UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

SOVAmed
Intelligent visualistics solutions. Made in Germany.
SOVAmed GmbH

Hybrides Raycasting zur optimierten Darstellung von Blutgefäßen

Masterarbeit

zur Erlangung des Grades einer Master of Science (M.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Martina Sekulla

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Dr. Matthias Raspe (SOVAmed GmbH)

Koblenz, im November 2014

SOVAmed GmbH
Universitätsstraße 3
56070 Koblenz

www.sovamed.com

Universität Koblenz-Landau, Campus Koblenz
Fachbereich 4:Informatik, Institut für Computervisualistik
Universitätsstraße 1
56070 Koblenz

www.uni-koblenz-landau.de/koblenz

Aufgabenstellung der Masterarbeit von Martina Sekulla (Matr. Nr. 209210053)

Thema: Hybrides Raycasting zur optimierten Darstellung von Blutgefäßen

In der modernen Medizin stehen zahlreiche Bildgebungsverfahren zur Verfügung, die unterschiedliche Strukturen des menschlichen Körpers erfassen und darstellbar machen. Insbesondere durch tomographische Verfahren wie CT oder MR werden dreidimensionale Aufnahmen ermöglicht. Die Visualisierung dieser Volumendaten ist wichtiger Bestandteil für Diagnose, Therapiekontrolle und Planung chirurgischer Eingriffe. Eine wesentliche Struktur für zahlreiche medizinische Disziplinen bilden Gefäße; dies betrifft sowohl zentrale arterielle Hauptgefäße als auch venöse Verzweigungen in die Seitengefäße.

Direkte Volumenvisualisierungen erreichen mit aktueller Grafikhardware hohe Qualität in Echtzeit, bleiben jedoch in ihrer Auflösung begrenzt. Daneben existieren indirekte Methoden zur expliziten Rekonstruktion von Oberflächen; sie reichen von einfachen echtzeitfähigen Ansätzen bis hin zu hochwertigen, rechenintensiven Systemen auf Basis detaillierter Parametrisierung. Als hybrides Rendering wird die Kombination von direkter und indirekter Visualisierung bezeichnet. Die Integration von Oberflächen- in Volumenrepräsentationen erlaubt es, sich der Vorteile beider zu bedienen und damit Qualität und Darstellungsgeschwindigkeit von 3D-Visualisierungen zu optimieren. Beispiele dafür sind die Vermeidung von visuellen Artefakten durch optimierte Samplingraten ebenso wie die Beschleunigung der Strahlverfolgung durch verbesserte Hüllgeometrie.

Ziel dieser Arbeit ist es, die dreidimensionale Darstellung von Blutgefäßen mithilfe von hybridem Rendering aufzuwerten. Ausgehend von vorhandener Segmentierungsinformation auf Voxelbasis sollen geeignete 3D-Oberflächen dynamisch erzeugt werden, wobei die Erhaltung von Detailstrukturen vorrangig ist. Ein Volumenrendering-System auf Basis von Raycasting soll diese Oberflächendaten zur Steigerung der Bildqualität integrieren und gleichzeitig Beschleunigungen gegenüber reinen Oberflächenrepräsentationen ermöglichen.

Schwerpunkte:

- Einarbeitung in die Thematik (Grundlagen der Verarbeitung und Visualisierung von Volumendaten, Entwicklungsumgebung und relevante medizinische Hintergründe)
- Recherche und Bewertung von Verfahren zur interaktiven Generierung von 3D-Oberflächen sowie zur Kombination mit Raycasting-Verfahren
- Konzeption und Umsetzung eines Prototypen zur Verarbeitung segmentierter Gefäß-CT-Daten
- Dokumentation, Auswertung und Integration der Ergebnisse

Betreuer/Gutachter:

- Dr. Matthias Raspe, SOVAmed GmbH
- Prof. Dr. Stefan Müller, Institut für Computervisuaistik

Beginn der Arbeit: 02.05.2014

Koblenz, den 02.05.2014	Koblenz, den 02.05.2014	Koblenz, den 02.05.2014
Dr. Matthias Raspe	Prof. Dr. Stefan Müller	Martina Sekulla

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☒	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☒	☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Die Medizinische Visualisierung komplexer Gefäßbäume hat das Potential den klinischen Alltag in der Gefäßchirurgie zu erleichtern. Dazu sind exakte, hochaufgelöste Darstellungen und echtzeitfähige Berechnungsmethoden notwendig. Bekannte Ansätze aus den Bereichen der direkten (z.B. Raycasting) und indirekten (z.B. Marching Cubes) Volumenvisualisierung sind nicht in der Lage alle Anforderungen zufriedenstellend zu erfüllen. Verbesserte Ergebnisse können mit hybriden Methoden erzielt werden, die unterschiedliche Visualisierungsverfahren kombinieren.

Im Rahmen dieser Arbeit wurde ein hybrides Renderingsystem zur Darstellung von Blutgefäßen entwickelt, das die Bildqualität durch Integration einer Marching Cubes Oberfläche in ein Raycasting–System optimiert, dabei Detailstrukturen erhält und ausreichende Performanz zur Interaktion bietet. Die Ergebnisse zeigen die verbesserte Plastizität und Genauigkeit der Darstellung. Anhand von Experten– und Laienbefragungen konnte der Nutzen des Systems vor allem für die Patientenaufklärung nachgewiesen werden. Die Erschließung zusätzlicher Anwendungsgebiete ist durch die Weiterentwicklung des Renderers möglich.

Abstract

The medical visualization of complex vascular tree structures has the potential to simplify the everyday clinical practice of vascular surgery. For this purpose, accurate, high–resolution displays and real–time calculation methods are required. Known approaches from the fields of direct (e.g. ray casting) and indirect (e.g. Marching Cubes) volume rendering are not able to meet all the requirements satisfactorily. Improved results can be obtained with hybrid methods which combine different visualization techniques.

In this work a hybrid rendering system for the visualization of blood vessels has been developed which optimizes the overall image quality by integrating a Marching Cubes surface into a ray casting system. Furthermore, detailed structures are preserved and sufficient performance for interaction is provided. The results show the improved plasticity and accuracy of the hybrid display. Based on expert and lay surveys the usefulness of the system was demonstrated primarily for patient education. Additional application areas can be opened up by further development.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Medizinische Visualisierung bezeichnet ein Teilgebiet der wissenschaftlichen Visualisierung (*scientific visualization*) [PB07]. Erfindungen, wie die Computertomographie (CT) oder die Magnetresonanztomographie (MRT), erlauben es Chirurgen und Radiologen anhand von patientenspezifischen 2D–Bildern Diagnosen zu stellen sowie notwendige Behandlungsschritte, insbesondere chirurgische Eingriffe, zu planen [Rob00]. Eine weiterführende Möglichkeit zur Verarbeitung dieser Bilder ist die Berechnung dreidimensionaler Volumenvisualisierungen [LHH08]. Die Intention dahinter ist es, natürlich wirkende Darstellungen aus abstrakten Bilddaten zu erzeugen (s. Abbildung 1). Sie sollen Mediziner dabei unterstützen, den vorliegenden Datensatz zu analysieren und bisher ungeklärte Fragen zum Zustand oder zur Behandlung des Patienten zu beantworten [HMIBG01]. Als Hilfsmittel zur Interpretation der Bilddaten können in die Volumenvisualisierung integrierte Annotationen eingesetzt werden. Beispiele dafür sind die Angabe der Maße pathologischer Strukturen oder die farbliche Hervorhebung kritische Regionen [PB07]. Derartige Präsentations– und Interaktionsmöglichkeiten machen dreidimensionale Darstellungen auch für die Patientenaufklärung, die Beratung unter Kollegen sowie die Aus– und Weiterbildung interessant [PB07, TIP05].

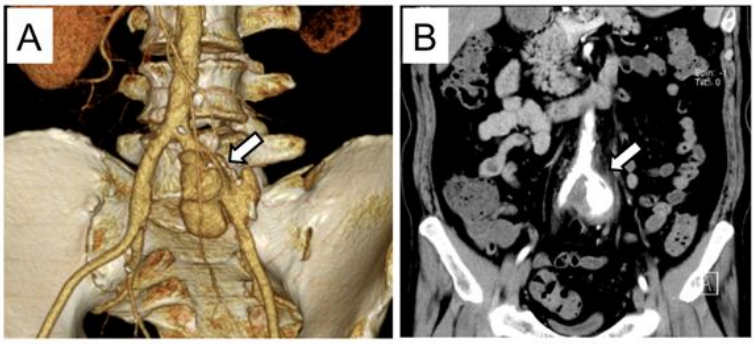


Abbildung 1: Gegenüberstellung von 3D–Visualisierung und koronalem Schichtbild[1] einer kontrastmittelgestützen Computertomographie–Aufnahme (nach Yamashita et al. [YYM+13]). **A:** Volumenvisualisierung eines Aneurysmas der linken Beckenarterie. **B:** Koronales CT–Schichtbild eines Aneurysmas der linken Beckenarterie.

Ein wichtiges Thema in der medizinischen Visualisierung ist die Darstellung vaskulärer Gefäße [PO08]. Der Nutzen dreidimensionaler Volumenvisualisierungen liegt dabei sowohl in der Diagnoseunterstützung, als auch in der präoperativen Eingriffsplanung im Rahmen der Behandlung vaskulärer Erkrankungen, wie beispielsweise Aneurysmen [KGNP12, Oel04]. Als An-

[1] *Koronalebene*: in der Medizin die Vorderseite des Menschen; hintereinander geschichtete Koronalebenen teilen den Körper von vorne nach hinten (s. dazu Abschnitt 2.3.1)

eurysmen werden permanente Gefäßerweiterungen bezeichnet, die aufgrund der steigenden Druckbelastung der Gefäßwand schlimmstenfalls platzen. Im Falle einer solchen Ruptur liegt die Überlebenswahrscheinlichkeit der betroffenen Patienten lediglich bei 10 % [Eck, SL07]. Klinisch von Bedeutung sind vor allem Aneurysmen der Aorta und der Hirnbasisarterien [SL07]. Sie treten in erster Linie bei Männern über 65 Jahren, vorwiegend bei Rauchern, auf [GKW08, KGJ10]. Die Wahrscheinlichkeit an einem Aneurysma zu erkranken steigt mit zunehmendem Alter [SL11]. Mit Hinblick auf die anhaltende Alterung der Bevölkerung in heutigen Industrienationen ist deshalb zwangsläufig auch eine Häufung der Aneurysma–Erkrankungen zu erwarten [KGJ10].

Um die Diagnose und Therapieplanung für vaskuläre Erkrankungen optimal unterstützen zu können, sind hochaufgelöste Darstellungen und echtzeitfähige Visualisierungsmethoden notwendig. Bisherige Ansätze konnten jeweils nur eines der beiden Kriterien zufriedenstellend erfüllen.

1.1 Motivation

Die Volumenvisualisierung vaskulärer Gefäße ist in mehrfacher Hinsicht ein komplexes Thema. Sie erfordert zum einen die Möglichkeit schmale Strukturen und topologische Verzweigungen präzise aus CT– und MRT–Datensätzen extrahieren zu können [KGNP12]; zum anderen sind aufgrund der Größe qualitativ hochwertiger Datensätze effiziente Verfahren für die Verarbeitung notwendig [PB07]. Werden interaktive Geschwindigkeiten erzielt, können Benutzer rundum die Volumendarstellung navigieren und den Fokus damit auf wesentliche Strukturen lenken [Rob00].

Es existieren diverse Ansätze, die für die Visualisierung von Blutgefäßen verwendet werden. Sie lassen sich in direkte Visualisierungsmethoden (DVR–Verfahren), wie beispielsweise *Raycasting*, und indirekte Visualisierungsmethoden (IVR–Verfahren), wie *Marching Cubes* und *MPU Implicits*, unterteilen. Jedes dieser Verfahren bildet einen Kompromiss aus hochaufgelöster Darstellung, präziser Abbildung von Detailstrukturen und Echtzeitfähigkeit.

Mithilfe von GPU–basierten DVR–Verfahren, die optimierte Datenstrukturen (z.B. Octrees für *Empty Space Skipping*, s. dazu Abschnitt 3.1.4) nutzen, sind hochqualitative Darstellungen in Echtzeit möglich. Allerdings ist die Auflösung der generierten Bilder aus technischen Gründen begrenzt [PO08, KGNP12]. Im Gegensatz dazu erzeugen IVR–Verfahren Darstellungen mit variabler Auflösung [PB07]. Zudem bieten sie Interaktionsmöglichkeiten, die über simple Navigation hinausgehen, wie beispielsweise die Integration von grafischen Annotationen durch Selektion eines Objekts [Sch06a].

IVR–Verfahren arbeiten in der Regel auf Segmentierungsergebnissen. Der zusätzliche Vorverarbeitungsschritt erhöht den Datenverlust und erzeugt somit im Vergleich zu DVR–Verfahren zwangsläufig weniger präzise Darstellungen [PB07, Sch06a]. Zwar können unter den IVR–Verfahren genauere und weniger genaue Ansätze unterschieden werden, allerdings wird Genauigkeit

dabei auf Kosten der Performanz oder der Bildqualität erzielt.

Hybride Visualisierungsverfahren haben das Potenzial die dreidimensionale Volumenvisualisierung durch geeignete Kombination von DVR– und IVR–Verfahren zu verbessern. Indem die jeweiligen Stärken der beiden Verfahrensklassen gleichzeitig genutzt werden, lassen sich Schwächen in den Einzelmethoden kompensieren. Dadurch ist es möglich, die Gesamtqualität einer 3D–Visualisierung zu optimieren. Dies betrifft insbesondere komplexe Strukturen, wie verzweigte, stellenweise schmale Gefäßbäume. Sie profitieren von der variablen Auflösung und den Interaktionsmöglichkeiten der IVR–Verfahren ebenso wie von den exakten Darstellungen schmaler Strukturen und Verzweigungen, die bei ausreichender Abtastrate mit DVR–Verfahren erzielt werden können. Die Schwierigkeit in der Entwicklung hybrider Systeme liegt in einer geeigneten Kombination direkter und indirekter Visualisierungen, ohne dass optisch sichtbare Brüche und Übergänge entstehen.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung und Umsetzung eines Konzepts zur optimierten Darstellung von Blutgefäßen durch hybrides Raycasting. Mithilfe von modernen Hardware–Technologien, wie Shader–Programmierung und GPGPU, sollen die von medizinischen Bildgebungssystemen erzeugten Datensätze in detaillierte Volumenvisualisierungen überführt und dabei interaktive Geschwindigkeiten erzielt werden. Dazu wird von einem CT–Datensatz inklusive der zugehörigen Gefäßsegmentierung ausgegangen. Anhand der Segmentierungsinformationen soll eine präzise Oberflächendarstellung durch indirekte Visualisierung generiert werden. Ein Raycasting–System soll die Oberflächendaten integrieren, um auf diese Weise eine gesteigerte Bildqualität zu erzielen.

Zur Beurteilung des hybriden Renderingsystems im Hinblick auf Genauigkeit werden Vergleiche mit Visualisierungen angestellt, die mittels Raycasting direkt auf dem CT–Datensatz generiert wurden. Zusätzlich wird die Performanz des Systems unter *Mac OS X* und *Windows* analysiert. Die Anwendungsmöglichkeiten sowie die Verständlichkeit der Visualisierung werden anhand einer Experten– und einer Laienbefragung eingeschätzt. Die Ergebnisse dieser Arbeit sollen das Potential hybrider Visualisierungen für die Anwendung im klinischen Bereich bewerten.

Die nachfolgenden Kapitel sind inhaltlich wie folgt aufgeteilt:

Kapitel 2 gibt einen Überblick über die relevanten medizinischen Grundlagen. Die Morphologie gesunder Blutgefäße wird krankhaften anatomischen Veränderungen gegenübergestellt. Zusätzlich werden die Möglichkeiten zur Diagnose und Therapie vaskulärer Erkrankungen durch Bildgebung gegen-

einander abgewogen. Im Rahmen dessen wird die Computertomographie näher erläutert.

Kapitel 3 stellt Verfahren zur direkten und indirekten Volumenvisualisierung vor. Basierend darauf wird die Verwendung hybrider Visualisierungsmethoden motiviert. Bisherige Ansätze, die Raycasting und Oberflächendarstellungen miteinander kombinieren, werden dargelegt.

Kapitel 4 entwickelt das Konzept zur hybriden Visualisierung von Blutgefäßen. Indirekte Volumenvisualisierungemethoden werden hinsichtlich ihrer Eignung zur Kombination mit Raycasting verglichen. Anhand definierter Anforderungen wird die Gefäßvisualisierungspipeline für die im nachfolgenden Kapitel beschriebene Implementierung entworfen.

Kapitel 5 erläutert die prototypische Umsetzung des hybriden Renderingsystems. Das modulare Rahmenprogramm, die Generierung der Oberflächendarstellung durch GPGPU, das Raycasting im Fragment–Shader sowie die Komposition beider Visualisierungen werden dargestellt. Das Kapitel enthält eine Auflistung der verwendeten Werkzeuge und Bibliotheken.

Kapitel 6 zeigt die Performanz des hybriden Renderingsystems auf. Die Bildqualität wird im Vergleich zu einer reinen Raycasting–Umsetzung bemessen. Die Ergebnisse einer Experten– und einer Laienevaluation mit Fokus auf Verständlichkeit und Anwendbarkeit der hybriden Visualisierung werden dargelegt.

Kapitel 7 bewertet das entwickelte hybride Renderingsystem hinsichtlich der im vorangegangenen Kapitel beschriebenen Ergebnisse. Möglichkeiten der Integration des Visualisierungsverfahrens in bestehende Systeme werden diskutiert.

Kapitel 8 fasst die Ergebnisse der Arbeit in einer abschließenden Betrachtung zusammen und regt im Ausblick zu weiterführenden Entwicklungen an.

2 Medizinischer Hintergrund

2.1 Morphologie vaskulärer Gefäße

Als wichtigste Transportwege des menschlichen Körpers versorgen Blutgefäße alle Zellen mit Sauerstoff und Nährstoffen und dienen gleichzeitig dem Abtransport von Stoffwechselendprodukten. Innerhalb des Herz–Kreislauf–Systems, auch kardiovaskuläres System genannt, werden diverse Arten von Blutgefäßen unterschieden (s. Abbildung 2). Die größte Schlagader des Körpers ist die **Aorta** [Huc07]. Sie führt von der linken Herzkammer aufsteigend als *Aorta ascendens* in den Aortenbogen (*Arcus aortae*) und verläuft dann absteigend als *Aorta thoracica* durch den Brustraum und als *Aorta abdominalis* durch den Bauchraum [Huc07, FS08, BRMR01]. Von der gesamten Aorta aus zweigen paarige **Arterien** zur Versorgung paariger Organe, wie beispielsweise der Nieren und Nebennieren, ab. Der Verdauungskanal mit seinen Drüsen, die Leber und die Milz werden von drei einzelnen Abzweigungen aus versorgt. Die Aorta endet auf Höhe des vierten Lendenwirbels, wo sie sich in zwei paarige Beckenarterien teilt [BRMR01].

Alle Arterien verzweigen in weitere kleine Verästelungen, die **Arteriolen** [Huc07]. Diese münden in den hauchdünnen **Kapillaren**, deren durchlässige Wand den Austausch von Sauerstoff, Nährstoffen und Stoffwechselendprodukten zwischen Gewebe und Blut ermöglicht [Huc07, Sch06b].

Von den Kapillaren aus strömt das Blut in die **Venolen**, die zu immer größeren **Venen** zusammengeführt werden. Sie münden letztlich in der oberen und unteren Hohlvene, der **Vena cava superior** und **inferior**. Durch diese fließt das Blut in den rechten Herzvorhof zurück. [Huc07]

Mit Ausnahme der Kapillaren werden alle genannten Blutgefäße aus drei Wandschichten aufgebaut:

- Der innere Hohlraum eines Gefäßes, das Gefäßlumen, wird von der *Tunica intima* umfasst. Diese besteht an der Grenzfläche zum Blut aus flachen, glatten Endothelzellen und schließt nach außen mit einer Schicht elastischer Fasern ab. [Sch06b, Huc07, KPKS10]

- Die mittlere Gefäßschicht, die *Tunica media*, enthält sowohl glatte Muskelzellen als auch Schichten elastischer Fasern. Sie wird durch eine elastische Membran von der äußeren Schicht der Gefäßwand getrennt. [Sch06b, Huc07, KPKS10]

- Die *Tunica externa*, auch *Adventitia* genannt, verbindet das Gefäß nach außen mit dem umliegenden Gewebe. Sie ist aus elastischen Fasern und lockerem Bindegewebe zusammengesetzt. Größere Gefäße enthalten in der *Tunica externa* selbst Nerven sowie kleinere Gefäße, die die Arterienwand versorgen. [Sch06b, Huc07, KPKS10]

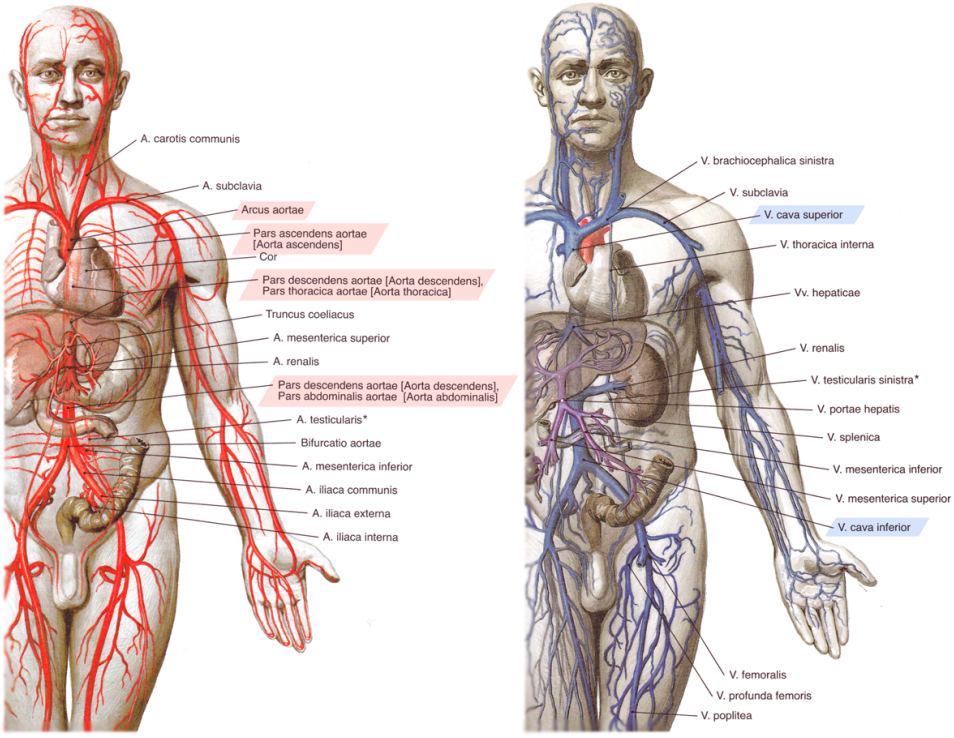


Abbildung 2: Darstellung des menschlichen Gefäßsystems (modifiziert nach Putz und Pabst [PP07]). **A:** Darstellung der Aorta, abgehender Arterien und Arteriolen. **B:** Darstellung der Vena cava, hinführender Venen und Venolen.

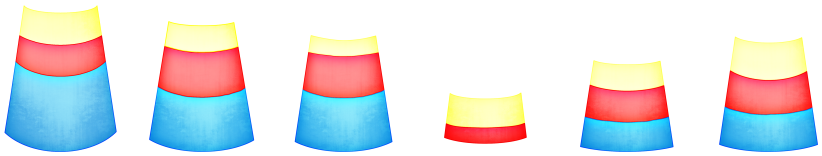
	Aorta	Arterie	Arteriole	Venole	Vene	V. cava
w	2,5mm	1mm	20μm	10μm	0,5mm	1,5mm
r_i	12,5 mm	2mm	20μm	30μm	2,5mm	15mm
w/r_i	0,2	0,5	1,0	0,3	0,2	0,1
						

Tabelle 1: Schematische Darstellung der relativen Wandzusammensetzung verschiedener Blutgefäße (nach Klinke et al. [KPKS10]). w/r_i bezeichnet das Verhältnis zwischen der Dicke der Gefäßwand (w) und dem Innenradius (r_i) des Gefäßes. In der untersten Zeile ist die relative Wandzusammensetzung aus Kollagen (gelb), glatter Muskulatur (rot) und Elastin (blau) dargestellt.

Tabelle 1 zeigt, dass trotz des grundsätzlich gleichen Aufbaus der Gefäßwand in allen Abschnitten des Herz–Kreislauf–Systems die Zusammensetzung der Gefäßwand variiert. So sind beispielsweise Arterien im Vergleich zu Venen relativ dickwandig und auch muskelstärker [KPKS10, SOBP07]. Dagegen ist der Innenradius von Venen verglichen zu Arterien leicht vergrößert. Dabei ist zu beachten, dass Blutgefäße nur im Idealzustand einen kreisrunden Querschnitt aufweisen und er insbesondere bei Venen unter äußerem Druck ellipsoid wird [SOBP07].

Erkrankungen der Blutgefäße führen im Allgemeinen zu einer Veränderung ihres Aufbaus und ihrer Struktur. Dies betrifft insbesondere den Gefäßdurchmesser sowie den Querschnitt. Beispiele morphologisch erkennbarer Gefäßerkrankungen sind die Arteriosklerose, das Aneurysma und die Aortendissektion.

Arteriosklerose. Die Arteriosklerose, im Volksmund auch "Verkalkung" genannt, ist eine generalisierte[2], degenerative[3] Erkrankung der großen Arterien [Huc07, SL11]. Sie führt zur zunehmenden Verengung des Gefäßlumens bis zur Entstehung sogenannter Stenosen und löst eine Regulationsstörung der Spannkraft der Gefäßwand aus. Infolgedessen tritt eine Mangeldurchblutung auf, die sowohl Gewebe als auch Organe betrifft [Huc07]. Die bedeutendste Form der Arteriosklerose ist die **Atherosklerose**, die durch eine Verdickung der *Tunica intima* gekennzeichnet ist [Huc07, SL07].

Die Arteriosklerose bedingt zahlreiche Herz–Kreislauf–Erkrankungen, die zu den häufigsten Todesursachen in heutigen Industriestaaten zählen [Huc07, SL11].

Aneurysma. Die dauerhafte Erweiterung einer Arterie oder einer Herzhöhle wird als Aneurysma bezeichnet [Huc07, SL11, SL07, Eck]. Es kann ausgehend von einer Anlagestörung, einer Atherosklerose in Kombination mit einer Hypertonie[4] oder einer chronischen Entzündung der Gefäßwand entstehen. Besonders kritisch sind Aneurysmen in den Hirnbasisarterien und in der Aorta [SL07]. Aortale Aneurysmen werden anhand ihrer Lage im menschlichen Oberkörper klassifiziert (s. Abbildung 3). In etwa 85% der auftretenden Fälle von Aortenaneurysmen handelt es sich um abdominale Aneurysmen; nur 15% sind thorakale Aneurysmen [Huc07].

Morphologisch wird zwischen dem echten Aneurysma (*Aneurysma versum*) und dem falschen Aneurysma (*Aneurysma spuridum*) unterschieden (s. Abbildung 4A, B). Während es sich beim echten Aneurysma um eine Gefäßerweiterung aller drei Wandschichten handelt, bezeichnet das falsche

[2] *Generalisierung*: Allgemeinerkrankung, die den gesamten Körper oder mindestens ein Organsystem betrifft

[3] *Degeneration*: Oberbegriff für zumeist strukturelle oder funktionale Abweichungen; in der Regel im Sinne einer Rückbildung von Organen oder Gewebe

[4] *Hypertonie*: Bluthochdruck

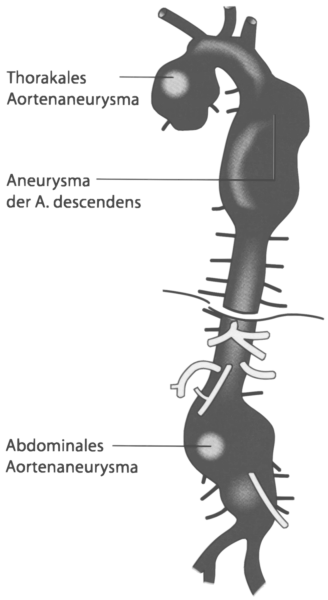


Abbildung 3: Klassifizierung von Aneurysmen anhand ihrer Lage im menschlichen Oberkörper (nach Steffel und Lüscher [SL11]).

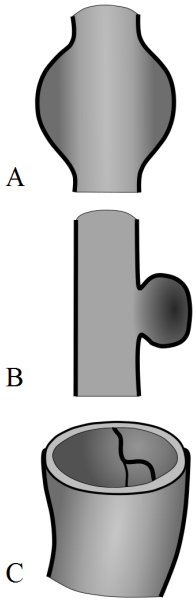


Abbildung 4: Klassifizierung von Aneurysmen anhand ihrer Morphologie.**A:** Aneurysma versum. **B:** Aneurysma spuridum. **C:** Aneurysma dissecans.

Aneurysma ein vor oder hinter der Gefäßwand gelegenes Hämatom. Dieses Hämatom hat seinen Ursprung in einer Verletzung der gesamten Gefäßwand und wird mit der Zeit von einer Bindegewebskapsel umschlossen. Betrifft die Gefäßwandverletzung lediglich die inneren Schichten, kommt es zu einem Eindringen des Bluts in die Gefäßwand, bezeichnet als **Aortendissektion** (*Aneurysma dissecans*, s. Abbildung 4C). [SL11]

Von einem echten Aneurysma wird ab einer Erweiterung von 50% im Vergleich zur Norm gesprochen [SL11, Eck]. Die jährliche Wachstumsrate steigt kontinuierlich mit dem Durchmesser des Aneurysmas [KGJ10]. Mit zunehmender Aussackung wächst auch das Risiko für einen Geweberiss, die sogenannte Ruptur [SL11, KGJ10]. Ihre Wahrscheinlichkeit nimmt in Abhängigkeit von der Wandspannung W zu. Diese lässt sich anhand des Laplace–Gesetzes aus dem Blutdruck P, dem Gefäßradius r und der Wanddicke h ableiten [SL11]:

$$W = P \cdot r/2h \tag{1}$$

Beträgt der Durchmesser eines Aortenaneurysmas mehr als 5 cm folgt aus (1) ein Rupturrisiko von etwa 10% pro Jahr [SL07]; bei über 7 cm liegt das Risiko bei bis zu 33% [SL11]. Für die betroffenen Patienten endet eine Ruptur in 90% der Fälle tödlich [Eck].

	Screening	Diagnose	Messung	Kontrolle (Aneurysmengröße)	Kontrolle (Stentgraft[5])
US	+++	++	(+)	++	0
DSA	0	+	(+)	0	(+)
MRT	++	++	+	++	(+)
CT	++	+++	+++	++	+++

Tabelle 2: Vergleich der zweidimensionalen Bildgebungsverfahren Ultraschall (US), Digitale Subtraktionsangiographie (DSA), Magnetresonanztomographie (MRT) und Computertomographie (CT) im Kontext der Diagnostik und Behandlung infrarenaler Aneurysmen und Beckenarterienaneurysmen [GKW08]

2.2 Diagnose und Therapie von Gefäßerkrankungen

Die in Abschnitt 2.1 vorgestellten Gefäßerkrankungen bleiben klinisch in der Regel unauffällig. So verläuft die Arteriosklerose bis zu deutlichen Verengungen ohne für den Patienten bemerkbare Symptome. Aneurysmen können zwar Kompressionen umliegender Organe verursachen, die dann in Bauchschmerzen, Rückenschmerzen, Atemnot oder Heiserkeit resultieren, Schmerzen an der Aorta werden allerdings erst durch Dehnung oder Riss der *Tunica externa* ausgelöst. Die Diagnose erfolgt häufig erst mit der Ruptur, wenn das Aneurysma zuvor nicht im Rahmen einer Vorsorgeuntersuchung oder zufällig entdeckt wird [SL11]. Im Folgenden werden die Möglichkeiten zur Diagnose und Therapie von Gefäßerkrankungen mittels medizinischer Bildgebung am Beispiel von Aortenaneurysmen erläutert (s. Tabelle 2).

Die **Sonographie** ist besonders geeignet für Vorsorgeuntersuchungen, die Diagnose sowie die Verlaufskontrolle von noch nicht versorgungspflichtigen abdominalen Aneurysmen und Beckenarterienaneurysmen [GKW08, SL11]. Infrarenale Aneurysmen sind mittels Ultraschall mit einer Sensitivität und Spezifität von über 90% diagnostizierbar. Auch Rupturen lassen sich auf diese Weise gut identifizieren, ohne den Patienten einer Strahlenbelastung aussetzen zu müssen. Betreffen die Gefäßveränderungen allerdings die Nierenarterien, ist eine Ultraschalluntersuchung kaum noch aussagekräftig. Schwierig erkennbar sind außerdem kleinere Beckenarterienaneurysmen [GKW08]. Im Brustraum ist eine Sonographie nicht möglich [KGJ10, Eck].

Eine gute Darstellung sämtlicher Gefäßabgänge liefert die **Digitale Subtraktionsangiographie (DSA)**. Sowohl Stenosen als auch Dissektionen sind mittels DSA deutlich erkennbar [KGJ10]. Aneurysmen hingegen lassen sich nur indirekt bestimmen, da in dieser Form der Kontrastmitteldarstellung lediglich das durchflossene Gefäßlumen und nicht die gegebenenfalls erwei-

[5] *Stentgraft*: röhrenförmige Prothese zur Positionierung in Blutgefäßen; dient der Stabilisierung der Gefäßwand zur Verhinderung von Rupturen [Eck, KGJ10]

terte Gefäßwand sichtbar wird. Dennoch bietet die DSA Möglichkeiten zur Planung von komplizierten vaskulären und endovaskulären Eingriffen und kann gegebenenfalls Anwendung in den darauffolgenden Nachuntersuchungen finden [GKW08].

Eine weitere Möglichkeit zur kontrastmittelgestützten Darstellung von Gefäßen bildet die **Magnetresonanztomographie (MRT)**. Neben dem gefärbten Gefäßlumen wird in dieser Darstellung zusätzlich der Kontext, insbesondere auch die Aortenwand sichtbar. Der Grad der Verkalkung lässt sich mittels MRT jedoch nicht bestimmen [KGJ10]. Auch eine Messung der Aneurysmengröße ist in der MRT–Darstellung nicht präzise möglich [GKW08]. Aus diesem Grund findet sie vorwiegend bei der präoperativen Beurteilung der Durchblutung von thorakalen Aneurysmen als Ausweichmethode zum Ultraschall Anwendung [KGJ10]. Ist ferner eine Strahlenbelastung des Patienten zu vermeiden, kann ein MRT außerdem anstelle einer Computertomographie herangezogen werden [Eck].

Die **Computertomographie (CT)** stellt derzeit die beste Methode zur Diagnose von Aneurysmen und zur Nachsorge im Anschluss an eine operative Behandlung dar. Das CT ermöglicht eine exakte Vermessung der Gefäße. Durch Kontrastmittelgabe wird das Gefäßlumen sichtbar; zusätzlich werden auch Verkalkungen abgebildet [GKW08, KGJ10]. CT-Angiographien werden in der Regel vor jedem endovaskulären Eingriff aufgenommen [GKW08]. Die Schichtdicke der CT–Aufnahme sollte 3 mm nicht überschreiten [GKW08, Eck].

Aufbauend auf den vorgestellten Bildgebungsmethoden bietet die Computergrafik mittlerweile dreidimensionale Verfahren zur Weiterverarbeitung und Analyse der zweidimensionalen (Schicht–)Bilder. Die **Finite Elemente Analyse (FEA)** ermöglicht die Berechnung und Simulation hämodynamischer Kräfte individuell für jeden Patienten. Zwar sind die Parameter zur Bestimmung des Rupturrisikos noch nicht trivial berechenbar, jedoch besitzt die FEA das Potential zur Lösung dieses Problems. [Eck]

Eine weitere Möglichkeit zur dreidimensionalen Darstellung von Gefäßen bietet die **Volumenvisualisierung**. Sie kann um zusätzliche Informationen zur individuellen Therapieplanung erweitert werden und ist speziell bei der Behandlung komplexer krankhafter Gefäßveränderungen von Nutzen. [Eck]

2.3 Grundlagen der Computertomographie

Grundsätzlich ist die dreidimensionale Visualisierung von Bilddaten nicht an CT–Aufnahmen gebunden, sondern kann auch aus MRT–Darstellungen gewonnen werden. Allerdings bietet das CT im Kontext der Diagnostik und Behandlung von Erkrankungen des kardiovaskulären Systems einige in Abschnitt 2.2 bereits genannte Vorteile. Auch seine gängige Anwendung in der Therapie von Gefäßerkrankungen, speziell Aneurysmen, macht das CT für dreidimensionale Gefäß–Visualisierungen interessant.

Die in dieser Arbeit entwickelte Volumenvisualisierung zur optimierten Darstellung von Blutgefäßen basiert auf CT–Datensätzen. Die nachfolgenden Kapitel geben einen Überblick über die Gewinnung, Interpretation und Beurteilung solcher Bilddaten.

2.3.1 Funktionsweise

Die Computertomographie basiert auf der Röntgendiagnostik, liefert im Gegensatz zu dieser jedoch volumetrische Objektrepräsentationen in Form von axial überlagerungsfreien Schnittbildern des menschlichen Körpers [PB07, Buz04]. Dazu rotiert ein Emitter–Detektor–System im CT–Rahmen, der sogenannten Gantry, um das zu scannende Objekt. Bei heutigen Spiral–CTs wird das Objekt gleichzeitig durch das System geschoben [PB07]. Auf diese Weise wird es unter verschiedenen Projektionswinkeln durchleuchtet (s. Abbildung 5). Das Ergebnis sind einzelne Röntgenbilder gleicher Auflösung, sogenannte Projektionsprofile. Sie zeigen an, wie stark die von den Emittern ausgehende Röntgenstahlung durch das Objekt abgeschwächt wird. Durch Schichtung aller Bilder entsteht ein volumetrischer Datensatz [PB07, Buz04].

Für die Aufnahme von CT–Bildern ist eine moderate bis hohe Röntgenstrahlungsdosis nötig, die in der Regel mit der Bildauflösung zunimmt. Kontrastmittel dient der Darstellung des Gefäßlumens und einer stärkeren Hervorhebung von Organen [PB07]. Weitere Parameter in der Aufnahmeplanung sind unter anderem die Schichtdicke, die Voxeldistanzen[6] zwischen den einzelnen Messwerten (s. dazu Abschnitt 3.1.1) sowie die Neigung der Gantry [PB07, Buz04].

Die CT–Aufnahme stellt primär die axialen Schichten des Volumens dar.

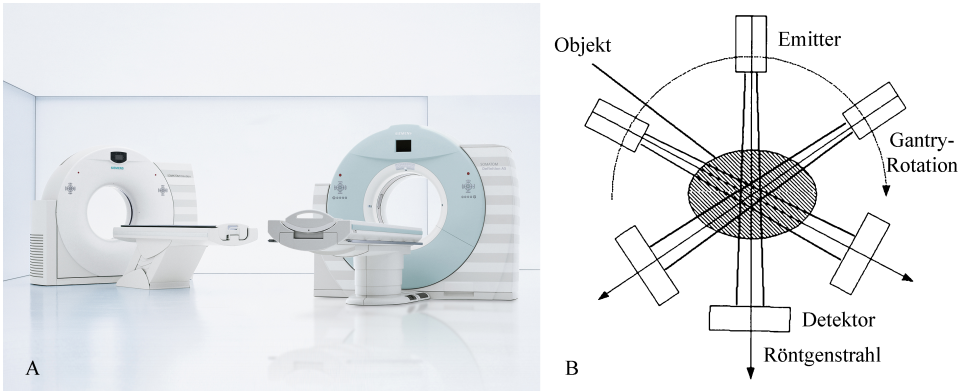


Abbildung 5: Computertomographie. **A:** Computertomograph von Siemens [Sie]. **B:** Schemazeichnung eines rotierenden Emitter–Detektor–Systems (nach Preim und Bartz [PB07]).

[6] *Voxel* (= volume element): Punkt in einem dreidimensionalen Gitter; 3D–Pendant zu einem Pixel in einem zweidimensionalen Gitter [UH00]

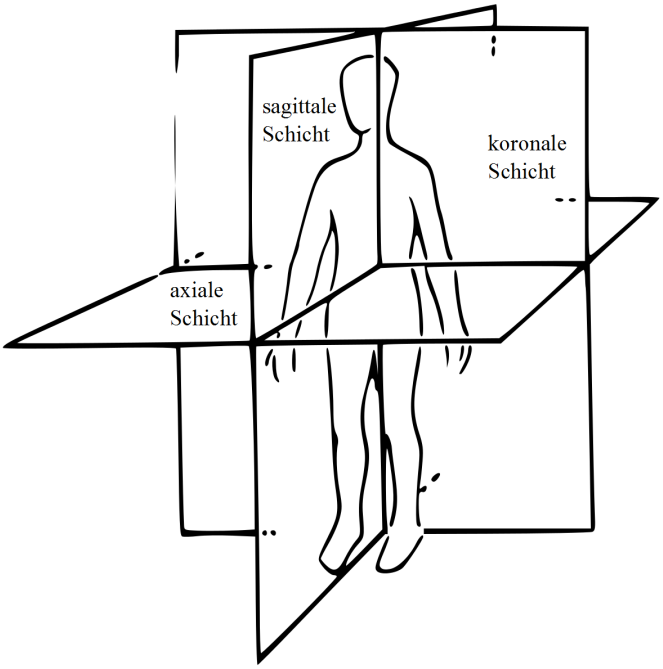


Abbildung 6: Darstellung der axialen, koronalen und sagittalen Schicht (modifiziert nach Fosanelli [Fos02]).

Die sagittalen und koronalen Schichten lassen sich daraus mithilfe von *Multi Planar Reformating* (MPR) interpolieren. Abbildung 6 zeigt die drei orthogonalen Hauptschichten und ihre Lage im Raum. [Buz04]

2.3.2 Darstellung und Interpretation der Daten

CT–Daten sind Volumendatensätze, die aus einer Reihe individueller Graustufenbilder zusammengesetzt werden. Jedes Graustufenbild wird aus einem vom CT aufgenommenen Projektionsprofil abgeleitet. [PB07]

Im Projektionsprofil repräsentiert jeder gemessene Intensitätswert die Dichte des gescannten Objekts am abgetasteten Ort. Anhand der Werte lassen sich Knochen, Organe und sonstige Gewebe unterscheiden. Dazu werden die Intensitätswerte in eine andere Einheit, die *Hounsfield Unit (HU)*, transformiert. In der **Hounsfield–Skala** wird Wasser auf 0 HU abgebildet. Luft liegt bei -1000 HU [PB07, Buz04]. Abbildung 7 zeigt, dass der unmittelbare Unterschied zwischen den diversen menschlichen Organen und Wasser gemessen in HU nur geringfügig ausfällt; die Wertebereiche für einige Organe sind sogar überlappend. Aus diesem Grund ist der Rückschluss vom HU–Wert auf das dargestellte Organ nicht immer leicht, sondern benötigt häufig eine differenzierte Transferfunktion (s. dazu Abschnitt 3.1.2) [Buz04].

Der Wertebereich der Hounsfield–Skala ist nach oben prinzipiell unbe-

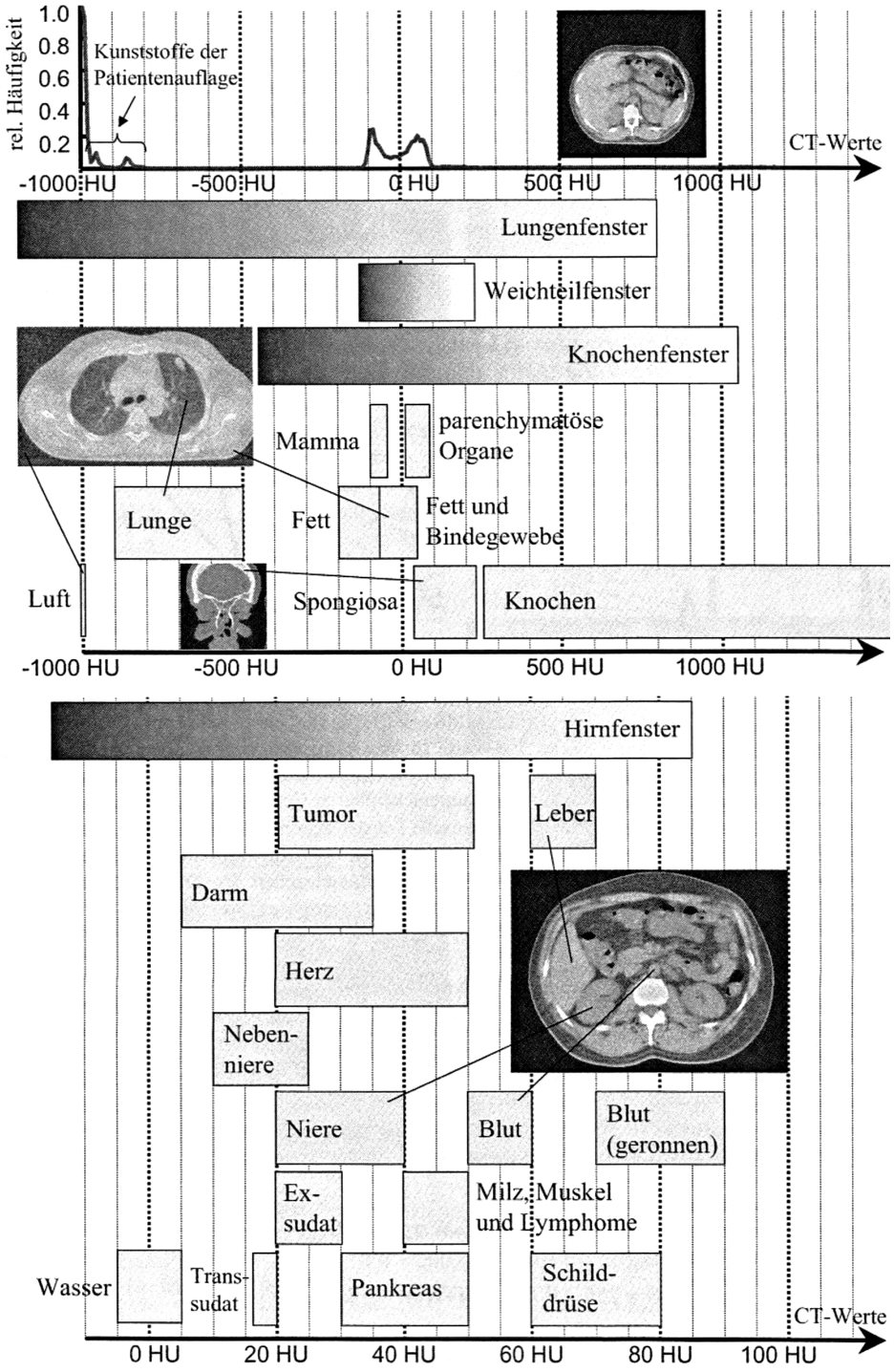


Abbildung 7: Einordnung einzelner Organe innerhalb der Hounsfield–Skala (nach Buzug [Buz04]).

schränkt. In der Praxis werden 3000 HU jedoch nicht überschritten. 12 Bit Graustufentiefe ermöglichen die Abbildung von bis zu 4000 HU und sind damit für CT–Datensätze ausreichend [Buz04]. Üblich ist die Arbeit mit 16 Bit (2 Byte) Graustufentiefe, da so der Voxelzugriff erleichtert wird [PB07].

2.3.3 Beurteilung der Bildqualität

Im Vergleich zum ursprünglichen Röntgenverfahren lokalisiert das CT anatomische Strukturen auch in der Tiefe und ermöglicht durch seine hohe Sensitivität quantitative Messungen [PB07]. Problematisch dabei ist die Abhängigkeit der Bildqualität von technischen Größen, wie beispielsweise dem Signal–Rausch–Verhältnis. Ungeeignete Parametrisierungen können diverse Artefakte im Bild erzeugen. Auch ein Fehlverhalten des zu scannenden Patienten sowie die Streuung der Röntgenstrahlung durch Metallimplantate haben negative Auswirkungen auf die Bildqualität. Zur Beurteilung der Darstellung ist die Frage entscheidend, inwiefern die Abbildung gegebenenfalls trotz vorkommender Artefakte diagnostisch relevante Strukturen erkennen lässt [Buz04].

Ein gängiger Fehler in CT–Aufnahmen ist das sogenannte **Partialvolumenartefakt**. Aufgrund der beschränkten Auflösung des Detektorsystems verschmieren eigentlich scharfe Kontrastkanten. Infolgedessen entstehen unerwünschte Streifen im Bild. Der Fehler kann sowohl zweidimensional innerhalb einer axialen Schicht als auch dreidimensional in z–Richtung auftreten und ist besonders störend in Regionen mit einer Vielzahl stark kontrastierter Details. [Buz04]

Aufhärtungsartefakte entstehen in Abhängigkeit vom Härtegrad der Röntgenstrahlung. Weiche Strahlung führt zu stärkerer Absorption durch das zu scannende Objekt, harte Strahlung bewirkt eine schwächere Absorption. Wird der Härtegrad nicht optimal eingestellt, resultiert dies in Verschmierungen, die sich über das gesamte Bild ausbreiten können. Eine rechnerische Korrektur ist lediglich für Gewebe möglich, die einen ähnlichen Intensitätswert wie Wasser aufweisen. Aus diesem Grund sind Aufhärtungsartefakte in erster Linie in Knochendarstellungen sichtbar. [Buz04]

Ein weiterer Parameter, der direkten Einfluss auf die Bildqualität ausübt, ist die Abtastrate. Aufgrund von Unterabtastung des zu scannenden Objekts durch den Computertomographen entstehen **Abtastartefakte** in Form von Bildrauschen (*Aliasing*). [Buz04]

Im Gegensatz zu diesen technisch bedingten Störungen resultieren **Bewegungsartefakte** aus vermeidbaren und unvermeidbaren Regungen des zu scannenden Patienten. Sowohl Bewegungen des Körpers als auch Atmung und Herzschlag führen zu Veränderungen in den einzelnen Schichtbildern. In aufeinanderfolgenden Schichten erscheinen gleiche Objekte fälschlicherweise nicht mehr in der gleichen Lage und Größe. In einer Volumenvisualisierung treten in diesem Fall Treppenstufeneffekte auf. [Buz04]

Sehr starke Störungen, die sich streifenförmig über das gesamte Bild ausbreiten, sind **Metallartefakte**. Ursächlich sind metallische Implantate, wie künstliche Hüftgelenke oder Zahnfüllungen, die im Extremfall zu einer Totalabsorption der Röntgenstrahlung führen und die CT–Aufnahme unbrauchbar machen. [Buz04]

Neben Metallen können auch natürliche Bestandteile des menschlichen Körpers Störstrahlungen auf dem Detektor verursachen. Diese **Streustrahlungsartefakte** treten in stark absorbierenden Körperregionen auf, wie beispielsweise an Schultern und Becken. [Buz04]

Bei den vorgestellten Störungen handelt es sich lediglich um eine Auswahl in CT–Daten auftretender Bildfehler; zahlreiche weitere können die Bildqualität mindern. Da Artefakte in der Regel nicht vollständig oder nur mit sehr großem Aufwand und unter dem Risiko von Verfälschung aus dem CT–Datensatz herausgerechnet werden können, haben Sie folgerichtig direkten Einfluss auf die Güte einer angewandten Volumenvisualisierung [Buz04].

3 Visualisierung von Volumina

CT– und MRT–Datensätze bestehen aus einer Reihe individueller Bilder, die jeweils eine Schicht des gescannten Objekts darstellen. Die Bilder setzen sich zusammen aus Pixeln, die in x– und y–Richtung entlang eines 2D–Gitters angeordnet sind. In z–Richtung liegen die einzelnen Schichtbilder in unterschiedlichen Tiefen hintereinander. Werden alle Schichtbilder zu einem volumetrischen Datensatz zusammengenommen, entsteht ein 3D–Gitter aus Voxeln [PB07, EHK+06]. Mithilfe von Volumenvisualisierungen lassen sich ganze volumetrische Datensätze als dreidimensionale Repräsentationen darstellen. Dazu werden die einzelnen Voxel des volumetrischen Datensatzes in einer Pipeline ausgewählt, gewichtet, kombiniert und auf die Bildfläche projiziert [PB07].

Es existieren diverse Renderingverfahren zur Volumenvisualisierung. Als wichtigste Varianten im medizinischen Kontext werden direkte und indirekte Volumenvisualisierung voneinander unterschieden [PB07]. Grundsätzlich folgen beide der gleichen Visualisierungspipeline (s. Abbildung 8):

Ausgangspunkt ist ein volumetrischer Datensatz mit jeweils genau einem Datenwert V pro Voxel. Im Falle eines CT–Datensatzes repräsentiert V die Restintensität der von den Emittern ausgesendeten Röntgenstrahlung an den gegenüberliegenden Detektoren. Zur Verbesserung der Bildqualität, beispielsweise zur Beseitigung oder Abschwächung von Artefakten, wird gegebenenfalls eine Filterung angewandt. Danach können einzelne Objekte im Datensatz segmentiert werden. Dieser Schritt ist im Allgemeinen lediglich für indirekte Volumenvisualisierungen notwendig und für direkte Volumenvisualisierungen optional. Ist ein Clipping des Datensatzes gewünscht, so erfolgt

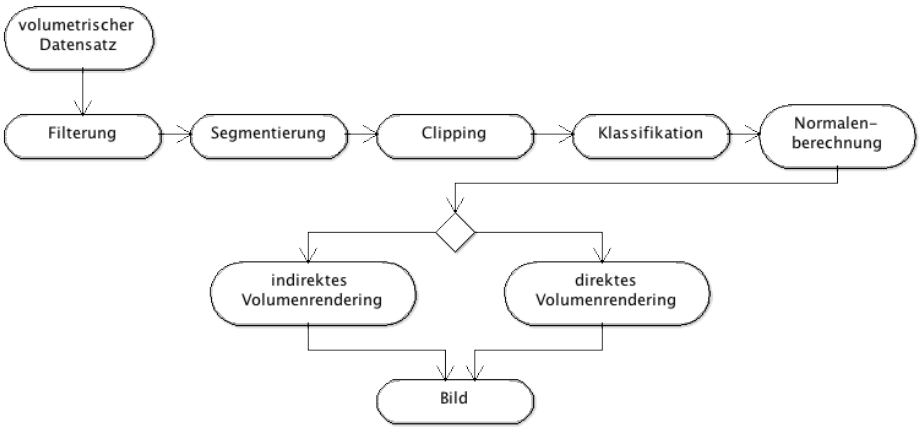


Abbildung 8: Allgemeine Volumenvisualisierungspipeline zur Umwandlung eines volumetrischen Datensatzes in eine dreidimensionale Darstellung (modifiziert nach Preim und Bartz [PB07]).

dies nach der Segmentierung. Sind alle Vorverarbeitungsschritte abgeschlossen, werden die darzustellenden Voxel mithilfe von Transferfunktionen klassifiziert. Eine Transferfunktion bestimmt anhand des Datenwerts die optischen Eigenschaften eines Voxels, das heißt Färbung und Transparenz, für die finale 3D–Darstellung. Sie muss anhand einer vorangegangenen Datenanalyse näher spezifiziert und an den aktuellen Datensatz angepasst werden (s. dazu Abschnitt 3.1.2). Danach werden Normalen für diejenigen Voxel, die in der finalen 3D–Darstellung sichtbar sind, berechnet. Eine gängige Methode dazu ist die Approximation anhand lokaler Gradienten im volumetrischen Datensatz. Abschließend werden die Voxel mittels direktem oder indirektem Volumenrendering auf die Bildfläche projiziert. [PB07]

3.1 Direkte Volumenvisualisierung

Direkte Volumenvisualisierungsverfahren (DVR–Verfahren) berechnen eine dreidimensionale Darstellung volumetrischer Datensätze ohne eine Oberflächenrepräsentation in Form eines Polygonnetzes (s. Abschnitt 3.2.1) generieren zu müssen [PB07, Köc05]. Sie eignen sich insbesondere für semi–transparente Darstellungen [Köc05].

DVR–Verfahren sind in der Lage hochgradig realistische Bilder mit wenigen Artefakten zu liefern – auf Kosten der Performanz. In jedem Rendervorgang muss der gesamte volumetrische Datensatz traversiert und die 3D–Visualisierung neu berechnet werden. [Köc05]

Das gebräuchlichste DVR–Verfahren ist *Volume Raycasting*. Daneben existieren weitere Methoden zur direkten Visualisierung von Volumina, beispielsweise *Shear–Warp Volume Rendering* oder *Splatting* [EHK+06]. Sie werden in diesem Kapitel nicht näher erläutert, da sich die vorliegende Arbeit gezielt mit der Erweiterung eines Raycasting–Systems befasst.

3.1.1 Datenstruktur von Volumina

In einem volumetrischen Datensatz werden alle Datenwerte der unterschiedlichen Schichtbilder einer CT–Aufnahme in einem 3D–Gitter angeordnet. Jeder Gitterpunkt repräsentiert ein Volumenelement, genannt **Voxel** [PB07, EHK+06]. Innerhalb des Gitters werden folgende Distanzen voneinander unterschieden: Als Schichtdistanz (*slice distance*) wird der Abstand zwischen aufeinanderfolgenden Schichtbildern bezeichnet; der Voxelabstand (*voxel spacing*) beschreibt die Distanz zwischen benachbarten Voxeln. Sind die Distanzen in allen drei Raumrichtungen identisch, heißt das 3D–Gitter isotrop. Variieren die Abstände je nach Raumrichtung, wird von einem anisotropen Gitter gesprochen. Im medizinischen Kontext sind anisotrope Datensätze mit einer zum Teil deutlich größeren Distanz in z–Richtung üblich. Anhand der Distanz–Werte eines volumetrischen Datensatzes sowie der Indizierung eines Voxels $V_{i,j,k}$ lässt sich die Position von $V_{i,j,k}$ im Raum berechnen. [PB07]

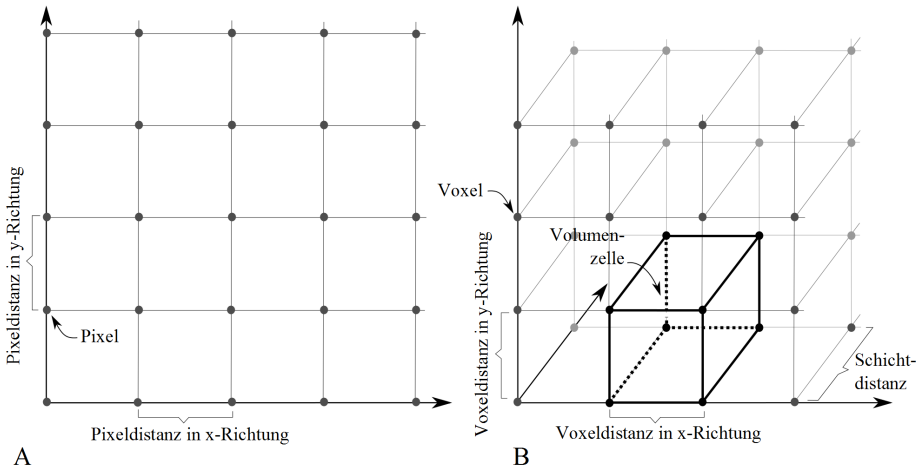


Abbildung 9: Aufbau und Struktur von Pixel– und Voxelgittern. **A:** Schematische Darstellung eines Pixelgitters. **B:** Schematische Darstellung eines Voxelgitters.

Üblicherweise wird in der medizinischen Visualisierung auf uniformen Gittern gearbeitet. Sie können als kartesische 3D–Koordinatensysteme betrachtet werden [PB07, EHK+06]. Uniforme Gitter zeichnen sich aus durch ihre klare Strukturierung. Als n–dimensionales Array lassen sie sich im Speicher trivial repräsentieren. Gleichzeitig wird ein schneller Datenzugriff sichergestellt. [EHK+06]

Abbildung 9 zeigt, wie einzelne Voxel zu **Volumenzellen** (*volume cells*) verbunden werden, um so die Interpretation und Darstellung des Gitters als ausgefülltes Volumen zu ermöglichen [EHK+06]. Acht benachbarte Voxel werden durch Kanten zu einem Quader verbunden [PB07]. Die Datenwerte, die an den Eckpunkten jeder Volumenzelle vorliegen, werden interpo-

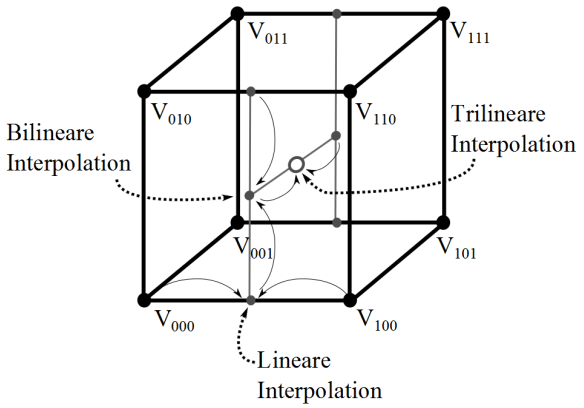


Abbildung 10: Trilineare Interpolation zwischen den Voxeln einer Volumenzelle.

liert, um somit auch das Innere der Volumenzelle mit Datenwerten zu füllen [PB07, EHK+06]. Eine einfache Methode, die gute Ergebnisse liefert, ist die trilineare Interpolation. Mithilfe der in Abbildung 10 dargestellten sieben Interpolationsschritte berechnet die trilineare Interpolation aus den acht Eckpunkten V_{000} bis V_{111} einer Volumenzelle alle innenliegenden Datenwerte. Mathematisch werden diese Interpolationsschritte wie folgt notiert [PB07]:

$$L(\alpha) = V_0 \cdot (1-\alpha) + V_1 \cdot \alpha \tag{2}$$

$$\begin{aligned} B(\alpha_1, \alpha_2) = &\quad L_0(\alpha_1) \cdot (1-\alpha_2) + L_1(\alpha_1) \cdot \alpha_2 \\ = &\quad (V_{00} \cdot (1-\alpha) + V_{10} \cdot \alpha) \cdot (1-\alpha_2) \\ &+ (V_{01} \cdot (1-\alpha) + V_{11} \cdot \alpha) \cdot \alpha_2 \end{aligned} \tag{3}$$

$$\begin{aligned} T(x,y,z) = &\quad B_0(x,y) \cdot (1-z) + B_1(x,y) \cdot z \\ = &\quad (L_0(x) \cdot (1-y) + L_1(x) \cdot y) \cdot (1-z) \\ &+ (L_2(x) \cdot (1-y) + L_3(x) \cdot y) \cdot z \\ = &\quad (V_{000} \cdot (1-x) + V_{100} \cdot x) \cdot (1-y) \\ &+ ((V_{010} \cdot (1-x) + V_{110} \cdot x) \cdot y) \cdot (1-z) \\ &+ (V_{001} \cdot (1-x) + V_{101} \cdot x) \cdot (1-y) \\ &+ ((V_{011} \cdot (1-x) + V_{111} \cdot x) \cdot y) \cdot z \end{aligned} \tag{4}$$

Die trilineare Interpolation $T(x,y,z)$ beruht auf einer Reihe von bilinearen und linearen Interpolationen $B(\alpha_1, \alpha_2)$ beziehungsweise $L(\alpha)$, wie sie in Formel 3 und 2 definiert sind. Die Variablen α, α_1, α_2, x, y und z liegen im Intervall [0,1] und bezeichnen die Werte, mit denen die Eckpunkte bei der Interpolation gewichtet werden. [PB07]

3.1.2 Transferfunktionen

Die Darstellung von Volumina mittels DVR–Verfahren basiert auf einer Klassifizierung K, die Intensitätswerte i aus dem volumetrischen Datensatz in Farb– und Opazitätswerte übersetzt [KGNP12, EHK+06]:

$$K(i) = (r, g, b, \alpha) \tag{5}$$

Zur Klassifizierung wird eine mathematische Funktion, die sogenannte Transferfunktion, auf die Intensitätswerte angewandt. Auf diese Weise lassen sich anatomische Strukturen in einem Datensatz unterscheiden und farblich codiert mit variabler Opazität darstellen. [KGNP12, EHK+06]

Um unnötige Wiederholungen immer gleicher Berechnungen zu vermeiden, werden Transferfunktionen üblicherweise in einem Vorverarbeitungsschritt auf den gesamten Wertebereich des volumetrischen Datensatzes angewandt. Die Ergebnisse werden in einer Tabelle in Form von RGBα–Vektoren

mit Werten im Intervall [0,1] gespeichert [KGNP12, EHK+06]. Für jede Position im Volumen lassen sich die vorberechneten Farb– und Opazitätswerte direkt interpolieren. Dies kann als Vorab–Klassifizierung (*pre–classification*) oder als nachträgliche Klassifizierung (*post–classification*) geschehen [PB07, KGNP12, EHK+06]. Abbildung 11 zeigt die Vorgehensweisen der beiden Klassifizierungsmethoden.

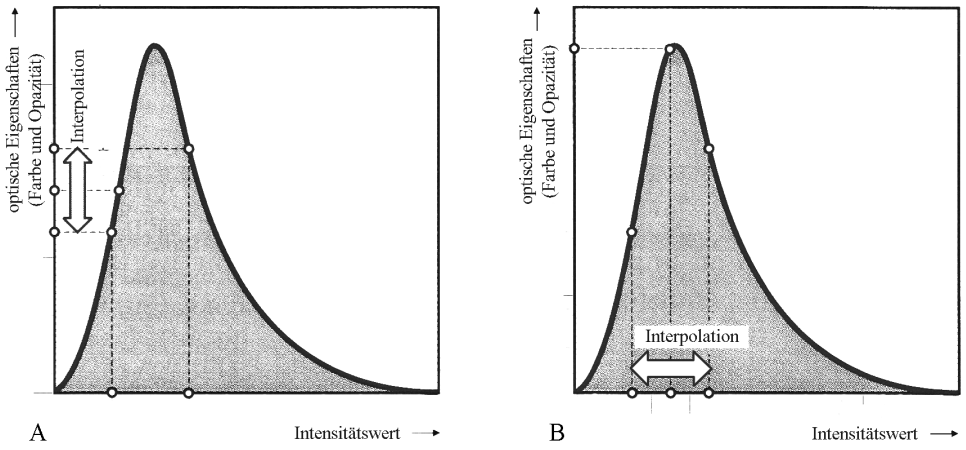


Abbildung 11: Interpolation zwischen Daten bei Vorab–Klassifizierung und nachträglicher Klassifizierung (modifiziert nach Engel et al. [EHK+06]). **A:** Bei der Vorab–Klassifizierung wird zunächst die Transferfunktion direkt auf die diskreten Abtastpunkte angewandt; anschließend werden die Ergebnisse interpoliert. **B:** Eine nachträgliche Klassifizierung interpoliert erst die Abtastposition und wendet die Transferfunktion auf das Ergebnis an.

Im Falle einer Vorab–Klassifizierung wird die Transferfunktion vor der Rekonstruktion direkt auf den diskreten Abtastpunkten angewandt [KGNP12, EHK+06]. Auf diese Art wird für Abtastpunkte innerhalb einer Volumenzelle zwischen den optischen Eigenschaften, Farbe und Opazität, interpoliert [EHK+06]. Dagegen arbeitet die nachträgliche Klassifizierung auf dem rekonstruierten Signal in Bildschirmauflösung und interpoliert zwischen den Intensitätswerten des volumetrischen Datensatzes. Die Ergebnisse weichen zwangsläufig voneinander ab (s. Abbildung 12). Eine merklich höhere Qualität wird mittels nachträglicher Klassifizierung erreicht [KGNP12, EHK+06]. Im Vergleich zur Vorab–Klassifizierung erzeugt sie ein glatteres Ergebnis ohne störende Artefakte. Dennoch kann auch eine Vorab–Klassifizierung sinnvoll sein, etwa bei der Verarbeitung von segmentierten Daten. Dabei soll explizit nicht zwischen Abtastpunkten interpoliert werden, um eine Verfälschung der im Datensatz segmentierten Kanten zu vermeiden [EHK+06].

Eine vergleichsweise einfache Variante zur Definition von Transferfunktionen ist die eindimensionale, stückweise lineare Funktion. Sie wird anhand

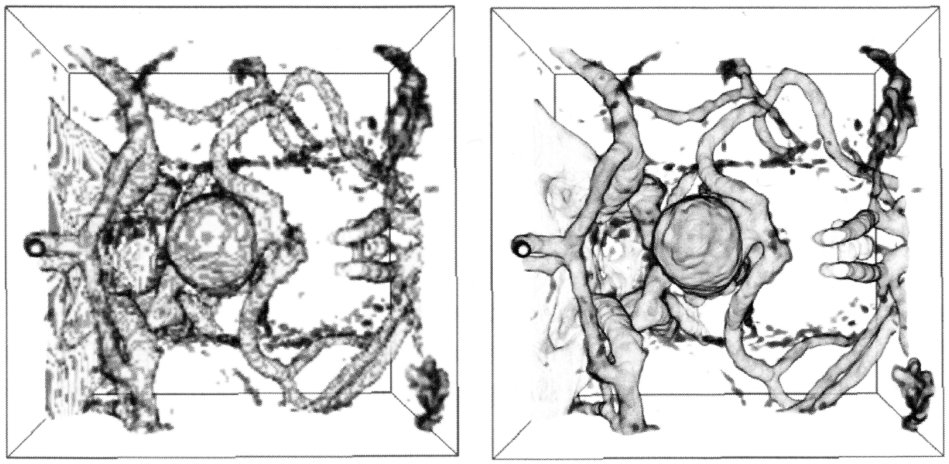


Abbildung 12: Vergleich der Ergebnisse, die mit Vorab–Klassifizierung (**A**) und nachträglicher Klassifizierung (**B**) erzielt werden (nach Engel et al. [EHK+06]). Beide Bilder wurden anhand der gleichen Transferfunktion auf Basis desselben Datensatzes generiert. Eine glatte Darstellung ohne auffällige Artefakte kann nur mit der nachträglichen Klassifizierung erreicht werden.

von Stützwerten definiert, die Farbhervorhebungen festlegen oder die Darstellung ausgewählter Intensitätswerte durch vollständige Transparenz unterdrücken [EHK+06]. Für die Visualisierung vaskulärer Strukturen sind eindimensionale Transferfunktionen jedoch ungeeignet. Gefäße lassen sich anhand der Intensitätswerte stellenweise nicht von Kontextobjekten unterscheiden. Insbesondere zu Knochen bestehen große Überschneidungen. Eine Lösungsmöglichkeit bieten multidimensionale Transferfunktionen (MDTFs). Sie definieren Farbe und Opazität nicht ausschließlich anhand der Intensitätswerte, sondern beziehen zusätzliche Parameter in die Berechnung ein. Unter anderem können Gradienten im Bild die Unterscheidung von Organen unterstützen [PO08, KGNP12]. Während kleine Gradienten homogene Regionen kennzeichnen, weisen größere Gradienten auf unterschiedliche, benachbarte Gewebe hin [KGNP12]. Nachteilig ist allerdings, dass MDTFs nicht trivial bestimmbar sind. Standardisierte Funktionen existieren nicht und auch automatisierte Definitionen erweisen sich als schwierig [PO08, KGNP12]. Semi–automatische Generierungen sind möglich, benötigen allerdings ausgefeilte Benutzerschnittstellen, um die Anwendbarkeit nicht auf ausgebildete Spezialisten zu beschränken [KGNP12].

Erste Ansätze automatisierter Methoden zur Generierung von ein– oder mehrdimensionalen Transferfunktionen nutzen bild– oder datengetriebene Mechanismen, wie die Histogrammanalyse [EHK+06]. Entsprechende Algorithmen wurden beispielsweise in [RSHSG00] und [VHST+04] vorgestellt. Im

Allgemeinen müssen Transferfunktionen jedoch weiterhin manuell und unter hohem Zeitaufwand bestimmt werden. Dabei ist detailliertes Wissen über die räumlichen Strukturen innerhalb des Datensatzes notwendig [EHK+06].

3.1.3 Volume Raycasting

Volume Raycasting ist ein klassisches Verfahren zur direkten Visualisierung von Volumina [PB07]. Es basiert auf der Verfolgung von Sehstrahlen, beginnend in der aktuellen Kameraposition, durch die Bildebene und den volumetrischen Datensatz. Im einfachsten Fall wird pro Pixel der Bildebene jeweils ein Strahl versendet. Der volumetrische Datensatz wird dann entlang des Strahls abgetastet [PB07, EHK+06, Köc05]. Abbildung 13 illustriert das Verfahren und hebt die einzelnen Abtastpunkte hervor.

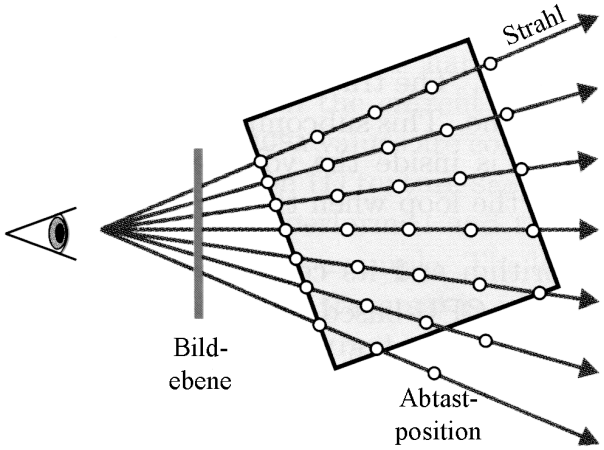


Abbildung 13: Schematische Darstellung des Raycasting–Verfahrens (modifiziert nach Engel et al. [EHK+06]). Pro Pixel der Bildebene wird ein Sehstrahl verfolgt und in gleichmäßigen Schritten abgetastet. Die Einzelergebnisse je Strahl werden zu einer Gesamtpixelfarbe und –opazität kombiniert.

An jedem Abtastpunkt wird der Intensitätswert des volumetrischen Datensatzes ausgelesen und mithilfe einer Transferfunktion in einen Farb– und Opazitätswert übersetzt [PB07, EHK+06, Köc05]. Der Opazitätswert bestimmt dabei, ob ein Abtastpunkt farblich zur 3D–Darstellung beiträgt (Opazität > 0) oder nicht (Opazität $= 0$) [PB07, EHK+06]. Die Ergebnisse an den einzelnen Abtastpunkten eines Strahls werden in einem Kompositionsschritt zusammengeführt und ergeben die Gesamtpixelfarbe an der Schnittstelle zwischen Strahl und Bildebene [Köc05].

Die Implementierung des Raycasting–Verfahrens iteriert für jeden ausgesendeten Strahl über folgende Funktionen:

- **Generierung des Strahls:** Anhand der Kameraparameter und der aktuellen Pixelposition in der Bildebene wird der Verlauf des Strahls festgelegt. Die Volumeneintrittsposition wird berechnet als Schnittpunkt zwischen dem Strahl und der Hüllgeometrie, die den gesamten volumetrischen Datensatz umschließt. [EHK+06]
- **Traversierung:** Beginnend am Volumeneintrittspunkt wird der Strahl in einer Schleife an diskreten Positionen abgetastet. Jede Iteration führt einen Datenzugriff und einen Kompositionsschritt aus, verschiebt die Abtastposition entlang des Strahls und überprüft die Bedingungen für eine Terminierung. [EHK+06]
  - **Datenzugriff:** Der Intensitätswert an der aktuellen Abtastposition wird aus dem volumetrischen Datensatz gelesen. Bei Positionen innerhalb von Voxelzellen muss der gesuchte Wert entsprechend Formel 4 interpoliert werden. Mithilfe einer vorab definierten Transferfunktion wird der Intensitätswert in Farbe und Opazität umgewandelt. [EHK+06]
  - **Komposition:** Die errechneten Werte werden auf die Summe aller bisher bestimmten Farb– und Opazitätsdaten des Strahls addiert. [EHK+06]
  - **Voranschreiten:** Die Abtastposition wird entlang des Strahls um eine definierte Schrittgröße verschoben [EHK+06]. Die Schrittgröße hat direkten Einfluss auf die Bildqualität und die Performanz des Algorithmus: Zu weite Schrittgrößen erzeugen Artefakte und können dazu führen, dass schmale Strukturen unbeabsichtigt übersprungen werden; schmale Schrittgrößen erhöhen dagegen die Berechnungszeit enorm. [KGNP12]
  - **Terminierung:** Wird die Hüllgeometrie verlassen, terminiert der Algorithmus und der nächste Strahl wird generiert [EHK+06].

Zur Darstellung von Gefäßen eignen sich insbesondere GPU–Implementierungen des Raycasting–Verfahrens, die die hohe Rechenleistung und Programmierbarkeit der Grafikhardware ausnutzen [KGNP12].

3.1.4 Optimierungsmethoden

Bis heute wurden diverse Strategien entwickelt, um Performanz und Bildqualität des Raycasting–Verfahrens zu steigern. Einige basieren auf einer Optimierung und Simplifizierung der Datenstruktur des zu traversierenden Volumens, andere nehmen algorithmische Verbesserungen vor.

Bereits 1990 wurde **Early Ray Termination** von Levoy [Lev90a] zur Steigerung der Performanz des Raycasting–Verfahrens vorgestellt. Die Methode beschleunigt die Strahlenverfolgung durch Erweiterung der Terminierungsbedingung. Der Strahl wird nicht mehr durch das gesamte Volumen

verfolgt. Stattdessen endet die Traversierung, sobald eine ausreichend große Opazität akkumuliert wurde. Dann leisten nachfolgende Abtastpunkte keinen Beitrag mehr zur Gesamtpixelfarbe und können als verdeckt angenommen werden. Der verwendete Schwellwert liegt in der Regel bei einer Opazität von 95%. [EHK+06, PB07]

Mit **Empty Space Skipping** stellten Levoy [Lev90a] sowie Levoy und Whitaker [LW90] einen weiteren Ansatz zur Reduktion der Menge aller Abtastpunkte vor. Die Idee ist es, die Hüllgeometrie, innerhalb derer der volumetrische Datensatz abgetastet wird, in einem Vorverarbeitungsschritt an das darzustellende Objekt anzupassen. Mithilfe der Transferfunktion werden diejenigen Bereiche des Volumens bestimmt, in denen Intensitätswerte in Alphawerte größer null übersetzt werden [EHK+06, Sch05]. Dazu wird das Volumen in gleich große Blöcke unterteilt und der minimale und maximale Intensitätswert pro Block gespeichert. Als Datenstruktur eignen sich Octrees oder 3D–Texturen [EHK+06]. Anhand der Transferfunktion und der Extremwerte kann bestimmt werden, ob ein Block in der finalen Darstellung sichtbar sein wird oder nicht. Die optimierte Hüllgeometrie umschließt dann nicht länger als Quader das gesamte Volumen, sondern wird ausschließlich aus den kleineren, sichtbaren Blöcken zusammengesetzt (s. Abbildung 14) [EHK+06, Sch05].

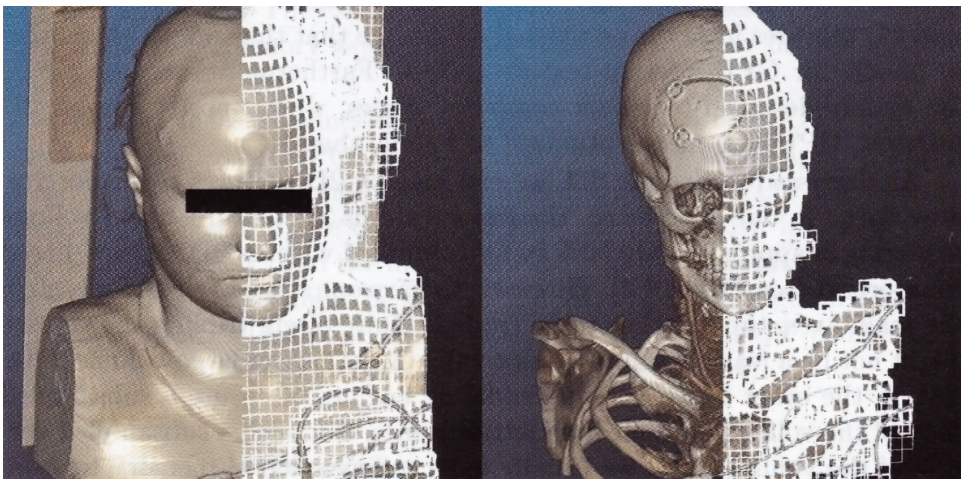


Abbildung 14: Darstellung optimierter Hüllgeometrien für Empty Space Skipping (nach Engel et al. [EHK+06]). Für die linke Transferfunktion benötigen etwa 40 % der Fragmente keine Verarbeitung durch Strahlenverfolgung. Im rechten Bild sind es sogar 80 %.

Obwohl die Hüllgeometrie auf diese Weise komplexer wird, ist dennoch eine signifikante Steigerung der Performanz messbar [Sch05]. Dies betrifft insbesondere die Visualisierung vaskulärer Strukturen, die je nach Datensatz und Transferfunktion nur 1 - 2% aller Voxel darstellen [PO08].

Zur Verbesserung der Bildqualität gilt es vor allem Artefakte, die durch

Abtastung entlang der Strahlen entstehen, zu entfernen. Bei größeren Abtastschritten erzeugt die plötzliche Veränderung des Tiefenwerts Holzmaserung–ähnliche Artefakte zwischen sichtbaren benachbarten Fragmenten, die zur gleichen Oberfläche gehören. Diese Artefakte verschwinden mit höheren Abtastraten auf Kosten der Performanz. Die Anwendung von **Stochastic Jittering** ermöglicht es, die Artefakte zu verstecken, ohne die Schrittweite minimieren zu müssen. Dazu wird an jeder Startposition eines neuen Strahls ein kleiner Offset entlang der Strahlrichtung aufaddiert. Die Offsets werden zufällig generiert und stellen damit sicher, dass alle Abtastpunkte in der Tiefe versetzt liegen. Auf diese Weise wird das Holzmaserung–ähnliche Artefakt durch Rauschen unterdrückt. Dies fällt in der 3D–Visualisierung allerdings deutlich weniger störend auf als ein gleichmäßiges Artefaktmuster. [EHK+06]

Interleaved Sampling bietet eine alternative Herangehensweise zur irregulären Abtastung von Strahlen. Während Stochastic Jittering für jeden Strahl eine eigens zufällig generierte Startposition bestimmt, arbeitet Interleaved Sampling mit einer beschränkten Auswahl an Offsets. Diese decken jeweils eine Gruppe von Strahlen ab und wiederholen sich regelmäßig. Auf diese Weise ist sichergestellt, dass benachbarte Strahlen immer in unterschiedlichen Tiefen beginnen. Die Bildqualität ist im Vergleich zu Stochastic Jittering gleichwertig. Eine Umsetzung von Interleaved Sampling ist beispielsweise mithilfe einer Modulo–Funktion, angewandt auf die aktuelle Bildschirmkoordinate, möglich.[Sch05]

Nähere Erläuterungen und Alternativen zu den oben genannten Verfahren sowie zahlreiche weitere Methoden zur Optimierung von Raycasting werden in [EHK+06] und [Sch05] beschrieben.

3.2 Indirekte Volumenvisualisierung

Im Gegensatz zu DVR–Verfahren erzeugen Indirekte Volumenvisualisierungsverfahren (IVR–Verfahren) keine unmittelbare Darstellung volumetrischer Datensätze, sondern überführen anatomische Strukturen zunächst in eine Zwischenrepräsentationen. Basierend auf Objektkonturen, die sich typischerweise aus Voxeln gleicher oder ähnlicher Intensitätswerte zusammensetzen, werden sogenannte Isoflächen (*isosurfaces*, s. dazu Abschnitt 3.2.1) generiert [PB07, Sch06a]. Ausgangspunkt gängiger Verfahren kann der unverarbeitete, volumetrische Datensatz sein; in der Regel wird allerdings auf Segmentierungsergebnissen gearbeitet [Sch06a].

Unter IVR–Verfahren wird zwischen **modellbasierten** und **modellfreien Ansätzen** differenziert. Erstere nutzen allgemeine Modellannahmen, um idealisierte Visualisierungen auf Kosten einer genauen Datenwiedergabe zu erstellen. Zuverlässigere Darstellungen erzeugen modellfreie Verfahren, die ohne Modellannahme auf den Segmentierungsdaten arbeiten [PO08]. Beispiele für modellfreie Algorithmen sind *Marching Cubes* [LC87], *Constrained Elastic Surface Nets* [Gib98] und *MPU Implicits* [OBA+05].

Die grundlegende, nicht immer korrekte Hypothese für modellbasierte Darstellungen vaskulärer Strukturen ist die Kreisförmigkeit der Gefäßquerschnitte [PO08, Sch06a]. Bekannte Verfahren sind die Visualisierung mit einfachen Primitiven, wie Kegelstümpfen, Zylindern und Halbkugeln [HPSP01, GKS+93], ebenso wie *Simplex Meshes* [BRB05], *Subdivision Surfaces* [FFKW02], *Convolution Surfaces* [BS91] und die Umsetzung mit Freiformflächen [EDKS94]. Einige dieser Methoden sind in [PB07] näher beschrieben.

Verglichen mit modellfreien Ansätzen benötigen modellbasierte Algorithmen eine aufwendigere Vorverarbeitung. Neben Rauschfilterung und Segmentierung ist zusätzlich eine Skelettierung notwendig. Diese kann sowohl die Performanz als auch die Genauigkeit der generierten Oberfläche negativ beeinflussen [PO08]. Gleichwohl bietet die modellbasierte Repräsentation analytische sowie explorative Möglichkeiten – beispielsweise ist die Messung von Pfadlängen und Verzweigungswinkeln ebenso wie die Färbung einzelner Teilstrukturen denkbar [Sch06a].

Neben ihrer Zugehörigkeit zur Gruppe der modellbasierten oder modellfreien Ansätze unterscheiden sich IVR–Verfahren in der Beschreibung der Oberflächenrepräsentation. Die häufigste Art der **expliziten Oberflächenbeschreibung** ist die Polygonmenge. Dreiecke, Vierecke oder auch komplexe n–Ecke setzten eine Oberfläche stückweise zusammen und können direkt als Grundlage für die Darstellung verwendet werden. Explizite Oberflächenrekonstruktionen werden beispielsweise mittels *Marching Cubes* und *Constrained Elastic Surface Nets* umgesetzt. Eine alternative, wenn auch weitaus weniger verbreitete Variante ist die erstmals von Blinn [Bli82] vorgestellte **implizite Oberflächenbeschreibung**. Dabei werden nicht, wie bei der expliziten Oberflächenbeschreibung, Punkte auf der Oberfläche definiert; stattdessen klassifiziert eine implizite Funktion gegebene Punkte danach, ob sie vor, auf oder hinter der Oberfläche liegen. Die Beschreibung ist somit indirekt. [Sch06a, Oel04]

Die Oberfläche einer im Ursprung gelegenen Kugel mit Radius $r = 1$ lässt sich implizit durch folgende Funktion beschreiben [Sch06a, Oel04, OP05]:

$$f(x, y, z) = x^2 + y^2 + z^2 - 1 \tag{6}$$

Für einen Punkt $P(x, y, z)$ lässt sich anhand der Funktion die Lage zur definierten Kugel bestimmen. Dabei kann P innerhalb der Kugel ($f(P) < 0$), außerhalb der Kugel ($f(P) > 0$) oder auf der Kugeloberfläche ($f(P) = 0$) liegen. [Sch06a, Oel04]

Verallgemeinert lautet die implizite Oberflächenbeschreibung [OM95]:

$$f(x, y, z) = F(x, y, z) - Iso = 0 \tag{7}$$

Die Skalarfeldfunktion $F(x, y, z)$ berechnet zu jedem Punkt $P(x, y, z)$ einen Skalarwert. Der Isowert Iso modifiziert die mithilfe von $F(x, y, z)$ be-

schriebene Oberfläche. Für die Kugel in obigem Beispiel ließe sich in Abhängigkeit vom Isowert der Radius variieren. [Sch06a, Oel04]

Implizite Oberflächen können sowohl um einzelne Punkte herum (z.B. *MPU Implicits*) als auch auf Basis eines Skeletts (z.B. *Convolution Surfaces*) generiert werden. Ist die Menge aller Nullstellen der impliziten Funktion gefunden, muss diese beispielsweise durch Polygonalisierung in eine darstellbare Repräsentation überführt werden. [Sch06a, Oel04, OBA+05]

Im Vergleich zu expliziten Oberflächen sind implizite Oberflächen kompakter [Oel04, OP05, PB07]. Sie sind in der Lage, das gesamte Objekt in einem Stück zu beschreiben und damit kontinuierliche Übergänge, auch an Verzweigungen, zu schaffen [OP05]. Damit eignen sich implizite Oberflächen speziell für die Modellierung glatter, deformierbarer Objekte [Oel04, OP05, Sch06a].

3.2.1 Datenstruktur von Oberflächen

Die Darstellung von Oberflächen durch IVR–Verfahren geschieht mithilfe von **Polygonnetzen** (*meshes*). Diese verbinden je zwei Vertices zu Kanten, wobei ein Vertex zu mehreren Kanten gehören kann. Eine Kante kann bis zu zwei Polygone begrenzen [FDFH97]. In der Computergrafik ist die Zusammensetzung von Polygonnetzen aus Dreiecken üblich [UH00].

Sowohl scharfkantige als auch abgerundete Oberflächen lassen sich durch Polygonnetze repräsentieren. Rundungen können dabei jedoch lediglich approximiert werden. Die Genauigkeit hängt von der Anzahl der verwendeten Polygone ab (s. Abbildung 15). [FDFH97]

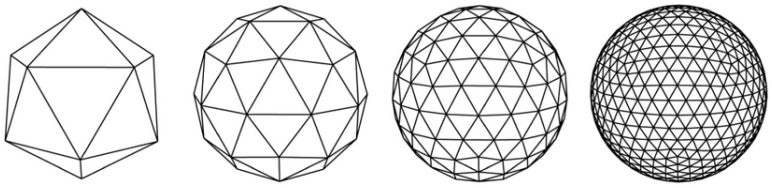


Abbildung 15: Approximation einer Kugel durch ein Polygonnetz mit wachsender Anzahl an Dreiecken (nach Fiedler [Fie]).

Eine besondere Form der Oberfläche ist die **Isofläche** (*isosurface*). Sie wird unter anderem in der medizinischen Visualisierung verwendet, um die Oberfläche ausgewählter Objekte eines volumetrischen Datensatzes zu repräsentieren [EHK+06]. Eine Isofläche $I(p)$ ist definiert als Menge aller Voxel p mit gleichem Intensitätswert (**Isowert**) $f(p)$ [EHK+06, UH00]:

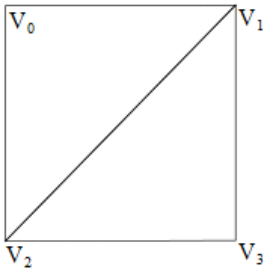
$$I(p) = \{x | f(x) = f(p)\} \tag{8}$$

Folglich bilden Isoflächen lediglich eine Teilmenge des ursprünglichen volumetrischen Datensatzes ab. Die Verbindung der zugehörigen Voxel zu einem Polygonnetz ergibt die Oberfläche des gesuchten Objekts. [EHK+06]

Zur Speicherung polygonaler Netze sind diverse Datenstrukturen denkbar. Eine triviale Beschreibung legt jedes n–eckige Polygon P als Liste seiner Vertexkoordinaten an [FDFH97]:

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)) \tag{9}$$

Innerhalb der Liste sind Vertices derart sortiert, dass Verbindungen zwischen aufeinanderfolgenden Vertices sowie zwischen dem letzten und ersten Vertex das Polygon aufspannen. Diese Form ist für die Darstellung eines einzelnen Polygons ausreichend effizient. Als Repräsentation für ein Polygonnetz eignet sie sich hingegen weniger, da dieselben Vertices mehrfach, und zwar für unterschiedliche, aneinandergrenzende Polygone, gespeichert werden. Dies lässt sich mithilfe einer indizierten Vertex–Liste vermeiden, in der alle im Polygonnetz vorkommenden Vertices genau einmal enthalten sind. Polygone werden dann nicht mehr direkt durch Vertices beschrieben, sondern indirekt anhand von Index–Listen; die Indizes zeigen jeweils auf einen Vertex der Vertex–Liste [FDFH97]. Das Prinzip ist in Abbildung 16 beispielhaft dargestellt.



Vertex Liste

	x	y	z
V_0	0.0	1.0	0.0
V_1	1.0	1.0	0.0
V_2	0.0	0.0	0.0
V_3	1.0	0.0	0.0

Index Liste

P_0	0	2	1
P_1	2	3	1

Abbildung 16: Beispiel einer indizierten Vertexliste zur Erzeugung zweier benachbarter Dreiecke.

Um die Verarbeitung von Oberflächen zu beschleunigen, ist es sinnvoll, ein Polygonnetz nicht in Form einer großen Liste, sondern hierarchisch als Baum anzulegen. Der Wurzelknoten beschreibt dabei das gesamte Volumen. Darunter werden zunehmend kleinere Strukturen gespeichert. **Octrees** halbieren Breite, Tiefe und Höhe volumetrischer Datensätze sukzessiv. Wie Abbildung 17 zeigt, entstehen auf diese Weise pro Knoten jeweils acht Kinder, die sogenannten **Octants**. Die Unterteilung wird solange durchgeführt, bis alle Octants genau eine Volumenzelle repräsentieren. Die Vorgehensweise setzt voraus, dass die Größe des Volumendatensatzes in allen drei Raumrichtungen einer Zweierpotenz entspricht. [PB07]

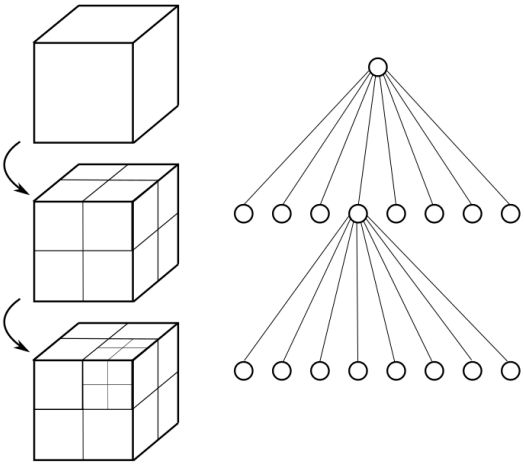


Abbildung 17: Darstellung der sukzessiven Teilung eines Würfels zur Speicherung in einer Octree–Struktur.

3.2.2 Explizite Oberflächenrekonstruktion

Explizite Oberflächenrekonstruktionen beschreiben Objekte als zusammenhängende Vertices in Form von Polygonnetzen [Sch06a, Oel04]. Das bekannteste Beispiel dieser Verfahrensklasse ist *Marching Cubes* [Sch06a, UH00]. Der Algorithmus wurde erstmals 1987 von Lorensen und Cline [LC87] vorgestellt. Eine Variante expliziter Oberflächen sind die *Constrained Elastic Surface Nets* von Gibson [Gib98] aus dem Jahr 1998. Beide Verfahren sollen im Folgenden konkretisiert werden.

Marching Cubes. Marching Cubes ist ein modellfreier Ansatz zur Erzeugung expliziter Oberflächenbeschreibungen aus volumetrischen Datensätzen [PO08]. Der Algorithmus untersucht nacheinander alle Volumenzellen auf einen Schnitt mit der gesuchten Isofläche. Dazu werden die Kanten der Volumenzellen betrachtet. Jede Kante ist zwischen zwei Voxeln aufgespannt, deren Intensitätswerte bekannt sind. Der Vergleich dieser Intensitätswerte mit dem Isowert der gesuchten Isofläche zeigt, ob eine Kante geschnitten wird oder nicht [Sch06a]. Dabei wird angenommen, dass pro Kante höchstens ein Schnitt existiert [PB07]. Wird ein Schnitt zwischen Isofläche und Kante detektiert, kann mittels linearer Interpolation der genaue Schnittpunkt festgestellt werden [PB07, Sch06a].

Marching Cubes unterscheidet 256 verschiedene Zustände für die Lage einer Teiloberfläche innerhalb einer Volumenzelle. Durch Rotation, Spiegelung oder Komplementbildung lassen sich einige dieser Konfigurationen aufeinander abbilden (s. Abbildung 18B). Dadurch ist eine Aufteilung der Konfigurationen auf die in Abbildung 18A dargestellten 15 Klassen möglich. Sie liegen zur Verarbeitung in einer Tabelle vor [PB07].

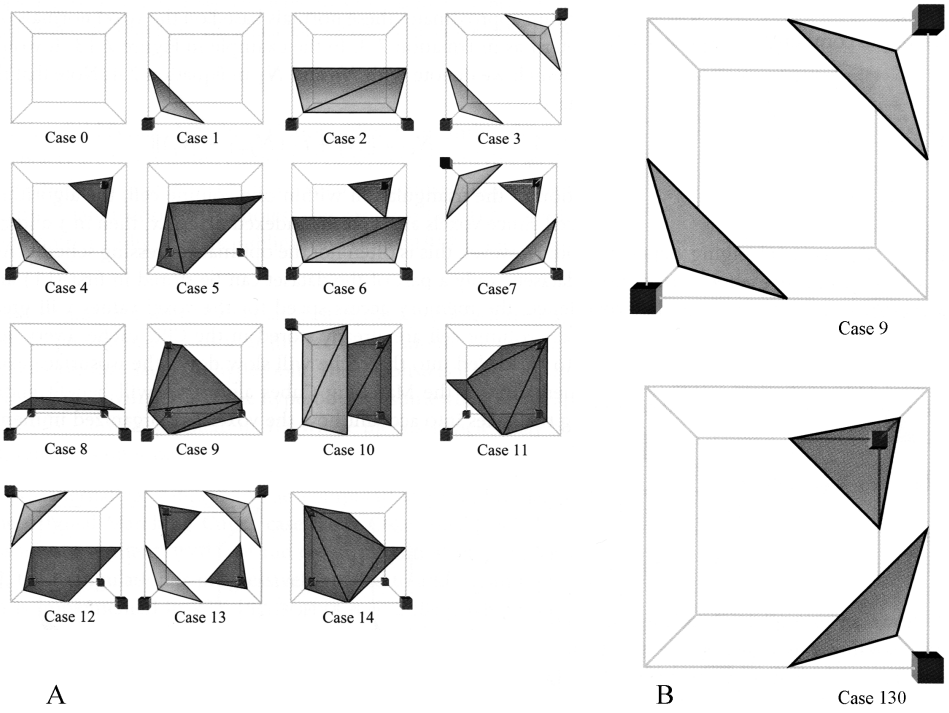


Abbildung 18: Die verschiedenen Marching Cubes Konfigurationen (modifiziert nach Preim und Bartz [PP07]). **A:** Darstellung der 15 Konfigurationsklassen, auf die sich alle möglichen 256 Marching Cubes Konfigurationen durch Rotation, Spiegelung oder Komplementbildung abbilden lassen. **B:** Durch Rotation können Case 9 und Case 130 aufeinander abgebildet werden.

Anhand der ermittelten Schnittpunkte und der Konfigurationstabelle werden die Vertices für die Triangulation der Isofläche berechnet [PB07]. Gleichzeitig können Oberflächennormalen durch lineare Interpolation der Gradienten an den Voxeln bestimmt werden [Sch06a]. Ist die Verarbeitung einer Volumenzelle abgeschlossen, wird die Triangulation für die nächste Volumenzelle durchgeführt [PB07].

Marching Cubes ist in der Lage die Isofläche äußerst genau zu selektieren [Sch06a]. Die Qualität der Visualisierung, insbesondere schmaler Strukturen, ist aufgrund eines ausgeprägten Treppenstufeneffekts jedoch gering [Sch06a, PO08]. Zusätzlich erzeugt der Algorithmus eine hohe Anzahl in etwa gleich großer Polygone, die nicht beeinflussbar und auch nicht von der lokalen Krümmung abhängig ist [Sch06a].

Es existieren diverse Möglichkeiten, den grundlegenden Marching Cubes Algorithmus zu optimieren, beispielsweise *Marching Tetrahedron* [ST90] oder *Dual Marching Cubes* [Nie04]. Auch eine Nachverarbeitung durch Glättung ist üblich, wenngleich dies einen Volumenverlust bewirkt [PB07, Sch06a].

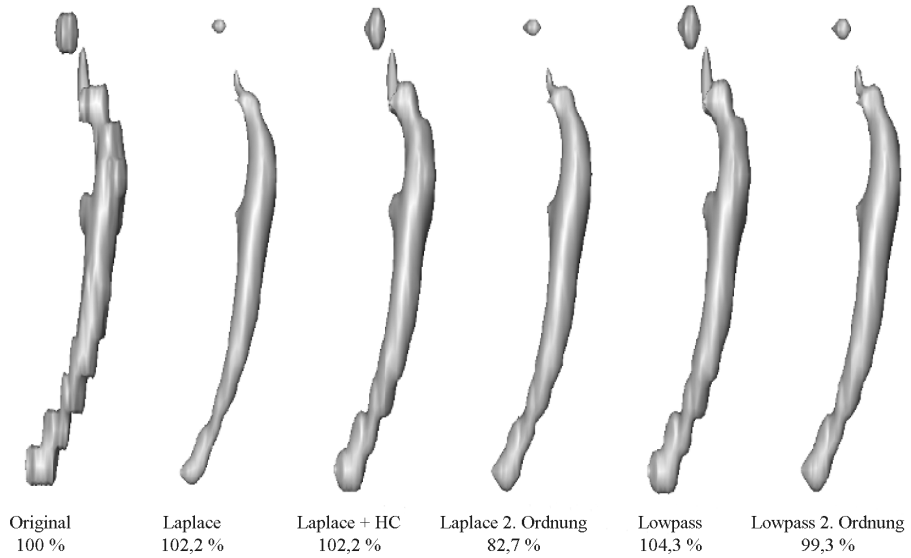


Abbildung 19: Glättung einer Oberflächendarstellung der Halsschlagader anhand unterschiedlicher Filtermethoden (nach Preim et al. [POT]). Die besten Ergebnisse können mit Laplace+HC sowie einem Low–Pass–Filter zweiter Ordnung erzielt werden.

Abbildung 19 stellt unterschiedliche Glättungsfilter einander gegenüber. Die besten Ergebnisse können mit Laplace+HC [VMM99] oder einem Low–Pass–Filter [Tau95] erzielt werden [PB07].

Zahlreiche weitere Methoden zur Verbesserung oder Erweiterung des Marching Cubes Algorithmus sind in [NY06] beschrieben.

Constraint Elastic Surface Nets. Eine Alternative zur modellfreien Generierung expliziter Oberflächen bieten Constrained Elastic Surface Nets (CESN). Das Verfahren erzeugt auf Basis segmentierter Volumendatensätze glatte Oberflächen. Dazu werden nacheinander alle Volumenzellen des segmentierten Datensatzes betrachtet. Besitzen alle acht Voxel einer Volumenzelle den gleichen Intensitätswert liegt die Zelle entweder komplett außerhalb oder komplett innerhalb des Objekts und wird nicht weiter verarbeitet. Zellen, die in ihren Eckpunkten unterschiedliche Intensitätswerte aufweisen, werden als *Surface Cubes* markiert. Im Zentrum jedes Surface Cubes wird ein Knoten platziert. Die Verbindung aller Knoten liefert das gesuchte *Surface Net* [Gib98, PB07, Sch06a].

Aufgrund der gitterartigen Knotenpositionierung weist das Surface Net starke Treppenstufeneffekte auf. Zur Verbesserung der Qualität wird die genaue Position der Knoten nachträglich angepasst. Dazu wird jeder Knoten in Richtung des Massezentrums seiner Nachbarn bewegt. Um Verzerrungen zu vermeiden, darf er seinen Surface Cube dabei nicht verlassen. In mehreren

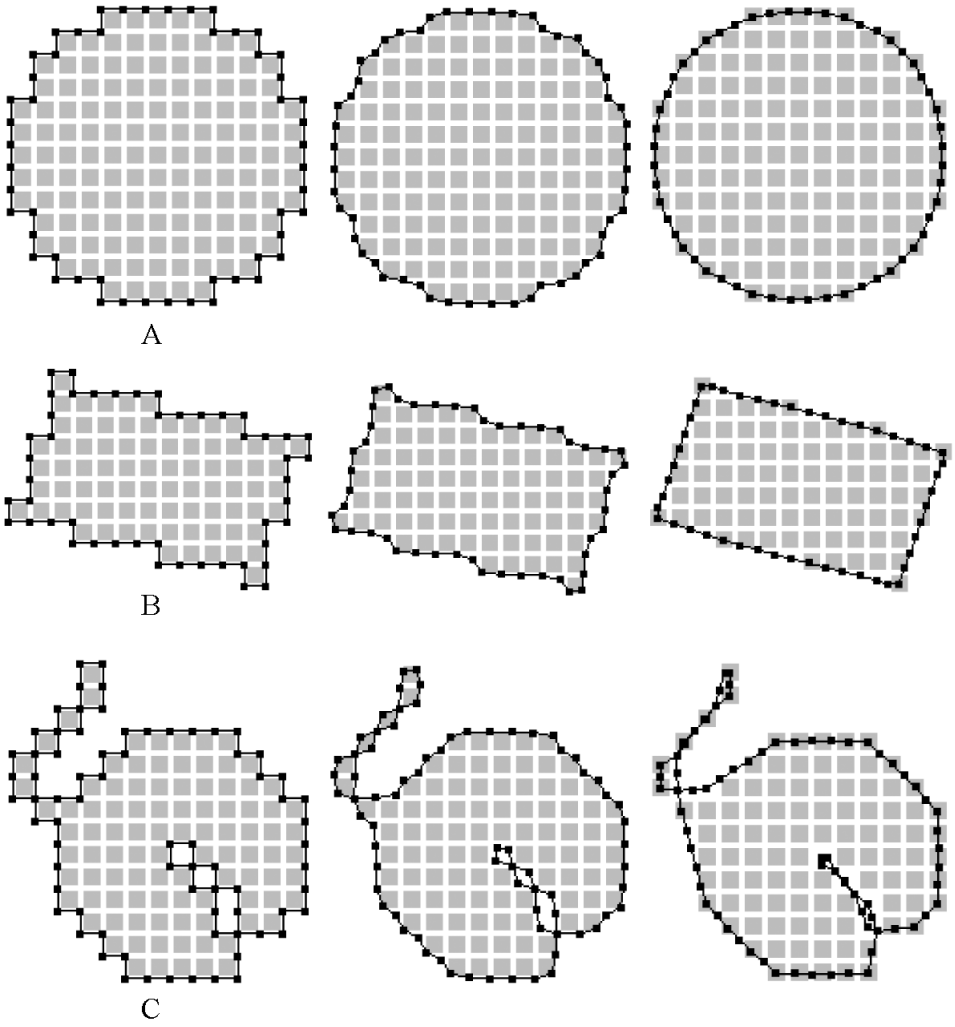


Abbildung 20: Beispiele für die Anwendung von Constrained Elastic Surface Nets auf zweidimensionale binäre Objekte (nach Gibson et al. [Gib98]). Pro Zeile wird jeweils eine Oberfläche in unterschiedlich vielen Iterationen geglättet. **A:** Annäherung eines Kreises mit 0, 1 und 10 Iterationen zur Glättung. **B:** Surface Net eines schiefen Rechtecks nach 0, 1 und 30 Iterationen. **C:** Darstellung eines rundlichen Objekts mit schmalen Strukturen nach 0, 1 und 20 Iterationen.

Iterationen wird die Oberfläche auf diese Weise geglättet (s. Abbildung 20). [Gib98, Sch06a]

Oberflächen, die mithilfe von Constrained Elastic Surface Nets modelliert werden, liegen sehr nahe am ursprünglichen Segmentierungsergebnis [Sch06a]. Trotz Glättung bleiben scharfe Ecken und Kanten in der Darstellung bestehen. Im Gegensatz zu klassisch gefilterten Polygonnetzen erhalten

Constrained Elastic Surface Nets auch schmale Strukturen [Gib98]; in der Regel verlieren sie allerdings deutlich an Volumen [PO08, Sch06a, SOBP07].

3.2.3 Implizite Oberflächenrekonstruktion

Implizite Oberflächenrekonstruktionen liefern indirekte Objektrepräsentationen in Form vom Funktionen [Sch06a, Oel04]. Eine skelettbasierte Variante sind die 1991 von Bloomenthal und Shoemake [BS91] vorgestellten *Convolution Surfaces*. Andere Verfahren, wie *Multi–Level Partition of Unity Implicits* (MPU Implicits) von Ohtake et al. [OBA+05] aus dem Jahr 2005, erzeugen die gesuchte Oberfläche anhand von Punkten. Nachfolgend werden beide Ansätze näher erläutert.

Convolution Surfaces. Convolution Surfaces ermöglichen die modellbasierte Beschreibung von Oberflächen [PO08]. Als Eingabe dient das Skelett[7] des darzustellenden Objekts. Es lässt sich vorab aus dem gegebenen volumetrischen Datensatz extrahieren und wird in Form eines Graphen repräsentiert. Jedes im Graphen abgelegte Segment wird um Informationen über den Abstand zwischen Skelett und Oberfläche ergänzt. Dabei wird angenommen, dass alle Querschnitte durch das Objekt kreisförmig sind und ihre Mittelpunkte auf dem Skelett liegen [OP05].

Als Skalarfeldfunktion nutzen Convolution Surfaces die Faltung des Skeletts mit einem Low–Pass–Filter, beispielsweise einem Gauß–Filter [Oel04, OP05, Sch06a]. Das resultierende Skalarfeld umspannt das Skelett. Auf diese Weise können auch komplexe Formen beschrieben werden (s. Abbildung 21) [Sch06a]. Eine anschließende Polygonalisierung wandelt das Skalarfeld in eine Oberfläche um [OP05].

Convolution Surfaces zeichnen sich durch ihr glattes, organisches Erscheinungsbild aus [PO08, OP05, Sch06a]. Die hohe visuelle Qualität zeigt sich vor

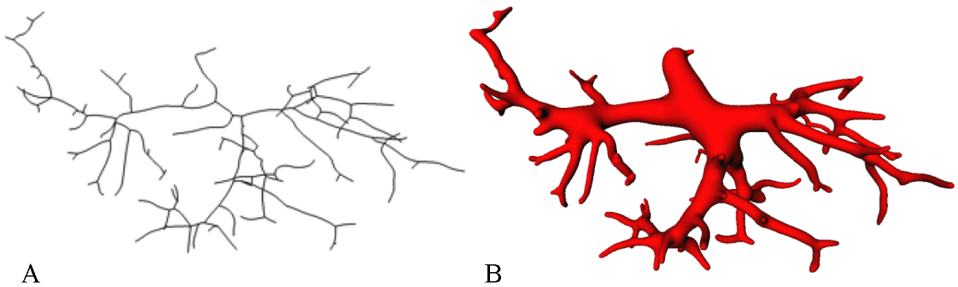


Abbildung 21: Darstellung einer Convolution Surface (**B**) mit zugehörigem Skelett (**A**) (nach Oeltze [OP05]).

[7] *Skelett*: Zusammensetzung aus Liniensegmenten mit einem Voxel Breite, die mittig durch das volumetrische Objekt verlaufen [PB07]

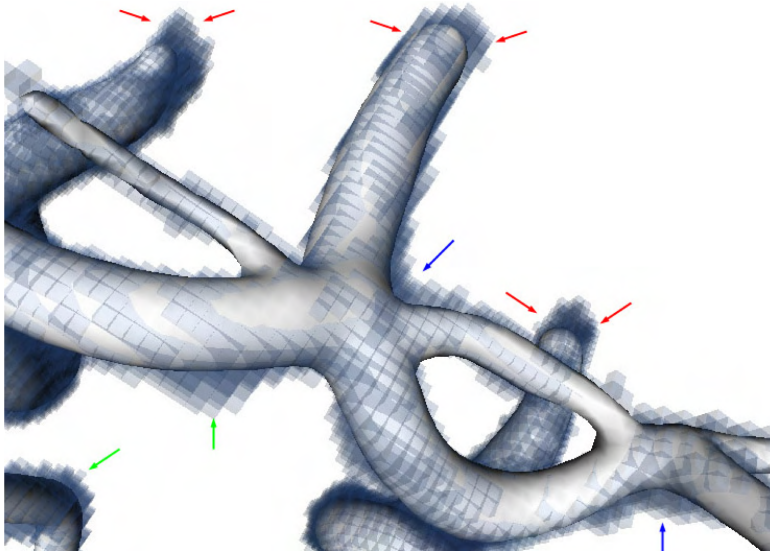


Abbildung 22: Überlagerung einer Convolution Surface mit dem zugrunde liegenden Segmentierungsergebnis (nach Schumann [Sch06a]). Abweichungen an Gefäßenden werden durch rote Pfeile, an Verzweigungen durch blaue Pfeile und entlang des Gefäßverlaufs durch grüne Pfeile hervorgehoben.

allem in der artefaktfreien Darstellung von Schatten und Verzweigungen. Objektenden werden automatisch mit Halbkugeln geschlossen [OP05, Sch06a]. Nachteilig wirkt sich dagegen die Annahme kreisrunder Querschnitte und die Glättung mittels Low–Pass–Filter aus. Beides erzeugt Ungenauigkeiten, wie Abbildung 22 durch die Überlagerung einer Convolution Surface mit dem ihr zugrunde liegenden Segmentierungengebnis zeigt [Sch06a]. Darüber hinaus erweisen sich die hohe Anzahl generierter Polygone, sowie die langen Berechnungszeiten, die für die Entwicklung des Skalarfeldes nötig sind, als ungünstig [PO08, OP05]. Eine beschleunigte Generierung von Convolution Surfaces wird durch die Verwendung moderner Grafikhardware ermöglicht. Eine GPU–basierte Implementierung wurde von Viola und Parulek [VP10] umgesetzt.

MPU Implicits. MPU Implicits sind eine modellfreie Variante zur Rekonstruktion impliziter Oberflächen [PO08]. Auf Basis des segmentierten Volumendatensatzes wird eine Punktewolke generiert, anhand derer die gesuchte Oberfläche beliebig genau approximiert werden kann [OBA+05, Sch06a]. Alle initialen Punkte liegen in den Volumenzellen am Rand des segmentierten Objekts (s. Abbildung 23) [PO08]. Mithilfe von *Least–Squares Fitting* wird auf allen Punkten ein Normalenvektor erzeugt. Der Algorithmus traversiert die gesamte Punktewolke und konstruiert für jeden Punkt diejenige Ebene, die am nächsten zu den übrigen Punkten in der Umgebung liegt. Die Senk-

rechte dieser Ebene wird anschließend als Normale des Punktes angenommen [Sch06a].

Zur besseren Approximation der Oberfläche an relevanten Stellen wird die Punktewolke als Octree repräsentiert. Anhand eines benutzerdefinierten Fehlermaßes wird festgestellt, welche Punkte das gesuchte Objekt noch nicht ausreichend annähern. Die Volumenzellen, in denen sich entsprechende Punkte befinden, werden in acht kleinere Zellen unterteilt. Für jede neue Randzelle wird ein zusätzlicher Punkt in der Punktewolke abgelegt (s. Abbildung 23). Der Octree wird so lange verfeinert, bis die Punktewolke die gesuchte Oberfläche hinreichend genau beschreibt. [OBA+05, Sch06a, SOBP07]

Unter Hinzunahme von Gewichtungsfunktionen lässt sich eine implizite Oberflächenbeschreibung aus dem Octree ableiten [Sch06a, SOBP07]. Eine Polygonalisierung des Ergebnisses ist zur abschließenden Darstellung notwendig [SOBP07].

Mit MPU Implicits ist es möglich auch zu topologisch komplexen Objekten hochqualitative und genaue Oberflächen mit glattem, organischen Erscheinungsbild zu berechnen [PO08, OBA+05, Sch06a, SOBP07]. Das Verfahren liefert kontinuierliche Schattierungen sowie scharfe Kanten und Ecken. Es ist zudem in der Lage Rauschen im Segmentierungsergebnis zu unterdrücken [Sch06a]. Dabei erfolgt die Berechnung der Oberfläche jedoch deut-

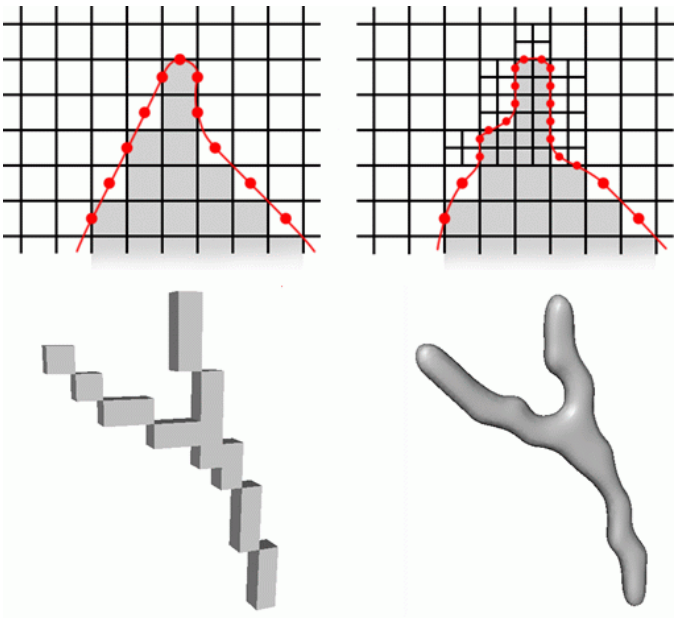


Abbildung 23: Vorgehensweise und Ergebnis der impliziten Oberflächenrekonstruktion mit MPU Implicits (nach Schumann [Sch06a]). Zunächst wird die Oberfläche mithilfe einer Punktewolke approximiert (links); das Ergebnis wird anschließend anhand eines benutzerdefinierten Fehlermaßes verfeinert (rechts).

lich langsamer, als bei anderen Rekonstruktionsverfahren [PO08]. Auch die Parametrisierung erweist sich in der Regel als aufwändig [SOBP07]. Insgesamt korreliert die Komplexität des Algorithmus positiv mit der Größe und Komplexität des Segmentierungsergebnisses [PO08, Sch06a].

3.3 Hybride Visualisierungsverfahren

Direkte und Indirekte Volumenvisualisierungsverfahren variieren nicht nur in der Verarbeitung des Volumendatensatzes; auch hinsichtlich Bildqualität, Darstellungsoptionen und allgemeiner Anwendbarkeit lassen sich Unterschiede erkennen.

DVR–Verfahren eignen sich insbesondere für Kontextdarstellungen, die einen schnellen Überblick über große Teile des Volumendatensatzes liefern [TIP05]. Abbildung 24 zeigt ein hochqualitatives Beispiel aus der medizinischen Visualisierung. Problematisch ist die genaue Differenzierung einzelner Objekte voneinander. Dies betrifft sowohl das optische Erscheinungsbild (Überlagerungen in der 3D–Visualisierung), als auch die vorzunehmenden Berechnungen (Bestimmung der optimalen Transferfunktion) [TIP05]. Aufwendig parametrisierte Transferfunktionen sind vor allem zur Darstellung scharfer Kanten notwendig [EHK+06]. Eine (photo–)realistische Beleuchtung kann den Effekt verstärken und verbessert gleichzeitig den andernfalls vergleichsweise mäßigen 3D–Eindruck [PO08, HMIBG01].

Insgesamt sind DVR–Verfahren in der Lage hochgradig realistische Darstellungen zu generieren, die neben der Objektform zusätzlich auch die Beschaffenheit der Oberfläche visualisieren können [HMIBG01, KGNP12]. Die

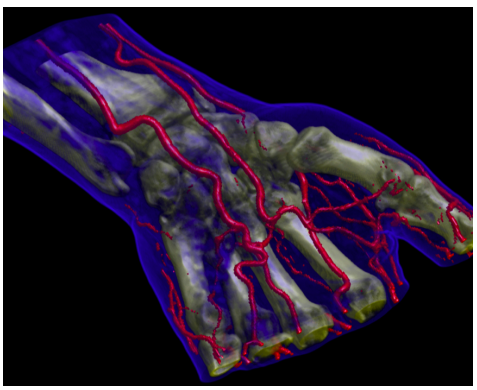


Abbildung 24: Medizinische Visualisierung der Knochen, Gefäßstrukturen und der Haut einer Hand mit DVR–Verfahren (nach Hadwiger et al. [HBH03]).

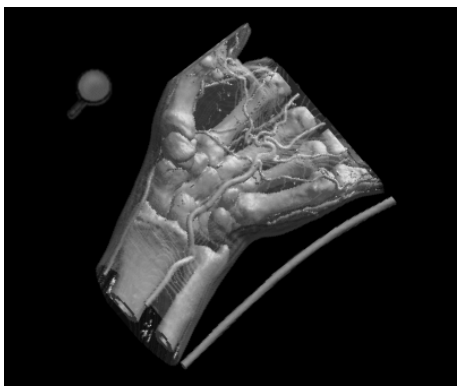


Abbildung 25: Medizinische Visualisierung der Knochen, Gefäßstrukturen und der Haut einer Hand mit IVR–Verfahren (nach Mroz et al. [MWG00]).

	Bildqualität	Transparenz	Selektierbarkeit	Segmentierung	Komplexität
DVR	T, G, (K), R	Ja	Nein	Nein	abh. von Datensatz und Bildauflösung
IVR	T, A, K, (R)	Ja	Ja	Ja	abh. von der Anzahl generierter Polygone

Tabelle 3: Vergleich von DVR– und IVR–Verfahren hinsichtlich Bildqualität, Darstellungsoptionen und Datenverarbeitung. Die Beurteilung der Bildqualität berücksichtigt einen guten Tiefeneindruck (T), die Möglichkeit zur Darstellung in unbegrenzter Auflösung (A), mit scharfen Kanten (K) und weichen Gewebeübergängen (G) sowie ein realistisches, genaues Gesamtergebnis (R). Angaben in Klammern zeigen die Abhängigkeit von konkreten Verfahren an.

Zahl der vorkommenden Artefakte ist gering. Sie treten in erster Linie aufgrund der begrenzten Auflösung des volumetrischen Datensatzes als Aliasing–Effekte an schmalen Strukturen auf [PO08, Sch06a]. Die Komplexität der Algorithmen hängt von der Anzahl der Voxel im volumetrischen Datensatz und von der Auflösung des gerenderten Bildes ab [PB07].

Im Gegensatz dazu werden IVR–Verfahren lediglich mit der Anzahl generierter Polygone, nicht aber mit steigender Bildauflösung komplexer [PB07]. Auf Basis segmentierter volumetrischer Datensätze generieren sie eine Zwischenrepräsentation [PB07, Sch06a]. Diese Form der Darstellung erlaubt Interaktionen mit der 3D–Visualisierung, die über simple Navigation hinausgehen und mit DVR–Verfahren nicht oder nur unter Aufwand möglich sind. Ein Beispiel ist die Selektion eines Objekts oder die Markierung von Positionen auf dem Objekt [Sch06a].

IVR–Verfahren sind insbesondere dazu geeignet die Grenzen von Objekten anzuzeigen. Sie erzeugen scharfe Objektkonturen und sogar mit simplen Beleuchtungsmethoden einen guten 3D–Eindruck [HMIBG01]. Gewebeübergänge gehen bei indirekter Visualisierung jedoch verloren. Genauigkeit und Realismus der Darstellung können abhängig vom konkreten Verfahren sehr gut (z.B. MPU Implicits), aber auch unzulänglich (z.B. Convolution Surfaces) sein. Eine hochqualitative Oberflächenvisualisierung ist in Abbildung 25 dargestellt.

Die Eigenschaften beider Volumenrendering–Klassen sind in Tabelle 3 zusammengefasst und einander gegenübergestellt. Daraus ersichtlich werden die gegensätzlichen Stärken in den Bereichen Bildqualität, Darstellungsoptionen und Datenverarbeitung. **Hybride Ansätze**, die indirekte und direkte Volumenvisualisierungsmethoden zu einem System kombinieren, versuchen die jeweiligen Vorzüge der beiden Verfahrensklassen zu nutzen und damit die Gesamtqualität der 3D–Visualisierung zu verbessern – optimalerweise einhergehend mit reduziertem Parametrisierungsaufwand. Die Schwierigkeit

dabei besteht in einer geeigneten Kombination der unterschiedlichen Datenstrukturen ohne optisch sichtbare Brüche und Übergänge. Frühe Ansätze lösen das Problem durch Konvertierung einer Datenrepräsentation in die jeweils andere: Einerseits lässt sich oberflächenapproximierende Geometrie aus Volumendaten extrahieren und anschließend mit vorab gegebenen Polygonnetzen rendern; andererseits ermöglicht es die Scankonvertierung [KS87] ein Polygonnetz in eine Voxelrepräsentation umzuwandeln und es gemeinsam mit einem gegebenen Volumen direkt zu visualisieren. Beide Varianten haben den Nachteil, dass sowohl Bildinformation als auch Qualität mit der Konvertierung verloren gehen. Weiterführende Ansätze erarbeiten deshalb Möglichkeiten Oberflächen– und Volumenrepräsentationen ohne vorangehende Konvertierung gemeinsam darzustellen [Köc05]. Levoy et al. veröffentlichten zu diesem Thema bereits 1989 ihren *Hybrid Raytracer* [Lev90b]. Weitere Verfahren sind das *Two–level Volume Rendering* von Hauser et al. [HMIBG01], *Focus–plus–Context* Visualisierungen nach Tietjen et al. [TIP05] sowie ein Ansatz zur *virtuellen Endoskopie* von Scharsach et al. [SHN+06]. Sie werden in den nachfolgenden Kapiteln näher erläutert.

3.3.1 Hybrid Raytracer nach Levoy et al.

Der Hybrid Raytracer[8] von Levoy et al. [Lev90b] ermöglicht die simultane Darstellung polygonaler und volumetrischer Datenstrukturen anhand eines Raycasting–Systems. Das Verfahren geht aus von gegebenen Volumen– und Polygondaten, die in einem gemeinsamen Speicher vorliegen. Ein Raycaster verarbeitet beide Datensätze getrennt voneinander. Das Verfahren ist in Abbildung 26 schematisch dargestellt.

Zunächst wird das Polygonnetz mit jeweils einem Strahl pro Pixel der Bildfläche abgetastet. Die Strahlen verlaufen dabei parallel und gehen vom Betrachter aus. Im Gegensatz zum herkömmlichen Raycasting, werden die Strahlen nicht schrittweise abgetastet. Stattdessen werden Schnittpunkte mit dem Polygonnetz bestimmt. Die Farbe und Opazität an den Schnittpunkten liefert das Phong'sche Beleuchtungsmodell. Die entlang eines Strahls ermittelten Werte werden der Tiefe nach sortiert und in eine geeignete Datenstruktur geschrieben.

Die Verarbeitung der Volumendaten erfolgt mittels klassischem Raycasting, das heißt durch schrittweise Abtastung der Strahlen. Die Beleuchtung wird anders als bei der Traversierung des Polygonnetzes nicht unmittelbar berechnet, sondern für jedes Voxel vorab in einer Wertetabelle gespeichert. An den Abtastpositionen müssen dann lediglich Farb– und Opazitätswerte aus der Tabelle ausgelesen werden. Liegt der Treffer mitten in einer Volumenzelle, wird die Farbinformation zwischen ihren acht Voxeln trilinear in-

[8] Genau genommen handelt es sich bei dem Verfahren von Levoy et al. entgegen des Namens nicht um einen Raytracer, sondern um einen Raycaster, da ausgesandte Strahlen nicht reflektiert oder transmittiert werden.

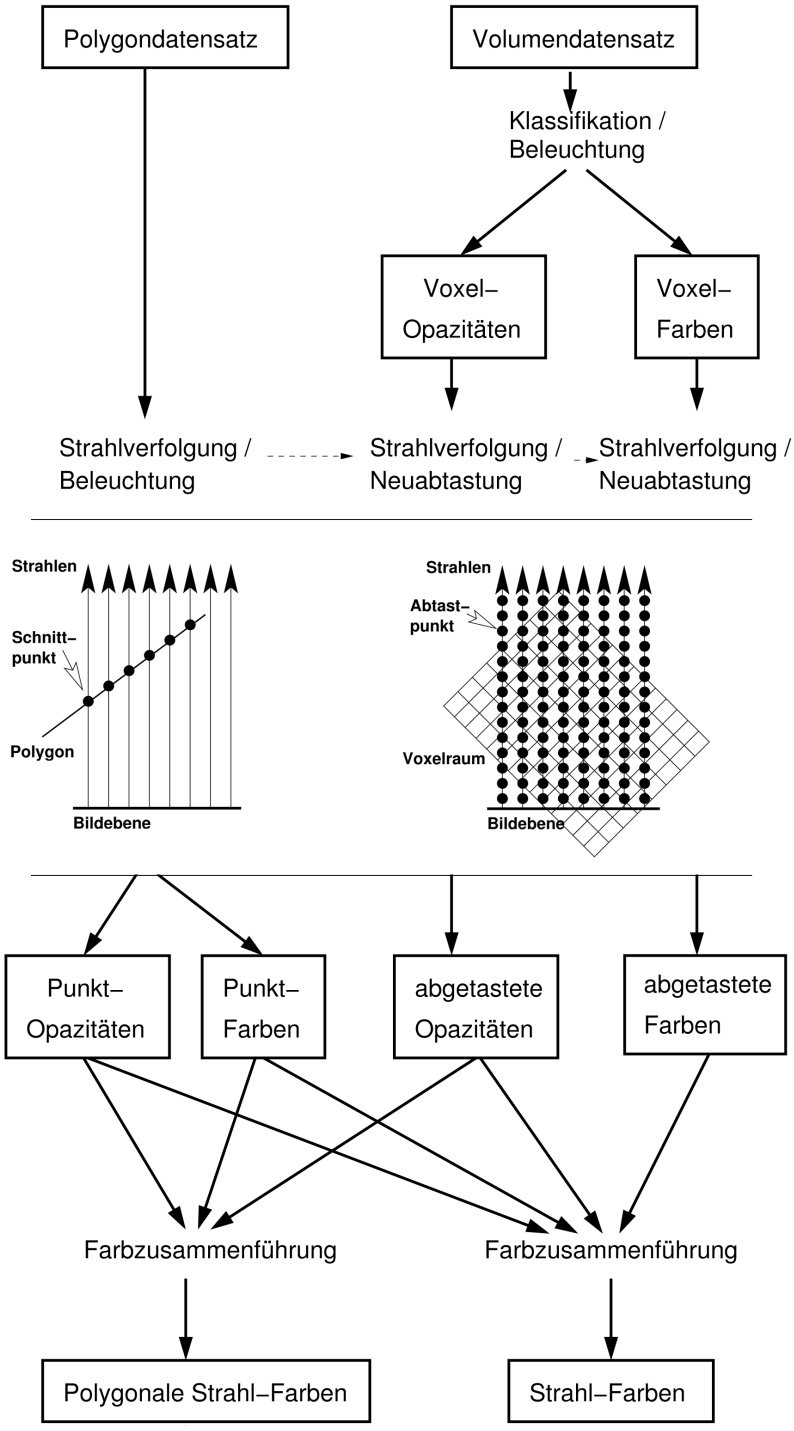


Abbildung 26: Ablauf des Renderings innerhalb des Hybriden Raytracers (modifiziert nach Köchy [Köc05] und Levoy [Lev90b]).

terpoliert. Die Ergebnisse pro Strahl werden ebenfalls der Tiefe nach sortiert gespeichert.

Die derart gesammelten Informationen werden in einem abschließenden Kompositionsschritt anhand einer Transparenzformel zusammengeführt:

$$\begin{aligned} C(n) &= C(n) + C_{next}(n) \cdot (1 - \alpha(n)) \\ \alpha(n) &= \alpha(n) + \alpha_{next}(n) \cdot (1 - \alpha(n)) \end{aligned} \tag{10}$$

Für jedes Pixel der Bildfläche werden alle hinter ihm liegenden Farb– und Opazitätswerte, die durch Polygon– und Volumen–Raycasting ermittelt wurden, betrachtet. Sortiert nach ihrer Tiefe werden die Werte aus beiden Raycasting–Schritten zusammengeführt und anschließend von vorne nach hinten gewichtet akkumuliert. $C(n)$ und $\alpha(n)$ bezeichnen dabei die bis zum n–tiefsten Abtastpunkt errechnete Gesamtfarbe beziehungsweise –opazität. $C_{next}(n)$ und $\alpha_{next}(n)$ sind die hinzuzurechnenden Farb– und Opazitätsinformationen an der nächsttiefer liegenden Abtastposition.

Der Hybrid Raycaster liefert gute Bildqualität [Köc05]. Die Darstellung ist frei von Artefakten, die bei Konvertierung der unterschiedlichen Datenstrukturen in ein gemeinsames Format aufgetreten wären. Darüberhinaus kann das Verständnis der dreidimensionalen Visualisierung durch die Kombination von Oberfläche und Volumendaten verbessert werden [Lev90b]. Aliasing–Artefakte, die durch das Raycasting produziert werden, lassen sich allerdings nicht vermeiden [Köc05]. Zudem sind für das zweifache Raycasting und die Speicherung aller Einzelergebnisse hohe Rechenkraft und Speicherkapazitäten erforderlich [PB07, Köc05].

Diverse Methoden zur Optimierung des Basiskonzepts sind in [Köc05] zusammengefasst.

3.3.2 Two–level Volume Rendering nach Hauser et al.

Two–level Volume Rendering von Hauser et al. [HMIBG01] ermöglicht die optische Trennung von Fokus–[9] und Kontextobjekten[10] durch die Verwendung verschiedener Rendertechniken. Dazu werden vorab mittels binärer Segmentierung alle darzustellenden Objekte aus dem volumetrischen Datensatz extrahiert. Jedem Einzelobjekt wird das gewünschte Visualisierungsverfahren zugeordnet. Ein Algorithmus zur Strahlenverfolgung verarbeitet die volumetrischen und segmentierten Daten. Abbildung 27 veranschaulicht das Verfahren im zweidimensionalen Raum.

Für jedes Pixel der Bildfläche wird ein Strahl durch den volumetrischen Datensatz gesendet. Anhand der Segmentierungsergebnisse kann jede beliebige Abtastposition entlang des Strahls genau einem Objekt zugeordnet

[9] *Fokusobjekt*: Objekte von Interesse, die in der Visualisierung besonders hervorgehoben werden sollen [TIP05]

[10] *Kontextobjekt*: irrelevante Objekte, die lediglich zur räumlichen Orientierung visualisiert werden [TIP05]

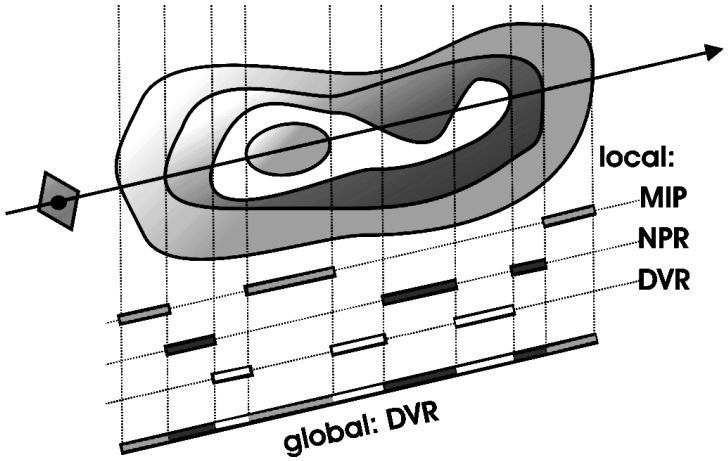


Abbildung 27: Vorgehensweise bei Two–level Volume Rendering exemplarisch gezeigt an einem Strahl im zweidimensionalen Raum (nach Hadwiger et al. [HBH03]). Auf lokaler Ebene wird für jedes (im Bild in Graustufen codierte) Objekt mit einem beliebigen Verfahren gerendert. Global kombiniert ein DVR–Verfahren die Einzelergebnisse.

werden. Damit lässt sich der Strahl in einzelne Segmente unterteilen, die jeweils ein Objekt im volumetrischen Datensatz traversieren.

Während der Abtastung eines gesamten Strahls (globales Rendering) werden die einzelnen Segmente zunächst separat verarbeitet (lokales Rendering). Dabei wird für jedes Segment das Visualisierungsverfahren verwendet, das dem jeweiligen Objekt vorab zugeordnet wurde. Die Traversierung des Strahlensegments liefert einen Farb– und einen Opazitätswert. Auf diese Weise werden Werte für alle Objekte entlang eines Strahls berechnet. Das globale Rendering kombiniert die Einzelergebnisse zu einem Gesamtpixelwert.

Der Ansatz bietet die Möglichkeit für unterschiedliche Objekte und Anwendungen das jeweils beste Visualisierungsverfahren zu nutzen. Hauser et al. kombinieren in ihrer Evaluation opake Oberflächen und DVR–Techniken (für fokussierte Objekte) sowie die Maximumintensitätsprojektion[11], transparente Oberflächen und Non–Photorealistic–Rendering–Methoden[12] (für den Kontext). Auf diese Art lassen sich die individuellen Objekte in der Gesamtdarstellung leicht voneinander unterscheiden. Das Ergebnis kann interaktiv gestaltet werden, beispielsweise durch die Veränderung einzelner Parametri-

[11] *Maximumintensitätsprojektion (MIP)*: Verfahren zur Visualisierung volumetrischer Datensätze basierend auf Strahlenverfolgung; bestimmt den maximalen Intensitätswert entlang eines Strahls und verwendet ihn als Pixelwert in der Gesamtdarstellung [PO08]

[12] *Non–Photorealistic–Rendering (NPR)*: Überbegriff für Techniken des illustrativen Renderings, wie beispielsweise Tone Shading, Hatching (Schraffierung) oder die Berechnung und Darstellung von Silhouetten [LM02]

sierungen oder den Tausch ganzer Visualisierungsverfahren auf lokaler Ebene. Überdies ermöglicht die Verwendung der Segmentierungen das stufenlose Ein– und Ausblenden einzelner Objekte.

Aufgrund der Vielzahl unterschiedlicher Berechnungen ist das Verfahren auf leistungsstarke Hardware angewiesen. Zur Optimierung der Performanz und der Bildqualität nutzen Hadwiger et al. [HBH03] deshalb die Rechenkraft der Grafikkarte.

3.3.3 Focus–plus–Context Visualisierung nach Tietjen et al.

Um Fokus– und Kontextobjekte optisch voneinander zu trennen, nutzen Tietjen et al. eine Szenegraph–basierte Variante für hybrides Rendering. Das Verfahren segmentiert unterschiedliche Objekte im volumetrischen Datensatz. Via Benutzereingabe wird jedem Objekt eine Wichtigkeit zugeordnet. Anhand dieser Wichtigkeit wird für die Visualisierung jedes einzelnen Objekts das passende Verfahren ausgewählt. Tietjen et al. unterscheiden dabei zwischen dem Rendering von Silhouetten, Oberflächen und direkten Volumendarstellungen. Zur Kombination der Einzelergebnisse wird ein DVR–Verfahren eingesetzt.

Für jedes aus dem volumetrischen Datensatz extrahierte Objekt wird ein Polygonnetz generiert, in eine Winged–Edge–Datenstruktur[13] umgewandelt und auf unterster Ebene im Szenegraphen eingefügt. Eine Ebene darüber wird das anzuwendende Visualisierungsverfahren abgelegt. Jeder einzelne Verfahrensschritt wird dabei als separater Knoten gespeichert. Eine Reihe von Knoten wird als Stilisierungspipeline bezeichnet.

Abbildung 28 zeigt den Szenegraphen für das kombinierte Rendering von Silhouetten, Oberflächen und Volumina. Zu Beginn wird der Tiefenpuffer

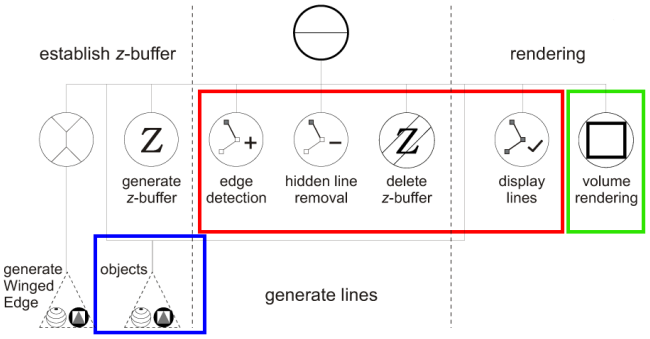


Abbildung 28: Szenegraph für das kombinierte Rendering von Silhouetten (rot), Oberflächen (blau) und direkter Volumenvisualisierung (grün) (modifiziert nach Tietjen et al. [TIP05]).

[13] *Winged–Edge–Datenstruktur*: effiziente Datenstruktur nach Baumgart [Bau72], die Polygonnetze als Liste ihrer Kanten repräsentiert

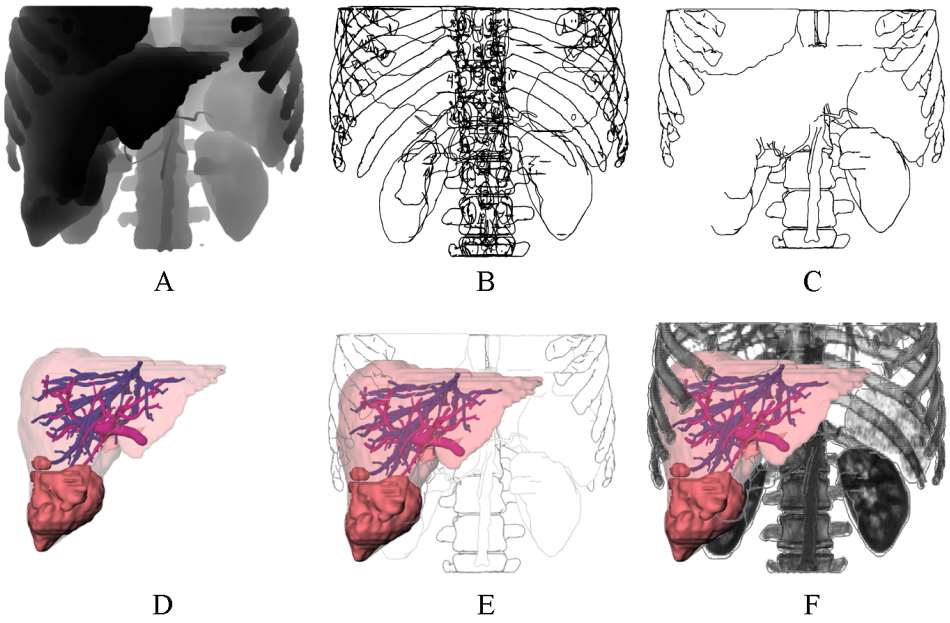


Abbildung 29: Ergebnisse der Zwischenschritte im Szenegraph gespeicherter Anweisungen für das kombinierte Rendering von Silhouetten, Oberflächen und direkter Volumendarstellung (nach Tietjen et al. [TIP05]). **A:** Rendern in den Tiefenpuffer. **B:** Zeichnen aller Konturen. **C:** Extrahieren der sichtbaren Linien unter Zuhilfenahme des Tiefenpuffers. **D:** Oberflächenrendering. **E:** Kombination aus Oberflächenrendering und Silhouette. **F:** Hybrides Bild nach Kombination mit DVR–Darstellung.

für alle Objekte gefüllt, die als Silhouette oder Oberfläche abgebildet werden sollen (s. Abbildung 29A). Für die Berechnung der Silhouetten werden Linien aus den Polygonnetzen extrahiert (s. Abbildung 29B). Verdeckte Linien können mithilfe des Tiefenpuffers entfernt werden (s. Abbildung 29C). Anschließend wird der Tiefenpuffer für das Rendering freigegeben. Um ungewollte Verdeckungen zu vermeiden, ist es essentiell, dass die unterschiedlichen Renderingtechniken in der richtigen Reihenfolge ausgeführt werden. Zunächst werden Oberflächendarstellungen gerendert (s. Abbildung 29D), gefolgt von gegebenenfalls stilisierten Silhouetten (s. Abbildung 29E). Beide Visualisierungen werden durch direktes Volumenrendering kombiniert und ergänzt (s. Abbildung 29F).

Tietjen et al. stellen heraus, dass dreidimensionale Visualisierungen insbesondere durch Silhouetten geeignet ergänzt werden könnten und hybride Darstellungen mit hervorgehobenen Kanten leichter verständlich seien.

3.3.4 Virtuelle Endoskopie nach Scharsach et al.

Virtuelle Endoskopie bezeichnet die Simulation eines chirurgisch durchgeführten, endoskopischen Eingriffs anhand von volumetrischen Datensätzen. Sie ermöglicht unter anderem den virtuellen Blick in Gefäße (*Angioskopie*), in das Innere des Darms (*Colonoskopie*) oder durch die Nase zum Gehirn (in der *Hypophysenchirurgie*). Dazu werden die Volumendaten als dreidimensionale Oberfläche visualisiert. Mithilfe einer virtuelle Kamera, die innerhalb der generierten Oberfläche platziert wird, ist die Betrachtung des Organinneren möglich [RSAH00]. Scharsach et al. präsentieren in [SHN+06] einen GPU–basierten, hybriden Ansatz für virtuelle Endoskopie.

Die Methode stellt Organwände mittels IVR–Verfahren als schattierte, semi–transparente Isoflächen dar. DVR–Raycasting visualisiert wichtige Kontextobjekte, beispielsweise die Gewebestruktur der Organwand, Nerven oder umliegende Organe. Zusätzlich ist die Integration dreidimensionaler Hilfsmittel, wie Pfeile und Raster, zur räumlichen Orientierung möglich.

In einem Vorverarbeitungsschritt wird der volumetrische Datensatz auf der CPU in Blöcke der Größe $8 \times 8 \times 8$ zerlegt. Für jeden Block lässt sich bestimmen, ob er für das Raycasting oder das Rendering der Isofläche relevant ist. Die Ergebnisse werden in zwei Bit–Arrays gespeichert. Im ersten Bit–Array wird jeder Block als aktiv markiert, wenn er nach Anwendung der Transferfunktion mindestens teilweise opak ist. Volltransparente Blöcke werden als inaktiv markiert. Die Informationen im zweiten Bit–Array ge-

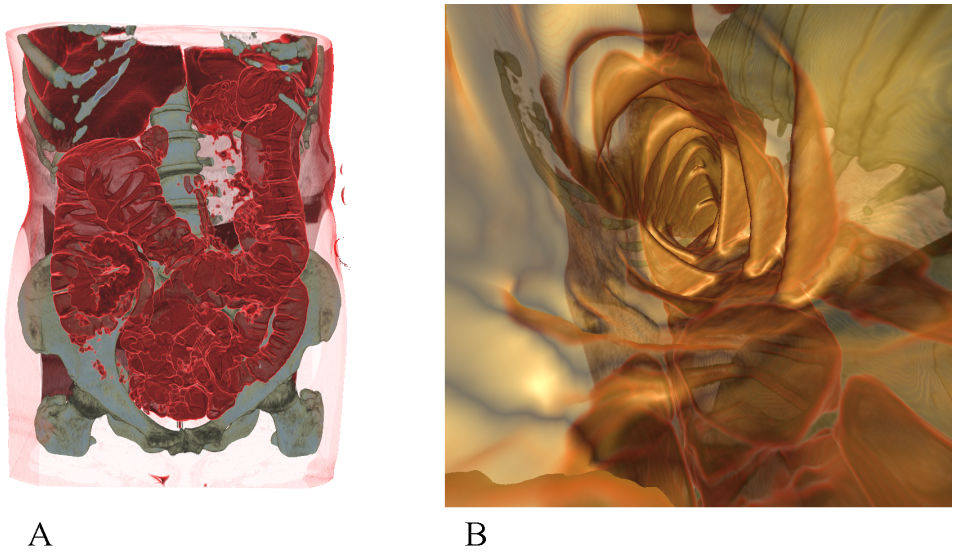


Abbildung 30: Visualisierung des Darms zur Durchführung einer virtuellen Colonoskopie (nach Scharsach et al. [SHN+06]). **A:** Hybride Darstellung des Colonoskopie–Datensatzes von außen. **B:** Hybride Darstellung während einer virtuellen Colonoskopie.

ben an, ob ein Schnittpunkt zwischen Block und Isofläche existiert. Dann gilt der Block im zweiten Bit–Array als aktiv, andernfalls als inaktiv. Beide Bit–Arrays werden an die GPU übergeben und dort zur Beschleunigung des Raycastings verwendet.

Auf der GPU wird zunächst das Raycasting initialisiert. Dazu wird der Startpunkt für jeden Strahl auf der vorderen Clippingebene (*near clipping plane*) bestimmt. Liegt der Startpunkt in einem inaktiven Block, wird stattdessen der vorderste Schnittpunkt mit einem aktiven Block gewählt. Damit kann die Länge jedes Strahls von der Startposition bis zum hintersten Schnittpunkt mit dem letzten aktiven Block berechnet werden. Sowohl die Startpositionen als auch die Längeninformationen werden in Texturen gespeichert. Das Raycasting läuft anschließend in zwei sukzessiven Schleifen ab:

- **First–Hit–Raycasting:** Zur Darstellung der Isoflächen werden Strahlen von ihrer Startposition aus durch alle im zweiten Bit–Array als aktiv markierten Blöcke verfolgt und auf Schnittpunkte mit Isoflächen getestet. An den Schnittpunkten werden Opazität und Farbe der getroffenen Isofläche anhand des Phong'schen Beleuchtungsmodells bestimmt. Am Ende der Traversierung wird der ermittelte Farbwert mit seiner Opazität gewichtet.
- **DVR–Raycasting:** Die Strahlen werden ein zweites Mal traversiert, durchqueren diesmal jedoch diejenigen Blöcke, die im ersten Bit–Array als aktiv markiert sind. Mit einer Transferfunktion wird für jeden Abtastpunkt die Farbe und Opazität an der entsprechenden Stelle berechnet. Ein Gesamtfarbwert wird akkumuliert und mit seiner Opazität gewichtet.

Durch Blending werden die Ergebnisse beider Raycasting–Schleifen kombiniert. Das Ergebnis ist eine nahtlose Verbindung von 3D–Volumendarstellung und Isofläche. Wie Abbildung 30 zeigt, ist das Verfahren sowohl für die Darstellung im Rahmen einer virtuellen Endoskopie als auch für Außenansichten geeignet. Dabei ist Echtzeitfähigkeit möglich, solange die Isoflächen opak dargestellt werden. Mit zunehmender Transparenz der Isoflächen sinkt die Framerate dagegen immens.

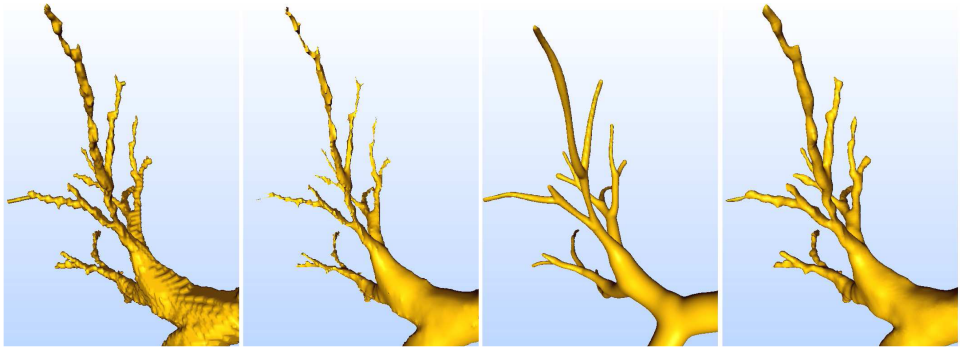
4 Konzept

Hybride Visualisierungsverfahren haben das Potenzial durch geeignete Kombination von direkter und indirekter Volumenvisualisierung die dreidimensionale Darstellung volumetrischer Datensätze zu verbessern. Die Visualisierung kann dabei von der hohen Qualität und der Echtzeitfähigkeit GPU–basierter DVR–Verfahren profitieren; gleichzeitig bieten IVR–Verfahren stärkere Plastizität und neue Interaktionsmöglichkeiten. Bisherige Ansätze nutzen unterschiedliche Visualisierungsmethoden zur Darstellung verschiedener Objekte eines volumetrischen Datensatzes (s. Abschnitt 3.3.1, 3.3.2, 3.3.3). Dabei ist die Zuordnung und Anwendung der jeweiligen Visualisierungsverfahren häufig nur durch eine umfangreiche Vorverarbeitung möglich, in der die einzelnen Objekte aus dem volumetrischen Datensatz extrahiert werden. Einen Schritt weiter gehen Scharsach et al. (s. Abschnitt 3.3.4), indem sie die nach innen gerichteten Organwände durch Isoflächen darstellen und mittels DVR–Visualisierung das äußere Gewebe desselben Organs ergänzen. Auf diese Weise wird ein Objekt mit zwei unterschiedlichen Methoden gerendert.

Im Rahmen der vorliegenden Arbeit wurde ein hybrides System zur Gefäßvisualisierung entwickelt, dass DVR– und IVR–Verfahren nicht strikt einzelnen Objekt(–teilen) zuordnet. Stattdessen soll eine mittels Raycasting generierte Darstellung von Gefäßen und umliegenden Objekten um eine indirekt bestimmte Gefäßoberfläche angereichert werden. Die Idee ist es, die Oberflächendarstellung nur an Stellen zu verwenden, an der sie ausreichend genau ist. Liegt die Abweichung vom Raycasting–Ergebnis hingegen über einem Schwellwert, wird an der entsprechenden Position das Volumen direkt dargestellt. Dadurch sollen Fehler in der Oberflächendarstellung abgefangen und gleichzeitig die Volumendarstellung plastischer gestaltet werden können. Die Herausforderung besteht in der Erzeugung fließender Übergänge zwischen den unterschiedlichen Visualisierungsformen.

Für die Umsetzung des Oberflächenrenderings innerhalb des hybriden Systems eignen sich die in Abschnitt 3.2.2 und Abschnitt 3.2.3 vorgestellten Verfahren. Tabelle 4 zeigt die Ergebnisse eines Vergleichs von Marching Cubes, Constrained Elastic Surface Nets, Convolution Surfaces und MPU Implicits hinsichtlich ihrer Genauigkeit und Darstellungsqualität.

Alle Visualisierungsverfahren sind in der Lage Gefäßenden zu schließen. Für Marching Cubes, Convolution Surfaces und MPU Implicits sind ungewollte Strukturen im Inneren der generierten Gefäßoberfläche auszuschließen. Für Constrained Elastic Surface Nets vermutet Schumann jedoch das Auftreten derartiger Fehler [Sch06a]. Glatte, organisch wirkende Repräsentationen können mit Constrained Elastic Surface Nets, Convolution Surfaces und MPU Implicits erzeugt werden (s. Abbildung 31B,C,D). Für alle drei Verfahren lässt sich die Glattheit anhand von Parametern regulieren. Keine organische Darstellung liefert Marching Cubes (s. Abbildung 31A). Stattdes-



A B C D

Abbildung 31: Gegenüberstellung der Bildqualität indirekter Visualisierungsverfahren anhand der Darstellung eines Bronchialasts (nach Museth et al. [MMY+08]). **A:** Marching Cubes liefert präzise, aber artefaktbehaftete Ergebnisse. **B:** Constrained Elastic Surface Nets generieren glatte Oberflächen, nehmen dafür jedoch eine starke Ausdünnung schmaler Strukturen in Kauf. **C:** Convolution Surfaces erzeugen glatte, jedoch wenig genaue Oberflächen. **D:** MPU Implicits ermöglichen präzise und gleichzeitig glatte Oberflächendarstellungen.

Verfahren	Genauigkeit	Organische Darstellung	Strukturen im Inneren	Gefäßenden
Marching Cubes	T, D, Q, S	nein	nein	ja
CESN	T, D, Q	ja, variabel	(ja)	ja
Convolution Surfaces	T, D, S	ja, variabel	nein	ja
MPU Implicits	T, D, Q, S, variabel	ja, variabel	nein	ja

Tabelle 4: Vergleich der Genauigkeit und der Darstellungsqualität von Marching Cubes, Constrained Elastic Surface Nets (CESN), MPU Implicits und Convolution Surfaces aus [Sch06a]. Die Beurteilung berücksichtigt die Genauigkeit der Topologie (T), des Durchmesserverlaufs (D), der Gefäßquerschnitte (Q) und dünner Strukturen (S). Angaben in Klammern stellen lediglich Vermutungen dar. Der Zusatz *variabel* gibt an, dass eine Eigenschaft durch Parametrisierung beeinflusst werden kann.

sen erzeugt der Algorithmus eine treppenartige Struktur die nur auf Kosten von Volumenverlust durch Glättung verringert werden kann. [Sch06a]

Entscheidend für die erfolgreiche Anwendbarkeit im hybriden System ist insbesondere die Genauigkeit der einzelnen Verfahren. Um fließende Übergänge zwischen der Oberflächenrepräsentation und der direkten Volumenvisualisierung schaffen zu können, sollte das verwendete IVR–Verfahren selbst bereits gute Ergebnisse erzielen. Eine genaue Gefäßdarstellung hinsichtlich der Topologie, des Durchmesserverlaufs, der Querschnitte und dünner Gefäße ist nur mithilfe von Marching Cubes und MPU Implicits möglich [Sch06a].

Datensatz	Marching Cubes	CESN	Convolution Surfaces	MPU Implicits
Bronchialbaum	166K (3 s)	166K (13 s)	201K (36 s)	180K (38 s)
Gefäßbaum der Leber	80K (2 s)	81K (5 s)	125K (15 s)	167K (14 s)
Zerebraler Gefäßbaum	115K (3 s)	115K (8 s)	93K (9 s)	142K (28 s)
Aneurysma	54K (1 s)	50K (2 s)	39K (11 s)	61K (5 s)

Tabelle 5: Vergleich der Effizienz von Marching Cubes, Constrained Elastic Surface Nets (CESN), MPU Implicits und Convolution Surfaces aus [MMY+08]. Die Tabelle enthält für jede Visualisierungstechnik die Anzahl generierter Polygone inklusive der für die Berechnung benötigten Zeit in Sekunden gemessen an vier unterschiedlichen Datensätzen[14].

Neben Genauigkeit und Darstellungsqualität ist auch die Effizienz der Visualisierungsverfahren abzuwägen. In einem hybriden System, das neben der Oberflächengenerierung auch Raycasting ausführt, ist eine hohe Performanz von Vorteil. Tabelle 5 stellt die Anzahl generierter Polygone und die Zeit, die Marching Cubes, Constrained Elastic Surface Nets, Convolution Surfaces und MPU Implicits für die jeweiligen Berechnungen benötigen, einander gegenüber. Anhand von vier unterschiedlichen Datensätzen haben Schumann et al. [MMY+08] Gefäßstrukturen visualisiert und Messungen durchgeführt. Das Ergebnis zeigt die überlegene Performanz des Marching Cubes Algorithmus insbesondere im Vergleich zu Convolution Surfaces und MPU Implicits.

Aufgrund ihrer vergleichsweise schlechten Genauigkeit wurden Constrained Elastic Surface Nets und Convolution Surfaces für die Integration in ein erstes prototypisches System bis auf weiteres verworfen. MPU Implicits versprechen zwar die optisch ansprechendsten Ergebnisse, würden die Performanz des Systems jedoch einschränken. Dagegen liefert der Marching Cubes Algorithmus schnell eine genaue, aber artefaktbehaftete Darstellung. Die Kombination mit Raycasting könnte bestenfalls helfen, den Treppenstufeneffekt zu kaschieren; schlimmstenfalls werden die Artefakte dadurch hervorgehoben. Auch um dies zu prüfen, wurde das hybride System als Kombination von Raycasting und Marching Cubes implementiert.

4.1 Anforderungen an das Visualisierungssystem

Das im Rahmen dieser Arbeit entwickelte hybride Renderingsystem soll die dreidimensionale Visualisierung von Blutgefäßen optimieren. Durch die zusätzliche Darstellung von Knochen oder Organen soll die räumliche Orientierung erleichtert werden. Als potentielle Anwendungsgebiete sind medizinische Aus– und Weiterbildung, Diagnoseunterstützung und Therapieplanung,

[14] Schumann et al. machen in [MMY+08] keine Angabe zur Implementierung der einzelnen Verfahren; vermutlich handelt es sich um GPU–basierte Implementierungen

insbesondere aber auch die Patientenaufklärung vorgesehen. Um in diesen Bereichen eingesetzt werden zu können, sind nachfolgende Anforderungen an die Gefäßvisualisierung vorrangig:

- **Genauigkeit:** Die wesentliche Eigenschaft von Darstellungen, die nicht nur zur medizinischen Ausbildung, sondern auch für die Gefäßdiagnostik eingesetzt werden sollen, ist die Genauigkeit. Um Gefäßerkrankungen, wie beispielsweise Aneurysmen, optimal einschätzen und früh erkennen zu können, muss der Visualisierungsalgorithmus die topologischen Merkmale Größe, Form und Position [LHH08] sowie die aus der CT–Aufnahme bekannten Detailstrukturen erhalten und in der 3D–Visualisierung wiedergeben. Auf eine organisch wirkende Darstellung kann zugunsten der korrekten Visualisierung von Verzweigungen, Gefäßquerschnitten und schmalen Strukturen verzichtet werden. Die Genauigkeit der hybriden Gefäßdarstellung soll anhand eines automatisierten Bildqualitätsmaßes im Vergleich zum reinen Raycasting bestimmt werden.
- **Verständlichkeit der Darstellung:** Die dreidimensionale Visualisierung soll für Mediziner selbsterklärend und für Laien gut verständlich sein. Sie soll eine anschauliche Möglichkeit bieten, bildgestützte Informationsgespräche im Rahmen einer diagnostischen Untersuchung oder einer anstehenden Operation mit den Patienten zu führen. Die Verständlichkeit der Darstellung soll anhand einer Experten– und einer Laienbefragung evaluiert werden.
- **Visuelle Qualität:** Das gerenderte Bild soll frei von störenden Artefakten sein, ohne dass dadurch die Genauigkeit der Darstellung beeinträchtigt wird. Zudem soll Plastizität in der Gefäßvisualisierung erkennbar sein. Als hybrides System soll der Ansatz die Vorteile von direktem und indirektem Volumenrendering miteinander kombinieren und die Nachteile der jeweiligen Methoden bestmöglich kompensieren. Dabei liegt besonderer Wert auf einer nahtlosen Verbindungen zwischen den unterschiedlichen Visualisierungsmethoden. Die Bildqualität soll anhand einer manuellen Identifikation von Artefakten sowie durch Befragung von Experten und Laien eingeschätzt werden.
- **Echtzeitfähigkeit der Anwendung:** Die Generierung des Oberflächenmodells soll ausreichend schnell und das anschließende hybride Rendering echtzeitfähig sein. In diesem Zusammenhang ist auch die Möglichkeit zur Interaktion mit dem Gefäßmodell gewünscht. Um ein optimales Ergebnis zu erzielen, wird die Auslagerung der Renderingmethode auf die GPU sowie der Einsatz von GPGPU angestrebt. Die Performanz soll anhand der Framerate und der Berechnungszeit pro Frame für unterschiedliche Datensätze und Systeme gemessen und ausgewertet werden.

4.2 Gefäßvisualisierungspipeline

Das in dieser Arbeit entwickelte hybride Renderingsystem stellt lediglich einen Teil einer Volumenvisualisierungspipeline dar. Die Vorverarbeitung des Volumendatensatzes und eine Segmentierung der kontrastierten Gefäße werden als gegeben vorausgesetzt. Das Segmentierungsergebnis wird zusammen mit dem CT–Datensatz als Eingabe für die in Abbildung 32 dargestellte Visualisierungspipeline verwendet. Einmalig wird eine Gefäßoberfläche durch Marching Cubes generiert. In zwei Renderpässen werden die Oberfläche und eine Volumendarstellung des Gefäßes inklusive Kontextinformation gezeichnet. Ein dritter Renderpass integriert die Oberfläche in die Volumendarstellung. Die einzelnen Verfahrensschritte werden im Folgenden näher erläutert.

Oberflächengenerierung mit Marching Cubes. Anhand des Segmentierungsergebnisses wird eine Oberflächendarstellung des Gefäßbaums generiert. Das hybride System nutzt dazu den Marching Cubes Algorithmus. Für eine effiziente Speicherverwaltung werden Histogramm Pyramiden (*Histo–Pyramiden*) verwendet. Sie enthalten in der untersten Ebene die Anzahl zu generierender Dreiecke pro Volumenzelle. Höhere Ebenen akkumulieren benachbarte Werte aus unteren Ebenen. Zur Berechnung von Oberflächenvertices inklusive der zugehörigen Normalen wird die Pyramide von oben nach unten traversiert. Histo–Pyramiden wurden erstmals von Ziegler et al. in [ZTTS06] vorgestellt. Eine Optimierung des Verfahrens liefern Smistad et al. [SEL12, Smi].

Das hybride Renderingsystem führt die Oberflächenrekonstruktion nach Smistad et al. als Vorverarbeitungsschritt der Volumenvisualisierung durch. Die Histo–Pyramide wird von unten nach oben aufgebaut (s. exemplarisch Abbildung 33A). Zur Konstruktion der untersten Pyramidenebene werden alle Volumenzellen des segmentierten Datensatzes betrachtet. Für jede Volumenzelle werden die Intensitätswerte in den acht zugehörigen Voxeln ausgelesen. Anhand der räumlichen Verteilung der Intensitätswerte lässt sich die Anzahl zu generierender Dreiecke für die betrachtete Volumenzelle ableiten und in der untersten Ebene der Histo–Pyramide speichern. Die nächsthöhere Ebene der Histo–Pyramide akkumuliert jeweils $2 \times 2 \times 2$ Werte der darunter

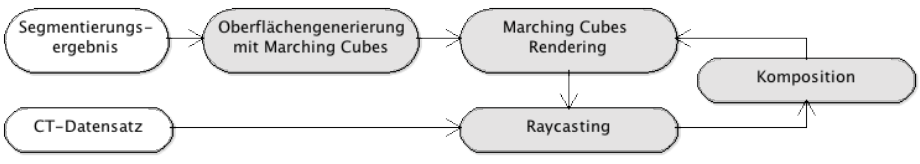


Abbildung 32: Visualisierungspipeline des im Rahmen der Arbeit entwickelten hybriden Renderingsystems. Der CT–Datensatz sowie die zugehörige Gefäßsegmentierung werden als Eingabe erwartet.

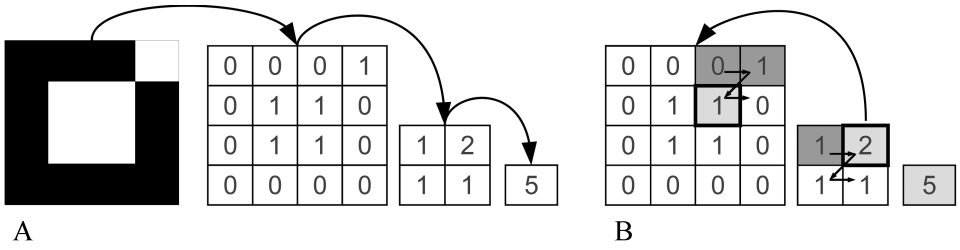


Abbildung 33: Aufbau und Traversierung einer Histo–Pyramide im zweidimensionalen Raum (modifiziert nach Smistad et al. [SEL12]). **A:** Aufbau der Histo–Pyramide. Für ein binäres Bild (links) werden in der untersten Ebene der Histo-Pyramide (zweites Bild von links) weiße Pixel lokalisiert. Die zweite Ebene (zweites Bild von rechts) fasst jeweils vier quadratisch angeordnete Zellen durch Summierung zu einem einzelnen Wert zusammen. Die oberste Ebene (rechts) enthält die Anzahl aller weißen Pixel im binären Bild. **B:** Traversierung der Histo–Pyramide auf der Suche nach dem dritten weißen Pixel. In jeder Ebene werden die einzelnen Zellen in einem Z–Muster betrachtet, bis die Summe der Werte aller durchlaufenen Zellen einschließlich der aktuellen Zelle größer oder gleich der ID des gesuchten Pixels ist. Danach beginnt die Traversierung der nächsten Ebene im zur aktuellen Zelle korrespondierenden 2×2 Block. In der untersten Ebene erreicht der Algorithmus diejenige Zelle, in der das dritte weiße Pixel lokalisiert ist.

gelegenen Ebene. Auf diese Weise werden alle übergeordneten Ebenen durch sukzessive Zusammenfassung der Werte untergeordneter Ebenen konstruiert. Die höchste Ebene enthält nur noch genau einen Wert, der die Summe aller Dreiecke der zu generierenden Oberfläche repräsentiert. Diese Summe wird verwendet, um eine ausreichende Menge an Speicher für die Oberflächenvertices und die zugehörigen Normalen zu reservieren.

Zur Oberflächenrekonstruktion wird die Histo–Pyramide von der zweithöchsten Ebene aus bis zur untersten Ebene einmal für jedes zu generierende Dreieck traversiert (s. exemplarisch Abbildung 33B). Jede Ebene wird in einem Z–Muster durchlaufen. Dabei betrachtete Werte werden akkumuliert. Für die Position, an der diese Summe größer oder genauso groß wird, wie der Index des aktuell betrachteten Dreiecks, wird der korrespondierende $2 \times 2 \times 2$ Würfel der nächstgrößeren Ebene durchlaufen. In der untersten Ebene terminiert der Algorithmus und gibt die Position einer Volumenzelle inklusive der Anzahl in ihr zu erzeugender Dreiecke zurück. Anhand dieser Werte und unter Hinzunahme einer Tabelle, die Informationen zur Lokalisation der Schnittpunkte zwischen Oberfläche und Volumenzelle enthält, kann das aktuell gesuchte Dreieck innerhalb der Volumenzelle erzeugt werden. Gleichzeitig wird eine gradientenbasierte Normalenberechnung durchgeführt, wie sie Lorensen und Cline in [LC87] vorschlagen. Sowohl die Vertices als auch die Vertexnormalen werden in einem Buffer gespeichert.

Marching Cubes Rendering. Die im Rahmen der Vorverarbeitung mittels Marching Cubes generierten Dreiecke werden in einem ersten Renderpass gezeichnet. In die Berechnung der Farbwerte werden der ambiente und der diffuse Term aus dem Phong'schen Beleuchtungsmodells einbezogen. Eine Tesselierung mit *Curved PN–Triangles* [VPBM01] zur weiteren Verfeinerung des Polygonnetzes mit gleichzeitiger Abrundung der Oberfläche ist optional.

Für jedes Pixel des gerenderten Bildes werden die Farbinformationen und die Tiefenwerte in einem Buffer gespeichert.

Raycasting. Mittels Raycasting werden in einem zweiten Renderpass die nicht–segmentierten CT–Daten verarbeitet. Dazu wird der komplette Datensatz in eine quaderförmige Hüllgeometrie eingepasst. Das im vorgestellten hybriden System integrierte Raycasting entspricht zu großen Teilen dem in Abschnitt 3.1.3 erläuterten Basisalgorithmus.

Durch jedes Pixel der Bildebene wird jeweils ein Sehstrahl von der Kamera aus in den volumetrischen Datensatz gesendet. Für jeden Strahl werden Volumeneintrittspunkt und Richtung bestimmt. Anschließend wird das Volumen entlang des Strahls traversiert und ein Farbwert für das aktuell betrachtete Pixel akkumuliert. Der Strahl wird dazu in regelmäßigen Schritten abgetastet.

Für jeden Abtastpunkt wird der Intensitätswert im volumetrischen Datensatz abgefragt. Mithilfe einer vorab definierten Transferfunktion lässt sich ein Intensitätswert in einen Farb– und einen Opazitätswert überführt. Die Transferfunktion des hybriden Systems ist derart formuliert, dass Gefäße (rot), Kalzifizierungen (weiß) und Knochen (vornehmlich weiß) dargestellt werden.

Um eine Marching Cubes Oberfläche weitestgehend nahtlos in das Raycasting Ergebnis integrieren zu können, müssen innerhalb der Volumendarstellung opake Gefäßsstrukturen erzeugt werden. Zur Ermittlung der Opazität wird deshalb zusätzlich ein variabler Isowert einbezogen. Anhand dieses Isowerts soll das Gefäß möglichst genau im volumetrischen Datensatz detektiert werden können. Entspricht nun der Intensitätswert an einem Abtastpunkt dem festgelegen Isowert, so wird die Opazität an dieser Stelle ungeachtet der Transferfunktion auf 1.0 gesetzt. Zusätzlich werden im Falle eines Isowert–Treffers der Tiefenwert der aktuellen Abtastposition sowie die ambiente und diffuse Beleuchtung an der entsprechenden Stelle bestimmt. Die für die Phong'sche Beleuchtung notwendigen Vertexnormalen werden gradientenbasiert approximiert.

Im letzten Verarbeitungsschritt werden alle entlang eines Strahls ermittelten Farb– und Opazitätswerte bis zum aktuellen Abtastpunkt akkumuliert. Mithilfe einer festen Schrittweite wird der nächste Abtastpunkt auf dem Strahl ermittelt. Spätestens wenn der neue Abtastpunkt die Hüllgeometrie des volumetrischen Datensatzes verlässt, terminiert die Strahlenverfolgung.

Um die Performanz des Raycastings zu verbessern, wird als zusätzliche Bedingung Early Ray Termination ab einer Opazität von 0.95 eingesetzt. Mit Stochasic Jittering werden Holzmaserung–ähnliche Artefakte kaschiert.

Ist die Traversierung eines Strahls abgeschlossen, wird der akkumulierte Farbwert für das Pixel zusammen mit dem anhand des Isowerts ermittelten Tiefenwert in einen Buffer geschrieben.

Komposition. Das hybride System kombiniert Marching Cubes und Raycasting durch Blending der jeweils berechneten Bilder in einem dritten Renderpass. Die Marching Cubes Oberfläche ist dabei nur für diejenigen Pixel relevant, an denen die Differenz der Tiefenwerte aus Marching Cubes und Raycasting einen Schwellwert nicht überschreitet. Zusätzlich wird die Verwendung des Marching Cubes Ergebnisses unterdrückt, wenn ein Pixel im Raycasting Ergebnis annähernd weiß ist. Damit lässt sich eine Überdeckung von Kalzifizierungen durch die Marching Cubes Oberfläche verhindern.

Im Falle einer Kombination geht das Marching Cubes Ergebnis zu 70 % und das Raycasting Ergebnis zu 30 % in die Gesamtpixelfarbe ein.

5 Implementierung

Das hybride System, das in der vorliegenden Arbeit entwickelt wurde, nutzt eine Kombination aus Raycasting und Marching Cubes zur Darstellung vaskulärer Strukturen. Mittels Raycasting werden dabei neben Blutgefäßen auch Kalzifizierungen und Knochen visualisiert; der Marching Cubes Algorithmus ergänzt die direkte Volumendarstellung um eine indirekt erzeugte Oberfläche. Eine Komposition aus beiden Verfahren wird ausschließlich für diejenigen Orte im Bildraum angewendet, an denen das Marching Cubes Ergebnis die Volumendarstellung ausreichend genau annähert. Ist dies nicht der Fall, wird an der entsprechenden Position lediglich das Raycasting Ergebnis gerendert.

Die Implementierung wurde plattfomübergreifend für *OS X 10.9 Mavericks* und *Microsoft Windows 8* in C++ realisiert. Für das Rendering wird die Grafik–API *OpenGL 4.0* genutzt. Zusätzlich werden *glfw 2.7*, *glew 1.10.0* und *glm 0.9.4.5* verwendet. Mithilfe von *OpenCL 1.2* wird der Einsatz von GPGPU (*General Purpose Computation on Graphics Processing Unit*) für die Berechnung der Marching Cubes Oberfläche ermöglicht (s. dazu Abschnitt 5.2).

Zur Beschleunigung des hybriden Systems wird neben der Oberflächenberechnung durch Marching Cubes auch das Rendering auf die GPU (*Graphics Processing Unit*) ausgelagert. Während die CPU (*Central Processing Unit*, dt.: Hauptspeicher) Daten vornehmlich sequentiell verarbeitet, ist die GPU auf parallele Datenverarbeitung spezialisiert. Zusätzlich bietet die GPU schnellen Zugriff auf einen eigenen Speicher. Beides ermöglicht effizientere Berechnungen für hochparallele Prozesse [Sch05].

Im Folgenden werden die einzelnen Komponenten des hybriden Systems anhand von Code–Ausschnitten und schematischen Abbildungen vorgestellt. Der vollständige Source Code ist über den Hosting–Dienst GitHub frei zugänglich (s. Anhang A).

5.1 Rahmenprogramm

Das Rahmenprogramm des hybriden Renderingsystems setzt sich aus sieben Hauptklassen zusammen (s. Abbildung 34). Sie alle sind über die Klasse `main` miteinander verbunden. Als Ausgangspunkt des Systems steuert `main` sowohl die Vorverarbeitung als auch die Renderschleife. Dazu wird zunächst eine Instanz der Klasse `RAWReader` erzeugt.

Ein CT–Datensatz im RAW–Format und die daraus erzeugte Segmentierung der Blutgefäße – ebenfalls im RAW–Format – werden mithilfe des `RAWReaders` in zwei Objekte des Typs `Volume` umgewandelt. Dazu liest der `RAWReader` zunächst für jeden Datensatz die zugehörige Info–Datei aus. Diese enthält Angaben über Anzahl und Auflösung der einzelnen Schichtbilder,

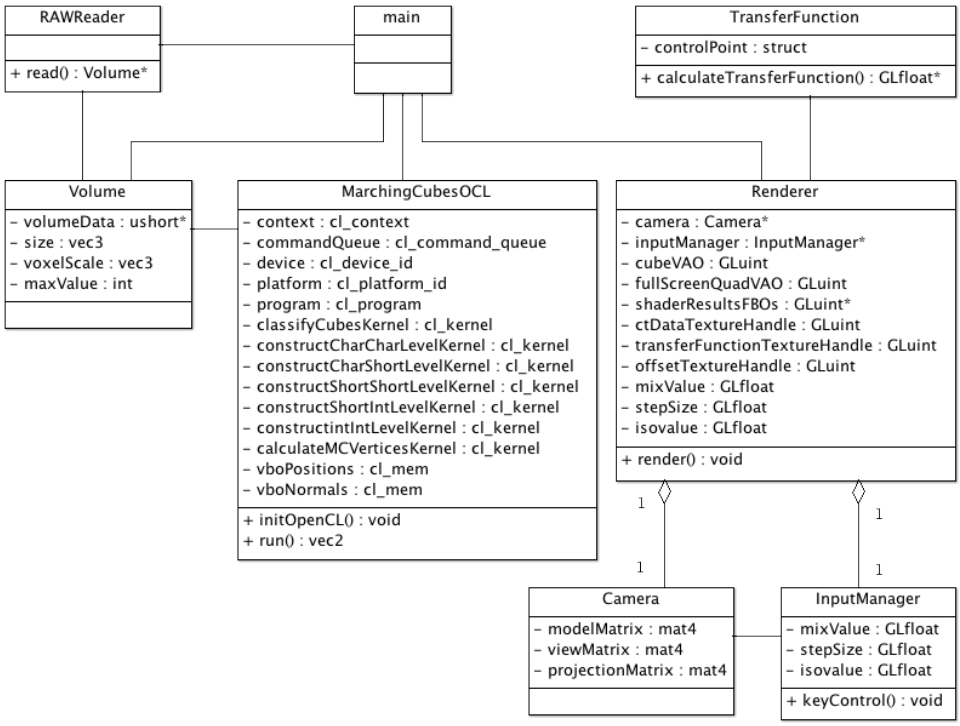


Abbildung 34: Darstellung der Aufteilung des hybriden Renderers in sieben Hauptklassen mit Angabe wichtiger Attribute und Methoden.

über die Abstände zwischen den Voxeln, minimale und maximale Intensitätswerte sowie das zur Speicherung verwendete Bit–Format. Anhand der Informationen wird für jeden Datensatz Speicher reserviert. Der Inhalt der RAW–Dateien kann nun, wie in Listing 1 beschrieben, ausgelesen und in jeweils einem Array gespeichert werden.

Um den Marching Cubes Algorithmus zur Generierung des Polygonnetzes mithilfe von Histo–Pyramiden effizienter umsetzen zu können, muss der

```
1  // Initialize variables
2      int volumeDataSize = size.x * size.y * size.z;
3      unsigned short *volumeData =
4                              new unsigned short[volumeDataSize];
5
6      FILE *file = std::fopen(filename.c_str(), "rb");
7
8      fread(    volumeData,
9                sizeof(unsigned short),
10               volumeDataSize,
11               file );
```

Listing 1: Auslesen einer RAW–Datei unter Mac OS X.

```
1  // Find largest size
2     int maxSize = max(max(size.x, size.y), size.z);
3
4  // Find smallest power of two greater than maxSize
5     int i = 0;
6     while (std::pow(2, i) < maxSize) i++;
7     int newSize = std::pow(2, i);
8
9  // Create voxel array of the new size and fill it with zeros
10    int tempVolumeDataSize = newSize * newSize * newSize;
11    unsigned short *tempVolumeData =
12                   new unsigned short[tempVolumeDataSize];
13    for(int j = 0; j < tempVolumeDataSize; j++)
14       tempVolumeData[j] = 0;
15
16 // Fill the new voxel array with volume data
17    for (int x = 0; x < size.x; x++) {
18       for (int y = 0; y < size.y; y++) {
19          for (int z = 0; z < size.z; z++) {
20             tempVolumeData[x + y * newSize +
21                            z * newSize * newSize]
22                = volumeData[0][x + y * (int)size.x +
23                     z * (int)size.x * (int)size.y];
24          }
25       }
26    }
27
28 // Update data
29    delete[] volumeData[0];
30    volumeData[0] = tempVolumeData;
```

Listing 2: Übertragung der Segmentierungsdaten in ein $2^x \times 2^x \times 2^x$ großes Array.

Segmentierungsdatensatz verlustfrei in ein Array der Größe $2^x \times 2^x \times 2^x$ übertragen werden (s. Listing 2). Der CT–Datensatz bleibt unverändert.

Die weitere Verarbeitung der CT– und Segmentierungsdaten ist in Abbildung 35 dargestellt. Nach Konvertierung beider Datensätze in jeweils ein `Volume`–Objekt wird das segmentierte Volumen an die GPU übergeben. Dort wird mit dem Marching Cubes Algorithmus ein Polygonnetz aus den Segmentierungsdaten extrahiert. Auf der CPU werden die ermittelten Vertices und Normalen, gekapselt in einem *Vertex Array Object (VAO)*, an das Shader–Programm des ersten Renderpasses angebunden. Dieser zeichnet das Polygonnetz auf der GPU in ein *Framebuffer Object (FBO)*. In einem zweiten Renderpass wird mittels Raycasting eine Volumendarstellung aus den CT–Daten berechnet und ebenfalls in ein FBO gezeichnet. Die Ausgabe der kombinierten Bilder benötigt einen dritten Renderpass auf der GPU. Wird das System nicht beendet, werden die drei Renderpässe erneut ausgeführt.

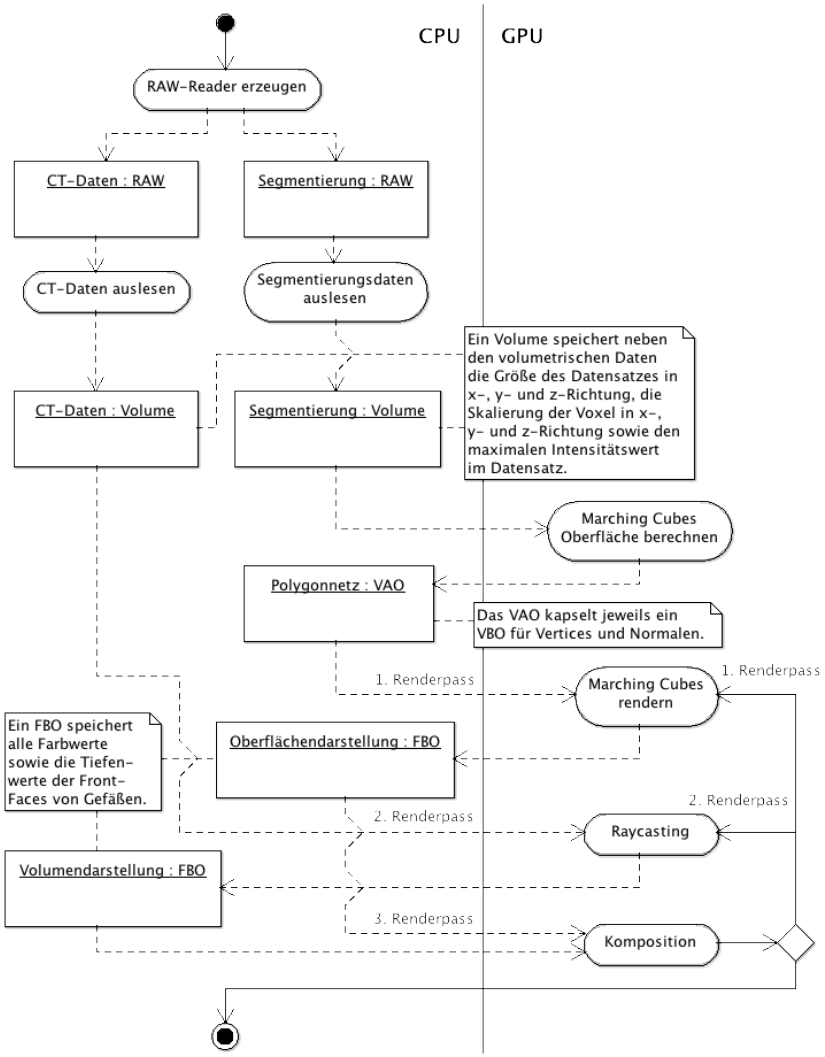


Abbildung 35: Übersicht über den grundlegenden Aktivitäts- und Datenfluss zur Verarbeitung des CT- und des Segmentierungsdatensatzes.

5.2 OpenCL–Kernel

Zur Generierung eines Polygonnetzes auf Basis segmentierter Daten wurde der Marching Cubes Algorithmus über GPGPU auf der Grafikhardware realisiert. Hatte die GPU in ihren Anfängen noch lediglich grafische Funktionalität, wie beispielsweise die Beschleunigung von Rendering oder Texture Mapping durch Parallelisierung, sind über GPGPU mittlerweile auch allgemeine, nicht–grafische Berechnungen auf dem Grafikprozessor möglich [SEL12, San13]. Spezielle Programmiersprachen, darunter OpenCL (*Open Computing Language*) und CUDA (*Compute Unified Device Architecture*),

vereinfachen den Einsatz von GPGPU.

Die Verwendung der Grafikhardware zur Oberflächengenerierung mittels Marching Cubes liegt nahe, da der Algorithmus jede Volumenzelle unabhängig vom restlichen Datensatz betrachtet. Zur parallelen Verarbeitung auf der GPU wird für jede Volumenzelle genau eine Verarbeitungseinheit (unter OpenCL: *Work–Item*) genutzt. Insbesondere für große Datensätze kann so eine hohe Beschleunigungen erzielt werden.

Eine OpenCL–Implementierung des Marching Cubes Algorithmus liefern Smistad et al. [SEL12, Smi]. Für die Erzeugung des Polygonnetzes im hier vorgestellten hybriden Renderingsystem wird ihre Variante umgesetzt.

Die Methode `initOpenCL()` der Klasse `MarchingCubesOCL` initialisiert alle benötigten OpenCL–Komponenten: das *device*, den *context*, die *command queue*, das *program* inklusive seiner *kernel* und die zur Speicherung der Daten verwendeten *memory*–Objekte. Die GPU wird als das für die Berechnungen zu verwendende *device* festgelegt. Zusammen mit den übrigen OpenCL–Komponenten wird das *device* vom *context*–Objekt verwaltet. Die insgesamt sieben *kernel* enthalten den Source Code zur Berechnung der Marching Cubes Oberfläche mithilfe von Histo–Pyramiden. Das *program* fasst sämtliche *kernel* zusammen. Über die *command queue* ordnet es die Ausführung der *kernel* durch das *device* an. Die Ergebnisse werden in diversen *memory*–Objekten gespeichert.

Im hybriden Renderingsystem sind die *memory*–Objekte in Form von Buffern implementiert. Durch das Schreiben in 3D–Images ließe sich vermutlich eine bessere Performanz erzielen; die Funktionalität steht unter *Mac OS X 10.9 Mavericks* jedoch nicht zur Verfügung. Im Gegensatz dazu ist ein effizientes Auslesen aus 3D–Images sowohl unter Mac, als auch unter Windows möglich. Unter anderem der Segmentierungsdatensatz wird aus diesem Grund als 3D–Image an die GPU übergeben.

Nach der erfolgreichen Initialisierung aller oben genannten Objekte werden die im Folgenden beschriebenen Kernel zur Berechnung eines Polygonnetzes mithilfe von Marching Cubes ausgeführt.

`classifyCubesKernel`. Der `classifyCubesKernel` wird für jedes Voxel des Segmentierungsdatensatzes parallel durch jeweils ein Work–Item ausgeführt. Für die rechte obere Volumenzelle des Voxels (s. Abbildung 36) bestimmt er die Anzahl zu generierender Dreiecke. Dazu wird der Segmentierungsdatensatz als 3D–Image (*read–only*) an die GPU übergeben und verarbeitet. Der Ablauf ist in Abbildung 37 schematisch dargestellt.

Anhand der ID des Work–Items, das die Berechnungen für das aktuell behandelte Voxel vornimmt, kann die Position des Voxels im Segementierungsdatensatz bestimmt werden. Für das Voxel und seine sieben, rechts oben gelegenen Nachbarn werden die binären Farbwerte aus dem 3D–Image

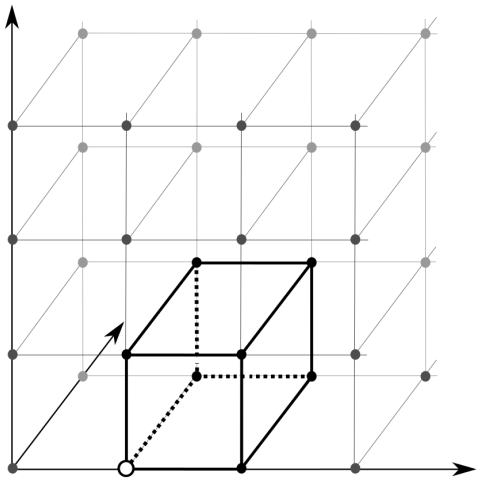


Abbildung 36: Rechte obere Volumenzelle des weiß markierten Voxels im dreidimensionalen Datensatz.

ausgelesen. Gehört ein Voxel zum segmentierten Gefäß, beträgt der Farbwert 1, andernfalls hat das Voxel den Farbwert 0. Alle vorliegenden Einsen werden abhängig von ihrer Position in der aktuell betrachteten Volumenzelle zunächst mit einem Shiftoperator bitweise verschoben und anschließend miteinander verodert. Das Ergebnis ist der Index für eine Wertetabelle, welche die Anzahl zu generierender Dreiecke enthält – insgesamt existieren 256 möglichen Farbwertverteilungen innerhalb einer Volumenzelle.

Der Index wird für die spätere Verwendung im `calculateMCVertices-Kernel` in ein Array geschrieben. In die unterste Ebene der Histo–Pyramide wird die Anzahl zu generierender Dreiecke, abgelesen aus der Wertetabelle, eingetragen.

`constructHPLevelKernel.` Um den Speicher der Grafikkarte nicht unnötig zu belasten, verwenden die einzelnen Ebenen der Histo–Pyramide unterschiedliche Datentypen. Es ist möglich, den maximal benötigten Speicher für jede Ebene zu bestimmen, da eine Volumenzelle nie mehr als fünf Dreiecke enthalten kann. Der Datentyp *char* bietet mit 8 Bit für die ersten beiden Ebenen der Histo–Pyramide (mit zu speichernden Maximalwerten von $5 \cdot 8 = 40$) ausreichend Speicherkapazität. Die nächsten drei Ebenen nutzen den Datentyp *short* mit 16 Bit. Erst für nachfolgende Ebenen wird der Datentyp *int* mit einer maximalen Speicherkapazität von 32 Bit verwendet.

Aufgrund der unterschiedlichen Datentypen werden insgesamt fünf Konstruktionskernel für den Aufbau der Histo–Pyramide benötigt. Jeder Kernel führt den in Abbildung 38 dargestellten Ablauf aus. Sie unterscheiden sich

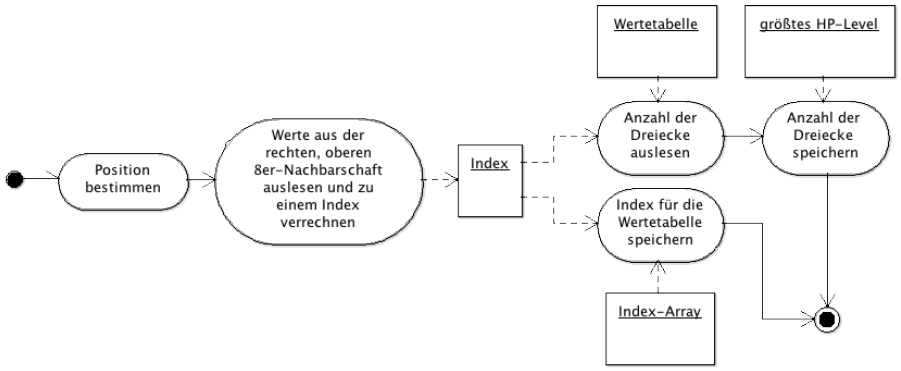


Abbildung 37: Berechnung der Anzahl zu generierender Dreiecke pro Volumenzelle im `classifyCubesKernel`. Der Segmentierungsdatensatz wird als 3D–Image an die GPU übergeben, verarbeitet und das Ergebnis in die unterste Ebene der Histo–Pyramide eingetragen. Die ermittelten Indizes werden in einem Array gespeichert.

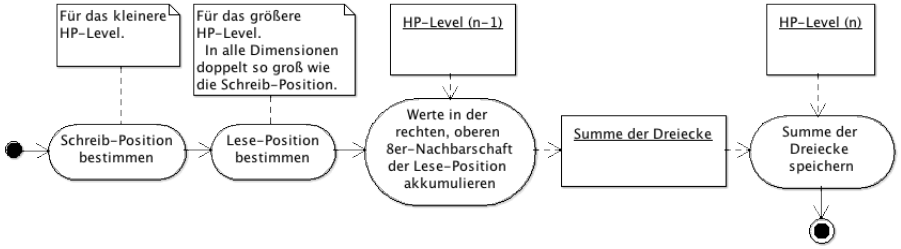


Abbildung 38: Aufbau der Histo–Pyramide zur beschleunigten Berechnung des Polygonnetzes im `constructHPLevelKernel`. Die zuletzt berechnete Ebene der Histo–Pyramide wird als Eingabe verwendet, in der nächstkleineren Ebene zusammengefasst und ausgegeben.

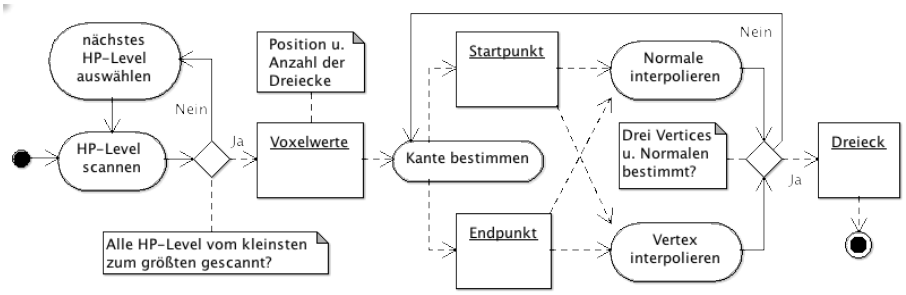


Abbildung 39: Berechnung eines Dreiecks für das Marching Cubes Polygonnetz im `calculateMCVerticesKernel`. Aus der Histo–Pyramide, den Segmentierungsdaten und dem Index–Array aus dem `classifyCubesKernel` werden Vertices und Normalen eines Dreiecks generiert, in jeweils einen Buffer gespeichert und an die CPU zurückgegeben.

lediglich im Datentyp der eingehenden beziehungsweise ausgehenden Ebene der Histo–Pyramide (*char–char*, *char–short*, *short–short*, *short–int*, *int–int*).

Als Eingabe verwenden die Konstruktions–Kernel die jeweils zuletzt bestimmte Ebene der Histo–Pyramide. Anhand der in ihr gespeicherten Werte, wird der Inhalt der nächstkleineren Ebene bestimmt. Die parallele Berechnung aller Werte der kleineren Ebene wird durch jeweils ein Work–Item ausgeführt.

Die Schreibposition in der kleineren Ebene lässt sich direkt aus der ID des Work–Items ableiten. Wird die Schreibposition in alle drei Raumrichtungen verdoppelt, folgt daraus die Leseposition in der größeren Ausgangsebene. Alle acht Werte aus der oberen rechten Nachbarschaft der Leseposition werden akkumuliert. Das Ergebnis wird an der Schreibposition in die kleinere Ebene eingetragen. Auf diese Weise wird die Anzahl der Dreiecke in benachbarten Volumenzellen über alle Ebenen hinweg wie in einem Octree zusammengefasst.

Die Konstruktion der Histo–Pyramide endet mit der Erzeugung einer $2 \times 2 \times 2$ großen Ebene. Sie wird auf der CPU vollständig ausgelesen. Die acht Werte ergeben summiert die Gesamtanzahl aller Dreiecke, die für das Polygonnetz generiert werden müssen. Damit ist die notwendige Allokation des Speichers für Vertices und Normalen möglich. Im `calculateMCVertices-Kernel` können anschließend alle Dreiecke berechnet und in den reservierten Speicher geschrieben werden.

`calculateMCVerticesKernel`. Der `calculateMCVerticesKernel` wird parallel für jedes zu generierende Dreieck ausgeführt. Als Eingabedaten werden ihm alle Ebenen der Histo–Pyramide, das 3D–Image des segmentierten Datensatzes und das Index–Array aus dem `classifyCubesKernel` übergeben.

Anhand der `triangleID`, die der Work–Item ID entspricht, lässt sich feststellen, welches Dreieck verarbeitet wird. Um die Position der Volumenzelle zu bestimmen, in der das Dreieck generiert werden soll, werden nacheinander alle Ebenen der Histo–Pyramide, beginnend mit der kleinsten, gescannt. Die Vorgehensweise entspricht im Wesentlichen der Traversierung eines Octrees.

Die kleinste Ebene der Histo–Pyramide teilt den zu Grunde liegenden Segmentierungsdatensatz in acht gleich große Zellen auf. Jede dieser Zellen wird in der nächstgrößeren Ebene wiederum in acht Zellen unterteilt. In der untersten Ebene sind die einzelnen Volumenzellen des Segmeniterungsdatensatzes dargestellt. Über die gesamte Histo–Pyramide hinweg wird akkumuliert, wie viele Dreiecke in den jeweiligen Zellen generiert werden müssen.

Die `triangleID` wird in der Implementierung genutzt, um festzustellen, wie viele Dreiecke in Zellen erzeugt werden müssen, die in der Traversierungsreihenfolge weiter vorne liegen. Enthält beispielsweise die erste Zelle der obersten Histo–Pyramiden–Ebene den Wert 8, folgt daraus, dass in dieser Zelle insgesamt acht Dreiecke mit den IDs 0 bis 7 generiert werden.

Beträgt die `triangleID` dagegen den Wert 10, handelt sich bei dem aktuell betrachteten Dreieck folglich um das elfte zu generierende Dreieck. Es kann somit nicht in einer Volumenzelle liegen, die in der ersten Zelle der obersten Histo–Pyramiden–Ebene zusammengefasst ist. Zur Untersuchung der zweiten Zelle der obersten Histo–Pyramiden–Ebene müssen die zuvor verworfenen acht Dreiecke aus der ersten Zelle berücksichtigt werden (vgl. `neighborsTriangleSum`). Das heißt, dass innerhalb der zweiten Zelle, wenn sie beispielsweise einen Wert von 5 speichert, die Dreiecke mit den IDs 8 bis 12 erzeugt werden müssen. Damit ist das gesuchte Dreieck in der obersten Ebene der Histo–Pyramide gefunden. In der darunter gelegenen Ebene muss es lediglich in den acht Zellen gesucht werden, die in der zweiten Zelle der obersten Histo–Pyramiden Ebene zusammengefasst sind. Auch hier ist für den korrekten Vergleich mit der `triangleID` die Berücksichtigung der verworfenen Dreiecke aus der ersten Zelle der obersten Histo–Pyramiden–Ebene notwendig (vgl. `currentTriangleSum`).

Für die Implementierung dieses Verfahrens (s. Abbildung 39) wird folglich in der obersten Ebene der Histo–Pyramide an Position $(0, 0, 0)$ mit der Traversierung begonnen. Da zu Beginn noch kein Dreieck verworfen worden sein kann, wird die `currentTriangleSum`, die über alle Histo–Pyramiden–Ebenen hinweg die Zahl der verworfenen Dreiecke speichert, mit Null initialisiert. Der Variable `neighborsTriangleSum`, die die Anzahl zu generierender Dreiecke innerhalb einer Histo–Pyramiden–Ebene akkumuliert, wird der Wert an Position $(0, 0, 0)$ zugewiesen. Anhand des Vergleichs der `neighborsTriangleSum` mit der `triangleID` kann nun festgestellt werden, ob das zu generierende Dreieck in einer Volumenzelle generiert werden muss, die an Position $(0, 0, 0)$ in der obersten Ebene der Histo–Pyramide zusammengefasst ist. Das Ergebnis wird in einem acht Stellen umfassenden Array (`cubeOffsetIndex`) eingetragen: Eine Eins zeigt an, dass die `triangleID` größer ist und damit in einer nachfolgenden Zelle erzeugt werden muss; ist die `triangleID` dagegen kleiner oder gleich der akkumulierten `neighborsTriangleSum` wird eine Null im Array notiert. Anschließend wird die `neighborsTriangleSum` um den Wert aus der zweiten Zelle der obersten Histo–Pyramiden–Ebene erhöht und erneut mit der `triangleID` verglichen (s. Listing 3). Auf diese Weise werden alle acht Zellen der Ebene der Reihe nach verarbeitet. Die Summe, der im `cubeOffsetIndex` gespeicherten Einsen liefert den Index für eine Wertetabelle, aus der die Startposition für die Verarbeitung der nächstgrößeren Ebene ausgelesen wird. In `currentTriangleSum` wird die Anzahl der Dreiecke gespeichert, die bis zu dieser neuen Startposition übersprungen worden sind. Die Berechnung der `currentTriangleSum` ist in Listing 4 implementiert.

Der Scan der zweiten Histo–Pyramiden–Ebene beginnt erneut mit der Initialisierung der `neighborsTriangleSum`. Ihr Wert entspricht der Summe aus vorab verworfenen Dreiecken, gespeichert in der `currentTriangleID` und der Anzahl zu generierender Dreiecke an der Startposition. Danach folgt die

```
1 neighborsTriangleSum = currentTriangleSum + neighbors.s0;
2 int8 cubeOffsetIndex = {1,1,1,1,1,1,1,1};
3
4 cubeOffsetIndex.s0   = neighborsTriangleSum <= triangleID
5                       ? 1 : 0;
6 neighborsTriangleSum += neighbors.s1;
7 cubeOffsetIndex.s1   = neighborsTriangleSum <= triangleID
8                       ? 1 : 0;
9 ...
```

Listing 3: Verarbeitung der `neighborsTriangleSum` beim Scan einer Ebene der Histo–Pyramide nach [Smi].

```
1 currentTriangleSum = currentTriangleSum +
2                      cubeOffsetIndex.s0 * neighbors.s0 +
3                      cubeOffsetIndex.s1 * neighbors.s1 +
4                      cubeOffsetIndex.s2 * neighbors.s2 +
5                      cubeOffsetIndex.s3 * neighbors.s3 +
6                      cubeOffsetIndex.s4 * neighbors.s4 +
7                      cubeOffsetIndex.s5 * neighbors.s5 +
8                      cubeOffsetIndex.s6 * neighbors.s6 +
9                      cubeOffsetIndex.s7 * neighbors.s7;
```

Listing 4: Update der `currentTriangleSum` für den Scan der nächstgrößeren Ebene der Histo–Pyramide nach [Smi]. Für die kleinste Ebene der Histo–Pyramide wird die `currentTriangleSum` mit Null initialisiert.

Verarbeitung der Ebene exakt der oben beschriebenen Implementierung. Der `currentOffsetIndex` wird am Ende der Berechnungen gegebenenfalls erhöht und erneut an die nachfolgende Ebene weitergereicht. Auf diese Weise werden alle Ebenen der Histo–Pyramide nacheinander abgearbeitet.

Der Scan der letzten Ebene der Histo–Pyramide gibt die Koordinaten der Volumenzelle zurück, in der das Dreieck generiert werden soll. Anhand des Index–Arrays aus dem `classifyCubesKernel` und der zugehörigen Wertetabelle wird noch einmal bestimmt, wie viele Dreiecke insgesamt in der Volumenzelle vorliegen. Diese Zahl wird benötigt, um die genaue Marching Cubes Konfiguration zu identifizieren. Aus einer zweiten Wertetabelle können dann in einer Schleife diejenigen Kanten der Volumenzelle ausgelesen werden, auf denen die Vertices des zu generierenden Dreiecks liegen müssen. Eine solche Kante wird von jeweils zwei Voxeln $p0$ und $p1$ der Volumenzelle aufgespannt. Durch Interpolation zwischen den Voxeln wird der Vertex auf der Kante bestimmt. Der zur Interpolation verwendete Faktor f wird folgendermaßen berechnet:

$$f = \frac{1 - p0}{p1 - p0} \tag{11}$$

Für die Interpolation der Vertexnormalen anhand von Vorwärtsdifferenzen wird derselbe Faktor verwendet. Die drei Vertices und Normalen, die der `calculateMCVerticesKernel` pro Work–Item berechnet, werden an die korrekte Stelle in jeweils ein *Vertex Buffer Object* (VBO) geschrieben. Haben alle Work–Items die Verarbeitung beendet, enthalten die VBOs alle Vertices und Vertexnormalen des Marching Cubes Polygonnetzes.

Auf der CPU werden beide VBOs in einem *Vertex Array Object* (VAO) gekapselt und an die Klasse `Renderer` übergeben. Die Berechnung des Polygonnetzes wird im hybriden Renderingssystem nur einmal als Vorverarbeitungsschritt durchgeführt.

5.3 Shader–Programme

In der *Main–Loop* des hybriden Renderingsystems werden nacheinander die Marching Cubes Oberfläche und die direkte Volumendarstellung durch Raycasting in jeweils ein *Framebuffer Object* (FBO) gezeichnet. Anschließend werden beide Einzelergebnisse durch hybrides Rendering zusammengeführt. Jeder dieser drei Renderschritte ist in einem separaten Shader–Programm realisiert. Für das Gesamtsystem ergibt sich dadurch ein modularer Aufbau, in dem sowohl die direkte als auch die indirekte Visualisierungsmethode mit geringem Aufwand ausgetauscht werden kann.

5.3.1 Marching Cubes

Das Marching Cubes Shader–Programm besteht aus einem Vertex–Shader, einer Tesselierungs–Einheit und einem Fragment–Shader. Der Vertex–Shader reicht den aktuellen Vertex unverändert und die Normale lediglich normalisiert an den Tessllation Control Shader weiter. Der Tessellation Control Shader bestimmt ein dreieckiges Patch zum aktuellen Vertex und gibt dieses zusammen mit den entsprechenden Vertexnormalen an den Tessellation Evaluation Shader weiter. Zusätzlich wird festgelegt, wie stark das Patch durch den Tessellation Evaluation Shader unterteilt werden soll.

Die Verfeinerung der Geometrie im Tessellation Evaluation Shader nutzt *PN–Triangles*, um eine glattere Oberfläche zu generieren. Dazu wird ein kubisches Patch aus den gegebenen Vertices b_{300}, b_{030} und b_{003}, den Tangenten Koeffizienten b_{210}, b_{120}, b_{021}, b_{012}, b_{102} und b_{201} sowie dem Zentrumskoeffizienten b_{111} definiert. Diese Koeffizienten, auch Kontrollpunkte genannt, bilden zusammen ein Kontrollnetz (s. Abbildung 40). Die Berechnung des Kontrollnetzes folgt einem festen Formalismus, wie er in Listing 5 implementiert ist. Sind alle Kontrollpunkte berechnet, werden sie anhand der Tesselierungskoordinaten gewichtet. Die Ergebnisse werden akkumuliert, mit der Model–View–Projection–Matrix transformiert und als neuer Vertex

```
#define U gl_TessCoord.y
#define V gl_TessCoord.z
#define W gl_TessCoord.x

...

vec3 b210 = (2 * b300 + b030 - w(1,2) * n200);
vec3 b120 = (2 * b030 + b300 - w(2,1) * n020);
vec3 b021 = (2 * b030 + b003 - w(2,3) * n020);
vec3 b012 = (2 * b003 + b030 - w(3,2) * n002);
vec3 b102 = (2 * b003 + b300 - w(3,1) * n002);
vec3 b201 = (2 * b300 + b003 - w(1,3) * n200);

vec3 e = (b210 + b120 + b021 + b012 + b102 + b201) / 18.0f;
vec3 v = (b300 + b030 + b003) / 3.0f;

vec3 b111 = e + (e - v) / 2.0f;

vec3 vertex = b300 * pow(W,3.0)+ b030 * pow(U,3.0)
            + b003 * pow(V,3.0)+ b210 * pow(W,2.0) * U
            + b120 * W * pow(U,2.0) + b201 * pow(W,2.0) * V
            + b021 * pow(U,2.0) * V + b102 * W * pow(V,2.0)
            + b012 * U * pow(V,2.0)
            + b111 * 6.0 * W * U * V;

mat4 mvMatrix   = viewMatrix * modelMatrix;
mat4 mvpMatrix  = projectionMatrix * mvMatrix;

f_vertex    = vec3(mvMatrix * vec4(vertex, 1.0f));
gl_Position = mvpMatrix * vec4(vertex, 1.0f));
```

Listing 5: Berechnung zusätzlicher Vertices mit PN–Triangles (für die Definition der Hilfsfunktion $w(i,j)$ s. Listing 7).

```
vec3 n110 = normalize(n200 + n020 - v(1,2) * (b030 - b300));
vec3 n011 = normalize(n020 + n002 - v(2,3) * (b003 - b030));
vec3 n101 = normalize(n002 + n200 - v(3,1) * (b300 - b003));

vec3 normal = n200 * pow(W,2.0) + n020 * pow(U,2.0) +
              n002 * pow(V,2.0) + n110 * W * U +
              n011 * U * V + n101 * W * V;

f_normal = normalize( normalMatrix * normal );
```

Listing 6: Berechnung der Vertexnormalen mit PN–Triangles (für die Definition der Hilfsfunktion $v(i,j)$ s. Listing 7).

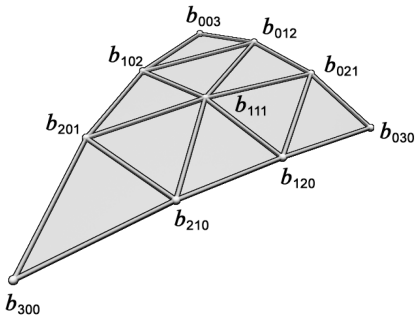


Abbildung 40: Koeffizienten im PN–Triangle–Kontrollnetz (nach Vlachos et al. [VPBM01]).

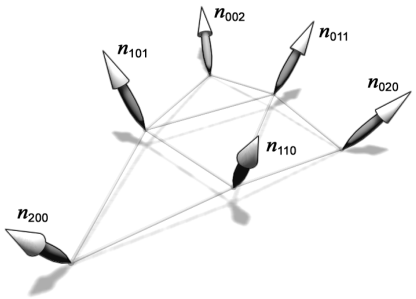


Abbildung 41: Normalen im PN–Triangle–Kontrollnetz (nach Vlachos et al. [VPBM01]).

an den Fragment–Shader übergeben. Zur weiteren Verarbeitung im hybriden Renderer wird die z–Koordinate des Vertex zusätzlich separat weitergereicht.

Neben den Vertices werden im Tessellation Evaluation Shader Normalen im Kontrollnetz definiert. In Abbildung 41 sind die aus dem Polygonnetz bekannten Basisnormalen n_{200}, n_{020} und n_{002} sowie die Normalenkoeffizienten n_{110}, n_{101} und n_{011} dargestellt. Die Normalen werden entsprechend der in Listing 6 umgesetzten Formeln berechnet, anhand der Tesselierungdkoordinaten gewichtet und zu einer Vertexnormalen akkumuliert. Nach Transformation mit der Normal–Matrix und anschließender Normalisierung wird sie ebenfalls

```
1  in vec3 e_vertex[];
2  in vec3 e_normal[];
3
4  out vec3 f_vertex;
5  out vec3 f_normal;
6
7  ...
8
9  float w( int i, int j ) {
10     return dot(e_vertex[j-1] - e_vertex[i-1],
11                e_normal[i-1]);
12 }
13
14 float v( int i, int j ) {
15     vec3 a = e_vertex[j-1] - e_vertex[i-1];
16     vec3 b = e_normal[j-1] + e_normal[i-1];
17     return 2.0f * (dot(a,b) / dot(a,a));
18 }
```

Listing 7: Hilfsfunktionen zur Berechnung der Vertices und der Vertexnormalen mit PN–Triangles.

an den Fragment–Shader übergeben.

Im Fragment–Shader wird zur Berechnung des Farbwerts C eines Vertex das Phong'sche Beleuchtungsmodell verwendet. Dazu wird ein ambienter und ein diffuser Term verwendet; ein spekularer Term wird nicht in das Gesamtergebnis integriert.

$$C = M \cdot L_a + M \cdot L_d \cdot \cos\phi \tag{12}$$

In Formel 12 bezeichnet $\cos\phi$ das Skalarprodukt aus dem normalisierten und negierten Lichtvektor und der Vertexnormalen. Im Falle eines negativen Ergebnisses wird auf Null gerundet.

Der Ergebnis des Fragment–Shaders wird als Vektor in einem FBO gespeichert; er besteht aus einer RGB–Farbkomponente und der aus dem Tesselation Evaluation Shader weitergereichten Tiefeninformation.

5.3.2 Raycasting

Für das Raycasting werden auf der CPU in der Methode `initRaycasting()` der Klasse `Renderer` einmalig eine eindimensionale Transferfunktion und eine zweidimensionale Offset–Textur vorberechnet. Die Transferfunktion wird als Wertetabelle mit $256 \cdot 4$ Nullen initialisiert. Sie kann folglich Intensitätswerten im Bereich $[0, 255]$ jeweils eine RGBA–Farbe zuordnen. Im hybriden Renderingsystem werden dazu insgesamt sechs Kontrollpunkte definiert. Jeder Kontrollpunkt ist als Struktur (*struct*) aus einer RGBA–Farbe und einem Intensitätswert zusammengesetzt. Lineare Interpolation liefert Farbübergänge, die zwischen den Kontrollpunkten liegen. Die auf diese Weise bestimmten Farb– und Opazitätswerte werden in einer 1D–Textur gespeichert.

Die Offset–Textur, die der Raycaster für Stochastic Jittering nutzt, wird in der Größe 32×32 angelegt und mit zufälligen Werten im Intervall $[0, 255]$ gefüllt. Sie wird, ebenso wie die 1D–Transferfunktion, an das Raycasting–Shader–Programm übergeben. Dabei ist zu beachten, dass alle in den Texturen enthaltenen Werte auf der GPU automatisch in den Bereich $[0, 1]$ eingepasst werden. Dies gilt auch für die volumetrischen CT–Daten, die in Form einer dreidimensionalen Textur im Grafikspeicher vorliegen.

In einem letzten Initialisierungsschritt wird die Hüllgeometrie, die den volumetrischen Datensatz umschließt, auf der CPU bestimmt. Dazu wird ein Quader in der Größe des volumetrischen Datensatzes definiert. Zur Speicherung der Geometrie wird eine indizierte Vertexliste verwendet. Zusätzlich werden den Ecken des Quaders Texturkoordinaten zugewiesen.

Das Shader–Programm besteht aus einem Vertex und einem FragmentShader. Der Vertex Shader reicht lediglich die unveränderten Texturkoordinaten der Hüllgeometrie sowie die transformierten Vertices weiter. Im Fragment Shader findet das eigentliche Raycasting pro Fragment statt.

Als Startposition für die Strahlenverfolgung wird die Texturkoordinate des aktuellen Fragments verwendet. Zur Optimierung der Bildqualität wird

```
vec2 offsetCoord = vec2(gl_FragCoord.xy / offsetSize.xy);
position += direction * stepSize * texture(offset,
                                           offsetCoord).x;
```

Listing 8: Stochastic Jittering zur Optimierung der Bildqualität.

```
    float lastValue    = 0.0f;
    float currentValue = 0.0f;

// Ray traversal
    for(int i = 0; i < numberOfSamples; i++)
    {
   // Transfer intensity value at current position to color
      voxel = texture(volume, position).x;
      color = texture(transferFunction, voxel * tfFactor);

   // Check for an intersection with the vessel
      currentValue = voxel - isovalue;
      if(currentValue * lastValue < 0.0f ||
            abs(currentValue) <= 0.00001f)
      {
         color.a = 1.0f
         color   = shade( ... );

         if(first == true)
         {
             depth = projectionMatrix * viewMatrix *
                     vec4(position * volumeMax, 1.0f);
             first = false;
         }
      }

      lastValue = currentValue;
      ...
    }
```

Listing 9: Strahlenverfolgung.

Stochastic Jittering eingesetzt und die Startposition ein Stück weit entlang des Strahls verschoben (s. Listing 8). Anschließend wird der Strahl in einer Schleife traversiert (s. Listing 9).

Für das Voxel an der aktuellen Abtastposition wird der Intensitätswert aus der CT–Textur gelesen. Die zum Intensitätswert korrespondierende Position innerhalb der Transferfunktion wird bestimmt; die dort hinterlegten Farb– und Opazitätswerte werden ebenfalls ausgelesen. Um ein an relevanten

Stellen opakes Ergebnisbild zu generieren, dass der Marching Cubes Oberfläche optisch ähnlich ist, wird anhand des Intensitätswerts überprüft, ob der Strahl zwischen der aktuellen und der letzten Abtastposition zum Gefäß gehörende Voxel durchquert hat. Ist dies der Fall, wird für die aktuelle Abtastposition der Opazitätswert auf 1.0 gesetzt und eine Beleuchtung berechnet. Die für letzteres notwendigen Normalen werden gradientenbasiert bestimmt. Die Beleuchtung setzt sich, wie beim Marching Cubes, aus einem ambienten und einem diffusen Term zusammen.

Wird entlang eines Strahls das erste Mal ein Gefäßvoxel durchquert, speichert der Algorithmus den Tiefenwert der entsprechenden Abtastposition. Er wird im hybriden Renderer für den Vergleich zwischen Marching Cubes Oberfläche und direkter Volumenvisualisierung verwendet.

```
1   dst.xyz = (1 - dst.a) * color.xyz * color.a  + dst.xyz;
2   dst.a   = mix(color.a , 1.0f, dst.a);
```

Listing 10: Komposition der Farb– und Opazitätswerte entlang des Strahls anhand einer Transparenzformel.

```
1  // Early-ray-termination
2     if (dst.a > 0.95f) break;
3
4  // Check if outside volume
5     vec3 temp1 = sign(position - volumeMin);
6     vec3 temp2 = sign(vec3(1.0f, 1.0f, 1.0f) - position);
7     float inside = dot(temp1, temp2);
8     if(inside < 3.0) break;
```

Listing 11: Abbruchkriterium für die Strahlenverfolgung.

Die Bearbeitung einer Abtastposition endet mit der Komposition aller bisher ermittelten Farb– und Opazitätswerte (s. Listing 10). Anschließend wird die nächste Abtastposition bestimmt. Liegt diese außerhalb der Hüllgeometrie oder wurde bereits eine Gesamtopazität von mindestens 0.95 erreicht, wird die Verfolgung des Strahls beendet (s. Listing 11).

Die direkte Volumenvisualisierung wird zusammen mit den pro Strahl ermittelten Tiefenwerten für Gefäßvoxel in einem FBO gespeichert.

5.3.3 Hybrides Rendering

Die Kombination der Marching Cubes Oberfläche und der direkten Volumenvisualisierung wird in einem dritten Renderpass in ein bildschirmfüllendes Rechteck gezeichnet. Die dafür notwendige Geometrie wird auf der CPU in der Methode `initHybridRendering()` der Klasse `Renderer` erzeugt. Den

vier Vertices werden jeweils Texturkoordinaten zugeordnet, mit denen die FBO–Texturen auf das bildschirmfüllende Rechteck abgebildet werden können. Ein VAO kapselt die Vertex– und Texturkoordinaten und wird an das Shader–Programm gebunden.

Der Vertex Shader reicht sowohl die Texturkoordinate als auch den Vertex unverändert an den Fragment Shader weiter. Dort werden die direkte und die indirekte Visualisierung, die beide als Textur vorliegen, gemeinsam gerendert. Die Komposition der Farbwerte basiert auf den Tiefeninformationen aus dem ersten und zweiten Renderpass. In Listing 12 ist das Blending implementiert.

```
1  float mcDepth  = texture(marchingCubes, f_texCoord).a;
2  float rayDepth = texture(raycasting, f_texCoord).a;
3
4  vec4 mcColor  = vec4((texture(marchingCubes,
5                                f_texCoord)).xyz, 1.0);
6  vec4 rayColor = vec4((texture(raycasting,
7                                f_texCoord)).xyz, 1.0);
8
9  float diff = abs(mcDepth - rayDepth);
10
11 color = (diff < mixValue && rayColor.z < 0.6)   ?
12                0.7 * mcColor + 0.3 * rayColor  :
13                rayColor;
```

Listing 12: Kombination der direkten und indirekten Visualisierung bei ausreichender Genauigkeit der Marching Cubes Oberfläche ohne Verdeckung von Kalzifizierungen.

Die Marching Cubes Oberfläche wird zu 70 % in das Raycasting Ergebnis integriert, wenn die Differenz ihrer Tiefenwerte einen festgelegten Schwellwert, den `mixValue`, nicht überschreitet. Unabhängig davon wird das Blending verhindert, wenn die Blau–Komponente der direkten Volumenvisualisierung mindestens 0.6 beträgt. In diesem Fall liegt am betroffenen Pixel kein roter Farbton und damit kein Gefäß vor. Der Zusatz ermöglicht eine zuverlässige Darstellung von Kalzifizierungen. Findet kein Blending statt, wird ausschließlich das Raycasting–Ergebnis abgebildet.

5.4 Benutzerschnittstelle

Über die Tastatur kann der Benutzer einige Parameter des hybriden Renderingsystems interaktiv verändern. Der Isowert, der im Raycaster zur Detektion von Gefäßvoxeln verwendet wird, kann im Bereich [40,160] eingestellt werden. Ebenso lässt sich die bei der Komposition maximal erlaubte Tiefendifferenz zwischen Marching Cubes Oberfläche und direkter Volumenvisua-

lisierung variieren. Das Intervall ist auf [0,20] beschränkt.

Zu Vergleichszwecken ist es möglich zwischen einer reinen Marching Cubes oder Raycasting Darstellung und der hybriden Visualisierung zu wechseln. Außerdem kann der Marching Cubes Anteil in der hybriden Visualisierung violett eingefärbt und damit hervorgehoben werden.

Die Tesselierung der Marching Cubes Oberfläche ist standardmäßig nicht aktiviert. Allerdings hat der Benutzer die Möglichkeit, die Tesselierung mit PN–Triangles dauerhaft einzuschalten oder sie ausschließlich dann zu unterdrücken, wenn die Kamera bewegt wird. Die Möglichkeiten zur Kameranavigation sind in Abbildung 42 dargestellt.

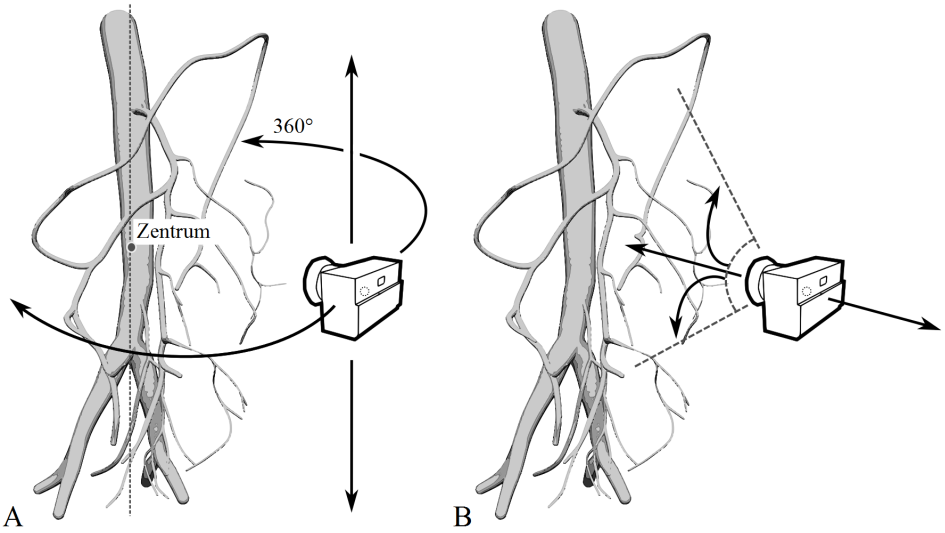


Abbildung 42: Navigationsmöglichkeiten im hybriden Renderingsystem. **A:** Die Kamera kann in der x–z–Ebene rund um die Visualisierung bewegt werden. Außerdem sind Auf- und Abbiegungen parallel zur y-Achse möglich. Das Blickzentrum liegt dabei immer auf der senkrechten Mittelachse durch die Hüllgeometrie des CT–Datensatzes. **B:** Die Kamera kann sowohl vorwärts– als auch rückwärts entlang ihrer Blickrichtung bewegt werden. In einem eingeschränkten Winkel lässt sie sich nach vorne oder nach hinten kippen.

Das Blickzentrum der Kamera ist so eingestellt, dass es immer auf einer senkrechten Achse durch den Mittelpunkt der Hüllgeometrie liegt. Um diese Achse kann die Kamera rotiert werden. Zusätzlich sind Auf– und Abbewegungen entlang der Achse oder Vorwärts– und Rückwärtsbewegungen entlang der Blickrichtung möglich. Die Kamera kann in einem beschränkten Neigungswinkel nach vorne oder hinten gekippt werden. Für die Bewegungsgeschwindigkeit lässt sich zwischen drei festgelegten Stufen wählen.

Eine Zusammenfassung aller Tastaturbefehle zur Anpassung von Parametern und zur Navigation ist in Tabelle 11 beziehungsweise 12 in Anhang B beigefügt.

6 Ergebnisse

6.1 Evaluation der Performanz

Im Rahmen dieser Arbeit sollte ein hybrides Renderingsystem entwickelt werden, das **interaktives Arbeiten** auf dreidimensionalen, CT-basierten Gefäßvisualisierungen sowie eine **echtzeifähige Navigatio**n innerhalb des virtuellen Raums ermöglicht. Eine Anwendung gilt als echtzeitfähig, solange sie durchgehend mit mindestens 15 FPS (*frames per second*) rendert. Interaktionen sind bereits mit 6 FPS möglich. Ab 72 FPS ist der Mensch nicht mehr in der Lage, unterschiedliche Frameraten voneinander zu differenzieren. Um ein flüssiges Bild und gleichzeitig optimale Reaktionszeiten zu erzielen, sollten 60 FPS nicht unterschritten werden [AMHH08].

Die Performanz des hybriden Renders wurde anhand dreier Testsysteme gemessen. Sie sind in Tabelle 6 näher spezifiziert. Die Messungen wurden für zwei RAW–Datensätze vorgenommen, die jeweils in halber und in voller Auf-

	MacBook Air (Mid 2012)	MacBook Air (Mid 2013)	Windows-PC
Betriebs-system	OS X 10.9.4 (Mavericks)	OS X 10.9.5 (Mavericks)	Microsoft Windows 8.1 Pro
Prozessor	Intel Core i5 1,8 (2,8) GHz	Intel Core i5 1,3 (2,6) GHz	Intel Xeon E3-1230v3 3,3 (3,7) GHz
Grafik-karte	Intel HD Graphics 4000 (1024 MB)	Intel HD Graphics 5000 (1536 MB)	AMD Sapphire Radeon R9-290 Vapor-X OC (4096 MB)
RAM	4 GB DDR3-1600	4 GB DDR3-1600	16 GB DDR3-1600
Festplatte	SSD	SSD	HDD

Tabelle 6: Spezifikation der Testsysteme. Aufgeführt werden die zur Evaluation der Performanz relevanten Komponenten Betriebssystem, Prozessor (inkl. der Grundtaktgeschwindigkeit und der maximalen Taktgeschwindigkeit), Grafikkarte (inkl. des verfügbaren Grafikkartenspeichers), RAM und Festplattentyp.

	Datensatz 1				Datensatz 2			
Auflösung	$256 \times 256 \times 389$		$512 \times 512 \times 389$		$256 \times 256 \times 533$		$512 \times 512 \times 533$	
Voxelabstände	1.07031	×	0.535156	×	0.957031	×	0.478516	×
	1.07031	×	0.535156	×	0.957031	×	0.478516	×
	1.0		1.0		0.700012		0.700012	

Tabelle 7: Spezifikation der Testdatensätze.

	$256 \times 256 \times 389$ ($1.07031 \times 1.07031 \times 1.0$)			$512 \times 512 \times 389$ ($0.535156 \times 0.535156 \times 1.0$)		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
Zeit (s)	2,82	0,86	4,33	2,76	0,89	4,33
Größe	$512 \times 512 \times 512$			$512 \times 512 \times 512$		
Dreiecke	139550			299186		

Tabelle 8: Performanzergebnisse für Datensatz 1 zur Berechnung der Marching Cubes Oberfläche. Gemessen wurde die Vorverarbeitungszeit in Sekunden, die Größe des Segmentierungsdatensatzes (nach Anpassung an die Histo–Pyramiden–Traversierung) und die Anzahl generierter Dreiecke.

	$256 \times 256 \times 533$ ($0.957031 \times 0.957031 \times 0.700012$)			$512 \times 512 \times 533$ ($0.478516 \times 0.478516 \times 0.700012$)		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
Zeit (s)	2,79	0,90	10,99 (4,35)	3,28	0,99	10,99 (4,35)
Größe	$512 \times 512 \times 512$		$1024 \times 1024 \times 1024$	$512 \times 512 \times 512$		$1024 \times 1024 \times 1024$
Dreiecke	296144		302520	639806		653370

Tabelle 9: Performanzergebnisse für Datensatz 2 zur Berechnung der Marching Cubes Oberfläche. Gemessen wurde die Vorverarbeitungszeit in Sekunden, die Größe des Segmentierungsdatensatzes (nach Anpassung an die Histo–Pyramiden–Traversierung) und die Anzahl generierter Dreiecke. Für die gleichzeitige Verarbeitung aller 533 Schichtbilder (eingepasst in ein Volumen von $1024 \times 1024 \times 1024$) erwies sich der Grafikspeicher der MacBooks als nicht ausreichend. Deshalb wurde in beiden Fällen lediglich auf den ersten 512 Schichtbildern und damit auf einem angepassten Volumen von $512 \times 512 \times 512$ gearbeitet. Für den Windows–PC ist die Berechnungszeit für 512 Schichtbilder in Klammern angegeben.

lösung vorliegen (s. Tabelle 7). Bei den Daten handelt es sich um CT–Bilder des Abdomens jeweils inklusive einer zugehörigen Gefäßsegmentierung.

Die Generierung der Marching Cubes Oberfläche benötigt auf dem MacBook Air (Mid 2012) über alle Datensätze hinweg minimal 2,76 s und maximal 3,28 s. Das Nachfolgemodell erreicht mit Berechnungszeiten von 0,86 s bis 0,99 s die besten Ergebnisse. Unter Windows wurden Werte von 4,33

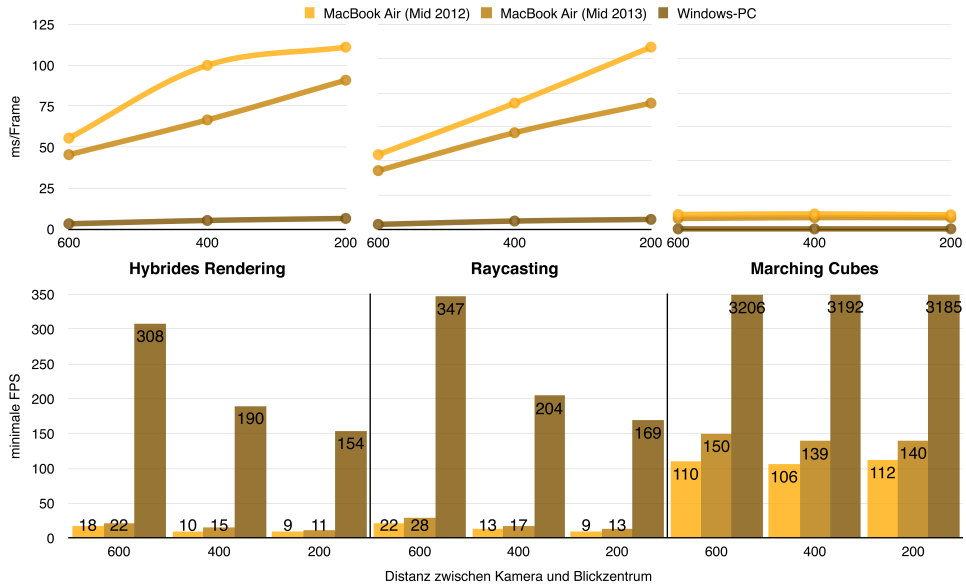


Abbildung 43: Performanzergebnisse für das Rendering von Datensatz 1 (Auflösung: $256 \times 256 \times 389$) auf allen drei Testsystemen. Die Framezeit (oben) sowie die Framerate (unten) des hybriden Renderingsystems (links), werden den Ergebnissen eines reinen Raycasting–Algorithmus (mitte) sowie einer Marching Cubes Implementierung (rechts) gegenübergestellt.

s bis 10,99 s gemessen. Dabei ist zu beachten, dass die Maximalzeit unter Windows bei der Verarbeitung des zweiten Datensatzes alle 533 Schichtbilder einbezieht. Da für den Aufbau der Histo–Pyramiden Auflösungen der Form $2^x \times 2^x \times 2^x$ benötigt werden, folgt für Datensatz 2 das Verarbeitungsformat $1024 \times 1024 \times 1024$. Auf beiden MacBooks konnten aufgrund der begrenzten Grafikspeicherkapazität nur 512 Schichtbilder des zweiten Datensatzes verarbeitet werden. Somit bleibt auch das Verarbeitungsformat auf $512 \times 512 \times 512$ beschränkt. Wird Datensatz 2 auf dem Windows–PC mit lediglich 512 Schichtbildern verarbeitet, ist die Berechnung nach 4,35 s abgeschlossen. Die dennoch vergleichsweise langen Berechnungszeiten unter Windows entstehen aufgrund einer schlechteren OpenCL–Kompatibilität. So dauert beispielsweise die OpenCL–Initialisierung bei beiden MacBooks maximal 0,17 s; unter Windows werden mindestens 4 s benötigt.

Die wichtigsten Eckdaten zur Generierung der Marching Cubes Oberfläche sind in Tabelle 8 für Datensatz 1 und in Tabelle 9 für Datensatz 2 zusammengefasst. Auffällig ist, dass die Berechnungszeit mit steigender Anzahl generierter Dreiecke kaum bis gar nicht schwankt. Ein größerer Datensatz, der schlimmstenfalls allein aufgrund vieler Schichtbilder in ein sehr großes Format der Form $2^x \times 2^x \times 2^x$ übertragen werden muss, verlangsamt den Algorithmus dagegen enorm.

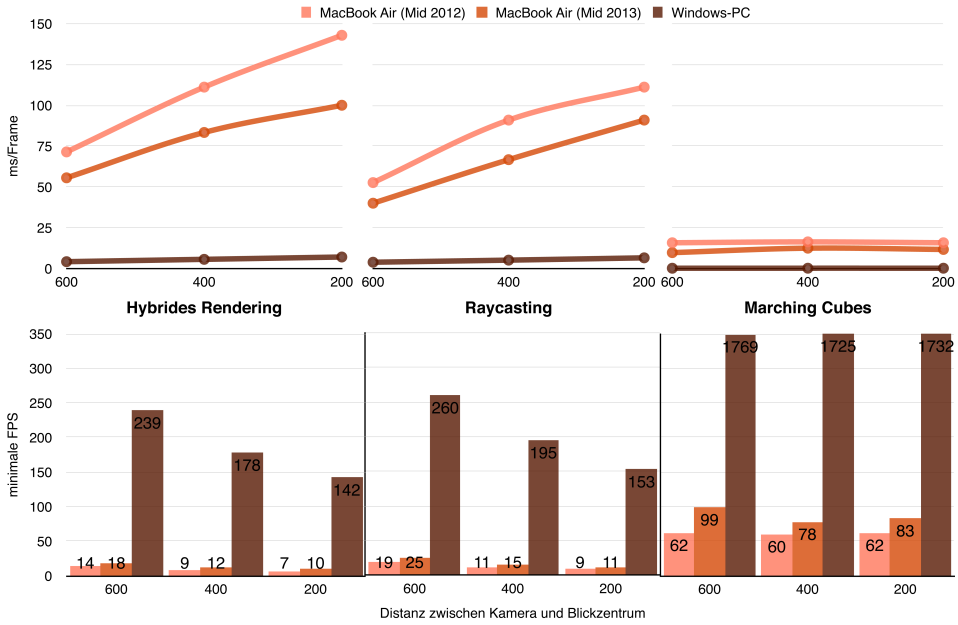


Abbildung 44: Performanzergebnisse für das Rendering von Datensatz 1 (Auflösung: 512×512×389) auf allen drei Testsystemen. Gemessen wurde die Framezeit in ms / Frame (oben) sowie die Framerate in FPS (unten). Die Werte, die durch hybrides Rendering erzielt werden (links), sind den Ergebnissen eines reinen Raycasting–Algorithmus (mitte) sowie einer Marching Cubes Implementierung (rechts) gegenübergestellt.

Die Performanz des Renderings ist in Abbildung 43 bis 46 grafisch dargestellt. Auf allen drei Testsystemen wurden die Framerate in FPS und die Framezeit in ms/Frame für den hybriden Renderer, eine Raycasting–Darstellung und die Visualisierung der Marching Cubes Oberfläche bei einer Fenstergröße von 1024 × 768 Pixeln gemessen. Die genauen Werte können Tabelle 13 bis 18 in Anhang C entnommen werden. Zwar steigt die Framezeit erwartungsgemäß mit wachsender Auflösung der Datensätze, dabei zeichnet sich allerdings ein gleichbleibendes Verhältnis zwischen den Messergebnissen des hybriden Renderings, des Raycastings und der Marching Cubes Visualisierung ab.

Die mit Abstand höchste Framerate erreicht die Visualisierung der Marching Cubes Oberfläche. Auf dem Windows–PC werden über alle Testdatensätze hinweg bis zu 3206 FPS, mindestens aber 849 FPS erreicht. Selbst das MacBook Air (Mid 2012) liefert echtzeitfähige Ergebnisse mit einem Minimum von 30 FPS. Das Nachfolgemodell erzielt je nach Testdatensatz bereits 10 bis 40 FPS mehr. Deutlich schlechter fallen Framerate und Framezeit für das Raycasting aus. Zufriedenstellende Ergebnisse liefert hier nur der Windows–PC mit mindestens 115 FPS und maximal 347 FPS. Beide Mac-

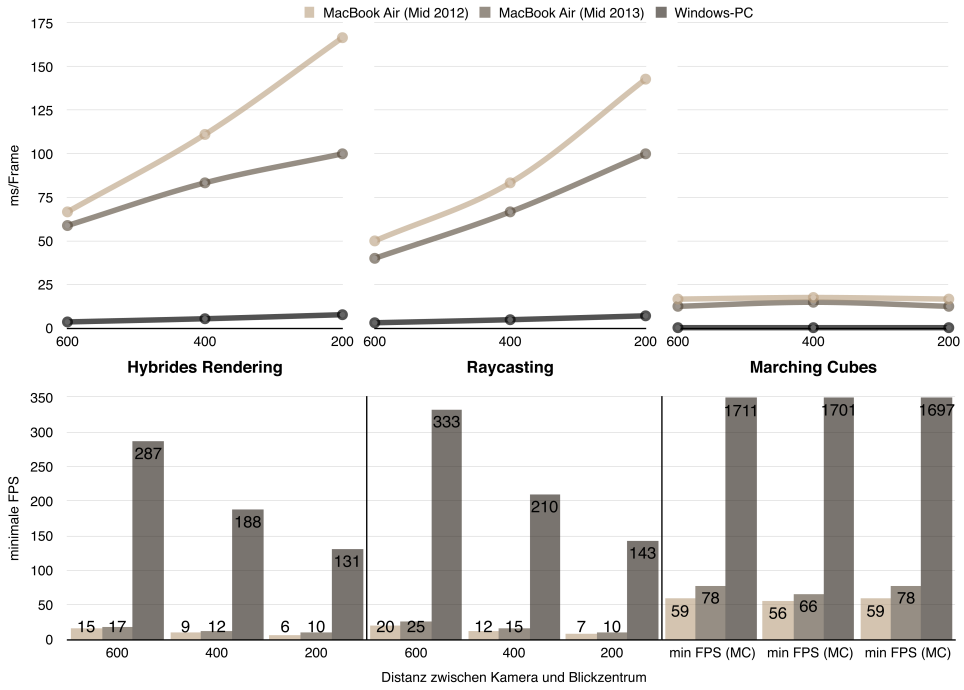


Abbildung 45: Performanzergebnisse für das Rendering von Datensatz 2 (Auflösung: $256 \times 256 \times 533$) auf allen drei Testsystemen. Die Framezeit (oben) sowie die Framerate (unten) des hybriden Renderingsystems (links), werden den Ergebnissen eines reinen Raycasting–Algorithmus (mitte) sowie einer Marching Cubes Implementierung (rechts) gegenübergestellt.

Books sind nicht in der Lage durchweg flüssige Ergebnisse zu erzielen. Die Framerate liegt hier bei 7 bis 22 FPS (Mid 2012) beziehungsweise bei 9 bis 28 FPS (Mid 2013). Die Ergebnisse des hybriden Renderings sind nur geringfügig schlechter als die Raycasting–Ergebnisse. Unter Windows wird mit 110 bis 308 FPS gerendert; unter OS X werden 5 bis 18 FPS (Mid 2012) und 7 bis 22 FPS (Mid 2013) erreicht. Im Vergleich zu den Raycasting–Ergebnissen bedeutet das für den Windows–PC ein Leistungsverlust von bis zu 11 %. Das MacBook Air (Mid 2012) verliert bis zu 22,2 % der Raycasting–Framerate. Mit 28,6 % büßt das MacBook Air (Mid 2013) die meiste Leistung ein.

Für jede Renderingtechnik wurde die Performanz bei Abständen zwischen Kamera und Blickzentrum von 600, 400 und 200 gemessen. Die Ergebnisse des hybriden Renderings und des Raycastings werden mit verringertem Abstand gleichermaßen schlechter. Lediglich das Marching Cubes Ergebnis zeigt in dieser Hinsicht keine nennenswerten Schwankungen.

Der Performanzverlust, der für einen Datensatz durch Verdopplung der Auflösung eintritt, ist in Abbildung 47A für Datensatz 1 sowie in Abbildung

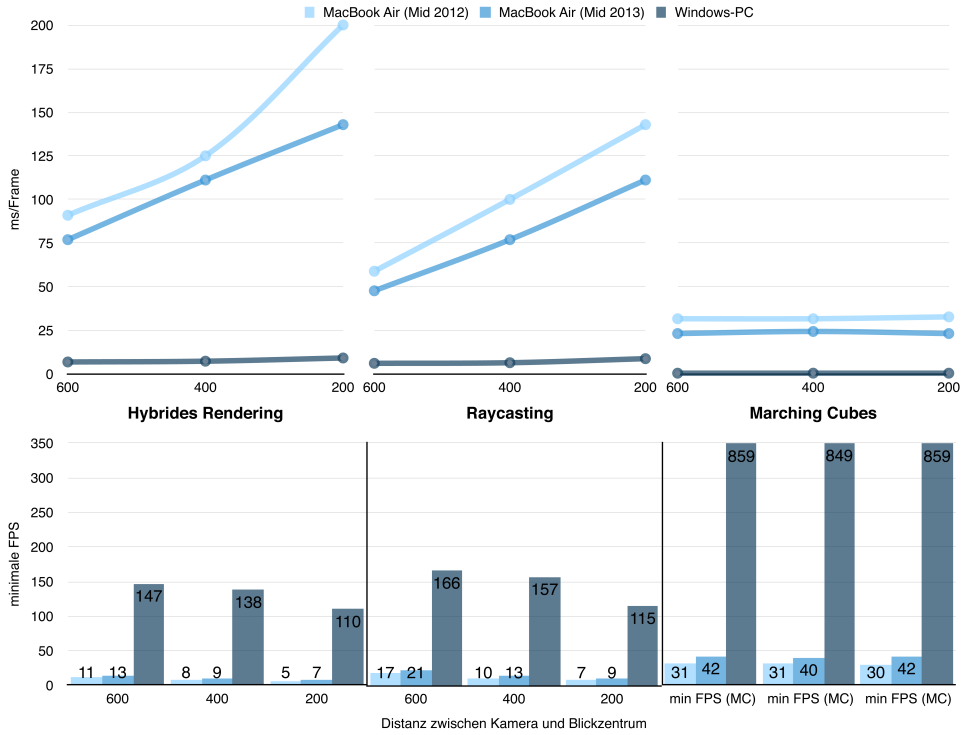


Abbildung 46: Performanzergebnisse für das Rendering von Datensatz 2 (Auflösung: $512 \times 512 \times 533$) auf allen drei Testsystemen. Die Framezeit (oben) sowie die Framerate (unten) des hybriden Renderingsystems (links), werden den Ergebnissen eines reinen Raycasting–Algorithmus (mitte) sowie einer Marching Cubes Implementierung (rechts) gegenübergestellt.

47B für Datensatz 2 ausschließlich für das hybride Rendering visualisiert. Die Grafiken zeigen jeweils an, wie viel Prozent der Framerate, die bei halber Auflösung erzielt wird, mit Verdopplung der Auflösung verbleibt. Dabei werden erneut unterschiedliche Distanzen zwischen Kamera und Blickzentrum betrachtet.

Datensatz 1 erzielt in voller Auflösung je nach gewählter Distanz und je nach Testsystem 77 bis 94 % der Framerate, die unter gleichen Bedingungen in halber Auflösung gemessen wurde. Für Datensatz 2 werden unter OS X 70 bis 89 %, unter Windows dagegen nur 51 bis 84 % erreicht. Der Unterschied könnte unter anderem daraus resultieren, dass für die Messung unter Windows die initiale Marching Cubes Berechnung nicht nur 512 sondern alle 533 Schichtbilder einbezieht; damit müssen in halber Auflösung etwa 6000, in voller Auflösung etwa 14000 Dreiecke mehr gerendert werden (s. Tabelle 9). Dies erklärt jedoch nicht, warum die verbleibende Framerate unter Windows ausschließlich bei einer Distanz von 600 zwischen Kamera und Blickzentrum derart stark von den Ergebnissen unter OS X abweicht. Weitere Messungen,

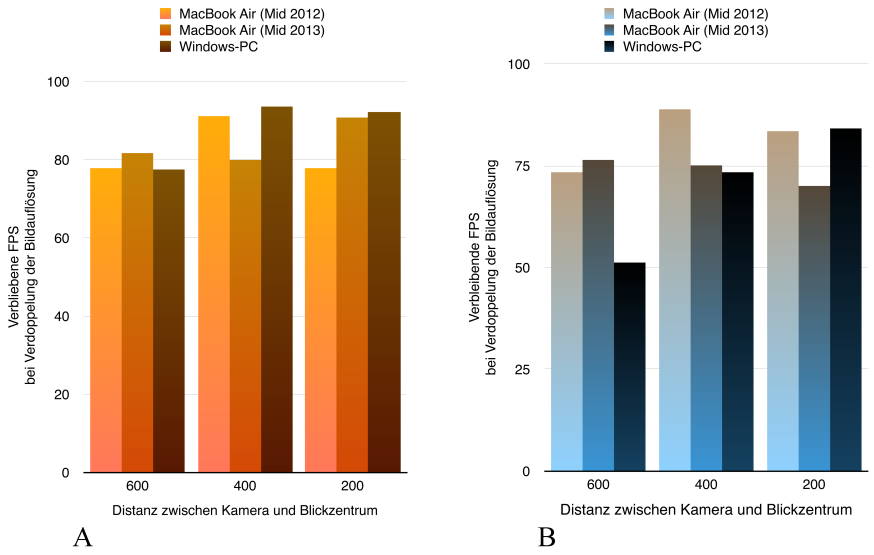


Abbildung 47: Verbleibende Framerate in Prozent bei Verdoppelung Datensatzauflösung. **A:** Verbleibende Framerate in Prozent gemessen für Datensatz 1. **B:** Verbleibende Framerate in Prozent gemessen für Datensatz 2. Entsprechend Tabelle 9 beziehen sich die Ergebnisse für beide MacBooks auf eine geringfügig kleinere Marching Cubes Oberfläche, die lediglich aus den ersten 512 Schichtbildern generiert wurde. Die Messungen unter Windows rendern das Polygonnetz, das aus allen 533 Schichtbildern gewonnen wird.

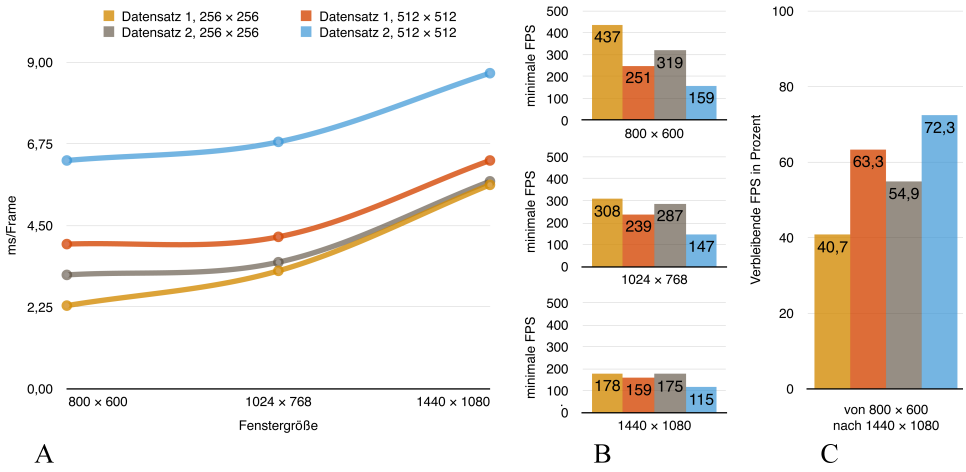


Abbildung 48: Performanzergebnisse für das hybride Rendering in Abhängigkeit zur Fenstergröße auf dem Windows–PC. **A:** Gemessene Framezeit für alle Datensätze. **B:** Gemessene Framerate für alle Datensätze. **C:** Verbleibende Framerate in Prozent mit wachsender Festergröße von 800×600 nach 1440×1080

gegebenenfalls auf weiteren, ähnlich rechenstarken Testsystemen, sind notwendig, um die Entstehung dieser Diskrepanz klären zu können.

Abschließend wurden Framerate und Framezeit zur Evaluation der Performanz des hybriden Renderingsystems in Abhängigkeit zur Fenstergröße betrachtet. Die Messungen wurden aufgrund der hohen Framerate und den folglich markanteren Unterschieden zwischen Einzelergebnissen lediglich auf dem Windows–PC durchgeführt. Die Distanz zwischen Kamera und Blickzentrum entspricht jeweils einem Wert von 600. Die genauen Messwerte sind in Tabelle 19 bis 21 in Anhang C beigefügt.

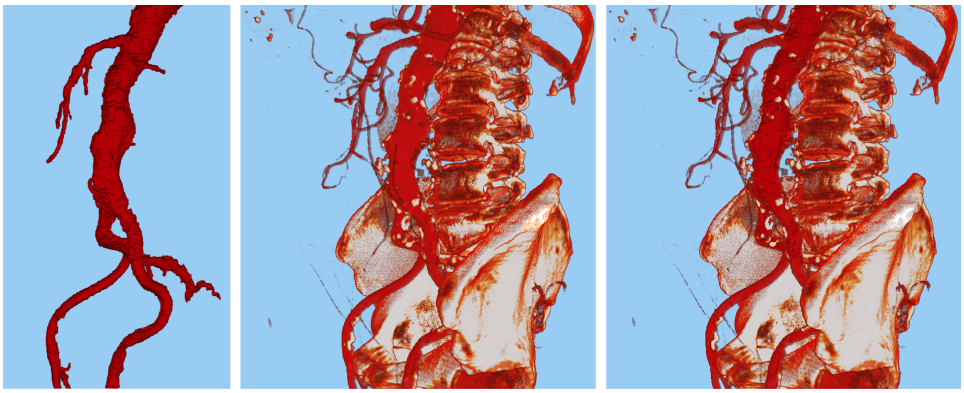
Die Framezeit wächst über alle Fenstergrößen für jeden Datensatz kontinuierlich. Die Steigung fällt dabei für die halb aufgelösten Datensätze höher aus, als für ihre voll aufgelösten Gegenstücke. Auf diese Weise nähern sich auch die Frameraten mit wachsender Fenstergröße immer weiter aneinander an. Die verbleibende Framerate in Prozent beträgt bei einer Vergrößerung des Fensters von 800×600 Pixeln auf 1440×1080 Pixeln für die halb aufgelösten Datensätze nur noch 40,7 % (Datensatz 1) beziehungsweise 54,9 % (Datensatz 2). Liegen die Daten dagegen hochaufgelöst vor, ist der Performanzverlust deutlich geringer. Datensatz 1 erreicht voll aufgelöst im größeren Fenster 63,3 % der ursprünglichen Framerate; für Datensatz 2 konnten sogar 72,3 % gemessen werden.

6.2 Evaluation der Bildqualität

Durch die Integration von Oberflächendaten in eine direkte Volumenvisualisierung sollte ein hybrides Renderingsystem entwickelt werden, dass in verschiedenen Bereichen des klinischen Alltags eingesetzt werden kann. Insbesondere für die Diagnoseunterstützung, aber auch bei der Therapieplanung ist eine exakte Genauigkeit der Darstellung unabdingbar.

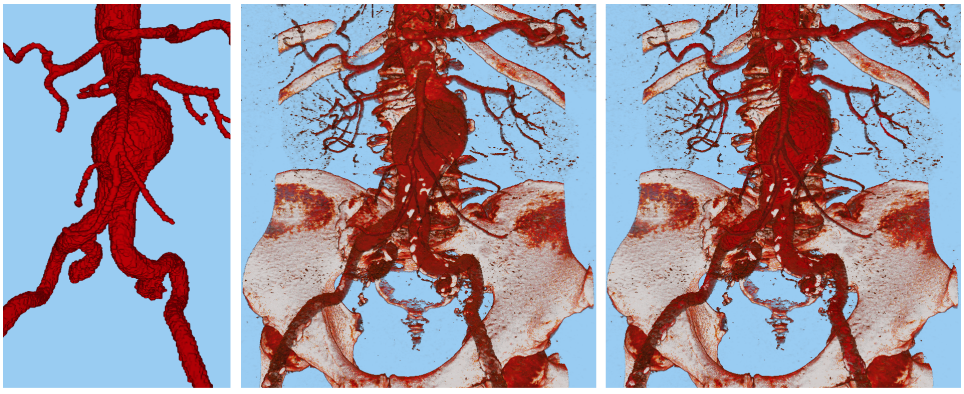
Die Genauigkeit und die visuelle Qualität des hybriden Renderingsystems wurden anhand von Datensatz 1 in voller Auflösung und Datensatz 2 in halber Auflösung evaluiert. Die dreidimensionalen Volumenvisualisierungen, die der Marching Cubes Algorithmus, Raycasting und das entwickelte hybride Renderingsystem aus diesen Datensätzen generieren, sind in Abbildung 49 und 50 dargestellt.

Aus Datensatz 1 konnte das Aneurysma aufgrund des schlecht kontrastierten Lumens nicht extrahiert werden. In Abbildung 49A erscheint die mit Marching Cubes dargestellte, eigentlich erkrankte Aorta deshalb klinisch unauffällig. Dagegen wird die Gefäßerweiterung im Raycasting–Ergebnis (s. Abbildung 49B) und im hybriden Rendering (s. Abbildung 49C) dadurch sichtbar, dass die direkte Visualisierung Kalzifizierungen entlang der Gefäßwand abbildet. Das mittig in beiden Bildern gelegene Oval aus weißen Flecken deutet damit den Umfang des Bauchaortenaneurysmas an.



A B C

Abbildung 49: Dreidimensionale Volumenvisualisierung berechnet auf Basis von Datensatz 1 (Auflösung: $512 \times 512 \times 389$). **A:** Visualisierung der aus Datensatz 1 segmentierten vaskulären Struktur durch Marching Cubes. **B:** Visualisierung des Gefäßes und der umliegenden Knochen anhand von Raycasting. **C:** Hybride Visualisierung des Gefäßes und der umliegenden Knochen.



A B C

Abbildung 50: Dreidimensionale Volumenvisualisierung berechnet auf Basis von Datensatz 2 (Auflösung: $256 \times 256 \times 533$). **A:** Visualisierung der aus Datensatz 1 segmentierten vaskulären Struktur durch Marching Cubes. **B:** Visualisierung des Gefäßes und der umliegenden Knochen anhand von Raycasting. **C:** Hybride Visualisierung des Gefäßes und der umliegenden Knochen.

Ein gut kontrastiertes Lumen wird durch Marching Cubes, Raycasting und hybrides Rendering anhand von Datensatz 2 visualisiert. Kalzifizierungen sind sowohl im Raycasting–Ergebnis (s. Abbildung 50B) als auch in der hybriden Darstellung (s. Abbildung 50C) deutlich zu erkennen; sie sind jedoch nicht im Segmentierungsdatensatz und folglich auch nicht in der Marching Cubes Oberfläche vorhanden (s. Abbildung 50D). Bei Datensatz 2

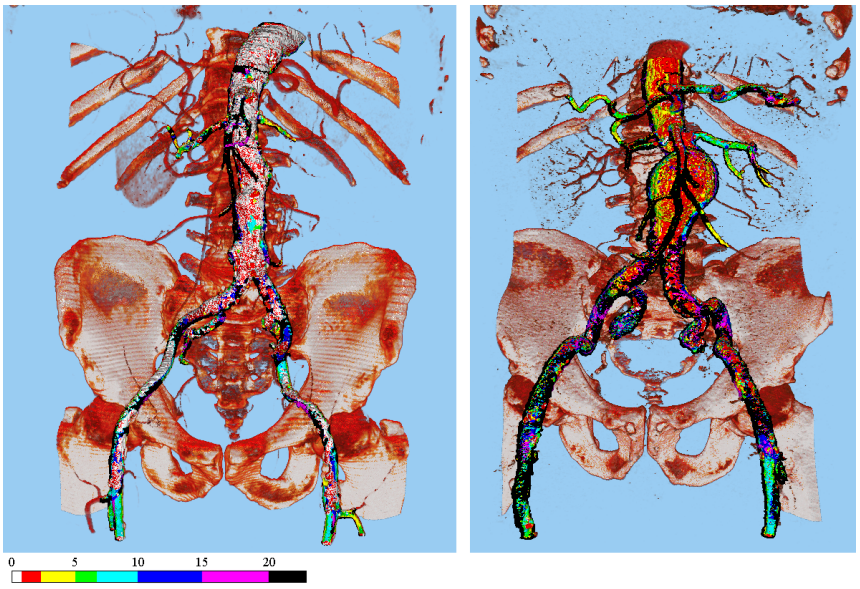


Abbildung 51: Farbcodierte Tiefendifferenz zwischen Marching Cubes Oberfläche und indirekter Volumendarstellung. **A:** Tiefendifferenz in Datensatz 1. Der hohe Weißanteil visualisiert die gute Qualität des Segmentierungsergebnisses. **B:** Tiefendifferenz in Datensatz 2. Das Segmentierungsergebnis ist im Vergleich zu Datensatz 1 deutlich ungenauer; dementsprechend größer ist der farbige Anteil im Bild. Schwarz eingefärbt sind hauptsächlich diejenigen Positionen, hinter denen das Raycasting kein Gefäß detektiert hat. Die Oberfläche wurde für Datensatz 2 aus 512 Schichtbildern berechnet, das Raycasting verarbeitet die gesamten 533 Schichtbilder.

gelang es allerdings nicht, auftretendes Rauschen im Raycasting–Ergebnis durch eine eindimensionale Transferfunktion vollständig zu filtern. Vereinzelnd werden aus diesem Grund auch im hybriden Bild rote Pixel – vor allem auf Höhe der Nieren und zwischen den Rippen – abgebildet.

Aufgrund der Nutzung einer maximal erlaubten Tiefendifferenz im Kompositionsschritt, besteht die Möglichkeit die **Genauigkeit** der hybriden Visualisierung bis zu einem gewissen Grad bewusst zu steuern. Um dies zu veranschaulichen wurden im Zuge der Evaluation die mittels Marching Cubes berechneten Tiefenwerte mit den Raycasting–Tiefen verglichen. Ausgehend von der Grundannahme, dass Raycasting sehr genaue Ergebnisse liefert, kann die Güte der hybriden Visualisierung anhand der Verteilung der Tiefendifferenzen über dem gesamten sichtbaren Gefäß geschätzt werden. Die Verteilung für Datensatz 1 und Datensatz 2 ist in Abbildung 51 farbcodiert dargestellt.

In Abbildung 51A zeigt der hohe Weißanteil eine relativ genau Oberflä-

	Datensatz 1 $512 \times 512 \times 389$			Datensatz 2 $256 \times 256 \times 533$		
	%	Ø	min	%	Ø	min
max $1,0$	38,8	0,2	$0,6^{-4}$	1,5	$0,8^{-2}$	$0,5^{-2}$
max $1,5$	50,3	0,3		3,2	$0,3^{-1}$	
max $5,0$	61,3	0,6		32,2	0,9	
max $10,0$	71,4	1,4		43,8	1,8	
max $20,0$	77,7	2,3		54,1	3,4	
unbeschränkt	98,8	12,1		94,3	22,6	

Tabelle 10: Abweichung der hybriden Gefäßvisualisierung vom Raycasting Ergebnis gemessen anhand der Differenz der jeweiligen Tiefenwerte. Für verschiedene maximal erlaubte Abweichungen wurde der prozentuale Anteil der Marching Cubes Oberfläche an der Gefäßvisualisierung (ohne Berücksichtigung der Kontextinformation) bestimmt. Zusätzlich sind für jeden Fall die durchschnittliche und die minimale Abweichung angegeben.

chenvisualisierung für Datensatz 1 an. Deutlich größer sind die Tiefendifferenzen zwischen Marching Cubes Oberfläche und Raycasting–Ergebnis für Datensatz 2 in Abbildung 51B. Die dicken schwarzen Konturen deuten zu den Seiten hin eine vorhandene Marching Cubes Oberfläche an, wo durch Raycasting kein Gefäß gefunden wurde. Das Segmentierungsergebnis war an diesen Stellen offensichtlich zu weit.

In beiden Darstellungen fällt auf, dass die Genauigkeit der Oberfläche entlang schmaler Strukturen deutlich abnimmt. Für viele dünne Gefäße liegt gar keine Segmentierung und somit auch keine Oberflächeninformation vor.

Die variable Einstellbarkeit der maximal erlaubten Tiefendifferenz ermöglicht es nun die ungenauen Abschnitte der Oberflächendarstellung zu ignorieren und lediglich den Anteil in das Endergebnis zu integerieren, der nach eigenen Maßstäben genau genug erscheint. Tabelle 10 zeigt, wie sich der durchschnittliche Bildfehler in der hybriden Visualisierung gemessen an der Raycasting–Darstellung für unterschiedliche maximale Tiefendifferenzen verhält. In beiden Datensätzen konnten Positionen lokalisiert werden, an denen Oberfläche und direkte Volumendarstellung nahezu identisch sind, sodass der minimale Bildfehler nahezu null beträgt. Für jede betrachtete maximal erlaubte Tiefendifferenz enthält die Tabelle zusätzlich den prozentuale Anteil der Oberfläche an der hybriden Gefäßdarstellung.

Wird die Tiefendifferenz beispielsweise mit einem Wert von 5,0 nach oben beschränkt, werden immer noch 61,3 % des Gefäßes in der Visualisierung von Datensatz 1 durch eine Oberfläche repräsentiert. Über alle Positionen, an denen die Oberfläche dargestellt wird, ergibt sich eine durchschnittliche Tiefendifferenz von gerade einmal 0,6. Die Gefäßvisualisierung für Datensatz 2 enthält im Falle einer maximalen Tiefendifferenz von 5,0 lediglich 32,2 %

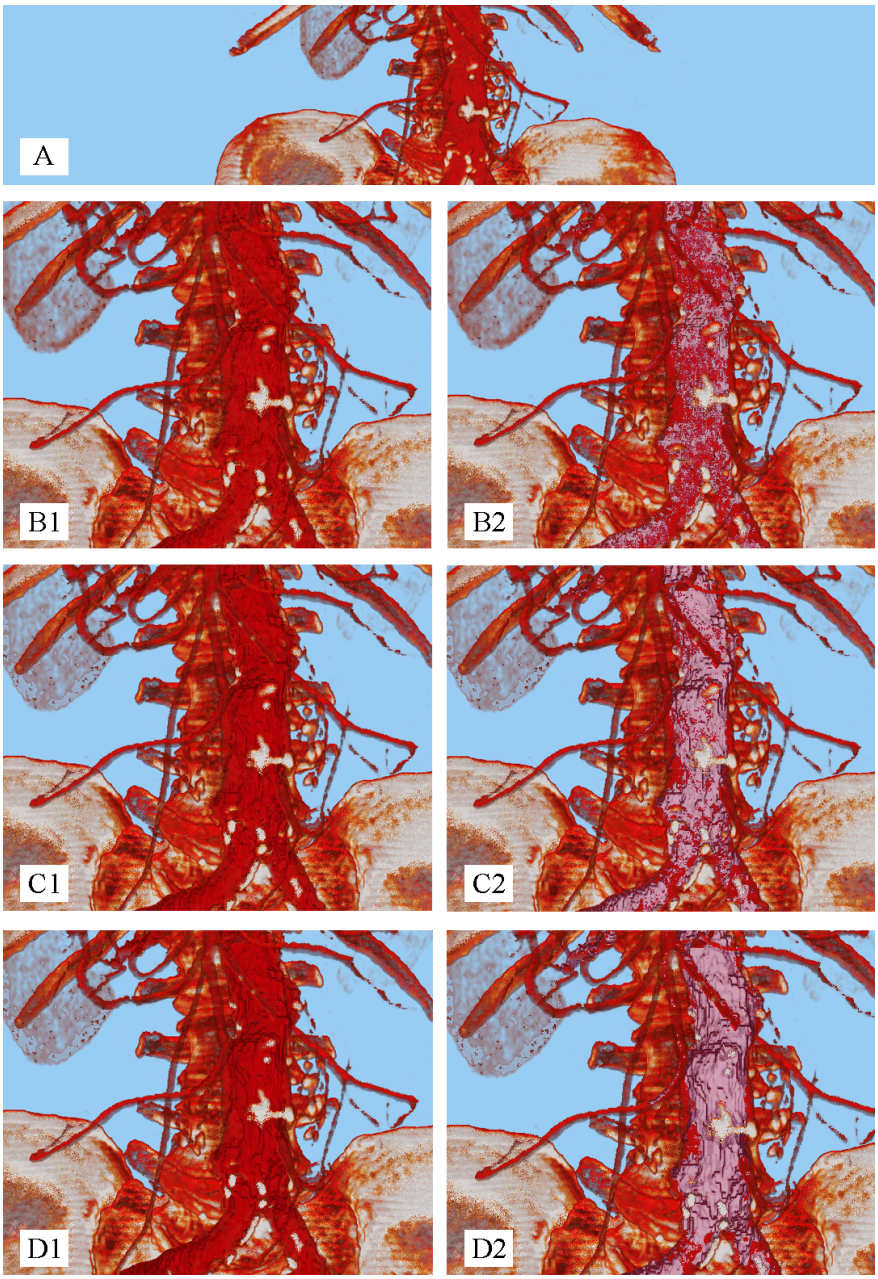


Abbildung 52: Anteil der Marching Cubes Oberfläche an der hybriden Visualisierung abhängig von der maximal erlaubten Tiefendifferenz zwischen Oberfläche und Raycasting–Ergebnis. **A:** indirekte Volumendarstellung ohne integrierte Oberfläche. **B, C, D:** Hybride Visualisierung mit einer maximal erlaubten Tiefendifferenz von 1,0 (B1, B2), 1,5 (C1, C2) sowie 5,0 (D1, D2). Der Marching Cubes Anteil ist in den rechten Bildern violett hervorgehoben.

Oberflächenanteil; dieser weißt allerdings nur eine durchschnittliche Tiefendifferenz von 0,9 auf. Würde die Marching Cubes Oberfläche vollständig und ohne Beschränkung durch eine maximale Tiefendifferenz gerendert werden, läge der durchschnittliche Bildfehler für Datensatz 1 bei 12,1 und für Datensatz 2 bei 22,6. Verglichen mit diesen Werten können durch den Einsatz maximal erlaubter Tiefendifferenzen verlässlichere Bilder generiert werden.

Neben der Genauigkeit hängt auch die **visuelle Qualität** der hybriden Darstellung von der maximal erlaubten Tiefendifferenz zwischen Marching Cubes Oberfläche und direkter Volumenvisualisierung ab. Abbildung 52A zeigt einen Ausschnitt der durch Raycasting erzeugten dreidimensionalen Darstellung von Datensatz 1. Die darunter angeordneten Bilder integrieren die Marching Cubes Oberfläche mit einer maximal erlaubten Tiefendifferenz von

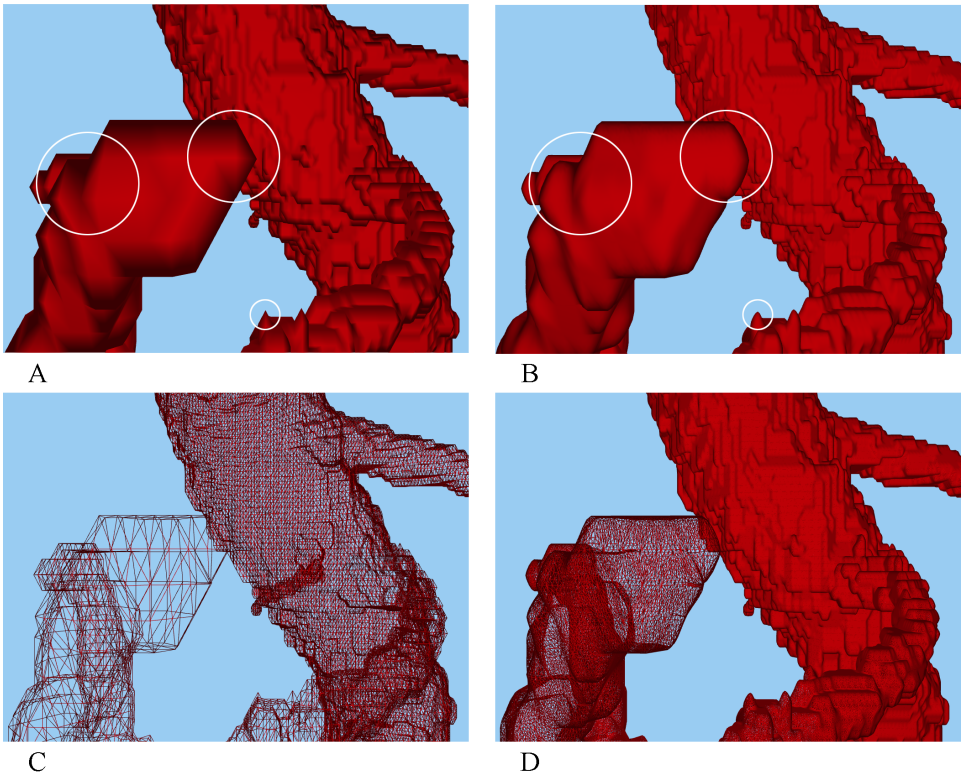


Abbildung 53: Ergebnis der Tesselierung der Marching Cubes Oberfläche. **A:** Visualisierung der Marching Cubes Oberfläche ohne Tesselierung. Die Markierungen heben die starke Eckigkeit der Darstellung, Beleuchtungsartefakte sowie die wenig genau berechnete Schattierung hervor. **B:** Tesselierte Visualisierung der Marching Cubes Oberfläche mit glatteren Konturen, genaueren Schatten und ohne Beleuchtungsartefakte. **C, D:** Polygonnetz der Marching Cubes Oberfläche mit aus– und eingeschalteter Tesselierung.

1,0 (s. Abbildung 52B), 1,5 (s. Abbildung 52C) und 5,0 (s. Abbildung 52D). In den rechten Bildern wird der Oberflächenanteil jeweils violett hervorgehoben. Die Plastizität der Darstellung nimmt mit der maximalen Tiefendifferenz und dem dadurch wachsenden Oberflächenanteil zu. Die Struktur der Gefäßoberfläche wird ebenso wie die äußere Kontur betont. Aus der Abbildung wird ersichtlich, dass mit guten Segmentierungsergebnissen bereits eine geringe maximale Tiefendifferenz ausreichend ist, um eine sichtbare Verbesserung der Darstellung zu erzielen.

Die Visualisierung ist nicht vollständig frei von Artefakten. Durch Stochastic Jittering konnten zwar die für Raycasting üblichen Holzmaserung–ähnlchen Artefakte entfernt werden, allerdings wirkt die direkte Volumendarstellung, insbesondere die Knochen, erwartungsgemäß verrauscht. Da das Fokusobjekt – die Aorta mit ihren abgehenden Gefäßen – vom Rauschen nicht betroffen ist, fällt dieses Artefakt jedoch kaum störend auf.

Der für Marching Cubes übliche Treppenstufeneffekt konnte durch Kombination mit Raycasting zwar nicht nennenswert kaschiert werden, dennoch wurde eine Integration der Marching Cubes Oberfläche in die direkte Volumenvisualisierung erreicht, ohne dass störende Übergänge zwischen den verschiedenen Darstellungsmethoden sichtbar sind. Durch Tesselierung des Polygonnetzes war eine Glättung der Oberfläche ebenfalls nicht möglich. Abbildung 53A zeigt einen Ausschnitt der nicht tesselierten Oberfläche gegenüber ihrem tesselierten Pendant in Abbildung 53B. An den markierten Stellen wird deutlich, dass eine Tesselierung sowohl glattere Konturen, als auch genauere Schatten erzeugt und dabei gleichzeitig Beleuchtungsartefakte entfernt. Für die Marching Cubes Oberfläche sind diese positiven Effekte jedoch nur lokal sichtbar, da das ursprüngliche Polygonnetz, wie in Abbildung 53C dargestellt bereits sehr feinmaschig ist. Der Performanzverlust, der aufgrund der Vielzahl neu generierter Vertizes im tesselierten Polygonnetz (s. Abbildung 53D) entsteht, überwiegt die global kaum wahrnehmbaren Verbesserungen deutlich. Aus diesem Grund wurde die Tesselierung standardmäßig deaktiviert und für die oben ausgeführte Evaluation der Bildqualität nicht berücksichtigt.

6.3 Expertenbefragung

Zur Evaluation der **Verständlichkeit** und der **Anwendbarkeit** des hybriden Systems wurde eine Gruppe von Experten – jeweils zwei Gefäßchirurgen und Radiologen – befragt. Im Rahmen einer Kurzpräsentation wurde ihnen das hybride Renderingsystem vorgestellt. Die Visualisierung wurde dabei gegenüber vollständig indirekt und vollständig direkt gerenderten Gefäßdarstellungen abgegrenzt. Im Anschluss sollte die hybride Visualisierung von den Medizinern anhand eines Fragebogens (s. Anhang D) bewertet werden.

Abbildung 54 fasst die Ergebnisse der Expertenbefragung hinsichtlich Verständlichkeit, Anwendbarkeit, Tiefeneindruck und Gesamteindruck zu-

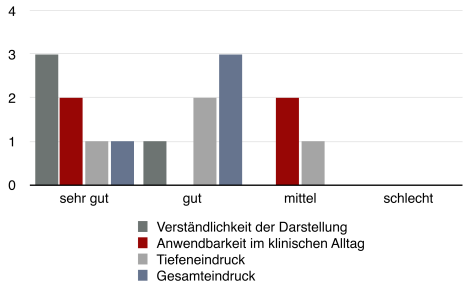


Abbildung 54: Expertenbeurteilung des hybriden Renderers mit Hinblick auf Verständlichkeit, Anwendbarkeit, Tiefen– und Gesamteindruck.

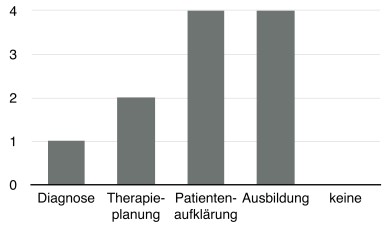


Abbildung 55: Klinische Anwendbarkeit des hybriden Renderers nach Einschätzung der befragten Experten.

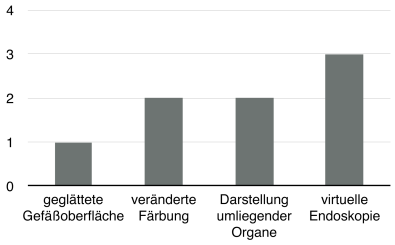


Abbildung 56: Von Expertenseite aus wünschenswerte Optimierungen und Ergänzungen des hybriden Systems.

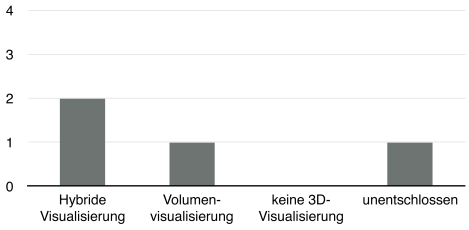


Abbildung 57: Bevorzugte Volumenvisualisierungstechnik der befragten Experten.

sammen. Im Durchschnitt bewerteten die Mediziner die hybride Visualisierung in jeder Kategorie mindestens mit *gut*. Die Verständlichkeit der Darstellung wurde sogar mehrheitlich mit *sehr gut* beurteilt. Die größten Unstimmigkeiten zwischen den Medizinern zeigten sich in der Bewertung der Anwendbarkeit des Systems im klinischen Alltag. Jeweils ein Radiologe und ein Chirurg befanden die Visualisierung als *sehr gut* beziehungsweise *mittelmäßig gut* anwendbar.

Einig waren sich die Mediziner allerdings in der Anwendbarkeit der hybriden Darstellung zur Patientenaufklärung sowie zu Aus– und Weiterbildungszwecken (s. Abbildung 55). Weiterführend wurden klinische Demonstrationen als potentielles Anwendungsgebiet genannt. Verwendungsmöglichkeiten in der Therapieplanung wurden dem System lediglich von den beiden Chirurgen zugeschrieben. Ein Chirurg bewertete die Visualisierung außerdem im Rahmen der Diagnose sowie unterstützend im Falle intraoperativer Komplikationen als sinnvoll.

Die hybride Visualisierung wurde als detaillierte Darstellung mit anatomischem Bezug durch integrierte Abbildung der Knochen positiv wahrgenommen. Der Kombination aus indirekter Oberflächendarstellung und direktem Rendering der Kalzifizierungen wurde das Potential für exaktere OP–Planungen beigemessen. Auf der anderen Seite sahen die Mediziner die Datenvereinfachung, die eine Verarbeitung der C-T-Bilder unweigerlich nach sich zieht, zumeist kritisch. Für eine Weiterentwicklung der hybriden Visualisierung wurde die Darstellung von Informationen über die Beschaffenheit der Gefäßwand angeregt. Für einen erhöhten Tiefeneindruck sollten Schatten implementiert werden.

Abbildung 56 zeigt die Präferenzen der Experten hinsichtlich im Fragebogen vorgeschlagener Optimierungs– und Erweiterungsmöglichkeiten. Die Integration einer virtuellen Endoskopie zur Betrachtung der Gefäßinnenseite wurde von drei der vier Befragten als wünschenswert benannt. Optimalerweise sollte im Rahmen einer solchen virtuellen Endoskopie auch die Möglichkeit der Messung von Winkeln und Durchmessern innerhalb des Gefäßes zur Planung realer Angioskopien möglich sein. Die Darstellung umliegender Organe hielten lediglich zwei Mediziner für sinnvoll. Ein dritter befand ausschließlich die Visualisierung der Knochen zur räumlichen Orientierung als nützlich. Ebenfalls zwei der befragten Mediziner gaben an, eine veränderte Färbung zu präferieren. Dies betraf zum einen die stärkere farbliche Differenzierung zwischen Gefäß und Knochen, zum anderen die Möglichkeit zur farblichen Hervorhebung einzelner Gefäßabschnitte. Eine glatte Gefäßoberfläche wurde von einem Experten gewünscht; zwei Mal wurde sie mit Hinweis auf den dadurch entstehenden Datenverlust entschieden abgelehnt.

In der Gesamtbeurteilung (s. Abbildung 57) entschieden sich zwei Mediziner bevorzugt die hybride Darstellung nutzen zu wollen, in einem Fall wurde die direkte Volumendarstellung mittels Raycasting präferiert. Einer der Radiologen gab an, abhängig vom Anwendungsfall, sowohl die hybride als auch die direkte Volumenvisualisierung nutzen zu können; sie würden dabei jedoch ausschließlich als ergänzende Darstellung zum CT–Datensatz herangezogen werden.

6.4 Laienbefragung

Zusätzlich zu der Expertenbefragung wurde in einer Laienbefragung mit zwölf Testpersonen die **Verständlichkeit** der hybriden Visualisierung und damit die **Anwendbarkeit** speziell für Patientenaufklärungsgespräche evaluiert. Dazu wurde ohne jegliche Bildunterstützung erläutert, was ein Aneurysma ist und wie es entsteht, diagnostiziert und behandelt wird. Anschließend wurde zunächst der CT–Datensatz eines Aneurysma–Patienten erklärt, gefolgt von einer direkten Volumenvisualisierung dieses Datensatzes und der hybriden Darstellung. In allen drei Fällen wurde explizit auf die Lage des Aneurysmas und die vorliegenden Kalzifizierungen hingewiesen. Im Anschluss

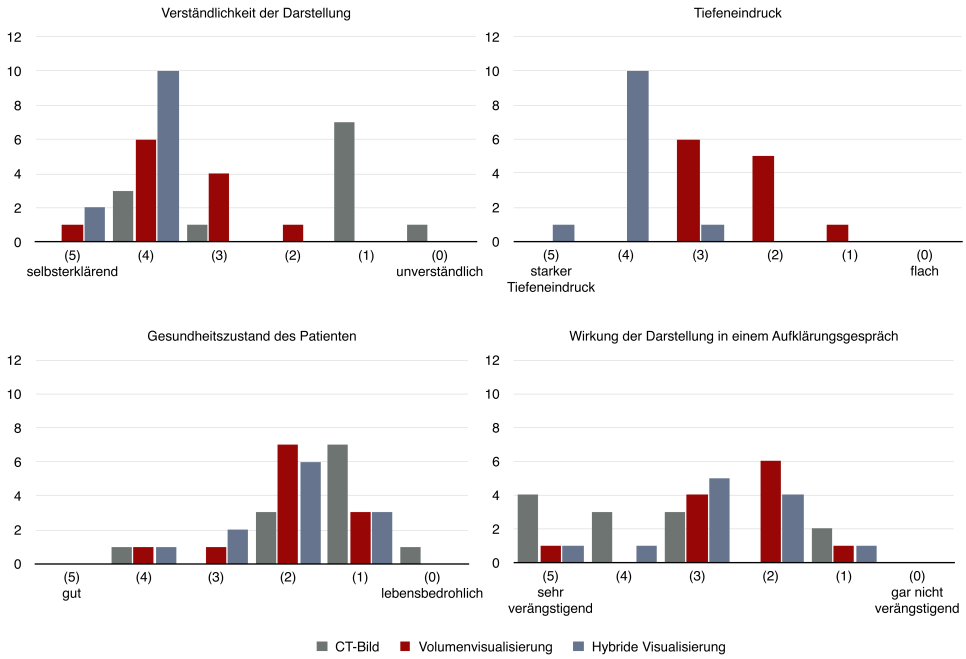


Abbildung 58: Laienbeurteilung der Darstellung medizinischer Sachverhalte durch CT–Daten, direkte Volumenvisualisierung und hybride Visualisierung. Die Bewertung umfasst die Verständlichkeit der Darstellung, den Tiefeneindruck bei dreidimensionalen Visualisierungen, die Einschätzung des Gesundheitszustands des Patienten sowie die erwartete Wirkung der Darstellung in einem Patientenaufklärungsgespräch.

sollten die Laien das CT–Bild, die direkte Volumenvisualisierung und die hybride Visualisierung in einem Fragebogen (s. Anhang E) bewerten.

Abbildung 58 fasst die Ergebnisse der Laienbefragung zusammen. Die hybride Darstellung wurde im Schnitt mit 4,2 von maximal 5 Punkten als sehr gut verständlich bewertet. Sie erreichte damit marginal bessere Bewertungen als die direkte Volumendarstellung mit 3,6 Punkten. Im Vergleich weit abgeschlagen liegt das CT–Bild mit einem Durchschnittswert von 1,8. Die schichtweise Betrachtung des CT–Datensatzes wurde damit als nahezu unverständlich eingestuft.

Beide Volumenvisualisierungsmethoden sollten weiterhin anhand des erzeugten Tiefeneindrucks verglichen werden. Deutlich bessere Ergebnisse konnte dabei die hybride Variante mit einer durchschnittlichen Bewertung von 4,0 erzielen; die direkte Volumenvisualisierung erreichte nur knapp mehr als die Hälfte dieser Punktzahl mit einem Durchschnittswert von 2,4 der maximal möglichen 5 Punkte.

Der Zustand des Patienten wurde mit durchschnittlich 1,4 von 5 Punkten am kritischsten anhand der CT–Bilder bewertet. Bei Durchschnittswerten

von 2,0 beziehungsweise 2,1 waren zwischen der direkten Volumendarstellung und der hybriden Visualisierung keine Unterschiede feststellbar.

Abschließend sollten sich die Befragten in die Situation eines Patientenaufklärungsgesprächs hineinversetzten und die Wirkung der einzelnen Darstellungen auf einer Skala von 0 (gar nicht verängstigend) bis 5 (sehr verängstigend) bewerten. Die CT–Bilder wurden zwar mit 3,6 Punkten durchschnittlich als die erschreckendste Visualisierungsmöglichkeit wahrgenommen, allerdings teilten sich die einzelnen Bewertungen auf nahezu der gesamten Skala relativ gleichmäßig auf. Bei den dreidimensionalen Visualisierungsmethoden konnte dagegen eine klare Tendenz festgestellt werden: Die direkte Volumenvisualisierung lag bei einem Durchschnittswert von 2,5. Demgegenüber wurde die hybride Visualisierung durchschnittlich mit 2,8 Punkten als leicht verängstigender bewertet.

Auf die Frage hin, welche Visualisierungstechnik sie in einem Patientenaufklärungsgespräch bevorzugen würden, gaben alle befragten Laien die hybride Darstellung an. Dabei wurde in fünf Fällen eine Kombination mit dem CT–Bild gewünscht; eine Testperson befand das Volumenrendering und die hybride Visualisierung als gleichwertig geeignet (s. Abbildung 59). Die Begründungen stützten sich in erster Linie auf die bessere Verständlichkeit der hybriden Darstellung gegenüber den CT–Bildern und die erhöhte Plastizität im Vergleich zur direkten Volumenvisualisierung. Beides erzeuge eine realistisch und natürlich wirkende Darstellung, die bekannten Bildern des Körperinneren entspricht. Dadurch sei die hybride Visualisierung deutlich leichter zugänglich als die vergleichsweise abstrakt wahrgenommenen CT–Bilder. Die zusätzliche Abbildung – und einfache Erkennbarkeit – umliegender Kontextobjekte erleichtere darüber hinaus die Einschätzung von Größenverhältnissen und die räumliche Orientierung.

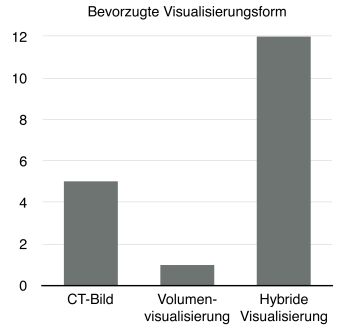


Abbildung 59: Bevorzugte Visualisierungsform der Laien für ein Patientenaufklärungsgespräch.

Abbildung 60: Von Seiten der Laien gewünschte Optimierungen und Ergänzungen des hybriden Systems.

Für eine kombinierte Anwendung von CT–Bildern und hybrider Visualisierung spricht laut Aussage der Laien, dass die beiden völlig unterschiedlichen Darstellungen den Sachverhalt auf jeweils andere Art und Weise veranschaulichen könnten. Während aus CT–Bildern die Schwere der Erkrankung – insbesondere bei schlecht kontrastiertem Lumen – ersichtlich würde, sei die hybride Visualisierung die beste Variante, um einen Überblick zu gewinnen und gleichzeitig den Fokus auf das Wesentliche lenken zu können.

In Abbildung 60 ist dargestellt, welche Verbesserungen am hybriden Rendering aus Sicht der Laien sinnvoll wären. Gewünscht wurde in erster Linie die Möglichkeit umliegende Organe (teilweise) ein– und ausblenden zu können. Neun der zwölf Befragten führten außerdem Ideen für eine veränderte Farbgebung an. So sollten kritische Stellen farblich hervorgehoben oder die Rupturgefahr entlang eines Gefäßes durch unterschiedliche Rottöne kodiert werden. Des Weiteren wurde eine stärkere farbliche Trennung von Gefäß, Knochen und Kalzifizierungen vorgeschlagen. Die Darstellung der Knochen könnte dabei durch Transparenz innerhalb des Gesamtbildes zurückgenommen werden. Transparenzen wurden außerdem als sinnvolles Mittel genannt, um die Grenzen des Körpers anzuzeigen und damit die Größenverhältnisse innerhalb des Bildes zusätzlich zu verdeutlichen. Die Nachfrage nach einer geglätteten Gefäßoberfläche war dagegen vernachlässigbar gering.

7 Diskussion

7.1 Beurteilung der hybriden Visualisierung von vaskulären Gefäßen

In dieser Arbeit wurde ein hybrides Rendering System zur optimierten Darstellung von Blutgefäßen durch Integration einer Marching Cubes Oberfläche in ein Raycasting–System entwickelt. Unter Verwendung moderner (GP)GPU–Technologien konnte auf einem hochleistungsfähigen Windows–PC problemlos Echtzeitfähigkeit erreicht werden; selbst auf dem MacBook Air wurde größtenteils mit interaktiven Framezeiten gerendert. Die Kombination des Polygonnetzes in die direkte Volumenvisualisierung gelang ohne die Entstehung auffälliger Übergänge. Im Vergleich zu Raycasting konnte mit der hybriden Visualisierung eine höhere Plastizität erreicht werden. Gleichfalls liefert das entwickelte System genauere Ergebnisse als eine reine Marching Cubes Darstellung. Im gerenderten Bild sind sowohl schmale Gefäßstrukturen als auch Kalzifizierungen entlang der Gefäßwand erkennbar. Zur besseren räumlichen Orientierung konnte Kontextinformation in Form der umliegenden Knochen in die Darstellung integriert werden.

Auf einem modernen Windows–PC konnten vaskuläre Gefäße und umliegende Knochen problemlos mit dem hybriden Rendering–System in **Echtzeit** visualisiert werden. Die Navigation erfolgt flüssig; weitere Interaktionsmöglichkeiten sollten ohne merklichen Performanzverlust integriert werden können. Die Framezeiten, die auf Basis der vorliegenden Implementierung mit einem MacBook Air erreicht werden können, erlauben interaktive Anwendungen innerhalb der virtuellen Umgebung. Die benötigte Reaktionszeit des Systems nimmt dabei jedoch mit zunehmender Nähe zum visualisierten Objekt deutlich zu. Auch die Navigation ist unter den OS X–Testsystemen mit leichtem Ruckeln verbunden.

Die längsten Berechnungszeiten während des Renderings benötigt die Raycasting–Komponente. Die Marching Cubes Oberfläche konnte auf allen Testsystemen in Echtzeit dargestellt werden. Für die hybride Visualisierung wird zusätzlich zu den Framezeiten von Raycasting und Marching Cubes nur ein geringer zeitlicher Mehraufwand durch die Kombination erzeugt. Daraus folgt, dass eine effiziente Implementierung der zu kombinierenden direkten und indirekten Visualisierungsverfahren zwangsläufig in einem effizienten hybriden System resultiert.

Den größten Einfluss auf die Performanz des Systems hat die Einstellung der Fenstergröße. Anhand der durchgeführten Messungen konnte gezeigt werden, dass schon die Vergrößerung von 800×600 Pixeln auf 1440×1080 Pixeln einen größeren Leistungsverlust bedeutet als die Verdopplung der Auflösung des CT–Datensatzes bei gleichbleibender Fenstergröße. Dies lässt sich dadurch erklären, dass die Raycasting–Komponente des hybriden Systems,

wie oben bereits ausgeführt, den meisten Rechenanteil eines Renderdurchlaufs ausmacht. Da während des Raycastings nicht pro Voxel, sondern pro Pixel gearbeitet wird, ist folglich die Fenstergröße ausschlaggebend für die erreichten Framezeiten. Auffällig ist, dass der Performanzverlust bei Fenstervergrößerung umso höher ausfällt, je geringer aufgelöst der verarbeitete CT–Datensatz ist. Eine Erklärung könnte darin liegen, dass für niedrig aufgelöste Datensätze bei weiten Fenstergrößen unter Umständen häufiger interpoliert werden muss. Außerdem könnte der Zugriff auf immer gleiche Speicherstellen den Algorithmus verlangsamen. Weitere Tests, bei denen die Berechnugszeiten gegebenenfalls auch kleinschrittiger bestimmt werden, sind notwendig, um die angestellten Vermutungen zu validieren und bisherige Messergebnisse zu bestätigen.

Auch die erhöhten Framezeiten mit abnehmendem Abstand zwischen Kamera und Objekt sind auf das Raycasting zurückzuführen. Die Leistung in der Darstellung der Marching Cubes Oberfläche verändert sich, wie die Testergebnisse zeigten, dagegen nicht nennenswert. Für das Raycasting bestimmt die Entfernung, wie viele Pixel des Fensters die Hüllgeometrie abbilden. Daraus ergibt sich, wie viele Strahlen verfolgt werden müssen. Zunehmende Nähe bedeutet dabei eine zunehmende Anzahl zu verfolgender Strahlen. Sobald die Hüllgeometrie den gesamten Sichtbereich einnimmt, ist keine weitere Verschlechterung der Framezeit zu erwarten.

Um die Gesamtleistung des hybriden Renderingsystems zu verbessern, ist nach obigen Ausführungen eine Optimierung der Raycasting–Komponente notwendig. Deutlich geringere Framezeiten sind mit der Implementierung von Empty Space Skipping zu erwarten. Im Rahmen der Vorverarbeitung benötigt die Aufteilung des CT–Datensatzes auf eine Octree–Struktur dann zusätzliche Rechenzeit. Auch die Veränderung des Isowerts verlangt bei Empty Space Skipping eine Neuberechnung der Hüllgeometrie. Beide Nachteile sollten durch die erheblich besseren Renderingszeiten jedoch aufgewogen werden.

Sowohl für die Bestimmung einer optimierten Hüllgeometrie als auch für das eigentliche Raycasting bietet sich die Verwendung von OpenCL an, da jeweils einzelne Datensatzblöcke beziehungsweise zu verfolgende Strahlen unabhängig voneinander verarbeitet werden können. Unter OS X ließe sich damit möglicherweise eine weitere Optimierung der Rechenzeit erzielen. Für Windows hat bereits die OpenCL–Umsetzung des Marching Cubes Algorithmus nur vergleichsweise schlechte Ergebnisse geliefert. Die Ursache konnte auf eine deutlich längere OpenCL–Initialisierungszeit zurückgeführt werden. Direct Compute – die windowsspezifische Variante der plattformunabhängigen Schnittstelle OpenCL – sollte eine bessere Kompatibilität aufweisen und könnte damit schnellere Berechnungen ermöglichen. Werden Nvidia Grafikkarten verwendet, bietet CUDA eine weitere Alternative. Seit OpenGL 4.4 ist für denselben Zweck der Einsatz von ComputeShadern möglich. Prinzipiell erwies sich die Verwendung von GPGPU hinsichtlich der Berechnungszeit

(abzüglich der OpenCL–Initialisierungszeit) auf allen Systemen als günstig. Mit dem Einsatz von 3D–Images anstelle von Buffern ist eine zusätzliche Beschleunigung des Marching Cubes Algorithmus möglich. Inwiefern GP-GPU auch unter Windows ohne lange Initialisierungszeiten eingesetzt werden kann, ist anhand von Direct Compute–, CUDA– und ComputeShader–Umsetzungen zu validieren.

Das größte Problem in der Verarbeitung der CT– und Segmentierungsdaten auf der GPU liegt im **begrenzten Grafikkartenspeicher**. Die Generierung der Marching Cubes Oberfläche ist abhängig von der genutzten Grafikkarte nur bis zu einer begrenzten Größe in einem Zuge durchführbar. Für die verwendeten OS X–Testsysteme konnten lediglich Segmentierungsdatensätze mit einer Auflösung von maximal $512 \times 512 \times 512$ verarbeitet werden. Scharsach stellt in [Sch05] eine Methode vor, die dasselbe Problem für Raycasting löst. Dazu wird der zugrunde liegende Datensatz in eine Blockstruktur unterteilt, wobei die Größe jedes Blocks den maximal verfügbaren Grafikkartenspeicher annähert. Auf diese Weise kann die Verarbeitung des Datensatzes auf der GPU stückweise ausgeführt werden.

Für die Marching Cubes Implementierung mittels OpenCL lässt sich das Verfahren adaptieren. Der Segmentierungsdatensatz würde dazu in gleich große, den Grafikspeicher ausfüllende Würfel unterteilt werden. Um die Entstehung von Lücken im Polygonnetz zu verhindern, ist eine Überlappung der Würfel nötig und die Generierung weniger doppelter Dreiecke unumgänglich.

Die Rechenzeit sollte bei dieser Technik wenn überhaupt lediglich durch den CPU–GPU–Transfer der Daten negativ beeinflusst werden. Die dafür benötigte Zeit ist in entsprechenden Tests zu messen und zu evaluieren. Die Initialisierung von OpenCL wäre nur einmalig notwendig.

Für das hybride Renderingsystem konnte – verglichen mit reinen Oberflächenvisualisierungen – eine verbesserte **Genauigkeit** nachgewiesen werden. Es wurde gezeigt, dass auch die Abweichung von einer direkten Volumenvisualisierung nur gering ausfällt, wenn mit einer maximal erlaubten Tiefendifferenz zwischen Marching Cubes Oberfläche und Raycasting–Ergebnis gearbeitet wird. Durch den Einsatz von Hitpoint–Refinement innerhalb der Raycasting–Komponente ließe sich die Genauigkeit weiterhin verbessern. Die hybride Visualisierung ist durch den maßgeblichen Einsatz von Raycasting in der Lage sowohl schmale Strukturen als auch die Durchmesser einzelner Gefäßabschnitte korrekt wiederzugeben. Die Kompensierung aller Nachteile von direkten und indirekten Visualisierungen war durch ihre Kombination jedoch nicht möglich. So konnte weder das Rauschen im Raycasting–Ergebnis, noch der Treppenstufeneffekt in der Marching Cubes Oberfläche verhindert werden.

Die Befragung der Laien und Experten ergab keinen Hinweis darauf, dass die genannten Artefakte im Gesamtbild stören. Zumindest der Treppenstu-

feneffekt ließe sich dennoch durch eine Glättung mit einem hinlänglich detaillerhaltenden Low–Pass–Filter verringern. Der auf diese Weise an einigen Stellen entstehende Fehler im Polygonnetz wird durch die maximal erlaubte Tiefendifferenz abgefangen und beeinflusst die Genauigkeit der hybriden Darstellung somit nicht negativ. Wenn eine nachträgliche Glättung vorgenommen wird, ist lediglich mit einem vergleichsweise geringeren Oberflächenanteil im Gesamtergebnis zu rechnen. Die Sorge einiger Experten, dass die Genauigkeit der Darstellung mit einer Glättung deutlich verloren ginge, ist damit unbegründet. Zwar kann der natürliche Datenverlust durch die Verarbeitung im hybriden System nicht verhindert werden, die getroffenen Vorkehrungen sind jedoch geeignet, um ihn bewusst einzuschränken.

Andere Artefakte, die durch die Beleuchtung des Polygonnetzes entstehen, konnten durch Tesselierung entfernt werden. Gleichzeitig ermöglichte der Einsatz von PN–Triangles eine verbesserte Schattierung sowie die Darstellung abgerundeter Ecken. Dabei musste jedoch festgestellt werden, dass die Tesselierung auf feinmaschigen Polygonnetzen im Verhältnis zur benötigten Rechenezeit nicht ausreichend optische Verbesserungen erzielt. Ob der Nutzen auf grobmaschigen Netzen deutlicher wird, müsste anhand erweiterter Implementierungen in zusätzlichen Tests validiert werden. Zu diesem Zweck ist sowohl der Einsatz anderer Oberflächenrekonstruktionsalgorithmen denkbar als auch eine nachträgliche Zusammenfassung planar angeordneter Dreiecke im Marching Cubes Polygonnetz. Im letzten Fall ist jedoch fragwürdig, ob die Vereinfachung der Oberflächenstruktur in akzeptabler Rechenzeit durchgeführt werden kann.

Die wichtigste Anpassung zur Verbesserung von Genauigkeit und Bildqualität ist der Einsatz optimierter, mehrdimensionaler Transferfunktionen für das Raycasting sowie für die Berechnung des Segmentierungsergebnisses. Nur so lässt sich beispielsweise das in Datensatz 1 nicht kontrastierte Gefäßlumen auf Höhe des Aneurysmas darstellen. Zusätzlich ermöglichen bessere Transferfunktionen eine exaktere Tiefenbestimmung im Raycasting–Algorithmus, eine genauere Darstellung schmaler Gefäßstrukturen sowie eine eindeutigere Trennung der Gefäße von anderen Objekten. Vermutlich nicht korrigieren lassen sich auf diese Weise Artefakte, die in schlechten CT–Aufnahmen vorliegen und in die dreidimensionale Visualisierung, wie im Falle des verrauschten Testdatensatzes 2, übertragen werden.

Anhand der Laienbefragung konnte gezeigt werden, dass die Integration der Marching Cubes Oberfläche in das Raycasting–System die **Plastizität** und damit auch die **Wahrnehmung der räumlichen Anordnung** innerhalb der Visualisierung erhöht. Durch geeignete Beleuchtung sollte der Effekt deutlich verstärkt werden können. Die vorliegende Implementierung nutzt lediglich eine simple Phong'sche Beleuchtung. Um den Tiefeneindruck weiter zu verbessern, kann eine Verschattung der Oberfläche und der direkten Volumendarstellung implementiert werden. Kubisch et al. schlagen außer-

dem *Ambient Occlusion* für eine bessere räumliche Orientierung innerhalb medizinischer Visualisierungen vor. Diese fortgeschrittene Beleuchtungstechnik ermöglicht nach Aussage der Autoren die optimierte Wahrnehmung sich überlagernder Gefäße. Der hohe Rechenaufwand würde die optische Verbesserung allerdings vorerst auf high–end Grafikkarten beschränken. Deshalb sollten aufwändige Beleuchtungsmethoden nicht standardmäßig sondern optional in das System integriert werden. Somit ließe sich die Anwendung auch auf rechenschwächeren, gegebenenfalls tragbaren Geräten problemlos ausführen. Wenn dabei auch auf photorealistische Darstellung verzichtet werden muss, so kann dennoch ein verbesserter 3D–Eindruck durch den Einsatz von hybridem Rendering im Vergleich zu Raycasting erreicht werden. Ist dagegen mit der verwendeten Hardware eine photorealistische Darstellung möglich, kann der Realitätsgrad innerhalb der Visualisierung deutlich erhöht werden, was insbesondere in Aufklärungsgesprächen die Verständlichkeit für den Patienten erleichtern sollte.

Das hybride Renderingsystem ist in der Lage die Topologie vaskulärer Strukturen zuverlässig zu visualisieren. Eine detaillierte **Darstellung der Gefäßwand und der Gewebestruktur** ist aktuell nicht implementiert. Multidimensionale Transferfunktionen sollten in der Lage sein, Informationen über die Wandbeschaffenheit und vorhandene Dissektionen aus dem CT–Datensatz zu extrahieren. Unter Hinzunahme von Transparenzen ist eine geeignete Darstellung dieser Informationen innerhalb des hybriden Systems denkbar. Der Einsatz von Transparenzen sollte dabei jedoch gering gehalten werden, da sie sonst die Rechenzeit stark verringern und das Verständnis der Darstellung – insbesondere die räumliche Orientierung – erschweren können.

Um die gesamte Gefäßwand semi–transparent sichtbar zu machen, ist es notwendig bei einem Isowert–Treffer im Raycasting lediglich die Tiefe zu speichern, nicht aber den Alpha–Wert auf eins zu setzen. In diesem Zusammenhang ist auch eine angepasste Komposition sinnvoll: Wird beim Raycasting neben der Tiefe bei einem Isowert–Treffer auch die Anzahl vorangegangener, nicht volltransparenter Positionen entlang eines Strahls gespeichert, lassen sich daraus gegebenenfalls Parameter für die Gewichtung im Blending ableiten. Auf diese Weise könnte der Marching Cubes Anteil an einer Position verringert werden, wenn im Raycasting eine Vielzahl an transparenten Informationen vor der Oberfläche detektiert wurde. Farbliche Übereinstimmungen zwischen Raycasting–Ergebnis und Marching Cubes Oberfläche sollten sich dadurch erzielen lassen, dass die Oberflächentextur lediglich die Beleuchtung auf weißer – nicht wie aktuell implementiert roter – Materialfarbe speichert. Wird der Marching Cubes Anteil beim Blending zusätzlich mit der im Raycasting ermittelten Farbe multipliziert, ist ein optisch unauffälliger Übergang zwischen oberflächenergänzter Darstellung und direkter Volumenvisualisierung zu erwarten. Diese Vorgehensweise könnte auch das explizite Abfangen von Kalzifizierungen innerhalb der Komposition ersetzen.

7.2 Integration der hybriden Visualisierung von vaskulären Gefäßen in ein bestehendes System

Die Integration einer Oberflächenstruktur in ein Raycasting–System macht die hybride Visualisierung zu einer mächtigen Methode für die Umsetzung diverser **Interaktionsmöglichkeiten**. So sollte es beispielsweise ohne Weiteres möglich sein durch Anklicken einer Position auf dem dargestellten Gefäß den Schnittpunkt mit der darunterliegenden, gegebenenfalls nicht einmal angezeigten Marching Cubes Oberfläche zu bestimmen und das korrespondierende CT–Bild im Split–Screen–Modus zu öffnen. Eine solche Funktion ließe sich kombinieren mit der Einblendung einer Ebene durch den Schnittpunkt, die zur y–Achse senkrecht steht und auf die bei Bedarf das CT–Bild als Textur projiziert werden kann. Mit der Auf– und Abbewegung dieser Ebene könnte dann das Blättern durch den im Split–Screen–Modus angezeigten CT–Datensatz realisiert werden. In einem solchen kombinierten System bietet die dreidimensionale Volumenvisualisierung den Vorteil, dass neben einem CT–Bild immer auch die gesamte Kontextinformation angezeigt wird und die aktuelle Lage im Raum festgestellt werden kann.

Neben Ebenen zur Orientierung sollten weitere Interaktionselemente, wie Pfeile oder dreidimensionale Darstellungen chirurgischer Instrumente und endoskopischer Werkzeuge, einfach in die hybride Visualisierung integrierbar sein. Derartige Hilfsmittel bieten das Potential wichtige Regionen im Datenraum hervorzuheben und die Interpretation zu erleichtern.

Eine Funktion, die das hybride Rendering innerhalb eines bestehenden Systems zusätzlich einschließen könnte, wäre die Möglichkeit mithilfe geeigneter Transferfunktionen weitere Kontextinformationen ein– und auszublenden. Beispielsweise wäre die Darstellung der Silhouette eines Körpers, die in einem zusätzlichen Renderpass berechnet werden müsste, oder die Visualisierung weiterer Organe denkbar.

Der entscheidende Vorteil der hybriden Visualisierung im Vergleich zur direkten Volumenvisualisierung besteht darin, dass durch Integration der Oberflächendarstellung Messungen innerhalb des Gefäßobjekts möglich sein müssten. Dies betrifft sowohl die Berechnung des Gefäßdurchmessers als auch die Bestimmung von Winkeln an Verzweigungen, beispielsweise anhand eines noch zu integrierenden Skeletts. Solche Vermessungen sind mit direkten Volumenvisualisierungsmethoden nicht umsetzbar [OP05]; für die Nutzbarkeit im klinischen Alltag sind sie jedoch von großer Bedeutung. Im hybriden System ist die Durchführung direkt anhand der Oberflächendarstellungen denkbar. Eine andere Möglichkeit wäre ein interner Aufruf des interessierenden CT– oder Segmentierungsbildes durch einen Mausklick auf die entsprechende Region und eine anschließende, möglichst automatisch ablaufende Vermessung innerhalb dieser 2D–Darstellung. Mit Messungen, die auf dem ursprünglichen Datensatz basieren, ließe sich verhindern, dass Fehler und Ungenauigkeiten akkumuliert werden, die durch die Visualisierung entstanden sind.

Eine weitere Funktion, die von den befragten Medizinern mehrfach als nützlich erachtet wurde, und die sich anhand der hybriden Visualisierung geeignet implementieren lassen müsste, ist die **virtuelle Endoskopie**. Ihr Nutzen besteht darin, Stenosen und Kalzifizierungen innerhalb von Gefäßen vorab zu lokalisieren, ohne chirurgische Eingriffe vornehmen zu müssen. Außerdem können derartige Anwendungen zur Operationsplanung oder zu Übungszwecken mit endoskopischen Geräten eingesetzt werden [SHN+06].

Im hybriden System profitiert eine virtuelle Endoskopie von der beliebig hoch einstellbaren Auflösung der indirekten Oberflächendaten ebenso wie von der Darstellung umliegenden Gewebes durch Raycasting mit multidimensionalen Transferfunktion. Die integrierte Oberfläche könnte die Orientierung innerhalb der Gefäßinnenansicht erleichtern und gleichzeitig den Fokus auf die Gefäßwand richten. Es ist anzunehmen, dass diese Anwendung aufgrund der großen Nähe zwischen Kamera und Visualisierung nur dann sehr gute Ergebnisse liefert, wenn die Oberflächendarstellung die direkte Volumenvisualisierung bereits an vielen Stellen gut annähert. Andernfalls ist die Entstehung von sichtbaren Brüchen zwischen den beiden Darstellungsformen möglich. Die Eignung des hybriden Renderingsystems für virtuelle Endoskopie und die tatsächliche Wichtigkeit einer hohen Übereinstimmung zwischen den kombinierten Datenstrukturen ist anhand einer entsprechenden Implementierung und der Evaluation von Worst–Case–Szenarien im Vergleich zu Average–Case–Szenarien zu validieren.

Bei der Konzipierung und Implementierung des hybriden Systems wurde bewusst Wert auf **Modularität** gelegt. Damit können für den hybriden Renderer in einem bestehenden System vorhandene Implementierungen von indirekten und direkten Volumenvisualisierungsmethoden beliebig kombiniert werden, solange sie ihr Ergebnis in Form von eines Vertex Buffer Objects für das Rendering bereitstellen beziehungsweise in ein Framebuffer Object rendern. Welche Darstellungsmöglichkeiten neben der in dieser Arbeit bereits präsentierten Umsetzung für hybrides Rendering geeignet sind, muss anhand entsprechender Tests validiert werden. Es ist anzunehmen, dass MPU Implicits und ein geglätteter Marching Cubes als indirekte Visualisierungskomponente gute Ergebnisse liefern können. Auch das Raycasting könnte beispielsweise durch Shear–Warp Volume Rendering ersetzt werden.

Stehen unterschiedliche Algorithmen in einem System für die hybride Kombination zur Wahl, ist es möglich dem Benutzer auf Basis einer Analyse der verfügbaren Hardwareleistung Renderingmethoden vorzuschlagen oder Warnungen vor längeren Berechnungszeiten auszuwerfen. Insgesamt könnte eine Vielzahl unterschiedlicher hybrider Visualisierungen von realistischen Bildern bis zu NPR–Darstellungen abgedeckt werden, sodass in einem einzelnen System mehrere Zielgruppen und Anwendungsfälle berücksichtig werden.

Im Rahmen der Experten– und Laienbefragung konnten die Patientenaufklärung sowie die Aus– und Weiterbildung als geeignete **Anwendungsgebiete** für das entwickelte hybride System identifiziert werden. Für den Einsatz innerhalb der Patientenaufklärung erwies sich eine kombinierte Darstellung mit CT–Bildern als sinnvolle Lösung, um Krankheitsbilder und Therapiemaßnahmen zu erläutern. Anhand der Testergebnisse ist zu erwarten, dass die hybride Visualisierung Patienten einen schnelleren Zugang zu den dargestellten Informationen bieten und damit unnötige Verängstigung angesichts der abstrakten CT–Bilder vermeiden kann. Gleichzeitig könnte das CT–Bild die Darstellung durch zusätzliche Details ergänzen und die korrekte Einschätzung des Gesundheitszustands unterstützen.

In der Aus– und Weiterbildung ist eine Nutzung der hybriden Visualisierung für realistische Simulationen zu Übungszwecken denkbar. Vor allem der Umgang mit endoskopischen Werkzeugen sollte nach Integration der benötigten virtuellen Objekte trainiert werden können. Für angehende Radiologen besteht die Möglichkeit durch unterstützenden Einsatz der hybriden Visualisierung in der Lehre die Zusammensetzung von CT–Bildern im Kopf schneller zu erlernen.

Der in dieser Arbeit entwickelte erste Prototyp des hybriden Systems war laut Testergebnissen noch nicht für die Diagnoseunterstützung und nur teilweise für die Therapieplanung geeignet. Zwar wird für Letzteres bereits die benötigte genaue Abbildung der Topologie geliefert – inklusive der Darstellung schmaler Strukturen –, allerdings fehlen bisher notwendige Informationen über die Wandbeschaffenheit. Die Umsetzung der in Abschnitt 7.1 diskutierten Optimierungsmöglichkeiten könnte die hybride Visualisierung in kombinierter Darstellung mit den CT–Daten auch für die Diagnoseunterstützung und für einen intraoperativen Einsatz interessant machen. Sehr wahrscheinlich ist dann eine gute Nutzbarkeit in der Therapieplanung. Um diese Einschätzung zu validieren, sind nach erfolgreicher Implementierung der genannten Maßnahmen neue Tests und Befragungen notwendig.

Wird die hybride Visualisierung, wie oben ausgeführt, in ein bestehendes System integriert, bietet sie die Möglichkeit, diverse Daten in einer einzigen Anwendung zu sammeln und kombiniert darzustellen. Auf diese Weise sollte Medizinern ein vereinfachter Zugang zu Messwerten, CT–Bildern, der Oberflächenbeschaffenheit oder räumlichen Zusammenhängen geboten werden können. Dabei ist zu vermuten, dass das Verfahren nicht nur für die Betrachtung von aortalen Aneurysmen oder anderen Gefäßerkrankungen geeignet ist. Auch für die Darstellung anderer Organe, beispielsweise Bronchialbäume, könnte eine hybride Visualisierung zufriedenstellende Ergebnisse erzielen.

8 Fazit

Die dreidimensionale Darstellung vaskulärer Strukturen findet in der Diagnose, der Therapiekontrolle und der Planung chirurgischer Eingriffe zunehmend Anwendung. Dazu werden CT– oder MRT–Daten gegebenenfalls auf Basis vorab durchgeführter Segmentierungen interessierender Objekte durch entsprechende Renderingalgorithmen verarbeitet. Geeignete Systeme müssen anhand der zumeist großen Datengrundlage hochaufgelöste, genaue Visualisierungen in Echtzeit berechnen können. Gängige Methoden des direkten und indirekten Volumenrenderings, wie beispielsweise Raycasting beziehungsweise MPU–Implicits, sind nicht in der Lage alle Anforderungen gleichermaßen zuverlässig zu erfüllen. Während direkte Volumenvisualisierungen in ihrer Auflösung begrenzt sind, erzeugen indirekte Verfahren über die Zwischenrepräsentation als Polygonnetz ungenauere Ergebnisse, die nur unter hohem Rechen– und Parametrisierungsaufwand optimiert werden können. Hybride Verfahren, die unterschiedliche Visualisierungsansätze und ihre jeweiligen Vorteile miteinander verbinden, versprechen sowohl eine verbesserte Darstellungsqualität als auch eine erhöhte Performanz.

Das Ziel dieser Arbeit war es, durch Kombination von direkten und indirekten Volumenvisualisierungsalgorithmen die dreidimensionale Darstellung vaskulärer Strukturen zu optimieren. Auf Basis einer Analyse potentiell geeigneter Methoden zur Oberflächengenerierung wurden Marching Cubes und MPU Implicits als geeignete Verfahren für hybrides Rendering identifiziert. Aufgrund des Performanzvorteils wurde der Marching Cubes Algorithmus für die Konzeption und Umsetzung eines hybriden Systems verwendet, das ein Polygonnetz unter Beachtung einer maximal erlaubten Tiefendifferenz in die direkte Volumenvisualisierung eines Raycasters integriert. Die Implementierung nutzt moderne (GP)GPU–Techniken, um eine interaktive Benutzung bei gleichzeitig hoher Bildqualität zu ermöglichen. Die Ergebnisse der Arbeit zeigen das Potential des prototypischen Systems für einen Einsatz im klinischen Umfeld. Sowohl eine Experten– als auch eine Laienbefragung konnten die Eignung zur Patientenaufklärung und darüber hinaus für die Aus– und Weiterbildung von Medizinern belegen. Das System stellt schmale Blutgefäße, den Gefäßquerschnitt sowie topologische Verzweigungen korrekt dar. Die Möglichkeit zur Kontrolle des maximalen Bildfehlers mithilfe der variabel zur Laufzeit einstellbaren maximal erlaubten Tiefendifferenz bildet die Basis für eine Anwendung in der Therapieplanung und zur intraoperativen Nutzung. Eine Weiterentwicklung des hybriden Systems mit Fokus auf die Darstellung der Gefäßwandbeschaffenheit könnte sogar die Verwendung zur Diagnoseunterstützung ermöglichen.

Ausblick. Das entwickelte hybride Renderingsystem erlaubt durch seinen modularen Aufbau eine beliebige Verwendung unterschiedlicher Visualisierungskomponenten und erscheint damit auch zur Integration in bestehen-

de Systeme besonders geeignet. In diesem Zusammenhang ist in weiterführenden Untersuchungen zu testen, welche Ergebnisse unter Einsatz anderer Oberflächenalgorithmen, wie beispielsweise MPU Implicits, oder durch eine Glättung des Marching Cubes Ergebnisses erzielt werden können. Eine Optimierung des implementierten Raycasters durch Empty Space Skipping sollte Echtzeitfähigkeit auch auf rechenschwächeren Geräten ermöglichen. Für hochleistungsfähige Hardware ist daneben die Umsetzung photorealistischer Beleuchtungsmethoden zu einer weiteren Verbesserung der Tiefenwahrnehmung sinnvoll.

Von besonderer Wichtigkeit für zukünftige Arbeiten ist die Integration multidimensionaler Transferfunktionen in den Raycasting–Algorithmus. Unter Hinzunahme von Transparenzen sollten sie die hybride Darstellung um Informationen über die Gefäßwand und gegebenenfalls vorliege Gewebeveränderungen anreichern können. Auch für die Segmentierung sind exakte Berechnungsmethoden notwendig. Werden diese beiden Eigenschaften erfüllt, könnte das hybride System eine verbesserte Genauigkeit bei gleichzeitig höherem Oberflächenanteil in der Volumenvisualisierung ermöglichen.

Literatur

[AMHH08] Akenine-Möller, T. ; Haines, E. ; Hoffman, N. ; 3 (Hrsg.): *Real-time rendering*. A K Peters, 2008

[Bau72] Baumgart, B. G.: Winged edge polyhedron representation / DTIC Document. 1972. – Forschungsbericht

[Bli82] Blinn, J. F.: A generalization of algebraic surface drawing. In: *ACM Transactions on Graphics (TOG)* 1 (1982), Nr. 3, S. 235–256

[BRB05] Bornik, A. ; Reitinger, B. ; Beichel, R.: Reconstruction and representation of tubular structures using simplex meshes. (2005)

[BRMR01] Betz, E. ; Reutter, K. ; Mecke, D. ; Ritter, H.: *Biologie des Menschen*. 15. Nikol Verlag, 2001

[BS91] Bloomenthal, J. ; Shoemake, K.: Convolution surfaces. In: *ACM SIGGRAPH Computer Graphics* Bd. 25 ACM, 1991, S. 251–256

[Buz04] Buzug, T. M.: *Einführung in die Computertomographie: Mathematisch-physikalische Grundlagen der Bildrekonstruktion*. Springer, 2004

[Eck] Eckstein, H. H. ; Klinik und Hochschulambulanz für Gefässchirurgie, Klinikum rechts der Isar, Technische Universität München (Hrsg.): *Erkrankungen der Aorta*. Klinik und Hochschulambulanz für Gefäßchirurgie, Klinikum rechts der Isar, Technische Universität München

[EDKS94] Ehricke, H. H. ; Donner, K. ; Koller, W. ; Strasser, W.: Visualization of vasculature from volume data. In: *Computers & Graphics* 18 (1994), Nr. 3, S. 395–406

[EHK+06] Engel, K. ; Hadwiger, M. ; Kniss, J. M. ; Rezk-Salama, C. ; Weiskopf, D.: *Real-time volume graphics*. A K Peters, 2006

[FDFH97] Foley, J. D. ; Dam, A. van ; Feiner, S. K. ; Hughes, J. F.: *Computer graphics: principles and practice*. 2. Addison-Wesley, 1997

[FFKW02] Felkel, P. ; Fuhrmann, A. ; Kanitsar, A. ; Wegenkittl, R.: Surface reconstruction of the branching vessels for augmented reality aided surgery. In: *BIOSIGNAL 2002* 16 (2002), S. 252–254

[Fie] Fiedler, G.: *Gafferongames, tessellating the go stone: triangle subdivision.* `http://gafferongames.com/virtualgo/tessellating-the-go-stone/`, Abruf: 15. Oktober 2014

[Fos02] Fosanelli, Sandra: *Jahresarbeit über Bewegungsstörungen, Spastik, Athetose, Ataxie und Alltagshilfen.* 2002

[FS08] Faller, A. ; Schünke, M.: *Der Körper des Menschen: Einführung in Bau und Funktion.* 15. Georg Thieme Verlag, 2008

[Gib98] Gibson, S. F. F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented datalastic Surface Nets: Generating Smooth Surfaces from Binary Segmented Data. In: *Medical Image Computing and Computer-Assisted Interventation—MICCAI'98.* Springer, 1998, S. 888–898

[GKS+93] Gerig, G. ; Koller, T. ; Székely, G. ; Brechbühler, C. ; Kübler, O.: Symbolic description of 3-D structures applied to cerebral vessel tree obtained from MR angiography volume data. In: *Information processing in medical imaging* Springer, 1993, S. 94–111

[GKW08] Gussmann, A. ; Kühn, J. ; Weise, U. ; Deutschen Gesellschaft für Gefässchirurgie (Hrsg.): *Leitlinien zum Bauchaortenaneurysma und Beckenarterienaneurysma.* Deutschen Gesellschaft für Gefäßchirurgie, August 2008

[HBH03] Hadwiger, M. ; Berger, C. ; Hauser, H.: High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In: *Visualization, 2003. VIS 2003. IEEE* IEEE, 2003, S. 301–308

[HMIBG01] Hauser, H. ; Mroz, L. ; Italo Bischi, G. ; Groller, E.: Two-level volume rendering. In: *Visualization and Computer Graphics, IEEE Transactions on* 7 (2001), Nr. 3, S. 242–252

[HPSP01] Hahn, H. K. ; Preim, B. ; Selle, D. ; Peitgen, H. O.: Visualization and interaction techniques for the exploration of vascular structures. In: *Visualization, 2001. VIS'01. Proceedings* IEEE, 2001, S. 395–578

[Huc07] Huch, R.: *Mensch Körper Krankheit.* 5. Urban & Fischer, 2007

[KGJ10] Koeppel, T. A. ; Greiner, A. ; Jacobs, M. J. ; Europäisches Gefässzentrum Aachen-Maastricht (Hrsg.): *Thorakale und thorakoabdominelle Aortenaneurysmen.* Europäisches Gefäßzentrum Aachen-Maastricht, September 2010

[KGNP12] Kubisch, C. ; Glasser, S. ; Neugebauer, M. ; Preim, B.: Vessel visualization with volume rendering. In: *Visualization in Medicine and Life Sciences II.* Springer, 2012, S. 109–132

[Köc05] Köchy, K.: *Patientenorientierte interaktive Visualisierung medizinischer Bilddaten.*, Berlin Institute of Technology, Diss., 2005

[KPKS10] Klinke, R. ; Pape, H. C. ; Kurtz, A. ; Silbernagl, S.: *Physiologie.* 6. Georg Thieme Verlag, 2010

[KS87] Kaufman, A. ; Shimony, E.: 3D scan-conversion algorithms for voxel-based graphics. In: *Proceedings of the 1986 workshop on Interactive 3D graphics* ACM, 1987, S. 45–75

[LC87] Lorensen, W. E. ; Cline, H. E.: Marching cubes: a high resolution 3D surface construction algorithm. In: *ACM Siggraph Computer Graphics* Bd. 21 ACM, 1987, S. 163–169

[Lev90a] Levoy, M.: Efficient ray tracing of volume data. In: *ACM Transactions on Graphics (TOG)* 9 (1990), Nr. 3, S. 245–261

[Lev90b] Levoy, M.: A hybrid ray tracer for rendering polygon and volume data. In: *Computer Graphics and Applications, IEEE* 10 (1990), Nr. 2, S. 33–40

[LHH08] Linsen, L. ; Hagen, H. ; Hamann, B.: *Visualization in medicine and life sciences.* Springer, 2008

[LM02] Lum, E. B. ; Ma, K.-L.: Hardware-accelerated parallel non-photorealistic volume rendering. In: *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* ACM, 2002, S. 67–ff

[LW90] Levoy, M. ; Whitaker, R.: Gaze-directed volume rendering. In: *ACM SIGGRAPH Computer Graphics* 24 (1990), Nr. 2, S. 217–223

[MMY+08] Museth, K. ; Möller, T. ; Ynnerman, A. u. a.: Model-free surface visualization of vascular trees. (2008)

[MWG00] Mroz, L. ; Wegenkittl, R. ; Groller, E.: Mastering interactive surface rendering for Java-based diagnostic applications. In: *Visualization 2000. Proceedings* IEEE, 2000, S. 437–440

[Nie04] Nielson, G. M.: Dual marching cubes. In: *Proceedings of the conference on Visualization'04* IEEE Computer Society, 2004, S. 489–496

[NY06] Newman, T. S. ; Yi, H.: A survey of the marching cubes algorithm. In: *Computers & Graphics* 30 (2006), Nr. 5, S. 854–879

[OBA+05] Ohtake, Y. ; Belyaev, A. ; Alexa, M. ; Turk, G. ; Seidel, H. P.: Multi-level partition of unity implicits. In: *ACM SIGGRAPH 2005 Courses* ACM, 2005, S. 173

[Oel04] Oeltze, S.: *Visualisierung baumartiger anatomischer Strukturen mit Convolution Surfaces*, Otto-von-Guericke-Universität Magdeburg, Diss., Februar 2004

[OM95] Opalach, A. ; Maddock, S.C.: An overview of implicit surfaces. In: *Introduction to modelling and animation using implicit surfaces* (1995), S. 1–1

[OP05] Oeltze, S. ; Preim, B.: Visualization of vasculature with convolution surfaces: method, validation and evaluation. In: *Medical Imaging, IEEE Transactions on* 24 (2005), Nr. 4, S. 540–548

[PB07] Preim, B. ; Bartz, D.: *Visualization in medicine.* Elsevier, 2007

[PO08] Preim, B. ; Oeltze, S.: 3D visualization of vasculature: an overview. In: *Visualization in Medicine and Life Sciences.* Springer, 2008, S. 39–59

[POT] Preim, B. ; Oeltze, S. ; Tietjen, C.: Visualisierung für die bildbasierte Diagnostik und Therapieplanung / Otto-von-Guericke-Universität Magdeburg. – Forschungsbericht

[PP07] Putz, R. ; Pabst, R.: *Sobotta - Der komplette Atlas der Anatomie des Menschen in einem Band.* 22. Urban & Fischer, 2007

[Rob00] Robb, R. A.: *Biomedical imaging, visualization and analysis.* Wiley-Liss, 2000

[RSAH00] Rogalla, P. ; Scheltinga, J. T. ; Aschoff, A. J. ; Hamm, B.: *Virtual endoscopy and related 3D techniques.* Springer, 2000

[RSHSG00] Rezk-Salama, C. ; Hastreiter, P. ; Scherer, J. ; Greiner, G.: Automatic adjustment of transfer functions for 3D volume visualization. In: *VMV*, 2000, S. 357–364

[San13] Sandgren, J.: Transfer Time Reduction of Data Transfers between CPU and GPU. (2013)

[Sch05] Scharsach, H.: Advanced GPU raycasting. In: *Proceedings of CESCG* 5 (2005), S. 67–76

[Sch06a] Schumann, C.: *Visualisierung baumartiger anatomischer Strukturen mit MPU Implicits*, Otto-von-Guericke Universität Magdeburg, Diss., Juli 2006

[Sch06b] Schwegler, J.: *Der Mensch - Anatomie und Physiologie: Schritt für Schritt Zusammenhänge verstehen.* 4. Georg Thieme Verlag, 2006

[SEL12] Smistad, E. ; Elster, A. C. ; Lindseth, F.: Real-time surface extraction and visualization of medical images using OpenCL and GPUs. In: *Norsk informatikkonferanse* 2012 (2012)

[SHN+06] Scharsach, H. ; Hadwiger, M. ; Neubauer, A. ; Wolfsberger, S. ; Bühler, K.: Perspective isosurface and direct volume rendering for virtual endoscopy applications. In: *Proceedings of the Eighth Joint Eurographics/IEEE VGTC conference on Visualization* Eurographics Association, 2006, S. 315–322

[Sie] *Siemens medical solutions, Somatom Emotion und Somatom Definition AS.* `http://www.siemens.com/press/de/pressebilder/?press=/de/pressebilder/2010/imaging_it/him201003028-01.htm`, Abruf: 06. Oktober 2014

[SL07] Schmidt, R. F. ; Lang, F.: *Physiologie des Menschen.* 30. Springer, 2007

[SL11] Steffel, J. ; Lüscher, T. F.: *Herz-Kreislauf.* Springer, 2011

[Smi] Smistad, E.: *A GPU implementation of the Marching Cubes algorithm for extracting surfaces from volumes using OpenCL and OpenGL.* `https://github.com/smistad/GPU-Marching-Cubes`, Abruf: 24. Oktober 2014

[SOBP07] Schumann, C. ; Oeltze, S. ; Bade, R. ; Preim, B.: Visualisierung von Gefäßsystemen mit MPU Implicits. In: *Bildverarbeitung für die Medizin 2007.* Springer, 2007, S. 207–211

[ST90] Shirley, P. ; Tuchman, A.: *A polygonal approximation to direct scalar volume rendering.* ACM, 1990

[Tau95] Taubin, G.: A signal processing approach to fair surface design. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* ACM, 1995, S. 351–358

[TIP05] Tietjen, C. ; Isenberg, T. ; Preim, B.: Combining silhouettes, surface, and volume rendering for surgery education and planning. In: *Proceedings of the Seventh Joint Eurographics/IEEE VGTC conference on Visualization* Eurographics Association, 2005, S. 303–310

[UH00] Udupa, J. K. ; Herman, G. T.: *3D imaging in medicine*. 2. CRC Press, 2000

[VHST+04] Vega Higuera, F. ; Sauber, N. ; Tomandl, B. ; Nimsky, C. ; Greiner, G. ; Hastreiter, P.: Automatic adjustment of bidimensional transfer functions for direct volume visualization of intracranial aneurysmsidimensional Transfer Functions for Direct Volume Visualization of Intracranial Aneurysms. In: *Medical Imaging 2004* International Society for Optics and Photonics, 2004, S. 275–284

[VMM99] Vollmer, J. ; Mencl, R. ; Mueller, H.: Improved laplacian smoothing of noisy surface meshes. In: *Computer Graphics Forum* Bd. 18 Wiley Online Library, 1999, S. 131–138

[VP10] Viola, I. ; Parulek, J.: GPU-based visualization of convolution surfaces. In: *The Eurographics Association* (2010)

[VPBM01] Vlachos, A. ; Peters, J. ; Boyd, C. ; Mitchell, J. L.: Curved PN triangles. In: *Proceedings of the 2001 symposium on Interactive 3D graphics* ACM, 2001, S. 159–166

[YYM+13] Yamashita, O. ; Yoshimura, K. ; Morikage, N. ; Furutani, A. ; Hamano, K.: A novel treatment strategy for infected abdominal aortic aneurysms. In: *Aortic Aneurysm - Recent Advances* (2013), April

[ZTTS06] Ziegler, G. ; Tevs, A. ; Theobalt, C. ; Seidel, H. P.: On-the-fly point clouds through histogram pyramids. In: *Workshop on Vision, Modeling, and Visualization (VMV 2006)*, 2006, S. 137–144

Anhang

A Hinweise zum Source Code

Der vollständige Source Code wird zusammen mit einem Videobeispiel über GitHub bereitgestellt. Datensätze können aus urheberrechtlichen Gründen nicht zur Verfügung gestellt werden. Das Verzeichnis ist unter nachfolgendem Link zu erreichen:

https://github.com/martinasekulla/HybridRenderer

B Tastaturbelegung zur Navigation und zur Einstellung variabler Parameter

Eingabe	Funktion	Default
$\leftarrow$, $\rightarrow$	maximale Differenz zwischen Marching Cubes Oberfläche und direkter Volumendarstellung um ± 0.1 verändern	1.5
$\downarrow$, $\uparrow$	Isowert um ± 0.1 verändern	55.5
T	Tesselierung einschalten (*true*), bei Bewegung ausschalten (*move*) oder vollständig ausschalten (*false*)	false
V	Visualisierungsmodus zwischen direkter Volumendarstellung mittels Raycasting und hybridem Rendering wechseln	hybrid
O	Marching Cubes Oberfläche rendern	false
M	Oberflächenanteil im hybriden Rendering violett einfärben	false
ESC	Programm beenden	false

Tabelle 11: Tastaturbefehle zur Anpassung der Parameter im hybriden Renderingsystem.

Eingabe	Funktion
W, S	Kamera nach vorne bzw. nach hinten bewegen
A, D	Kamera in einem Kreis nach links bzw. nach rechts um das Objekt herum bewegen
Q, E	Kamera nach unten bzw. nach oben bewegen
Q + Shift, E + Shift	Kamera nach vorne bzw. nach hinten neigen
1, 2, 3	Geschwindigkeit verändern ($1\times$, $2\times$, $4\times$)

Tabelle 12: Tastaturbefehle zur Navigation der Kamera im hybriden Renderingsystem.

C Performanzergebnisse

	$256 \times 256 \times 389$ $(1.07031 \times 1.07031 \times 1.0)$			$512 \times 512 \times 389$ $(0.535156 \times 0.535156 \times 1.0)$		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
min FPS (H)	18	22	308	14	18	239
ms/Frame (H)	55,56	45,45	3,25	71,43	55,56	4,18
min FPS (R)	22	28	347	19	25	260
ms/Frame (R)	45,45	35,71	2,88	52,63	40,00	3,85
min FPS (M)	110	150	3206	62	99	1769
ms/Frame (M)	9,09	6,67	0,31	16,13	10,10	0,57
	Die Fensterauflösung bei allen Tests lag bei 1024×768 Pixeln.					

Tabelle 13: Performanzergebnisse für Datensatz 1 gemessen mit einer Distanz von 600 zwischen Kamera und Blickzentrum. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	$256 \times 256 \times 389$ $(1.07031 \times 1.07031 \times 1.0)$			$512 \times 512 \times 389$ $(0.535156 \times 0.535156 \times 1.0)$		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
min FPS (H)	10	15	190	9	12	178
ms/Frame (H)	100,00	66,67	5,26	111,11	83,33	5,62
min FPS (R)	13	17	204	11	15	195
ms/Frame (R)	76,92	58,82	4,90	90,91	66,67	5,13
min FPS (M)	106	139	3192	60	78	1725
ms/Frame (M)	9,43	7,19	0,31	16,67	12,82	0,58
	Die Fensterauflösung bei allen Tests lag bei 1024×768 Pixeln.					

Tabelle 14: Performanzergebnisse für Datensatz 1 gemessen mit einer Distanz von 400 zwischen Kamera und Blickzentrum. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	$256 \times 256 \times 389$ ($1.07031 \times 1.07031 \times 1.0$)			$512 \times 512 \times 389$ ($0.535156 \times 0.535156 \times 1.0$)		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
min FPS (H)	9	11	154	7	10	142
ms/Frame (H)	111,11	90,91	6,49	142,86	100,00	7,04
min FPS (R)	9	13	169	9	11	153
ms/Frame (R)	111,11	76,92	5,92	111,11	90,91	6,54
min FPS (M)	112	140	3185	62	83	1732
ms/Frame (M)	8,93	7,14	0,31	16,13	12,05	0,58
	Die Fensterauflösung bei allen Tests lag bei 1024×768 Pixeln.					

Tabelle 15: Performanzergebnisse für Datensatz 1 gemessen mit einer Distanz von 200 zwischen Kamera und Blickzentrum. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	$256 \times 256 \times 533$ ($0.957031 \times 0.957031 \times 0.700012$)			$512 \times 512 \times 533$ ($0.478516 \times 0.478516 \times 0.700012$)		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
min FPS (H)	15	17	287	11	13	147
ms/Frame (H)	66,67	58,82	3,48	90,91	76,92	6,80
min FPS (R)	20	25	333	17	21	166
ms/Frame (R)	50,00	40,00	3,00	58,82	47,62	6,02
min FPS (M)	59	78	1711	31	42	859
ms/Frame (M)	16,95	12,82	0,58	32,26	23,81	1,16
	Die Fensterauflösung bei allen Tests lag bei 1024×768 Pixeln.					

Tabelle 16: Performanzergebnisse für Datensatz 2 gemessen mit einer Distanz von 600 zwischen Kamera und Blickzentrum. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	$256 \times 256 \times 533$ (0.957031 × 0.957031 × 0.700012)			$512 \times 512 \times 533$ (0.478516 × 0.478516 × 0.700012)		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
min FPS (H)	9	12	188	8	9	138
ms/Frame (H)	111,11	83,33	5,32	125,00	111,11	7,25
min FPS (R)	12	15	210	10	13	157
ms/Frame (R)	83,33	66,67	4,76	100,00	76,92	6,37
min FPS (M)	56	66	1701	31	40	849
ms/Frame (M)	17,86	15,15	0,59	32,26	25,00	1,18
	Die Fensterauflösung bei allen Tests lag bei 1024×768 Pixeln.					

Tabelle 17: Performanzergebnisse für Datensatz 2 gemessen mit einer Distanz von 400 zwischen Kamera und Blickzentrum. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	$256 \times 256 \times 533$ (0.957031 × 0.957031 × 0.700012)			$512 \times 512 \times 533$ (0.478516 × 0.478516 × 0.700012)		
	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC	MacBook Air (Mid '12)	MacBook Air (Mid '13)	Windows-PC
min FPS (H)	6	10	131	5	7	110
ms/Frame (H)	166,67	100,00	7,63	200,00	142,86	9,09
min FPS (R)	7	10	143	7	9	115
ms/Frame (R)	142,86	100,00	6,99	142,86	111,11	8,70
min FPS (M)	59	78	1697	30	42	859
ms/Frame (M)	16,95	12,82	0,59	33,33	23,81	1,16
	Die Fensterauflösung bei allen Tests lag bei 1024×768 Pixeln.					

Tabelle 18: Performanzergebnisse für Datensatz 2 gemessen mit einer Distanz von 200 zwischen Kamera und Blickzentrum. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	Datensatz 1		Datensatz2	
	256×256×389	512×512×389	256×256×389	512×512×389
min FPS (H)	437	251	319	159
ms/Frame (H)	2,29	3,98	3,13	6,29
min FPS (R)	504	286	332	171
ms/Frame (R)	1,98	3,50	3,01	5,85
min FPS (M)	3385	1785	1758	872
ms/Frame (M)	0,30	0,56	0,57	1,15
	Alle Tests wurden unter Windows mit einer Distanz zwischen Kamera und Blickzentrum von 600 ausgeführt.			

Tabelle 19: Performanzergebnisse bei einer Fenstergröße von 800×600. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	Datensatz 1		Datensatz2	
	256×256×389	512×512×389	256×256×389	512×512×389
min FPS (H)	308	239	287	147
ms/Frame (H)	3,25	4,18	3,48	6,80
min FPS (R)	347	260	333	166
ms/Frame (R)	2,88	3,85	3,00	6,02
min FPS (M)	3206	1769	1711	859
ms/Frame (M)	0,31	0,57	0,58	1,16
	Alle Tests wurden unter Windows mit einer Distanz zwischen Kamera und Blickzentrum von 400 ausgeführt.			

Tabelle 20: Performanzergebnisse bei einer Fenstergröße von 1024×768. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

	Datensatz 1		Datensatz2	
	256×256×389	512×512×389	256×256×389	512×512×389
min FPS (H)	178	159	175	115
ms/Frame (H)	5,62	6,29	5,71	8,70
min FPS (R)	195	180	209	132
ms/Frame (R)	5,13	5,56	4,78	7,58
min FPS (M)	2860	1623	1604	828
ms/Frame (M)	0,35	0,62	0,62	1,21
	Alle Tests wurden unter Windows mit einer Distanz zwischen Kamera und Blickzentrum von 400 ausgeführt.			

Tabelle 21: Performanzergebnisse bei einer Fenstergröße von 1440×1080. Gemessen werden die Framerate und die Framezeit des hybriden Renderingsystems (H) im Vergleich zu Raycasting (R) und Marching Cubes (M).

D Expertenfragebogen

Frage 1: *Bitte bewerten Sie die hybride Visualisierung nach folgenden Kriterien:*

	sehr gut	gut	mittel	schlecht
Verständlichkeit der Darstellung	❍	❍	❍	❍
Anwendbarkeit im klinischen Alltag	❍	❍	❍	❍
Gesamteindruck	❍	❍	❍	❍

Frage 2: *Welche Anwendungsmöglichkeiten sehen Sie für die hybride Visualisierung? (Mehrfachantworten möglich)*

❍ Diagnose ❍ Therapieplanung ❍ Patientenaufklärung ❍ Ausbildung ❍ keine
❍ andere: _____

Frage 3: *Welche Vorteile bietet die hybride Visualisierung Ihrer Meinung nach?*

_____

_____

_____

_____

Frage 4: *Welche Probleme weist die hybride Visualisierung Ihrer Meinung nach auf?*

_____

_____

_____

_____

Frage 5: *Welche Verbesserungen und Erweiterungen würden Sie sich für die hybride Visualisierung wünschen?*

❍ geglättete Gefäßoberfläche
❍ veränderte Färbung, z.B.: _____

_____

❍ Möglichkeit der Darstellung aller umliegenden Organe
❍ Möglichkeit der Durchführung einer virtuellen Endoskopie
❍ andere: _____

_____

_____

Frage 6: *Welche Visualisierungsform würden Sie bevorzugen?*

❍ Hybride Visualisierung ❍ Volumenvisualisierung ❍ keine 3D-Visualisierung

Informationen zur Person

Alter	❍ <30	❍ 30 - 49	❍ >50
Geschlecht	❍ weiblich	❍ männlich	
Arztgruppe	❍ Innere Medizin	❍ Radiologie	❍ Gefäßchirurgie
	❍ andere: _____		

E Laienfragebogen

Frage 1: *Wie bewerten Sie die Verständlichkeit der unterschiedlichen Darstellungen?*

CT-Bild	selbsterklärend	❍	❍	❍	❍	❍	❍	unverständlich
Volumenvisualisierung	selbsterklärend	❍	❍	❍	❍	❍	❍	unverständlich
Hybride Visualisierung	selbsterklärend	❍	❍	❍	❍	❍	❍	unverständlich

Frage 2: *Wie schätzen Sie den Gesundheitszustand des Patienten in den unterschiedlichen Darstellungen ein?*

CT-Bild	gut	❍	❍	❍	❍	❍	❍	lebensbedrohlich
Volumenvisualisierung	gut	❍	❍	❍	❍	❍	❍	lebensbedrohlich
Hybride Visualisierung	gut	❍	❍	❍	❍	❍	❍	lebensbedrohlich

Frage 3: *Angenommen Sie seien der Patient: Inwiefern fühlen Sie sich von den unterschiedlichen Darstellungen eingeschüchtert oder verängstigt?*

CT-Bild	sehr verängstigt	❍	❍	❍	❍	❍	❍	gar nicht
Volumenvisualisierung	sehr verängstigt	❍	❍	❍	❍	❍	❍	gar nicht
Hybride Visualisierung	sehr verängstigt	❍	❍	❍	❍	❍	❍	gar nicht

Frage 4: *Welche Visualisierungsform würden Sie in einem Patientenaufklärungsgespräch bevorzugen? Bitte begründen Sie kurz Ihre Antwort.*

❍ CT-Bild ❍ Volumenvisualisierung ❍ Hybride Visualisierung

Frage 5: *Welche Verbesserungen und Erweiterungen würden Sie sich für die hybride Visualisierung wünschen?*

❍ geglättete Gefäßoberfläche
❍ veränderte Färbung, z.B.:

❍ Möglichkeit der Darstellung aller umliegenden Organe
❍ andere:

Informationen zur Person

Alter ❍ <30 ❍ 30 - 49 ❍ >50
Geschlecht ❍ weiblich ❍ männlich

Haben Sie bereits Erfahrung mit Aortenaneurysmen? (z.B. durch eigene Erkrankung oder Erkrankungen im Bekanntenkreis?) ❍ *ja* ❍ *nein*

Wie schätzen Sie Ihre PC-Kenntnisse ein?
❍ sehr gute Kenntnisse ❍ gute Kenntnisse ❍ erste Kenntnisse ❍ keine Kenntnisse

Wie schätzen Sie Ihre Erfahrung mit 3D-Anwendungen ein?
❍ sehr gute Erfahrung ❍ gute Erfahrung ❍ erste Erfahrung ❍ keine Erfahrung