatzbereiche der vorgestellten Algorithmen und Methoden, der Gegenstand vieler aktueller Forschungsprojekte ist. Zudem wurde mit ARION™ ein System vorgestellt, das auf Augmented Reality-Methoden basiert – dieses zeigte schon 2002, dass der Einsatz von Augmented Reality-basierenden Systemen für onkologische Operationen möglich ist. Durch eine weitere Verbesserung der Technik sowie der Algorithmen wird der Einsatz von medizinischen Visualisationssystemen, speziell auch der von Tumorvisualisationssystemen, in den nächsten Jahren weiter steigen und damit chirugische Eingriffe sicherer und effektiver machen.

[52] Intrahepatisch – Innerhalb der Leber liegend
[53] Zitiert aus [SVH+02b, Kapitel 2, Seite 3]

A. Die Begleit-CD

Dieser Arbeit liegt eine Begleit-CD bei, die die folgenden Daten enthält:

- Quellcode von VERTEBRA im Verzeichnis `vertebra` sowie eine Anleitung zur Übersetzung in der Datei `INSTALL` sowie eine kurze Bedienungsanleitung in der Datei `README` in ebendiesem Verzeichnis
- Doxygen[54]-Dokumentation von VERTEBRA als PDF im Verzeichnis `doc` sowie als HTML im Verzeichnis `doc/html`
- LaTeX-Quellcode dieser Seminararbeit sowie die Arbeit als PDF im Verzeichnis `latex`; die Bibliographie, die für ausgewählte Publikation die Zusammenfassungen (Abstracts) enthält, ist im BibTeX-Format in der Datei `latex/visualization.bib` gespeichert.
- Eine Auswahl der in dieser Arbeit verwendeten, frei verfügbaren Publikationen (sowie auch einige wenige weitere, nicht verwendete Publikationen) als PDFs im Verzeichnis `library`

Hinweis: Alle Quellcodekommentare sowie Dokumentationen zu VERTEBRA wurden in englischer Sprache verfasst, da dies eine mögliche Weiterentwicklung des Projektes vereinfachen würde.

Zusätzlich liegt eine Sicherungskopie der Begleit-CD mit identischem Dateninhalt bei, die zur Absicherung gegen technische Defekte dient.

B. Testplattform

Die in dieser Arbeit angeführten Geschwindigkeitsmessungen wurden am 7. November 2010 auf der folgenden Plattform durchgeführt:

- Dell Inspiron 530
- CPU: Intel® Core™2 Duo E8300; 2.83GHz Taktfrequenz
- Betriebssystem: KUbuntu 10.10 x86_64; Kernel 2.6.35-22-generic x86_64
- Grafikhardware: NVidia® GeForce 9400 GT, PCI Express x16
  Treiber 260.19.06 (Konfiguration: High Performance)

[54]Doxygen: Programm, um Dokumentationen aus Quellcode zu extrahieren

C. Verzeichnisse

Abbildungsverzeichnis

Literatur- und Quellenverzeichnis

[AMHH08] Akenine-Möller, Tomas ; Haines, Eric ; Hoffman, Natty: *Real-Time Rendering 3rd Edition*. Natick, MA, USA : A. K. Peters, Ltd., 2008. – 1045 S. – ISBN 987–1–56881–424–7

[Bru04] Bruckner, Stefan: *Efficient Volume Visualization of Large Medical Datasets*. Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Diplomarbeit, Mai 2004. `http://www.cg.tuwien.ac.at/research/publications/2004/bruckner-2004-EVV/`

[Fog10] Fog, Agner: *Optimizing software in C++ - An optimization guide for Windows, Linux and Mac platforms*. `http://agner.org/optimize/optimizing_cpp.pdf` : Web, 09 2010

[GPR+95] Günther, T. ; Poliwoda, C. ; Reinhart, C. ; Hesser, J. ; Männer, R. ; Meinzer, H. P. ; Baur, H. J.: Virim: A massively parallel processor for real-time volume visualization in medicine. In: *Computers & Graphics* 19 (1995), Nr. 5, 705 - 710. `http://dx.doi.org/DOI:10.1016/0097-8493(95)00049-6`. – DOI DOI: 10.1016/0097–8493(95)00049–6. – ISSN 0097–8493

[HMPL+05] Haie-Meder, Christine ; Pötter, Richard ; Limbergen, Erik V. ; Briot, Edith ; Brabandere, Marisol D. ; Dimopoulos, Johannes ; Dumas, Isabelle ; Hellebust, Taran P. ; Kirisits, Christian ; Lang, Stefan ; Muschitz, Sabine ; Nevinson, Juliana ; Nulens, An ; Petrow, Peter ; Wachter-Gerstner, Natascha: Recommendations from Gynaecological (GYN) GEC-ESTRO Working Group (I): concepts and terms in 3D image based 3D treatment planning in cervix cancer brachytherapy with emphasis on MRI assessment of GTV and CTV. In: *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology* 74 (2005), March, S. 235–245. `http://dx.doi.org/10.1016/j.radonc.2004.12.015`. – DOI 10.1016/j.radonc.2004.12.015

[KAB+08] Kutter, O. ; Aichert, A. ; Bichlmeier, C. ; Bichlmeier, Christoph ; Heining, S. M. ; Ockert, B. ; Euler, E. ; Navab, N.: Real-time Volume Rendering for High Quality Visualization in Augmented

Reality. In: *International Workshop on Augmented environments for Medical Imaging including Augmented Reality in Computer-aided Surgery (AMI-ARCS 2008)*. New York, USA, Sept. 2008

[KDH+03] Krempien, Robert C. ; Daeuber, Sascha ; Hensley, Frank W. ; Wannenmacher, Michael ; Harms, Wolfgang: Image fusion of CT and MRI data enables improved target volume definition in 3D-brachytherapy treatment planning. In: *Brachytherapy* 2 (2003), January, S. 164–171. http://dx.doi.org/10.1016/S1538-4721(03)00133-8. – DOI 10.1016/S1538–4721(03)00133–8

[KHBF96] Kuszyk, B.S. ; Heath, D.G. ; Bliss, D.F. ; Fishman, EK: Skeletal 3-D CT: advantages of volume rendering over surface rendering. In: *Skeletal radiology* 25 (1996), Nr. 3, S. 207–214

[KHKH10] Kirk, David B. ; Hwu, Wen-mei W. ; Kirk, David B. ; Hwu, Wen-mei W.: *Programming Massively Parallel Processors: A Hands-on Approach*. 1st. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2010 http://books.google.de/books?id=qW1mncii_6EC. – ISBN 0123814723, 9780123814722

[Kib09] Kibritev, Nikola A.: *Anpassung eines Linux-Betriebssystems an ein FPGA-basiertes eingebettetes Mikroprozessorsystem*, Technische Universität Dresden, Diplomarbeit, May 2009

[Kni95] Knittel, Günter: A scalable architecture for volume rendering. In: *Computers & Graphics* 19 (1995), Nr. 5, 653 - 665. http://dx.doi.org/DOI:10.1016/0097-8493(95)00044-5. – DOI DOI: 10.1016/0097–8493(95)00044–5. – ISSN 0097–8493

[KW03] Kruger, J. ; Westermann, R.: Acceleration techniques for GPU-based volume rendering. In: *Visualization, 2003. VIS 2003. IEEE* IEEE, 2003, S. 287–292

[LCM+05] Leeser, M. ; Coric, S. ; Miller, E. ; Yu, H. ; Trepanier, M.: Parallel-beam backprojection: an FPGA implementation optimized for medical imaging. In: *The Journal of VLSI Signal Processing* 39 (2005), Nr. 3, S. 295–311

[LS87] Leong, Joseph C. ; Stracher, Michael A.: Visualization of internal motion within a treatment portal duringa radiation therapy treatment. In: *Radiotherapy and Oncology* 9 (1987), Nr. 2, 153 - 156. http://dx.doi.org/DOI:10.1016/S0167-8140(87)80203-7. – DOI DOI: 10.1016/S0167–8140(87)80203–7. – ISSN 0167–8140

[MEM02] Mildenberger, Peter ; Eichelberg, Marco ; Martin, Eric: [Without Title]. In: *European Radiology* 12 (2002), 920-927. http://dx.doi.org/10.1007/s003300101100. – ISSN 0938–7994. – 10.1007/s003300101100

[MHS08] Marsalek, L. ; Hauber, A. ; Slusallek, P.: High-speed volume ray casting with CUDA. In: *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* IEEE, 2008, S. 185

[MKW+02] Meissner, M. ; Kanus, U. ; Wetekam, G. ; Hirche, J. ; Ehlert, A. ; Strasser, W. ; Doggett, M. ; Forthmann, P. ; Proksa, R.: VIZARD II: a reconfigurable interactive volume rendering system. In: *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware.* Aire-la-Ville, Switzerland, Switzerland : Eurographics Association, 2002. – ISBN 1–58113–580–7, S. 137–146

[MVHWB05] Maupu, D. ; Van Horn, MH ; Weeks, S. ; Bullitt, E.: 3D stereo interactive medical visualization. In: *Computer Graphics and Applications, IEEE* 25 (2005), Nr. 5, S. 67–71. – ISSN 0272–1716

[OKKS94] Okudera, H. ; Kyoshima, K. ; Kobayashi, S. ; Sugita, K.: Intraoperative CT scan findings during resection of glial tumours. In: *Neurological research* 16 (1994), Nr. 4, S. 265

[OPL+97] Osborne, Rändy ; Pfister, Hanspeter ; Lauer, Hugh ; Ohkami, TakaHide ; McKenzie, Neil ; Gibson, Sarah ; Hiatt, Wally: EM-Cube: an architecture for low-cost real-time volume rendering. In: *HWWS '97: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware.* New York, NY, USA : ACM, 1997. – ISBN 0–89791–961–0, S. 131–138

[PHK+99] Pfister, Hanspeter ; Hardenbergh, Jan ; Knittel, Jim ; Lauer, Hugh ; Seiler, Larry: The VolumePro real-time ray-casting system. In: *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques.* New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1999. – ISBN 0–201–48560–5, S. 251–260

[PHML+06] Pötter, Richard ; Haie-Meder, Christine ; Limbergen, Erik V. ; Barillot, Isabelle ; Brabandere, Marisol D. ; Dimopoulos, Johannes ; Dumas, Isabelle ; Erickson, Beth ; Lang, Stefan ; Nulens, An ; Petrow, Peter ; Rownd, Jason ; Kirisits, Christian: Recommendations from gynaecological (GYN) GEC ESTRO working group (II): Concepts and terms in 3D image-based treatment planning in cervix cancer brachytherapy—3D dose volume parameters and aspects of 3D image-based anatomy, radiation physics, radiobiology. In: *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology* 78 (2006), January, S. 67–77. `http://dx.doi.org/10.1016/j.radonc.2005.11.014`. – DOI 10.1016/j.radonc.2005.11.014

[RF00] Rolland, Jannick P. ; Fuchs, Henry: Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization. In: *Presence: Teleoperators*

and Virtual Environments 9 (2000), Nr. 3, 287-309. http://dx.doi.org/10.1162/105474600566808. – DOI 10.1162/105474600566808

[Ros06] Rost, Randi J.: *OpenGL® Shading Language*. 2. Boston : Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA, 2006

[SK10] Sanders, Jason ; Kandrot, Edward: *CUDA by Example: An Introduction to General-Purpose GPU Programming*. 1. o.O. : Addison-Wesley Professional, 2010 http://www.worldcat.org/isbn/0131387685. – ISBN 0131387685

[SVH+02a] Suthau, T. ; Vetter, M. ; Hassenpflug, P. ; Meinzer, H.P. ; Hellwich, O.: A concept work for Augmented Reality visualisation based on a medical application in liver surgery. In: *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES* 34 (2002), Nr. 5, S. 274–280

[SVH+02b] Suthau, T. ; Vetter, M. ; Hassenpflug, P. ; Meinzer, H.P. ; Hellwich, O.: Konzeption zum Einsatz von Augmented Reality in der Leberchirurgie. (2002)

[THL09] Thomas, D.B. ; Howes, L. ; Luk, W.: A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation. In: *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays* ACM, 2009, S. 63–72

[Tö10] Tönnis, Marcus: *Einführung in die Augmented Reality*. Heidelberg : Springer Verlag, 2010. – ISBN 978–3–642–14178–2

[UHC91] Udupa, Jayaram ; Hung, Hsiu-Mei ; Chuang, Keh-Shih: Surface and volume rendering in three-dimensional imaging: A comparison. In: *Journal of Digital Imaging* 4 (1991), 159-168. http://dx.doi.org/10.1007/BF03168161. – ISSN 0897–1889. – 10.1007/BF03168161

[WS00] Wright, Richard S. ; Sweet, Michael: *OpenGL® SuperBible*. Second Edition. Indianapolis : Waite Group Press (o.O.), 2000. – ISBN 978–1571691644

D. Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel (einschließlich Onlinequellen und elektronischer Medien) benutzt habe.

Ulrich Köhler