Otto-von-Guericke-Universität Magdeburg
Institut für Simulation und Graphik
der Fakultät für Informatik

Visualisierung baumartiger anatomischer Strukturen mit Convolution Surfaces

Diplomarbeit

von: Steffen Oeltze

1. Gutachter: Prof. Dr.-Ing. Bernhard Preim
2. Gutachter: Prof. Dr. Klaus-Dietz Tönnies

Bearbeitungszeitraum: 11.06.2003 - 13.02.2004

Zusammenfassung

Diese Arbeit präsentiert eine Methode zur Visualisierung baumartiger anatomischer Strukturen, wie Blutgefäßsysteme und Bronchialbäume, basierend auf klinischen CT- oder MR-Daten. Das Gefäßskelett sowie die Durchmesserinformation pro Skelettvoxel dienen als Eingabe. Die Methode generiert, unter Beachtung dieser Daten, weiche Übergänge an den Verzweigungen und geschlossene, abgerundete Gefäßenden mit Hilfe von *Convolution Surfaces*. Ähnlich einiger verwandter Arbeiten basiert auch diese Methode auf der Annahme kreisrunder Gefäßquerschnitte. Im Gegensatz zu anderen Visualisierungsverfahren, welche sich auf die explizite Beschreibung einer Geometrie stützen, werden hier implizite Oberflächen eingesetzt, um qualitativ hochwertige Visualisierungen zu erreichen. Die Methode wurde auf eine große Anzahl von Gefäßstrukturen angewandt und liefert gute Resultate in einer angemessenen Zeit durch den effizienten Einsatz von Hüllkörpern *bounding volumes*. Sie ist für den Einsatz in der Therapieplanung und in Ausbildungssystemen bestimmt.

This work presents a method for visualizing tree-like anatomic structures, such as vasculature and bronchial trees based on clinical CT- or MR data. The vessel skeleton as well as the diameter information per skeleton voxel serve as input. The method adheres to these data, while producing smooth transitions at branchings and closed, rounded ends by means of convolution surfaces. Similar to related work the method is based on the assumption of a circular cross-section of vasculature. In contrast to other authors who relied on the explicit description of the geometry implicit surfaces are employed here to achieve high quality visualization. The method has been applied to a large number of vessel trees and produces good results in a reasonable time which is due to the efficient use of bounding volumes. It is intended for use in therapy planning and educational systems.

Selbstständigkeitserklärung

Hiermit versichere ich, Steffen Oeltze (Matrikel-Nr. 154211), die vorliegende Arbeit allein und nur unter Verwendung der angegebenen Quellen angefertigt zu haben.

Steffen Oeltze

Danksagung

Mein Dank gilt in erster Linie meinem Betreuer Prof. Dr. Bernhard Preim, der mich während dieser Arbeit nicht nur hervorragend fachlich unterstützt hat, sondern mir auch auch in weniger produktiven Phasen stets motivierend zur Seite stand. Zahlreiche fruchtbare Diskussionen und Hinweise haben zur Entstehung dieser Arbeit beigetragen. Weiterhin möchte ich mich bei dem Centrum für medizinische Diagnosesysteme und Visualisierung (MeVis) für die Überlassung aller in dieser Arbeit verwendeten Datensätze bedanken. Besonders danke ich Olaf Konrad-Verse, Milo Hindennach und Dr.med. Holger Bourquain für ihre Ideen und Anregungen zur Gefäßvisualisierung. Meine Arbeit mit impliziten Oberflächen wurde durch die wertvollen Ratschläge von Prof. Jules Bloomenthal, Dr. Andrei Sherstyuk und Alexis Angelidis bereichert. Ich bin Christian Puck Tietjen tief verbunden für seine Unterstützung bei der Programmierung mit OpenInventor und der Softwareplattform ILAB. Nicht zuletzt möchte ich mich bei meinen Eltern bedanken, die mich während des gesamten Studiums uneingeschränkt unterstützt haben und immer wieder aufmunternde Worte fanden.

Steffen Oeltze, Februar 2004

Inhaltsverzeichnis

1 Einleitung **1**

2 Gefäßvisualisierung - Ein Überblick **5**
2.1 Konventionelle Techniken der Gefäßvisualisierung 5
2.1.1 Darstellungen in 2D 6
2.1.2 Darstellungen in $2\frac{1}{2}$D 6
2.1.3 Darstellungen in 3D 8
2.2 Rekonstruktion eines Gefäßmodells 10
2.2.1 Segmentierung 10
2.2.2 Skelettierung 11
2.2.3 Graphenanalyse 12
2.3 Modellbasierte Gefäßvisualisierung 13
2.3.1 Gefäßvisualisierung mit Zylindern 14
2.3.2 Gefäßvisualisierung mit Kegelstümpfen 16
2.3.3 Gefäßvisualisierung mit Freiformflächen 17
2.3.4 Gefäßvisualisierung mit Subdivision Surfaces 18
2.3.5 Direkte Visualisierung der symbolischen Gefäßbeschreibung 19
2.4 Zusammenfassung 20

3 Implizite Oberflächen **23**
3.1 Geschichte und Anwendungen 24
3.2 Modellierung 25
3.2.1 Punktbasierte Modellierung 26
3.2.2 Skelettbasierte Modellierung 27
3.2.3 Bulging 31
3.2.4 Unwanted Blending 34
3.3 Convolution Surfaces im Detail 36
3.3.1 Filterwahl 37
3.3.2 Durchmesserverlauf 39
3.4 Visualisierung 46
3.4.1 Ray Tracing 47
3.4.2 Polygonisierung 49
3.4.3 Weitere Verfahren 55
3.5 Konsequenzen für den Entwurf 57

4 Entwurf **59**
4.1 Erste Visualisierungstests 59
4.1.1 Testergebnisse 60

4.2 Blending . . . 64
4.2.1 Blendingstärke . . . 64
4.2.2 Bulging . . . 69
4.2.3 Unwanted Blending . . . 77
4.3 Effektive Polygonisierung von Convolution Surfaces . . . 81
4.3.1 Entwurf einer Datenstruktur für die effiziente Polygonisierung . . . 81
4.3.2 Hüllkörper . . . 85
4.4 Generierung eines geometrischen Modells . . . 91
4.4.1 Parameter „Würfelgröße“ . . . 91
4.4.2 Parameter „Startpunkt“ . . . 92
4.4.3 Parameter „Iterationstiefe“ . . . 93
4.4.4 Parameter „Zelltyp“ . . . 94
4.4.5 Parameter „Ausbreitungsgrenze“ . . . 94
4.4.6 Oberflächennormalen . . . 94
4.5 Interaktion . . . 94
4.5.1 Selektion von Teilbäumen . . . 95
4.5.2 Einfärbung der Gefäßstruktur . . . 96
4.5.3 Transparenz und Gefäßskelett . . . 96

5 Realisierung und Resultate **99**
5.1 Programmierwerkzeuge . . . 99
5.1.1 Die 3D-Graphikbibliothek OpenInventor . . . 100
5.1.2 Die Softwareplattform ILAB . . . 101
5.2 Vorverarbeitung . . . 102
5.2.1 Hüllkörper . . . 104
5.3 Generierung eines geometrischen Modells . . . 109
5.3.1 Filterberechnung . . . 110
5.3.2 Polygonisierung und Vorbereitung einer kantenbasierten Interaktion . . . 110
5.3.3 Darstellung des polygonalen Modells mit OpenInventor . . . 112
5.4 Interaktion . . . 114
5.5 Resultate . . . 115

6 Validierung und Evaluierung **121**
6.1 Validierung . . . 121
6.2 Evaluierung . . . 123

7 Zusammenfassung und Ausblick **127**

Abbildungsverzeichnis **131**

Literaturverzeichnis **135**

A Anhang **141**
A.1 Gefäßanalysedaten . . . 141
A.2 Evaluierungsbogen . . . 142
A.2.1 Lebergefäßbaum 1 (G1) . . . 143
A.2.2 Lebergefäßbaum 2 (G2) . . . 144
A.2.3 Lebergefäßbaum 3 (G3) . . . 145

1 Einleitung

Sowohl für die präoperative Eingriffsplanung als auch für die medizinische Ausbildung ist die Visualisierung von Gefäßbäumen von großem Interesse. Hierbei stehen ein Verständnis der relativen Lage zu krankhaften Veränderungen, Topologie und Morphologie der Strukturen im Vordergrund. Neue klinische Fragestellungen erfordern eine strukturelle Analyse der Gefäßstruktur [GKS+93]. Hierzu zählen die Bestimmung von Teilbäumen, das Erkennen von Verzweigungstypen, die Vermessung des Durchmesserverlaufs entlang eines Gefäßes, Distanzmessungen und quantitative Vergleiche von mehreren Gefäßstrukturen. Traditionelle Verfahren, wie *Isosurface-Rendering* und *Maximum Intensity Projection* (MIP), sind zur Erfüllung dieser Zielstellungen nicht geeignet. Aufgrund von Bildrauschen, Partialvolumeneffekt und der begrenzten Auflösung von Computertomographie (CT) und Magnetresonanzbildgebung (MRI), erzeugen diese Verfahren eine artefaktbehaftete Darstellung. Weiterhin ist die visuelle Separation von kontrastverstärkten Gefäßstrukturen und anderen Strukturen mit hoher Signalintensität, wie Knochen, oft nur eingeschränkt möglich. Für Ausbildungszwecke und die Therapieplanung sollte daher ein Modell der Gefäßstruktur, basierend auf den Patientendaten, rekonstruiert werden.

Pionierarbeit auf dem Gebiet der modellbasierten Gefäßvisualisierung leisteten [GKS+93], welche Gefäßabschnitte durch Zylinder mit einem aus der Gefäßdicke berechneten Durchmesser darstellten. Eine lokale Gefäßverjüngung kann durch diese Methode jedoch nicht abgebildet werden. Weiterhin sind abrupte Übergänge und Diskontinuitäten der Oberflächennormalen vor allem an Verzweigungen zu beobachten. Darauffolgende Arbeiten konzentrierten sich hauptsächlich auf die korrekte Durchmesserabbildung entlang der gesamten Gefäßstruktur [HPSP01] und auf die Modellierung pathologischer Gefäßveränderungen [PTN97]. Nur wenige Arbeiten hingegen wurden der Generierung weicher, organisch wirkender Übergänge zwischen einzelnen Gefäßabschnitten gewidmet [EDKS94], [FFKW02]. Die Vermeidung von Diskontinuitäten entlang der Gefäßoberfläche erfordert speziell an Verzweigungen einen beträchtlichen Konstruktionsaufwand und zählt zu den schwierigsten Herausforderungen an die modellbasierte Gefäßvisualisierung. Verschiedene klinische Aufgaben wie die Definition eines Sicherheitsrandes bei der Tumorresektion oder die Vermessung einzelner Gefäßäste erfordern eine Nahansicht der interessierenden Gefäßteile. Hier sollte der Betrachter nicht durch Diskontinuitäten entlang der Gefäßoberfläche abgelenkt werden (Abb. 1.1).

Innerhalb dieser Arbeit wird von Bildanalyseergebnissen ausgegangen, die einen Gefäßbaum durch das Skelett und einer Durchmesserinformation für jedes Voxel beschreiben [SPSP02]. Das Gefäßskelett dient als Eingabe für den Visualisierungsprozess. Es liegt als gerichteter Graph vor [SPSP02]. Die Kanten dieses Graphen sind durch Liniensegmente approximiert, welche die Mittelpunkte benachbarter Skelettvoxel miteinander verbinden. Jedes Segment ist vollständig durch seine zwei Endpunkte und die dort assoziierten Radien beschrieben.

Ziel der Arbeit ist die Entwicklung einer Visualisierungsmethode, welche die Gefäßoberfläche auf der Basis dieser Information und der Modellannahme von kreisrunden Querschnitten nichtpathologischer Gefäße [GKS+93] rekonstruiert. Dabei wird eine korrekte Abbildung des Durch-

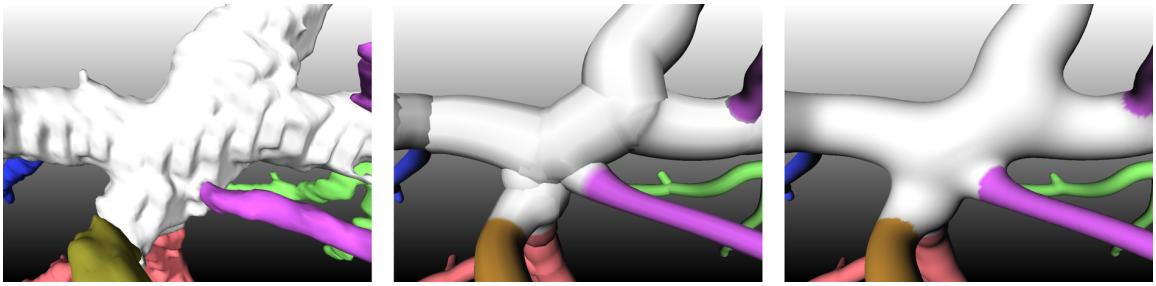


Abbildung 1.1: Nahansicht einer Verzweigung. Isosurface (links), Kegelstümpfe nach [HPSP01] (Mitte) und *Convolution Surface* (rechts).

messerverlaufs der Gefäße angestrebt. Des Weiteren soll eine weiche, geometrisch stetige Gefäßform besonders an Verzweigungen generiert werden. Gefäßenden sollten geschlossen werden. Eine letzte Maßgabe ist die Vermeidung der Konstruktion von Strukturen im Inneren des Gefäßbaums. Diese Ziele sind motiviert durch Lehrbuchillustrationen und Diskussionen mit Ärzten. Die folgende Auflistung fasst die Anforderungen an eine ideale Gefäßvisualisierungsmethode zusammen:

- korrekte Abbildung des Gefäßdurchmessers
- weiche, organische Gefäßform
- einheitliche Behandlung aller Verzweigungstypen
- geschlossene Gefäßenden
- Vermeidung von Strukturen im Gefäßinneren

Bei existierenden Techniken zur modellbasierten Gefäßvisualisierung stellt die Sicherstellung geometrischer Kontinuität an den Verzweigungen das größte Problem dar. Sie ist entweder nicht gewährleistet oder erfordert einen beträchtlichen Konstruktionsaufwand. Diese Erkenntnis und der Wunsch nach einer weichen Gefäßform motivieren den alternativen Einsatz von impliziten Oberflächen. Die Modellierung organisch wirkender Strukturen durch die Generierung weicher, geometrisch kontinuierlicher Übergänge zwischen einzelnen Objektteilen ist eine der großen Stärken von impliziten Oberflächen. Letzteres kann hier, allein basierend auf einer mathematischen Beschreibung, ohne zusätzlichen Konstruktionsaufwand erreicht werden.

Die vorliegende Arbeit ist wie folgt gegliedert:

Kapitel 2 beginnt mit einer Einführung in die konventionellen Verfahren der Gefäßvisualisierung. Basierend auf den hier auftretenden Problemen wird die Rekonstruktion eines Modells der Gefäßstruktur motiviert. Abschließend erfolgt eine Analyse von Visualisierungsmethoden, welche das Gefäßmodell auf darstellbare Graphikprimitive abbilden.

Kapitel 3 gibt einen Überblick zu impliziten Oberflächen. Sowohl die Modellierung mit dieser speziellen Art von Oberflächen, als auch deren Visualisierung wird hier behandelt. Den Schwerpunkt des Kapitels bildet die Diskussion der skelettbasierten Modellierung mittels *Convolution Surfaces* und die hierbei zu beachtenden Schwierigkeiten.

Kapitel 4 erarbeitet das Konzept einer Gefäßvisualisierungsmethode basierend auf *Convolution Surfaces* mit Hinblick auf die in diesem Kapitel definierten Anforderungen. Neben wichtigen Aspekten der Modellierung und einer impliziten Beschreibung der Gefäßstruktur wird deren Überführung in eine für die Graphikkarte handhabbare Repräsentation diskutiert. Letzteres erfordert ein Optimierungsverfahren zur Performanzsteigerung, dessen Entwurf ebenfalls Teil dieses Kapitels ist. Zuletzt werden einige Aspekte der Interaktion mit dem Visualisierungsresultat diskutiert.

Kapitel 5 ist einer Realisierung der entworfenen Visualisierungsmethode gewidmet. Hier werden die verwendeten Programmierwerkzeuge und Datenstrukturen erläutert. Die Implementierung der Optimierungs-, Darstellungs- und Interaktionstechniken ist innerhalb dieses Kapitels beschrieben.

Kapitel 6 beschäftigt sich mit der Validierung und Evaluierung des Visualisierungsresultats. Die Ergebnisse der Befragung erfahrener Chirurgen und Radiologen geben Auskunft über den möglichen klinischen Nutzen der entwickelten Visualisierungsmethode.

Kapitel 7 fasst die Ergebnisse dieser Arbeit zusammen, beschreibt noch offene Probleme und regt damit zu einer weiterführenden Betrachtung an.

2 Gefäßvisualisierung - Ein Überblick

Die ersten medizinischen Bildgebungsverfahren, welche zur Visualisierung und Exploration von Gefäßen im menschlichen Körper eingesetzt wurden, sind die konventionelle Digitale Subtraktionsangiographie (DSA) und 2D Ultraschall (US). Beide Methoden liefern zweidimensionale Ansichten der interessierenden Gefäßstruktur. Neben fehlender Tiefeninformation und dem Problem der Verdeckung von Gefäßen gestaltet sich die Orientierung in den Daten ohne dreidimensionale Kontextinformation als schwierig. Neuere Bildgebungsverfahren, wie die Computer Tomographie Angiographie (CTA) und die Magnet-Resonanz-Angiographie (MRA) gestatten eine dreidimensionale Repräsentation des gesamten aufgenommenen Gebietes. Die Größe der dabei anfallenden Datensätze erfordert jedoch geeignete Visualisierungsverfahren als Alternative zu einer ineffizienten, schichtweisen Auswertung durch den Radiologen oder Arzt. Die heutzutage populärsten hierfür entwickelten Techniken stellen die Daten direkt, bzw. über eine Zwischenrepräsentation dar. Neben den dabei auftretenden, allgemeinen Problemen, bedingt durch Bildrauschen oder Partialvolumeneffekt, motivieren gefäßspezifische, klinische Fragestellungen bezüglich der Morphologie des Gefäßsystems jedoch die Rekonstruktion des Gefäßbaums [GKS+93]. Zu den Aufgaben einer strukturellen Analyse des resultierenden Gefäßmodells zählen die Bestimmung von Teilbäumen, das Erkennen von Verzweigungstypen, die Vermessung des Durchmesserverlaufs entlang eines Gefäßes, Distanzmessungen und quantitative Vergleiche von mehreren Gefäßstrukturen. Im Folgenden wird eine Auswahl existierender Verfahren zur Gefäßvisualisierung beschrieben, mit Fokus auf modellbasierten Techniken. Diese nutzen die in der Rekonstruktion gewonnene Information und bilden sie auf geometrische Primitive ab. Für die hier vorliegende Arbeit sind die dabei angewandten Methoden von fundamentalem Interesse.

Der folgende Abschnitt gibt einen Überblick zu den konventionellen Techniken der Gefäßvisualisierung basierend auf dreidimensionalen Datensätzen. Abschnitt 2.2 ist der Gefäßanalyse, d.h. der Rekonstruktion des Gefäßbaums mittels Verfahren der Bildanalyse gewidmet. Neben allgemeinen Aspekten wird hier speziell auf den Gefäßanalyseteil der in [HPSP01] vorgestellten gefäßspezifischen Visualisierungspipeline eingegangen. Alle innerhalb dieser Arbeit verwendeten Gefäßmodelle stellen ein Ergebnis dieses konkreten Analyseteils dar. Abschnitt 2.3 fokussiert eine Visualisierung des durch die Rekonstruktion gewonnenen Modells. Abschließend werden die Vorteile und Nachteile der untersuchten Visualisierungsverfahren zusammengefasst und der alternative Einsatz von impliziten Oberflächen wird motiviert.

2.1 Konventionelle Techniken der Gefäßvisualisierung

Neuere Bildgebungsverfahren, wie CTA und MRA liefern eine dreidimensionale Repräsentation des aufgenommenen Gebietes. Die mittlerweile beträchtliche Größe (>1000 Schichten) der dabei anfallenden Datensätze erfordert geeignete Visualisierungsverfahren.

2.1.1 Darstellungen in 2D

Slicing ist die einfachste Methode zur Exploration dreidimensionaler Datensätze. Orthogonale Schichten ermöglichen eine Navigation in axialer, sagittaler oder coronaler Richtung durch den Datensatz. Für die Gefäßdiagnose ist dieses Verfahren nur bedingt geeignet, da ein Gefäß sehr selten vollständig in einer Schicht verläuft. Weiterhin ermöglicht der technische Fortschritt auf dem Gebiet der Bildgebungsverfahren heute Bildstapel mit mehr als 1000 Einzelbildern. Eine Unterstützung des Radiologen oder Arztes bei der Beurteilung dieser Flut von Information ist wünschenswert.

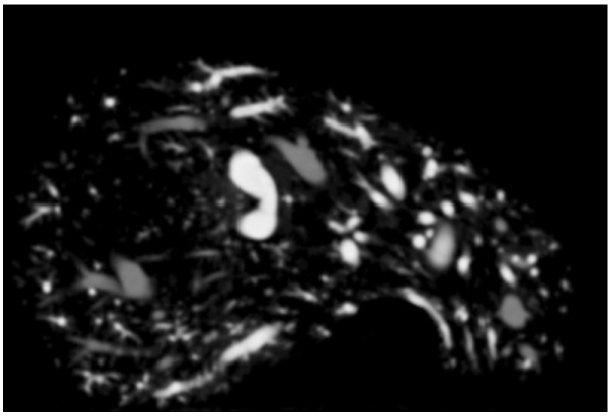


Abbildung 2.1: Zweidimensionale Schichtdarstellung eines Leber-CT-Datensatzes mit Kontrastmittelapplikation. *Quelle*: [HSEP00]

2.1.2 Darstellungen in $2\frac{1}{2}$D

Multiplanare Rekonstruktion (MPR) erlaubt die Festlegung beliebig orientierter Ebenen durch den Datensatz. Dies unterstützt die Exploration erheblich und ermöglicht z.B. die Bestimmung der Ausdehnung von Objekten in jeglicher Lage. Der nicht-planare Verlauf von Gefäßen ist jedoch weiterhin problematisch zu beurteilen.

Die *Curved Planar Reformation (CPR)* [KFW+02] gestattet es, zweidimensionale Ansichten eines kompletten Gefäßes zu erzeugen, obwohl dieses durch mehrere Schichten verläuft. Sie ist dahingehend eine Erweiterung von *Slicing* und MPR. Das Prinzip der CPR basiert auf einer gekrümmten Ebene im dreidimensionalen Raum, welche durch eine Kurve und einen Vektor orthogonal zu dieser definiert ist (Abb. 2.2).

Die Erzeugung einer zweidimensionalen Ansicht des durch die Kurve approximierten Gefäßes wird erreicht durch eine Bestimmung aller Voxel, welche von der Ebene geschnitten werden und die anschließende planare Darstellung dieser Menge (Abb. 2.3). Ein Hauptproblem dieser Methode ist die Verzerrung von Entfernungen und anatomischen Zusammenhängen im Ergebnisbild. Außerdem muss für jedes Gefäß eine eigene, gekrümmte Ebene per Hand definiert werden, was sowohl zeitaufwändig, als auch fehleranfällig ist.

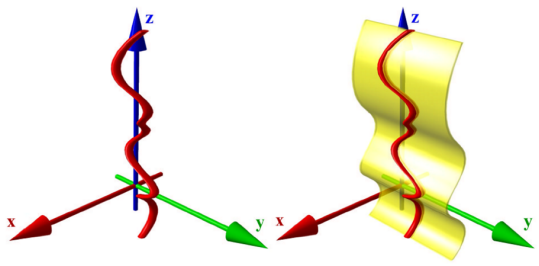


Abbildung 2.2: Prinzip der *Curved Planar Reformation*. *Quelle*: [KFW+02]

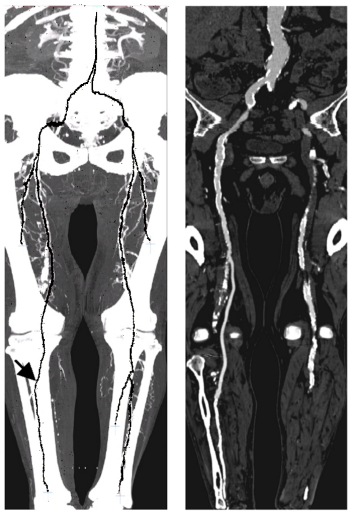


Abbildung 2.3: Verlauf der interessierenden Gefäße (*links*). CPR des mit Pfeil markierten Gefäßabschnitts (*rechts*). Die Knochenstrukturen (sehr helle Bereiche im linken Bild) sind hier segmentiert und entfernt wurden. *Quelle*: [KWF+01]

2.1.3 Darstellungen in 3D

Eine Darstellung des gesamten Datensatzes ermöglicht *Direct Volume Rendering (DVR)*. Diese Methode bildet Datenwerte mit Hilfe von so genannten Transferfunktionen auf Farb- und Transparenzwerte ab. Verschiedene Techniken wie *Ray Casting*, *Shear-warp Factorization* und *Splatting* haben sich hier im Laufe der Zeit etabliert. Deren genaue Erläuterung und Evaluierung ist nicht Ziel dieser Arbeit. Ein sehr detaillierter Überblick ist in [RS01] zu finden. In Anlehnung an [Kan01] werden im Folgenden verschiedene Projektionsmethoden am Beispiel von *Ray Casting* vorgestellt. Diese bilden die vorhandene Dateninformation auf unterschiedliche Art und Weise ab.

2.1.3.1 Projektionsmethoden

Ray Casting ist ein bildbasiertes Renderingverfahren, welches für jedes Pixel des Ergebnisbildes einen Strahl vom Betrachter aus durch Pixelmittelpunkt und den Datensatz konstruiert. Aus den vom Strahl getroffenen Voxeln und deren mit Hilfe der Transferfunktionen berechneten Eigenschaften (Farbe/Transparenz) wird der Pixelwert bestimmt. Dabei werden die Voxel entlang eines Strahls unterschiedlich gewichtet:

Bei einer Kontrastmittelgabe haben Gefäßvoxel einen besonders hohen Datenwert. Dies wird durch die *Maximum Intensity Projection* (MIP) ausgenutzt (Abb. 2.4 , links). Hier entscheidet nur das Voxel mit dem höchsten Datenwert entlang des Strahls über den korrespondieren Pixelwert des Bildes. Nach [EDKS94] ist diese Vorgehensweise jedoch problematisch, da kleine Gefäße und Gefäße mit geringem Kontrast nicht dargestellt werden. Andere Mängel sind eine Reduzierung des Gefäßdurchmessers und verfälschte bzw. nicht vorhandene Tiefeninformation. In CT-Daten haben Knochen ebenfalls sehr hohe Datenwerte und verdecken sowohl davor- als auch dahinterliegende Gefäße.

Die *Closest Vessel Projection (CVP)* [Zui95] (Abb. 2.4 , rechts) ist eine Verbesserung der MIP. Sie widmet sich dem Problem von verfälschter Tiefeninformation und ist außerdem in der Lage, auch sehr dünne Gefäße darzustellen. Die CVP erfordert die Festlegung eines Schwellwertes. Die Datenwerte entlang eines Strahls seien als Funktion in Abhängigkeit von der Position des zugehörigen Voxels gegeben. Statt dem Voxel mit dem höchsten Datenwert entscheidet jetzt das erste Voxel entlang des Strahls vom Betrachter aus gesehen mit einem lokalen Maximum größer dem Schwellwert über den finalen Pixelwert. Liegt nun ein dünnes Gefäß mit niedrigerem Datenwert vor einem großen Gefäß wird es trotzdem dargestellt, wenn der Datenwert größer dem Schwellwert ist (Abb. 2.5). Die Schwierigkeit bei dieser Methode liegt in der Auswahl eines geeigneten Schwellwertes.

Compositing berücksichtigt alle Voxel entlang des Strahls und berechnet aus deren Transparenz- und Farbwerten den Pixelwert. Anhand der Reihenfolge, in welcher die Voxel betrachtet werden unterscheidet man zwischen zwei Berechnungsmethoden: *front-to-back* (vom Benutzer weg) und *back-to-front* (zum Benutzer hin). Die erste Vorgehensweise akkumuliert die Transparenzwerte und erlaubt einen vorzeitigen Abbruch des Traversals bei hoher kumulierter Opazität. *Back-to-front compositing* hingegen gestattet einen sukzessiven Bildaufbau und somit die Darstellung von im finalen Ergebnisbild verdeckten Strukturen. Mit einer gezielten Einstellung der Transferfunktionen können Gefäßstrukturen hervorgehoben werden.

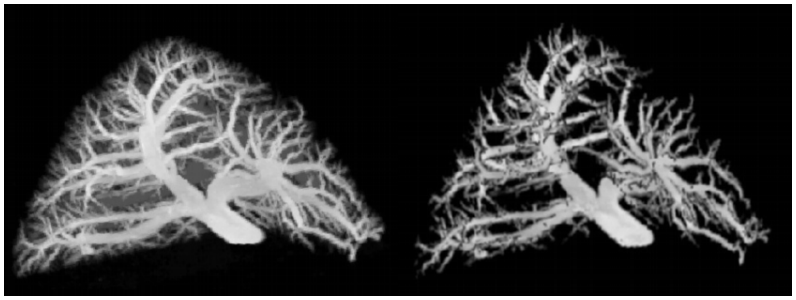


Abbildung 2.4: MIP (*links*) und CVP mit hohem Schwellwert (*rechts*) eines Lebergefäßbaums. Nach [HSEP00]

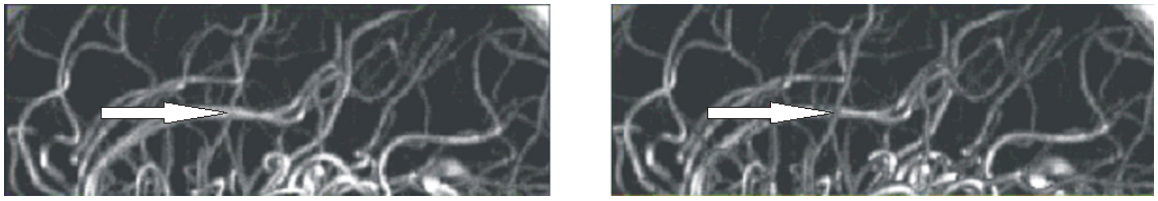


Abbildung 2.5: Vergleich von MIP (*links*) und CVP (*rechts*). Der Pfeil markiert eine kritische Stelle. Erst die CVP zeigt, dass das dunkle, vertikal verlaufende Gefäß vor dem helleren, horizontal verlaufenden Gefäß liegt. Diese Tiefeninformation wurde in der MIP verfälscht. Nach [Zui95]

Direct Volume Rendering-Methoden sind für eine strukturelle Analyse von Gefäßsystemen kaum geeignet. Zu den technisch bedingten Problemen der Bildaufnahme, wie Bildrauschen, Partialvolumeneffekt und geringe Auflösung der Bilddaten kommt das Fehlen einer quantitativen, handhabbaren Repräsentation des Gefäßbaums. Dies erschwert auch eine gefäßbezogene Interaktion.

2.1.3.2 Surface Rendering

Eine Alternative zu DVR-Techniken bildet *Surface Rendering*. Hier werden die Daten mit Hilfe eines vom Benutzer festgelegten Schwellwertes (*Isowert*) in „zum Gefäßbaum gehörend“ und Hintergrund klassifiziert. Diese binäre Entscheidung entspricht einer sehr einfachen Segmentierung. Statt einer direkten Darstellung der Daten wird eine Zwischenrepräsentation, z.B. ein Dreiecksnetz, generiert. Weitverbreitet hierfür ist das *Marching Cubes*-Verfahren [LC87]. Hierbei werden auf der Basis der Mittelpunkte von je vier Voxeln einer Schicht und vier Voxeln einer benachbarten Schicht so genannte *Zellen* konstruiert. Jedem Eckpunkt einer Zelle ist der jeweilige Datenwert des Voxels zugeordnet. Durch einen einfachen Vergleich des Isowertes mit den Datenwerten der Zelleckpunkte kann nun festgestellt werden, ob die durch den Isowert definierte Oberfläche (*Isosurface*) diese Zelle durchquert. Ist das der Fall, so werden die Schnittpunkte von Zelle und Oberfläche mittel trilinearer Interpolation approximiert. Basierend auf der Verteilung der berechneten Schnittpunkte entlang der Zelle kann der in ihr enthaltene Oberflächenanteil nun trianguliert werden. Nach der Betrachtung aller Zellen ist ein Dreiecksnetz entstanden, welches die Gefäßstruktur beschreibt.

Ein großes Problem von *Surface Rendering* ist die Wahl eines geeigneten Schwellwertes. Sowohl durch schon genannte, technisch bedingte Probleme der Bildaufnahme, wie auch durch eine ungünstige Schwellwertwahl entstehen nicht-zusammenhängende Gefäße (Abb. 2.6). Andererseits

können Strukturen mit ähnlichem Datenwert, wie z.B. Knochen, mit der Gefäßstruktur verschmelzen. Für die strukturelle Analyse von Gefäßsystemen ist dieses Verfahren nicht geeignet, da das entstandene geometrische Modell nur bedingt die topologischen und geometrischen Eigenschaften des Gefäßbaums wiederspiegelt. Verbesserungen sind allerdings möglich, z.B. durch eine Glättung der Isosurface oder eine *connected component analysis*. Letztere bestimmt auf der Basis von Nachbarschaften zusammenhängende Gebiete in einem Bild oder Volumen. Durch Bildrauschen entstandene sehr kleine Gebiete können auf diese Weise gefiltert werden.

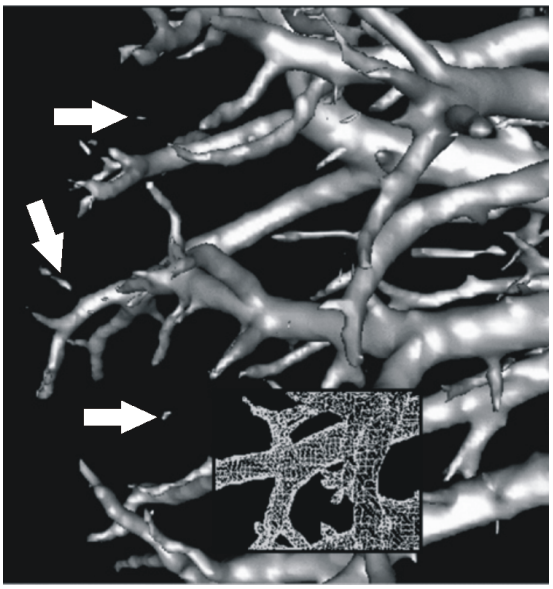


Abbildung 2.6: *Surface Rendering* eines Gefäßsystems. Die Pfeile markieren fälschlicherweise durch die Segmentierung abgetrennte Gefäßteile. Das Inset zeigt die Auflösung des zugrundeliegenden Dreiecksnetzes. Nach [HSEP00]

Zusammenfassend lässt sich sagen, dass es allen bisher betrachteten Methoden, die nur die Intensitätswerte der Originaldaten berücksichtigen, vor allem an einer aussagekräftigen, handhabbaren Repräsentation des Gefäßbaums mangelt. Die grobe Segmentierung der *Surface Rendering*-Methode weist eine vielversprechende Richtung, ist jedoch keineswegs ausreichend.

2.2 Rekonstruktion eines Gefäßmodells

In der von [HPSP01] vorgestellten gefäßspezifischen Visualisierungspipeline (Abb. 2.7) ist die Segmentierung des Gefäßbaums der erste Schritt nach der Bildaufnahme und Bildvorverarbeitung (z.B. Rauschunterdrückung) in Richtung einer kompakten, geometrischen Repräsentation.

2.2.1 Segmentierung

Im Laufe der Jahre wurden zahlreiche Verfahren für die Segmentierung von Gefäßstrukturen entwickelt. Deren detaillierte Behandlung ist nicht Teil dieser Arbeit. Einige allgemeine gefäßspezifische Anforderungen sind jedoch festzuhalten:

- Automatische Erkennung der Gefäßstruktur.
- Berücksichtigung von Auflösungsdefiziten der Daten sowie Bildrauschen und Partialvolumeneffekt.
- Detektion selbst sehr dünner, verzweigter Gefäße.

Die meisten bis heute entwickelten Verfahren sind aufgrund der anatomischen Vielfalt nur halb-automatisch und bedürfen oft einer zeitintensiven Benutzerinteraktion. Verbreitet sind *Region-Growing*-Methoden [SPSP02], die ausgehend von einem selektierten Saatpunkt und einem Schwellwert Voxel akkumulieren, welche das Schwellwertkriterium erfüllen. Wenn die Detektion sehr kleiner Gefäße im Vordergrund steht, werden Multiskalenansätze verwendet und Segmentierungsergebnisse verschiedener Ebenen kombiniert. Für spezielle Gefäßsegmentierungsaufgaben kommen modellbasierte Ansätze zum Tragen [SPSP02].

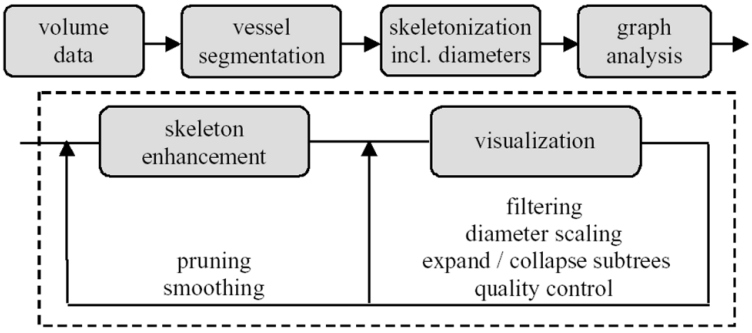


Abbildung 2.7: Gefäßspezifische Visualisierungspipeline. Der nicht gestrichelt umrandete Bereich markiert die einzelnen Schritte der Gefäßanalyse. *Quelle*: [HPSP01]

2.2.2 Skelettierung

Es wurde frühzeitig erkannt [GKS+93], dass eine reine Segmentierung für neue klinische Fragestellungen an die Gefäßmorphologie nicht ausreichend ist. Hierzu zählen die Vermessung des Durchmesserverlaufs entlang eines Gefäßes, Distanzmessung, die Erkennung von Teilbäumen und anderer topologischer Information sowie quantitative Vergleiche von mehreren Gefäßstrukturen. Die aus der Segmentierung resultierende Voxelstruktur muss daher in eine aussagekräftigere Datenstruktur transformiert werden. Dieses Ziel kann durch eine Skelettierung des Gefäßbaums erreicht werden, welche Schritt 3 in der Visualisierungspipeline von [HPSP01] darstellt (Abb. 2.7).

Der Begriff Skelett wurde durch [Blu67] eingeführt, um biologische Formen charakterisieren und beschreiben zu können. Das Skelett eines Objektes wird durch die *Medial Axes Transform (MAT)* beschrieben. Unter der MAT versteht man „die Ortslinie der Mittelpunkte aller maximalen Innkreise bzw. Innkugeln innerhalb der Oberfläche eines Objektes [Blu67]“[1].

Nach [PTN97] liefert die Skelettierung des Segmentierungsresultats eine symbolische Beschreibung des Gefäßbaums, welche ein Verständnis von dessen Struktur und Topologie wesentlich erleichtert. Im Hinblick auf eine Visualisierung ermöglicht diese Beschreibung die Generierung eines polygonalen Modells, welches die Gefäßoberfläche approximiert. Solch eine Visualisierung

[1] *the locus of the centers of all maximal inscribable discs/spheres within the boundary of an object*

ist besser geeignet für Interaktion, Navigation und Fließsimulation, als die durch *Surface Rendering* generierte geometrische Repräsentation.

Auch für die Skelettierung wurden bereits viele Techniken entwickelt. Die wichtigsten Anforderungen sind hier:

- Ausdünnung des Segmentierungsresultats auf Voxellisten.
- Berechnung der idealen Mittellinie (Skelett) des Gefäßes.
- Gewinnung und Speicherung von Oberflächeninformation, z.B. Gefäßdicke, an Abtastpunkten entlang des Skeletts.
- Aus Skelett und gespeicherter Oberflächeninformation sollte die „originale“ Gefäßoberfläche so gut wie möglich approximiert werden können.

Sowohl in 2D, als auch in 3D ist die Skelettierung von diskret repräsentierten Objekten mit einigen Schwierigkeiten verbunden. So würde die einfache Skelettierung der „gezackten“ Gefäßoberfläche in zahlreichen kleinen Seitenästen resultieren. Weiterhin hat dieser Prozess in 3D normalerweise eine Mittelfläche statt einer Mittellinie zum Ergebnis. Daher wird in [SPSP02] *Thinning* eingesetzt. Diese Methode erodiert Voxel sukzessive bis eine 1 Voxel breite Mittellinie verbleibt (Abb. 2.8). Die Erosion muss symmetrisch erfolgen, was aufgrund der meist anisotropen Ausdehnung der Voxel besondere Sorgfalt erfordert. Aus der Skelettierung mittels *Thinning* resultiert eine voxelbasierte Repräsentation der Mittelachsen des Gefäßsystems, zusammen mit einer Radiusinformation pro Skelettvoxel. Abschließend werden Voxel identifiziert, welche entweder Verzweigungen oder Gefäßenden darstellen.

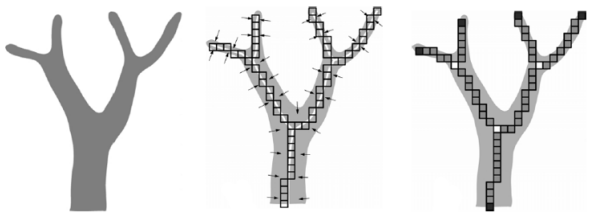


Abbildung 2.8: 2D Illustration der Gefäßskelettierung. Das Segmentierungsresultat (*links*) wird durch *Thinning* (*Mitte*) auf die ein Voxel breite Gefäßachse reduziert. Anschließend werden Skelettvoxel bestimmt, welche Verzweigungen und Gefäßenden repräsentieren (*rechts*). Nach [SSPP00]

2.2.3 Graphenanalyse

Die Graphenanalyse bildet Schritt 4 der in [HPSP01] vorgestellten gefäßspezifischen Visualisierungspipeline (Abb. 2.7). In einigen Organen des menschlichen Körpers sind mehrere Gefäßsysteme lokalisiert. Das Gefäßsystem der Leber beispielsweise umfaßt die Pfortader, Leberarterien, Lebervenen und die Gallengänge. Diese sind allein durch Segmentierung und Skelettierung oft nicht separabel. Die begrenzte Auflösung der Originaldaten verursacht häufig eine Verschmelzung von Gefäßsystemen, wenn zwei Voxel unterschiedlicher Systeme benachbart sind (Abb. 2.9 links). Eine Aufgabe der Graphenanalyse ist es, diese Stellen zu finden und die Gefäßsysteme zu trennen. Weiterhin wird das Gefäßskeletts in eine hierarchisch organisierte Graphenstruktur $G = (V, E)$,

mit V Knoten und E Kanten überführt. Knoten repräsentieren Verzweigungen und Kanten beschreiben jeweils einen Gefäßast zwischen zwei Verzweigungen. Die Trennung der Gefäßsysteme basiert auf der Annahme, das der Gefäßdurchmesser von der Wurzel zur Peripherie relativ stetig abnimmt [SPSP02].

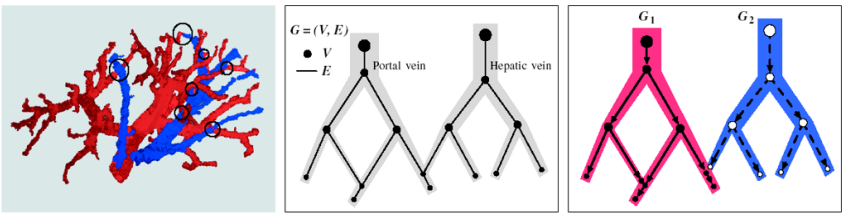


Abbildung 2.9: Graphenanalyse von Lebergefäßsystemen. Pfortader und Lebervenen verschmelzen durch die Segmentierung an den durch Kreise markierten Punkten miteinander (*links*). Der Graph zweier sich berührender Gefäßsysteme (*Mitte*) und die gerichteten azyklischen Graphen der beiden Gefäßsysteme (*rechts*). *Quelle*: [SSPP00]

Die wichtigsten Resultate der Graphenanalyse sind eine Trennung der einzelnen Gefäßsysteme und die Überführung in eine hierarchische Graphenstruktur. Letztere ist über ein dreidimensionales Voxelgitter definiert, welches dem der originalen Daten aus der Bildaufnahme entspricht. Die Graphenstruktur kodiert Topologie- und Konnektivitätsinformationen in einem gerichteten azyklischen Graph (Abb. 2.9 rechts). Jeder Kante des Graphen ist eine Liste mit Skelettvoxeln und korrespondierender Radiusinformation zugeordnet. Diese Informationen ermöglichen sowohl eine Rekonstruktion der Gefäßoberfläche im Rahmen eines Visualisierungsprozesses, als auch geometrische und strukturelle Analysen der Gefäßstruktur. Sie sind weiterhin Basis für eine Vielzahl von Interaktionen. So können Teilbäume ausgewählt, ausgeblendet oder eingefärbt werden. Zusätzlich ist die Bestimmung der Länge und des Volumens einzelner Teilstrukturen möglich.

Die nach [HPSP01], [SPSP02] durch Segmentierung, Skelettierung und Graphenanalyse gewonnene Information dient als Eingabe für die in dieser Arbeit vorgestellte Visualisierungsmethode. Sie liegt aufbereitet in einer Textdatei (*.txt) bzw. im *Extensible Markup Language*-Format (*.xml) vor. In Anhang A ist ein Auszug aus solch einer Datei gegeben. Der folgende Abschnitt behandelt die Visualisierung des rekonstruierten Gefäßmodells.

2.3 Modellbasierte Gefäßvisualisierung

Die modellbasierte Gefäßvisualisierung setzt eine symbolische Beschreibung der Gefäßstruktur voraus, welche ihre topologischen und geometrischen Eigenschaften adäquat kodiert. Die Gefäßoberfläche sollte aus dieser Beschreibung so originalgetreu wie möglich rekonstruiert werden können.

Eine Möglichkeit hierfür bieten die so genannten *surface reconstruction*-Techniken. Diese verlangen eine Beschreibung der Oberfläche entlang des Skeletts durch Konturen (Abb. 2.10). Benachbarte Konturen werden zur Konstruktion der Gefäßoberfläche Stück für Stück miteinander verbunden. In der Gefäßvisualisierung entsprechen Querschnitte durch das Gefäß den Konturen. Diese werden entlang des Gefäßskeletts an wichtigen Punkten, z.B. signifikante Änderung der Gefäßdicke, bestimmt. Gefäßquerschnitte sind oft einfache geometrische Gebilde welche sich nicht

überlappen dürfen, da Gefäße keine Löcher aufweisen. Zumeist basiert die Beschreibung des Modells auf der Annahme kreisrunder Querschnitte nicht-pathologischer Gefäße, wie in [MMD96] diskutiert.

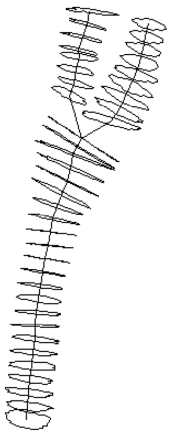


Abbildung 2.10: Gefäßskelett mit zugehörigen Querschnitten durch das Gefäß.
Quelle: [EDKS94]

2.3.1 Gefäßvisualisierung mit Zylindern

2.3.1.1 Barillot

Schon 1985 entwickelten [BGSC85] eine symbolische Beschreibung für zerebrale Gefäßbäume. Für die Skelettierung werden Abtastpunkte entlang des Skeletts manuell festgelegt. Die dort gemessene Dicke des Gefäßes wird gespeichert. Somit entstehen kreisförmige Querschnitte entlang des Gefäßskeletts, welche die Durchmesserverteilung wiederspiegeln. Die Visualisierung erfolgt hier durch die Konkatenation von Zylindern mit einem Durchmesser äquivalent zur gemessenen Gefäßdicke (Abb. 2.11).

Diese Konstruktionsmethode ist effizient und für Aufgaben wie das Erkennen von Verzweigungen vollkommen ausreichend. Die Verwendung von Zylindern zur Approximation der Gefäße bringt jedoch auch erhebliche Probleme mit sich. Veränderungen der lokalen Gefäßdicke können aufgrund des invarianten Gefäßdurchmessers nicht abgebildet werden. Die Übergänge zwischen den Primitiven, besonders an Verzweigungen sind abrupt, nicht geometrisch kontinuierlich und wirken daher unnatürlich. Desweiteren entstehen an den Verzweigungen Strukturen im Inneren des Gefäßbaums, was eine virtuelle Endoskopie stark behindert und den Renderingaufwand unnötig erhöht. Außerdem würde die transparente Darstellung von Gefäßteilen, wie sie in einer Interaktion zur Akzentuierung bestimmter Regionen denkbar ist, dadurch erheblich gestört werden. Auch eine Vereinfachung des geometrischen Modells durch *Mesh Simplification*-Methoden könnte so leicht in einer artefaktbehafteten Visualisierung resultieren.

2.3.1.2 Gerig

Die Erzeugung einer symbolischen Beschreibung zerebraler Gefäßbäume, welche aus MRA-Daten gewonnen wurden, wird in [GKS+93] beschrieben. Nach einer initialen Segmentierung durch 3D-

Hysteresis Thresholding erfolgt ein 3D-*binary thinning*, welches das Voxelskelett des Gefäßbaums liefert. Über eine 3D-Distanztransformation wird für jedes Gefäßvoxel aus der lokalen Gefäßbreite der korrespondierende Radius abgeschätzt. Die Autoren heben weiterhin die Bedeutsamkeit einer Graphenrepräsentation des Gefäßbaums hervor. Diese erlaubt z. B. die Kennzeichnung von Teilbäumen und gestattet die Anwendung von Analysemethoden der Graphentheorie. Daher wird das Voxelskelett in einem abschließenden Schritt in einen Graphen umgewandelt, welcher sowohl die Topologie des Gefäßbaums, als auch seine geometrischen Eigenschaften kodiert. Die Visualisierung erfolgt hier ebenfalls durch die Konkatenation von Zylindern mit einem Durchmesser äquivalent zur gemessenen Gefäßdicke (Abb. 2.11). Die dabei auftretenden Probleme wurden weiter oben bereits erwähnt.

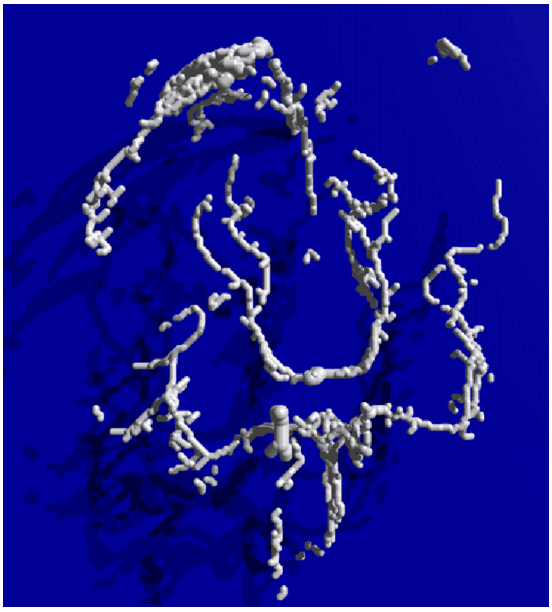


Abbildung 2.11: *Ray Tracing* der symbolischen Beschreibung eines zerebralen Gefäßbaums, bestehend aus Zylindern mit einem Radius äquivalent zu der gemessenen, lokalen Gefäßbreite. *Quelle*: [GKS+93]

2.3.1.3 Masutani

Die Extraktion topologischer Information, basierend auf einer *Region-Growing*-Technik und morphologischen Operationen, ist in [MMD96] erläutert. Die besondere Aufmerksamkeit gilt hier der Kontrolle von sehr kleinen Seitenästen und der Erhaltung der Gefäßtopologie. Die resultierende Repräsentation des Gefäßbaums umfasst dessen Skelett und zusätzlich gespeicherte Radiusinformation. Auch hier werden kreisförmige Querschnitte entlang des Gefäßskeletts angenommen. Das Modell wird durch Konkatenation von Zylindern mit einem Durchmesser äquivalent zur gemessenen Gefäßdicke visualisiert. Die dabei auftretenden Probleme wurden bereits beschrieben.

2.3.2 Gefäßvisualisierung mit Kegelstümpfen

2.3.2.1 Puig

[PTN97] widmet sich speziell der Identifikation und Modellierung von Gefäßkrankheiten, wie Stenosen und Aneurysmen. Die Segmentierung des Gefäßbaums erfolgt mit Hilfe bekannter Methoden. Ein *Distance-Map* basierter Algorithmus liefert das Gefäßskelett. Dieses wird in eine Graphstruktur konvertiert. Deren Knoten beschreiben Verzweigungen und die Kanten repräsentieren Gefäßabschnitte. In der Implementierung erfolgt die Erkennung besonderer Merkmale, wie Bifurkationen oder signifikanter Veränderungen im Durchmesserverlauf, durch Abtastung der Gefäßäste und Verfolgung des lokalen Gefäßdurchmessers. Erfolgt eine signifikante Änderung, wird der Punkt zusammen mit der Gefäßdicke abgespeichert.

Neben verbesserten DVR-Methoden schlägt [PTN97] eine direkte Darstellung des symbolischen Modells vor (Abb. 2.12, links). Weiterhin wird in [Pui98a] ein theoretisches Visualisierungsmodell skizziert. Dieses basiert auf verallgemeinerten Zylindern, welche durch eine Profilkurve und durch Konturkurven beschrieben sind. Der Profilkurve entspricht die Skelettkurve, welche durch Punkte approximiert wird deren Anordnung C^1-Kontinuität garantiert. Die Konturkurven können durch Polygone, Kreise, Ellipsen oder Splines repräsentiert werden. Die ermöglicht eine adaptive Darstellungsqualität in Abhängigkeit von der Anwendung: Zylinder (Fließsimulation) bis Approximation Freiformflächen (Navigation).

In [PTN97] beschränkt sich die Visualisierung auf den Einsatz von Kegelstümpfen zwischen benachbarten Querschnitten und Kugeln an Verzweigungen und Aneurysmen (Abb. 2.12, rechts). Gegenüber Zylindern macht die Verwendung von Kegelstümpfen auch die Abbildung lokaler Veränderungen der Gefäßdicke möglich. Die Übergänge zwischen zwei Gefäßabschnitten, entlang nicht verzweigter Gefäßäste, wirken wesentlich natürlicher. Alle anderen Probleme, wie zum Beispiel die Diskontinuitäten an Verzweigungen, treten jedoch auch hier auf.

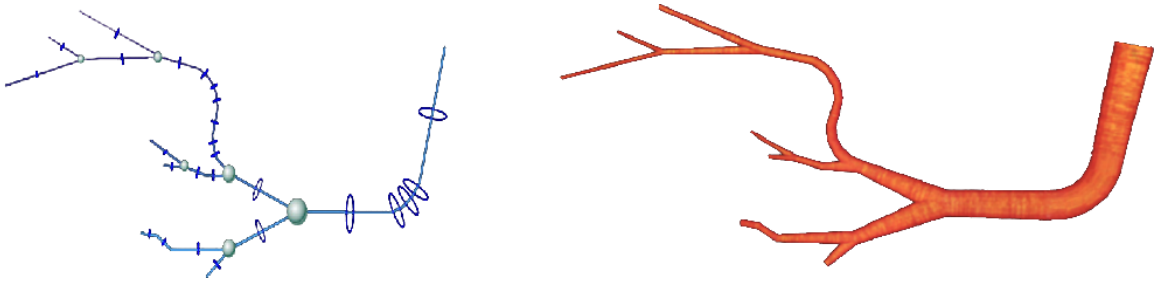


Abbildung 2.12: Direkte Darstellung des symbolischen Modells (*links*). Kugeln repräsentieren Verzweigungen. Polylinien beschreiben die Kanten und Kreise kodieren die lokale Gefäßdicke. Polygonale, gerenderte Repräsentation desselben Gefäßmodells (*rechts*). Nach [Pui98b]

2.3.2.2 Hahn

[HPSP01] streben durch eine symbolische Beschreibung von Gefäßbäumen ein besseres Verständnis der Hierarchie und die Ermöglichung von Interaktion an. Die dafür entwickelte Visualisierungspipeline ist speziell auf die Bedürfnisse der Gefäßvisualisierung zugeschnitten (Abb. 2.7). Der Gefäßanalyseteil der Pipeline wurde bereits in Abschnitt 2.2 detailliert beschrieben. Das

Resultat der Graphenanalyse ist ein gerichteter, azyklischer Graph, dessen Knoten die Verzweigungen repräsentieren, während die Kanten jeweils einen Gefäßabschnitt zwischen zwei Verzweigungen beschreiben. Jede Kante speichert Informationen über die Voxel, welche sie überspannt und pro Voxel eine Radiusinformation. Diese definiert kreisförmige Querschnitte entlang des Gefäßskeletts.

Aufgrund der diskreten Natur der Originaldaten werden in [HPSP01] einige Vorverarbeitungsschritte zur Verbesserung des Gefäßskeletts vorgeschlagen. So resultiert der Skelettierungsschritt oft in fälschlicherweise konstruierten, sehr kleinen Seitenästen. Eine Korrekturmethode, welche sich diesem Problem widmet wurde bereits in [MMD96] als *Pruning* vorgestellt. Hierbei wird ein Teilast eliminiert, wenn seine Länge kleiner ist als der Durchmesser des Nachbargefäßes auf höherer Hierarchiestufe multipliziert mit einer Konstante.

Ein weiteres Artefakt ist der „gezackte“ Verlauf des Gefäßskeletts (Abb. 2.13 links). Mit Hilfe eines Binomialfilters werden daher die Kanten des Skeletts geglättet (Abb. 2.13 rechts). Die Stärke der Glättung ist als Parameter vom Benutzer einstellbar. Besondere Sorgfalt ist hier an den Verzweigungen notwendig [HPSP01].

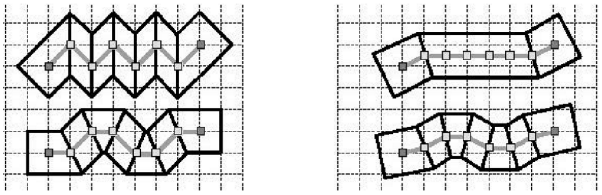


Abbildung 2.13: Glättung durch einen Binomialfilter. Das gezackte Skelett (*links*) wird durch den Binomialfilter geglättet (*rechts*). Die Endpunkte eines Gefäßabschnitts werden jedoch nicht beeinflusst. *Quelle*: [HPSP01]

Ein zusätzlicher störender Effekt ist die teilweise hohe Variation des Gefäßdurchmessers direkt benachbarter Skelettvoxel. Der Verlauf des Gefäßdurchmessers kann jedoch mit einer ähnlichen Methode, wie für die Kanten des Gefäßskeletts bereits erwähnt, geglättet werden. Auch hier ist den Verzweigungen gesonderte Aufmerksamkeit zu schenken. Der Glättungsfaktor ist als Parameter für den Benutzer zugänglich.

Die finale Visualisierung des Gefäßmodells basiert auf der Verwendung von Kegelstümpfen für Gefäßabschnitte zwischen benachbarten Querschnitten und Halbkugeln zum Schließen der Gefäßenden. Auf die mit der Konkatenation von Kegelstümpfen verbundenen Probleme (Abb. 2.14) wurden bereits weiter vorn eingegangen.

2.3.3 Gefäßvisualisierung mit Freiformflächen

In [EDKS94] repräsentieren Spline-Kurven den Verlauf des Gefäßskeletts. Konturen werden jeweils durch eine die Gefäßoberfläche einfassende Voxelstruktur beschrieben. Weiter sind keine vereinfachenden Annahmen hinsichtlich der Form des Gefäßquerschnitts getroffen. Ziel der hier vorgestellten Visualisierungsmethode ist die Erzeugung einer glatten, geometrisch kontinuierlichen Oberfläche, welche genügend Flexibilität besitzt, um Stenosen und Aneurysmen modellieren zu können. Dies wird durch eine *Mean Surface Approximation* mit Hilfe von Freiformflächen erreicht (Abb. 2.15). Die Autoren geben zu bedenken, dass die geometrische Kontinuität vor allem

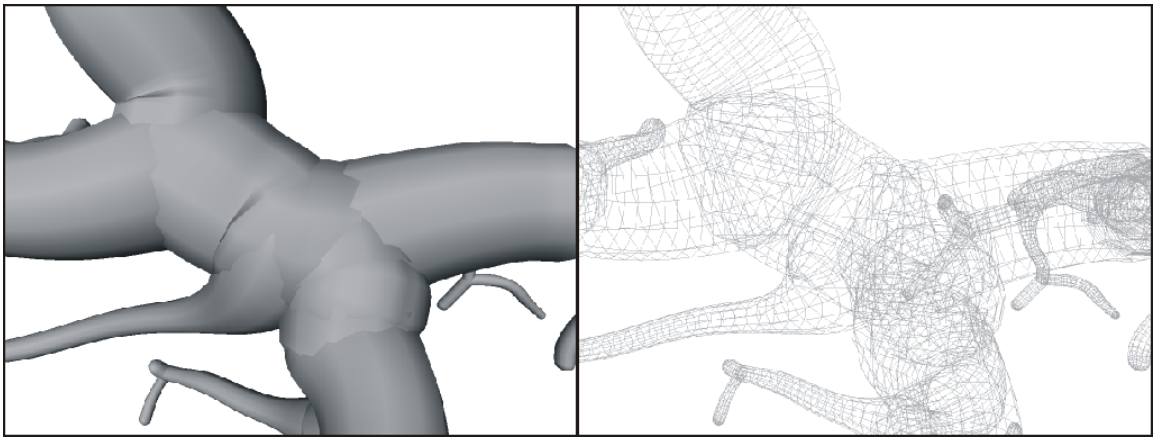


Abbildung 2.14: Diskontinuitäten der Gefäßoberfläche an Verzweigungen (*links*) und störende Strukturen im Inneren des Gefäßbaums (*rechts*).

an Verzweigungen genau überwacht werden muss. Leider gibt es weder Angaben zu den Erfolgsaussichten hierbei, noch sind Bilder größerer Gefäßabschnitte oder Komplexitätsmessungen der Konstruktionsphase enthalten.

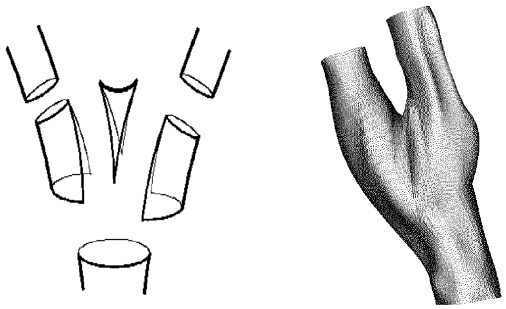


Abbildung 2.15: Modellierung mit Freiformflächen (*links*) und eine abstrakte Darstellung der Gefäßtopologie (*rechts*). *Quelle*: [EDKS94]

2.3.4 Gefäßvisualisierung mit Subdivision Surfaces

Die in [FFKW02] veröffentlichte Visualisierungsmethode wurde im Rahmen des Projektes ARAS[2] (*Augmented Reality Aided Surgery*) zur Computergestützten Chirurgie für Leberoperationen entwickelt. Während der Operation wird der aus präoperativen Daten rekonstruierte Gefäßbaum über die Operationsstelle projiziert und in Echtzeit an die Blickrichtung des Chirurgen angepasst. Die Hauptanforderung an die Visualisierungsmethode ist daher die Gewährleistung einer hohen Bildwiederholrate bei einer Interaktion mit dem Modell. Das Skelett des Gefäßbaums sowie zugehörige Radiusinformation wird nach [KWF$^+$01] bestimmt. Gefäßquerschnitte werden wiederum als kreisförmig angenommen. In einem Vorverarbeitungsschritt wird das Gefäßskelett geglättet. Weiterhin wird für jeden Skelettpunkt ein lokales Koordinatensystem berechnet. Dieses spannt einen so genannten *Reference-Frame* auf, dessen Berücksichtigung entlang eines

[2] `http://www.vrvis.at/br1/aras/`; Stand: 9.02.2004

Gefäßsegments Windungen der zu konstruierenden Oberfläche verhindert. Die Visualisierung besteht im Wesentlichen aus zwei Schritten.

Zuerst wird ein grobes Basismodell konstruiert. Hierfür wird der kreisrunde Gefäßquerschnitt durch das kleinste, ihn umschließende Quadrat approximiert. Der Abschnitt zwischen zwei Querschnitten ist nun durch ein annähernd quaderförmiges Gebilde beschrieben. Für alle vier gedachten Seitenflächen dieser Struktur wird in einer rekursiven Prozedur berechnet, ob ein neuer Gefäßast in diese Richtung abzweigt. Ist dies nicht der Fall, wird der Gefäßabschnitt dort durch ein viereckiges Füllstück geschlossen. Andernfalls wird der Gefäßabschnitt mit dem nächstliegenden Ast verbunden. Die Rekursion besteht in der Suche nach abgehenden Gefäßästen, in Richtung jedes neuen viereckigen Füllstücks (Abb. 2.16).

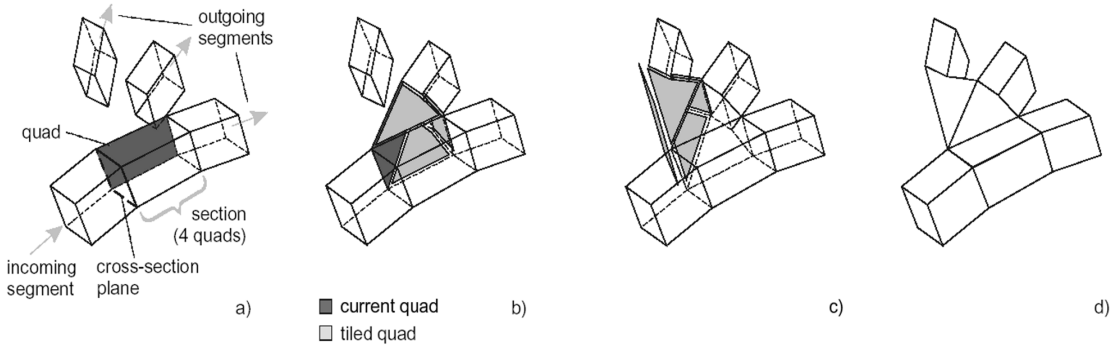


Abbildung 2.16: Rekonstruktion einer Trifurkation. In Richtung des dunklen Füllstücks wird ein abzweigender Ast gefunden (a). Der aktuelle und der abzweigende Ast werden miteinander verbunden (b). Durch die rekursive Betrachtung der neuen Füllstücke wird ein weiterer Ast entdeckt und mit der existierenden Bifurkation verbunden (c). Das fertige Basismodell der Trifurkation ist in (d) dargestellt. Nach [FFKW02]

Im zweiten Schritt der Visualisierung kann das entstandene Basismodell mit Hilfe von *Subdivision Surfaces* [DKT98], [BZ01] beliebig oft iterativ verfeinert werden. Die Glätte und die Komplexität (Anzahl der Polygone) des finalen Modells sind von der Iterationstiefe abhängig. Die Korrektheit des abgebildeten Durchmesserverlaufs ist schwer einzuschätzen. Den Autoren genügt eine Approximation, da ihnen Lage und Orientierung des Gefäßbaums in ihrer Anwendung wichtiger erscheinen. An den Verzweigungen treten keine Diskontinuitäten auf (Abb. 2.17). Interessant wäre eine Klärung, inwieweit dies von der Iterationstiefe abhängig ist. Weiterhin sind keine Aussagen gemacht worden über die Verhältnisse von Iterationstiefe zu Glätte des Modells und zu Polygondichte. Letztere ist entscheidend für die Gewährleistung einer hohen Bildwiederholrate. Eine Stärke des Verfahrens ist die Vermeidung der Konstruktion von Strukturen im Inneren des Gefäßbaums. Gefäßenden werden jedoch nicht geschlossen. Teilweise treten Verdickungen an Verzweigungen auf, welche versehentlich als Aneurysmen interpretiert werden könnten. Die Anwendung des Verfahrens im Rahmen der Gefäßdiagnostik ist jedoch nicht vorgesehen.

2.3.5 Direkte Visualisierung der symbolischen Gefäßbeschreibung

Die automatische Generierung der symbolischen Beschreibung von Gefäßbäumen aus CT-Daten wird in [ZJP95] vorgestellt. Das Resultat ist ein Baum aus nummerierten Zweigen, welche an Bifurkationsknoten miteinander verbunden sind. Zu jedem dieser Knoten ist der Durchmesser der dort eingehenden Gefäße und deren Orientierung gespeichert. Obwohl diese Informationen

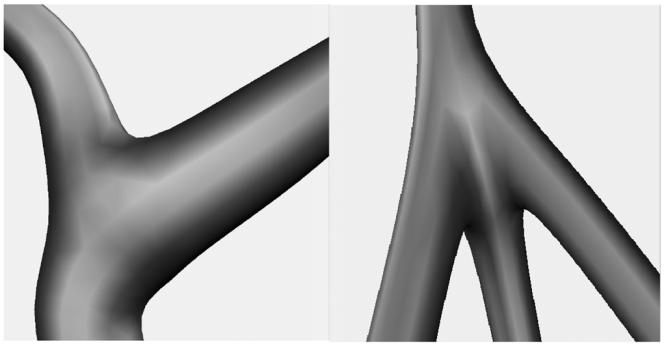


Abbildung 2.17: Nahansichten einer Bifurkation (*links*) und einer Trifurkation (*rechts*). *Quelle*: [FFKW02]

eine oberflächenorientierte Darstellung gestatten, wird in [ZJP95] das symbolische Modell direkt dargestellt (Abb. 2.18).

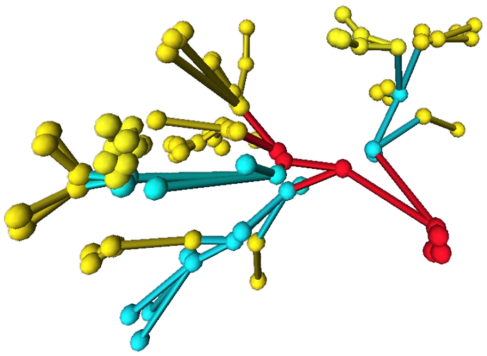


Abbildung 2.18: Grauwertkodierte Darstellung der Hierarchie eines Lebergefäßsystems. Kugeln repräsentieren Verzweigungen. *Quelle*: [ZJP95]

2.4 Zusammenfassung

Slicing und Multiplanare Rekonstruktion sind heute im klinischen Alltag weit verbreitet und werden innerhalb der Gefäßdiagnostik eingesetzt. Aufgrund der bei neueren Bildgebungsverfahren (CTA und MRA) anfallenden Datenmengen ist die schichtweise Exploration der Bildstapel jedoch sehr zeitintensiv für den Radiologen oder Arzt. Weiterhin erstrecken sich Gefäße über viele Schichten, was die Erfassung der Gefäßtopologie erschwert.

Die große Stärke von DVR-Methoden liegt in der interaktiven Visualisierung des gesamten dreidimensionalen Datensatzes. Leider liefert DVR keine handhabbare Repräsentation des Gefäßsystems, was Voraussetzung ist für eine strukturelle Analyse und für gefäßspezifische Interaktion. *Surface Rendering*-Methoden unterteilen den Datensatz mit Hilfe eines Schwellwerts in relevante Information (Gefäßbaum) und Kontextinformation. Der Gefäßbaum wird durch ein Polygonnetz approximiert. Für die strukturelle Analyse von Gefäßsystemen ist dieses Verfahren ebenfalls nicht geeignet, da das entstehende geometrische Modell nur bedingt die Topologie des

Gefäßbaums wiederspiegelt. In der Therapieplanung sowie in der medizinischen Ausbildung ist es jedoch wünschenswert, dass die Gefäßtopologie korrekt aus der Visualisierung abgeleitet werden kann. So sollte beispielsweise erkennbar werden, welcher Teil eines Gefäßbaumes ausfällt, falls ein Gefäßast durchtrennt werden muss. Weiterhin sollten Krümmung, Tiefeninformation und Durchmesserverlauf der Gefäße adäquat abgebildet werden. Herkömmliche Methoden der medizinischen Volumenvisualisierung sind hierfür nicht geeignet, da sie keine Repräsentation des Gefäßbaums liefern, welche topologiebezogene und geometrische Informationen kodiert. Problematisch sind weiterhin Bildrauschen, Partialvolumeneffekt und die begrenzte Auflösung von CTA- oder MRA-Daten. Zudem ist die Trennung von kontrastverstärkten Gefäßen und anderen Strukturen, wie Knochen in CTA-Daten, mit erheblichen Schwierigkeiten verbunden.

Die Segmentierung des Gefäßbaums ist der erste Schritt nach der Bildaufnahme und Bildvorverarbeitung in Richtung einer kompakten, geometrischen Repräsentation. Jedoch ist dies allein für die Beantwortung klinischer Fragestellungen an die Gefäßmorphologie nicht ausreichend. Die aus der Segmentierung resultierende Voxelstruktur muss daher in eine aussagekräftigere Datenstruktur transformiert werden. Dieses Ziel wird durch eine Skelettierung und eine Verzweigungsanalyse des Gefäßbaums erreicht. Das resultierende Gefäßskelett spiegelt den Gefäßverlauf wieder. Die Gefäßoberfläche wird durch Querschnitte approximiert, welche zusammen mit den korrespondierenden Skelettpunkten abgespeichert werden. Diese Querschnitte sind meist durch einfache geometrische Strukturen (Polygon, Kreis, Ellipse) repräsentiert, welche die lokale Gefäßdicke kodieren. Eine solche Repräsentation transformiert in eine Graphenstruktur, erlaubt beispielsweise die Kennzeichnung von Teilbäumen und gestattet die Anwendung von Analysemethoden der Graphentheorie [GKS+93], [HPSP01].

Um eine Exploration des rekonstruierten Gefäßmodells zu ermöglichen, wird die symbolische Beschreibung der Gefäßstruktur auf Graphikprimitive abgebildet. Im Laufe der letzten zwei Jahrzehnte wurden einige Verfahren entwickelt, welche sich dieser Problematik widmen. Die hier vorgestellten *Surface Reconstruction*-Techniken verbinden benachbarte Gefäßquerschnitte durch Primitive verschiedener Komplexität miteinander.

Tabelle 2.1 zeigt einen Vergleich der in Abschnitt 2.3 vorgestellten modellbasierten Visualisierungsverfahren hinsichtlich der angewandten Primitive und der in Kapitel 1 aufgestellten Anforderungen an eine ideale Visualisierungsmethode für Gefäßstrukturen. Die Verfahren sind nach verwendeten Primitiven geordnet. Da die meisten Veröffentlichungen ihren Schwerpunkt auf Segmentierung und Skelettierung des Gefäßbaums legen, wird die Visualisierung des resultierenden Modells selten ausführlich beschrieben. Daher sind die Ausführungen in Tabelle 2.1 teils spekulativ und aus der Art des verwendeten Primitivs, den Abbildungen in der Veröffentlichung und dem Fehlen gegenteiliger Aussagen geschlossen. Dies ist jedoch stets durch ein Fragezeichen im Feld markiert. Die in [ZJP95] vorgeschlagene direkte Visualisierung der symbolischen Beschreibung modelliert die Gefäßoberfläche nicht und wurde daher in Tabelle 2.1 nicht berücksichtigt.

Die Verwendung von Zylindern [BGSC85], [GKS+93], [MMD96] gestattet keine Abbildung lokaler Änderungen der Gefäßdicke. Abrupte Übergänge und Diskontinuitäten sind entlang der Gefäßäste und vor allem an den Verzweigungen zu beobachten. Dort entstehen außerdem störende Strukturen im Inneren der Gefäße. Der Einsatz von Kegelstümpfen [PTN97], [HPSP01] erlaubt die Abbildung einer kontinuierlichen Gefäßverlaufs. In [PTN97] wird eine Navigation im Inneren der Gefäße vorgeschlagen. Dies spricht gegen dort vorhandene, störende Strukturen, was jedoch nicht explizit erwähnt wurde. Die einzige Methode, welche die Gefäßenden berücksichtigt, wurde durch [HPSP01] entwickelt. Die in [EDKS94] und [FFKW02] präsentierten Verfahren streben

Tabelle 2.1: Vergleich von Verfahren zur modellbasierten Gefäßvisualisierung hinsichtlich der Kriterien an eine ideale Gefäßvisualisierungsmethode: (1) korrekte Abbildung des Gefäßdurchmessers, (2) weiche, organische Gefäßform, (3) einheitliche Behandlung aller Verzweigungstypen, (4) geschlossene Gefäßenden, (5) Vermeidung von Strukturen im Gefäßinneren. Das Fragezeichen ? kennzeichnet eine spekulative Aussage.

Methode	**Primitiv**	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**
[BGSC85], [GKS+93], [MMD96]	Zylinder	bedingt	nein	ja?	nein?	nein?
[PTN97], [HPSP01]	Kegelstumpf (und Kugel, in [PTN97])	ja ja	nein nein	ja? ja	nein? ja	nein? nein
[EDKS94]	Freiformfläche	ja?	ja?	nein?	nein?	ja?
[FFKW02]	Subdivision Surface	ja?	ja	ja	nein	ja

geometrische Kontinuität der Oberfläche sowohl entlang von Gefäßästen, als auch an Verzweigungen an. Eine genaue Evaluierung der Resultate dahingehend fehlt leider in beiden Arbeiten. Weiterhin ist hier unklar, ob der Gefäßdurchmesser korrekt abgebildet wird. Eine Besonderheit der in [EDKS94] vorgestellten Methode ist der Verzicht auf eine Vereinfachung der Form von Gefäßquerschnitten. Während alle anderen Verfahren auf der Annahme kreisrunder Gefäße basieren, kann hier jegliche, in der Skelettierung erschlossene Form abgebildet werden.

Die Untersuchung bisher existierenden Techniken zur modellbasierten Gefäßvisualisierung zeigt, dass die Generierung einer weichen, organischen Gefäßform durch die Sicherstellung geometrischer Kontinuität an den Verzweigungen das größte Problem darstellt. Sie ist entweder nicht gewährleistet (Zeilen 2,3 der Tabelle 2.1) oder erfordert einen beträchtlichen Konstruktionsaufwand (Zeilen 4,5). Diese Erkenntnis und der Wunsch nach einer ringsum glatten Gefäßstruktur motivieren den alternativen Einsatz von impliziten Oberflächen. Die Generierung von weichen, geometrisch kontinuierlichen Übergängen zwischen zwei Objekten **ohne** jeglichen Konstruktionsaufwand ist eine der großen Stärken von impliziten Oberflächen.

3 Implizite Oberflächen

Implizite Oberflächen sind zweidimensionale, geometrische Formen, welche basierend auf einer mathematischen Beschreibung konstruiert werden können. Diese Beschreibung wird als implizite Funktion bezeichnet. Die implizite Spezifikation einer Oberfläche ist oft wesentlich kompakter als ihr parametrisches Gegenstück wie das klassische Beispiel einer Kugel mit Radius 1 zeigt [BBB+97]:

trigonometrisch:	$f(\alpha, \beta) = (\cos\alpha\cos\beta, \sin\alpha, \cos\alpha\sin\beta), \alpha \in [0, \pi], \beta \in [0, 2\pi]$
rational:	$x = 4st/w, y = 2t(1-s^2)/w, z = (1-t^2)(1+s^2)/w, w = (1+s^2)(1+t^2)$
implizit:	$f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$

Obwohl impliziten Oberflächen immer mehr Aufmerksamkeit innerhalb der Computergrafik zuteil wird, ist die Anwendung parametrischer Oberflächen bis heute dominant. Ein Grund hierfür ist die einfache Konvertierung der parametrischen Beschreibung in ein geometrisches Objekt [BBB+97]. In obigem Beispiel genügt zur Berechnung von Oberflächenpunkten das einfache Einsetzen von Werten für die Parameter (α, β) bzw. (s, t) innerhalb des jeweiligen Wertebereichs. Die implizite Beschreibung hingegen erfordert hierzu die oft komplizierte Bestimmung der Nullstellenmenge (*zero-set*) der impliziten Funktion. Diese umfasst alle Punkte $p(x, y, z)$ mit einem Funktionswert gleich 0. Dennoch gewinnen implizite Oberflächen mehr und mehr an Bedeutung aufgrund ihrer Fähigkeit, ein Volumen zu beschreiben und die Verschmelzung (*blending*) solcher Volumen zu repräsentieren. Während die parametrische Spezifikation einer Oberfläche normalerweise aus stückweisen so genannten Füllstücken (*patches*) zusammengesetzt ist, beschreibt die implizite Funktion das gesamte Innere des Objektes.

Eine der großen Stärken impliziter Oberflächen ist die Gewährleistung geometrischer Kontinuität ohne jeglichen Konstruktionsaufwand. *Blends* sind hier die Alternative zu komplizierten parametrischen Füllstücken, Rundungen und anderen komplizierten Freiformflächen. Deren Zusammensetzung muss mit größter Sorgfalt geschehen, um geometrische Kontinuität sicherzustellen. Daher werden in zunehmendem Maße implizite Oberflächen bei der Modellierung und Animation organischer Formen erfolgreich eingesetzt [BBB+97]. Weiterhin gestattet die implizite Beschreibung eine einfach Klassifizierung der Lage von Punkten hinsichtlich der impliziten Oberfläche (*Point Classification*). So kann anhand des Vorzeichens des zu einem Punkt korrespondierenden Funktionswertes bestimmt werden, ob dieser innerhalb, außerhalb oder auf der Oberfläche liegt.

In Anlehnung an [OM95] wird innerhalb dieser Arbeit folgende Schreibweise einer impliziten Funktion genutzt:

$$F(p) - Iso = 0 \tag{3.1}$$

$F(p)$ ist als *Skalarfeldfunktion* (*scalar field function*) oder auch kurz *Feldfunktion* (*field function*) bekannt, da zu jedem Punkt p ein Skalarwert berechnet werden kann. Die Menge aller Punkte p und der jeweils korrespondierenden Skalarwerte wird als *Skalarfeld* bezeichnet. Mit Hilfe des Isowertes Iso können verschiedene implizite Oberflächen konstruiert werden, welche

jeweils Punkte gleichen Skalarwertes miteinander verbinden. Auf das obige Beispiel der Kugel übertragen, können so Kugeln mit unterschiedlichem Radius beschrieben werden.

Im folgenden Abschnitt wird die Historie impliziter Oberflächen zusammen mit einigen Anwendungsgebieten kurz rekapituliert. Abschnitt 3.2 ist der Modellierung mit impliziten Oberflächen gewidmet. Hierzu werden verschiedene Ansätze diskutiert und die jeweiligen Probleme verdeutlicht mit Schwerpunkt auf den in dieser Arbeit verwendeten *Convolution Surfaces*. Diesen ist aus gegebenem Grund mit Abschnitt 3.3 eine weiterführende Abhandlung gewidmet. Abschnitt 3.4 diskutiert die gängigsten Visualisierungsmethoden für implizite Oberflächen. Abschließend werden die Konsequenzen dieses Kapitels für den Entwurf einer Gefäßvisualisierungsmethode gezogen.

3.1 Geschichte und Anwendungen

Der amerikanische Wissenschaftler, Professor, Autor und Schauspieler Carl Sagan (1934-1996) schrieb 1978 zusammen mit der Produzentin Ann Druyan das Skript für „COSMOS“, eine 13-teilige Dokumentationsreihe zu astronomischen und astrologischen Themen [Pou00]. Diese in den 80er Jahren vom Public Broadcasting Service (PBS) in Amerika ausgestrahlte Serie wurde vor allem durch die Verwendung neuer Spezialeffekte berühmt. So konstruierte und animierte der Computergrafiker Jim Blinn beispielsweise ein DNA-Molekül für COSMOS (Abb. 3.1). Er entwickelte eine Methode zur Beschreibung und dynamischen Simulation der Elektronendichtefelder von Atomen. Das Elektronendichtefeld entspricht einem Skalarfeld um das Atomzentrum. Zur Visualisierung dieser Felder führte er das Konzept der impliziten Oberflächen unter dem Namen *Blobby Molecules* (kurz: „Blobs“) in die Computergrafik ein [Bli82].

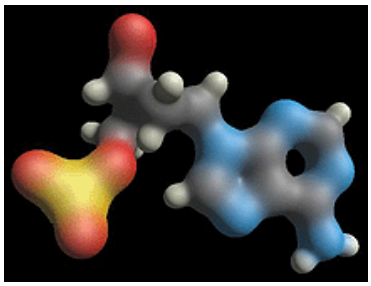


Abbildung 3.1: Adeninmolekül, welches zur Illustration einer DNA Reproduktionssequenz für die PBS Serie COSMOS konstruiert wurde. *Quelle*: [Hal01]

Kurze Zeit später entwickelte eine Gruppe von Wissenschaftlern der Universität von Osaka und der Toyo Links Corporation unter Führung von Koichi Omura so genannte *Metaballs* [NHK+85]. Diese basieren auf demselben Konzept wie *Blobby Molecules*, nutzen aber eine einfachere Funktion zur Beschreibung des Skalarfeldes um eine Punkt. Der Künstler Yoichiro Kawaguchi verwendet die im Laufe dieses Projektes entstandene Software seit vielen Jahren für seine auf der SIGGRAPH gezeigten, spektakulären Kurzfilme und Bilder [Kaw96], [Sch99].

Die Brüder Brian und Geoff Wyvill entwarfen zusammen mit Craig McPheeters ein wenig später die so genannten *Soft Objects* [WMW86]. Basierend auf demselben Prinzip wie *Blobby Molecules* und *Metaballs*, vermeidet die von ihnen eingeführte Funktion zur Berechnung des Skalarfeldes jedoch einige rechenintensive Kalkulationen der Quadratwurzel.

Alle bisher erwähnten Methoden beschreiben das Skalarfeld um einen Punkt im dreidimensionalen Raum. Basierend auf dem Punkt als Modellierprimitiv lassen sich einfache Formen, wie Kugeln oder Ellipsoide, konstruieren. Komplexere Objekte müssen aufwendig aus diesen zusammengesetzt werden. Jules Bloomenthal und Ken Shoemake erweiterten in [BS91] die Modellierung mit impliziten Oberflächen auf Primitive beliebiger Art, z.B. Liniensegmente, Kurven oder Polygone.

Implizite Oberflächen kommen heutzutage in vielen verschiedenen Gebieten der computergestützten Modellierung, Simulation und Animation zum Einsatz. In [TBG95] werden sie zur Rekonstruktion von Organen aller Art und Form aus Punktwolken eingesetzt. *Blobby Molecules* werden in [Bou97] für die Modellierung des menschlichen Cortex aus MRI-Daten genutzt. Die Rekonstruktion eines anatomischen Modells des Thorax basierend auf fuzzy impliziten Oberflächen wird in [LvdGRR+99] beschrieben. Anwendung in der physikalisch basierten Animation finden impliziten Oberflächen z.B. in [SAC+99]. Hier werden sie zusammen mit Partikel-Systemen zur Animation von Lava-Flüssen eingesetzt. Die Simulation von Erdrütschen zur Prävention von Naturkatastrophen wird in [GCD+98] diskutiert. Dabei werden implizite Oberflächen für die Visualisierung von Matsch- und Schlammlawinen verwendet. Die Modellierung von natürlich geschaffenen Formen ist ein weiteres Anwendungsgebiet. In [Blo95b] wird u.a. am Beispiel einer menschlichen Hand die skelettbasierte Modellierung motiviert (Abb. 3.2). Der naturgetreuen Nachbildung einer Auswahl von Meereslebewesen widmet sich [She97]. Weiterhin sind implizite Oberflächen schon in kommerzielle Modellierumgebungen, wie 3D-Studio, und frei erhältliche *Ray Tracing*-Pakete, z.B. *Rayshade*[1] [KB91] und *POV-ray*[2] [Cas91], integriert. Für eine Auflistung zahlreicher weiterer Anwendungsgebiete sei hier auf [OM95] verwiesen.

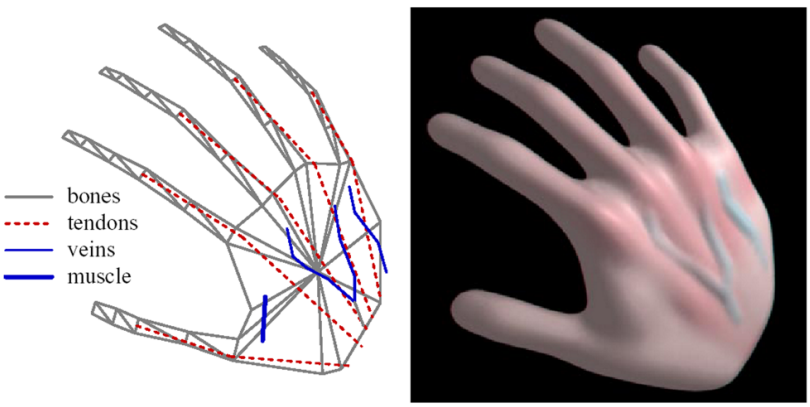


Abbildung 3.2: Skelett einer menschlichen Hand (*links*), bestehend aus Dreiecken und Liniensegmenten. Die korrespondierende *Convolution Surface* (*rechts*). *Quelle*: [Blo95b]

3.2 Modellierung

Die Modelliertechniken auf der Basis impliziter Oberflächen lassen sich nach [OM95] in zwei Gruppen einteilen: punktbasierte Modellierung und skelettbasierte Modellierung. Zur ersten Kategorie zählen alle Methoden, welche Modelle um einen einzelnen Punkt herum konstruieren, z.B.

[1] Erhältlich über ftp: `ftp://graphics.stanford.edu/pub/rayshade/`, Stand: 10.02.04
[2] `http://www.povray.org/download/`, Stand: 10.02.04

Kugeln oder Ellipsoide. Für die Beschreibung komplexer Objekte können Punktmengen definiert werden. Dies ist jedoch mit einem hohen Aufwand verbunden und erfordert eine große Sorgfalt bei der Platzierung der einzelnen Punkte. Die zweite Kategorie umfasst Verfahren, welche das Modell auf der Basis eines Skeletts generieren. Komplexe Objekte können durch ihr Skelett, zusammengesetzt aus einfachen Primitiven, wie Liniensegmente, Kurven oder Polygone, intuitiv beschrieben werden. Durch die in [BS91] eingeführte Technik, wird die Konstruktion eines Modells um Primitive beliebiger Art herum ermöglicht.

3.2.1 Punktbasierte Modellierung

Die punktbasierte Modellierung beschreibt die Konstruktion von Modellen um einen einzelnen Punkt herum. In [Bli82] wird das Elektronendichtefeld eines Atoms durch *Blobby Molecules* visualisiert. Hierzu wird die Verteilung der Elektronendichte um das Zentrum des Atoms (das Skalarfeld) durch folgende Feldfunktion beschrieben:

$$F(p) = be^{-ad^2} \tag{3.2}$$

wobei d die Entfernung ist von Punkt p zum Zentrum des Atoms. Der exponentielle Term beschreibt eine Gauß-Funktion mit dem Zentrum in Entfernung d, der Höhe b und der Standardabweichung a. Von besonderem Interesse ist jedoch die gegenseitige Beeinflussung mehrerer Atome. Eine Beschreibung der Elektronendichteverteilung von i Atomen erfolgt in [Bli82] durch:

$$F(p) = \sum_i b_i e^{-a_i d_i^2} = Iso \tag{3.3}$$

Die Skalarfelder der einzelnen Atome überlappen sich, was durch eine Summation ausgedrückt werden kann. Die implizite Oberfläche ist definiert durch alle Punkte, an denen das resultierende Skalarfeld gleich einem gegebenen Isowert (Iso) ist. Sie beschreibt somit eine Region im Raum mit einheitlicher Elektronendichte (Abb. 3.3). Die Verschmelzung mehrerer implizit definierter Objekte wird *blending* genannt. Die so genannte *Blendingfunktion* kontrolliert diesen Prozess. Im Fall von Blinn's Atomen ist die Blendingfunktion eine einfache Summe.

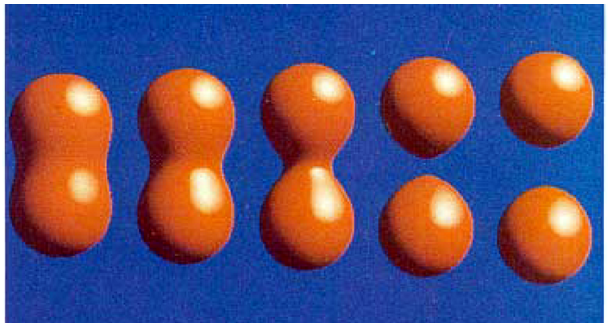


Abbildung 3.3: *Blobby Molecules.* Elektronendichtefelder der Atome überlappen sich, wenn diese nah genug beieinander liegen. Zur Visualisierung der impliziten Oberfläche wurde für alle Entfernungen derselbe Isowert verwendet. *Quelle*: [Bli82]

Metaballs [NHK+85] und *Soft Objects* [WMW86] basieren auf demselben Prinzip, nutzen allerdings andere Funktionen zur Berechnung des Skalarfeldes um einen Punkt. Für *Metaballs* ist die

Feldfunktion gegeben durch:

$$F(p) = \begin{cases} w\left(1 - 3\left(\frac{d}{R}\right)^2\right), & 0 \leq d \leq \frac{R}{3}; \\ \frac{3w}{2}\left(1 - \frac{d}{R}\right)^2, & \frac{R}{3} < d \leq R; \\ 0, & d > R. \end{cases} \tag{3.4}$$

Das „Gewicht“ eines *Metaballs* ist mit w bezeichnet. d entspricht der Distanz zwischen p und dem Punkt, um den der *Metaball* definiert ist. R ist der so genannte *radius of influence* und beschreibt die Ausdehnung des Skalarfeldes um den Punkt p. Die Feldfunktion eines *Soft Objects* berechnet sich wie folgt:

$$F(p) = \begin{cases} 1 - \frac{4}{9}\left(\frac{d}{R}\right)^6 + \frac{17}{9}\left(\frac{d}{R}\right)^4 - \frac{22}{9}\left(\frac{d}{R}\right)^2, & d \leq R; \\ 0, & d > R. \end{cases} \tag{3.5}$$

Die Bedeutungen von d und R entsprechen denen in Gleichung 3.4. Während Gleichung 3.2 auf alle Punkte des Raumes Einfluss hat, unabhängig von d, sind die hier verwendeten Funktionen per Definition gleich 0 ab der Entfernung R. Dies ist wichtig für eine effiziente Berechnung der Feldfunktion an einem Punkt p bei sehr vielen Primitiven. Beide Methoden nutzen polynomielle Funktionen, da diese einfacher zu berechnen sind als der exponentielle Term in Gleichung 3.2. Weitere Funktionen, welche auch zur Beschreibung des Skalarfeldes um einen Punkt genutzt werden können, findet der interessierte Leser in [She99b].

3.2.2 Skelettbasierte Modellierung

Das Skelett eines Objektes ist eine natürliche Abstraktion, um dessen Form charakterisieren und beschreiben zu können [Blu67], [Nev82]. Komplexe Objekte können durch ihr Skelett, zusammengesetzt aus einfachen Primitiven, wie Liniensegmente, Kurven oder Polygone, intuitiv beschrieben werden. In [BS91] wird daher das Konzept der Modellierung mit impliziten Oberflächen auf beliebige Primitive ausgedehnt. Diese werden auch häufig als *Skeletal Primitives* bezeichnet. Mit ihrer Hilfe kann die Oberfläche eines Objektes auf der Basis des Skeletts und zusätzlicher Oberflächeninformation konstruiert werden (Abb. 3.4). Die erweiterte Auswahl an Primitiven beseitigt weiterhin einen Nachteil punktbasierter Modellierung. Deren Schwäche bei der Konstruktion von flachen Oberflächen oder glatten Zylindern wurde bereits in [She98a] erkannt.

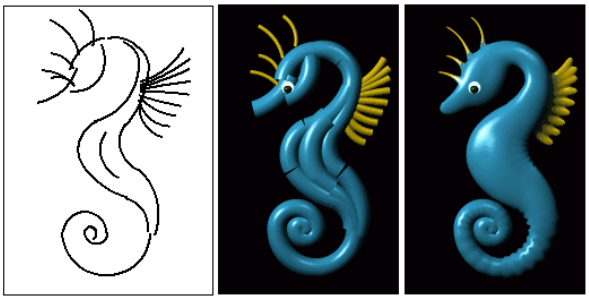


Abbildung 3.4: Seepferdchen. Skelett (*links*) und Modellierung mit expliziten (*Mitte*) bzw. impliziten Oberflächen (*rechts*). *Quelle*: [She97]

In [BS91] werden zwei Techniken zur skelettbasierten Konstruktion eines Modells vorgestellt: *Distance Surfaces* und *Convolution Surfaces*.

3.2.2.1 Distance Surfaces

Das Skalarfeld an Punkt p wird hier aus dem nächstliegenden Punkt des gesamten Skeletts wie folgt berechnet:

$$F(p) = f(S,p) = max_{s \in S} e^{\left(\frac{-\|s-p\|^2}{2}\right)} \tag{3.6}$$

wobei S das Skelett beschreibt und s einen Punkt auf diesem Skelett definiert. $f(S,p)$ ist dort maximal, wo der Abstand zwischen Skelett und Punkt p am geringsten ist. Mit dem max-Operator beschreibt Gleichung 3.6 die Vereinigung (*Union*) der Volumina aller individuellen Punkte des Skeletts. Daher wird diese Art der *Distance Surface* auch als *Union Surface* bezeichnet. Problematisch sind hier Diskontinuitäten der Oberflächennormalen, welche an konkaven Stellen der *Union Surface* auftreten (Abb. 3.5 (rechts)). Diese Artefakte werden auch als Einknicke (*Creases*) bezeichnet.

Die Einführung einer einfachen Blendingfunktion beseitigt jedoch dieses Problem. An Punkt p wird nun aus dem nächstliegenden Punkt jedes Skelettprimitivs jeweils ein Skalarwert berechnet. Wie in Gleichung 3.3 werden die einzelnen Werte dann addiert. Diese Vorgehensweise beseitigt zwar die Diskontinuitäten, schafft aber auch ein neues Problem. An konvexen Stellen der *Distance Surface* sind Verdickungen *Bulges* zu beobachten (Abb. 3.5 (Mitte)).

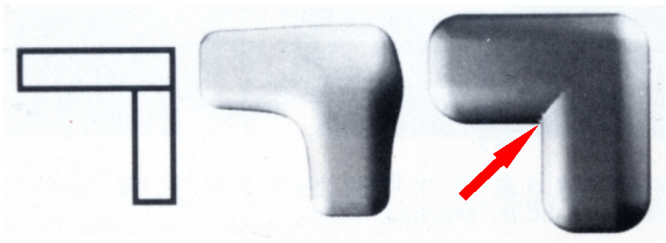


Abbildung 3.5: *Distance Surfaces*. Skelett (*links*), *Blending* (*Mitte*)und *Union Surface* (*rechts*). Der Pfeil markiert Diskontinuitäten an der konkaven Stelle der *Union Surface*. *Quelle*: [BS91]

Ursache hierfür ist, dass die Oberfläche jedes Teilskeletts zwar eine Vereinigung der einzelnen Punktvolumina bildet, die Gesamtoberfläche jedoch durch Summation erzeugt wird. Gegeben seien zwei aneinander grenzende Liniensegmente L_1 und L_2 (Abb. 3.6). Die *Union Surface* wird an der Stelle p ausschließlich unter Berücksichtigung des nächstliegenden Punktes s des gesamten Skeletts berechnet (Abb. 3.6 oben). Unter Verwendung der Blendingfunktion hat jedoch der nächstliegende Punkt jedes einzelnen Skelettteils Einfluss an dieser Stelle (s_1 und s_2 in Abb. 3.6 unten). Die Summe der zu s_1 und s_2 korrespondierenden Skalarwerte ist entlang der Konkatenation von L_1 und L_2 nicht konstant. Das Ergebnis ist eine Verdickung, welche in vielen Anwendungen störend ist. In der Gefäßvisualisierung beispielsweise könnte dieses Artefakt als krankhafte Veränderung, z.B. Aneurysma, interpretiert werden.

3.2.2.2 Convolution Surfaces

Convolution Surfaces lösen das Problem von Diskontinuitäten und Verdickungen für nichtverzweigte Strukturen. Statt nur der nächstliegenden Punkte, werden für die Berechnung des

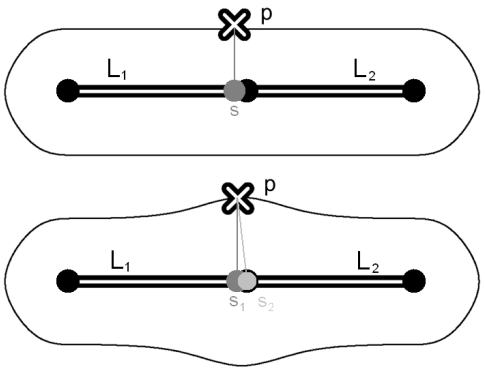


Abbildung 3.6: An konvexen Stellen ist die *Union Surface* artefaktfrei (*oben*). *Blending* resultiert dort jedoch in einer Verdickung (*unten*). Nach [AC02]

Skalarfeldes an der Stelle p alle Skelettpunkte betrachtet. Daher entfällt die Vereinigung und das Resultat ist eine pure Summation:

$$F(p) = f(S,p) = \sum_{s \in S} e^{\left(\frac{-\|s-p\|^2}{2}\right)} \tag{3.7}$$

bzw. eine Integration:

$$F(p) = f(S,p) = \int_S \left(e^{\left(\frac{-\|s-p\|^2}{2}\right)} \right) ds \tag{3.8}$$

Gleichung 3.8 beschreibt die Faltung des Skeletts mit einem dreidimensionalen Gauß-Filter. *Convolution Surfaces* bedienen sich des Konzeptes der Faltung, welches aus der Signalverarbeitung bekannt ist. Hierbei wird ein Signal durch einen Filter modifiziert. Gleichung 3.8 kann für einen Gauß-Filter mit der Höhe 1, Mittelwert 0 und der Standardabweichung 1

$$h(p) = e^{\frac{-\|p\|^2}{2}} \tag{3.9}$$

und unter Berücksichtigung von Gleichung 3.1 wie folgt umgeschrieben werden:

$$f(S,p) = \int_S h(s-p)ds = (h \otimes S)(p) = Iso \tag{3.10}$$

S benennt das Skelett und entspricht hier dem Signal. h ist die Filterfunktion und $\otimes$ kennzeichnet den Faltungsoperator. Das Integral wird als *Faltungsintegral* bezeichnet. Die grundlegende Idee der Verwendung von *Convolution Surfaces* besteht in der Glättung des Skeletts. Hohe Frequenzen sollen sanft in die Umgebung abfallen. Eine dafür geeignete Filterfunktion sollte nach [She98a] folgende Anforderungen erfüllen: Kontinuität, Monotonie, vernachlässigbar kleine Funktionswerte ab einer bestimmten Entfernung von dem Zentrum der Funktion und ein Gradient von annähernd 0 in dieser Entfernung. Mit anderen Worten, sie sollte Tiefpaßcharakteristik besitzen, wie z.B. der Gauß-Filter. Das durch die Glättung resultierend Feld um das Skelett entspricht dem Skalarfeld. Eine implizite Oberfläche ist nun definiert durch alle Punkte, an denen das Skalarfeld gleich einem gegebenen Isowert ist (Abb. 3.7). *Convolution Surfaces* realisieren ein *blending* der Oberflächen um die einzelnen Skelettprimitive durch Integration und bedürfen nicht der gesonderten Spezifikation einer Blendingfunktion.

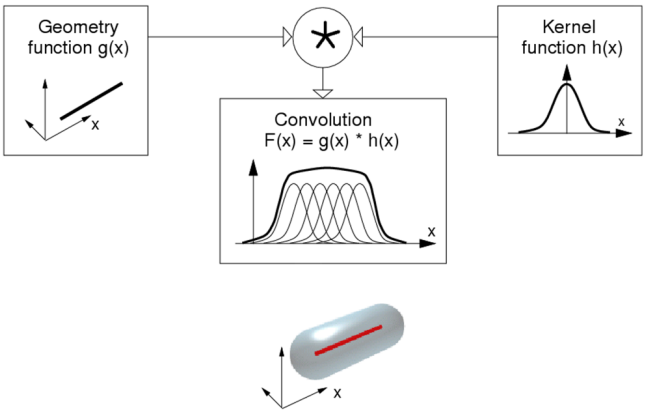


Abbildung 3.7: *Convolution Surfaces.* Basierend auf der Geometrie (*oben links*), einer Filterfunktion (*oben rechts*) und der Skalarfeldfunktion (*Mitte*) wird die *Convolution Surface* (*unten*) modelliert. Das Symbol an der Verbindung zwischen den drei oberen Komponenten kennzeichnet den Faltungsoperator. *Quelle*: [She99a]

Wie eingangs erwähnt, ist die *Convolution Surface* frei von Verdickungen für nicht-verzweigte Strukturen. Dies resultiert aus einer besonderen Eigenschaft der Faltung, der so genannten *Superposition* (Abb. 3.8):

$$h \otimes (S_1 + S_2) = (h \otimes S_1) + (h \otimes S_2) \tag{3.11}$$

> Da die Faltung ein linearer Operator ist, ist die Summe der Faltungen jeder Unterteilung des Skeletts gleich der einfachen Faltung des gesamten Skeletts. Das garantiert beispielsweise, dass zwei aneinander grenzende, kollineare Segmente [...] die gleiche *Convolution Surface* erzeugen wie ein einzelnes Segment welches die Vereinigung der beiden darstellt. [3] [Blo95b]

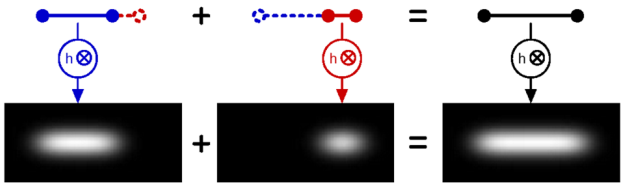


Abbildung 3.8: Superpositions-Eigenschaft der Faltung. Die Faltung des gesamten Liniensegments ist gleich der Summe der Faltungen zweier beliebiger Teilsegmente. *Quelle*: [Blo95b]

Die Invariabilität der *Convolution Surface* bei einer Unterteilung des Skeletts kennzeichnet einen wesentlichen Vorteil gegenüber *Distance Surfaces*. Die Superpositions-Eigenschaft der Faltung ermöglicht zusätzlich eine effiziente Implementierung von *Convolution Surfaces* und ein inkrementelles Design. So kann ein komplexes Modell Stück für Stück konstruiert werden statt einer Betrachtung des Skeletts als Ganzes.

[3] Because convolution is a linear operator, the sum of convolutions of any division of a skeleton is identically equal to the single convolution of the entire skeleton. This guarantees, for example, that two abutting, collinear segments [...] produce the same convolution as does the single segment that is their union.

3.2.3 Bulging

Die skelettbasierte Modellierung mit Hilfe von *Convolution Surfaces* vermeidet die Entstehung von Verdickungen entlang nicht verzweigter Strukturen. Das Skelett vieler komplexer Objekte ist jedoch hierarchisch organisiert und besitzt Verzweigungen unterschiedlicher Ordnung. Für Gefäßbäume beispielsweise sind Verzweigungen bis zur dritten Ordnung (Trifurkation) zu beobachten, d.h. ein Gefäßast teilt sich in drei neue Äste auf. Wie ist das Verhalten der *Convolution Surface* nun entlang solcher Gabelungen? Diese Fragestellung wurde in [Blo95b] anhand eines T-förmigen Skeletts untersucht (Abb. 3.9 links). Schnitte durch die *Convolution Surface* in der x,y-Ebene (Abb. 3.9 Mitte) und x,z-Ebene (Abb. 3.9 rechts) zeigen eine Verdickung entlang der Verbindung von $segment_1$ und $segment_2$.

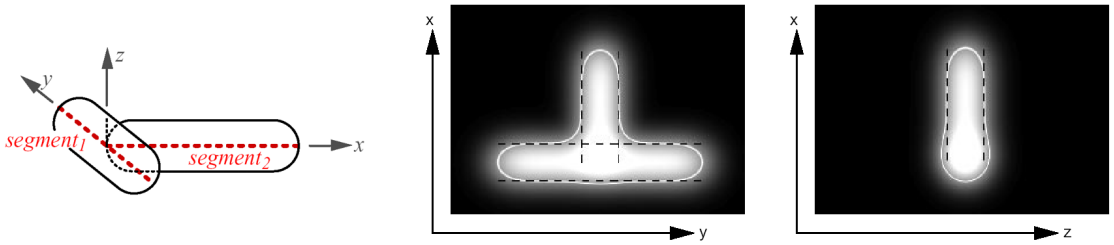


Abbildung 3.9: *Bulging* an Verzweigungen. Ein T-Skelett aus zwei Liniensegmenten (*links*). Schnitte durch die korrespondierende Oberfläche in der x, y-Ebene (*Mitte*) und x, z-Ebene (*rechts*) zeigen eine Verdickung an der Verzweigung. Quelle: nach [Blo95b]

Eine genauere Untersuchung der zugrundeliegenden impliziten Funktion $f(S, p)$ und ihrer Komponenten $f(segment_1, p)$ und $f(segment_2, p)$ verdeutlicht die Entstehung dieser Verdickung. Gegeben sei ein Punkt p, welcher direkt über der Verbindung ($x = 0$) im Abstand r platziert ist. p wird entlang der Linie $(x, 0, r)$ in positiver x-Richtung bewegt. Der Verlauf der impliziten Funktionen $f(segment_1, p)$ und $f(segment_2, p)$, sowie ihrer Summe $f(S, p)$ während der Bewegung von p ist in Abb. 3.10 skizziert. Die Verdickung wird an der Verbindung der beiden Segmente vorausgesagt.

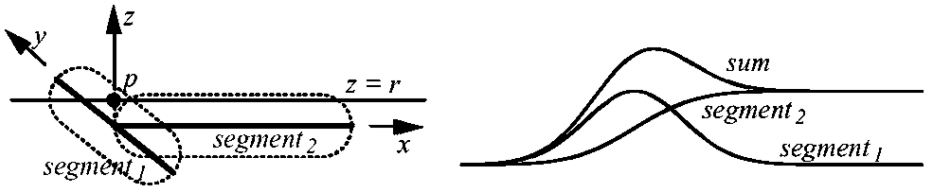


Abbildung 3.10: Untersuchung der impliziten Funktion an einer Verbindung. Punkt p wandert entlang der Linie $(x, 0, r)$ in positiver x-Richtung (*links*). Verhalten der impliziten Funktionen von $segment_1$, $segment_2$ und von deren Summe (*rechts*) während dieser Bewegung. *Quelle*: [Blo95b]

Drei Möglichkeiten zur Verringerung bzw. Beseitigung solcher Verdickungen werden in [Blo95a] vorgestellt und sollen im Folgenden näher erläutert werden. In [FCGA97] wird eine weitere Methode basierend auf kubischen Bézier Dreiecken vorgestellt. Diese berücksichtigt jedoch nur Verzweigungen zweiter Ordnung (Bifurkation).

3.2.3.1 Separierung von Skelettprimitiven

Die räumliche Separierung von Skelettprimitiven verringert den Umfang der Verdickung, beseitigt diese aber nicht vollständig. Der linke Endpunkt von $segment_2$ wird um r nach rechts verschoben, d.h. $segment_2$ wird gekürzt. Die Summe $f(S,p)$ der beiden impliziten Funktionen $f(segment_1,p)$ und $f(segment_2,p)$ ist zwar nicht konstant, variiert aber deutlich weniger (Abb. 3.11 rechts). Wie durch den Verlauf der Funktion $f(S,p)$ angedeutet, werden in der Visualisierung der korrespondierenden Oberfläche eine geringe Verdickung gefolgt von einer kleinen Eindellung sichtbar [Blo95b].

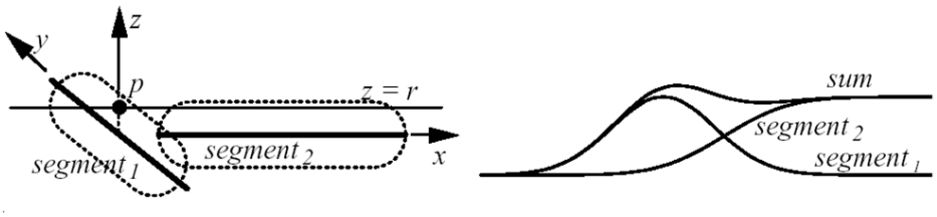


Abbildung 3.11: Verminderung von *bulging* durch Separation der Skelettprimitive. *Quelle*: [Blo95a]

3.2.3.2 Combination Surface

Aus Abb. 3.9 (Mitte) geht hervor, dass *blending* in der Ebene der Verbindung des T-Skeletts die gewünschten weichen Übergänge liefert. Außerhalb der Ebene (Abb. 3.9 (rechts)) muss *blending* jedoch vermieden werden, da sonst Verdickungen auftreten. Basierend auf dieser Beobachtung, wurde in [Blo95b] die *Combination Surface* entwickelt. Durch sie ist eine Kombination aus *Convolution Surface* (*blending*) und *Union Surface* (kein *blending*) beschrieben. Als repräsentativeres Beispiel soll im weiteren Verlauf eine Verzweigung vierter Ordnung dienen (Abb. 3.12).

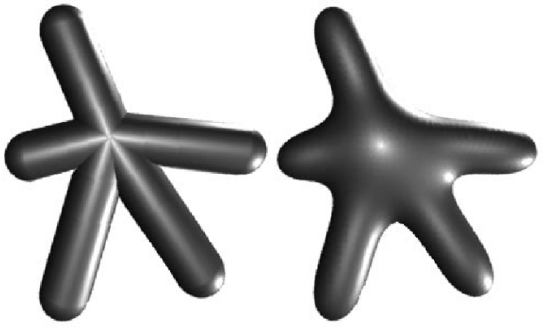


Abbildung 3.12: *Union Surface* (*links*) und *Convolution Surface* (*rechts*) einer Verzweigung aus fünf Primitiven. *Quelle*: [Blo95a]

Die Feldfunktion der *Combination Surface* ist gegeben durch:

$$f_{combination} = convolutionValue + Convexity * (unionValue - convolutionValue) \quad (3.12)$$

Der Einfluss von *Convolution Surface* ($convolutionValue$) und *Union Surface* ($unionValue$) an einem Punkt p wird durch den Faktor $Convexity$ bestimmt. Dieser Wert basiert auf dem Winkel

zwischen der Oberflächennormale der *Convolution Surface* an der Stelle p und der Normale einer Ebene durch die Verzweigung. Letztere ist die Normale eines „fast“ planaren Polygons, welches durch die Endpunkte der abgehenden Segmente aufgespannt wird. Die Normale dieses Polygons kann mit Hilfe von Newell's Methode berechnet werden [Sun02]. Für Punkte p in der Ebene ist $Convexity \approx 0$ und die *Union Surface* hat kaum Einfluss. Liegt p jedoch weit entfernt von dieser Ebene ist $Convexity \approx 1$ und die *Convolution Surface* hat nur geringen Einfluss. Dies ist essenziell für die Vermeidung einer Verdickung (Abb. 3.12). Wie schon in [Blo95b] bemerkt, sind jedoch leichte Diskontinuitäten der Oberflächennormalen an den konkaven Stellen der *Combination Surface* zu beobachten.

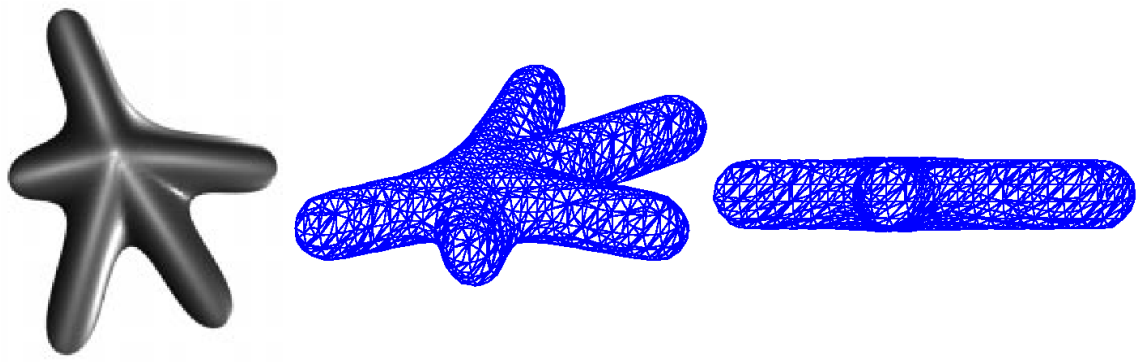


Abbildung 3.13: *Combination Surface. Bulge*-freie Kombination aus *Union Surface* und *Convolution Surface* in Abb. 3.12. Nach [Blo95a] und [Blo95b]

3.2.3.3 Höherdimensionale Skelettprimitive

Eine weitere, in [Blo95a] eingeführte Technik zur Vermeidung von Verdickungen ist der Ersatz i-dimensionaler durch i+1-dimensionale Skelettprimitive. So werden beispielsweise Liniensegmente (1D) durch Polygone (2D) ersetzt Abb. 3.14 (links). Hierbei ist laut [Blo95a] darauf zu achten, daß aneinander grenzende Polygone an ihrer Verbindung eine gemeinsame Fläche formen, statt sich zu überlappen Abb. 3.14 (rechts). Ein Verstoß gegen diese Anforderung resultiert erneut in einer Verdickung entlang einer Verzweigung.

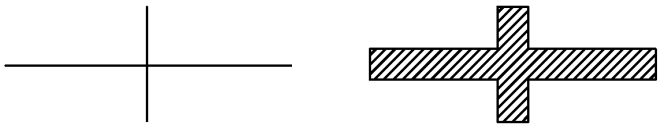


Abbildung 3.14: 2D Skelettprimitive. 1D Primitive (*links*) werden durch Polygone ersetzt (*rechts*). Nach [Blo95a]

Ein weiterer wichtiger Faktor bei der Verwendung von Polygonen ist deren Breite. Sie muss mindestens der Breite des Trägerintervalls der verwendeten Filterfunktion entsprechen, da sonst ebenfalls Verdickungen auftreten [Blo95a]. Diese Anforderung führt jedoch zu einer Formänderung der *Convolution Surface* von kreisförmig zu ellipsoidal. Auch wenn die skelettbasierte Modellierung mit Polygonen die Bildung von Verdickungen komplett unterbindet, so ist doch die Faltung zweidimensionaler Skelettelemente mit beträchtlichem Aufwand verbunden. Ein Beispiel für die Faltung eines Polygons mit einer Filterfunktion ist im Anhang von [Blo95b] gegeben.

3.2.4 Unwanted Blending

In der skelettbasierten Modellierung beschreibt das *Unwanted Blending*-Problem die ungewollte Verschmelzung der Oberflächen nicht aneinander grenzender Primitive (Abb. 3.15). Obwohl dieser Effekt beispielsweise als stilistisches Element in der Animation von Figuren eingesetzt werden kann, ist er doch für die meisten Anwendungen nicht wünschenswert. So würde *Unwanted Blending* im Falle der Gefäßvisualisierung die Topologie eines Gefäßbaums verfälschen. Im Laufe der Jahre wurden einige Methoden zur Prävention von *Unwanted Blending* entwickelt. Zumeist mit Fokus auf eine Modellierumgebung, um dem Designer die Kontrolle der Verschmelzung von Objekten zu ermöglichen.

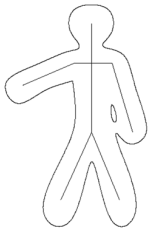


Abbildung 3.15: *Unwanted Blending*-Problem. Nach [TGMC03]

3.2.4.1 Blending-Kategorien

In [OM93] wird zur Vermeidung von *Unwanted Blending* jedem Primitiv ein *Blending*-Typ zugewiesen. Objekte mit gleichem Typ dürfen miteinander verschmelzen. Objekte mit unterschiedlichem *Blending*-Typ können, müssen aber nicht miteinander verschmelzen. Informationen hinsichtlich des *Blending*-Typs können sowohl aus der Skelettstruktur hergeleitet, als auch selbst festgelegt werden. Bei der Berechnung der Feldfunktion an einem Punkt p können nun drei Fälle unterschieden werden: alle p beeinflussenden Objekte verschmelzen miteinander (Abb. 3.16 links), kein Primitiv verschmilzt mit dem anderen (Abb. 3.16 Mitte) oder nur bestimmte Objekte verschmelzen miteinander (Abb. 3.16 rechts).

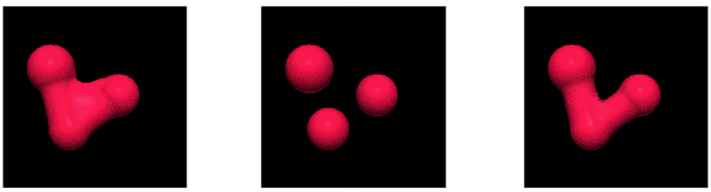


Abbildung 3.16: Unterschiedliche *Blending*-Typen. Alle Objekte verschmelzen miteinander (*links*), kein Objekt verschmilzt mit dem anderen (*Mitte*) und nur bestimmte Objekte verschmelzen miteinander (*rechts*). Nach [OM93]

In [OM93] wird neben der Verschmelzung von Objekten auch eine Kontaktmodellierung betrachtet. Objekte, die nicht miteinander verschmelzen sollen, deren Skalarfelder sich jedoch überlappen, schneiden sich. Alternativ soll die Kontaktstelle modelliert werden. Im Folgenden wird die Herangehensweise für zwei Objekte beschrieben. Die Modellierung des Kontaktes mehrerer Objekte erfolgt analog.

Nachdem die beiden kollidierenden Primitive bestimmt worden, wird zunächst das Primitiv i mit dem größten Einfluss an der Stelle p bestimmt. Für das andere Primitiv j wird eine Deformation durch die *contraction function* c_j beschrieben:

$$c_j(p) = Iso - F_j(p) \tag{3.13}$$

$F_j(p)$ kennzeichnet hier die Feldfunktion von Primitiv j. c_j ist negativ, damit das Objekt gequetscht wird. Der finale Skalarwert an der stelle p berechnet sich wie folgt:

$$F(p) = F_i(p) + Iso - F_j(p) \tag{3.14}$$

Der Kontakt zwischen zwei *Blobby Objects* ist in Abb. 3.17 dargestellt.

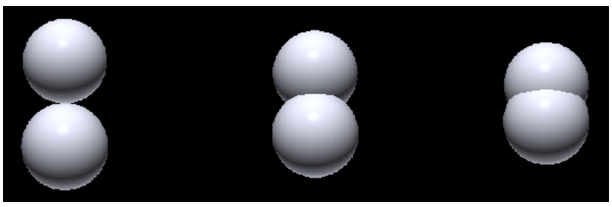


Abbildung 3.17: Kontaktmodellierung. Zwei *Blobs* kommen in Kontakt. Statt zu verschmelzen, verformt sich eines der beiden Objekte. *Quelle*: nach [OM93]

Die Kontaktstellen wirken abrupt und sehen unnatürlich aus, da die Oberfläche dort nicht C^1-stetig ist [OM93], [AC02].

3.2.4.2 Local Convolution

In [CH01] wird eine Lösung des *Unwanted Blending*-Problems vorgeschlagen, welche zusätzlich C^1-Stetigkeit entlang der gesamten Oberfläche gewährleistet. Alle in [CH01] konstruierten Modelle basieren auf einem Skelett aus so genannten *Subdivision-Curves*. Solche Kurven erlauben eine einfache *Level-of-Detail*-Darstellung des korrespondierenden Modells und werden durch Liniensegmente beliebig genau approximiert. Das hier vorgestellte Verfahren wird von den Autoren als *Local Convolution* bezeichnet. Hiernach darf für die Berechnung der Feldfunktion an einem Punkt p nur eine begrenzte, zusammenhängende Region des Skeletts betrachtet werden. Andere Liniensegmente, welche im Bezug auf die Topologie des Skeletts weit von p entfernt liegen, haben dort einen vernachlässigbar kleinen Einfluss. Soll nun der Skalarwert an der Stelle p berechnet werden, so wird zuerst das Liniensegment mit dem größten Einfluss bestimmt. Ausgehend von diesem Segment, werden rekursiv benachbarte Segmente entlang des Skeletts betrachtet. Die Rekursion bricht ab, wenn der Einfluss vernachlässigbar klein ist. Durch diese Methode ist es möglich, *Unwanted Blending* an weitläufigen Knicken des Skeletts zu vermeiden (Abb. 3.18).

C^1-Stetigkeit wird dadurch garantiert, dass Primitive nur dann nicht berücksichtigt werden, wenn ihr Einfluss ohnehin vernachlässigbar klein ist. Ein Nachteil des Verfahrens ist jedoch, dass *Unwanted Blending* entlang sehr kurzer Knicke, geformt aus wenigen Skelettprimitiven, weiterhin auftritt [CH01]. Hier schließt der durch Rekursion bestimmte Skelettabschnitt den gesamten Knick ein.

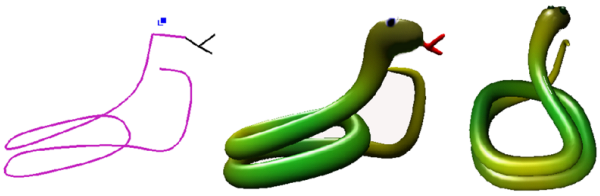


Abbildung 3.18: *Local Convolution. Unwanted Blending* wird vermieden, wenn die Schlange sich kringelt. *Quelle*: [CH01]

3.2.4.3 Verbesserte Local Convolution

Die Vermeidung des *Unwanted Blending*-Problems auch entlang sehr kurzer Knicke des Skeletts wird in [AC02] diskutiert. Der zu betrachtende Skelettabschnitt wird hier noch weiter eingegrenzt. Nur noch das Segment mit dem größten Einfluss sowie seine direkte Nachbarn werden betrachtet. Für den Punkt p wird hier das Skelettprimitiv S_i bestimmt, welches den größten Einfluss hat. Dann wird die Projektion p' von p auf S_i berechnet. $u \in [0, 1]$ sei der Parameter zu p' auf Liniensegment S_i. Mit Hilfe von u und einer Funktion $\alpha(u)$ wird nun der Einfluss der zu S_i direkt benachbarten Segmente S_j an der Stelle p gewichtet. Die Funktion $\alpha(u)$ wird so gewählt, dass S_j einen geringeren Einfluss an der Stelle p besitzt, je weiter p' von S_j entfernt ist. Sie sollte langsam abfallen, um die Stetigkeit der Oberfläche zu gewährleisten [AC02]. Die Verbesserung des ursprünglichen *Local Convolution*-Verfahrens vermeidet *Unwanted Blending* auch an sehr engen Knicken (Abb. 3.19). Zusätzlich bietet sie eine Effizienzsteigerung bei der Berechnung der Skalarwerte, da nur noch sehr kleine Abschnitte des Skeletts betrachtet werden müssen.

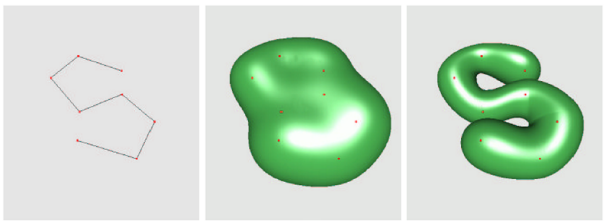


Abbildung 3.19: Verbesserte *Local Convolution*. Vergleich der Lösungen für das *Unwanted Blending*-Problem aus [CH01] (*Mitte*) und [AC02] (*rechts*) an Skelettknicken, bestehend aus wenigen Primitiven. *Quelle*: [AC02]

3.3 Convolution Surfaces im Detail

Zu Beginn von Abschnitt3.2 wurden die punktbasierte und die skelettbasierte Modellierung vorgestellt. Letztere scheint im Rahmen der Gefäßvisualisierung geeignet, da die zu visualisierende Gefäßstruktur gerade durch ihr Skelett beschrieben wird (siehe Abschnitt 2.2). Mit *Distance Surfaces* und *Convolution Surfaces* wurden zwei verschiedene skelettbasierte Ansätze vorgestellt. Im Gegensatz zu *Distance Surfaces* vermeidet die Modellierung mit *Convolution Surfaces* sowohl Diskontinuitäten der Oberflächennormalen, als auch *Bulging* entlang der Oberfläche nicht-verzweigter Strukturen. Dies entspricht den Anforderungen an eine ideale Gefäßvisualisierungsmethode aus Kapitel 1.

Nach der Einführung von *Convolution Surfaces* im vorigen Abschnitt sollen im Folgenden wichtige Details dieser Form der skelettbasierten Modellierung diskutiert werden. Dies geschieht mit Hinblick auf eine Anwendung innerhalb der Gefäßvisualisierung und die Struktur des dort gegebenen Gefäßskeletts. Das Skelett ist durch Liniensegmente approximiert, welche die Mittelpunkte benachbarter Skelettvoxel miteinander verbinden. Jedem Endpunkt eines solchen Segments ist die Radiusinformation des korrespondierenden Voxels zugeordnet. Dies definiert einen linearen Durchmesserverlauf der Gefäßoberfläche.

Ein wichtiger Aspekt bei der Modellierung mit *Convolution Surfaces* ist die Wahl einer geeigneten Filterfunktion. Verschiedene Funktionen wurden in diesem Kontext bereits in der Literatur diskutiert [She99b] und werden in Abschnitt 3.3.1 vorgestellt.

Entscheidend für die Erfüllung der Anforderungen an eine ideale Gefäßvisualisierungsmethode ist die korrekte Abbildung des gegebenen Durchmesserverlaufs. Die *Convolution Surface* eines Liniensegments besitzt jedoch initial eine konstante Dicke. Abschnitt 3.3.2 ist daher der Abbildung eines variierenden Durchmesserverlaufs durch *Convolution Surfaces* gewidmet. Hier wird deutlich werden, dass die Wahl einer adäquaten Filterfunktion durch die Anforderung einer korrekten Abbildung beeinflusst wird.

3.3.1 Filterwahl

Eine geeignete Filterfunktion sollte nach [She98a] Tiefpaßcharakteristik besitzen. In [She99b] werden sieben in der Literatur zu *Convolution Surfaces* diskutierte Filterfunktionen hinsichtlich ihrer Eignung für die Faltung von Primitiven verschiedener Art untersucht. Diese Funktionen sind:

1. **Gauß:** siehe Gleichung 3.2; [Bli82], [BS91]
2. **Cauchy:** $h(x) = 1/(1 + s^2x^2)^2, x > 0$ (s kontrolliert die Breite des Filters); [MS98]
3. **Inverse:** $h(x) = 1/x, x > 0$; [WvO96]
4. **Inverse Quadratische:** $h(x) = 1/x^2, x > 0$
5. **Metaballs:** siehe Gleichung 3.4; [NHK$^+$85]
6. **Soft Objects:** siehe Gleichung 3.5; [WMW86]
7. **Biquadratisches Polynom:** $h(x) = \begin{cases} (1 - \frac{x}{R}^2)^2, & x \leq R; \\ 0, & x > R. \end{cases}$ [She99b]
   (R kontrolliert die Breite des Filters)

Da alle polynomiellen Filterfunktionen (*Metaballs*, *Soft Objects* und biquadratisches Polynom) einen sehr ähnlichen Funktionsverlauf aufweisen, wird in [She99b] beispielhaft das biquadratische Polynom untersucht. Dessen mathematische Formulierung ist sehr einfach und bereits standardmäßig in frei erhältliche *Ray Tracing*-Pakete, wie *POV-Ray* [Cas91] und *Rayshade* [KB91] integriert. Die Filterfunktionen werden hinsichtlich der folgenden zwei Kriterien untersucht:

- Existenz einer abgeschlossenen Lösung (*closed-form solution*) des Faltungsintegrals 3.10
- Berechnungskomplexität des Integrationsresultats (die Feldfunktion)

Eine Lösung ist nach [Cho99] abgeschlossen, wenn sie durch eine endliche Kombination aus konstanten Funktionen, Feldoperationen $(+, -, *, /, **, \sqrt{})$, algebraischen, Exponential- sowie Logarithmusfunktionen und der jeweiligen Inversen unter wiederholter Komposition ausgedrückt werden kann. Die Faltung folgender Primitive wurde in [She99b] eruiert: Punkt, Liniensegment, Ebene, Kurve und Dreieck. Im Hinblick auf die eingangs beschriebene Struktur des Gefäßskeletts beschränkt sich die folgende Beleuchtung der Untersuchungsergebnisse auf das Liniensegmentprimitiv. Nach [She98a] liefert die Faltung mit jeder der betrachteten Filterfunktion hier eine abgeschlossene Lösung. Aus streng mathematischer Sicht [Cho99] ist dies jedoch nicht korrekt, da die Lösung des Faltungsintegrals unter Verwendung des Gauß-Filters die Fehlerfunktion $erf(x)$ enthält. Diese Funktion kann z.B. mittels der Taylor-Reihe, abgeschätzt werden. Sie lässt sich dagegen nicht durch eine endliche Anzahl der oben erwähnten einfachen Funktionen ausdrücken. Die Kontaktaufnahme mit dem Autor von [She99b], Andrei Sherstyuk, hat ergeben, dass er eine andere Definition der Abgeschlossenheit während seiner Untersuchung herangezogen hat. Er betrachtet das Integrationsresultat als abgeschlossen, wenn es ausschließlich „algebraische Operationen $(+, -, *, etc)$ und Standardfunktionen, wie $\sin, \cos, erf()$, beinhaltet. Dies bedeutet, dass die Feldfunktion mit einer maschinellen Gleitkommapräzision berechnet werden kann. In einem Computer werden alle nicht-algebraischen Operationen durch Mikroprogramme berechnet, welche z.B. eine Taylor-Zerlegung nutzen und das Durchlaufen von Schleifen erfordern. Dies ist unvermeidbar, da elementare Maschinenbefehlsätze niemals $\sin, \cos, etc$ enthalten. [Sherstyuk]“

Die Berechnungskomplexität der Feldfunktionen wird in [She99b] anhand der Anzahl enthaltener Gleitkommaoperationen $(+, -, *, /)$ und Spezialfunktionen (trigonometrisch, exponential, logarithmisch,...) beurteilt. Zusätzlich wird die Zeit gemessen, welche für die Faltung eines Liniensegments mit dem jeweiligen Filter notwendig ist. Dazu wurde das Liniensegment in einem Voxelgitter der Dimension 150^3 platziert. Die Evaluierung der Feldfunktion an jedem Knotenpunkt des Gitters resultiert in mehr als drei Millionen Berechnungen. Die benötigte Rechenzeit auf einem 90 MHz Pentium Prozessor variiert zwischen $15,14s$ (Polynom) und $48,2s$ (Gauß). Die restlichen Plätze in absteigender Reihenfolge wurden wie folgt vergeben: $28.62s$ (inverse quadratische Funktion), $29.54s$ (inverse Funktion) und $40.79s$ (Cauchy).

Da die Anwendung aller Filterfunktionen eine abgeschlossene Lösung des Faltungsintegrals für Liniensegmente liefert, kann die Wahl eines geeigneten Filters anhand der Berechnungskomplexität getroffen werden. Hier liegt das biquadratische Polynom an erster Stelle. Die Verwendung dieser Filterfunktion erfordert jedoch besondere Sorgfalt. Aufgrund der gegen $-\infty$ bzw. $+\infty$ konvergierenden Ausläufer (Abb. 3.20) ist die Eingrenzung des Definitionsbereiches auf $x \in [-R, +R]$ zu beachten. Eine Vorgehensweise hierzu wird in [She99b] beschrieben.

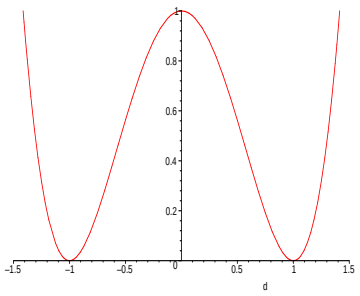


Abbildung 3.20: Biquadratisches Polynom mit $R = 1$. Die infiniten Ausläufer des Polynoms müssen bei der Verwendung als Filterfunktion berücksichtigt werden.

Neben der Wahl einer geeigneten Filterfunktion, ist die korrekte Abbildung eines gegebenen Durchmesserverlaufs durch die *Convolution Surface* von entscheidender Bedeutung für die Gefäßvisualisierung. Initial hat die gesamte *Convolution Surface* entlang eines Liniensegments einen einheitlichen Durchmesser. Aus der Gefäßanalyse ist jedoch für jedes Segment eine lineare Durchmesserverteilung bekannt. Der folgende Abschnitt ist daher der Generierung einer *Convolution Surface* mit variierendem Durchmesser gewidmet.

3.3.2 Durchmesserverlauf

In der Vergangenheit wurden einige Verfahren zur Generierung einer *Convolution Surface* mit variierendem Durchmesser veröffentlicht. So wird in [She99a] die Berücksichtigung einer Profilfunktion entlang eines Liniensegments vorgeschlagen (Abb. 3.21).

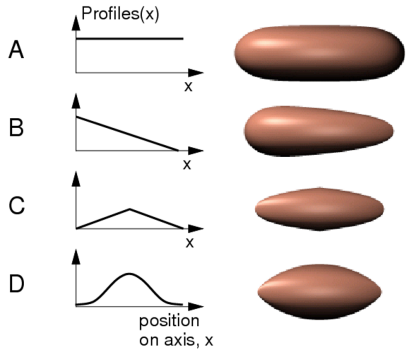


Abbildung 3.21: *Convolution Surface* eines Liniensegments modifiziert durch Profilfunktionen: konstante Funktion (A), lineare Funktion (B), Hutfunktion (C), $\sin^2 x\pi$ (D).
Quelle: [She99a]

Der Wert der Feldfunktion an einem Punkt p wird dem Funktionswert der Profilfunktion an der Stelle p skaliert. Zur Abbildung eines linearen Durchmesserverlaufs wird eine lineare Profilfunktion festgelegt (Funktion (B) aus Abb. 3.21). Diese Vorgehensweise generiert eine *Convolution Surface* mit variierendem Durchmesser. Es bleibt jedoch unklar, wie die Profilfunktion und auch der Isowert für die exakte Abbildung eines gegebenen Durchmesserverlaufs gewählt werden müssen. Allerdings kann mit Hilfe von Profilfunktionen eine hohe Formvariabilität erzielt werden.

Eine weitere Möglichkeit der Durchmesservariation wird in [JTFP01] und [JT02] vorgeschlagen. Das vorrangige Ziel hier ist, den Durchmesserverlauf entlang eines Liniensegments an beliebig vielen Punkten variabel zu gestalten. Entlang des Segments wird zuerst eine polynomielle Wichtungsfunktion definiert. Der Grad des Polynoms bestimmt die Anzahl der Stellen, an denen der Durchmesser entlang des Segments verändert werden kann. Ein Polynom dritten Grades ermöglicht beispielsweise vier Einstellpunkte. Diese Größe hat sich in der Praxis als ausreichend erwiesen [JTFP01]. Für eine lineare Durchmesserverteilung wäre eine Wichtungsfunktion ersten Grades ausreichend, da diese zwei Einstellmöglichkeiten liefert. Um die komfortable Einstellung des Durchmessers innerhalb einer Modellierumgebung zu gestatten, wird die Wichtungsfunktion in der Bézier-Form mit vier Kontrollpunkten angegeben (Abb. 3.22).

Die korrekte Abbildung eines gegebenen Durchmesserverlaufs wird auch durch diese Methode nicht erreicht, da die Kontrollkurve das Profil der entstehenden *Convolution Surface* nur approximiert und nicht direkt auf der Oberfläche verläuft [JTFP01]. Die Manipulation des Durchmessers

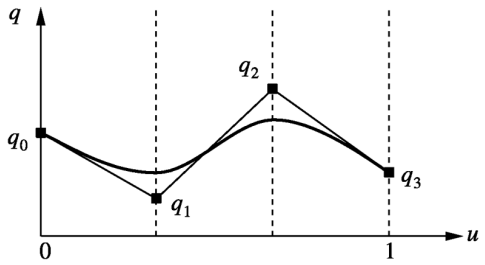


Abbildung 3.22: Kubische Kontrollkurve zur Variation des Durchmesserverlaufs entlang eines Liniensegments. u entstammt der Parametrisierung des Segments. Die Kontrollpunkte sind durch q_0 bis q_3 gegeben. *Quelle*: [JT02]

entlang eines Liniensegments mittels mehrerer Kontrollpunkte eröffnet dem Modellierer jedoch neue Möglichkeiten (Abb. 3.23).

Abbildung 3.23: *Convolution Surface* eines T-Skeletts. Entlang des vertikalen Liniensegments wurden vier Radien spezifiziert. *Quelle*: [JTFP01]

Beide bisher vorgestellten Methoden wurden für die Integration in eine Modellierumgebung entwickelt. Hier steht das Erreichen einer möglichst hohen Formvielfalt im Vordergrund. Die korrekte Abbildung eines gegebenen Durchmesserverlaufs wird nicht angestrebt. Dies ist jedoch essentiell zur Erfüllung der Anforderungen aus Kapitel 1 an eine ideale Gefäßvisualisierungsmethode.

Die bislang veröffentlichten Methoden zur korrekten Abbildung einer gegebenen Durchmesserdistribution modifizieren entweder die aus der Faltung resultierende Feldfunktion [Blo95b], [CH01] nachträglich oder integrieren den Durchmesser bzw. Radius vor der Faltung in die Filterfunktion [HAC03].

3.3.2.1 Abbildung eines gegebenen Durchmesserverlaufs nach Bloomenthal

In [Blo95b] wird der Gauß-Filter aus Gleichung 3.9 für die Faltung des Skeletts verwendet. Das korrespondierende Faltungsintegral ist durch Gleichung 3.8 gegeben. Die Faltung entspricht der Integration eines dreidimensionalen Gauß-Filters entlang des gesamten Skeletts.

Evaluierung des Faltungsintegrals

Zur Berechnung des Faltungsintegrals kann das Skelett aufgrund der Superpositions-Eigenschaft der Faltung in einzelne Primitive zerlegt werden. Diese Primitive können nach [BS91] beliebiger Art sein. Jedoch ist schon die Faltung eines einfachen Polygons nicht ohne weiteres möglich,

da hier in zwei „Richtungen“ integriert werden muss. Die Annäherung durch eine genügend dichte Menge von Punktprimitiven ist sehr ineffizient und ungeeignet für die Konstruktion glatter, flacher Oberflächen [She98b]. Wie in [Hec86] beobachtet wurde, ist der Gauß-Filter jedoch separabel. Gleichung 3.9 kann daher wie folgt umgeschrieben werden:

$$h(p) = e^{\frac{-\|p\|^2}{2}} = e^{\frac{-(p_x^2+p_y^2+p_z^2)}{2}} = e^{\frac{-p_x^2}{2}} e^{\frac{-p_y^2}{2}} e^{\frac{-p_z^2}{2}} \tag{3.15}$$

In Anlehnung an Gleichung 3.8 kann die dreidimensionale Faltung des Skelettprimitivs somit durch drei eindimensionale Integrationen des Gauß-Filters wie folgt formuliert werden:

$$f(S,p) = \int_S \left(e^{\left(\frac{-(s_x-p_x)^2}{2}\right)} \right) ds_x \int_S \left(e^{\left(\frac{-(s_y-p_y)^2}{2}\right)} \right) ds_y \int_S \left(e^{\left(\frac{-(s_z-p_z)^2}{2}\right)} \right) ds_z \tag{3.16}$$

Für die Faltung eines in die x, y-Ebene projizierten Polygons sind somit zwei eindimensionale Integrationen notwendig. Die Integration in z-Richtung wird auf eine triviale Berechnung der Gauß-Funktion reduziert, da das Polygon dort eine unendlich kleine Dicke besitzt. Für die Faltung eines auf die x-Achse, zwischen Ursprung und $(a, 0, 0)$ projizierten Liniensegments der Länge a ist eine eindimensionale Integration in x-Richtung erforderlich. Die Integrationen in y- und z-Richtung werden auf eine einfache Berechnung der Gauß-Funktion reduziert, da das Liniensegment dort eine unendlich kleine Dicke besitzt. Gleichung 3.16 lässt sich für das Liniensegment wie folgt umschreiben:

$$f(S,p) = \int_0^a \left(e^{\left(\frac{-(s_x-p_x)^2}{2}\right)} \right) ds_x e^{\frac{-(p_y^2+p_z^2)}{2}} \tag{3.17}$$

Nach [Blo95b] wird der erste Term (das Integral) als *Integrationsfilter* und der zweite Term (die e-Funktion) als *Distanzfilter* bezeichnet. Die Evaluierung des Faltungsintegrals ist also reduziert auf die Berechnung eines eindimensionalen Integrals und eine einfache Evaluierung des Gauß-Filters. Die Multiplikation der beiden Terme beschreibt den finalen Skalarwert an der Stelle p. Da der Gauß-Filter symmetrisch ist, kann diese Vereinfachung für Liniensegmente beliebiger Position und Orientierung im Raum vorgenommen werden. Zur Illustration der Generalisierung siehe Abb. 3.24.

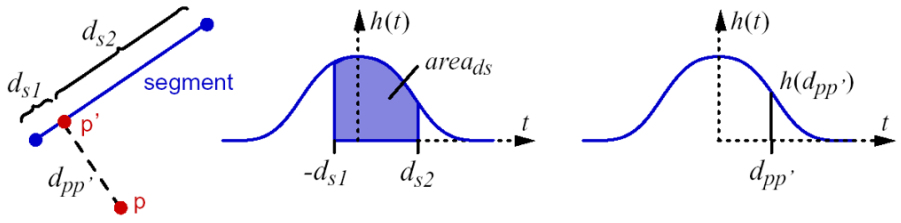


Abbildung 3.24: Evaluierung der Faltung eines Liniensegments. Die Projektion p' von Punkt p auf das Liniensegment wird berechnet und die Integrationsgrenzen d_{s1} und d_{s2} werden bestimmt (*links*). Der Wert des Integrationsfilters korrespondiert zu der Fläche $area_{ds}$ (*Mitte*). Die Berechnung des Distanzfilters entspricht einer einfachen Evaluierung der Gauß-Funktion (*rechts*). *Quelle*: [Blo95b]

Für die Berechnung des Integrationsfilters wird zuerst der Punkt p auf die Linie projiziert, welche das Liniensegment enthält. Diese Projektion sei p'. Das Zentrum des Gauß-Filters wird nun an diesem Punkt ausgerichtet. Die Integrationsgrenzen sind bestimmt durch die Entfernung

d_{s1} von p' zu dem nächstgelegenen Segmentendpunkt und die Entfernung d_{s2} von p' zu dem verbleibenden Endpunkt. Die Integration des Filters von $-d_{s1}$ bis d_{s2} entspricht der Fläche $area_{ds}$. Die Evaluierung des Distanzfilters erfordert eine einfache Berechnung der Gauß-Funktion für die euklidische Distanz $d_{pp'}$ von p zu p'. Die Multiplikation von Integrations- und Distanzfilter ergibt den finalen Skalarwert an der Stelle p.

Bevor die Durchmesserabbildung näher untersucht wird, scheint es notwendig einen wichtigen Aspekt der Signalverarbeitung näher zu erläutern. Gegeben sei eine allgemeinere Form des Gauß-Filters aus Gleichung 3.9:

$$h(p) = e^{-\omega \ \|p\|^2} \tag{3.18}$$

ω ist der so genannte *width coefficient* und entspricht $1/(2\sigma^2)$. Eine Gauß-Funktion mit $\omega = \pi$ hat einen Integral der Größe 1. Diese Eigenschaft ist für die Faltung eines Signals mit einer Filterfunktion sehr wichtig, da die ursprüngliche Signalintensität hierdurch erhalten bleibt.

Durchmesserabbildung

Für die Abbildung eines gegebenen Durchmesserverlaufs schlägt [Blo95b] eine Modifikation des Distanzfilters vor. Dieser bestimmt die Dicke der *Convolution Surface*. Gegeben sei ein auf die x-Achse projiziertes Liniensegment der Länge a. Die Modifikation des Distanzfilters besteht aus einer Normalisierung der Distanz $d_{pp'} = \sqrt{p_y^2 + p_z^2}$ gegen den linear interpolierten Radius r an der Stelle p':

$$h(d_{pp'}) = e^{\frac{-\omega(p_y^2+p_z^2)}{r^2}} \tag{3.19}$$

Mit p sei ein Punkt gegeben, welcher auf der *Convolution Surface* an der Stelle $x = a/2$ lokalisiert ist. a sei ausreichend groß, so dass das Trägerintervall der Filterfunktion weitestgehend abgedeckt wird. Der Integrationsfilter an der Stelle p ist somit ≈ 1. Für Gleichung 3.17 ergibt sich daraus:

$$f(S,p) = e^{\frac{-\omega(p_y^2+p_z^2)}{r^2}} \tag{3.20}$$

Das Resultat ist eine pure Evaluierung des Distanzfilters. Da p direkt auf der Oberfläche liegt, ist die Distanz $\sqrt{p_y^2 + p_z^2}$ zu der korrespondierenden Projektion p' gleich r. Somit gilt:

$$f(S,p) = e^{\frac{-\omega(r^2)}{r^2}} = e^{-\omega} \tag{3.21}$$

Für die korrekte Abbildung des Durchmesserverlaufs durch die *Convolution Surface* müssen zusätzlich ein Isowert und der *width coefficient* festgelegt werden. In [Blo95b] wird gefordert, dass die *Convolution Surface* durch die isolierten Endpunkte eines Segments verläuft. Der Distanzfilter nimmt an einem Endpunkt den Wert 1 an. Der Integrationsfilter hat dort, unter Verwendung einer Gauß-Funktion mit Integral 1, den Wert 1/2 (Abb. 3.25). Somit gilt für Gleichung 3.17: $f(S,p) = 1/2 * 1 = 1/2$. Aus Gleichung 3.1 folgt, dass eine mit Hilfe von Isowert $Iso = 1/2$ konstruierte Oberfläche durch die isolierten Segmentendpunkte verläuft. Durch Einsetzen der Feldfunktion 3.21 in Gleichung 3.1 erhält man:

$$e^{-\omega} - Iso = 0 \tag{3.22}$$

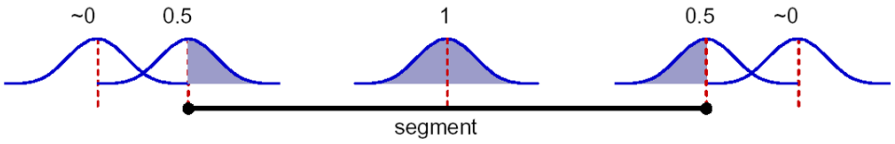


Abbildung 3.25: Der Wert des Integrationsfilters an verschiedenen Punkten eines Liniensegments. *Quelle*: [Blo95b]

Mit Isowert $Iso = 1/2$ resultiert hieraus ein *width coefficient* $\omega = \ln 2$. Da ein Gauß-Filter mit diesem *width coefficient* keinen Integral gleich 1 mehr besitzt, müssen Integral- und Distanzfilter skaliert werden, um eine Veränderung der Signalintensität zu vermeiden [Blo95b].

Isowert und *width coefficient* stehen in engem Zusammenhang bei der korrekten Abbildung eines gegebenen Durchmesserverlaufs. Wird einer der beiden verändert, so muss der andere Wert unter Berücksichtigung von Gleichung 3.22 entsprechend angepasst werden. Für einen *width coefficient* $\omega = ln2$ besitzt die mit Isowert $Iso = 1/2$ konstruierte Oberfläche den korrekten Radius r an der Stelle p.

3.3.2.2 Abbildung eines gegebenen Durchmesserverlaufs nach Cani

Zusätzlich zu den in [She99b] untersuchten Filterfunktionen wird in [CH01] die inverse kubische Funktion im der Modellierung mit *Convolution Surfaces* vorgestellt:

$$h(x) = \frac{1}{x^3}, x > 0 \tag{3.23}$$

Die Funktion ist in Abb. 3.26 veranschaulicht.

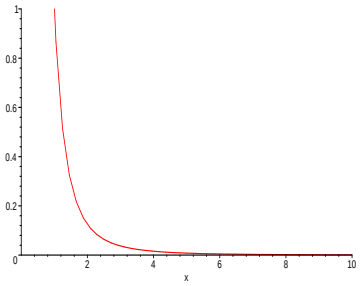


Abbildung 3.26: Die inverse kubische Funktion $h(x) = 1/x^3, x > 0$.

Bei einer Faltung des Liniensegmentes L mit $h(x)$ beschreibt x die Distanz zwischen einem Punkt p und den Punkten auf L. Das Quadrat dieser Distanz kann nach einer Parametrisierung des Segments L mit der Länge l wie folgt ausgedrückt werden [JTFP01]:

$$d^2(t) = \| \vec{v} \|^2 + t^2 - 2t\vec{v} \bullet \vec{a} \tag{3.24}$$

wobei $t \in [0, l]$ der Parametrisierung von L entstammt. $\vec{v}$ ist der Vektor von dem Punkt p zu dem Startpunkt des Liniensegments. Der Vektor vom Startpunkt des Liniensegments zu dessen Endpunkt wird mit $\vec{a}$ bezeichnet. $\bullet$ entspricht dem Skalarprodukt zweier Vektoren. Das

Faltungsintegral kann nun unter Berücksichtigung der Gleichungen 3.23 und 3.24 wie folgt notiert werden:

$$\int_0^l \frac{1}{(\parallel \overrightarrow{v} \parallel^2 + t^2 - 2t\overrightarrow{v} \bullet \overrightarrow{a})^{3/2}} dt \tag{3.25}$$

Die Lösung dieses Integrals kann mit Hilfe des Kosinussatzes und der Berechnung der Projektion H von p auf L mittels des Kreuzproduktes vereinfacht werden zu:

$$F(p) = \frac{\sin\alpha_1 + \sin\alpha_2}{d(p,H)^2} \tag{3.26}$$

wobei $d(P,H)$ die Distanz beschreibt zwischen dem Punkt p und seiner Projektion H (Abb. 3.27). $d(P,H)$ entspricht somit der Distanz $d_{pp'}$ aus der Evaluierung des Distanzfilters nach [Blo95b] (siehe voriger Abschnitt). α_1 und α_2 sind die vorzeichenbehafteten Winkel zwischen $[pH], [pC_1]$ bzw. $[pH], [pC_2]$.

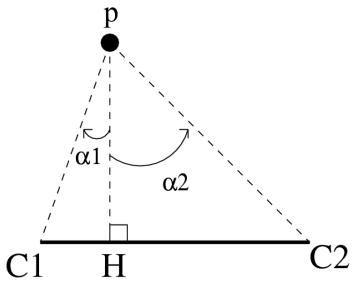


Abbildung 3.27: Berechnung der Feldfunktion aus Gleichung 3.26. Quelle: [CH01]

Die Berechnungskomplexität der Feldfunktion aus Gleichung 3.26 wurde in Anlehnung an die Vorgehensweise in [She99b] getestet (siehe Abschnitt 3.3.1). Die Anzahl der Gleitkommaoperationen $(+, -, *, /)$ und Spezialfunktionen (trigonometrisch, exponential, logarithmisch,...) der Feldfunktion wurde bestimmt. Die Verwendung der inversen kubischen Filterfunktion und des biquadratischen Polynoms resultieren in einer annähernd gleichen Anzahl. Von der neuen Filterfunktion ist demnach eine ähnlich hohe Performanz zu erwarten.

Die Generierung einer *Convolution Surface* mit variierendem Durchmesser wird durch eine Normalisierung der Distanz $d(p,H)$ gegen den an der Stelle H definierten Radius r erreicht. Hierzu wird der Term $d(p,H)^2$ in Gleichung 3.26 ersetzt durch:

$$d(p,H)^2 = 2\frac{d(p,H)^2}{r(H)^2} \tag{3.27}$$

Mit p sei ein Punkt gegeben, welcher auf der *Convolution Surface* auf halber Länge des Liniensegments lokalisiert ist. Das Liniensegment sei ausreichend lang, so dass der Trägerintervall der Filterfunktion weitestgehend abgedeckt wird. Je länger das Liniensegment ist, desto stärker konvergiert die Größe der Winkel α_0 und α_1 gegen 90°. Der Sinus der beiden Winkel ist dann ≈ 1. Da p auf der *Convolution Surface* liegt, ist die Distanz $d(p,H)$ gleich $r(H)$. Aus Gleichung 3.26 resultiert damit:

$$\frac{1+1}{2\frac{r(H)^2}{r(H)^2}} - Iso = 1 - Iso = 0 \tag{3.28}$$

Für die Generierung einer Oberfläche mit korrektem Durchmesserverlauf muss demnach die Isosurface an der Stelle $Iso = 1$ konstruiert werden.

3.3.2.3 Abbildung eines gegebenen Durchmesserverlaufs nach Hornus

In [HAC03] wird eine neue Filterfunktion für die korrekte Abbildung einer gegebenen Durchmesserdistribution eingeführt. Der abzubildende Radius r wird hier in die Filterfunktion integriert:

$$h(x) = \frac{r^2}{x^2}, x > 0 \tag{3.29}$$

Diese Funktion ist in Abb. 3.28 für verschiedene Radien r veranschaulicht.

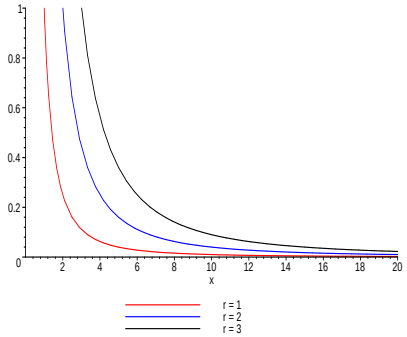


Abbildung 3.28: Die Filterfunktion nach [HAC03] $h(x) = r^2/x^2, x > 0$ für verschiedene Radien r. Mit zunehmendem Radius r wächst die Filterbreite.

In Abb. 3.29 ist die Berechnung des korrespondierenden Faltungsintegrals illustriert.

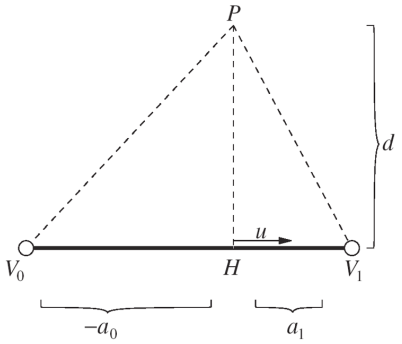


Abbildung 3.29: Bestimmung des Faltungsintegrals für die Filterfunktion aus Gleichung 3.29. *Quelle*: [HAC03]

a_0 sei die Distanz von H zu dem am weitesten von p entfernt liegenden Segmentendpunkt. a_1 ist dann die Entfernung von H zu dem verbleibenden Endpunkt. s ist ein Punkt auf dem Liniensegment $[V_0, V_1]$ und $u(s)$ ist die vorzeichenbehaftete Entfernung zwischen H und s, mit $u(V_1) > 0$. r_0 und r_1 seien die beiden Radien assoziiert mit den Endpunkten V_0 und V_1. Hieraus resultiert folgendes Faltungsintegral:

$$f(S,p) = \int_{-a_0}^{a_1} \frac{\frac{r_1 - r_0}{a_1 - a_0} u + \frac{r_0 a_1 - r_1 a_0}{a_1 - a_0}}{d(p,H)^2 + u^2} du \tag{3.30}$$

und dessen abgeschlossene Lösung:

$$F(p) = \frac{(d(p,H) * C - D^2/d(p,H))) * A + (r_0 - r_1) * D * B + (a_0 - a_1) * C}{(a_0 - a_1)^2} \tag{3.31}$$

mit:

$A = \arctan \frac{a_1}{d(p,H)} + \arctan -\frac{a_0}{d(p,H)}$

$B = \log\left(a_1^2 + d(p,H)^2\right))/(a_0^2 + d(p,H)^2)$

$C = (r_0 - r_1)^2$

$D = (r_0 a_1 - r_1 a_0)$

Damit ein gegebener Durchmesserverlauf korrekt abgebildet werden kann, wird auch hier ein Isowert gleich 1 vorgeschlagen. Leider wurden für die Faltung mit dem neuen Filter weder Zeitmessungen veröffentlicht noch Vergleiche mit anderen Filterfunktionen wie in [She99b] durchgeführt.

3.3.2.4 Konsequenzen für die Filterwahl

Die Wahl einer geeigneten Filterfunktion im Rahmen der Gefäßvisualisierung kann nicht allein anhand der Berechnungskomplexität der korrespondierenden Feldfunktion getroffen werden. Weiterhin ist die Möglichkeit der korrekten Abbildung eines gegebenen Durchmesserverlaufs mit Hilfe der Filterfunktion zu berücksichtigen. In der Literatur wurden bisher drei Funktionen hinsichtlich dieser Anforderung untersucht.

Während in [Blo95b] eine Gauß-Funktion zur Faltung des Skeletts verwendet wird, wendet [CH01] die inverse kubische Funktion an. Beide berechnen die Feldfunktion an einem Punkt p, basierend auf der Distanz zwischen p und seiner Projektion p' auf das Liniensegment dividiert durch einen aus den Segmentendpunkten interpolierten Radius . Interessant wäre zu untersuchen, ob die Faltung mit anderen in [She99b] vorgestellten Filterfunktionen ebenfalls auf Basis dieser Distanz berechenbar ist. So könnten mit Hilfe der gleichen Normalisierung zusätzliche Filter für eine korrekte Durchmesserabbildung in Frage kommen. Dies ist aus Zeitgründen im Rahmen dieser Arbeit nicht analysiert worden. Im Unterschied zu [Blo95b] und [CH01] wird der Radius in [HAC03] bereits vor der Faltung des Skeletts in die Filterfunktion integriert. Diese basiert auf der inversen quadratischen Funktion.

3.4 Visualisierung

Visualisierung spielt in den meisten Anwendungen bei der Exploration, Kognition und Explanation von Strukturen und Prozessen eine zentrale Rolle. Im Kontext der Informatik werden mit ihrer Hilfe abstrakte Daten in ein sichtbares Bild überführt. Viele Visualisierungsmethoden wurden bereits für implizite Oberflächen untersucht, angewendet und optimiert. Ihre Hauptaufgabe besteht hier in der Bestimmung des *zero-sets* der zugrundeliegenden impliziten Funktion. Die dabei vorwiegend verwendeten Techniken sind nach [AG01] *Ray Tracing* und *Polygonisierung*, welche in den folgenden beiden Abschnitten näher betrachtet werden. In Abschnitt 3.4.3 werden einige weitere Methoden, wie z.B. Partikelsysteme, vorgestellt. Der begrenzte zeitliche Rahmen dieser Arbeit gestattet keine komplette und detaillierte Beschreibung aller Visualisierungsmethoden. Stattdessen sollen die wichtigsten Verfahren kurz vorgestellt werden, um ihre Qualifikation für die Gefäßvisualisierung einschätzen zu können.

3.4.1 Ray Tracing

Lichtstrahlen bewegen sich in geraden Linien von einer Lichtquelle weg. Sie werden auf ihrem Weg absorbiert, gebrochen und reflektiert (Abb. 3.30). *Ray Tracing* generiert, basierend auf diesen grundlegenden Prinzipien des Lichttransports, ein Bild der beleuchteten Szene.

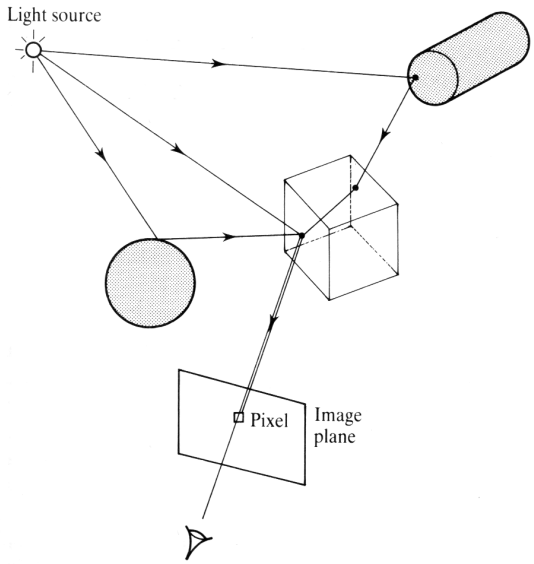


Abbildung 3.30: Lichtstrahlen werden auf dem Weg von der Lichtquelle (*Light source*) zur Bildebene (*Image plane*) gebrochen (Würfel) und reflektiert (Zylinder). *Quelle*: [WW92]

Da ein Großteil der Lichtstrahlen die Bildebene nie kreuzt, geht der grundlegende *Ray Tracing*-Algorithmus den effizienteren, umgekehrten Weg. Strahlen werden vom Betrachter aus durch die Bildebene in die Szene konstruiert und verfolgt. Dazu wird für jedes Pixel des Ergebnisbildes ein Punkt festgelegt, durch den der Strahl verlaufen soll, z.B. der Pixelmittelpunkt. Um die Farbe eines Pixels des Ergebnisbildes bestimmen zu können, müssen zuerst die Schnittpunkte des Strahls mit den Objekten der Szene bestimmt werden. Aus den Materialeigenschaften des getroffenen Objektes und der Beleuchtungssituation an dem Schnittpunkt wird der finale Farbwert berechnet. Die Beleuchtung ist von dem Winkel aus Oberflächennormale und der Richtung des einfallenden Lichts abhängig. Für die Visualisierung impliziter Oberflächen ist die Berechnung der Schnittpunkte mit den implizit definierten Objekten von besonderem Interesse. Die größte Herausforderung hierbei bilden komplexe, durch *blending* entstandene Strukturen. Für weiterführende Informationen zu allgemeinen *Ray Tracing*-Prinzipien sei der interessierte Leser an [Gla89] und [WW92] verwiesen.

Im Folgenden soll in Anlehnung an ein Beispiel aus [BBB+97] die Vorgehensweise bei der Schnittpunktberechnung verdeutlicht werden. Drei *Soft Objects* [WMW86] bilden eine komplexere Form durch *blending* Abb. 3.31.

Die in [WMW86] vorgestellten Feldfunktion für *Soft Object* i ist in Anlehnung an Gleichung 3.5 gegeben durch:

$$F_i(p) = 1 - \frac{4}{9}\frac{\|p - c_i\|^6}{{R_i}^6} + \frac{17}{9}\frac{\|p - c_i\|^4}{{R_i}^4} - \frac{22}{9}\frac{\|p - c_i\|^2}{{R_i}^2} \tag{3.32}$$

c_i bezeichnet hier das Zentrum des Soft Objects, während R_i dessen Einflußradius beschreibt.

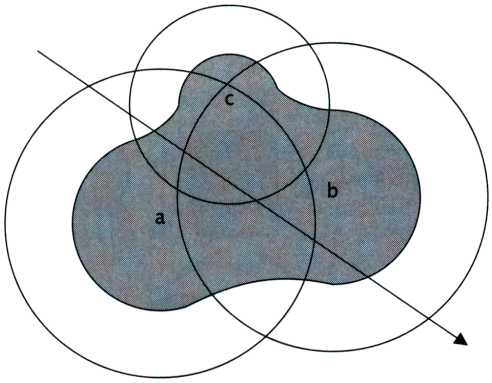


Abbildung 3.31: Drei *Soft Objects* und ihr jeweiliger sphärischer Einflussbereich werden von einem Strahl mehrfach geschnitten. *Quelle*: [BBB+97]

Die implizite Funktion mehrerer *Soft Objects* ist in Anlehnung an Gleichung 3.3 gegeben durch:

$$\sum_i F_i(p) - Iso = 0 \tag{3.33}$$

Gegeben sei weiterhin ein Strahl in der parametrischen Form $p = u + t * v$, wobei u und v den Startpunkt bzw. die Richtung des Strahls angeben. Für die Berechnung der Schnittpunkte des Strahls mit den durch Gleichung 3.33 definierten Objekten wird die Gleichung des Strahls in Gleichung 3.33 eingesetzt:

$$\sum_i F_i(u + t * v) - Iso = 0 \tag{3.34}$$

Die Summe polynomieller Funktionen ist wieder ein Polynom. Zur Bestimmung der Schnittpunkte muss dieses Polynom 6. Grades nach t aufgelöst werden. Hierfür können herkömmliche mathematische Methoden, wie das in [RR78] vorgestellte Verfahren benutzt werden. Durch Einsetzen von t in die Strahlengleichung werden die Schnittpunkte gefunden.

Auf seinem Weg durch die Szene durchquert der Strahl die Einflussgebiete der drei *Soft Objects* (Abb. 3.31). An jedem der Schnittpunkte mit den einhüllenden Kugeln (*bounding spheres* ändert sich Gleichung 3.33, da neue Primitive hinzukommen und andere wegfallen. Zwischen benachbarten Schnittpunkten ist die Anzahl der dort Einfluss besitzenden Primitive jedoch konstant. Der Schnittpunkt mit dem eigentlichen Objekt kann hier aus einer kleinen Anzahl von Primitiven bestimmt werden. Ein Algorithmus zur Bestimmung der Schnittpunkte beinhaltet nach [BBB+97] die folgenden Schritte:

1. Bestimme die Schnittpunkte des Strahls mit allen *bounding spheres* und ordne sie entlang des Strahls.
2. Löse Gleichung 3.34 nach t in jedem Intervall.
3. Gib die Schnittpunkte zurück, welche innerhalb des Intervalls liegen.

Ein erheblicher Nachteil von *Ray Tracing* ist die Notwendigkeit der Neuberechnung der Schnittpunkte nach jeder Sichtänderung auf die Szene. Dies behindert eine flüssige Interaktion mit komplexeren Objekten. Zudem ist Schritt eins des Algorithmus besonders zeitaufwendig, wenn sehr viele Objekte in der Szene liegen. Zahlreiche Optimierungen widmen sich jedoch der Beschleunigung dieses Vorgangs. So kann eine Partitionierung des Raumes um die Objekte in

Zellen, die Anzahl der zu betrachtenden *bounding spheres* erheblich einschränken. In einem Vorverarbeitungsschritt wird für jedes Objekt bestimmt, welche Zellen es beeinflusst. Verläuft nun der Strahl durch die Szene, so werden alle von ihm getroffenen Zellen bestimmt. Objekte, welche nicht in diesen Zellen liegen, werden bei der Berechnung nicht berücksichtigt. Für weitere Verfahren zur Effizienzsteigerung siehe [BBB+97], [Gla89] und [WW92]. Abbildungen 3.32(a) und 3.32(b) zeigen zwei durch *Ray Tracing* erzeugte Ansichten impliziter Oberflächen.

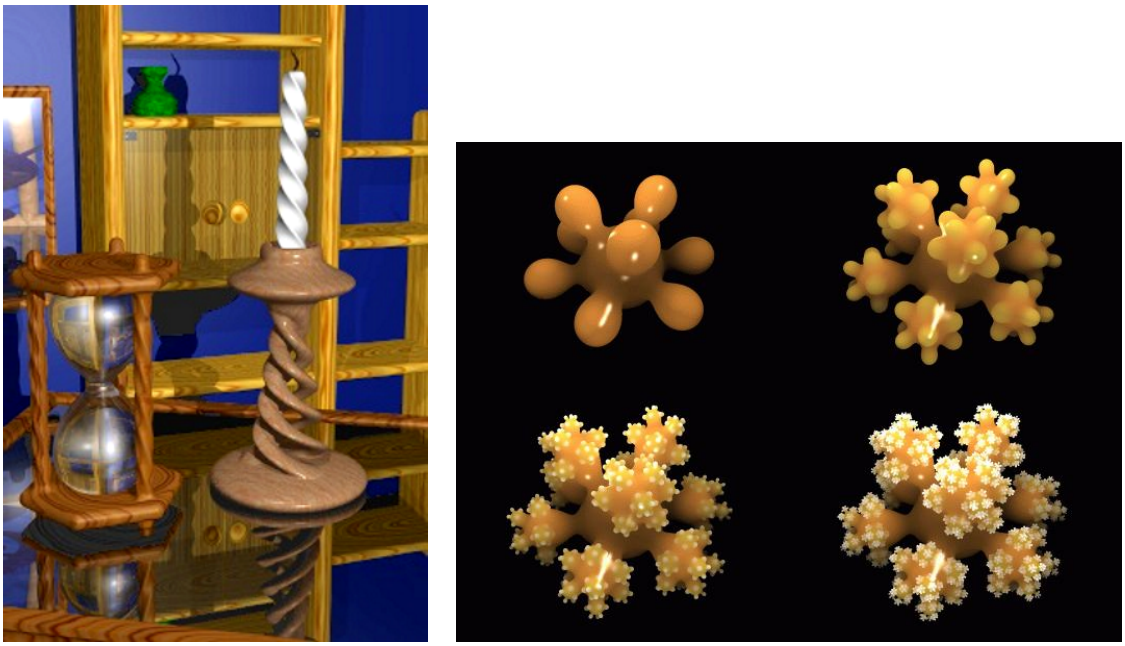


Abbildung 3.32: Eine Szene, modelliert von Eric Galin unter Verwendung des *BlobTrees* [WGG99] (*links*). Der BlobTree ist eine CSG-Erweiterung für implizite Oberflächen. *Quelle*: [WGG99]. Schneeflocken aus 10, 91, 820 und 7381 *Blobs*, kreiert von Andrei Sherstyuk mit dem *Ray Tracer* RATS. *Quelle*: [She98b].

3.4.2 Polygonisierung

Wie in [AG01] und [MS03] bemerkt, ermöglicht die Polygonisierung der impliziten Oberfläche eine interaktive Visualisierung. Ist die Nullstellenmenge der impliziten Funktion einmal in eine polygonale Repräsentation überführt worden, so kann diese mit Hilfe herkömmlicher Grafikhardware effizient dargestellt werden. Für einfache Formen der Interaktion, wie die Sichtänderung auf die Szene, sind keine weiteren Berechnungen der impliziten Funktion notwendig. Nach [AG01] lassen sich die heutzutage existierenden Polygonisierungsmethoden wie folgt klassifizieren: *Spatial Sampling*, *Surface Tracking* und *Surface Fitting*.

3.4.2.1 Spatial Sampling

Spatial Sampling-Techniken unterteilen den gesamten Raum um die implizit beschriebenen Objekte in Zellen (Abb. 3.33 (links)) und bestimmen anschließend die von der impliziten Oberfläche geschnittene Teilmenge. Die Zellen haben entweder die Form eines Würfels (*Marching Cubes*) [WMW86], [LC87] oder eines Tetraeders (*Tetrahedral Decomposition* [ST90]. Zunächst

wird die implizite Funktion für jeden Eckpunkt einer Zelle berechnet. Anhand des Vorzeichens kann für jeden Eckpunkt bestimmt werden, ob er innerhalb, außerhalb oder auf der impliziten Oberfläche liegt. Kanten der Zelle mit entgegengesetzter Polarität werden von der Oberfläche geschnitten. Mittels trilinearer Interpolation, *regula falsi* oder *binary subdivision* kann der Schnittpunkt bestimmt werden. Die letzten beiden Methoden sind rechenintensiver, liefern aber eine genauere Approximation der Oberfläche [Blo88]. Basierend auf den Schnittpunkten aller Kanten einer Zelle, kann das *zero-set* innerhalb des Würfels oder Tetraeders polygonisiert werden (Abb. 3.33 (rechts))

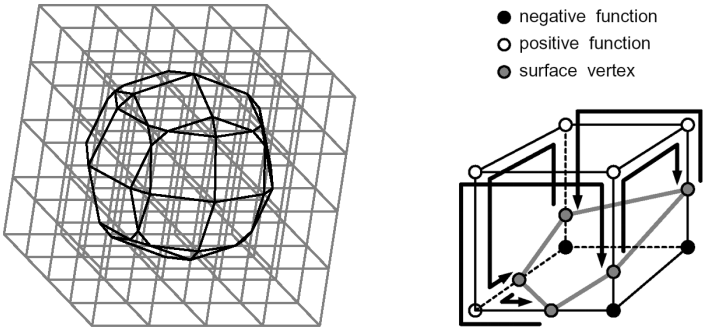


Abbildung 3.33: *Spatial Sampling.* Unterteilung des Raumes um eine Kugel in kubische Zellen (*links*) und Polygonisierung der implizit definierten Oberfläche innerhalb einer solchen Zelle (*rechts*). Die Pfeile beschreiben eine Ordnung der Oberflächenpunkte entgegen dem Uhrzeigersinn. Nach [Blo88] und [Blo94]

Bei der Verwendung würfelförmiger Zellen können für bestimmte Konstellationen von Schnittpunkten mehrere Polygonisierungsmöglichkeiten existieren, so genannte *Mehrdeutigkeiten* (*ambiguity cases*). Eine falsche Entscheidung führt hier zu Löchern in der Oberfläche. Dieses Problem wird durch die Verwendung von Tetraedern gelöst [Blo88]. Ein Nachteil der Verwendung tetraederförmiger Zellen ist jedoch die Entstehung einer Vielzahl degenerierter Dreiecke, d.h. das Verhältnis der Radien von Umkreis und Innkreis des Dreiecks weicht stark von 1 ab. Weiterhin zeigt ein Vergleich der beiden Zelltypen in [AG01], dass mit Tetraedern ungefähr die dreifache Anzahl von Dreiecken generiert wird. Da auch die *Marching Cubes*-Technik degenerierte Dreiecke erzeugt und bei ausreichend feiner Zelldichte in einer hohen Anzahl von Dreiecken resultiert, wird generell eine Polygonreduktion (*Mesh Simplification*) in einem Nachbearbeitungsschritt angewandt [AG01]. Die Zellgröße ist ein wichtiger Faktor der Polygonisierung. Sie ist indirekt proportional zu der Anzahl der erzeugten Dreiecke. Weiterhin entscheidet sie über die korrekte Darstellung der Topologie eines Objektes. Ist die Zellgröße zu hoch, so werden filigrane Teile des Objektes möglicherweise nicht angezeigt. Ist sie hingegen zu niedrig, werden massivere Objektteile unnötig dicht trianguliert. Weiterhin sollten gekrümmte Abschnitte dichter polygonisiert werden als gerade. Wünschenswert sind daher Polygonisierungsmethoden, welche die Zellgröße an die lokale Geometrie der impliziten Oberfläche anpassen. Eine Vielzahl solcher Methoden wurde bereits entwickelt [Blo88], [Blo95a]. Abb. 3.34 zeigt die adaptive Einteilung eines generalisierten Zylinders in Abhängigkeit von der lokalen Oberflächenkrümmung. Benachbarte Zellen unterschiedlicher Größe müssen mit besonderer Sorgfalt polygonisiert werden, da sonst Löcher in der Oberfläche auftreten.

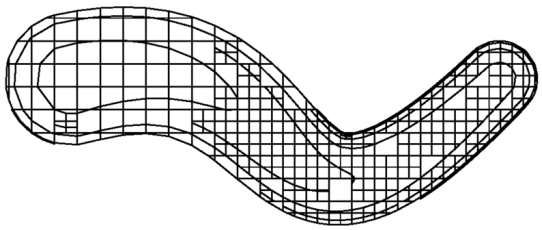


Abbildung 3.34: *Adaptive Spatial Sampling.* Adaptive Unterteilung eines generalisierten Zylinders. *Quelle*: [Blo88]

3.4.2.2 Surface Tracking

Während *Spatial Sampling* den gesamten Raum unterteilt, generiert *Surface Tracking* ausschließlich Zellen welche tatsächlich von der impliziten Oberfläche geschnitten werden. Die Komplexität ist hier $O(n^2)$, wobei n die Größe der Oberfläche beschreibt. *Spatial Sampling* hat im Vergleich dazu eine Komplexität von $O(n^3)$. Während einige *Surface Tracking*-Verfahren Zellen entlang der impliziten Oberfläche generieren und diese dann triangulieren [Blo94], konstruieren andere Methoden die Oberflächendreiecke direkt [HSIW96].

Der in [Blo94] vorgestellte *Implicit Polygonizer* unterteilt den Raum um die implizite Oberfläche mit Hilfe eines würfelförmigen Partitionselements und trianguliert im Anschluss das entstandene Würfelgitter. Die Größe des Würfels ist während des gesamten Partitionierungsprozesses fix. Dies erlaubt zwar eine einfache Implementierung, erfordert jedoch die sehr sorgfältige Bestimmung der Würfelgröße. Zu kleine Würfel erzeugen unnötig viele Dreiecke, während durch zu große Würfel Teile der Oberfläche nicht dargestellt werden.

Die Überführung der impliziten Definition des Objektes in ein geometrisches Modell beginnt mit der Bestimmung eines Startpunktes auf der Oberfläche. Hierzu gibt der Benutzer einen Punkt vor, welcher in der Nähe des Objektes lokalisiert ist. Anhand des Vorzeichens der impliziten Funktion an diesem Punkt wird nun bestimmt, ob sich der Punkt innerhalb oder außerhalb des Objektes befindet. Danach beginnt in der unmittelbaren Nähe des Punktes die Suche nach einem Pendanten mit entgegengesetzter Polarität, d.h. ein Punkt welcher sich auf der entgegengesetzten Seite der Objektoberfläche befindet. Die Verbindung zwischen den beiden Punkten wird dazwischen von der impliziten Oberfläche geschnitten. Der Schnittpunkt kann mittels *binary subdivision* bestimmt werden. Dies bedeutet eine Annäherung an die Nullstelle der impliziten Funktion durch ihre wiederholte Berechnung in einem enger werdenden Intervall.

Ist der Startpunkt bestimmt, so wird ein Startwürfel an diesem Punkt zentriert. Für alle Eckpunkte des Würfels wird die implizite Funktion berechnet. Um den Rechenaufwand gering zu halten wird die Funktion an jedem Eckpunkt eines Würfels nur einmal berechnet, zwischengespeichert und mittels einer *hashing*-Technik nach [WMW86] indiziert. Kanten mit unterschiedlicher Polarität werden von der impliziten Oberfläche geschnitten. An jeder Würfelfläche die solche Kanten enthält, wird ein neuer Würfel generiert. Dies entspricht dem Fortführungsschema *continuation scheme* aus [WMW86]. Der Prozess ist beendet, wenn die gesamte Oberfläche von Würfeln eingeschlossen ist (Abb. 3.35).

Die Triangulierung des Oberflächenanteils in einer entstandenen Zelle erfolgt nun entweder direkt mittels eines Algorithmus aus [Blo88] oder nach einer Zerlegung der Zelle in Tetraeder *Tetrahedral Decomposition* [PT90]. Beide Ansätze vermeiden Löcher in der Oberfläche, welche

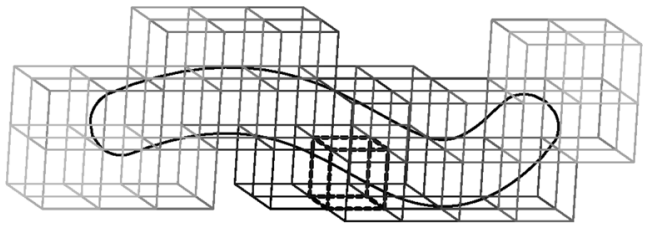


Abbildung 3.35: *Surface Tracking.* Der Startwürfel ist gestrichelt dargestellt. Dunkel eingefärbte Würfel wurden eher generiert als hell eingefärbte Würfel. *Quelle*: [Blo88]

durch Mehrdeutigkeiten bei den möglichen Polaritätskombinationen eines Würfelelements entstehen können. Letztere Methode beschreibt den eleganteren Weg [Blo94], resultiert jedoch auch in einer deutlich größeren Anzahl von Dreiecken und einem höheren Rechenaufwand.

Die Berechnung von Schnittpunkten der impliziten Oberfläche mit den Kanten eines Würfel erfolgt wiederum durch *binary subdivison*. Im Gegensatz zu der bei diskreten Daten häufig verwendeten trilinearen Interpolation wird diese Vorgehensweise der kontinuierlichen Natur der impliziten Funktion gerecht und gestattet eine beliebig genaue Oberflächenapproximation. Jeder Oberflächenpunkt wird nur einmal berechnet und ähnlich den Eckpunkten des Partitionierungselements mittels einer *hashing*-Technik indiziert.

Zu jedem Oberflächenpunkt wird weiterhin eine Normale berechnet. Die Normale an einem Punkt der Oberfläche entspricht dem dortigen normalisierten Gradienten und kann durch drei weitere Evaluierungen der impliziten Funktion approximiert werden [Blo88]. Die Normalen sowie die zugehörigen Oberflächenpunkte werden aus Effizienzgründen in dem so genannten *Punkte/-Polygone*-Format gespeichert. Jeder Punkt und jede Normale wird hier, obwohl zu mehreren Dreiecken gehörend, nur einmal in einem Datenfeld gespeichert. Eine weitere Datenstruktur beschreibt die Dreiecke durch Indizes in diese Felder.

Neben den durch die fixe Würfelgröße bedingten Problemen und dem Auftreten degenerierter Dreiecke (beides wurde bereits unter *Spatial Sampling* diskutiert) bringt das zellbasierte *Surface Tracking* eine zusätzliches Problem mit sich. So muss getrennt für jedes Objekt ein Startpunkt durch den Benutzer festgelegt werden.

Eine Alternative zu der zellbasierten Polygonisierung wurde durch [HSIW96] entwickelt. Das Verfahren bekannt als *Marching Triangles* generiert die Oberflächendreiecke auf direktem Wege. Initial wird ein Startdreieck auf der impliziten Oberfläche platziert. Ausgehend von dessen Kanten und unter Berücksichtigung einer gelockerten Delaunay-Bedingung, werden neue Dreiecke generiert. Dieser Prozess ist beendet, wenn die gesamte Oberfläche erfasst wurde. Ein besonderes Augenmerk bei diesem Verfahren liegt auf der Vermeidung der Konstruktion von degenerierten Dreiecken. Wie in [AG01] bemerkt, ist *Marching Triangles* jedoch nicht ohne weiteres auf geschlossene implizit beschriebene Objekte anwendbar. Hier können Löcher in der Oberfläche auftreten. Weiterhin wird die Kantenlänge in [HSIW96] weitestgehend konstant gehalten, was keine Anpassung der Dreiecksgröße an die Krümmung der Oberfläche gestattet. Bei zu großer Krümmung sind Artefakte zu beobachten. Die in [AG01] vorgestellte Erweiterung löst diese Probleme. Ein adaptives Verfahren variiert die Kantenlänge entsprechend der lokalen Krümmung. In einem Nachbearbeitungsschritt werden die Löcher in der Oberfläche geschlossen Abb. 3.36.

In [AG01] werden die Verfahren *Marching Cubes*, *Tetrahedral Decomposition* und die verbesserte

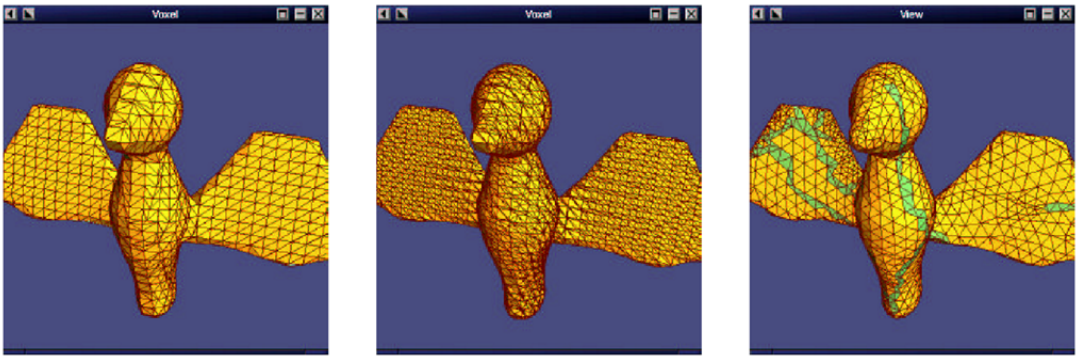


Abbildung 3.36: *Marching Triangles*. Polygonisierung eines impliziten beschriebenen Vogelmodells mittels *Marching Cubes* (*links*), der Zerlegung in Tetraeder (*Mitte*) und *Marching Triangles* (*rechts*). Die dunklen Dreiecke in der rechten Darstellung kennzeichnen die im Nachbearbeitungsschritt geschlossenen Löcher. Quelle: [AG01]

Marching Triangles Variante anhand von vier Modellen miteinander verglichen. *Marching Triangles* generiert nicht nur weniger Dreiecke in vergleichbarer Zeit, sondern verringert auch die Anzahl degenerierter Dreiecke erheblich. Aus der Ähnlichkeit von *Spatial Sampling*-Methoden und zellbasiertem *Surface Tracking* lässt sich schließen, dass auch letztere Verfahren hinter *Marching Triangles* zurücktreten. Die direkte Konstruktion von Dreiecken entlang der Oberfläche gewährleistet, dass auch sehr filigrane Strukturen erfasst werden. Die Speicheranforderungen sind gering, da keine *Zellstruktur* im Speicher gehalten werden muss.

3.4.2.3 Surface Fitting

Surface Fitting-Methoden passen ein initiales Polygonnetz sukzessive an die implizite Oberfläche an. Hierbei sind zwei verschiedene Herangehensweisen zu unterscheiden: lokale Methoden und globale Methoden. Verfahren der ersten Kategorie sind besonders für die skelettbasierte Modellierung geeignet. Hier wird jedes Skelettprimitiv getrennt betrachtet. Dies ermöglicht das Einfügen und Entfernen von Primitiven ohne eine erneute Polygonisierung des gesamten Objekts. In [DTG96] wird für jedes Skelettelement zuerst ein Hüllkörper (*bounding box*) bestimmt (Abb. 3.37 links oben). Auf der Oberfläche dieses Hüllkörpers wird entsprechend der vom Benutzer festgelegten Abtastdichte ein Gitter konstruiert. Die Gitterpunkte werden dann auf das Skelett projiziert. Dies beschreibt eine Achse ausgehend vom Skelett in Richtung Hüllkörper. (Abb. 3.37 oben Mitte). Die projizierten Punkte werden nun entlang der jeweiligen Achse um den Wert e bzw. e' verschoben (Abb. 3.37 rechts oben). Ausgehend von den dadurch entstandenen Startpunkten wird auf jeder Achse mit Hilfe des Vorzeichens der impliziten Funktion ein Punkt außerhalb und ein Punkt innerhalb der Oberfläche gesucht. Mittels *regula falsi* wird nun basierend auf diesen beiden Punkten der Schnittpunkt mit der impliziten Oberfläche bestimmt (Abb. 3.37 unten).

Für die Visualisierung der impliziten Oberfläche werden die berechneten Punkte auf so genannte *scales* abgebildet (Abb. 3.38 links). Die Erzeugung dieser Darstellung ist effizient möglich und kann als Voransicht des Objektes dienen. Für komplexe Modelle wird jedoch alternativ eine stückweise Polygonisierung vorgeschlagen. Hierzu kann für jedes Skelettprimitiv anhand der korrespondierenden Startpunkte eine initiales Polygonnetz konstruiert werden. Konvergie-

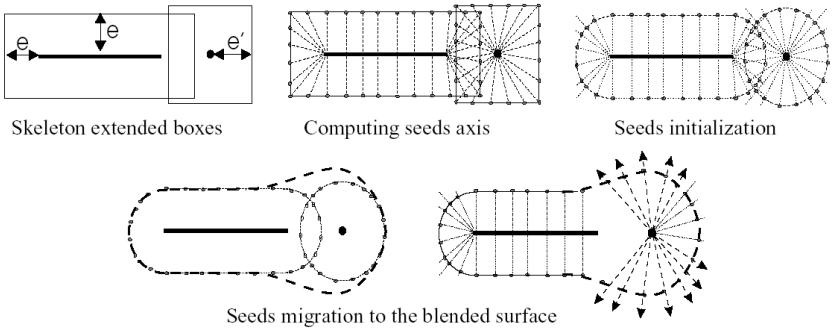


Abbildung 3.37: *Surface Fitting.* Von der Initialisierung der Startpunkte bis hin zur Berechnung der Oberflächenpunkte. *Quelle*: [DTG96]

ren die Startpunkte nun mittels *regula falsi* gegen die Oberfläche, verformt sich dieses Netz entsprechend. Das Polygonnetz der gesamten Oberfläche wird zuletzt aus den einzelnen Teilnetzen zusammengesetzt. Hierbei wird jeweils nur der Netzabschnitt verwendet, an welchem gerade das zugehörige Primitiv den größten Einfluss besitzt. Durch diese Vorgehensweise entstehen jedoch Diskontinuitäten an den Übergängen zwischen Polygonnetzen benachbarter Primitive (Abb. 3.38 rechts).

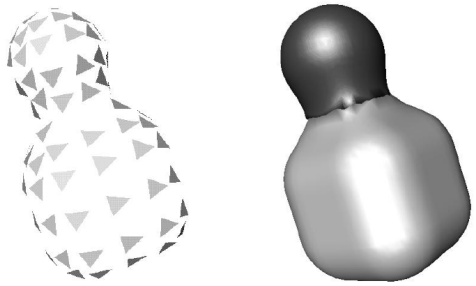


Abbildung 3.38: Visualisierung durch *scales* (*links*) und Polygonisierung (*rechts*). Nach [DTG96]

Dieses Problem wird in [CH01] durch eine lokale Überlappung der Teilnetze gelöst. Die zusätzliche Konstruktion von Dreiecken hat nach [CH01] nur einen geringen Einfluss auf die Gesamtperformanz. Eine weitere Verbesserung der Methode [AJC02] passt die Abtastdichte des Gitters (Abb. 3.37 oben mitte) der Oberflächenkrümmung an.

Neben den lokalen Methoden, welche jedes Skelettprimitiv getrennt betrachten, werden durch [vOW93] und [SH97] globale Ansätze eingeführt. Das *Shrink-wrap* Verfahren [vOW93] konvergiert ausgehend von einem das Objekt komplett einhüllenden Polygonnetz, gegen die implizite Oberfläche. Ein Nachteil dieser Vorgehensweise ist, dass versteckt im Objekt liegende Löcher nicht gefunden werden. Eine in [SH97] präsentierte Lösung dieses Problems konstruiert die implizite Oberfläche durch die Ausdehnung einfacher Basisnetze. Hierbei müssen so genannte *critical points* berücksichtigt werden. Dies sind Punkte im Raum, an denen die Oberflächen benachbarter Primitiven verschmelzen, Löcher geschlossen werden oder Löcher im Objekt lokalisiert sind. Nachdem alle *critical points* gefunden worden, werden die Basisnetze platziert und der Ausdehnungsprozess wird gestartet. Löcher im Objekt werden so gefunden und korrekt modelliert. In [AG01] wird jedoch betont, dass dieses Verfahren nur in der punktbasierten Modellierung

anwendbar ist, da *critical points* für komplexere Skelettprimitive nicht überall bestimmt werden können.

Weitere Informationen bezüglich der Visualisierung impliziter Oberflächen durch Polygonisierung findet der interessierte Leser in [Blo88], [BBB+97], [VGdF02] und [MS03].

3.4.3 Weitere Verfahren

Neben *Ray Tracing* und Polygonisierung wurden im Laufe der Zeit einige weitere Verfahren zur Visualisierung impliziter Oberflächen angewandt. Hierzu zählen *Section Display*, *Contour Tracing* oder auch *Partikelsysteme*. Diese Techniken werden im Folgenden kurz skizziert.

3.4.3.1 Section Display

Section Display generiert ein 2D-Schnittbild durch eine implizite Funktion [BW90]. Die Interpretation einer solchen Ansicht erfordert vom Betrachter jedoch einiges an Erfahrung und gibt nur einen schlechten Überblick zur Gesamttopologie des beschriebenen Objektes (Abb. 3.39). In einer Modellierumgebung kann eine solche Ansicht allerdings zur Vorschau eingesetzt werden, da ihre Berechnung sehr effizient möglich ist. Die Qualität der 2D-Ansicht ist von der Sampledichte der Funktion, d.h. von der Auflösung des Schnittbildes abhängig.

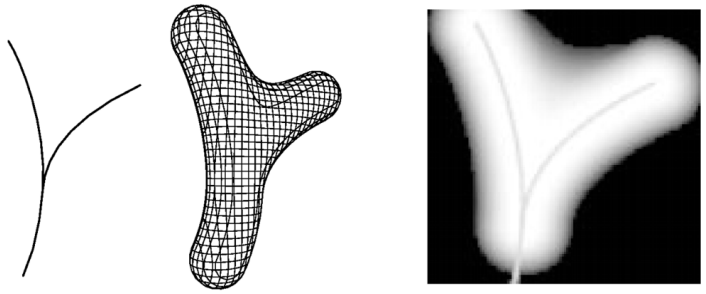


Abbildung 3.39: *Section Display*. Das Skelett (*links*), die korrespondiere implizite Oberfläche (*Mitte*) und ein *Section Display* (*rechts*). Nach [BW90]

3.4.3.2 Contour Tracing

In [BBB+97] wird das so genannte *Contour Tracing* beschrieben, welches mit Hilfe mehrerer Schnittebenen eine Konturliniendarstellung der impliziten Oberfläche generiert. Hierbei werden ausgehend vom Betrachter Ebenen senkrecht zum Sichtstrahl durch die implizite Funktion konstruiert. Für jede Ebene wird das lokale *zero-set* der impliziten Funktion gezeichnet (Abb. 3.40). Ausgenommen hiervon sind Abschnitte, welche von näherliegenden Ebenen verdeckt werden. Nach [BBB+97] ist dieses Verfahren vor allem für die technische Konstruktion geeignet.

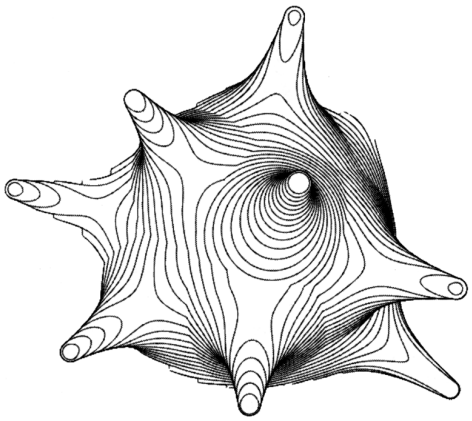


Abbildung 3.40: *Contour Tracing.* Schnittebenen durch das implizit beschriebene Modell und die *zero*-Kontur pro Ebene. *Quelle*: [BBB+97]

3.4.3.3 Partikelsysteme

Partikelsysteme werden in [WH94] zur Echtzeitdarstellung der Nullstellenmenge einer impliziten Funktion genutzt (Abb. 3.41). Für Animationszwecke werden die Partikel an ein sich bewegendes Objekt angepasst. Zur Modellierung können sie auch bewegt werden, was in einer Anpassung der Objektform resultiert. Durch das Konzept der *adaptive mutual repulsion* verteilen sich die Partikel gleichmäßig auf der Oberfläche. Wird das Objekt modifiziert, so werden Partikelgröße und Partikeldichte automatisch angepasst, um eine Echtzeitinteraktion zu gewährleisten. Die Polygonisierung des *zero-sets* kann durch eine Triangulierung der Partikelmenge erreicht werden. Die dafür notwendige Zeit verlangsamt jedoch den gesamten Visualisierungsprozess um ein Vielfaches [AG01].

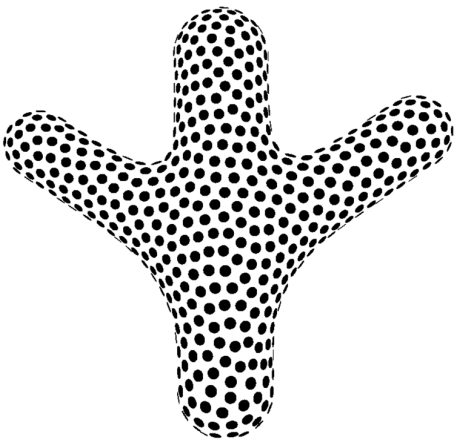


Abbildung 3.41: Partikelsysteme. Partikel verteilen sich gleichmäßig entlang der impliziten Oberfläche. *Quelle*: [WH94]

Weitere Methoden der Visualisierung impliziter Oberflächen sind in [BW90] und [VGdF02] beschrieben.

3.5 Konsequenzen für den Entwurf

Ziel des folgenden Kapitels ist es eine Visualisierungsmethode zu entwickeln, welche das in der Gefäßanalyse rekonstruierte Gefäßmodell, bestehend aus dem Gefäßskelett und assoziierter Oberflächeninformation (siehe Abschn. 2.2), auf geometrische Primitive abbildet. Dabei sollen die in Kapitel 1 aufgestellten Anforderungen an eine ideale Gefäßvisualisierungsmethode berücksichtigt werden. Wie können implizite Oberflächen genutzt werden, um dieses Ziel zu erreichen ?

Die Antwort hierauf beginnt mit einer Modellierung der Gefäßstruktur mittels impliziter Methoden. In diesem Kapitel wurden die punktbasierte und die skelettbasierte Modellierung vorgestellt. Aufgrund der Struktur der Daten aus der Gefäßanalyse ist die skelettbasierte Modellierung bestens geeignet. So kann das Gefäßskelett durch Liniensegmente approximiert werden, welche benachbarte Mittelpunkte von Skelettvoxeln miteinander verbinden.

Mit *Distance Surfaces* und *Convolution Surfaces* wurden zwei verschiedenen Techniken der skelettbasierten Modellierung vorgestellt. Ein Nachteil bei der Verwendung von *Distance Surfaces* ist das Auftreten von Diskontinuitäten entlang konkaver Stellen der Oberfläche bzw. *Bulging* entlang konvexer Stellen. Dies widerspricht den Anforderungen einer glatten, artefaktfreien Oberfläche und einer korrekten Abbildung des Durchmesserverlaufs. So können irritierende Verdickungen als pathologische Veränderungen, z.B. Aneurysma, interpretiert werden. *Convolution Surfaces* sind absolut frei von Diskontinuitäten der Oberflächennormalen und vermeiden ebenfalls *Bulging*, zumindest für nicht-verzweigte Strukturen. Da ein Gefäßbaum jedoch zahlreiche Verzweigungen enthält, muss sorgfältig geklärt werden wie stark die Verdickungen sind und ob sie eventuell vermieden werden können. Weiterhin sollte das *Unwanted Blending*-Problem berücksichtigt werden, um Änderungen der Gefäßtopologie zu vermeiden.

Ein sehr wichtiger Punkt bei der Arbeit mit *Convolution Surfaces* ist die Wahl einer geeigneten Filterfunktion. Obwohl in der Literatur zahlreiche Funktionen vorgestellt worden, grenzt die Anforderung an eine korrekte Abbildung des Gefäßdurchmessers die Auswahl ein. Die drei verbleibenden Filterfunktionen müssen für die Gefäßvisualisierung getestet werden.

Nachdem eine implizite Beschreibung des Gefäßmodells generiert wurde soll die Gefäßstruktur basierend auf dieser Information dargestellt werden. *Section Display* ist im Rahmen der Gefäßvisualisierung nicht geeignet, da es nur eine zweidimensionale Schnittebene durch die implizite Funktion konstruiert und keinen guten Überblick betreffs der Topologie des gesamten Gefäßbaums erlaubt. *Contour Tracing* liefert eine zu abstrakte Darstellung der Objekte und ist eher für technische Illustrationen anwendbar. Die Visualisierung mit Partikelsystemen ist wahrscheinlich das schnellste aller Verfahren, resultiert jedoch ebenfalls in einer sehr abstrakten Darstellung. Die Möglichkeit einer Triangulierung der Partikelmenge besteht zwar, wurde jedoch innerhalb dieser Arbeit nicht berücksichtigt. Die Verwendung von Partikelsystemen ist für Echtzeitinteraktion mit Objekten nicht für qualitativ hochwertige Darstellungen gedacht.

Die beiden größten Konkurrenten bei der Visualisierung impliziter Oberflächen sind *Ray Tracing* und Polygonisierung. Eine große Stärke von *Ray Tracing* ist die Generierung extrem realistischer Bilder durch die Berücksichtigung von Beleuchtung, Schatten und Reflektionen. Weiterhin können Objekte beliebig nah fokussiert werden ohne Qualitätsverlust. *Ray Tracing* ist speichereffizient, da es die implizite Oberfläche direkt darstellt, ohne eine kostspielige Zwischenrepräsentation über Polygone. Ein wesentlicher Nachteil von *Ray Tracing* ist jedoch die Notwendigkeit der Neuberechnung von Schnittpunkten bei jeder Sichtänderung auf die Szene. Bei sehr komple-

xen Objekten ist daher eine interaktive Visualisierung, trotz aller Optimierungen, meist nicht zu gewährleisten [AG01]. Weiterhin liefert das Verfahren keine geometrische Repräsentation der dargestellten Oberflächen, was eine Interaktion mit den Objekten erschwert. Die Polygonisierung der impliziten Oberfläche bietet hier einen erheblichen Vorteil. Ist das *zero-set* der impliziten Funktion einmal in eine polygonale Repräsentation überführt worden, so kann diese mit Hilfe herkömmlicher Grafikhardware dargestellt werden. Dies ermöglicht eine effiziente Interaktion mit dem Modell.

Da eine interaktive Visualisierung sowie die Möglichkeit der Interaktion mit der Gefäßstruktur in klinischen Applikationen wünschenswert sind, wurde sich im Rahmen dieser Arbeit für die Polygonisierung entschieden. Um die hohe Komplexität von *Spatial Sampling*-Methoden zu vermeiden, wurde der in Abschnitt 3.4.2.2 vorgestellte *Implicit Polygonizer* [Blo94] als Vertreter der *Surface Tracking*-Techniken gewählt. Dieser ist kostenlos im Internet verfügbar [4]. Die weiteren neben der Visualisierung auftretenden Probleme haben eine Umsetzung und umfangreiche Tests der zahlreichen anderen Polygonisierungsmethoden nicht gestattet.

[4] `http://www.unchainedgeometry.com/jbloom/misc/polygonizer.c`, Stand: 5.02.2004

4 Entwurf

Aufbauend auf den theoretischen Überlegungen beschreibt dieses Kapitel den Entwurf einer Visualisierungsmethode für baumartige anatomische Strukturen basierend auf *Convolution Surfaces*. Begonnen wird mit der Präsentation erster Visualisierungsresultate, welche in einem frühen Stadium dieser Arbeit bereits generiert werden konnten. Die hieraus ersichtlichen Probleme sowie Aspekte der geometrischen Repräsentation eines Gefäßmodells und einer anschließenden Interaktion werden in den weiteren Abschnitten diskutiert.

4.1 Erste Visualisierungstests

Mit Hilfe des *Implicit Polygonizers* [Blo94] konnten erste Visualisierungstests durchgeführt werden. Hierbei wurden die in Abschnitt 3.3.2 beschriebenen Methoden zur Generierung einer *Convolution Surface* mit vorgegebenem Durchmesserverlauf untersucht. Die drei Lebergefäßbäume (LG_1, LG_2 und LG_3) dienten als Testmodelle (Abb. 4.1).

Abbildung 4.1: Skelette der Lebergefäßbäume LG_1 (*links*), LG_2 (*Mitte*) und LG_3 (*rechts*).

Die Komplexität eines Modells sei durch die Anzahl der Liniensegmente beschrieben, aus denen sich das Skelett zusammensetzt. In Relation zu den während der gesamten Arbeit visualisierten Gefäßstrukturen sind LG_1 und LG_2 von geringer (1652 Liniensegmente) bzw. mittlerer (3101 Liniensegmente) Komplexität, während LG_3 eine relativ hohe (4445 Liniensegmente) Komplexität aufweist.

Der *Implicit Polygonizer* zählt zu den in Abschnitt 3.4.2.2 vorgestellten *Surface Tracking*-Methoden. Diese erfassen die gesamte implizit beschriebene Oberfläche durch Zellen. Die Zellgröße ist entscheidend für eine adäquate Visualisierung und wurde für die drei Testmodelle vorerst empirisch bestimmt.

Mittels einer so genannten *brute-force*-Taktik wurde die Feldfunktion einem Punkt p unter Berücksichtigung des gesamten Skeletts bestimmt. So sieht es die Definition aus (Gleichung 3.10) vor. Hierbei konnte aufgrund der Superpositions-Eigenschaft der Faltung jedes Liniensegments getrennt betrachtet werden. Gemessen wurde die Zeit vom Einlesen der Gefäßdaten bis hin zu

einer vollständigen Darstellung der rekonstruierten Gefäßoberfläche auf dem Bildschirm. Weiterhin wurde die Dreieckszahl der Modelle bestimmt, um die Komplexität des entstandenen Dreiecksnetzes zu illustrieren. Zusätzlich wurde die Anzahl der gesamten Feldfunktionsberechnungen für jedes Modell festgehalten. Die Ergebnisse sind in Tabelle 4.1 zusammengefasst.

Tabelle 4.1: Rechnerische und geometrische Komplexität von drei Testmodellen. Hardware: Pentium 4 CPU 3.06GHz, 1024MB RAM, ATI Radeon 9600.

	Rechenzeit (m:s)			**Dreiecke**			**Funktionsaufrufe** $*10^6$		
Filter	LG_1	LG_2	LG_3	LG_1	LG_2	LG_3	LG_1	LG_2	LG_3
Gauß	11:26	27:45	54:43	126.908	128.644	198.592	951	2.089	3.986
$1/x^3$	4:38	11:04	25:21	132.536	134.428	225.884	993	2.189	4.537
r^2/x^2	2:02	9:12	14:30	31.072	37.980	80.490	242	639	1.650

4.1.1 Testergebnisse

Die Ergebnisse aus Tabelle 4.1 und die erzielten Visualisierungsresultate lassen verschiedene Schlussfolgerungen hinsichtlich von Rechengeschwindigkeit und einer Erfüllung der in der Einleitung aufgestellten Anforderungen an eine ideale Gefäßvisualisierungsmethode zu. Für die Einschätzung der korrekten Durchmesserabbildung und der visuellen Qualität wurden die Testmodelle mit der in [HPSP01] vorgestellten Methode (siehe Abschnitt 2.3.2.2) visualisiert. Diese bildet einen gegebenen Durchmesserverlauf durch Konkatenation von Kegelstümpfen exakt ab. Die so konstruierten Modelle wurde mit den neuen Visualisierungsresultaten verglichen. Die im folgenden festgehaltenen Beobachtungen haben den weiteren Entwurf der Gefäßvisualisierung mit *Convolution Surfaces* wesentlich beeinflusst.

4.1.1.1 Rechenzeit

Wie sowohl die Messungen in [She99b] als auch die Schlussfolgerungen in [CH01] und [HAC03] voraussagen, erzielt die inverse kubische Funktion die besten Performanzergebnisse, gefolgt von r^2/x^2 und der Gauß-Funktion. Die Rechenzeit beträgt weniger als 50 % der gemessenen Zeit unter Verwendung eines Gauß-Filters, trotz einer höheren Anzahl von Feldfunktionsberechnungen. Die Faltung des Skeletts mit der Filterfunktion aus [HAC03] zeigt ebenfalls bessere Performanzwerte. Insgesamt muss jedoch festgestellt werden, dass die gemessenen Zeiten jenseits einer für den klinischen Betrieb akzeptablen Grenze liegen. Optimierungen mit Hinblick auf eine Reduzierung der Rechenzeit sind daher dringend erforderlich.

4.1.1.2 Durchmesserverlauf

Ein Vergleich des Visualisierungsresultats nach [HPSP01] mit den Ergebnissen nach [Blo95b] und [CH01] zeigt, dass beide den gegebenen Durchmesserverlauf grob befolgen. Der relative Verlauf wird korrekt wiedergegeben. Ist ein Gefäß zur Wurzel hin dicker als zur Peripherie, so ist dies auch aus der Visualisierung erkennbar (Abb. 4.2 rechts oben u. links unten). Die Gefäßstrukturen wirken jedoch teils wulstig und scheinen insgesamt dicker als die Konstruktion nach [HPSP01].

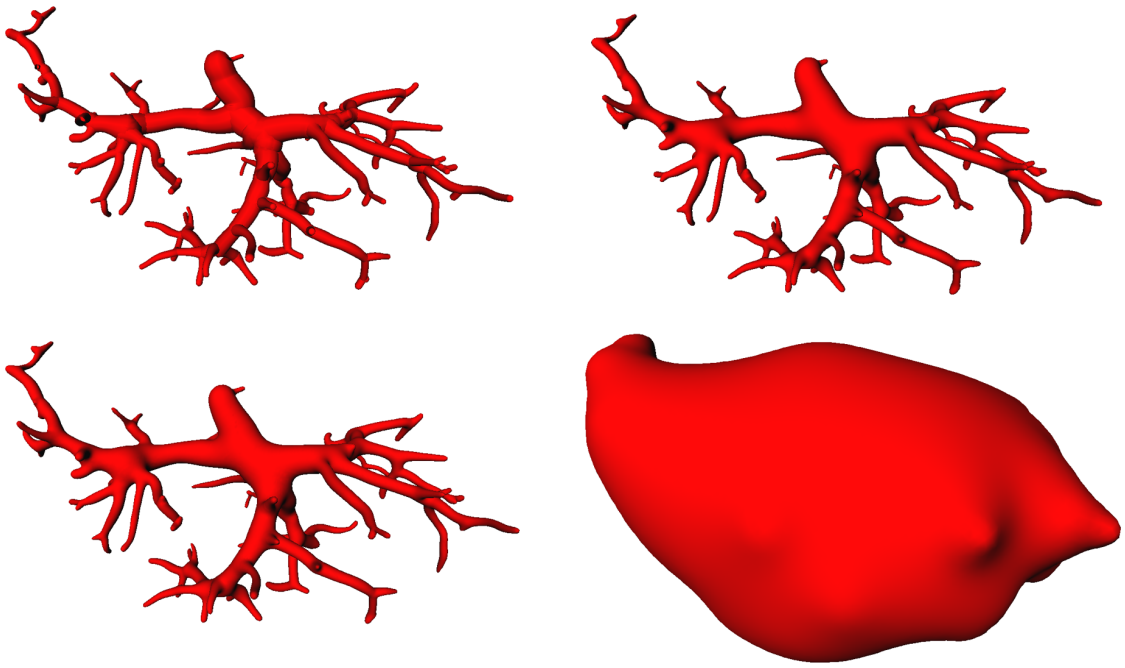


Abbildung 4.2: Lebergefäßbaum LG_1 modelliert durch die Konkatenation von Kegelstümpfen nach [HPSP01] (*links oben*) und durch *Convolution Surfaces* nach [Blo95b] (*rechts oben*), [CH01] (*links unten*) und [HAC03] (*rechts unten*).

Die korrekte Abbildung des Durchmessers entlang eines isolierten Liniensegments (Abb. 4.3 links u. Mitte) lässt vermuten, dass die wulstige Form durch *blending* der *Convolution Surfaces* benachbarter Liniensegmente verursacht wird. Um eine korrekte Durchmesserabbildung zu gewährleisten, sollte dieses Phänomen genauer untersucht werden. Es ist zu erwarten, dass eine Reduzierung der Blendingstärke sowohl in einem enger dem Skelett folgenden Gefäßverlauf resultiert, als auch die wulstige Form der Gefäße mindert und damit die Qualität der Durchmesserabbildung verbessert.

Abbildung 4.3: Liniensegment und korrespondierende *Convolution Surface* modelliert mit Hilfe einer Gauß-Funktion [Blo95b] (*links*), der inversen kubischen Funktion [CH01] (*Mitte*) und einer Abwandlung der inversen quadratischen Funktion [HAC03] (*rechts*). Das Segment hat eine Länge von 20 mm. Die Radien an den Endpunkten sind gleich 3 mm. Der Zylinder ist mit parametrischen Methoden zu Testzwecken konstruiert worden und hat einen Radius von exakt 3 mm.

Die Verwendung der Filterfunktion aus [HAC03] führte in der Anfangsphase zu überraschenden Ergebnissen (Abb. 4.2 rechts unten und Abb. 4.3 rechts). Nach zahlreichen scheinbar gescheiterten Implementierungsversuchen sollte eine Kontaktaufnahme mit einem der Autoren, Alexis Angelidis, Klarheit bringen. Dieser stellte seine Implementierung der Feldfunktion zur Verfügung was jedoch keine Veränderung des Visualisierungsresultats zur Folge hatte. In einem erneuten

Schriftwechsel wurde die beträchtliche Abweichung des Durchmessers der *Convolution Surface* wie folgt erklärt: „Wenn du einen Endpunkt C_0 getrennt von dem Rest des Segments betrachtest, würdest Du eine Kugel haben mit einem Radius welcher einzig und allein durch r_0 kontrolliert wird. Durch die Kontribution des restlichen Segments wird die Oberfläche jedoch aufgeblasen [Angelidis]“. Im Unterschied zu [Blo95b] und [CH01] wird in [HAC03] der abzubildende Radius vor der Faltung in die Filterfunktion integriert, d.h. die Filterbreite variiert während der Integration. Dies sichert nach [HAC03] eine mathematisch korrekte Faltung führt jedoch offensichtlich nicht zu einer korrekten Durchmesserabbildung. Zur Lösung des Problems wurde eine Erhöhung des Isowerts oder eine Skalierung der Feldfunktion mit einem konstanten Wert vorgeschlagen. Genauere Angaben zur Bestimmung dieser Werte im Rahmen der korrekten Durchmesserabbildung sind leider nicht gemacht worden. Die Abschnitte 4.2.1 und 4.3.2.4 liefern zusätzliche Argumente, welche die Vernachlässigung der Filterfunktion aus [HAC03] im Rahmen dieser Arbeit rechtfertigen.

4.1.1.3 Visuelle Qualität

Die Visuelle Qualität der ersten mit *Convolution Surfaces* erzeugten Gefäßstrukturen wurde durch einen erfahrenen Radiologen und Gefäßexperten (Dr.med. Holger Bourquain; MeVis, Bremen) eingeschätzt. Die artefaktfreie und glatte Oberfläche entlang der Verzweigungen wird als wesentlicher Fortschritt gegenüber bisheriger Methoden (siehe Abschnitt 2.3) angesehen (Abb. 4.4).

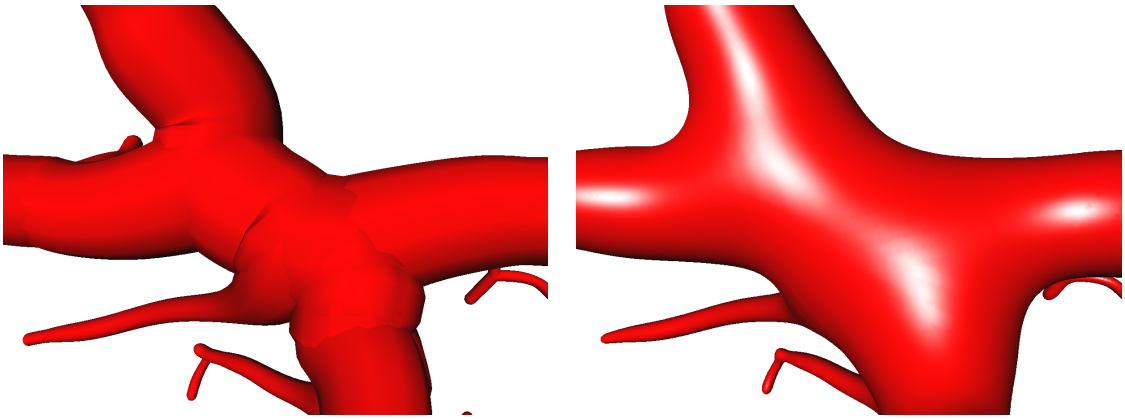


Abbildung 4.4: Verzweigung in LG_1 modelliert mit parametrischen Oberflächen nach [HPSP01] (*links*) und impliziten Oberflächen nach [Blo95b] (*rechts*). Die Übergänge zwischen einzelnen Gefäßästen in der rechten Darstellung wirken weich und zeigen keine Diskontinuitäten der Oberflächennormalen. Die Beleuchtung wurde hier speziell gewählt, um die Glätte der Oberfläche zu hervorzuheben.

Sowohl die Visualisierung mit Hilfe des Gauß-Filters als auch unter Verwendung der inversen kubischen Funktion zeigen jedoch ein sehr weiträumiges *blending* benachbarter Gefäße. Besonders an Verzweigungen divergiert die Oberfläche zu stark von dem gegebenen Gefäßskelett. Dies konnte anhand einer Darstellung des Skeletts mit überlagerter transparenter Gefäßoberfläche beobachtet werden (Abb. 4.5).

Ferner wurde auch von Dr. Bourquain die teils wulstige Form der Gefäße kritisiert. Zusammen mit in den Visualisierungen nach [Blo95b] und [CH01] auftretendem *bulging* (Abb. 4.6) könnte

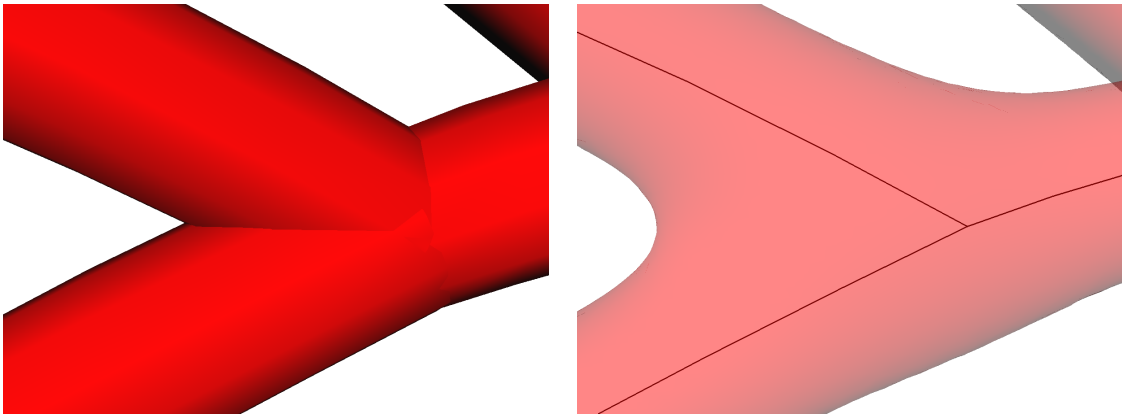


Abbildung 4.5: Bifurkation in LG_3 konstruiert nach [HPSP01] (*links*). Das nach [Blo95b] implizit beschriebene Äquivalent (*rechts*) zeigt besonders entlang des konkaven Teils der Verzweigung ein sehr weitläufiges *blending*. Die Oberfläche divergiert hier zu stark von dem gegebenen Gefäßskelett.

dies fälschlicherweise als Gefäßanomalie interpretiert werden. Hier besteht dringend Verbesserungsbedarf.

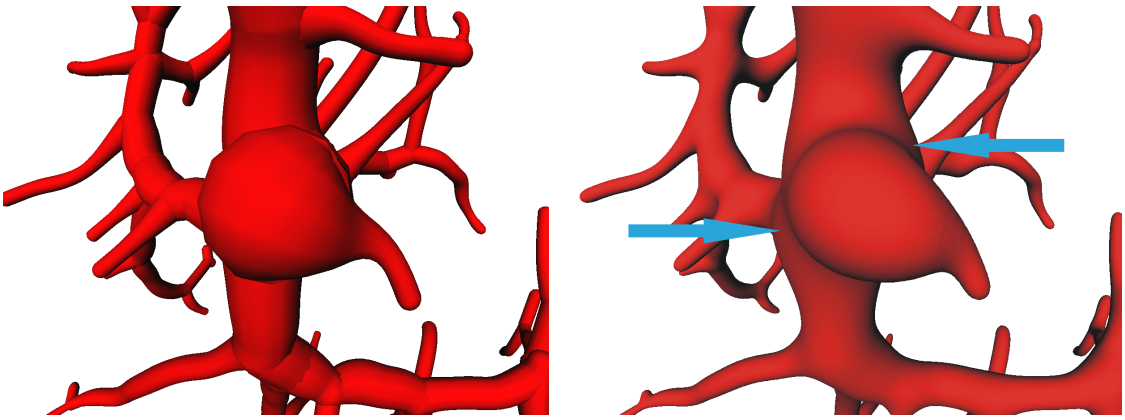


Abbildung 4.6: Verzweigung nahe der Wurzel des Gefäßbaums LG_1 modelliert mit parametrischen Oberflächen nach [HPSP01] (*links*) und impliziten Oberflächen nach [CH01] (*rechts*). Zwei Verdickungen sind in der rechten Darstellung durch Pfeile markiert.

Unwanted Blending war nur bei Gefäßbaum LG_2 entlang der *vena cava* zu beobachten (Abb. 4.7). Dies resultiert jedoch aus einer fehlerhaften Gefäßanalyse nach [SPSP02]. Statt einem einzigen dicken Gefäßast wurden viele kleine, verästelte Teilstrukturen extrahiert, deren Abstand voneinander sehr gering ist. Dennoch ermahnt das hier auftretende *Unwanted Blending* zu einer weiteren sorgfältigen Betrachtung dieses Problems.

4.1.1.4 Gefäßenden und Gefäßinneres

Die Gefäßenden aller mit impliziten Methoden visualisierten Testmodelle sind geschlossen. Die Geschlossenheit der Oberfläche um ein Primitiv ist eine inhärente Eigenschaft von *Convolution Surfaces* und bedarf keines zusätzlichen Konstruktionsaufwandes.

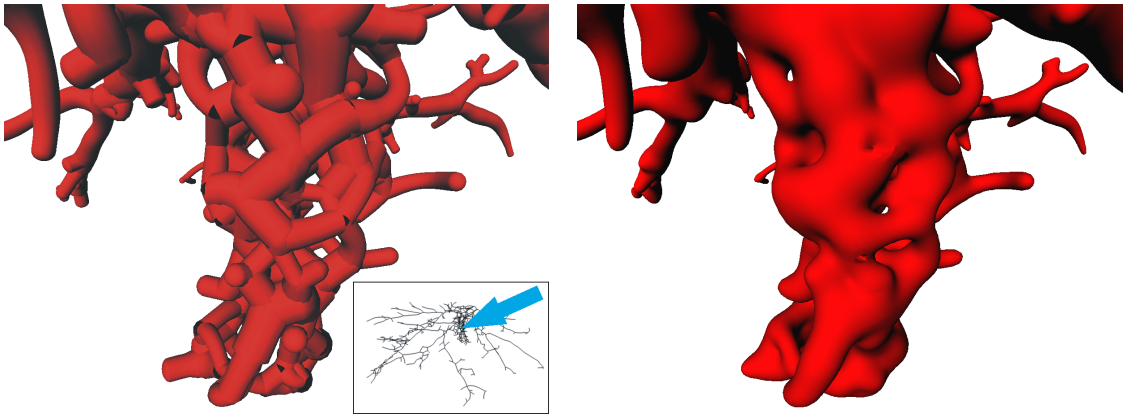


Abbildung 4.7: Die *vena cava* in LG_2 modelliert durch Konkatenation von Kegelstümpfen nach [HPSP01] (*links*) und mit Hilfe von *Convolution Surfaces* nach [CH01] (*rechts*). Das im Inset markierte Gefäßgewirr zeigt die Lage der *vena cava*. Die rechte Darstellung zeigt deutlich das *Unwanted Blending*-Problem an Gefäßästen mit geringer Entfernung zueinander.

Die Untersuchung des Gefäßinneren mittels einer Drahtmodelldarstellung zeigt, dass dort keine störenden Strukturen generiert werden. Dies resultiert aus der Eigenschaft impliziter Oberflächen ein Volumen zu umschließen. Auch die durch *blending* mehrerer Objekte generierte Gesamtoberfläche umschließt wieder ein Volumen.

Diese zwei Anforderungen an eine ideale Gefäßvisualisierungsmethode sind allein durch die Applikation impliziter Oberflächen bereits erfüllt.

4.2 Blending

Im vorigen Abschnitt wurde spekuliert, dass *blending* für die wulstige Gefäßform verantwortlich ist. Eine Modifikation der Blendingstärke könnte in einer verlässlicheren Durchmesserabbildung resultieren. Dies soll im Folgenden analysiert werden.

4.2.1 Blendingstärke

Die Modellierung eines Objektes erfordert die Abbildung sowohl globaler als auch lokaler Objekteigenschaften. Im Falle der skelettbasierten Modellierung mit *Convolution Surfaces* sind die globalen Objekteigenschaften, wie das allgemeine Layout, die Position und die grundlegende Form der Objektteile, durch das Skelett beschrieben [She99a]. Die Definition lokaler Charakteristika rundet die Objektbeschreibung ab. Im Rahmen der Modellierung von Atomen wurden durch [Bli82] zwei dieser lokalen Eigenschaften etabliert: *radius in isolation* und *blobbiness*. Der *radius in isolation* beschreibt den Radius eines isolierten Atoms, welches nicht durch *blending* mit anderen Atomen verschmilzt. Die Entsprechung in der skelettbasierten Modellierung mit Liniensegmenten ist der Durchmesserverlauf entlang eines isolierten Segments. Dieser ist durch an den Segmentendpunkten definierte Radien beschrieben. Die *blobbiness* ist ein Maß für die Blendingstärke benachbarter Objektteile. Sie kann für jedes Skelettprimitiv getrennt definiert werden. Verschmelzen Objektteile mit hoher *blobbiness*, resultiert daraus ein kugelähnliches Objekt, welches wenig bis gar kein Detail der durch das Skelett vorgegebenen Form erhält [She99a].

Objekte mit geringer *blobbiness* ähneln stärker ihrem Skelett (Abb. 4.8). Mit Blick auf die in Abb. 4.2 (links oben u. rechts unten) und Abb. 4.5 veranschaulichten Effekte scheint eine Reduktion der *blobbiness* entlang der Gefäßstruktur notwendig.

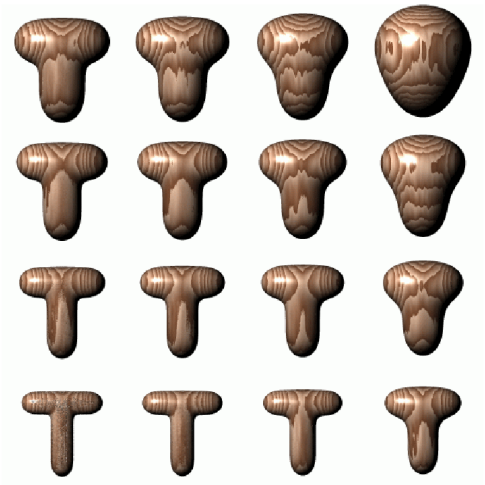


Abbildung 4.8: *Convolution Surfaces* eines T-förmigen Skeletts aus zwei Liniensegmenten. Anstieg von *blobbiness* (von links nach rechts) und *radius in isolation* (von unten nach oben). *Quelle*: [She99a]

Für die Modellierung von Atomen wird in [Bli82] die Verteilung der Elektronendichte um das Zentrum des Atoms durch eine Gauß-Funktion beschrieben (siehe Glg. 3.2). Eine Kontrolle der *blobbiness* ist hier durch die Modifikation der Standardabweichung a möglich. Die Größe der Standardabweichung entscheidet über die Breite der Gauß-Funktion. So resultiert eine Senkung des Wertes von a in einem schmaleren Trägerintervall. Äquivalent hierzu kann bei der Modellierung mit *Convolution Surfaces* die Filterbreite variiert werden [She99a]. Eine schmalere Filterfunktion geht mit einem engeren Skalarfeld um das geglättete Primitiv einher (Abb. 4.9). Daraus folgt eine geringere Überlappung der Skalarfelder benachbarter Skelettprimitive und somit ein geringeres *blending* und eine Reduktion der *blobbiness*.

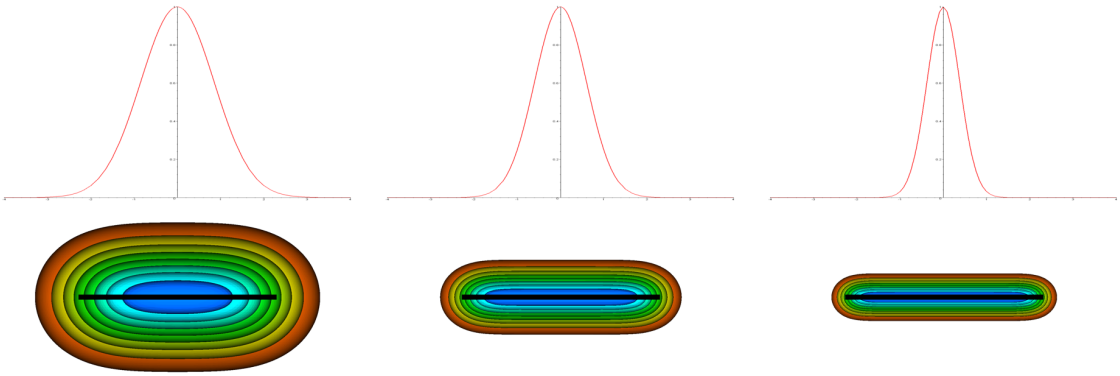


Abbildung 4.9: Faltung eines Liniensegments mit abnehmender Filterbreite (von links nach rechts). Das resultierende Skalarfeld ist mit geringerer Filterbreite enger um das Primitiv gepackt. Die Isokonturen kennzeichnen Skalarwerte von ≈ 0 (außen) bis 1.0 (innen). Der Gauß-Filter besitzt zwar ein unendlich breites Trägerintervall jedoch sind die Funktionswerte ab einer bestimmten Entfernung vom Filterzentrum vernachlässigbar klein.

Die Untersuchung der Abschnitt 3.3.2 betrachteten Filterfunktion zeigt, dass weder die inverse kubische Funktion noch die Abwandlung der inversen quadratischen Funktion über einen Parameter verfügen, dessen Modifikation eine Veränderung der Filterbreite zulässt. Der Trägerintervall des Filters ist hier fest (Abb. 3.26) bzw. durch den Radius r vorgegeben (Abb. 3.28). Die Gauß-Funktion hingegen besitzt mit der Standardabweichung eine solche Einstellmöglichkeit. Übertragen auf Gleichung 3.18 kann durch den *width coefficient* $\omega = 1/(2\sigma^2)$ die Breite des Trägerintervalls gesteuert werden. So resultiert eine Anhebung von ω in einer schmaleren Filterfunktion (Abb. 4.10).

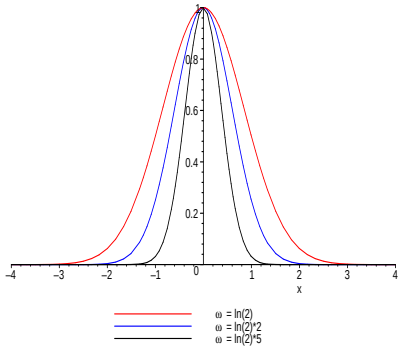


Abbildung 4.10: Gauß-Funktion $h(x) = e^{-\omega x^2}$ mit verschiedenen *width coefficients* ω. Mit steigendem Wert von ω sinkt die Breite der Funktion.

Die Auswirkungen einer systematischen Modifikation von ω auf die Form der *Convolution Surface* wurden zuerst am Beispiel einer künstlichen Trifurkation untersucht (Abb. 4.11). Verzweigungen höherer Ordnung konnten bei den im Rahmen dieser Arbeit visualisierten Gefäßstrukturen nicht beobachtet werden. Der in [Blo95b] vorgeschlagene *width coefficient* $\omega = ln(2)$ wurde mit einem Faktor $i = 2, 3, 4, ...$ multipliziert (siehe Abb. 4.10). Hierbei ist eine Adaption des Isowertes nach Gleichung 3.22 im Hinblick auf eine korrekte Durchmesserabbildung zu beachten. Die Visualisierung der Trifurkation zeigt, dass die Oberfläche mit steigendem *width coefficient* enger dem Skelettverlauf folgt (Abb. 4.11). Ein Nebeneffekt der Isowertanpassung besteht darin, dass die *Convolution Surface*, aufgrund eines von $1/2$ verschiedenen Isowertes, nicht mehr durch die isolierten Endpunkte des Skeletts verläuft. Stattdessen sind weiter auslaufende Gefäßäste zu beobachten. Dies entspricht sogar eher den in der Gefäßanalyse gewonnenen Daten. Der an dem Gefäßendpunkt gemessene Radius wird dort mit einem *width coefficient* $\omega = ln(2)$ und dem korrespondieren Isowert $Iso = 1/2$ nicht korrekt abgebildet (Abb. 4.11 links). Die Verjüngung der *Convolution Surface* muss hier schon vor dem Endpunkt beginnen, um diesen dann zu durchkreuzen. Mit steigendem *width coefficient*, d.h. sinkendem Isowert, ist eine Abschwächung dieses Problems zu beobachten (Abb. 4.11 rechts).

Im Zuge einer weiteren Evaluierung wurde dasselbe Testverfahren auf verschiedene klinische Daten angewandt, u.a. die Lebergefäßbäume LG_1, LG_2 und LG_3 (Abb. 4.1). Die generierten Modelle wurden zum einen subjektiv hinsichtlich ihrer *blobbiness* bewertet (Abb. 4.12). Hierbei wurde beobachtet, dass diese mit zunehmendem *width coefficient* abnimmt und die *Convolution Surface* dem Referenzmodell [HPSP01] (Abb. 4.12 (rechts unten)) mehr und mehr ähnelt.

Weiterhin wurde die Korrektheit der Durchmesserabbildung geprüft. Diese konnte anhand einer integrierten Darstellung der *Convolution Surface* und des Referenzmodells beurteilt werden. Der implizit beschriebene Gefäßbaum wurde hierzu als Drahtmodell dargestellt und der durch Kegelstümpfe modellierten Oberfläche überlagert (Abb. 4.13). Angemerkt sei hier, dass die stets

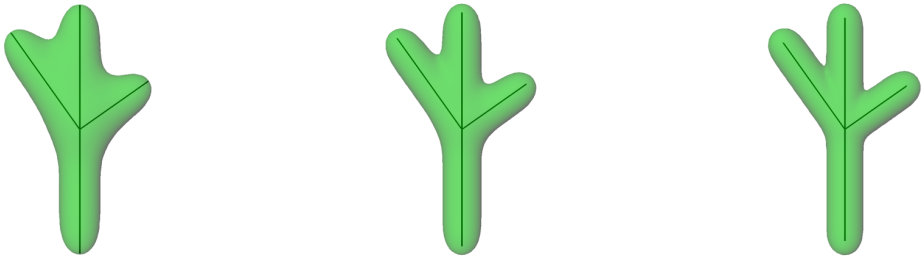


Abbildung 4.11: *Convolution Surface* einer künstlichen Trifurkation modelliert nach [Blo95b] mit den *width coefficients* $\omega = \ln 2$, $\omega = \ln 2 * 2$ und $\omega = \ln 2 * 5$ (von links nach rechts).

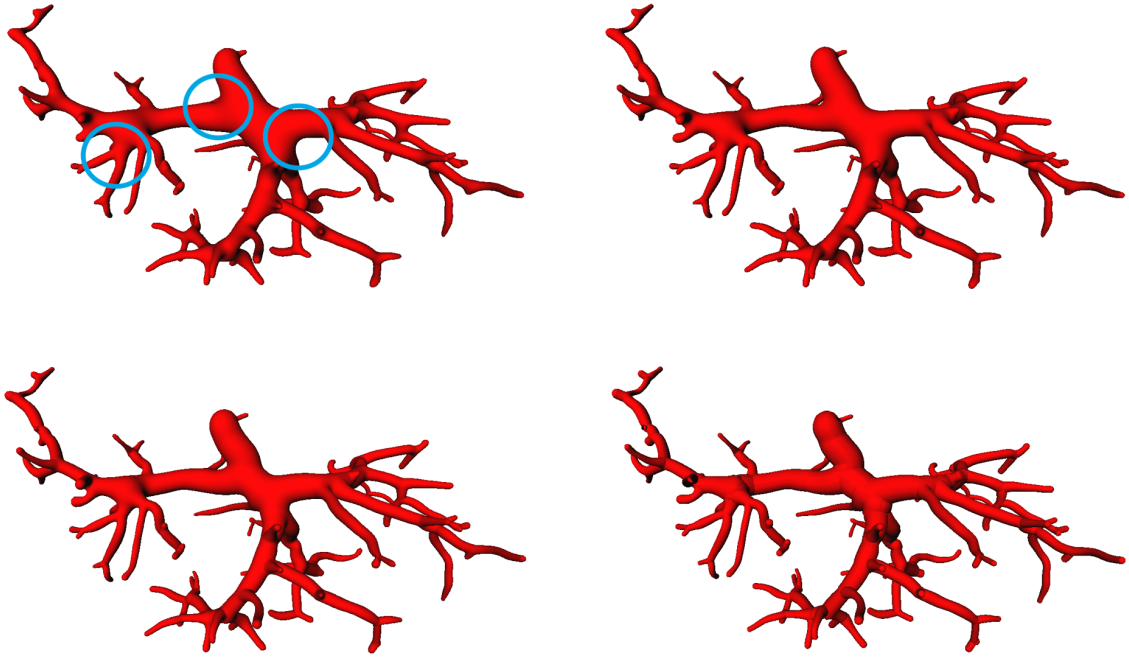


Abbildung 4.12: *Convolution Surface* des Lebergefäßbaums LG_1 modelliert nach [Blo95b] mit den *width coefficients* $\omega = \ln 2$ (*links oben*), $\omega = \ln 2 * 2$ (*rechts oben*) und $\omega = \ln 2 * 5$ (*links unten*), sowie nach [HPSP01] (*rechts unten*). An den markierten Stellen sind die Veränderungen besonders signifikant.

weiter auslaufenden Gefäßenden in dem parametrisch beschriebenen Modell aus der Halbkugelkonstruktion zum Schließen der Gefäßenden nach [HPSP01] resultieren.

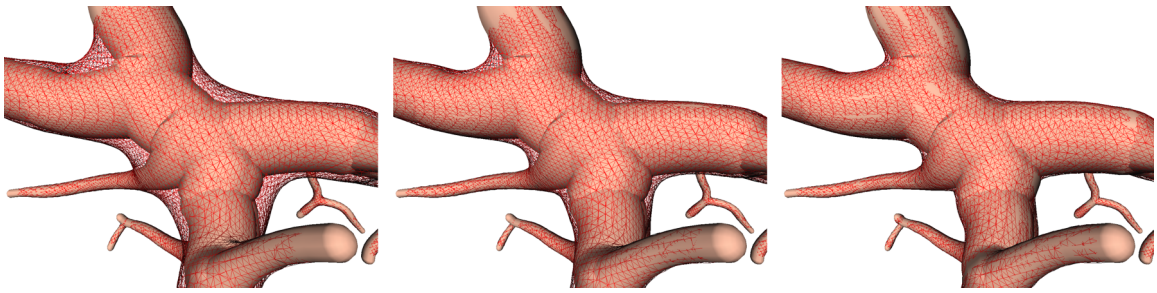


Abbildung 4.13: Nahansicht des Lebergefäßbaums LG_1. Die *Convolution Surface* (Drahtmodell) ist dem Referenzmodell [HPSP01] überlagert. Mit zunehmendem Wert ω (von links nach rechts: $\omega = \ln 2$, $\omega = \ln 2 * 2$ und $\omega = \ln 2 * 5$) folgt die *Convolution Surface* enger dem Referenzmodell.

Die Tests haben gezeigt, dass eine Erhöhung von ω ein *blending* benachbarter Gefäßteile reduziert. Dadurch wird eine Verringerung der *blobbiness* der gesamten Gefäßstruktur erreicht und die Verlässlichkeit einer korrekten Durchmesserabbildung erhöht. Weitere Tests sind jedoch notwendig, um diese Korrektheit genau quantifizieren zu können.

Die Modellierung zahlreicher klinischer Testdatensätze hat ergeben, dass ein Wert von $\omega = ln(2) * 5$ für die Gefäßvisualisierung geeignet ist. Der korrespondierende Isowert nach Gleichung 3.22 ist $Iso = e^{-\omega} = e^{-ln(2)*5} = 1/32$. Gefäßform und Durchmesserabbildung wurden von dem Radiologen Dr. Holger Bourquain im Vergleich mit den Referenzmodellen nach [HPSP01] hier als korrekt bewertet. Es ist vermutlich nicht möglich, einen Sollwert für ω im Rahmen der Gefäßvisualisierung zu bestimmen, da das genaue Aussehen der Übergänge zwischen den Gefäßästen und somit die Blendingstärke nicht den Daten zu entnehmen ist. Eine obere Grenze für ω scheint jedoch gegeben. So ist mit ansteigendem *width coefficient* kaum noch oder gar kein *blending* mehr festzustellen. Dies führt an den Übergängen teilweise zu Diskontinuitäten der Normalen entlang der Gefäßoberfläche (Abb. 4.14).

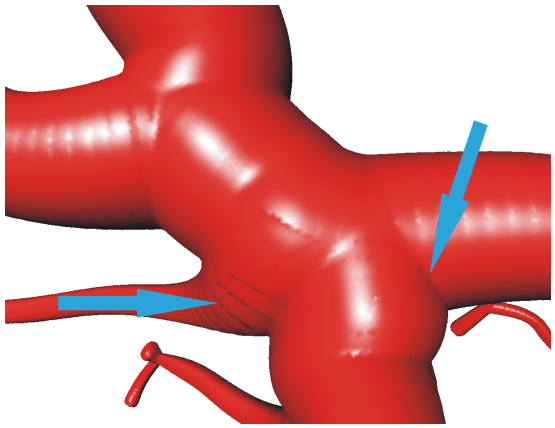


Abbildung 4.14: Nahansicht des Gefäßbaums LG_1, modelliert nach [Blo95b] mit einem sehr hohen *width coefficient* $\omega = \ln 2 * 50$. Die Pfeile markieren Einknicke an den Übergängen und Diskonituitäten der Oberflächennormalen. Für einen Qualitätsvergleich siehe Abb. 4.4 (rechts). Diese zeigt dieselbe Nahansicht von LG_1 erstellt unter Verwendung des width coefficient $\omega = \ln 2$.

Eine wulstige Gefäßform kann durch die Reduktion der Blendingstärke vermieden werden. Die präzisere Abbildung des Gefäßdurchmessers ist möglich da die Gefäßoberfläche enger der durch dass Gefäßskelett vorgegebenen Form folgt. Unter Verwendung einer Gauß-Funktion für die Modellierung mit *Convolution Surfaces* gestattet der *width coefficient* ω eine Kontrolle der Blendingstärke. Keine der in [CH01] und [HAC03] verwendeten Filterfunktionen besitzt einen solchen Parameter. Tests haben jedoch gezeigt, dass die Kontrolle der *blobbiness* einer Gefäßstruktur entscheidend sein könnte für die Vermeidung einer Fehlinterpretation wulstiger Gefäßabschnitte als pathologische Veränderungen. Trotz geringerer Berechnungskomplexität der inversen Funktionen, ist die Gauß-Funktion somit zu favorisieren. Im weiteren Verlauf dieser Arbeit werden die Filterfunktionen nach [CH01] und [HAC03] daher vernachlässigt und nur zu gegebenem Anlass in Abschnitt 4.3.2.4 erneut betrachtet.

4.2.2 Bulging

Die skelettbasierte Modellierung mit Hilfe von *Convolution Surfaces* vermeidet die Entstehung von Verdickungen entlang nicht verzweigter Strukturen. Das Skelett einer Gefäßstruktur ist jedoch hierarchisch organisiert und besitzt Verzweigungen unterschiedlicher Ordnung. Die innerhalb dieser Arbeit verwendeten klinischen Datensätze zeigen sowohl Bifurkationen als auch seltene Trifurkationen.

Das Verhalten der *Convolution Surface* entlang von Verzweigungen wurde bereits in [Blo95a] und [Blo95b] untersucht. Bei jedem der verwendeten Testmodelle (Abb. 3.9 und Abb. 3.12) waren Verdickungen entlang der Verzweigung festgestellt worden. Dieser unerwünschte Effekt konnte ebenfalls in den ersten Gefäßvisualisierungen in Abschnitt 4.1 beobachtet werden (siehe Abb. 4.6). Da *bulges* fälschlicherweise als Gefäßanomalien, z.B. Aneurysmen, interpretiert werden könnten, ist ihr Auftreten dringend zu unterbinden. Im Folgenden soll daher die Eignung der in Abschnitt 3.2.3 beschriebenen Methoden zur Vermeidung von *bulging* ([Blo95a], [Blo95b]) im Rahmen der Gefäßvisualisierung analysiert werden. Vorab scheint jedoch die Untersuchung von Auswirkungen der Filtermodifikation in Abschnitt 4.2.1 auf *bulging* sinnvoll. Welchen Einfluss hat die Verwendung eines engeren Gauß-Filters bei der Faltung auf das *bulging*-Problem ?

4.2.2.1 Auswirkungen der Filtermodifikation

Der in [Blo95b] vorgestellte Gauß-Filter hat einen *width coefficient* $\omega = \ln 2$. In Abschnitt 4.2.1 hat sich ein Wert von $\omega = \ln 2 * 5$ für die Gefäßvisualisierung als günstig erwiesen. Durch die modifizierte Filterfunktion wird eine Reduzierung der *blobbiness* entlang der Gefäßstruktur erreicht. Die *Convolution Surface* folgt enger der durch das Gefäßskelett vorgeschriebenen Form und bildet somit den gegebenen Durchmesserverlauf genauer ab. Um die Auswirkungen der Modifikation des *width coefficients* auf das *bulging*-Problem quantifizieren zu können, wird nun die *Convolution Surface* der künstlichen Trifurkation aus Abb. 4.11 näher untersucht. Dazu ist das Modell sowohl mit Hilfe der originalen Filterfunktion nach [Blo95b] als auch unter Verwendung des modifizierten Filters generiert worden. Zur weiteren Illustration siehe Abb. 4.15.

Entlang jedes Gefäßastes der Trifurkation ist ein Radius von 3 mm definiert. Daher sollte das Modell in der Seitenansicht eine Dicke von exakt 6 mm besitzen. Diese Sollausdehnung ist durch vertikale Linien markiert. Die beiden Punkte links und rechts dieser Linien kennzeichnen jeweils den Punkt mit der größten Abweichung von dem angestrebten Gefäßradius. Obwohl die

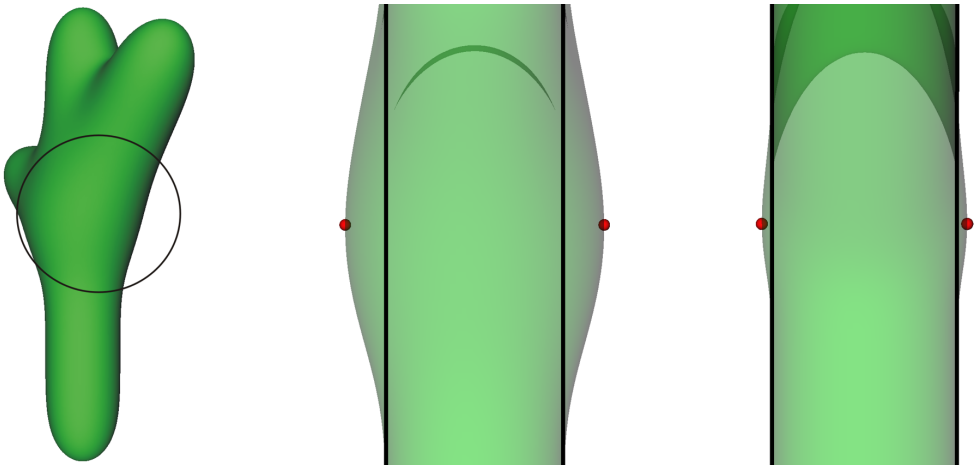


Abbildung 4.15: Ausmaß der Verdickung entlang der Verzweigung einer künstlichen Trifurkation. Der betrachtete Abschnitt ist durch eine Umrandung hervorgehoben (*links*). Die Seitenansichten zeigen die Trifurkation modelliert mit dem Gauß-Filter nach [Blo95b] (*Mitte*) und der modifizierten Filterfunktion (*rechts*). Die vertikalen Linien beschreiben die Sollausdehnung. Die größte Ausdehnung in horizontaler Richtung ist durch Punkte gekennzeichnet.

Struktur aus diesem Blickwinkel symmetrisch sein sollte, da alle Gefäßäste in derselben Ebene liegen, sorgt die Polygonisierung hier für geringe Differenzen. Die mittlere Radiusabweichung in der Konstruktion mit dem Originalfilter beträgt $\approx$ 1.4 mm (Abb. 4.15 Mitte). Der fehlerhafte Zuwachs des Gefäßdurchmessers entspricht somit 46.7% des abzubildenden Durchmessers. Durch die Verwendung der modifizierten Filterfunktion konnte eine Verringerung der Radiusabweichung auf $\approx$ 0.3 mm erreicht werden (Abb. 4.15 rechts). Der fehlerhafte Zuwachs des Gefäßdurchmessers wurde dadurch auf 10.9% des abzubildenden Durchmessers reduziert.

Die Verwendung einer schmaleren Filterfunktion resultiert in einer signifikanten Abschwächung des *bulging*-Problems. Die Blendingstärke, welche über die Filterbreite reguliert wird, ist demnach direkt proportional zu dem Ausmaß auftretender *bulges*. Beide Attribute sind eng miteinander verknüpft. Im Hinblick auf eine vollkommene Vermeidung von *bulging* werden im Folgenden die in Abschnitt 3.2.3 beschriebenen Methoden ([Blo95a], [Blo95b]) näher untersucht. Alle dafür generierten Illustrationen wurden mit Hilfe des originalen Gauß-Filters erstellt. Dies ermöglicht eine anschaulichere Darstellung, da das *bulging*-Problem hier offensichtlicher zu Tage tritt. Eine Generalisierung ist jedoch ohne weiteres möglich.

4.2.2.2 Separierung von Skelettprimitiven

Bulging kann durch eine Reduzierung der „Skelettdichte“ an Verzweigungen verringert werden [Blo95a]. Als Testmodell dient erneut die künstliche Trifurkation aus Abb. 4.11. Eine Reduzierung der Dichte des Skeletts wird durch die Separierung von Skelettprimitiven erreicht. Dies ist gleichbedeutend mit einer Verkürzung der Gefäßäste ausgehend von dem Verzweigungspunkt (Abb. 4.16).

Leider untersucht [Blo95a] nicht, um welchen Betrag genau jedes Segment gekürzt werden muss. Die Abstände in Abb. 4.16 wurden empirisch bestimmt. Abbildung 4.17 zeigt die *Convolution Surfaces* der originalen Skeletts und der modifizierten Version. Interessanterweise ist hier eben-

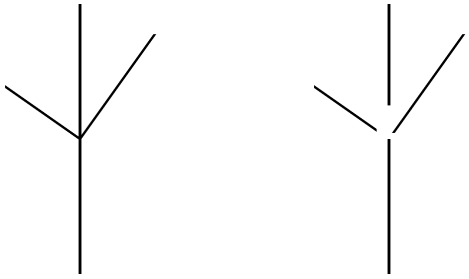


Abbildung 4.16: Separierung von Skelettprimitiven an einer Verzweigung. Die verzweigenden Segmente des originalen Skeletts (*links*) werden zur Reduzierung der Skelettdichte ausgehend von dem Verzweigungspunkt gekürzt (*rechts*).

falls das in [Blo95a] beschriebene Phänomen einer seichten Eindellung gefolgt von einer geringen Verdickung zu beobachten (Abb. 4.18).

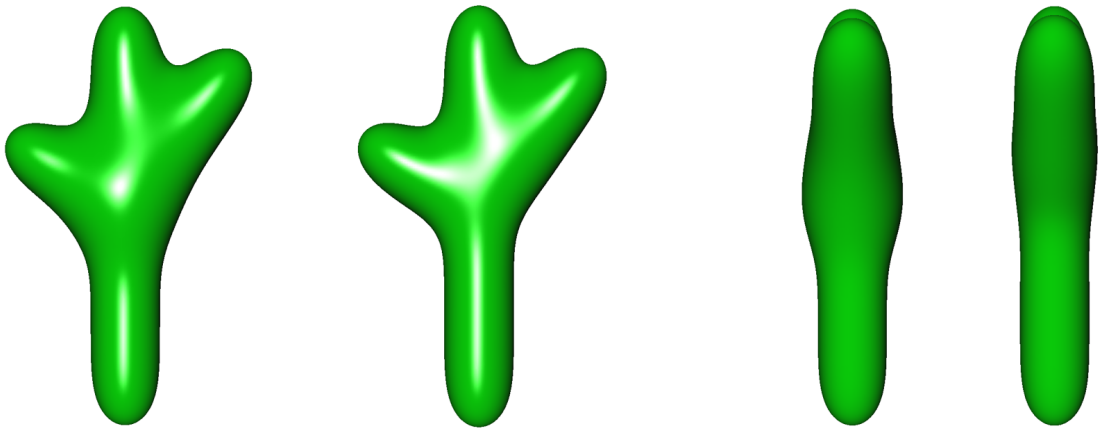


Abbildung 4.17: Vorderansicht und Seitenansicht einer Trifurkation modelliert basierend auf dem Originalskelett (*links*) und der durch Separierung modifizierten Version (*rechts*). Eine deutliche Reduzierung des *bulges* entlang der Verzweigung kann durch Separierung von Skelettprimitiven erreicht werden.

Bulging wird nach [Blo95a] durch die Separierung von Skelettprimitiven verringert. Eine komplette Behebung des Problems kann dagegen nicht erzielt werden. Diese Behauptung ist aufgrund empirischer Untersuchungen kaum zu validieren. Leider werden in [Blo95a] und [Blo95b] keine genaueren Aussagen getroffen. Aus Zeitgründen konnte eine intensivere Untersuchung der Methode innerhalb dieser Arbeit nicht angestrengt werden. Ein Gedanke sei jedoch an dieser Stelle notiert. Die Testmodelle aus [Blo95a] und [Blo95b] (Abb. 3.9 und Abb. 3.12) sind komplett in einer Ebene modelliert. Dies macht eine Lokalisation des *bulges* entlang der Verzweigung einfach. Unklar ist jedoch wo genau die Verdickung an einer Bi- oder Trifurkation beliebiger Morphologie einzugrenzen ist. Eine Separierung würde dies jedoch bedingen, da die Skelettprimitive so separiert werden sollten, dass die Summe ihrer Feldfunktionswerte an einem Punkt p konstant ist entlang der Verdickung. Hierzu ist es notwendig, die exakte Lage der Verdickung zu kennen.

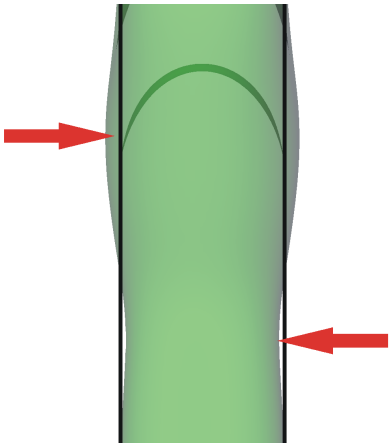


Abbildung 4.18: Die Seitenansicht der Trifurkation modelliert basierend auf dem durch Separierung modifizierten Skelett zeigt eine leichte Eindellung (unterer Pfeil) gefolgt von einer geringfügigen Verdickung (oberer Pfeil). Die vertikalen Linien kennzeichnen den Sollwert der Gefäßdicke.

4.2.2.3 Combination Surface

Die Untersuchung einer Verzweigung vierter Ordnung in [Blo95a] (Abb. 3.12) hat gezeigt, dass *blending* nicht entlang der gesamten Verzweigung wünschenswert ist. Stattdessen sollte die Blendingstärke basierend einer Ebene variiert werden, welche durch die Verzweigung verläuft. Diese Ebene ist durch den Verzweigungspunkt und eine Ebenennormale definiert. Die Normale entspricht der eines „fast“ planaren Polygons, welches durch die Endpunkte der verzweigenden Segmente aufgespannt wird. Sie kann mit Hilfe von Newell's Methode nach [Sun02] berechnet werden. Liegen alle Punkte der Verzweigung in einer Ebene, so verläuft das Polygon exakt durch die gesamte Punktmenge (Abb. 4.19). Ist dies jedoch nicht der Fall, so sind Polygon und korrespondierende Ebene nur eine Approximation durch die Punktwolke.

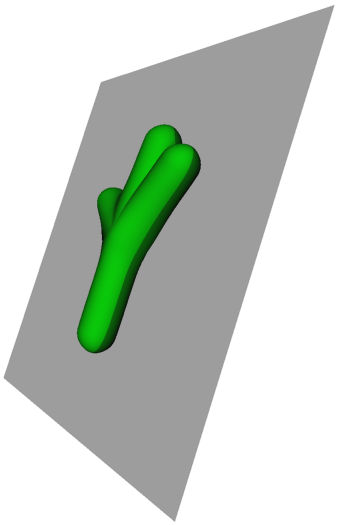


Abbildung 4.19: Ebene durch die Verzweigung bestimmt nach [Sun02]. Basierend auf dieser Ebene wird die Blendingstärke entlang der Verzweigung reguliert.

Innerhalb der Ebene sollte *blending* erfolgen, um die weichen Übergänge zwischen einzelnen Gefäßästen zu gewährleisten. Außerhalb der Ebene sollte es jedoch vermieden werden, um *bulging* auszuschließen. Hierzu wird der Skalarwert an einem Punkt p durch eine gewichtete Kombination aus *Union Surface* (kein *blending*) und *Convolution Surface* (*blending*) berechnet. Die Wichtung erfolgt mit Hilfe des Parameters *convexity* aus Gleichung 3.12. Dieser basiert auf dem Winkel zwischen der Oberflächennormalen der Convolution Surface an der Stelle p und der Ebenennormale. Bei einem Winkel von 90° liegt p innerhalb der Ebene und *convexity* ist gleich 0. Daher besitzt die *Union Surface* hier keinerlei Einfluss. Ein Winkel von 0° hingegen resultiert in dem Wert 1 für *convexity* und der Einfluss der *Convolution Surface* wird vollkommen vernachlässigt. Zwischen diesen Winkeln wird interpoliert. Für die weitere Evaluierung der *Combination Surface* im Rahmen der Gefäßvisualisierung wird wiederum die Trifurkation aus Abb. 4.11 herangezogen. Abbildung 4.20 zeigt die *Union Surface*, die *Convolution Surface* und die *Combination Surface* dieser Trifurkation.

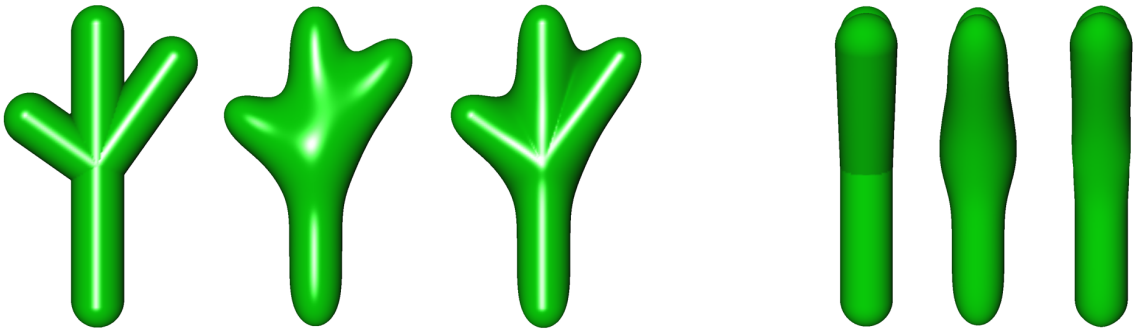


Abbildung 4.20: Vorderansicht und Seitenansicht einer Trifurkation zeigen jeweils die *Union Surface*, die *Convolution Surface* und die *Combination Surface* (von links nach rechts). Die „guten“ Eigenschaften von *Union Surface* (kein *bulging*) und *Convolution Surface* (weiche Übergänge zwischen den Gefäßästen) sind in der *Combination Surface* vereint.

Wie in einer Seitenansicht der Trifurkation zu sehen, wird die Verdickung an der Verzweigung durch die *Combination Surface* vermieden (Abb. 4.20 rechts). Leider sind jedoch leichte Diskontinuitäten der Oberflächennormalen entlang der Verbindungen zwischen den einzelnen Gefäßästen zu beobachten (Abb. 4.21). Dies wurde bereits in [Blo95b] bemerkt und geht auf den Einfluss der *Union Surface* zurück.

Soweit erscheint die *Combination Surface* als eine vielversprechende Lösung. Um in der Gefäßvisualisierung eingesetzt werden zu können, sind dennoch weitere Tests notwendig. Wie schon weiter oben beschrieben, sind in [Blo95a] und [Blo95b] wurden nur Verzweigungen untersucht worden, welche vollständig in einer Ebene lokalisiert waren. Bei „realen“ anatomischen Gefäßstrukturen kann davon jedoch nicht ausgegangen werden. Für eine aussagekräftigere Analyse wurde daher die Morphologie der Trifurkation leicht verändert. Abb. 4.22 zeigt die *Combination Surface* des modifizierten Skeletts zusammen mit der nach [Sun02] berechneten Ebene durch die Verzweigung.

Auf den ersten Blick wirkt die *Combination Surface* artefaktfrei. Eine genauere Betrachtung zeigt allerdings eine Verdickung entlang der Verzweigung (Abb. 4.23 links). Die Rotation der Trifurkation um ihre vertikale Achse im Uhrzeigersinn (Abb. 4.23 rechts) enthüllt den Grund dafür. Per Definition der *Combination Surface* soll *blending* innerhalb der Ebene stattfinden. Der

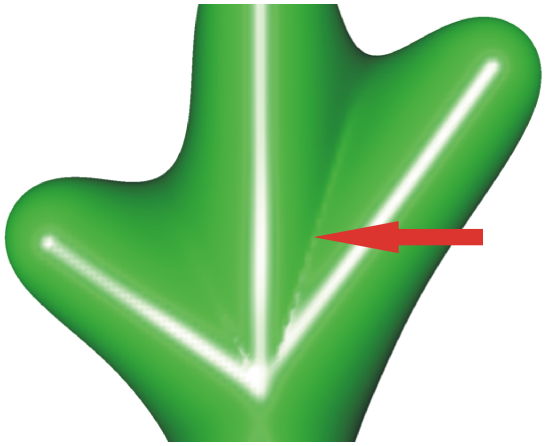


Abbildung 4.21: Eine Nahansicht der *Combination Surface* offenbart die schon in [Blo95b] diskutierten leichten Diskontinuitäten (Pfeil) zwischen den einzelnen Segmenten.

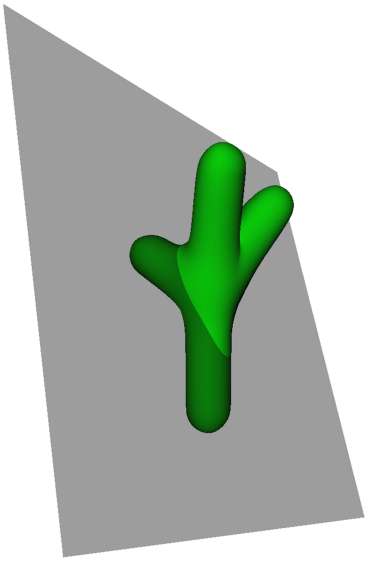


Abbildung 4.22: Modifikation der Trifurkation aus Abb. 4.19 und die korrespondierende Ebene durch die Verzweigung nach [Sun02].

Winkel zwischen Ebenennormale und Oberflächennormale ist hier nah 90° und die *Convolution Surface* hat einen hohen Einfluss. Gerade an dieser Stelle ist jedoch keine weiche Verbindung zwischen Gefäßästen erforderlich und der Einfluss der *Union Surface* sollte dominieren.

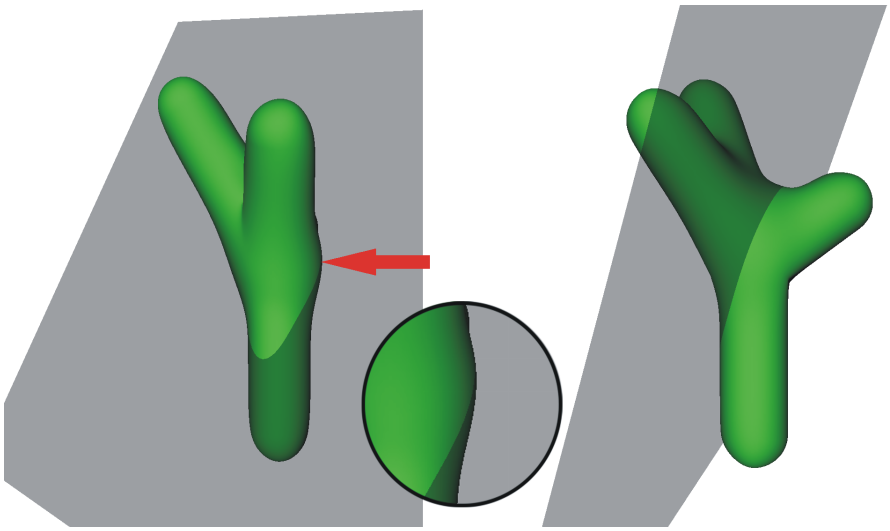


Abbildung 4.23: Die Nahansicht der *Combination Surface* zeigt eine Verdickung entlang der Verzweigung (*links*). Eine Rotation des Modells (*rechts*) erleichtert das Verständnis dieses Phänomens. *Blending* ist an dieser Stelle unnötig, findet aber per Definition in der Ebene statt und verursacht den *bulge*.

Weiterhin sind Diskontinuitäten zwischen den Gefäßästen zu beobachten (Abb. 4.24). An beiden in der Abbildung gezeigten Stellen ist der Winkel zwischen Ebenennormale und Oberflächennormale nah 0°. Aufgrund der Definition der *Combination Surface* darf hier kein *blending* stattfinden und der Einfluss der *Union Surface* überwiegt. Genau an diesen Stellen sollten jedoch weiche Übergänge generiert und *blending* durch die *Convolution Surface* realisiert werden.

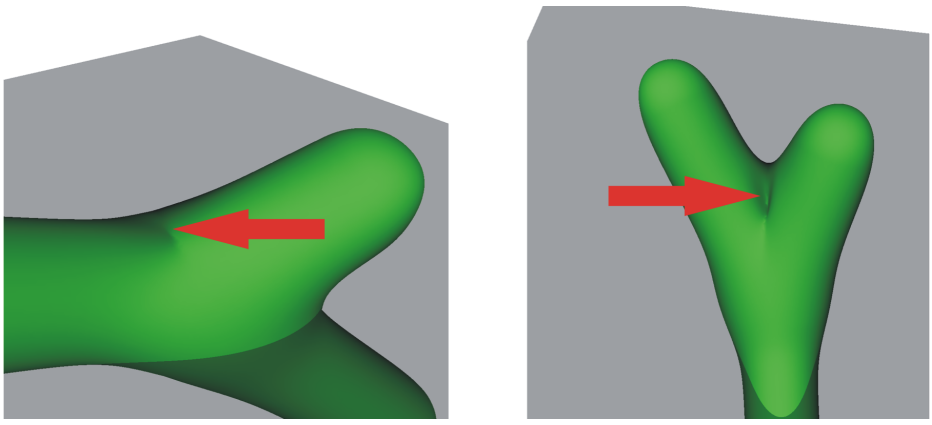


Abbildung 4.24: Nahansichten der *Combination Surface* enthüllen Diskontinuitäten entlang der Verzweigung. *Blending* wäre an diesen Stellen erforderlich, wird aber aufgrund des geringen Winkels zwischen Ebenennormale und Oberflächennormale per Definition vermieden.

Es ist festzustellen, dass die Klärung wo entlang einer Verzweigung *blending* stattfinden soll, nur im planaren Fall mit Hilfe der in [Blo95b] vorgeschlagenen approximativen Ebene korrekt erfolgen

kann. Hier ist die Lokalisation des *bulges* bekannt und die Vermeidung kann mit Hilfe der *Combination Surface* gezielt gesteuert werden. Sind die Segmente jedoch vom Verzweigungspunkt aus strahlenförmig angeordnet, ist die Lage der Verdickung schwer einzugrenzen. Die approximative Ebene liefert keinen verlässlichen Hinweis in diesem Fall und das Auftreten von Artefakten ist die Folge. Aus diesem Grund ist die *Combination Surface* nicht für die Gefäßvisualisierung geeignet. Weiterhin ist die Berechnungskomplexität der *Combination Surface* höher als die der *Convolution Surface*, da zusätzlich die *Union Surface* berechnet werden muss. Hier ist vor allem die besondere Sorgfalt bei der Vermeidung unstetiger Oberflächennormalen mit beträchtlichem Rechenaufwand verbunden (siehe Anhang in [Blo95b]). Zudem muss für jede Verzweigung mit Hilfe von Newell's Methode eine Ebenennormale berechnet werden.

4.2.2.4 Höherdimensionale Skelettprimitive

Eine letzte, in [Blo95a] eingeführte Technik zur Vermeidung von Verdickungen ist der Ersatz i-dimensionaler durch $i+1$-dimensionale Skelettprimitive. So werden Liniensegmente (1D) durch Polygone (2D) ersetzt Abb. 3.14 (links). Diese Polygone dürfen sich nicht überlappen, um erneute Verdickungen zu vermeiden (Abb. 3.14). Weiterhin muss ihre Breite mindestens der Filterbreite entsprechen, da sonst ebenfalls *bulging* zu beobachten ist. Die Abstimmung von Polygon- und Filterbreite im Hinblick auf die Vermeidung von *bulges* und eine korrekte Durchmesserabbildung führt wie in [Blo95a] gezeigt, zu einer Formänderung der *Convolution Surface* von kreisförmig zu ellipsoidal (Abb. 4.25). Obwohl tatsächlich anzunehmen ist, dass anatomische Gefäßstrukturen keine exakt kreisrunden Querschnitte besitzen, ist die hier eingeführte zusätzliche Richtungsinformation willkürlich und nicht in den Daten enthalten. Diese sollten jedoch im Rahmen einer korrekten Visualisierung unverfälscht abgebildet werden. Die Nutzung höherdimensionaler Skelettprimitive kommt daher für die Gefäßvisualisierung nicht in Frage. Ließe sich während der Gefäßanalyse jedoch ein weiterer Radius orthogonal zu dem bisher gemessenen Radius bestimmen, könnten sich Polygone als geeignete Alternative zu Liniensegmenten durchsetzen. Die höhere Berechnungskomplexität der Faltung zweidimensionaler Skelettprimitive wiederum wirft einen kleinen Schatten auf diese Methode zur Vermeidung von *bulging*. Die recht komplexe Faltung eines Polygons ist im Anhang von [Blo95b] beschrieben. Wie schon in Abschnitt 3.3.2.1 diskutiert, ist hier eine Integration der Filterfunktion in zwei Richtungen notwendig.

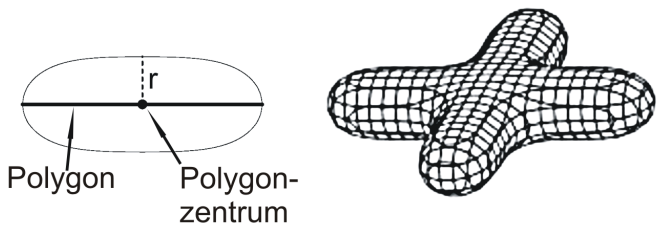


Abbildung 4.25: Querschnitt durch eine ellipsoide *Convolution Surface* (*links*). Dargestellt ist die notwendige Polygonbreite, um einen Radius der Größe r unter Verwendung des originalen Gauß-Filters aus [Blo95b] abbilden zu können. Hieraus resultiert eine ellipsoide Oberfläche (*rechts*). Nach [Blo95a]

4.2.3 Unwanted Blending

Das *Unwanted Blending*-Problem beschreibt ein unbeabsichtigtes Verschmelzen von Oberflächen nicht direkt benachbarter Skelettprimitive. Für die Gefäßvisualisierung würde dies eine Veränderung der Morphologie des Gefäßbaums bedeuten, welche dringend unterbunden werden muss. Obwohl *Unwanted Blending* in den ersten Gefäßvisualisierungen in Abschnitt 4.1 nur entlang fehlerhaft segmentierter Gefäßabschnitte betrachtet werden konnte (Abb. 4.7), ist das Auftreten dieses Artefakts nicht gänzlich auszuschließen. In Abschnitt 3.2.4 wurden bereits einige Verfahren zur Vermeidung von *Unwanted Blending* vorgestellt. Im Folgenden liegt der Fokus auf der in [CH01] eingeführten *Local Convolution*, da diese im Gegensatz zu der Verwendung von *Blending*-Kategorien [OM93] die Kontinuität der Oberfläche gewährleistet. Vorab scheint jedoch eine Untersuchung von Auswirkungen der Filtermodifikation in Abschnitt 4.2.1 auf *Unwanted Blending* sinnvoll. Welchen Einfluss hat die Verwendung eines engeren Gauß-Filters bei der Faltung auf dieses Problem ?

4.2.3.1 Auswirkungen der Filtermodifikation

Der originale Gauß-Filter aus [Blo95b] besitzt einen *width coefficient* von $\omega = \ln 2$. In Abschnitt 4.2.1 wurde ein schmalerer Gauß-Filter mit *width coefficient* $\omega = \ln 2 * 5$ als geeignet im Rahmen der Gefäßvisualisierung eruiert. Die Auswirkungen dieser Filtermodifikation auf das *Unwanted Blending*-Problem werden im weiteren Verlauf anhand der *Convolution Surfaces* zweier isolierter Liniensegmente untersucht. Hierzu wird der Durchmesser der Oberflächen gleichmäßig erhöht, bis die *Convolution Surfaces* beider Segmente miteinander verschmelzen (Abb. 4.26). Der Abstand zwischen den Primitiven beträgt 6 mm, d.h. jede der korrespondierenden Oberflächen kann unter Berücksichtigung einer einheitlichen Dicke den Maximalradius von 3 mm erreichen. Die Konstruktion der *Convolution Surfaces* basierend auf dem Originalfilter [Blo95b] stoppt bei einem Radius von 2,12 mm (Abb. 4.26 mitte rechts). Dann ist *Unwanted Blending* zu beobachten. Dies entspricht $\approx 71\%$ des Maximalradius. Unter Verwendung der modifizierten Filterversion ist einen artefaktfreie Modellierung bis zu 2,73 mm möglich, d.h. 91% des Maximalradius (Abb. 4.26 rechts). Damit wurde der erforderliche Sicherheitsabstand zwischen den Primitiven auf weniger als 1/3 der ursprünglich notwendigen Distanz reduziert.

Abbildung 4.26: Untersuchung des *Unwanted Blending*-Problems anhand von zwei isolierten Liniensegmenten mit Abstand 6 mm (*links*). Ist der Durchmesser gering, sind zwei separate *Convolution Surfaces* erkennbar (*Mitte links*). Unter Verwendung des originalen Gauß-Filters [Blo95b] verschmelzen die beiden Oberflächen ab einem Radius von 2,13 mm (*Mitte rechts*). Durch die Filtermodifikation ist diese Verhalten erst ab 2,74 mm zu beobachten (*rechts*).

4.2.3.2 Local Convolution

Unter *Local Convolution* nach [CH01] versteht man die Eingrenzung des zu betrachtenden Skelettabschnitts bei der Berechnung der Feldfunktion an einem Punkt p auf ein lokales, zusammenhängendes Gebiet. Dadurch wird das unbeabsichtigte Verschmelzen von Objektteilen, welche entlang des Skeletts weit voneinander entfernt liegen, vermieden. Die Bestimmung des relevanten Skelettabschnitts erfolgt ausgehend von dem Skelettprimitiv mit dem höchsten Einfluss an der Stelle p. Während in [CH01] Segmente aus dessen lokaler Umgebung welche einen signifikant hohen Einfluss besitzen dem Bereich hinzugefügt werden, schlägt [AC02] die alleinige Berücksichtigung der direkten Nachbarsegmente vor. Eine erhebliche Verringerung der Rechenzeit ist in [AC02] aufgrund dieser weiteren Einschränkung des zu betrachtenden Skelettbereichs protokolliert worden. Weiterhin wird im Gegensatz zu [CH01] *Unwanted Blending* entlang sehr kurzer Knicke, geformt aus wenigen Skelettprimitiven, verhindert. Dieser zweite Aspekt ist für die Gefäßvisualisierung jedoch nicht von Bedeutung, da der anatomische Gefäßverlauf weder kurze noch weitläufige Kehrtwendungen wie in Abb. 3.19 aufweist. So verlaufen Gefäße von der Wurzel relativ kontinuierlich in Richtung Peripherie.

Die Evaluierung des verbesserten *Local Convolution*-Ansatzes für die Gefäßvisualisierung erfolgt in Anlehnung an [AC02] anhand eines S-förmigen Skeletts, bestehend aus 5 Kanten bzw. 24 Liniensegmenten (Abb. 4.27 links). Obwohl Knicke wie die der S-Struktur in Gefäßskeletten nicht zu erwarten sind, ist sie zu Illustrationszwecken gut geeignet. Um eine Vergleichbarkeit mit realen medizinischen Daten zu gewährleisten, beträgt die Segmentlänge 1 mm. Dies entspricht ungefähr einer gängigen Voxelausdehnung in x- und y-Richtung. Hier sei daran erinnert das ein Gefäßskelett aus Liniensegmenten besteht, welche benachbarte Voxelmittelpunkte verbinden. Die Knicke des S-Skelett haben eine Ausdehnung von 3 mm. Somit ist unter Berücksichtigung einer einheitlichen Dicke der *Convolution Surface* ein Maximalradius von 1,5 mm zu erreichen. Der genaue Zeitpunkt des Auftretens von *Unwanted Blending* ist für die S-Form schwer zu definieren. So verschmelzen die horizontalen Objektteile immer weitläufiger an ihrer vertikalen Verbindung, bis sie über ihre gesamte Länge miteinander verbunden sind. Bei einem Radius von 1,4 mm ist das Problem jedoch unbestritten zu erkennen (Abb. 4.27 rechts).

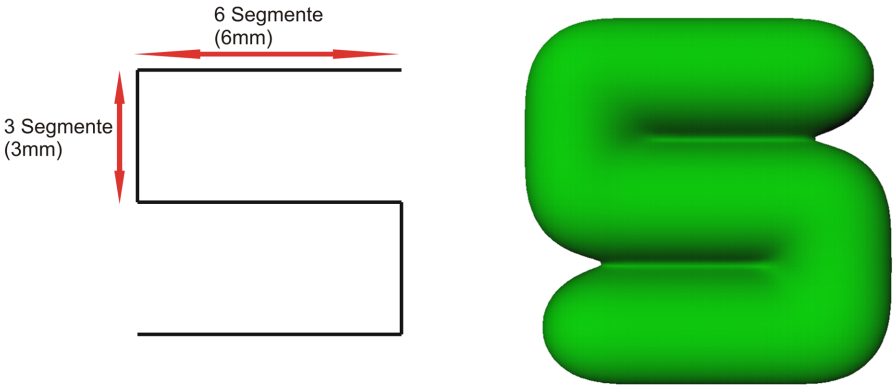


Abbildung 4.27: *Unwanted Blending* entlang der *Convolution Surface* eines S-förmigen Skeletts.

Um *Unwanted Blending* zu vermeiden, muss ein *blending* aller Segmente miteinander verhindert werden. Nach [AC02] wird daher die Feldfunktion an einem Punkt p basierend auf dem zu

p nächstliegenden Segment S_p und dessen direkten Nachbarn berechnet. Da die S-Form keine Verzweigungen besitzt, hat jedes Segment maximal zwei Nachbarn. Ferner liegt das Skelett als gerichteter Graph vor, so dass eine Bestimmung des Vorgängers $S_V p$ und des Nachfolgers $S_N p$ von S möglich ist. Zunächst wird die Projektion p' von p auf S_p berechnet. $u \in [0,1]$ sei der Parameter zu p' aus der Parametrisierung des Liniensegments S_p. Der Einfluss der beiden Nachbarsegmente wird mit Hilfe einer Funktion $\alpha(u)$ gewichtet. Diese sollte einen weichen Verlauf besitzen und langsam gegen 0 abfallen, um die Stetigkeit der Oberfläche zu gewährleisten [AC02]. Leider wird in [AC02] nicht erwähnt, welche Funktion hierfür genutzt wurde. Für weitere Tests wurde hier die Funktion $\alpha(u) = (1-u^2)^2$; verwendet. Die Feldfunktion an der Stelle p berechnet sich nun wie folgt:

$$f(S,p) = f(S_p,p) + \alpha(u) * f(S_V p) + (1-\alpha(u)) * f(S_N p) \tag{4.1}$$

Am Anfang des Segments S_p gilt $u = 0$ und daher $\alpha(u) = 1$. Das Vorgängersegment $S_V p$ entfaltet hier seinen vollen Einfluss, während der Nachfolger $S_N p$ keinen Einfluss besitzt. Am Ende des Segments ist $u = 1$ und die Umkehrung gilt. Dazwischen wird interpoliert. Die *Convolution Surface* des S-Skeletts berechnet nach Gleichung 4.1 ist in Abb. 4.28 (links) dargestellt. Leider sind erhebliche Artefakte entlang der Oberfläche zu beobachten (Abb.4.28 Mitte und rechts).

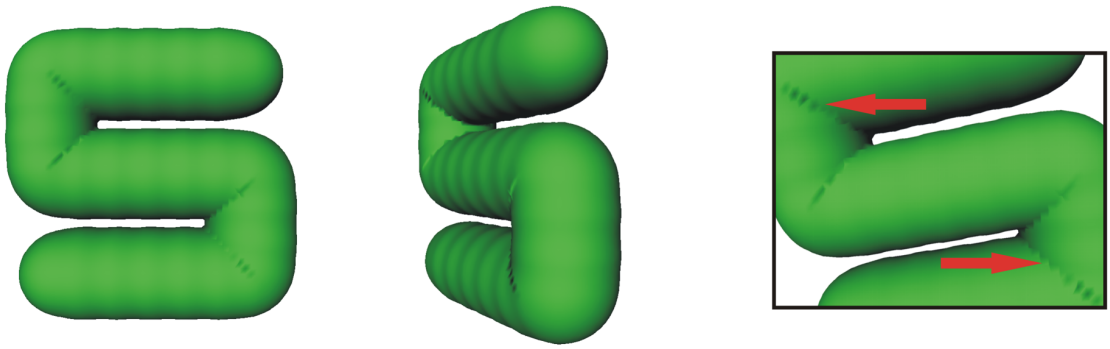


Abbildung 4.28: *Improved Local COnvolution* am Beispiel eines S-Skeletts (*links*). Eine Rotation (*Mitte*) und eine Nahansicht der *Convolution Surface* (*rechts*) zeigen ringförmige Artefakte bzw. Diskontinuitäten entlang der Oberfläche.

Die Ursache dieser Artefakte ist die Länge oder vielmehr die Kürze der Segmentprimitive. In [AC02] wird davon ausgegangen, dass das Skalarfeld um ein Primitiv S_p nur von denen der direkten Nachbarn beeinflusst wird. Sind die Segmente ausreichend lang in Bezug auf die Breite des verwendeten Filters, so ist diese Anforderung auch erfüllt. Um diese Behauptung zu untermauern wird nun die Länge der horizontalen Primitive des S-Skeletts von 1 mm auf 4 mm erhöht. Wie angenommen verschwinden die Artefakte tatsächlich (Abb. 4.29). Die Rücksprache mit einem der Autoren von [AC02], Alexis Angelidis, hat ergeben, dass die in seiner Arbeit verwendeten Liniensegmente erheblich länger waren als die des S-Skeletts. Im Falle der Gefäßvisualisierung ist jedoch mit sehr kurzen Segmenten zu rechnen. Hier beeinflusst eine größere Auswahl von Nachbarprimitiven das Segment S_p.

Die Definition eines lokalen zu betrachtenden Skelettbereichs, wie in [CH01] ursprünglich vorgeschlagen, scheint für die Gefäßvisualisierung eher geeignet. Experimente zeigen, dass die Betrachtung eines 6 Nachbarprimitive um S_p umfassenden Bereichs das *Unwanted Blending*-Problem

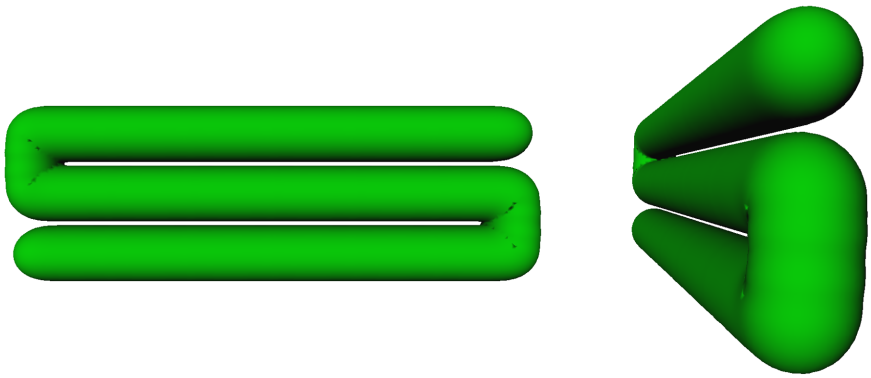


Abbildung 4.29: Eine Verlängerung der horizontalen Liniensegmente vermeidet die ringförmigen Artefakte aus Abb. 4.28.

für das S-Skelett vollständig löst (Abb. 4.30). Die Feldfunktion an der Stelle p berechnet sich dann aus der Summe von $f(S_p, p)$ und den Feldfunktionswerten für die Segmente des Nachbarbereichs. Für eine Generalisierung im Hinblick auf „reale“ Gefäßmodelle sind jedoch genauere Untersuchungen notwendig. So sollte der Einfluss der Liniensegmentlänge auf die Größe des zu wählenden Skelettbereichs evaluiert werden. Weiterhin wäre eine genaue Abschätzung der Skalarfeldausdehnung unter Verwendung eines bestimmten Filters hilfreich. Da *Unwanted Blending* an keinem während dieser Arbeit visualisierten Gefäßmodell beobachtet werden konnte, sind jedoch keine weiteren Anstrengungen in diese Richtung unternommen worden. Im Bedarfsfall ist eine Implementierung allerdings ohne weiteres möglich, da die Gefäßtopologie aus der Gefäßanalyse bekannt ist. Somit kann der Nachbarbereich eines Segments einfach bestimmt werden kann.

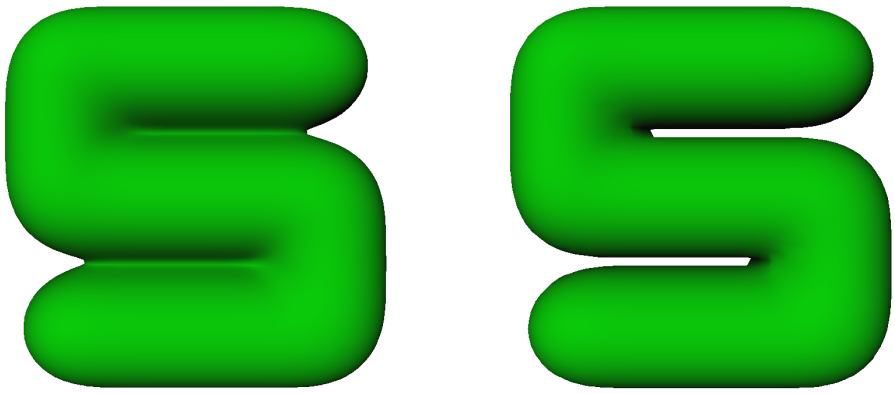


Abbildung 4.30: *Local Convolution*. Die Definition eines Skelettbereichs von jeweils sechs Nachbarprimitiven beseitigt das *Unwanted Blending*-Problem.

4.3 Effektive Polygonisierung von Convolution Surfaces

Die ersten Visualisierungstests in Abschnitt 4.1 haben gezeigt, dass die Erstellung eines polygonalen Modells der Gefäßstruktur beträchtliche Rechenzeit in Anspruch nimmt. Die gemessenen Zeiten liegen jenseits einer für den klinischen Betrieb akzeptablen Grenze. Es besteht daher ein dringender Bedarf an Optimierungen welche den Polygonisierungsprozeß beschleunigen. In den folgenden beiden Abschnitten wird eine Strategie zur Performanzsteigerung entwickelt bzw. die Hilfsmittel zur Umsetzung dieser Strategie werden diskutiert.

4.3.1 Entwurf einer Datenstruktur für die effiziente Polygonisierung

Die Polygonisierung der impliziten Oberfläche erfordert es, die zugrundeliegende implizite Funktion an jedem Punkt des Raumes berechnen zu können. Im Rahmen des *brute-force*-Ansatzes aus Abschnitt 4.1 wurde hierfür das gesamte Gefäßskelett an jedem Punkt p betrachtet. Dabei wurde die Feldfunktion jedes Liniensegments des Skeletts bestimmt und die Summe der resultierenden Skalarwerte gebildet. Diese Vorgehensweise generiert zwar die gewünschte Gefäßoberfläche, ist allerdings sehr ineffizient.

Die Faltung eines Liniensegments mit einem Gauß-Filter resultiert aufgrund des unendlichen Trägerintervalls des Filters in einem theoretisch unbeschränkten Skalarfeld um das Primitiv. Das Feld nimmt jedoch ab einer bestimmten Entfernung von dem Liniensegment vernachlässigbar kleine Werte an. Daher besitzen weit entfernte Skelettprimitive keinen signifikanten Einfluss an der Stelle p und die Berechnung ihrer Feldfunktion steigert unnötig die Rechenzeit. Infolgedessen ist es essenziell für die Performanzsteigerung, an jedem Punkt des Raumes den relevanten Skelettabschnitt[1] bestimmen zu können. Besonders die Polygonisierungstechniken *Spatial Sampling* und *Surface Tracking* aus Abschnitt 3.4.2 würden von dieser Information profitieren. Die Überführung der impliziten Beschreibung des Objektes in ein geometrisches Modell erfolgt hier im Gegensatz zu *Surface Fitting* nach [DTG96] nicht elementweise, d.h. Primitiv für Primitiv, entsprechend der Skelett-Topologie. Während des Polygonisierungsprozesses ist somit nicht unmittelbar ersichtlich, welcher Skelettabschnitt an einem bestimmten Punkt gerade relevant ist.

Das Prinzip des in dieser Arbeit genutzten *Implicit Polygonizers* [Blo94] verdeutlicht dieses Problem. Entsprechend eines Fortführungsschemas nach [WMW86] wird hier die implizite Oberfläche komplett durch ein Würfelraster eingefasst und innerhalb der einzelnen Würfel polygonisiert. Während der Abtastung der Oberfläche werden dabei die Grenzen einzelner Skelettprimitive überschritten, ohne den korrespondierenden Oberflächenanteil vorher vollständig erfasst und polygonisiert zu haben (Abb. 4.31). Aus dem Verfahren ist nicht abzuleiten, welcher Teil der Gefäßstruktur gerade polygonisiert wird. Somit ist auch nicht bekannt, welche Skelettprimitive bei der Berechnung der Feldfunktion an einem Punkt p berücksichtigt werden sollten bzw. vernachlässigt werden können.

Im Folgenden soll daher eine Datenstruktur entworfen werden, welche während der Polygonisierung Anfragen hinsichtlich des an einem Punkt p relevanten Skelettabschnitts beantwortet. Diese Datenstruktur kann vollständig in einem Vorverarbeitungsschritt initialisiert und aufgebaut

[1] Hier und im Folgenden ist mit Skelettabschnitt **kein** zusammenhängendes Gebiet des Skeletts gemeint, sondern eine Menge einzelner Skelettprimitive (Liniensegmente)

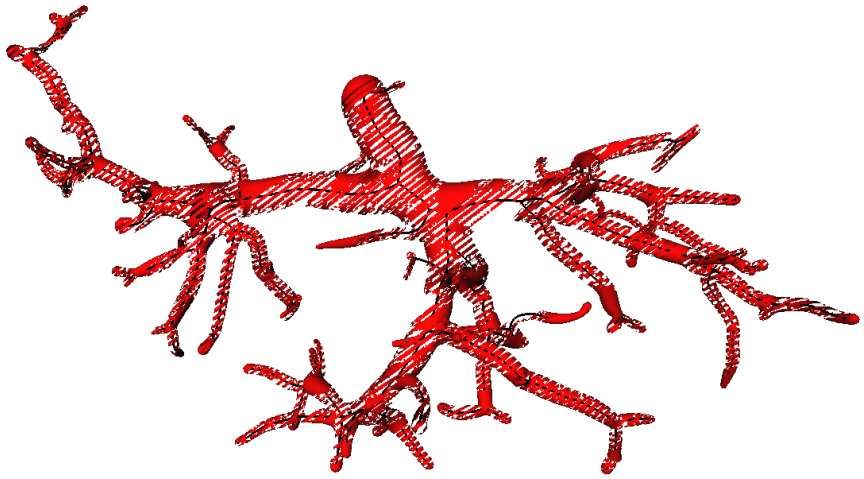


Abbildung 4.31: Die Polygonisierung eines Gefäßbaums mit Hilfe des *Implicit Polygonizers* wurde unterbrochen und das Zwischenresultat dargestellt. Hier ist deutlich zu erkennen, dass die Gefäßstruktur nicht elementweise, entsprechend der Skelett-Topologie bearbeitet wird. Stattdessen verläuft der Pfad des *Polygonizers* quer über die gesamte Oberfläche.

werden, da das Gefäßmodell statisch ist. Die wesentliche Idee besteht in einer Partitionierung des Raumes um das Gefäßmodell. Für jede Einheit der Partitionierung wird der dort relevante Skelettabschnitt bestimmt und in der Datenstruktur abgespeichert. Während der Polygonisierung kann dann bestimmt werden, innerhalb welcher Einheit sich der Punkt p befindet und die Berechnung der Feldfunktion kann auf den korrespondierenden Skelettabschnitt beschränkt werden. Die hier vorgeschlagene Unterteilung entspricht der inhärenten Voxelstruktur des Originaldatensatzes. Sie wird daher im folgenden als *Voxelgitter* bezeichnet. Das Voxelgitter ist ursprünglich durch das dreidimensionale Voxelkoordinatensystem (VKS) in welchem der Datensatz definiert ist beschrieben. Mit Hilfe der Transformationsmatrix aus Anhang A.1 können die Voxelmittelpunkte (bzw. Gittermittelpunkte) jedoch in das Weltkoordinatensystem (WKS) überführt werden. Da die Polygonisierung innerhalb dieses Koordinatensystems erfolgt, ist die wechselseitige Bestimmung der Unterteilung des Raumes notwendig. Die Rücktransformation von Punkten in das VKS wird durch die Inverse der Transformationsmatrix möglich.

Ein durch die Bildaufnahme gewonnener Volumendatensatz ist im VKS definiert. Die Mittelpunkte der Voxel des Datensatzes sind hier an nicht-negativen, ganzzahligen Koordinaten lokalisiert. Jedes Voxel besitzt eine isotrope Ausdehnung gleich 1. Der Ursprung des Koordinatensystems befindet sich im Zentrum des Voxels mit den Koordinaten $(0, 0, 0)$. Das Gefäßskelett ist zunächst im VKS bestimmt. Für die Visualisierung der Gefäßstruktur ist jedoch die Überführung in das WKS notwendig, um sowohl ihre „realen“ Ausmaße als auch die räumliche Beziehung zu weiteren Objekten ausdrücken zu können. Die Voxelmittelpunkte sind im WKS an ihrer eigentlichen Position innerhalb der „realen“ Welt des Objektes lokalisiert. Ihre Koordinaten sind

selten ganzzahlig und die Entfernung zwischen den Voxelmittelpunkten variiert zumeist in x-,y- und z-Richtung. Die Voxel besitzen somit eine anisotrope Ausdehnung.

Für jedes Voxel soll der relevante Skelettabschnitt bestimmt und in einer Datenstruktur gespeichert werden. Umgekehrt, für jedes Liniensegment muss das aus der Faltung resultierende Skalarfeld voxelweise erfasst werden. Für die hierbei akkumulierte Voxelmenge wird das aktuelle Liniensegment vermerkt. Somit besteht die Notwendigkeit einer zweidimensionalen Datenstruktur. Die erste Dimension entspricht den Voxeln des Voxelgitters. Da die Gefäßstruktur und somit auch der signifikante Teil des umgebenden Skalarfeldes nur einen Teil dieses Gitters abdeckt, wäre es ineffizient in einer statischen Datenstruktur für jeden Voxel einen Eintrag vorzusehen. Stattdessen sollte diese Dimension dynamisch gestaltet werden, und nur Voxel enthalten, welche tatsächlich durch einen Teil des Skalarfeldes der Gefäßstruktur beeinflusst werden. Die zweite Dimension beschreibt die Liniensegmente, deren Skalarfeld den jeweiligen Voxel einschließt. Da bei der Initialisierung der Datenstruktur schlecht abzuschätzen ist, wieviele Liniensegmente einen Voxel beeinflussen und diese Zahl von Voxel zu Voxel variiert, sollte auch diese Dimension der Datenstruktur dynamisch gestaltet werden. Die Datenstruktur ist in Abb. 4.32 skizziert.

Sowohl die Voxel, als auch die Liniensegmente sollten anhand einer speichereffizienten, eindeutigen Identifikation (ID) genau bestimmt werden können. Aus den drei Voxelkoordinaten x, y, z und der Dimension des Datensatzes $(x_{dim}, y_{dim}, z_{dim})$ lässt sich für jedes Voxel leicht eine ID wie folgt berechnen: $Vox\_ID = z * (y_{dim} * x_{dim}) + y * (x_{dim}) + x$. Ein Liniensegment lässt sich mit Hilfe der Information des Gefäßgraphen aus der Gefäßanalyse (siehe Anhang A.1) identifizieren. Danach kann ein Tupel der Form $< e, v >$ konstruiert werden, wobei e die zugehörige Kantennummer und v die Position des Startvoxels in der Voxelliste dieser Kante beschreibt. Die ID des Voxels dient als Schlüssel und ermöglicht während der Polygonisierung einen Zugriff auf den diesen Teil des Raumes beeinflussenden Skelettabschnitt. Hierzu muss lediglich der Punkt p aus dem WKS in das VKS transformiert und die ID des getroffenen Voxels berechnet werden.

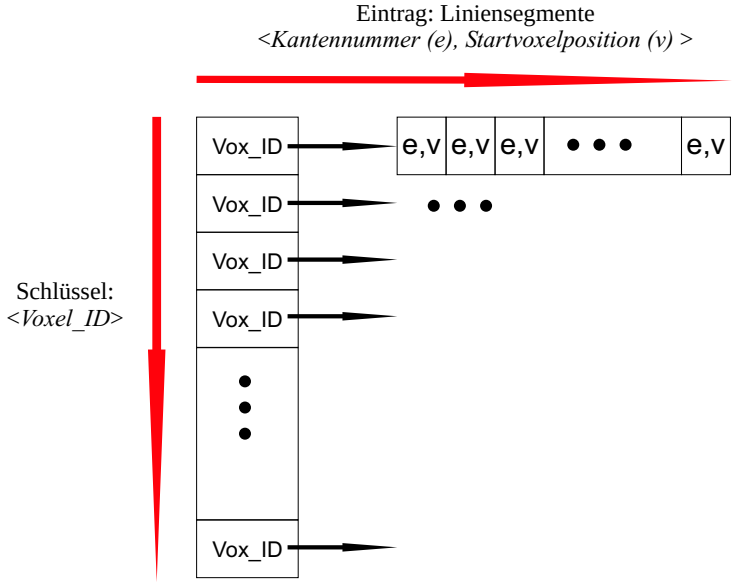


Abbildung 4.32: Datenstruktur zur effizienten Polygonisierung. Jedes hier gespeicherte Voxel wird durch einen Teil des gesamten Skalarfeldes um die Gefäßstruktur beeinflusst. Dieser Teil ist durch den korrespondierenden Eintrag in Liniensegmenten ausgedrückt.

Vor einer Beschreibung von Methoden zur Auffüllung der Datenstruktur scheint eine Abgrenzung der Bezeichnungen Gefäßdurchmesser und Skalarfelddurchmesser sinnvoll. Zur Illustration siehe Abb. 4.33. Der Gefäßdurchmesser ist durch einen gegebenen Durchmesserverlauf exakt beschrieben. Er kann mit Hilfe eines von der Filterfunktion abhängigen Isowertes korrekt abgebildet werden. Während der Polygonisierung werden hierzu Punkte mit einem Skalarwert gleich dem Isowert miteinander verbunden. Der Isowert beschreibt **eine** mögliche Oberfläche durch das Skalarfeld um die Gefäßstruktur. Der Skalarfelddurchmesser reicht über den Isowert und somit über den Gefäßdurchmesser hinaus. Im Falle des modifizierten Gauß-Filters aus Abschnitt 4.2.1 kann der Gefäßdurchmesser durch den Isowert 1/32 abgebildet werden. Der Teil des Skalarfeldes $< 1/32$ beschreibt den Abschnitt außerhalb des Gefäßes. Für den Gauß-Filter ist die Ausdehnung aufgrund des unendlichen Trägerintervalls sogar unbegrenzt. Allerdings nehmen die Skalarwerte ab einer bestimmte Entfernung von einem Skelettprimitiv vernachlässigbar kleine Werte an. Im Falle eines isolierten Liniensegments ist der überstehende Anteil nicht von Bedeutung. An einer einfachen Verbindung oder Verzweigung entlang des Gefäßskeletts führt die Vernachlässigung dieses Anteils jedoch zu Artefakten entlang der Gefäßoberfläche. Wie im folgenden Abschnitt noch gezeigt wird, ist dieser Anteil im Zuge der Formung weicher Verbindungen zwischen Gefäßteilen mittels *blending* dringend mit einzubeziehen.

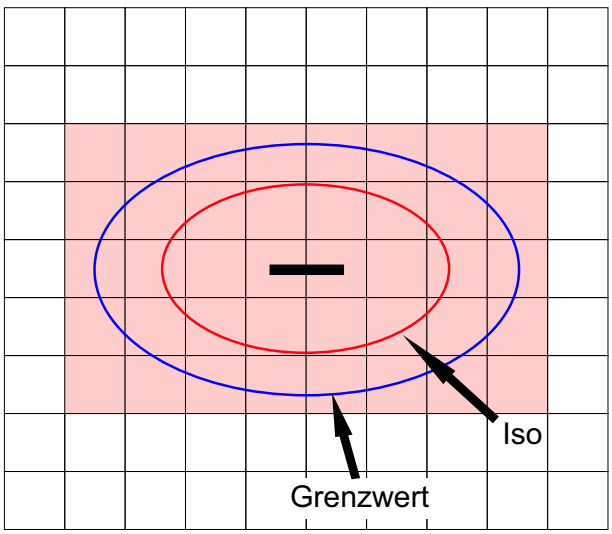


Abbildung 4.33: Voxelweise Erfassung des signifikanten Skalarfeldbereichs um ein Liniensegment. Während die innere ellipsoide Form den Umriss der zugehörigen *Convolution Surface* mit gegebenem Durchmesser beschreibt, markiert der äußere Ellipsoid die Grenze des signifikanten Skalarfeldbereichs. Die Voxelmenge welche diesen Bereich vollständig einfasst ist farblich hervorgehoben.

Die entworfene Datenstruktur kann in einem Vorverarbeitungsschritt vollständig initialisiert und aufgebaut werden. Dazu muss um jedes Skelettprimitiv eine Voxelmenge bestimmt werden, welche den signifikanten Teil des korrespondierenden Skalarfeldes, **nicht** den Gefäßdurchmesser, komplett beinhaltet. Für diese Aufgabe wurden hier zwei verschiedene Hüllkörper (*bounding volumes*) entworfen, welche im nächsten Abschnitt vorgestellt werden. Um eine adäquate Größe der Hüllkörper festzulegen, wird weiterhin die Bezeichnung eines signifikanten Skalarfeldbereichs konkretisiert.

4.3.2 Hüllkörper

Hüllkörper umfassen ein kompliziert strukturiertes Objekt vollständig mit einer einfacheren Form, welche von der originalen Objektform beliebig stark abstrahiert. Häufig verwendete einfache geometrische Formen sind: Würfel, Quader, Kugel, Zylinder, ... Im Kontext von *Convolution Surfaces* wurden Hüllkörper bereits für die Darstellung von Objekten mittels *Ray Tracing* verwendet [She98b], [MS98]. Die notwendigen Schnittpunkttests eines Strahls mit den Objekten einer Szene werden aus Performanzgründen oft zuerst mit deren Hüllkörper durchgeführt. Wird der Hüllkörper des Objektes nicht geschnitten, so wird aus das korrespondierende Objekt nicht von dem Strahl getroffen und kann daher vernachlässigt werden. Im Rahmen der Gefäßvisualisierung und der Polygonisierung impliziter Oberflächen sollen Hüllkörper hier einen ganz anderen Zweck erfüllen. Sie dienen der voxelweisen Abschätzung des signifikanten Skalarfeldbereichs um ein Primitiv des Gefäßskeletts.

4.3.2.1 Form

Das aus der Faltung eines Liniensegments mit einem Filter resultierenden Skalarfeld ist zylindrisch. Im Falle der Gefäßvisualisierung wird jedoch ein variierender Durchmesser entlang eines Liniensegments abgebildet. Daher ist meist eine konische Form des Skalarfeldes zu erwarten. Ein kegelstumpfförmiger Hüllkörper scheint hier geeignet.

In einem Vorverarbeitungsschritt soll für jedes Liniensegment des Gefäßskeletts der Skalarfeldbereich in Voxeln abgeschätzt werden. Dies erfordert eine Bestimmung aller Voxel des Voxelgitters welche entweder vollständig oder teilweise in diesem Einflussbereich liegen. Dabei ist die anisotrope Voxelausdehnung zu beachten. Angenommen ein konischer Hüllkörper ist bereits für das Liniensegment konstruiert. Zu klären bleibt, welche Voxel hinsichtlich ihrer Lage zu dem Hüllkörper getestet werden sollen. Aus Effizienzgründen scheidet eine Betrachtung aller Voxel des Voxelgitters aus. Stattdessen wird ein weiterer Hüllkörper eingeführt. So liefert die Bestimmung eines voxelbasierten, achsenparallelen Hüllkörpers (VAH) um das Liniensegment eine Startauswahl an Voxeln (Abb. 4.34 links). Diese Auswahl kann dann mit Hilfe eines adäquateren Hüllkörpers verfeinert werden. Besonders wirkungsvoll ist so eine Ausdünnung entlang diagonal im Voxelgitter ausgerichteter Liniensegmente. Orthogonal zu dem Liniensegment werden hier unnötig viele Voxel umschlossen (Abb. 4.34 links). Ein kegelstumpfförmiger Hüllkörper ist hierfür zwar geeignet, kann jedoch im Hinblick auf eine Verringerung der Rechenzeit weiter vereinfacht werden. Da der Gefäßradius entlang eines Skelettprimitivs anatomisch bedingt nur schwach variiert, ist eine zylindrische Form hinreichend. Die Bestimmung, ob ein Voxel vollständig oder teilweise innerhalb dieses Zylinders lokalisiert ist, kann mit Hilfe der Eckpunkte des Voxels erfolgen. Ein implizit beschriebener Zylinder ist hierfür hervorragend geeignet, da die implizite Beschreibung sehr einfache *point classification* Tests basierend auf dem Vorzeichen der impliziten Funktion gestattet. Die in dieser Arbeit verwendete Beschreibung inklusive dem *point classification* Test ist in [She94] vorgestellt worden. Der implizit beschriebene zylindrische Hüllkörper wird im Folgenden durch iZH abgekürzt.

Da der implizit beschriebene Zylinder unendlich lang ist, können Voxel in Richtung des Liniensegments, unabhängig von ihrer Entfernung zu den Segmentendpunkten, als innenliegend klassifiziert werden (Abb. 4.34 Mitte). Der signifikante Skalarfeldbereich ist jedoch auch in dieser Richtung begrenzt (Abb. 4.33). Obwohl die VAH den Voxelbereich schon eingrenzt, werden

an den Enden diagonal im Voxelgitter ausgerichteter Liniensegmente unnötig viele Voxel ausgewählt. Daher wird zusätzlich für jede Endpunkt des Liniensegments orthogonal zu diesem eine Ebene definiert. Diese Ebenen überführen den infiniten iZH in einen finiten zylindrischen Hüllkörper (ZH) (Abb. 4.34 rechts). Der Abstand einer Ebene von dem korrespondierenden Endpunkt wird entsprechend dem Durchmesser des signifikanten Skalarfeldbereichs gewählt.

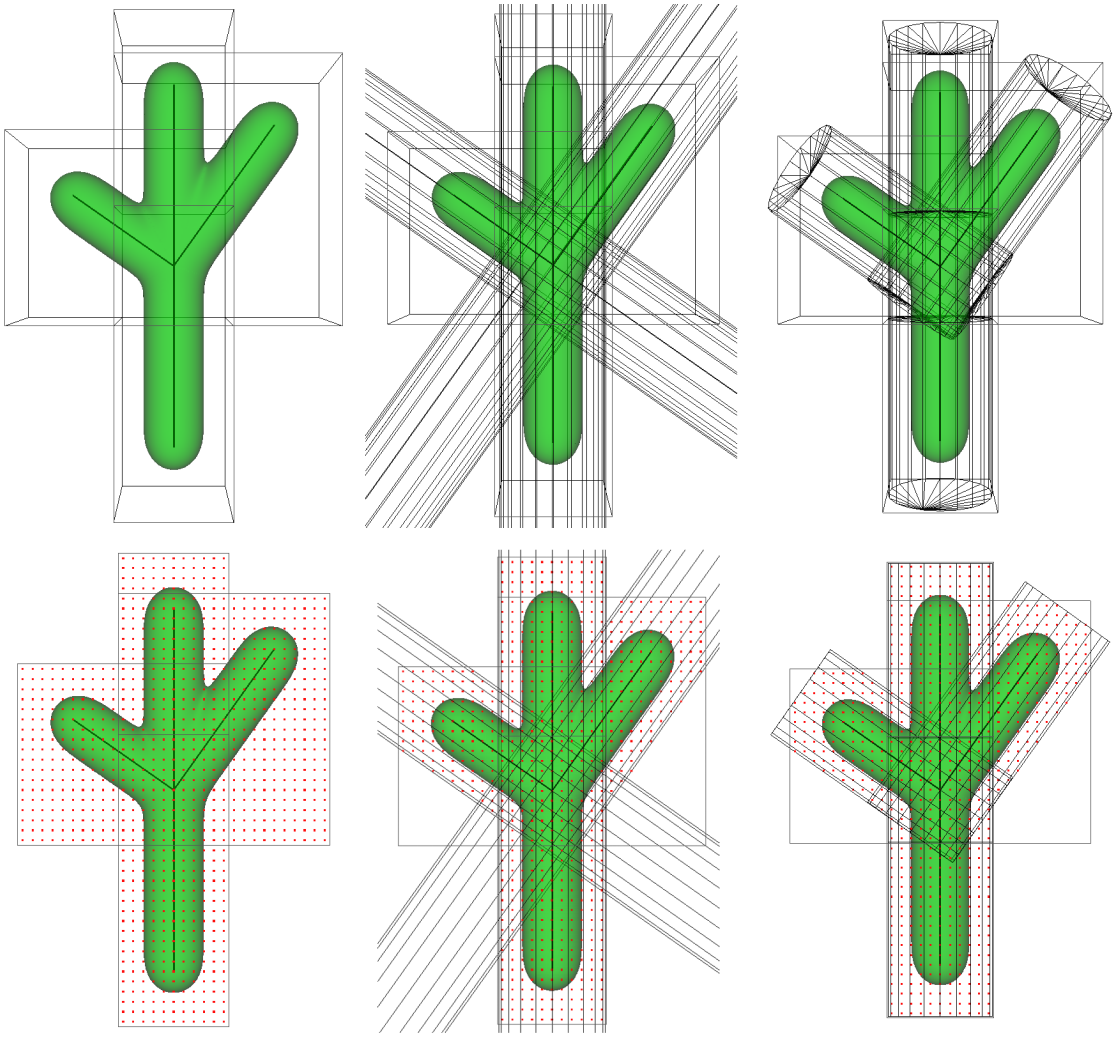


Abbildung 4.34: Hüllkörper an einer künstlichen Trifurkation. Um eine bessere Übersichtlichkeit zu gewährleisten, wurden die Hüllkörper pro Kante (sonst pro Liniensegment) dargestellt. Dies ist hier möglich, da alle Voxel einer Kante entlang einer geraden Linie angeordnet sind. *Links*: VAH, *Mitte*: iZH, *Rechts:* ZH. Die Punkte in der unteren Zeile der Darstellung kennzeichnen die jeweils in dem Hüllkörper enthaltenen Voxelmittelpunkte. Die Abbildungen dieser Zeile sind zur besseren Anschaulichkeit mittels Parallelprojektion visualisiert.

Bei den *point classification*-Tests für jeden Voxel müssen nun zusätzlich die beiden Ebenen berücksichtigt werden. Ein Punkt liegt innerhalb des ZH, wenn er sowohl innerhalb des iZH, als auch zwischen den beiden Ebenen liegt. Die Lage eines Punktes P bezüglich dieser Ebenen kann mit Hilfe der Berechnung des Skalarprodukts aus zwei Vektoren bestimmt werden. Gegeben sei die Achse Z des ZH mit den beiden Endpunkten z_0 und z_1. Diese Achse beinhaltet das korrespondierende Liniensegment. Die Lage des Punktes P relativ zu der Ebene E_{z_0} durch z_0

lässt sich nun wie folgt bestimmen. P liegt:

$$\begin{aligned} links\ von\ E_{z_0} &\Leftrightarrow \overrightarrow{z_0 z_1} \bullet \overrightarrow{z_0 P} < 0, \\ auf\ E_{z_0} &\Leftrightarrow \overrightarrow{z_0 z_1} \bullet \overrightarrow{z_0 P} = 0, \\ rechts\ von\ E_{z_0} &\Leftrightarrow \overrightarrow{z_0 z_1} \bullet \overrightarrow{z_0 P} < 0. \end{aligned}$$

Die Betrachtung der Ebene E_{z_1} erfolgt analog. Liegt der Punkt P rechts von oder auf der Ebene E_{z_0}, links von oder auf der Ebene E_{z_1} und innerhalb des iZH, so liegt er innerhalb des ZH.

4.3.2.2 Größe

Gegeben sei ein Liniensegment L mit den beiden Endpunkten e_1 und e_2, sowie den dort definierten Radien $r(e_1)$ und $r(e_2)$. Die Größe des VAH und der zylinderförmigen Hüllkörper basiert auf dem maximalen Durchmesser des signifikanten Skalarfeldbereichs um ein Liniensegment. Dieser ist wiederum durch den maximalen Radius $r_{max} = \max(r(e_1), r(e_2))$ der *Convolution Surface* plus einem gewissen Offset bestimmt. Der Offset dient der Erfassung des Skalarfeldbereichs, welcher über den Isowert, d.h. die *Convolution Surface* hinausragt. Die Größe des Offsets ist entscheidend für die visuelle Qualität der konstruierten Gefäßoberfläche. Die Verwendung eines sehr geringen Werts resultiert in einer kleinen Voxelumgebung und spart daher Rechenzeit und Speicherplatz. Jedoch werden so Teile des signifikanten Skalarfeldbereichs eventuell abgeschnitten, was zu Artefakten entlang der Oberfläche, besonders an Verzweigungen, führt (Abb. 4.35). Gerade hier sorgt jedoch die Überlappung der Skalarfelder benachbarter Skelettprimitive für die Entstehung weicher Verbindungen zwischen einzelnen Gefäßsegmenten. Die Größe des Offsets ist von der verwendeten Filterfunktion abhängig.

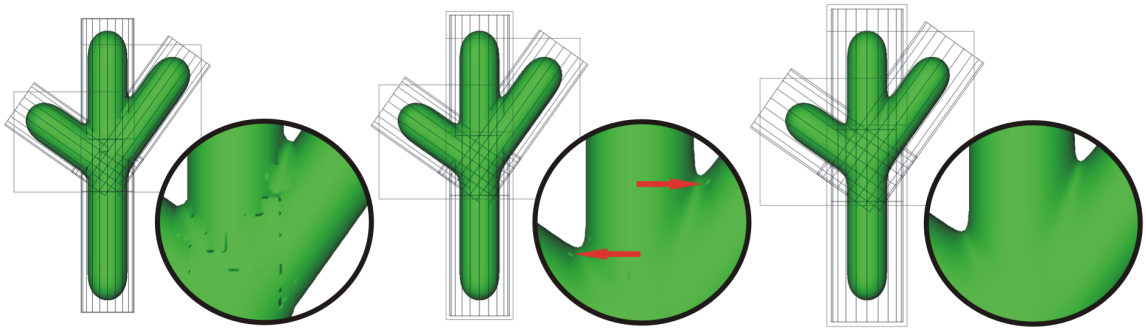


Abbildung 4.35: Hüllkörper unterschiedlicher Größe. Je weiter das Skalarfeld umschlossen wird (von links nach rechts) desto weniger Artefakte sind entlang der Oberfläche zu beobachten. Ab einer bestimmten Entfernung sind die Skalarwerte vernachlässigbar klein und die Oberfläche ist artefaktfrei (*rechts*).

Im Falle des Gauß-Filters wird der finale Skalarwert an einem Punkt p aus der Multiplikation von Integrations- und Distanzfilter berechnet (Gleichung 3.17). Die Größe des Integrationsfilters variiert zwischen 0 und 1. Die Größe des Distanzfilters ist von dem abzubildenden Durchmesser und der Distanz zwischen p und dem Liniensegment L abhängig (Gleichung 3.19). Ein geeigneter Offset kann mit Hilfe des Distanzfilters bestimmt werden. So konvergiert dieser bei zunehmender Entfernung von p zu L gegen 0 und das Ergebnis der Multiplikation mit dem Integrationsfilter

wird vernachlässigbar klein. Zu klären bleibt, ab welcher Distanz dies der Fall ist und ab wann ein Skalarwert als vernachlässigbar klein bezeichnet werden kann.

Der in Gleichung 3.19 enthaltene Wert $d_{pp'}$ beschreibt die euklidische Distanz von einem Punkt p zu dessen Projektion p' auf dem Liniensegment L. Für einen Punkt p, welcher auf halber Länge von L direkt auf der *Convolution Surface* liegt, entspricht die Distanz $d_{pp'}$ dem abzubildenden Radius $r = r(p')$. Für den modifizierten Gauß-Filter aus Abschnitt 4.2 mit *width coefficient* $\omega = 5 \ln 2$ gilt somit nach Gleichung. 3.19: $h(r) = e^{\frac{-\omega \ r^2}{r^2}} = 1/32 = 0.03125$. Bei einer Distanz von r besitzt das Skalarfeld somit den Wert 0.03125, welcher dem Isowert entspricht mit dessen Hilfe die *Convolution Surface* konstruiert wird. Obwohl dieser Wert bereits sehr klein ist, macht sich ein zu zeitiges Abschneiden des Feldes hinter dieser Grenze in der Darstellung störend bemerkbar (Abb. 4.35). Die Visualisierung und sorgfältige Betrachtung zahlreicher Testmodelle hat gezeigt, dass ein Skalarwert als vernachlässigbar klein angesehen werden kann, wenn mindestens die ersten 3 Nachkommastellen gleich 0 sind. Ab welcher Distanz ist dies nun der Fall ? Da der Durchmesser des Skalarfeldes maßgeblich von dem abzubildenden Radius r abhängt, ist es sinnvoll den Offset durch die Multiplikation von r mit einem Faktor auszudrücken:

$$\begin{aligned} h(r*1.1) &= e^{\frac{-\omega \ (r*1.1)^2}{r^2}} \approx 0.0151 \\ h(r*1.2) &= e^{\frac{-\omega \ (r*1.2)^2}{r^2}} \approx 0.0068 \\ ... &= ... \\ h(r*1.5) &= e^{\frac{-\omega \ (r*1.5)^2}{r^2}} \approx 0.0004 \end{aligned}$$

Ab einem Faktor von 1.5 sind mindestens die ersten 3 Nachkommastellen gleich 0. Multipliziert mit dem Integrationsfilter dessen maximaler Wert gleich 1 ist, führt dies zu einem vernachlässigbar kleinen Skalarwert. Der Radius des signifikanten Skalarfeldbereichs um ein Liniensegment L ist somit durch den Wert $r_{max} * 1.5$ gegeben.

4.3.2.3 Anwendung

Für die Konstruktion der VAH muss nun die Voxelumgebung bestimmt werden, welche diese Ausdehnung in x-,y- und z-Richtung komplett einschließt. Dabei ist die unterschiedliche Voxelausdehnung in allen drei Richtungen zu beachten. Die Überführung des Gefäßskeletts aus dem VKS in das WKS erfordert eine zusätzliche Anpassung des VAH. Während im VKS die Voxelmittelpunkte ganzzahlig indiziert werden, sind es im WKS die Gitterpunkte. Den Ursprung dieses Koordinatensystems $(0, 0, 0)$ bildet der Schnittpunkt der drei Koordinatenachsen. Daher ist nach der Transformation des Gefäßskeletts eine zusätzliche Translation notwendig. Die Voxelauflösung in x-, y- und z-Richtung sei $(x_{vox}, y_{vox}, z_{vox})$. Der Translationsvektor für jeden Punkt des Gefäßskeletts ist dann gegeben durch $(-x_{vox}/2, -y_{vox}/2, -z_{vox}/2)$. Um diese Verschiebung bei der Startvoxelauswahl zu berücksichtigen, wird der VAH um jeweils einen Voxel in negativer x-, y- und z-Richtung erweitert (Abb. 4.36).

Die Startauswahl der in dem VAH enthaltenen Voxel soll nun im nächsten Schritt mit Hilfe eines ZH ausgedünnt werden. Der Radius des iZH wird hierzu unter Berücksichtigung des Offsets auf $r_{max} * 1.5$ festgelegt. Die beiden Ebenen werden jeweils in diesem Abstand von den Endpunkten des Liniensegments platziert. Bevor die Startvoxelauswahl eingeschränkt werden kann, muss das Gefäßskelett in das WKS überführt werden, um die reale Ausdehnung der Gefäßstruktur

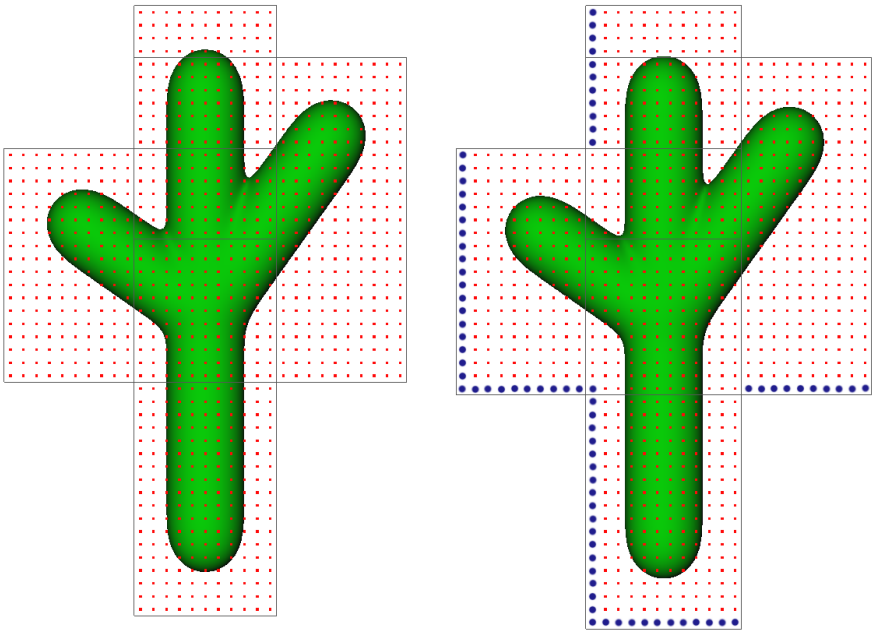


Abbildung 4.36: Trifurkation und initiale Startvoxelauswahl mittels VAH(*links*). Die unterschiedliche Indizierung in Voxel- und Weltkoordinatensystem erfordert eine Translation der Gefäßstruktur um jeweils die halbe Voxelauflösung in negativer x-, y- und z-Richtung(*rechts*). Um diese Verschiebung zu kompensieren, wird die Startvoxelauswahl um einen Voxel in jeder der drei Richtungen erweitert. Dies ist hier durch Kreise gekennzeichnet.

erfassen zu können. Für die *point classification*-Tests ist daher ebenfalls eine Transformation des Voxelgitters notwendig. Natürlich könnte nun jeder der acht Eckpunkte eines Voxels hinsichtlich seiner Lage zu dem ZH getestet werden. Um die Rechenzeit weiter zu senken wird hier jedoch nur der Voxelmittelpunkt betrachtet. Diese Einschränkung erfordert allerdings eine Adaption der Größe des ZH. So kann es passieren, dass ein Voxel zwar teilweise in dem Hüllkörper liegt, der Voxelmittelpunkt sich jedoch außerhalb befindet. Um dies zu vermeiden, wird die größte mögliche Ausdehnung eines Voxels (A_{max}) bestimmt. Die Hälfte des resultierenden Wertes wird dann zu dem Radius des ZH addiert: $r_{max} * 1.5 + A_{max}/2$. Auf diese Wiese wird ein Voxel selbst dann erfasst, wenn die Grenze des Skalarfeldbereichs durch einen seiner Eckpunkte verläuft. Die Hüllkörper in Abb. 4.35 sind basierend auf den bisherigen Überlegungen mit den Faktoren $1.1, 1.3$ und 1.5 (von links nach rechts) erzeugt worden.

Für jedes Voxel, welches den finalen Hüllkörpertest besteht, wird dessen ID aus den ganzzahligen Koordinaten des VKS berechnet. Sollte diese ID schon in der Datenstruktur existieren, weil es ebenfalls in dem signifikanten Skalarfeldbereich eines früher betrachteten Gefäßsegments liegt, wird der korrespondierende Eintrag einfach um das aktuelle Liniensegment erweitert. Ist die ID noch nicht vorhanden, wird ein neuer Eintrag in der Datenstruktur angelegt.

4.3.2.4 Anmerkung zu anderen Filterfunktionen

Wie im vorigen Abschnitt bereits erwähnt, ist die Größe des Offsets und somit die Größe der Hüllkörper von der verwendeten Filterfunktion abhängig. Die obere Grenze des vernachlässigbaren Skalarfeldbereichs um ein Primitiv wurde hier auf Werte mit drei Nachkommastellen gleich 0 festgelegt. Interessant ist nun, wie die Größe der Hüllkörper unter der Verwendung anderer Filterfunktionen gewählt werden muss. Betrachtet werden sollen hier der originale Gauß-Filter aus [Blo95b], die inverse kubische Funktion [CH01] und die Filterfunktion nach [HAC03] (siehe Abschnitt 3.3.2).

Im vorhergehenden Abschnitt wurde der Durchmesser des signifikanten Skalarfeldbereichs für den modifizierten Gauß-Filter durch die Multiplikation des abzubildenden Radius r mit einem Faktor ausgedrückt (Glg. 4.2). Ab einem Faktor der Größe 1.5 sind hier die Werte des Skalarfeldes vernachlässigbar klein. Analog zu dieser Vorgehensweise kann der Faktor für die originale Filterfunktion aus [Blo95b] bestimmt werden. Ab einer Entfernung von $r*3.2$ nimmt das Skalarfeld hier vernachlässigbar kleine Werte an. Dieser erhöhte Faktor ist begründet durch den langsameren Abfall der originalen Filterfunktion gegen 0, verglichen mit dem modifizierten Filter (Abb. 4.10). Die Entfernung vom Filterzentrum ab welcher die Funktionswerte vernachlässigbar klein sind ist hier größer. Der signifikante Skalarfeldbereich verläuft weiträumiger um das Primitiv (Abb. 4.9). Der höhere Faktor resultiert in einem mehr als 4 mal so großen Hüllkörper, was sich sowohl in der Rechenzeit, als auch bei dem Speicherbedarf negativ bemerkbar macht.

Sowohl die inverse kubische Filterfunktion (Abb. 3.26), als auch die Abwandlung der quadratischen Funktion (Abb. 3.28) fallen noch langsamer gegen 0 ab als beide Versionen des Gauß-Filters. Daher müssen hier noch größere Hüllkörper konstruiert werden, um den signifikanten Skalarfeldbereich komplett einzuschließen und Artefakte entlang der Oberfläche zu vermeiden. Adäquate Hüllkörper umschließen hier eine sehr große Voxelmenge und resultieren daher in einer geringeren Performanzsteigerung. Der Speicherbedarf steigt erheblich, da insgesamt mehr Voxel in der Datenstruktur gespeichert werden müssen und zudem mehr Liniensegmente pro Voxel einen Einfluss haben. Die Applikation beider Filterfunktionen ist daher mit der hier vorgestellten

Hüllkörper-Methodik nicht vereinbar.

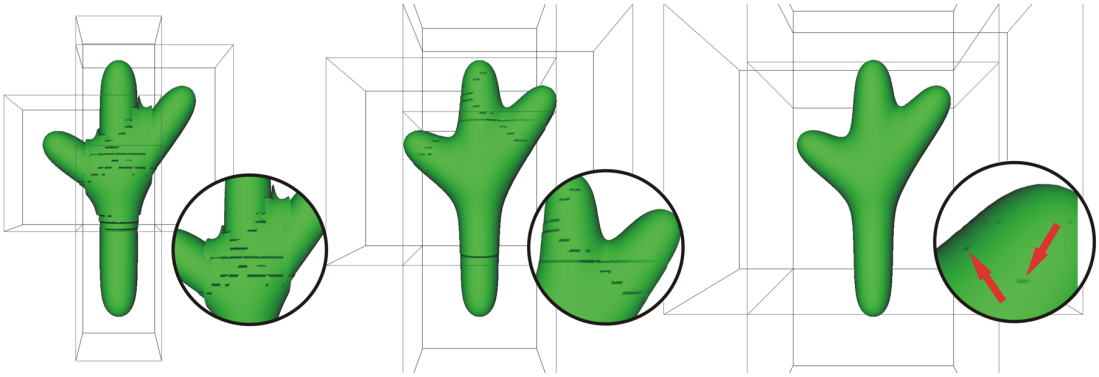


Abbildung 4.37: Trifurkation modelliert unter Verwendung der inversen kubischen Filterfunktion [CH01]. Aus Gründen der Übersichtlichkeit wurde auf die Darstellung der zylindrischen Hüllkörper verzichtet. Der signifikante Skalarfeldbereich ist durch Multiplikation von des maximalen Radius der *Convolution Surface* r_{max} mit den Faktoren $1.5, 3.0$ und 4.5 (von links nach rechts) festgelegt worden. Selbst bei einer 9-fachen Hüllkörpergröße (Faktor 4.5) sind Artefakte entlang der Oberfläche zu beobachten. Weitere Tests haben gezeigt, dass die Oberfläche erst ab einem Faktor gleich 6.0 artefaktfrei ist.

4.4 Generierung eines geometrischen Modells

Alle Abbildungen von realen und künstlichen Gefäßstrukturen innerhalb dieses und der folgenden Kapitel wurden mit Hilfe des *Implicit Polygonizers* [Blo94] generiert. Diese konkrete Umsetzung eines *Surface Tracking*-Verfahrens wurde bereits in Abschnitt 3.4.2.2 vorgestellt. Dieser Abschnitt konzentriert sich daher ausschließlich auf eine Adaption des zugrundeliegenden Algorithmus im Rahmen der Gefäßvisualisierung. Weiterhin sollen Parameter des Verfahrens extrahiert werden, deren Kontrolle in einer Anwendung möglicherweise an den Benutzer übergeben werden kann.

4.4.1 Parameter „Würfelgröße“

Zunächst muss die Größe des würfelförmigen Partitionierungselements festgelegt werden. Diese sollte nicht zu klein gewählt werden, um die Generierung unnötig vieler Dreiecke zu vermeiden. Ist der Würfel andererseits zu groß, werden sehr dünne Gefäßteile eventuell nicht angezeigt. Ein adäquater Standardwert sollte bestimmt werden, welcher garantiert in einer kompletten Darstellung der Gefäßstruktur resultiert. Im Rahmen der Gefäßvisualisierung kann die Voxelausdehnung des zugrundeliegenden Datensatzes genutzt werden, um einen solchen Standardwert festzulegen. Ein Voxel bildet die kleinste Einheit dieses Datensatzes und wurde während der Gefäßsegmentierung entweder als Ganzes der Gefäßstruktur zugeordnet oder als nicht relevant klassifiziert. Da unterhalb der Voxelausdehnung keine Zuordnung erfolgen kann, sind keine Gefäße mit einem Durchmesser kleiner der minimalen Voxelausdehnung detektiert worden. Die Größe eines Voxels sei $(x_{vox}, y_{vox}, z_{vox})$. Die korrespondierende Würfelgröße ist dann gegeben durch: $min(x_{vox}, y_{vox}, z_{vox})$. Unter Verwendung dieses Wertes wird die Gefäßstruktur komplett dargestellt. Die visuelle Qualität der Polygonisierung sehr dünner Gefäßäste kann durch eine

kleinere Würfelgröße noch verbessert werden. Hierbei ist jedoch zu beachten, dass eine Halbierung der Größe des Würfels eine ungefähr um den Faktor 4 erhöhte Dreieckszahl zur Folge hat. Weiterhin steigt die für die Polygonisierung benötigte Zeit erheblich.

Die Kontrolle des Parameters *Würfelgröße* kann dem Benutzer überlassen werden. Ein sinnvoller Standardwert ist das Minimum der Voxelauflösung in x-, y- und z-Richtung. Dieser Wert sollte gleichzeitig die obere Grenze definieren. Für alle Werte darunter sollten die Konsequenzen (zu erwartende Dreieckszahl und Rechenzeit) vor der Umsetzung abgeschätzt und dem Benutzer angezeigt werden.

4.4.2 Parameter „Startpunkt“

Der *Implicit Polygonizer* verlangt die Vorgabe eines Startpunktes in der Nähe des Objektes durch den Benutzer. Für die Gefäßvisualisierung hingegen kann ein beliebiger Punkt des Gefäßskeletts genutzt werden. Da der Gefäßbaum als gerichteter Graph vorliegt, empfiehlt sich die Verwendung eines Knotens, wie z.B. der Wurzel. Das Vorzeichen der impliziten Funktion ist für einen auf diese Art und Weise gewählten Startpunkt bereits bekannt, da dieser mit Sicherheit innerhalb der implizit beschriebenen Oberfläche lokalisiert ist. Der Parameter *Startpunkt* wird vollkommen durch das Programm kontrolliert und muss nicht vom Benutzer festgelegt werden.

Gesucht wird nun ein Pendant mit entgegengesetztem Vorzeichen, d.h. ein Punkt außerhalb der Gefäßstruktur. Dazu wird die Umgebung des Startpunktes ausgehend von diesem radial abgetastet (Abb. 4.38). Der Abstand von dem Startpunkt wird systematisch erhöht und die Richtung zufällig gewählt. An jedem Abtastpunkt wird die implizite Funktion erneut berechnet. Der Vorgang bricht ab, sobald ein Punkt mit entgegengesetztem Vorzeichen gefunden wurde.

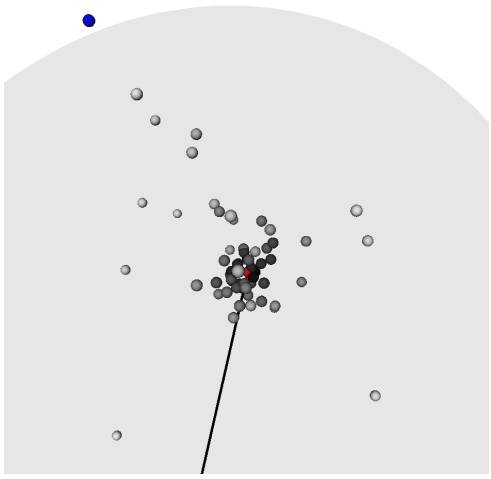


Abbildung 4.38: Bestimmung eines Punktes außerhalb der Gefäßstruktur. Die Umgebung des Startpunktes (rote Kugel an der Segmentspitze) wird radial abgetastet (graue Kugeln), bis ein Punkt außerhalb der Gefäßstruktur (blaue Kugel) gefunden ist. Dunklere Kugeln entsprechen früher geprüften Abtastpunkten.

Eine Gerade welche den Startpunkt mit seinem Pendanten verbindet, wird dazwischen von der impliziten Oberfläche geschnitten. Der Schnittpunkt wird mittels *binary subdivision* bestimmt und ein Startwürfel an diesem Punkt zentriert (Abb. 4.39).

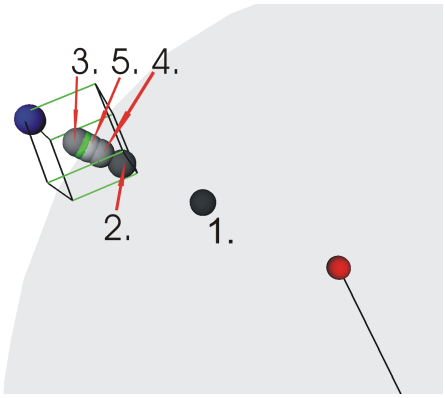


Abbildung 4.39: *Binary subdivision*-Prozess zur Detektion eines initialen Oberflächenpunktes. Die Gerade durch einen Punkt innerhalb der Gefäßstruktur (rote Kugel) und einen Punkt außerhalb (blaue Kugel) wird von der impliziten Oberfläche geschnitten. Der Schnittpunkt (verdeckte grüne Kugel) wurde hier nach 5 Iterationen festgelegt. Ein Würfel kennzeichnet das initiale Partitionierungselement zentriert an dem Schnittpunkt.

4.4.3 Parameter „Iterationstiefe“

Nachdem die Oberfläche komplett durch Würfel eingefasst wurde, erfolgt die Polygonisierung des jeweils in einem Würfel enthaltenen Oberflächenabschnitts. Für die Bestimmung der Schnittpunkte von Würfelkanten und der impliziten Oberfläche wird wiederum *binary subdivision* eingesetzt. Die Iterationstiefe des *binary subdivision*-Algorithmus entscheidet über die Genauigkeit, mit welcher die implizit beschriebene Oberfläche durch das polygonale Modell approximiert wird. Da Iterationstiefe und benötigte Rechenzeit proportional sind, muss ein Kompromiss zwischen Geschwindigkeit und Genauigkeit gefunden werden. Dieser Aspekt ist im Rahmen der Gefäßvisualisierung von besonderem Interesse angesichts der sehr komplexen hier zu visualisierenden Modelle. Zu klären ist, ob die in [Blo94] vorgeschlagene Tiefe von 10 Iterationen gesenkt werden kann, ohne große Einbußen bei der Genauigkeit in Kauf nehmen zu müssen.

Die Genauigkeit ist von der Größe g_{PE} des Partitionierungselements PE und von der Iterationstiefe i abhängig. Wird eine Kante von PE von der impliziten Oberfläche geschnitten, so wird sie durch *binary subdivision* i mal unterteilt. Hieraus folgt, dass der Schnittpunkt mit der Oberfläche nach Terminierung des *binary subdivision*- Prozesses garantiert innerhalb eines $g_{PE}/2^i$ großen Abschnitts der Kante lokalisiert ist. Somit ist der durchschnittliche Fehler zwischen impliziter Beschreibung der Oberfläche und tatsächlichem Oberflächenpunkt kleiner als $g_{PE}/2^i mm$. In Abschnitt 4.4.1 wurde die minimale Ausdehnung eines Voxels Standardwert für g_{PE} vorgeschlagen. Bei medizinische Volumendatensätze entspricht g_{PE} meist einer Größe $< 1mm$. Hieraus resultiert eine obere Grenze für den durchschnittlichen Fehler von $1/2^i mm$. Bei einer Iterationstiefe gleich 10 entspricht dies einem Wert von $\approx 0,00098mm$. Im Rahmen dieser Arbeit wurde aus Performanzgründen ein Standardwert von 3 Iterationen festgelegt was in einer oberen Grenze von $0.125mm$ resultiert.

Der Parameter *Iterationstiefe* sollte in eine feste Konstante innerhalb der Anwendung überführt werden. Die Genauigkeit der Approximation der impliziten Oberfläche ist für den Benutzer nicht sichtbar und kann daher auch nicht von ihm kontrolliert werden.

4.4.4 Parameter „Zelltyp“

Die Polygonisierung des Oberflächenanteils eines Partitionierungselements erfolgt entweder direkt [Blo88] oder nach einer Zerlegung des Würfels in Tetraeder (*Tetrahedral Decomposition* [PT90]). Die Implementierung beider Ansätze vermeidet die Entstehung von Löchern in der Oberfläche durch Mehrdeutigkeiten (*ambiguity cases*) bei der Polygonisierung eines Würfels [Blo88]. Letztere Methode beschreibt den eleganteren Weg [Blo94], resultiert jedoch in einer deutlich größeren Anzahl von Dreiecken und einem höheren Rechenaufwand. Ein Vergleich der Zelltypen Würfel und Tetraeder in [AG01] hat gezeigt, dass durch *Tetrahedral Decomposition* ca. die dreifache Anzahl von Dreiecken generiert wird.

Die Übernahme des Parameters *Zelltyp* scheint im Rahmen der Gefäßvisualisierung nicht sinnvoll. Aus Performanzgründen wurde sich hier gegen *Tetrahedral Decomposition* entschieden. Ein Qualitätsvergleich mit der Methode der direkten Polygonisierung wurde aus Zeitgründen nicht durchgeführt. Die auf diesem Weg erzeugten Gefäßmodelle sind jedoch von hervorragender visueller Qualität und offenbaren keine Notwendigkeit für den Einsatz von Tetraedern.

4.4.5 Parameter „Ausbreitungsgrenze“

Ein letzter Parameter des *Implicit Polygonizers* beschränkt die mögliche Ausbreitung der Unterteilung des Raumes um das Objekt. Dies ist sinnvoll für die Visualisierung unbegrenzter Objekte wie einer Ebene oder der Äquipotenzialfläche zweier Torii [Blo94]. Ohne Begrenzung würde der Algorithmus, im speziellen das Fortführungsschema nach [WMW86], hier nicht terminieren. Da Gefäßstrukturen jedoch stets in einem endlichen Raum definiert sind, besteht diese Gefahr hier nicht. Der Parameter *Ausbreitungsgrenze* (*bounds*) kann daher vernachlässigt werden.

4.4.6 Oberflächennormalen

Während der Polygonisierung muss die implizite Funktion sowohl für die Berechnung der Oberflächenpunkte, als auch für die Bestimmung der Normalen an einem Punkt p berechnet werden. Hierbei wird die in Abschnitt 4.3.1 entwickelte und in einem Vorverarbeitungsschritt generierte Datenstruktur genutzt (Abb. 4.32). Diese enthält den in einem bestimmten Bereich des Raumes relevanten Gefäßabschnitt. Zuerst wird Punkt p aus dem Weltkoordinatensystem in das Voxelkoordinatensystem transformiert. Die Identifikation des Voxels Vox_ID welches p enthält, wird nun aus dessen Koordinaten berechnet. Diese bildet den Schlüssel für eine Anfrage an die Datenstruktur hinsichtlich des an der Stelle p relevanten Skelettabschnitts. Basierend auf den so gewonnenen Liniensegmenten wird die implizite Funktion am Punkt p kalkuliert.

4.5 Interaktion

Zusammen mit einer Visualisierung der Gefäßstruktur spielt die Interaktion mit dem fertigen polygonalen Modell eine wichtige Rolle in der medizinischen Therapieplanung. Neben einfachen Formen der Interaktion, wie rotieren eines Objektes und zoomen, sind fortgeschrittenere Interaktionsmöglichkeiten wünschenswert. Essenziell für die Erforschung von Gefäßstrukturen ist

die Möglichkeit der Selektion von Teilbäumen oder einzelnen Gefäßästen [HPSP01]. Die Vorbereitung einer solchen Interaktion soll im nächsten Abschnitt beschrieben werden. Eine weitere Interaktionsform bildet die Einfärbung der Gefäßstruktur entsprechend der Verzweigungsstruktur aus der Gefäßanalyse (siehe Anhang A.1). Diese wird in Abschnitt 4.5.2 näher erläutert.

4.5.1 Selektion von Teilbäumen

Bei einer Interaktion durch den Benutzer ist es notwendig, diesem eine visuelle Rückmeldung über die Folgen seines Handelns zu geben. So sollten Gefäßteile bei einer Selektion hervorgehoben werden. Dies erfordert die Extraktion eines bestimmten Bereichs der polygonalen Struktur und die Änderung der korrespondierenden Materialeigenschaften. Dazu ist es notwendig, eine Korrespondenz zwischen Gefäßabschnitten und den sie repräsentierenden Dreiecken herzustellen. Da eine kantenbasierte Interaktion hier als ausreichend detailliert angesehen wird, ist die Etablierung einer Korrespondenz zwischen den Kanten des Gefäßgraphen und den zugehörigen Dreiecken erforderlich (Abb. 4.40).

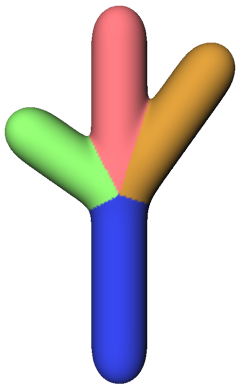


Abbildung 4.40: Zuordnung der Oberflächendreiecke zu den Kanten einer Trifurkation.

Die Ausgabe des *Implicit Polygonizers* erfolgt, wie in Abschnitt 3.4.2.2 beschrieben, im *Punkte/-Polygone*-Format. Die Dreiecke werden hier durch Indizes in die Felder der Oberflächenpunkte und Oberflächennormalen beschrieben. Sie sind alle in einem Feld gespeichert. Stattdessen sollte jede Kante des Gefäßgraphen eine eigene Struktur halten, welche die Indizes der zu ihre korrespondierenden Dreiecke enthält. Während der Polygonisierung muss nun für jedes Dreieck bestimmt werden, zu welcher Kante es gehört. Dann kann es in die entsprechende Datenstruktur eingefügt werden.

Die Zugehörigkeit kann leicht innerhalb des Polygonisierungsschrittes während der Berechnung der Feldfunktion an einem Punkt p geklärt werden. Nachdem der an p relevante Skelettabschnitt bestimmt wurde, wird für alle dort enthaltenen Liniensegmente ihr Beitrag zu dem finalen Wert der Feldfunktion berechnet. Das Segment mit dem maximalen Beitrag hat den größten Einfluss an der Stelle p. Die zu dem Segment gehörende Kante e des Gefäßgraphen wird zwischengespeichert. Ist p ein Oberflächenpunkt (der *binary subdivision*-Prozess ist terminiert), so wird e für diesen endgültig vermerkt. Sind alle drei Eckpunkte eines Dreiecks berechnet, so wird aus den jeweils korrespondierenden Kanten die endgültige Zugehörigkeit des Dreiecks bestimmt.

4.5.2 Einfärbung der Gefäßstruktur

In der Gefäßanalyse kann Teilen der Gefäßstruktur entsprechend 4 verschiedener Kriterien eine Farbe zugewiesen werden. Die Kriterien sind:

- *maxtrees*: Nach der Erzeugung möglichst großer Unterbäume eines Gefäßbaums wird jedem Unterbaum eine Farbe zugewiesen. Die korrespondierenden Kanten werden entsprechend eingefärbt.
- *hierarchy*: Jeder Hierarchiestufe des Gefäßbaums wird eine Farbe zugewiesen. Alle Kanten werden dann entsprechend ihrer Hierarchiestufe im Gefäßbaum eingefärbt.
- *risk*: Die Einfärbung jedes Skelettvoxels erfolgt entsprechend des Abstandes zu einem benutzerdefinierten Punkt. Dies ist in der Therapieplanung typischerweise ein Tumor.
- *distance:*: Jeder Skelettvoxel wird abhängig von seinem Abstand zu der Gefäßwurzel eingefärbt.

Die Farbinformation wird für jede Kante und dort für jeden Voxel einzeln abgespeichert (siehe Anhang A.1). Sie soll nun auf das polygonale Modell abgebildet werden, d.h. jedem Dreieck wird eine Farbe zugewiesen. Hierzu ist wie bei der Selektion von Teilbäumen im vorigen Abschnitt die Etablierung einer Korrespondenz zwischen den Kanten des Gefäßgraphen und den zugehörigen Oberflächendreiecken notwendig. Da selbst den Voxeln einer Kante unterschiedliche Farben zugeschrieben sein können (siehe die letzten beiden Einfärbekriterien) ist eine noch differenziertere Zuordnung nötig. Für jedes Dreieck muss zusätzlich eruiert werden können, welche Skelettvoxel die Farbgebung des Dreiecks beeinflussen. Um kontinuierliche Farbübergänge zwischen unterschiedlich eingefärbten Gefäßteilen zu gewährleisten, sollte jedem Eckpunkt eines Dreiecks ein Farbwert zugewiesen werden können. Die Dreiecksfarbe wird dann aus den Farben der Eckpunkte interpoliert. Neben den Feldern mit Oberflächenpunkten und deren Normalen ist also ein weiteres Feld erforderlich, welches die Farbe zu jedem Punkt der Oberfläche enthält.

Im vorigen Abschnitt wurde bereits beschrieben, wie eine Zugehörigkeit zwischen Oberflächendreieck und einer Kante e des Gefäßgraphen bestimmt werden kann. Diese Korrespondenzbeziehung wird nun im selben Schritt noch verfeinert. Neben der Bestimmung des Liniensegments mit dem höchsten Beitrag an der Stelle p wird auch der Endpunkt dieses Segments mit maximalem Einfluss auf p bestimmt. Der zu diesem Endpunkt korrespondierende Voxel wird zusammen mit der zu dem Segment korrespondierenden Kante vermerkt. Diese Information gestattet eine Umschaltung der Einfärbung durch den Benutzer und sollte im Speicher gehalten werden, um eine Farbwechsel jederzeit zu ermöglichen. Für jeden Eckpunkt eines Dreiecks ist so der korrespondierende Skelettvoxel bekannt und dessen vier Farbwerte können entsprechend appliziert werden.

4.5.3 Transparenz und Gefäßskelett

Die Wichtigkeit einer visuellen Rückmeldung für den Benutzer bei der Selektion von Teilbäumen wurde bereits in Abschn. 4.5.1 betont. Eine Möglichkeit bietet die Änderung der Farbe des ausgewählten Gefäßabschnitts. Es existieren jedoch Anwendungsszenarien, in denen eine abgeschwächte Darstellung der selektierten Struktur wünschenswert ist. Sind mehrere Gefäßbäume (z.B. Pfortader u. Arterien der Leber) gleichzeitig visualisiert, wird ein besserer Überblick durch

die halbtransparente Darstellung eines Baumes erreicht [HPSP01]. Durch diese Vorgehensweise kann sich eine Untersuchung auf den opaken Gefäßbaum konzentrieren ohne durch andere Strukturen behindert zu werden. Neben den Feldern mit Oberflächenpunkten, deren Normalen und Farbwerten ist ein weiteres Feld erforderlich, welches die Transparenz zu jedem Punkt der Oberfläche enthält. Die Werte des Feldes können dann während einer Interaktion verändert werden.

Die halbtransparente Darstellung einer Gefäßstruktur verschlechtert mit zunehmendem Grad der Transparenz ein Verständnis der Gefäßtopologie. Eine Möglichkeit zur Erhaltung dieser Information bietet die zusätzliche Einblendung des Gefäßskeletts. Die alleinige Darstellung des Skeletts ohne jegliche Oberflächeninformation ist zur Hervorhebung der anderen Strukturen ebenso denkbar.

5 Realisierung und Resultate

Dieses Kapitel beschreibt die Realisierung der zuvor konzipierten Gefäßvisualisierungsmethode basierend auf *Convolution Surfaces*. Begonnen wird mit der Vorstellung hierzu verwendeter Programmierwerkzeuge. Anschließend wird die Implementierung einer Datenstruktur zur effizienten Polygonisierung diskutiert. Darauf folgt eine Darlegung von programmiertechnischen Aspekten der Polygonisierung, Darstellung und Vorbereitung einer Interaktion mit der Gefäßstruktur. Den Abschluss dieses Kapitels bilden die Präsentation einiger während dieser Arbeit erzielter Visualisierungsresultate und deren Einschätzung hinsichtlich der in der Einleitung aufgestellten Anforderungen an eine ideale Gefäßvisualisierungsmethode.

5.1 Programmierwerkzeuge

Ein wichtiger Gesichtspunkt bei der Auswahl der Programmierwerkzeuge war die Ermöglichung einer späteren Integration der Gefäßvisualisierungsmethode in ILAB (*Image Laboratory*). ILAB ist eine Softwareplattform für die Forschung und Entwicklung in der computergestützten medizinischen Bildgebung. Mit der Integration wurde bereits während dieser Arbeit begonnen. Bis zu einer vollen Funktionsfähigkeit sind jedoch weitere Schritte wie wie die Abbildung von Parametern der Polygonisierung durch eine Benutzerschnittstelle nötig.

Für die Implementierung der zuvor konzipierten Methode zur Gefäßvisualisierung wurde mit *C++* eine objektorientierte Programmiersprache gewählt, um eine übersichtliche Strukturierung, die leichte Erweiterbarkeit sowie die Wiederverwendbarkeit des Programms zu ermöglichen. C++ besitzt eine hohe Portabilität und ist unter den meisten heutzutage üblichen Systemen (UNIX, LINUX, MS-WINDOWS, ...) einsetzbar. Zur Umsetzung der Visualisierung und Interaktion mit der Gefäßstruktur wurde sich für die 3D-Graphikbibliothek *OpenInventor* entschieden, da diese innerhalb der Softwareplattform ILAB unterstützt wird. OpenInventor ist in C++ implementiert, was wiederum die Entscheidung zu Gunsten dieser Programmiersprache beeinflusst hat. Die in Abschnitt 4.3.1 entworfene Datenstruktur zur Performanzsteigerung sowie zahlreiche während der Generierung eines geometrischen Modells benötigte Datenstrukturen wurde mit Hilfe der *Standard Template Library* (STL) von Dinkumware, Ltd. [1] realisiert. Diese C++ Bibliothek stellt mit verschiedenen Containerklassen und Algorithmen viele der grundlegenden Verfahren und Datenstrukturen der Informatik bereit. Als Entwicklungsumgebung für das gesamte Projekt Gefäßvisualisierung dienten Microsoft's *Visual Studio C++ 6.0* [2] und ILAB.

[1] `http://www.dinkumware.com/`, Stand: 5.02.2004
[2] `http://msdn.microsoft.com/visualc/`, Stand: 5.02.2004

5.1.1 Die 3D-Graphikbibliothek OpenInventor

OpenInventor ist eine auf der 3D-Graphikbibliothek *OpenGL* basierende Bibliothek aus Objekten und Methoden zur Generierung interaktiver 3D-Graphikapplikationen. Einfache OpenGL-Basisfunktionen sind hier zu komplexen Klassen verknüpft und in einem hierarchischen System zusammengefasst. Dieses System ist aufgrund des objektorientierten Ansatzes leicht modifizierbar und erweiterbar. Die Architektur von OpenInventor ist in Abb. 5.1 skizziert. Die grundlegenden Bausteine bilden OpenGL und das jeweilige Betriebsystem (hier exemplarisch durch UNIX beschrieben). Darauf aufbauend folgt das Herzstück von OpenInventor bestehend aus einer 3D-Datenbasis (*Scene Database*), *Nodekits* und Manipulatoren (*Manipulators*). Die 3D-Datenbasis enthält alle zur Generierung einer Szene notwendigen Objekte, wie Primitive, Materialeigenschaften, Kameras und Lichtquellen. Ein *Nodekit* ist die vordefinierte Anordnung einer Gruppe von Knoten und dient als Muster für die Konstruktion von Objekten mit ähnlichen Eigenschaften. So besteht ein Auto meist aus denselben Bauteilen, die sich nur im Detail unterscheiden und verändert werden müssen. Manipulatoren sind Interaktionskomponenten und ermöglichen die direkte Veränderung von Objekteigenschaften, wie Größe oder Position, durch den Benutzer. Eine weitere Einheit der OpenInventor-Architektur ist das 3D-Datenformat (*3D Interchange File Format*). Hiermit können 3D-Szenen abgespeichert und innerhalb verschiedener Anwendungen ausgetauscht werden. Die finale Darstellung der Szene auf dem Bildschirm erfolgt mit Hilfe der OpenInventor-Komponentenbibliothek (*Component Library*). Bedingt durch die Anforderung an eine hohe Portabilität enthält OpenInventor selbst keine Fensterobjekte. Die Komponentenbibliothek bietet daher systemspezifische *Viewer*, die in eigene Applikationen eingebettet werden können.

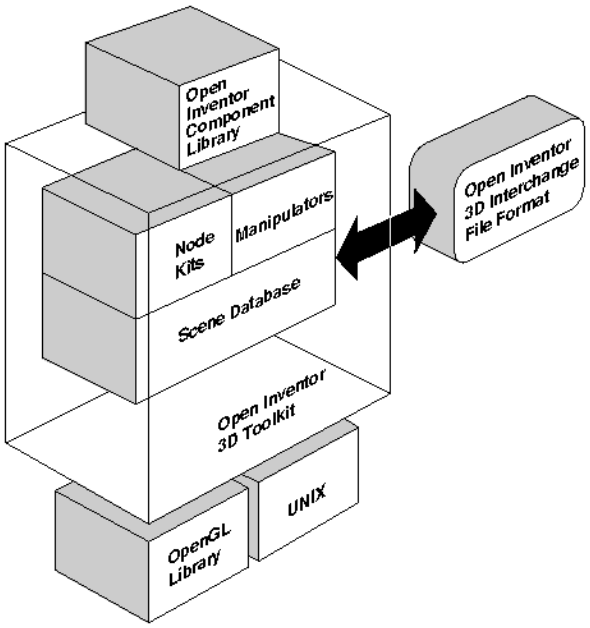


Abbildung 5.1: OpenInventor-Architektur. Quelle: [Wer].

Ein großer Vorteil von OpenInventor gegenüber OpenGL ist die Möglichkeit der Strukturierung einer Szene und die Abspeicherung dieser Struktur in einer Datei. So lassen sich Szeneneigenschaften, wie Beleuchtung, Graphikprimitive und Materialien, durch so genannte *Knoten*

beschreiben und in einem gerichteten, azyklischen *Szenengraphen* anordnen. Zur Illustration dieses Konzeptes ist in Abb. 5.2 (rechts) ein einfacher Szenengraph dargestellt. Das in Anlehnung an [Wer] konstruierte Beispiel beschreibt die polygonale Oberfläche eines Obelisken (Abb. 5.2 links).

Während des Rendering-Schrittes erfolgt die Traversierung des Graphen immer von links nach rechts und von oben nach unten. Ist ein Objekt weiter rechts oder weiter unten im Szenengraphen lokalisiert so gelten alle vorher spezifizierten Eigenschaften. Da dies nicht immer wünschenswert ist, können durch den Einsatz von *Separatoren* Teilbäume erzeugt werden. Diese bilden eine in sich geschlossene, isolierte Einheit. In dem Beispielgraphen aus Abb. 5.2 (rechts) werden zuerst die Kamera (2) und die Lichtquelle (3) in einer Szene spezifiziert. Danach erfolgt die Traversierung des durch einen Separator (4) isolierten Teilbaumes. Die zu Beginn festgelegte Transformation (5) wird auf die nachfolgenden Koordinaten der Oberfläche (6) angewendet. In diesem Beispiel wurde der Obelisk zur besseren räumlichen Vorstellung leicht um seine vertikale Achse gedreht. Danach werden die Normalen des Objektes (7,8) und dessen Materialeigenschaften (9,10) beschrieben. Abschließend folgt die Verknüpfung zu dem eigentlichen Oberflächenobjekt (11), dem Obelisken. Eine hervorragende Einführung zu OpenInventor sowie nützliche Tipps für den bereits versierten Nutzer gibt [Wer].

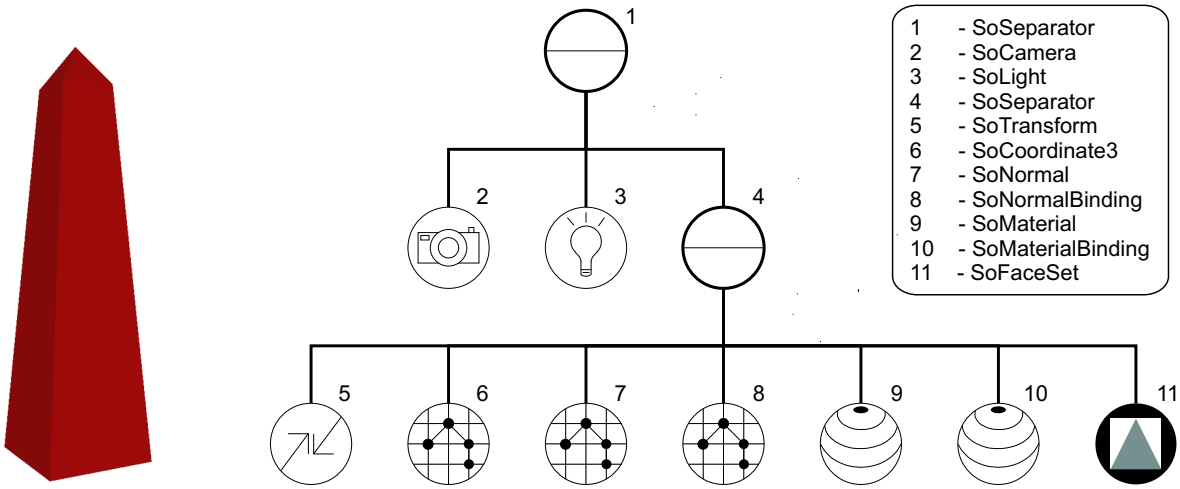


Abbildung 5.2: Modell eines Obelisken (*links*) und der korrespondierende OpenInventor-Szenengraph (*rechts*). Die einzelnen Elemente des Graphen werden als Knoten bezeichnet.

5.1.2 Die Softwareplattform ILAB

ILAB ist eine Softwareplattform für die Forschung und Entwicklung in der computergestützten medizinischen Bildgebung. Es wird seit 1993 von den Forschungszentren CeVis (*Center of Complex Systems and Visualization*) und MeVis (*Center for Medical Diagnostic Systems and Visualization*) in Bremen entwickelt. ILAB existiert momentan in der vierten Version für die Betriebssysteme MS-WINDOWS und LINUX. Es bietet vielfältige Möglichkeiten zur Verarbeitung und Visualisierung von 3D- und 4D-Daten. Das modulare Konzept von ILAB 4 gestattet die einfache Integration neuer Algorithmen und Visualisierungswerkzeuge als Module. Über eine graphische Schnittstelle können einzelne Module zu komplexen Netzwerken verknüpft werden.

Dies erlaubt eine intuitive Konzeption von auf spezifische klinische Aufgaben angepassten Applikationen ohne Kenntnis einer Programmiersprache.

Die Architektur von ILAB 4 ist in drei Ebenen gegliedert: Modulebene, Verarbeitungsebene und Anwendungsebene (Abb. 5.3). Ein zentrales Element dieser Architektur bildet die in C++ implementierte, beliebig erweiterbare ML-Bildverarbeitungsbibliothek (ML-*MeVis Image Processing Library*). Diese beinhaltet u.a. Methoden zur Segmentierung, morphologischen Analyse oder Registrierung von anatomischen und pathologischen Strukturen. Für alle Visualisierungsaufgaben innerhalb von ILAB 4 wird die zweite zentrale Komponente, die 3D-Graphikbibliothek OpenInventor genutzt. ILAB 4 bietet die volle OpenInventor Funktionalität inklusive zusätzlicher Erweiterungen wie *Volume Rendering* oder *Surface Rendering*. Die auf Modulebene definierten Module nutzen jeweils einen Teil der Funktionalität aus ML-Bildverarbeitungsbibliothek oder OpenInventor. Auf der Anwendungsebene kann für jedes Modul eine angepasste Benutzerschnittstelle entworfen werden. Hierzu wird die eigens für ILAB entworfene Auszeichnungssprache MDL (*Module Description Language*) genutzt. Alle gängigen Elemente einer graphischen Benutzeroberfläche, wie *buttons, sliders oder check boxes*, werden durch die Sprache unterstützt. ILAB nutzt zusätzlich JAVASCRIPT, um die statische Auszeichnungssprache MDL mit einer dynamischen Komponente zu erweitern. Beispielhaft für die Architektur und Funktionalität von ILAB 4 ist in Abb. 5.4 eine Anwendung zur Volumenbestimmung der menschlichen zerebralen Ventrikel dargestellt. Weitere Informationen zu ILAB 4 findet der interessierte Leser in [HLP03].

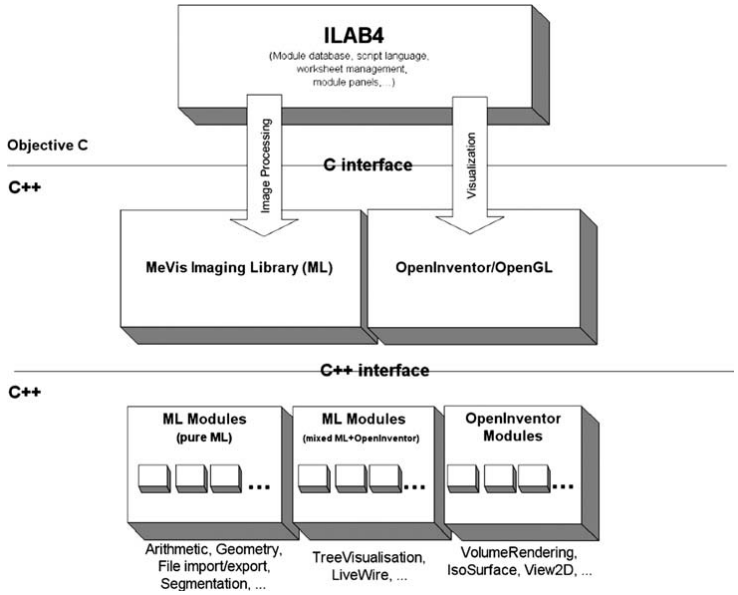


Abbildung 5.3: 3-Ebenen Architektur von ILAB 4: Modulebene, Verarbeitungsebene und Anwendungsebene (von unten nach oben). Quelle: [HLP03].

5.2 Vorverarbeitung

Die in der Gefäßanalyse gewonnenen Daten sind in einer Datei abgespeichert (siehe Anhang A.1) und müssen zu Beginn eingelesen und auf Datenstrukturen abgebildet werden. Ein Teil des von O. Konrad-Verse im Rahmen seiner Tätigkeit bei MeVis entwickelten ILAB-Moduls zur

Abbildung 5.4: Ein Netzwerk aus mehreren bestimmten Modulen liefert die gewünschte Funktionalität (a). Über die graphische Benutzerschnittstelle eines Moduls können dessen Parameter verändert werden (b). Die ILAB 4 Applikation *Ventricel Volumetry* ermöglicht die Segmentierung und Volumenbestimmung der menschlichen zerebralen Ventrikel (c). Mittels *Volume Rendering* kann das Segmentierungsresultat dargestellt werden (d). Quelle: [HLP03].

Visualisierung von Gefäßstrukturen nach [HPSP01] bewältigt diese Aufgabe. Dieser Teil wurde extrahiert und in ein neues Modul für die Gefäßvisualisierung mit *Convolution Surfaces* integriert. Der so genannte `SoVesselParser` ist eine in C++ implementierte Klasse und bildet die Gefäßinformation auf Instanzen der Unterklassen: `Knoten` (*node*), `Kante` (*edge*) und `Metainformation` (*treemeta*) ab. Die Struktur der Klassen entspricht exakt der in Anhang A.1 gegebenen Dateistruktur. Daher wird hier auf eine Darstellung der Klassenstruktur verzichtet. Die Variablen der Instanz einer Klasse speichern die lokale Gafäßinformation während die Klassenmethoden im Wesentlichen das Setzen und Auslesen dieser Information steuern. So besitzt die Klasse Kante beispielsweise eine Variable `_edge_id` vom Datentyp *Integer*, deren Wert durch die Methoden `setEdgeID(int id) {_edge_id = id;};` und `getEdgeID()` gesetzt bzw. ausgelesen werden kann. Eine erweiterte Funktionalität des `SoVesselParser` besteht in der Umsetzung der Nachbearbeitungsschritte nach [HPSP01] (siehe Abschnitt 2.3.2.2) zur Verbesserung des Gefäßskeletts. So stehen Methoden zur Punktglättung und zur Radiusglättung zur Verfügung. Die Kontrolle des Glättungsfaktors kann als Parameter an die Benutzerschnittstelle übergeben werden.

Im Hinblick auf eine effiziente Polygonisierung wurde in Abschnitt 4.3.1 eine Datenstruktur entworfen, welche vollständig in einem Vorverarbeitungsschritt generiert werden kann und während der Polygonisierung Anfragen hinsichtlich des an einem Punkt p relevanten Skelettabschnitts beantwortet. Der folgende Abschnitt ist der Implementierung dieser Datenstruktur gewidmet. Weiterhin wird hier die Performanzsteigerung durch die Anwendung dieser Datenstruktur evaluiert.

5.2.1 Hüllkörper

Zu jedem Liniensegment des Gefäßskeletts soll der signifikante Skalarfeldbereich voxelweise erfasst werden. Für die hierbei akkumulierte Voxelmenge wird das aktuelle Liniensegment dann in einer Datenstruktur vermerkt. Somit müssen zwei Arten von Information gespeichert werden. Zum einen die Voxel, welche beeinflusst werden und zum anderen die sie beeinflussenden Liniensegmente. Eine zweidimensionale Datenstruktur ist hierfür geeignet (Abb. 4.32). Da ein Voxel in dem dem signifikanten Skalarfeldbereich mehrerer Liniensegmente liegen kann, sollte ein mehrmaliges Einfügen zur Vermeidung von Redundanz ausgeschlossen werden. Jeder Voxel erhält deshalb eine einzigartige Identifikation (Vox_ID) berechnet aus seinen Koordinaten im Voxelkoordinatensystem. Anhand dieser Identifikation kann bestimmt werden, ob ein Voxel schon in der Datenstruktur enthalten ist oder nicht. Auf diese Weise können zu einem bereits bestehenden Eintrag Liniensegmente hinzugefügt werden bzw. ein neuer Eintrag kann angelegt werden.

Sowohl die Suche nach einer gegebenen Voxelidentifikation, als auch das Einfügen eines neuen Eintrags sollte möglichst schnell erfolgen, da bei komplexen Gefäßbäumen mit einer beträchtlichen Anzahl von Voxeln zu rechnen ist. Das Einfügen neuer Liniensegmente in einen bestehenden Eintrag muss ebenfalls effizient möglich sein. Da anfangs nicht bekannt ist, wieviele Voxel des gesamten Voxelgitters überhaupt erfasst werden und auch keine Aussage darüber getroffen werden kann, wieviele Liniensegmente Einfluss in einem Voxel besitzen, empfiehlt sich die Verwendung dynamischer Datenstrukturen. Die Anforderungen an eine geeignete Datenstruktur sind wie folgt noch einmal zusammengefasst:

- zweidimensional
- dynamisch in jeder Dimension
- schnelles Einfügen und Suchen
- keine Datenredundanz

5.2.1.1 Die Datenstrukturen map und vector

Eine Kombination der Datenstrukturen *map* und *vector* aus der C++ *Standard Template Library* (STL) erfüllt diese Anforderungen. Eine *map* ist ein dynamischer assoziativer Container welcher Objekte vom Typ *Schlüssel* mit Objekten vom Typ *Daten* verknüpft. Ein Eintrag in der *map* hat die folgende Form: $< Schluessel, Daten >$, wobei *Schluessel* der einzigartigen Voxelidentifikation entspricht, während *Daten* die Liniensegmente pro Voxel repräsentieren soll. Eine Besonderheit der Datenstruktur *map* ist, das keine zwei Einträge denselben Schlüssel besitzen. Dies entspricht genau der Anforderung einer Vermeidung von Datenredundanz. Weiterhin sind die Einträge hinsichtlich ihres Schlüssels sortiert. Die *map* ist als ausgewogener binärer Suchbaum (*balanced binary search tree*), genauer als Rot/Schwarz-Baum implementiert. Dies ermöglicht das Einfügen und Suchen eines Elements in $O(\log n)$ Zeit, wobei n die Anzahl der Einträge in der *map* beschreibt.

Zu jedem Schlüssel in der Datenstruktur korrespondiert ein Objekt vom Typ *Daten*, welches die Liniensegmente pro Voxel enthalten soll. Dieses Objekt repräsentiert die zweite Dimension der Datenstruktur und wird durch einen *vector* realisiert. Ein *vector* ist ein dynamisch erweiterbares

eindimensionales Feld. Dessen Elemente sind zusammenhängend angeordnet, was Zugriffsoperationen in konstanter Zeit $O(1)$ ermöglicht. Die Zeit für das Einfügen eines Elementes in einen *vector* ist entweder eine $O(1)$- oder $O(n)$-Operation, wobei n die Anzahl der Einträge beschreibt. Letzteres gilt, wenn die Größe des *vectors* erweitert werden muss, da hierzu die gesamte Struktur kopiert wird. Für große zu erwartende Datenmengen ist es demnach angebracht, den *vector* mit einer sinnvoll abgeschätzten Größe zu initialisieren und zusätzlich zu vermerken, bis wohin die Datenstruktur bereits gefüllt ist. Im Rahmen der Gefäßvisualisierung ist dieser zusätzliche Aufwand jedoch nicht notwendig, da die Anzahl der Liniensegmente pro Voxel gering ist. Dies wird später in diesem Abschnitt noch durch Messungen belegt werden.

Jedes Liniensegment kann wie in Abschn. 4.3.1 erläutert durch ein Tupel der Form $< e, v >$ charakterisiert werden, wobei e die zugehörige Kantenidentifikation und v die Position des Startvoxels in der Voxelliste dieser Kante beschreibt. Dieses Tupel lässt sich unter C++ durch eine *Struktur* mittels *struct* definieren. Eine Struktur besitzt Felder, welche von unterschiedlichem Datentyp sein können. Die Felder hier sind Kantenidentifikation und Startvoxelposition des Liniensegments. Es existiert weiterhin eine alternative Struktur für die Repräsentation eines Liniensegments, bestehend aus Feldern für die beiden Endpunkte und die korrespondierenden Radien. Während Instanzen der ersten Struktur aus Effizienzgründen in der *map* gespeichert werden, dient die zweite Struktur einem leichteren Zugriff auf die Liniensegmentattribute. Die folgende Auflistung fasst alle für das weitere Verständnis der Vorverarbeitung wichtigen Klassen und Datenstrukturen zusammen:

- Klasse `SoVesselParser`: liest die Daten aus der Gefäßanalyse ein und bildet sie auf die folgenden Klassen ab:
- Klassen `Knoten`, `Kante`, `Metainformation`: speichern die durch den `SoVesselParser` eingelesenen Daten aus der Gefäßanalyse
- Struktur `Punkt3`: markiert einen Punkt oder Vektor in 3D; Form einer Instanz: <x,y,z>
- Struktur `Kante_Voxel`: repräsentiert ein Liniensegment durch die Identifikation der zugehörigen Kante e des Gefäßgraphen und die Position des zu dem ersten Segmentendpunkt korrespondierenden Voxels v in der Voxelliste von e; Form einer Instanz: <e,v>
- Struktur `Segment`: repräsentiert ein Liniensegment durch die Koordinaten der beiden Endpunkte e_0, e_1 und die mit den Endpunkten assoziierten Radien r_0, r_1; Form einer Instanz: <e_0<`Punkt3`>, e_1<`Punkt3`>,r_0,r_1>
- *vector* `Liniensegmente`: speichert die Liniensegmente pro Voxel; Form einer Instanz: <`Kante_Voxel`, `Kante_Voxel`, ...>
- *map* `Voxel`: Resultat der Vorverarbeitung; speichert die Identifikation der detektierten Voxel (Vox_ID) und die jeweils korrespondierenden Liniensegmente; Form einer Instanz: < <Vox_ID, `Liniensegmente`>, <Vox_ID, `Liniensegmente`>, ... >

5.2.1.2 Datentypen der Datenstruktur

Die Voxelidentifikation wird aus den positiven, ganzzahligen Koordinaten des Voxelmittelpunktes im Voxelkoordinatensystem berechnet. Sie kann daher keine negativen Werte annehmen. Der Datentyp *Unsigned Integer* ist innerhalb von C++ für nicht-negative, ganzzahlige Werte vorgesehen. Der von diesem Datentyp beanspruchte Speicherplatz beträgt auf den meisten Systemen 4

Byte. Er kann somit Werte im Bereich $0...2^{32}-1$ annehmen. Dies ist selbst für höher-dimensionale Volumendatensätze (z.B. 1024x1024x1024) ausreichend. Die Tupel zur Repräsentation von Liniensegmenten bestehen aus zwei positiven, ganzzahligen Werten. Der Definitionsbereich des ersten Wertes ist durch die höchstmögliche Anzahl von Kanten festgelegt. Diese obere Grenze wurde von MeVis auf 10.000 festgelegt. Die Anzahl der Voxel pro Kante liegt in dem Bereich < 100. Aus diesen Gründen ist die Verwendung eines speichereffizienteren Datentyps als für die Voxelidentifikation möglich. Hier wird der Datentyp *Unsigned Short Integer* genutzt, welcher 2 Byte beansprucht und somit Werte im Bereich $0...2^{16}-1$ repräsentieren kann. Die erweiterte Struktur zur Repräsentation von Liniensegmenten sieht jeweils den Datentyp *float* für ihre vier Felder vor.

5.2.1.3 Auffüllung der Datenstruktur

Nachdem die *map* ($influences$) vom Typ `Voxel` initialisiert und eine Instanz (seg) der Struktur `Segment` angelegt wurde, iteriert der Algorithmus über alle Kanten des Gefäßbaumes. Für jede Kante wird jeweils aus den Koordinaten zweier benachbarter Voxel und den korrespondierenden Radien die Struktur *seg* aktualisiert. Die Koordinaten der Endpunkte in *seg* sind zu Beginn noch im VKS definiert. Für jedes Liniensegment wird nun mit Hilfe des VAH bestimmt, welche Voxelmenge V den signifikanten Skalarfeldbereich um das Segment einschließt. Hierbei sind der Skalarfelddurchmesser und eine geringfügige Justierung des VAH zu beachten (siehe Abschnitt 4.3.2.3). Die Mittelpunkte der in V enthaltenen Voxel werden dann gegen den ZH getestet. Vorher sind die in *seg* gespeicherten Koordinaten der Endpunkte in das WKS zu überführen.

Die Implementierung ist bis zu diesem Punkt einfach und wird deswegen hier nicht näher erläutert. Der Test gegen den ZH sowie das Einfügen der gewonnenen Information in die *map* sollen jedoch im Folgenden detaillierter beschrieben werden. Die Umsetzung des Hüllkörpertests erfolgt in Anlehnung an [She94]. Zusätzlich zu der dort beschriebenen Konstruktion eines impliziten Zylinders werden hier zwei Ebenen definiert, welche den iZH begrenzen. Innerhalb der folgenden *C++*-Notation des Algorithmus finden sich zahlreiche kleinere Funktionen.$MAXf$(float a, float b) berechnet das Maximum der Zahlen a und b. Die Funktionen *Add, Sub, Mul, Unit, Dot* und *Length* realisieren eine Vektoraddition-, -subtraktion, -multiplikation mit einem Skalar, die Normalisierung eines Vektors, das Skalarprodukt zweier Vektoren und die Bestimmung der Vektorlänge. Die Methoden *push_back* und *insert* dienen jeweils dem Einfügen eines Elements in die Datenstrukturen *vector* bzw. *map*. Die Methode $find(key)$ sucht einen Eintrag in der *map* passend zu dem Schlüssel *key*.

```
computeBoundingVolumes() {                                                    1
 ...                                                                          2
                                                                              3
 radius = MAXf(seg.r0, seg.r1)*1.5+radius_increase;                           4
 ZH_e0 = Add(seg.e0, Mul(Unit(Sub(seg.e0,seg.e1)),radius));                   5
 ZH_e1 = Add(seg.e1, Mul(Unit(Sub(seg.e1,seg.e0)),radius));                   6
 u = Unit(Sub(seg.e0,ZH_e0));                                                 7
                                                                              8
 for (x = borders[0]; x <= borders[3]; x++) {                                 9
  for (y = borders[1]; y <= borders[4]; y++) {                                10
   for (z = borders[2]; z <= borders[5]; z++) {                               11
                                                                              12
    query_pt = transformToWorldSpace(x, y, z);                                13
    t1 = Length((Sub(query_pt,vertex)));                                      14
    t2 = Dot(Sub(query_pt,vertex),u);                                         15
    dot1  = Dot(Sub(query_pt,ZH_e0),Sub(ZH_e1,ZH_e0));                        16
    dot2  = Dot(Sub(query_pt,ZH_e1),Sub(ZH_e0,ZH_e1));                        17
                                                                              18
    if ( ((t1*t1-t2*t2) <= (radius*radius)) && !(dot1<0 || dot2<0)            19
    {                                                                         20
      Kante_Voxel ev(edge->getEdgeID(), j);                                   21
                                                                              22
      Vox_ID = z*(y_dim*x_dim) + y*(x_dim) + x;                               23
      ptr_infl = influences.find(vox_id);                                     24
                                                                              25
      if (ptr_infl == influences.end()) {                                     26
       Liniensegmente Ls;                                                     27
       Ls.push_back(ev);                                                      28
       influences.insert( Vox_ID, Ls );                                       29
      }                                                                       30
      else {                                                                  31
       ptr_infl->second.push_back(ev);                                        32
}}}}}}                                                                        33
```

In Zeile (4) wird der Radius des impliziten Zylinders festgelegt. Dieser basiert auf dem Radius des signifikanten Skalarfeldbereichs um das Liniensegment ($MAXf(seg.r0, seg.r1) * 1.5$) plus der halben maximalen Ausdehnung eines Voxels (*radius_increase*). Diese Erhöhung stellt sicher, dass ein Voxel selbst dann den Test gegen den ZH besteht, wenn die Grenze des Skalarfeldbereichs durch einen seiner Eckpunkte verläuft. In den Zeilen (5) und (6) wird die Höhe des Zylinders durch die Länge seiner Achse festgelegt. Diese Achse verläuft zwischen den Punkten *ZH_e0* und *ZH_e1*. Beide Punkte sind konstruiert worden durch eine entgegengesetzte Verschiebung der Endpunkte von *seg* entlang der Achse durch das Liniensegment um den Betrag *radius*. Zeile (7) berechnet berechnet den normalisierten Vektor u vom Startpunkt des Liniensegments zu dem Startpunkt der Zylinderachse. In den nachfolgenden *for*-Schleifen werden alle Mittelpunkte der Voxel aus V gegen den ZH getestet. Das Feld *borders* enthält jeweils die minimale und die maximale Koordinate der Voxel aus V in x-, y- und z-Richtung. Vor dem Test wird der Voxelmittelpunkt zuerst in Zeile (13) in das WKS transformiert (*query_pt*). Dann wird sowohl die Länge des Vektors vom Startpunkt der Zylinderachse zu *query_pt* bestimmt (14), als auch das Skalarprodukt aus diesem Vektor und u (15). Die Zeilen (16) und (17) dienen der Lokalisation des Voxelmittelpunktes hinsichtlich der beiden Ebenen des ZH durch ZH_e0 bzw. ZH_e1 (siehe Abschnitt 4.3.2.1). Basierend auf diesen Ergebnissen kann der finale Hüllkörpertest ausgeführt werden (19). Hier wird zunächst getestet, ob sich der Voxelmittelpunkt im bzw. auf dem

impliziten Zylinder befindet. Ist dies nicht der Fall, gilt die *if*-Bedingung als nicht erfüllt und der Test als nicht bestanden. Wenn doch, wird der Test hinsichtlich der beiden Ebenen des ZH durchgeführt. Hat der Voxelmittelpunkt alle Tests bestanden, so wird das Liniensegment *seg* für diesen Voxel in der *map* vermerkt (21-33). In Zeile (21) wird zuerst eine speichereffiziente Form von *seg* angelegt. Das Liniensegment lässt sich vollständig durch die Identifikation der zugehörigen Kante und die Position j des zu e_0 korrespondierenden Voxels in der Voxelliste der Kante beschreiben. In Zeile (23) wird die einzigartige Voxelidentifikation *Vox_ID* berechnet. Mit Hilfe dieses Schlüssels kann nun in der *map* nach einem bereits existierenden Eintrag gesucht (24). Diese Suche resultiert in einem Zeiger *ptr_infl* auf den gefundenen Eintrag in der Datenstruktur. In der folgenden *if*-Bedingung (26-33) wird geprüft, ob der Zeiger auf ein gültiges Element verweist oder auf das Ende der *map* (*influences.end()*). Ist der Voxel noch nicht enthalten (27-29), wird ein neuer *vector* vom Typ `Liniensegmente` angelegt (27) und das Segment in der Form `Kante_Voxel` in diesen *vector* eingefügt (28). Danach wird der *vector* seinerseits als neuer Eintrag in die *map* eingefügt (29). Ist bereits ein Eintrag mit dem Schlüssel Vox_ID in der *map* vorhanden, wird der zugehörige *vector* um eine `Kante_Voxel`-Struktur erweitert (32).

Sind die Hüllkörpertests für alle Liniensegmente bzw. Kanten durchgeführt worden, terminiert der Algorithmus und die Datenstruktur steht für ihren Einsatz während der Polygonisierung bereit.

5.2.1.4 Performanztests

Die in Abschnitt 4.1 durchgeführten Visualisierungstests hatten gezeigt, dass eine Optimierung hinsichtlich der Rechenzeit dringend notwendig ist. Der Performanzgewinn durch die entworfenen Hüllkörper sollte nun diesbezüglich untersucht werden. Hierzu wurden die in Abschnitt 4.1 vorgestellten Gefäßstrukturen erneut visualisiert und die Ergebnisse miteinander verglichen (Tab. 5.1 bis 5.3). Die Zeiten markieren jeweils den Zeitraum vom Einlesen der Daten bis zu einer initialen Darstellung auf dem Bildschirm. Sukzessiv wurden die einzelnen Hüllkörper einbezogen und der jeweilige zusätzliche Gewinn gemessen. Der weiterhin bestimmte Speicherbedarf bezieht sich ausschließlich auf die *map*. Er repräsentiert die erforderliche Gegenleistung für eine Performanzverbesserung. Die letzten beiden Spalten der Tabellen spiegeln die Anzahl von Einträgen in der *map* bzw. die durchschnittliche Länge des *vectors* pro Eintrag wieder. Die Tests wurden durchgeführt unter Zuhilfenahme folgender Hardware: Intel Pentium 4 CPU 3.06GHz, 1024MB RAM, ATI Radeon 9600.

Tabelle 5.1: Performanzmessungen für den Lebergefäßbaum LG_1.

Hüllkörper	Zeit (m:s)	Speicher (MB)	Voxel	∅ Liniensegmente
ohne	11:07	/	/	/
VAH	0:11	10,4	204.089	11
iZH	0:09	6,8	125.961	12
ZH	0:08	6,1	117.232	11

Im Durchschnitt wird für die Visualisierung der drei Lebergefäßbäume durch den Einsatz der Hüllkörper die Rechenzeit auf weniger als 1% der ursprünglichen benötigten Zeit reduziert. Die leicht unterschiedlichen Rechenzeiten der Polygonisierung ohne Hüllkörper im Vergleich zu Tabelle 4.1 sind durch die Verwendung eines schmaleren Gauß-Filters bedingt. Die Gefäßform folgt

Tabelle 5.2: Performanzmessungen für den Lebergefäßbaum LG_2.

Hüllkörper	Zeit (m:s)	Speicher (MB)	Voxel	∅ Liniensegmente
ohne	27:18	/	/	/
VAH	0:20	17,6	290.248	14
iZH	0:16	11,4	209.296	12
ZH	0:14	10,0	189.003	12

Tabelle 5.3: Performanzmessungen für den Lebergefäßbaum LG_3.

Hüllkörper	Zeit (m:s)	Speicher (MB)	Voxel	∅ Liniensegmente
ohne	54:05	/	/	/
VAH	0:28	28,6	664.899	9
iZH	0:23	19,8	427.279	10
ZH	0:21	17,9	406.276	10

hierdurch enger dem Skelettverlauf und weniger Dreiecke werden konstruiert. Der Speicheraufwand für die *map* ist moderat und für heutige Rechnerarchitekturen leicht zu bewältigen. An dieser Stelle sei jedoch erwähnt, dass dieser Speicher lediglich die theoretisch notwendige Kapazität zur Speicherung der Voxel und Liniensegmente beschreibt. Weitere Datenstrukturen sind notwendig, um die Zugriffszeiten in $O(\log n)$ zu gewährleisten. Abhängig von dem verfügbaren Speicher werden diese Strukturen automatisch angelegt und genutzt. Die theoretisch notwendige Kapazität würde jedoch im Zweifelsfall ausreichen, wenn auch auf Kosten von erheblichen Performanzeinbußen. Die durchschnittliche Anzahl der Voxel in einer *map* macht deutlich, dass eine statische Datenstruktur mit einem Eintrag für jeden Voxel des Volumendatensatzes sehr ineffizient wäre. Im Durchschnitt sind bei diesen drei Beispielbäumen nur etwa 1% der Voxel enthalten. Die durchschnittliche Anzahl der Liniensegmente pro Voxel ist ebenfalls gering.

5.3 Generierung eines geometrischen Modells

Während der Polygonisierung der impliziten Beschreibung des Gefäßbaums muss die Faltung eines Liniensegments mit dem Gauß-Filter berechnet werden. Eine effiziente Lösung nach [Blo95b] wird in Abschnitt 5.3.1 diskutiert. Für die Visualisierung der im Rahmen dieser Arbeit betrachteten Gefäßstrukturen wurde der *Implicit Polygonizer* [Blo94] verwendet. Dieser ist im Original in der Programmiersprache C realisiert und steht kostenlos im Internet zur Verfügung [3]. Mittlerweile existiert auch eine objektorientierte Version in C++ [4], welche jedoch zu Beginn dieser Arbeit noch nicht verfügbar war. Daher wurde der *Implicit Polygonizer* eigens in eine C++ Klasse `ImplicitPolygonizer` konvertiert. Die Grundstruktur des Algorithmus wurde hierbei nicht verändert. Lediglich die einzelnen Funktionen wurden in Methoden der Klasse transformiert. Weiterhin sind die Datenstrukturen zur Speicherung der Oberflächenpunkte und -normalen durch Strukturen der *Standard Template Library* ersetzt worden. Die Polygonisierung mit Hilfe der Datenstruktur aus Abschnitt 5.2.1, das Datenformat des Polygonisierungsresultats,

[3] `http://www.unchainedgeometry.com/jbloom/misc/polygonizer.c` , Stand: 5.02.2004
[4] `http://www.unchainedgeometry.com/jbloom/papers.html`, Stand: 5.02.04

sowie die Vorbereitung einer kantenbasierten Interaktion werden in Abschnitt 5.3.2 beschrieben. In Abschnitt 5.3.3 ist die finale Visualisierung der Gefäßstruktur mit Hilfe von OpenInventor skizziert. Abschnitt 5.3.3 zeigt die Auswirkungen der Variation verschiedener Parameter des *Implicit Polygonizers*.

5.3.1 Filterberechnung

Die Berechnung der Faltung eines Liniensegments mit einem Gauß-Filter kann nach [Blo95b] reduziert werden auf die Berechnung eines eindimensionalen Integrals und eine einfache Evaluierung des Gauß-Filters (3.17). Wie jedoch schon in Abschnitt 3.3.1 bemerkt, resultiert die Integration der Gauß-Funktion über ein Intervall $[a, b]$ nicht in einer abgeschlossenen Lösung. Die Lösung enthält zweimal die Fehlerfunktion $erf()$, welche nur über Reihen angenähert werden kann. Um diese aufwändige Rechenoperation zu vermeiden, schlägt [Blo95b] daher die Verwendung einer im voraus berechneten look up-Tabelle *look-up table* für die Berechnung des Integrals vor. Die Tabelle deckt den signifikanten Trägerintervall des Filters ab und kann mit Hilfe der Trapezregel erstellt werden. Die Trapezregel beschreibt ein mathematisches Verfahren, zur Berechnung der Fläche unter einer Kurve $f(x)$ im Intervall $[a, b]$. Die Fläche wird hierbei durch die Zerstückelung in mehrere kleine Trapeze approximiert. [Blo95b] empfiehlt die Erfassung der Werte im Bereich [0..10] in einem Feld der Größe 10.000. Sowohl die Generierung der Tabelle, als auch der Zugriff während der Polygonisierung sind im Anhang von [Blo95b] beschrieben und hier entsprechend implementiert worden.
Die Nachschlagetabelle ist als eindimensionales Feld (*array*) vom Datentyp *float* realisiert und global deklariert. Somit können mehrere Instanzen des `Implicit Polygonizer` auf dieselbe Tabelle zugreifen.

5.3.2 Polygonisierung und Vorbereitung einer kantenbasierten Interaktion

Während des Polygonisierungsschrittes wird ein polygonales Modell der Gefäßstruktur bestehend aus Dreiecken konstruiert. Jedes Dreieck ist durch drei Indizes in die Felder der Oberflächenkoordinaten -und normalen beschrieben. Neben diesen Informationen sollte für jeden Dreieckspunkt eine Farbe und ein Transparenzwert definiert werden können (siehe Abschnitt 4.5). Weiterhin muss in Vorbereitung einer kantenbasierten Interaktion für jedes Dreieck die Zugehörigkeit zu einer Kante e des Gefäßgraphen geklärt werden. Eine separate Indexliste für jede Kante reflektiert diese Korrespondenzbeziehung. Die Beziehung ist im Hinblick auf eine korrekte Einfärbung des Gefäßgraphen noch zu verfeinern. Da die Farbinformation innerhalb der Skelettvoxel von e variieren kann, muss das die Farbgebung eines Dreieckspunktes bestimmende Voxel von e eruiert werden. Folgende Datenstrukturen wurden hierfür mit Hinblick auf eine spätere Visualisierung der Gefäßstruktur mittels OpenInventor definiert:

- *vector* `Koordinaten`: globales Feld mit den Koordinaten aller Oberflächenpunkte; Form eines Eintrags: dreielementiger Vektor (*SbVec3f<x,y,z>*)
- *vector* `Normalen`: globales Feld mit einer Normalen je Oberflächenpunkt; Form eines Eintrags: dreielementiger Vektor (*SbVec3f<x,y,z>*)
- *vector* `Farben`: globales Feld mit einer Farbinformation je Oberflächenpunkt; Form eines Eintrags: RGB-Farbvektor (*SbColor<R,G,B>*)

- *vector* `Transparenzen`: globales Feld mit einer Transparenzinformation je Oberflächenpunkt; Form eines Eintrags: *float*-Wert zwischen 0.0 und 1.0
- *vector* `Korrespondenzen`: globales Feld mit einer Struktur je Oberflächenpunkt, welche die korrespondierende Kante e des Gefäßgraphen und das die Farbinformation des Punktes beeinflussende Skelettvoxel v von e speichert; Form eines Eintrags: `Kante_Voxel<e,v>`
- *vector* `Indizes`: lokales Feld pro Kante mit Indizes in die globalen Datenstrukturen; Form eines Eintrags: *Integer*-Wert zwischen 0 und Anzahl der Oberflächenpunkte minus 1

Die Auffüllung der obigen Datenstrukturen geschieht während der Berechnung der Feldfunktion an einem Punkt p. Der Algorithmus hierzu ist im Folgenden skizziert:

```
computeImplicitFunction(x,y,z) {                                          1
                                                                          2
 Vox_ID = calculateVoxID( transformToVoxelSpace(x, y, z) );               3
 ptr_infl = influences.find(Vox_ID);                                      4
                                                                          5
 float seg_convolutionValue, convolutionValue, convolutionValue_max = 0;  6
 Kante_Voxel _newCorrEdgeVox;                                             7
                                                                          8
 for (ptr_segs = ptr_infl->second.begin();                                9
      ptr_segs != ptr_infl->second.end();                                 10
      ++ptr_segs) {                                                       11
                                                                          12
   Kante edge = SoVesselParser->getEdge(ptr_segs->e);                     13
   Segment seg = constructSegment(edge, ptr_segs->v);                     14
                                                                          15
   seg_convolutionValue = convolveSegment(seg, x,y,z);                    16
   if (seg_convolutionValue > convolutionValue_max)    {                  17
     convolutionValue_max = seg_convolutionValue;                         18
     _newCorrEdgeVox = *ptr_segs;                                         19
   }                                                                      20
   convolutionValue = convolutionValue + seg_convolutionValue;            21
 }                                                                        22
                                                                          23
 return (convolutionValue);                                               24
}                                                                         25
```

Die Polygonisierung erfolgt innerhalb des WKS. Der aktuelle Punkt p wird in der Form (x,y,z) an die Funktion (*computeImplicitFunction*) übergeben (Zeile 1). In Zeile (3) wird p in das VKS überführt (*transformToVoxelSpace*) und aus den ganzzahligen Koordinaten des getroffenen Voxels wird dessen Identifikation (*Vox_ID*) berechnet (*calculateVoxID*). Zeile (4) führt eine Anfrage an die *map* durch und gibt einen Zeiger auf die zu *Vox_ID* korrespondierenden Daten bzw. Liniensegmente zurück. Die in Zeile (6) erstellte Variable *seg_convolutionValue* speichert das Ergebnis der Faltung eines Liniensegments. *convolutionValue* hingegen speichert die Summe aller berechneten Werte von *seg_convolutionValue* und beschreibt somit den Wert der Feldfunktion an der Stelle p. Hier wird die Superpositions-Eigenschaft der Faltung ausgenutzt. Die Variable *convolutionValue_max* beschreibt den maximalen Wert von *seg_convolutionValue*. Die Variable *_newCorrEdgeVox* in Zeile (7) ist global deklariert und speichert das Liniensegment mit dem größten Beitrag zum finalen Wert der Feldfunktion an p im Format `Kante_Voxel`. In der *for*-Schleife (Zeile 9-22) wird über alle Liniensegmente welche aus der Anfrage an die *map* resultierten iteriert. Die Liniensegmente liegen im Format `Kante_Voxel<e,v>` vor. Dieses Tupel wird in den

Zeilen (13-14) in die Struktur `Segment` überführt. Die Faltung des aktuellen Segments *seg* findet in Zeile (16) statt und ist entsprechend des Anhangs von [Blo95b] implementiert. Hier wird die in Abschnitt 5.3.1 beschriebene look up-Tabelle zur Berechnung des Faltungsintegrals genutzt. Innerhalb des *if*-Konstrukts in Zeile (17-20) wird *_newCorrEdgeVox* aktualisiert. Die Summe der Faltungen aller Liniensegmente an der Stelle p wird durch Zeile (21) realisiert. Die Rückgabe des finalen Werts der Feldfunktion erfolgt in Zeile (24).

Die Berechnung der Feldfunktion wird mehrere Male innerhalb des *binary subdivison*-Prozesses ausgeführt. Terminiert dieser steht ein neuer Oberflächenpunkt p_O fest und Einträge in die obigen Datenstrukturen können generiert werden. p_O wird in `Koordinaten` eingefügt. Danach wird die korrespondierende Normale durch Approximation des Oberflächengradienten nach [Blo88] bestimmt und in `Normalen` eingetragen. Da die Gefäßstruktur initial einfarbig dargestellt werden soll, wird das entsprechende RGB-Tripel in `Farben` eingetragen. Die Transparenz wird zu Beginn für jeden Oberflächenpunkt auf 0.0 (opak) gesetzt und in `Transparenzen` eingefügt.

Zur Bestimmung der Korrespondenz zwischen p_O und einer Kante e (im Hinblick auf die kantenbasierte Interaktion) bzw. zwischen p_O und einem Voxel von e (zur Festlegung der lokalen Farbinformation bei einer Einfärbung der Gefäßstruktur) wird die Variable *_newCorrEdgeVox* genutzt. Diese kodiert das Liniensegment L_{max} mit dem höchsten Beitrag an p_O. Da beide Endpunkte von L_{max} zu derselben Kante gehören steht e bereits fest und ist in *_newCorrEdgeVox* gespeichert. Zu klären ist nun, welcher Endpunkt des Segments, d.h. welcher der zugehörigen Voxel, die Farbinformation bestimmt. Diese Entscheidung kann basierend auf dem Minimum der euklidischen Distanz zwischen p_O und den beiden Endpunkten bzw. den zugehörigen Voxelmittelpunkten getroffen werden. Die Variable *_newCorrEdgeVox* wird ggf. entsprechend des Endpunkts mit minimaler Distanz aktualisiert und in `Korrespondenzen` eingefügt. Dieser *vector* wird später bei dem Umschalten zwischen verschiedenen Einfärbemöglichkeiten der Gefäßstruktur genutzt.

Durch den *Implicit Polygonizer* wird für p_O neben den Koordinaten ein Index berechnet. Dieser dient dem Zugriff auf die bisher beschriebenen Datenstrukturen. Nachdem alle Punkte p_O eines Dreiecks berechnet wurden, sind somit auch deren drei Indizes bekannt. Mit Hilfe dieser Information werden aus `Korrespondenzen` die drei zugehörigen Einträge der Form `Kante_Voxel` extrahiert. Sind alle drei Einträge gleich, so wird das Dreieck dieser Kante zugewiesen, d.h. die drei Indizes werden in den *vector* `Indizes` der Kante eingefügt. Sind nur zwei Einträge gleich, ist die Vorgehensweise analog. Wenn alle Einträge verschieden sind, kann die Zuweisung beliebig erfolgen. Die Aufteilung aller Dreiecke des polygonalen Modells einer Gefäßstruktur auf die korrespondierenden Kanten des Gefäßgraphen ist in Abb. 5.5 dargestellt.

5.3.3 Darstellung des polygonalen Modells mit OpenInventor

Für die finale Darstellung der Gefäßstruktur mit OpenInventor muss die in der Polygonisierung gewonnene Information auf entsprechende Knoten abgebildet werden. Zur Illustration der folgende Ausführungen ist in Abb. 5.6 der Szenengraph für einen Gefäßbaum dargestellt. Enthält ein Datensatz mehrere Gefäßbäume, so wird für jeden eine solche Struktur angelegt.

Der in Abb. 5.6 grau markierte Teil beschreibt das Gefäßskelett. Der Nutzen einer zusätzlichen Darstellung des Skeletts wurde in Abschn. 4.5.3 hervorgehoben. Das Skelett ist durch Liniensegmente approximiert und wird mit Hilfe von Polylinien dargestellt. Knoten (3) ermöglicht die

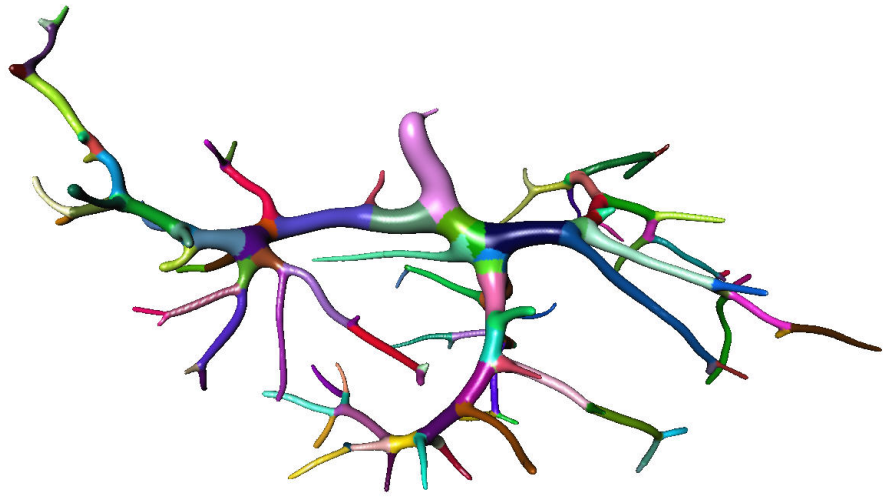


Abbildung 5.5: Einfärbung eines Gefäßbaums nach der Zugehörigkeit von Dreiecken des polygonalen Modells zu den Kanten des Gefäßgraphen.

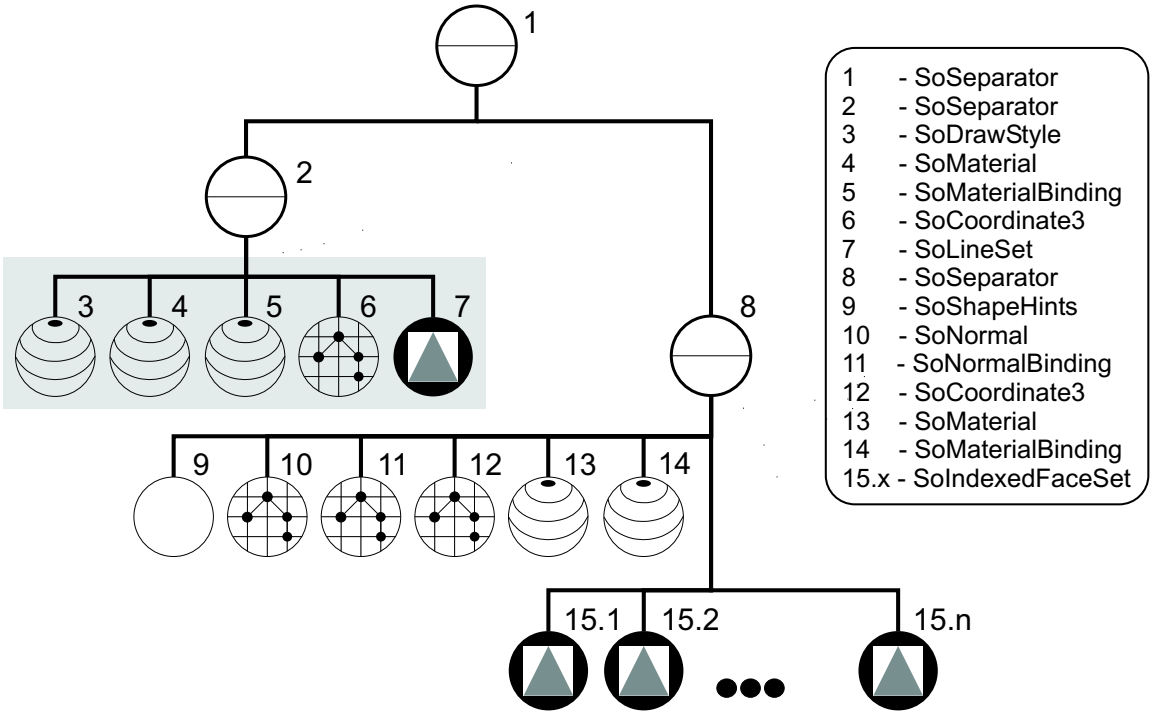


Abbildung 5.6: Der Szenengraph repräsentiert die interne Organisation des polygonalen Modells einen Gefäßbaums unter OpenInventor. Der grau markierte Teil beschreibt das Gefäßskelett, während die restlichen Elemente die Gefäßoberfläche repräsentieren.

Einstellung der Dicke dieser Linien. Knoten (4) spezifiziert eine Farbe für das Skelett (typischerweise schwarz) und Knoten (5) regelt die Zuweisung der Farbe zu den Polylinien. Letzterer gewinnt jedoch erst dann an Bedeutung, wenn die Polylinien des Skeletts unterschiedlich eingefärbt werden. Die Koordinaten der Skelettpunkte werden durch Knoten (6) beschrieben. Der sie repräsentierende *vector* wurde als Nebenprodukt bei der Berechnung der Hüllkörper angelegt. Hier musste ohnehin über alle Liniensegmente iteriert werden (siehe Abschnitt 5.2.1.3). Knoten (7) beschreibt das eigentliche geometrische Primitiv.

Der restliche Teil des Szenengraphen verkörpert die Gefäßoberfläche. Separator (8) sorgt dafür, dass Eigenschaften des Gefäßskeletts nicht die Gefäßoberfläche beeinflussen. Durch Knoten 9 können einige Eigenschaften der polygonalen Struktur beschrieben werden, um die Performanz des Darstellungsprozesses zu beeinflussen. So kann z.B. erreicht werden, dass nicht sichtbare Oberflächenteile nicht gezeichnet werden (*backface culling*) oder dass der innere Teil einer geschlossenen Oberfläche nicht beleuchtet wird (kein *two-sided lighting*). Die restlichen Knoten repräsentieren die in Abschn. 5.3.2 gewonnene Information. Knoten (10) beinhaltet die Daten aus dem *vector* `Normalen`. Knoten (11) regelt die Zuweisung der Normalen an die Oberfläche. In diesem Fall ist spezifiziert, dass zu jedem Oberflächenpunkt eine Normale korrespondiert. In Knoten (12) werden die Koordinaten der Oberflächenpunkte aus `Koordinaten` gespeichert. Knoten (13) und (14) beschreiben die Materialeigenschaften der Gefäßoberfläche. Knoten (13) enthält die Informationen aus `Farben` und `Transparenzen`, während Knoten (14) auf gleich Weise wie Knoten 11 eine Zuweisung regelt. Die verbliebenen Knoten (15.1 bis 15.n) repräsentieren jeweils die zu einer Kante des Gefäßgraphen gehörende Geometrie bestehend aus einzelnen Dreiecken. Jeder Knoten kennt die Information aus dem *vector* `Indizes` der jeweiligen Kante und ist somit in der Lage über die Indizes auf die Knoten (10), (12) und (13) zuzugreifen.

Durch den Anstoß eines Rendering-Prozesses unter OpenInventor wird der Szenengraph traversiert und der Gefäßbaum dargestellt.

5.4 Interaktion

Die Interaktion mit der visualisierten Gefäßstruktur wurde innerhalb dieser Arbeit nur am Rand behandelt. Wichtige Schritte in Richtung einer kantenbasierten Interaktion wie in Abschnitt 4.5 erläutert sind jedoch bereits durchgeführt worden. So konnte eine Korrespondenz zwischen den Kanten des Gefäßgraphen und den Dreiecken der Gefäßoberfläche etabliert werden. Dies kann für eine sehr einfache Interaktion unter OpenInventor bereits ausgenutzt werden. Der zum Umfang dieser Graphikbibliothek gehörende Knoten *SoExtSelection* erlaubt die Selektion eines Bildausschnitts der gerenderten Darstellung (Abb. 5.7 links). Alle Graphikprimitive innerhalb dieses Ausschnitts werden erfasst und sind in einer Liste als *SoIndexedFaceSet* verfügbar. Jeder dort gespeicherte Knoten hält wiederum eine Liste von Indizes, welche die Dreiecke des korrespondierenden Oberflächenanteils repräsentieren. Mit Hilfe dieser Indizes kann u.a. auf die global deklarierten Materialeigenschaften (*SoMaterial*) zugegriffen werden. Somit sind Farbe und Transparenz des ausgewählten Bereichs veränderbar (Abb. 5.7 rechts).

Eine weitere einfache Form der Interaktion ist die Einfärbung der Gefäßstruktur entsprechend von Farbinformationen aus der Gefäßanalyse (siehe Anhang A.1). Nach der initialen Darstellung der Gefäße in einer einheitlichen Farbe kann der Benutzer zwischen vier Einfärbungen wählen. Es müssen jedoch nicht immer alle Einfärbungen deklariert sein. So kann bei der Gefäßanalyse

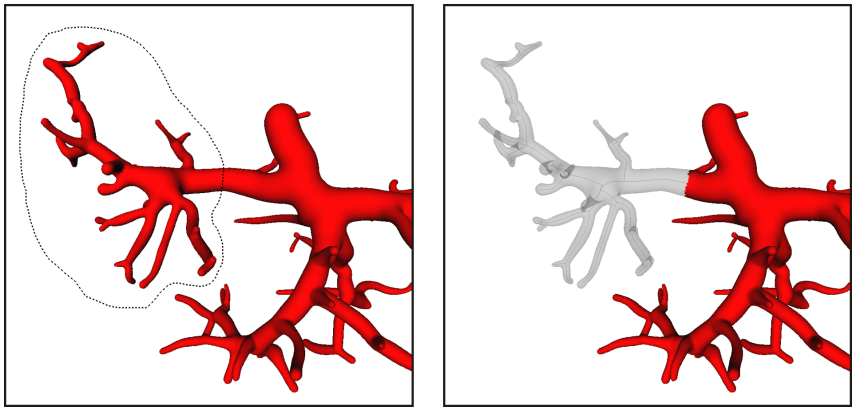


Abbildung 5.7: Zuerst wird der gewünschte Bereich mit der Maus umfahren (*links*). Dann werden die Materialeigenschaften des selektierten Bereichs verändert. Hier wurde die Einfärbung in grau festgesetzt. Durch eine zusätzliche semi-transparente Darstellung wird das Gefäßskelett sichtbar (*rechts*).

lediglich die Definition von maximal großen Teilbäumen (Farbgebung *maxtrees*) von Interesse gewesen sein.

Schaltet der Benutzer zwischen den Farbgebungen um, wird für jeden Dreieckspunkt mit Hilfe der Datenstrukturen `Indizes` und `Korrespondenzen` das die Farbgebung beeinflussende Skelettvoxel und die korrespondierende Kante bestimmt. Aus der gewählten Farbgebung wird dann ermittelt, auf welche Farbinformation des Voxels (siehe Anhang A.1, die Information *Labs* einer Kante) zurückgegriffen wird. Abb. 5.8 zeigt die Einfärbung Gefäßbaums hinsichtlich des Kriteriums *maxtrees*.

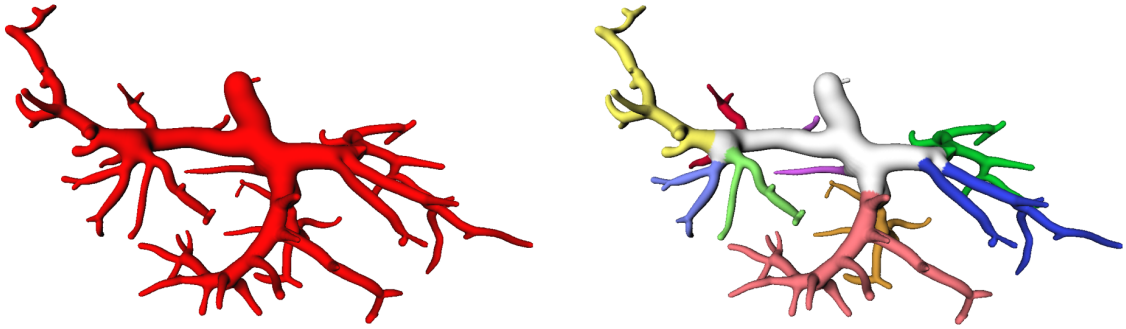


Abbildung 5.8: Nach der initialen Darstellung des Gefäßbaums in einer einheitlichen Farbe (*links*) kann der Benutzer zwischen vier Farbgebungen aus der Gefäßanalyse umschalten (*rechts*). In der rechten Darstellung sind die maximalen Teilbäume des Gefäßbaums unterschiedlich gefärbt.

5.5 Resultate

Dieser Abschnitt zeigt einige Resultate der Gefäßvisualisierung mit *Convolution Surfaces*. Die hier ausgewählten Gefäßstrukturen sind repräsentativ in zweierlei Hinsicht. Zum einen spiegeln

sie die sehr variable Komplexität der innerhalb dieser Arbeit verwendeten Modelle wieder. Zum anderen bilden sie einen Querschnitt durch die unterschiedlichen visualisierten Gefäßtypen. So sind Blutgefäßsysteme (Abb. 5.9 links und 5.10 rechts), zerebrale Gefäße (Abb. 5.9 rechts) und Bronchialgefäße (Abb. 5.10 links) dargestellt. Weiterhin fasst Tabelle 5.4 wichtige Maße der polygonalen Modelle zusammen, um die in dieser Arbeit erreichte Performanz der Visualisierung mit *Convolution Surfaces* zu verdeutlichen. Die Spalte Rechenzeit beinhaltet hier jeweils den Zeitraum vom Einlesen der Daten bis zu einer initialen Darstellung auf dem Bildschirm. Der Speicherbedarf bezieht sich ausschließlich auf die Datenstruktur *map* aus der Vorverarbeitung und die für Polygonisierung und Interaktion in Abschnitt 5.3.2 beschriebenen Strukturen.

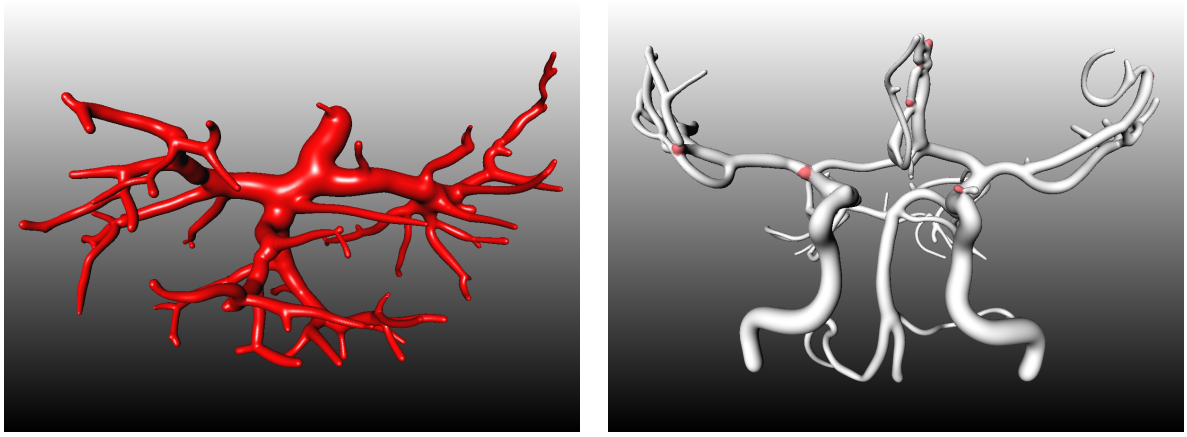


Abbildung 5.9: *links*: Arteriensystem einer menschlichen Leber. Glanzlichter wurden gesetzt, um die Glattheit der Oberfläche zu betonen. *rechts*: Zerebrales Gefäßsystem eines menschlichen Gehirns. Die farbigen Hervorhebungen markieren Aneurysmen.

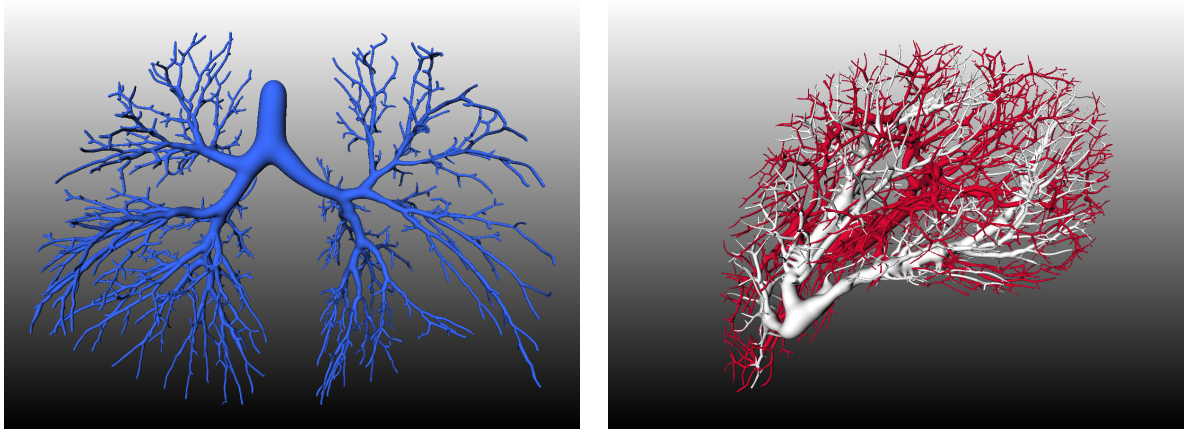


Abbildung 5.10: *links*: Ausgusspräparat der Bronchialgefäße einer menschlichen Lunge. *rechts*: Ausgusspräparat einer Schweineleber. Pfortader (rot) und Leberarterien (weiß) sind in der Gefäßanalyse getrennt identifiziert und eingefärbt worden.

Die Tabelle zeigt, dass bei den komplexen Gefäßstrukturen (Zeile 3 und 4) mit einer Vielzahl von Dreiecken zu rechnen ist. Die hier verwendeten Ausgusspräparate bilden allerdings momentan die Spitze dieses Eisberges und stellen eine Ausnahme dar. Weiterhin kann mit herkömmlichen Polygonreduktionstechniken in einem Nachbearbeitungsschritt die Dreieckszahl reduziert werden. Eine elegantere Lösung ist die initiale Generierung von weniger Dreiecken. Adaptive Ansätze bei

Tabelle 5.4: Performanzmaße für unterschiedlicher anatomische Gefäßstrukturen. Verwendete Hardware: Intel Pentium 4 CPU 3.06GHz, 1024MB RAM, ATI Radeon 9600.

Gefäßbaum	Kanten	Dreiecke	Rechenzeit (s)	Speicher (MB)	fps
Leberarterien	136	124.884	7,5	14,9	64
zerebrale Gefäße	149	252.836	9,8	18,2	42,8
Bronchialbaum	1504	1.125.540	65,4	101,1	14,6
Schweineleber	3461	2.366.008	65	97,6	1,8

der Polygonisierung sollten sich Gefäßdicke und -krümmung anpassen. Einige Beispiele solcher Optimierungen sind in Abschnitt 3.4.2 schon erwähnt worden.

Die Rechenzeit konnte im Vergleich zu den ersten Visualisierungstests (siehe Abschnitt 4.1) durch die Verwendung von Hüllkörpern drastisch reduziert werden. Sie ist jedoch naturgemäß stark von der verwendeten Auflösung bei einer Polygonisierung abhängig. Obwohl mittlerweile Zeiten im Sekundenbereich erzielt werden können sind weitere Optimierungen wünschenswert. Die ähnlichen Rechenzeiten in Zeile 3 und 4 trotz erheblich abweichender Dreieckszahl sind bedingt durch die Größe *map*. Der Durchmesser von Gefäßen des Bronchialbaums ist im Durchschnitt größer als der des Ausgusspräparats der Schweineleber. Dies erfordert größere Hüllkörper und führt somit zu einer komplexeren *map*. Der Speicherbedarf ist annähernd gleich, da mehr Dreiecke und korrespondierende Information für die Visualisierung des Ausgusspräparats der Schweineleber gespeichert werden müssen.

Der Speicherbedarf ist mit den heutigen Rechnerarchitekturen und einem Arbeitsspeicher moderater Größe zu bewältigen. Die hier gemessenen Werte spiegeln den theoretisch benötigten Speicherplatz wieder. Speziell die Datenstruktur *map* aus der Vorverarbeitung und auch die Datenstruktur *vector* generieren jedoch zusätzliche Strukturen, die einen schnellen Zugriff gestatten sollen. Deren konkreter Speicherbedarf ist allerdings schwer zu bestimmen, da die zugrundeliegenden Implementierungen nicht zugänglich sind.

Die Visualisierung mit *Convolution Surfaces* erfüllt alle in der Einleitung definierten Anforderungen an eine ideale Gefäßvisualisierungsmethode:

- korrekte Abbildung des Gefäßdurchmessers
- weiche, organische Gefäßform
- einheitliche Behandlung aller Verzweigungstypen
- geschlossene Gefäßenden
- Vermeidung von Strukturen im Gefäßinneren

Wie die Untersuchungen bezüglich der Blendingstärke in Abschnitt 4.2.1 gezeigt haben, folgt die *Convolution Surface* mit abnehmender Filterbreite enger dem vorgegebenen Durchmesserverlauf. Zusätzlich wird das Problem von irritierenden Verdickungen an Verzweigungen vermieden. Die Validierung des Durchmesserverlaufs erfolgte innerhalb dieses Abschnitts rein visuell mit Hilfe von Referenzmodellen. Differenzierte Bewertungsmethoden sind notwendig, um eine Fehleranalyse durchführen zu können.

Die Generierung weicher, organischer Formen ist eine Stärke der Modellierung mit impliziten Oberflächen. Die in diesem Abschnitt gezeigten Gefäßmodelle zeigen eine glatte Oberfläche ohne Diskontinuitäten der Oberflächennormalen. Mittels *blending* werden geometrisch kontinuierliche Übergänge zwischen einzelnen Abschnitten des Gefäßbaums erreicht (Abb. 5.11 rechts). Dies alles ist ohne jeglichen Konstruktionsaufwand möglich. Lediglich die Transformation der impliziten Beschreibung der Gefäßstruktur in ein polygonales Modell ist notwendig.

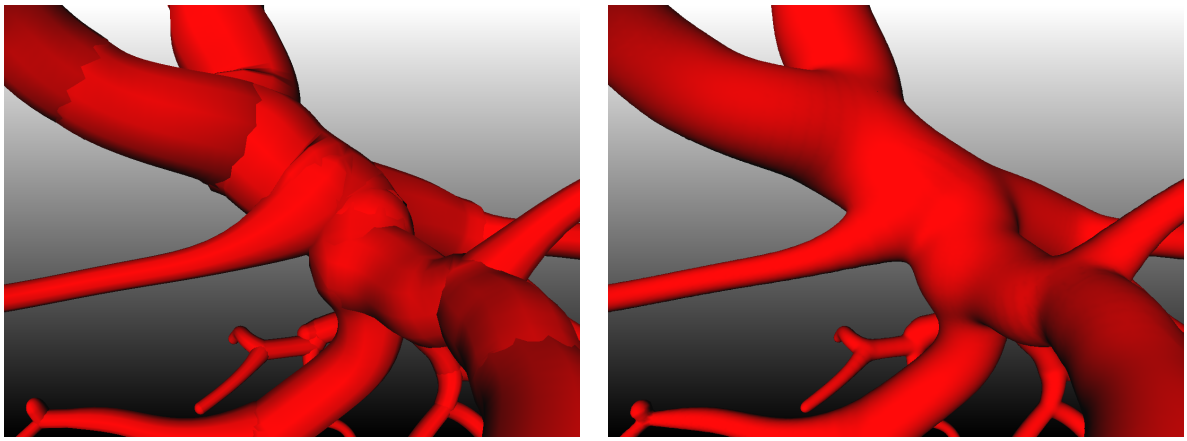


Abbildung 5.11: *links*: Verzweigung modelliert durch Konkatenation von Kegelstümpfen nach [HPSP01]. Störende Diskontinuitäten sind an den Übergängen zu beobachten. *rechts*: Die *Convolution Surface* ist glatt und artefaktfrei.

Da es die skelettbasierte Modellierung mit *Convolution Surfaces* gestattet, jedes Skelettprimitiv unabhängig vom Rest der Struktur zu behandeln, spielt der Verzweigungstyp keine Rolle. Nicht einmal die Unterscheidung zwischen Verzweigungen und einfachen Übergängen an Gefäßabschnitten ist notwendig.

Die Forderungen geschlossener Gefäßenden und der Vermeidung von Strukturen im Gefäßinneren werden ebenfalls ohne einen Konstruktionsaufwand erfüllt (Abb. 5.12). Beides sind inhärente Eigenschaften von *Convolution Surfaces* geometrischer Primitive.

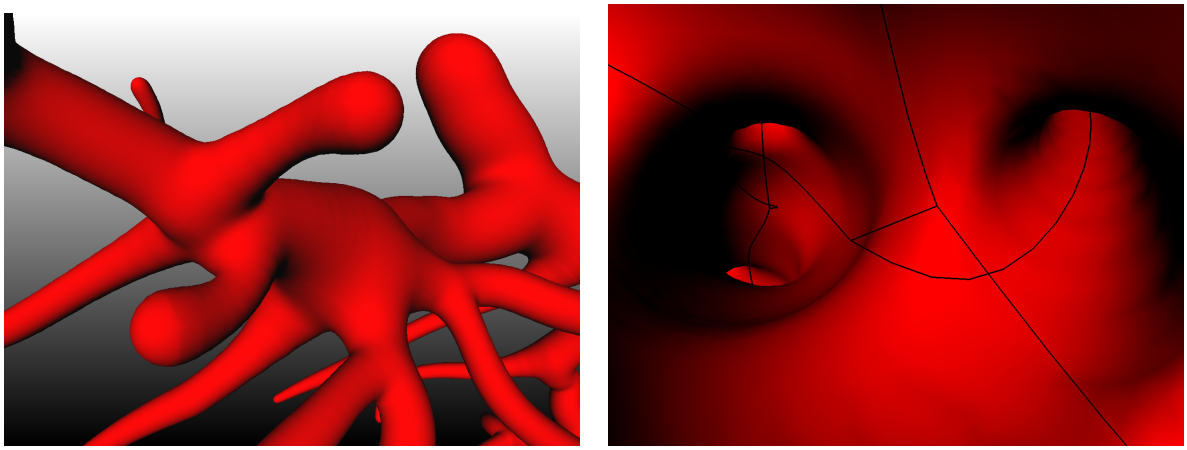


Abbildung 5.12: *links*: Die Gefäßenden werden automatisch geschlossen. *rechts*: Ein Blick in das Innere einer Gefäßstruktur zeigt dass keine störenden Strukturen konstruiert werden. Das Gefäßskelett wurde hier zur besseren Orientierung eingeblendet.

Die Bestimmung der geometrischen Kontinuität von *Convolution Surfaces* ist in der Literatur bisher kaum diskutiert worden. So widmet sich lediglich eine Randnotiz in [Blo95b] diesem Thema. Hier wird spekuliert, dass die geometrische Kontinuität der Oberfläche G^{s+k} abhängig ist von der parametrischen Kontinuität der Filterfunktion C^k und der geometrischen Kontinuität des Skeletts G^s. Das hier verwendete Skelett bestehend aus Liniensegmenten ist G^0-stetig. Die geometrische Kontinuität der *Convolution Surface* eines solchen Skeletts ist ausschließlich durch die parametrische Kontinuität der Filterfunktion bestimmt. Innerhalb dieser Arbeit wurde sich für die Verwendung der Gauß-Funktion entschieden. Da die e-Funktion unendlich oft differenzierbar ist, wird theoretisch geometrische Kontinuität unendlich hohen Grades erreicht. Diese wird allerdings naturgemäß durch die Überführung der impliziten Beschreibung in ein Dreiecksnetz mittels Polygonisierung nicht bewahrt. Dennoch zeigt die resultierende Oberfläche keinerlei Artefakte und ist von hervorragender visueller Qualität.

6 Validierung und Evaluierung

6.1 Validierung

Die Einschätzung der Korrektheit eines Visualisierungsresultats ist besonders für die spätere Verwendung in klinischen Applikationen von großer Wichtigkeit. Eine Visualisierungsmethode ist nicht anwendbar, wenn sie die gegebene Information verfälscht abbildet. Der begrenzte Zeitrahmen dieser Arbeit hat es leider nicht gestattet, differenzierte Validierungsmethoden für die Visualisierung mit *Convolution Surfaces* zu entwickeln und durchzuführen. Die Anwendung zweier rein visuellen Bewertungen sowie einer quantitativen Einschätzung soll jedoch nicht unerwähnt bleiben.

Das Visualisierungsresultat kann nur so genau sein wie die in der Gefäßanalyse gewonnene Information. Daher wurde das rekonstruierte Gefäßmodell mit dem Segmentierungsergebnis verglichen. Letzteres wurde mittels *Surface-Rendering* unter ILAB dargestellt und mit der *Convolution Surface* überlagert (Abb. 6.1). Bei dieser Vorgehensweise muss allerdings vorausgesetzt werden können, dass die Skelettierung des Segmentierungsresultats fehlerfrei verlief. Weiterhin sind die in Abschnitt 2.3.2.2 erwähnten Nachbearbeitungsschritte zu berücksichtigen, da diese das Gefäßskelett z.B. durch Glättungsoperationen, nachträglich verändert haben. Grobe Abweichungen in der Gefäßtopologie hätten jedoch auf diese Weise erkannt werden können. Dies war nicht der Fall. Der Durchmesser der *Convolution Surface* ist generell etwas kleiner als der Durchmesser der *Isosurface*. Dies ist begründet durch die Verwendung des minimalen Gefäßradius (siehe Anhang A.1, Feld *minDist* einer Kante) bei der impliziten Beschreibung des Gefäßbaums. Dieser repräsentiert an jedem Skelettpunkt den größtmöglichen Innkreis der Gefäßoberfläche. Der maximale Gefäßradius (Feld *maxDist*) wurde zwar ebenfalls bestimmt, ist jedoch aufgrund eines noch instabilen Berechnungsverfahrens nicht zuverlässig.

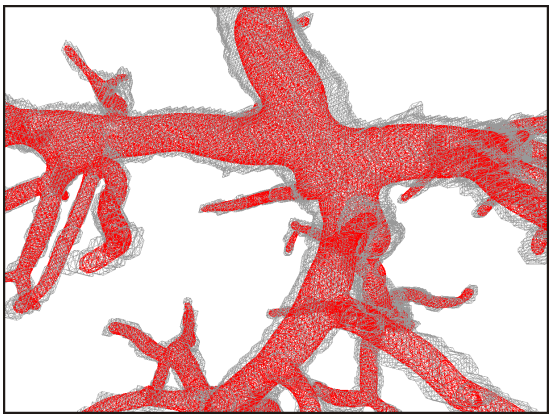


Abbildung 6.1: Die *Convolution Surface* (rot) wurde mit einem *Surface Rendering* des Segmentierungsresultats (grau) aus der Gefäßanalyse überlagert.

Ein weiterer visueller Vergleich wurde durch eine gekoppelte Darstellung von implizit beschriebener Gefäßoberfläche mittels *Convolution Surfaces* und der Konstruktion mit parametrischen Methoden nach [HPSP01] arrangiert. Die Methodik wurde bereits in Abschnitt 4.2.1 skizziert. Dieser Vergleich würde im speziellen Abweichungen des lokalen Durchmesserverlaufs entlang von Gefäßästen enthüllen. Nur an den Übergängen zwischen einzelnen Gefäßteilen waren signifikante Abweichungen zu beobachten (Abb. 4.13). Dies spiegelt jedoch eher eine Stärke der *Convolution Surfaces* wieder, nämlich die Fähigkeit der Generierung weicher Übergänge zwischen einzelnen Objekten. Diese entsprechen stärker der anatomischen Realität wie durch die Befragung erfahrener Chirurgen und Radiologen festgestellt werden konnte.

Für einen quantitativen Vergleich von Segmentierungsresultat und *Convolution Surface* wurde letztere mittels eines in ILAB bereits eingebundenen Moduls voxelisiert. Das Gefäßmodell wird hierzu in das Voxelkoordinatensystem des Segmentierungsresultats überführt. Anschließend wird ein Volumendatensatz derselben Dimension erzeugt und jedem Voxel was von dem Modell geschnitten wird ein fixer Wert ungleich 0 zugewiesen (Abb. 6.2 Mitte). Die Überlagerung der beiden Datensätze zeigt in jeder Schicht die Stärke eventueller Abweichungen (Abb. 6.2 rechts). Ein einfaches Maß für diese Abweichung wurde in [ZDMP94] diskutiert:

$$\frac{2 \mid V_i \cap V_j \mid}{\mid V_i \mid + \mid V_j \mid} \tag{6.1}$$

V_i und V_j bezeichnen hier zwei Volumendatensätze. Der so genannte *similarity index* ist ein globales Maß für die Abweichung zweier Volumina bzw. zweier Schichten voneinander. Er gibt keine Auskunft über den lokalen Unterschied zwischen korrespondierenden Strukturen. Ein Wert von 1 bedeutet, dass beide Volumina vollkommen identisch sind. Bevor diese Ähnlichkeitsmaß berechnet werden kann ist jedoch zu berücksichtigen, dass die *Convolution Surface* hohl ist und daher keine Voxel korrespondierend zu dem umschlossenen Volumen generiert werden (Abb. 6.2 Mitte). Dieser Missstand konnte durch die Anwendung eines ebenfalls in ILAB integrierten *region growing*-Operators behoben werden. Exemplarische Tests für einen Datensatz haben gezeigt, dass der *similarity index* pro Schicht zwischen 0.85 und 0.9 variiert. Bei dieser Validierungsmethode sind dieselben Aspekte, wie schon bei dem visuellen Vergleich von *Surface Rendering* und *Convolution Surface* weiter oben erläutert, zu berücksichtigen.

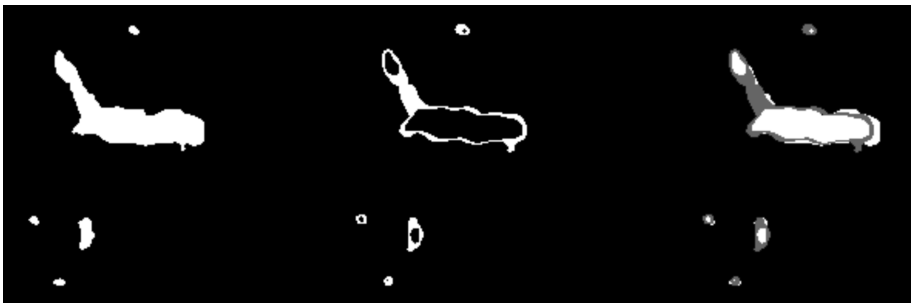


Abbildung 6.2: Einen Schicht durch das Segmentierungsresultat (*links*) und die korrespondierende Schicht durch die voxelisierte *Convolution Surface* (*Mitte*). Die Überlagerung der beiden Schichten (*rechts*) gibt Aufschluss über eventuelle Abweichungen.

6.2 Evaluierung

Um den klinischen Nutzen der im Rahmen dieser Arbeit vorgestellten Gefäßvisualisierung mit *Convolution Surfaces* zu evaluieren, wurden mehrere Visualisierungsergebnisse zusammengestellt und an vier Chirurgen sowie fünf Radiologen zur Bewertung verschickt. Die besondere Qualität der mit impliziten Methoden modellierten Gefäßstrukturen sollte hierbei durch einen Vergleich mit den Ergebnissen zweier weiterer Visualisierungsmethoden hervorgehoben werden. Diese sind das in ILAB integrierte *Surface Rendering* und die dort ebenfalls implementierte Konkatenation von Kegelstümpfen nach [HPSP01] (siehe Abschnitt 2.3.2.2). Während *Surface Rendering* als Repräsentant der nicht-modellbasierten Techniken gewählt wurde, vertritt die Methode nach [HPSP01] die bisherigen modellbasierten Verfahren. Die Vergleichskriterien für eine Evaluierung waren:

1. Übersichtlichkeit
2. Interpretierbarkeit der räumlichen Verhältnisse
3. Ähnlichkeit der Gefäßform im Vergleich zu intraoperativen Ansichten
4. Visuelle Qualität

Die Beurteilung unter Berücksichtigung des dritten Kriterium war optional, da bei einem Radiologen die Kenntnis intraoperativer Gefäßansichten nicht vorausgesetzt werden kann. Für jedes Kriterium konnten Punkte von 1 (ungenügend) bis 5 (sehr gut) vergeben werden. Evaluiert wurden die Darstellungen dreier Gefäßbäume, welche alle auf CT-Daten der Leber basieren.

Für jeden Gefäßbaum wurde zu Beginn eine Überblicksdarstellung im Kontext der Leber angeboten (Abb. 6.3). Diese sollte die räumliche Vorstellung bei darauffolgenden Nahansichten unterstützen. Weiterhin wurden drei Nahansichten präsentiert, welche jeweils mittels der drei oben beschriebenen Visualisierungsmethoden erstellt worden waren. Abb. 6.2 zeigt beispielhaft eine Nahansicht des ersten Lebergefäßbaums. Sowohl der Evaluierungsbogen, als auch die restlichen Überblicksdarstellungen und Nahansichten sind in Anhang A.2 gegeben. Aus Platzgründen wurden hier alle drei Nahansichten und die korrespondierende Kontextdarstellung eines Gefäßbaums auf einer Seite zusammengefasst. In der Evaluierung war dieses Konglomerat, entsprechend Abb. 6.3 und Abb. 6.2, auf vier Blätter verteilt.

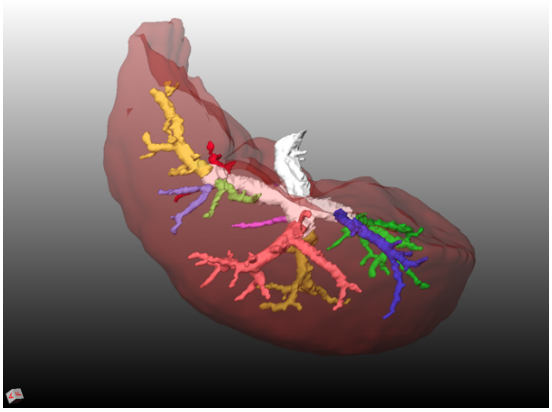


Abbildung 6.3: Die Leberarterie im Kontext der Leber. Das Organ, sowie die Gefäßstruktur sind mittels *Surface Rendering* visualisiert.

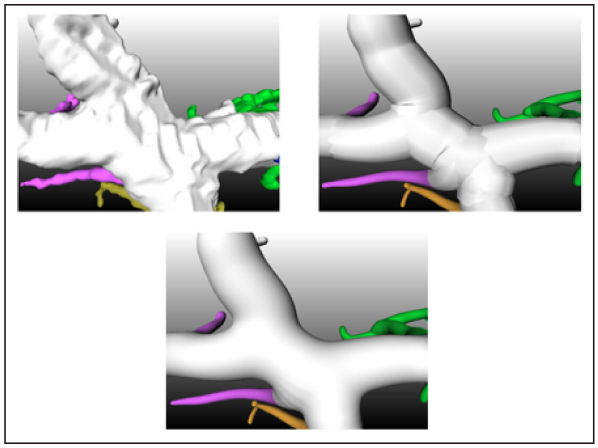


Abbildung 6.4: Vergleich von drei Visualisierungsmethoden anhand einer Nahansicht des Gefäßbaums aus Abb. 6.3. *Surface Rendering* (*oben links*), Konkatenation von Kegelstümpfen (*oben rechts*) und *Convolution Surfaces* (*unten*).

Aufgrund ihren sehr knapp bemessenen Zeit waren bis zum Zeitpunkt der Fertigstellung dieser Arbeit leider nur ein Chirurg und vier Radiologen in der Lage, ihre Rückantwort zu übermitteln. Alle Befragten kannten die Hahn-Methode aus Vorträgen bzw. durch Nutzung der Methode:

- 1 Facharzt für Chirurgie (Spezialisierung Abdominalchirurgie), männlich
- 3 Fachärzte für Radiologie, alle männlich
- 1 Assistenzärztin (Radiologie, 2 Jahre Erfahrung in der Radiologie), weiblich

Die erhaltenen Ergebnisse sind in Tabelle 6.1 zusammengefasst. Zu jedem Paar aus Vergleichskriterium und Visualisierungsmethode wurde jeweils das arithmetische Mittel (∅) der vergebenen Punkte bestimmt. Weitere statistische Maße, wie Mittelwert oder Standardabweichung wurden aufgrund der niedrigen Anzahl von befragten Ärzten nicht erhoben. Stattdessen ist zusätzlich der Intervall ($[min, max]$) der jeweils vergebenen Bewertungen vermerkt. Die Visualisierungsergebnisse wurden hinsichtlich des dritten Kriteriums durch 1 Chirurgen, 2 Radiologen und die Assistenzärztin bewertet.

Tabelle 6.1: Vergleich der Gefäßvisualisierung durch *Convolution Surfaces* mit *Surface Rendering* und der Konkatenation von Kegelstümpfen [HPSP01]. ∅ bezeichnet den Mittelwert der abgegebenen Bewertungen, während $[min, max]$ deren Intervall beschreibt. Es konnten Punkte von 1 (ungenügend) bis 5 (sehr gut vergeben werden).

	Übersichtlichkeit		**Verständnis räumlicher Verhältnisse**		**Ähnlichkeit zu intraop. Ansichten**		**Visuelle Qualität**	
Methode	∅	[min,max]	∅	[min,max]	∅	[min,max]	∅	[min,max]
Isosurface	1,6	[1,3]	1,7	[1,3]	1,1	[1,2]	1,3	[1,2]
Kegelstümpfe	3,6	[1,5]	3,8	[1,5]	3,1	[1,5]	3,6	[1,5]
Convolution Surfaces	4	[1,5]	4	[1,5]	3,9	[1,5]	4,2	[1,5]

Die Ergebnisse der Evaluierung zeigen, dass die Visualisierung mit *Convolution Surfaces* durchweg die besten Resultate erzielt. Der Unterschied der Bewertungen zu der modellbasierten Visualisierung nach [HPSP01] ist nicht signifikant, liefert jedoch Indizien dafür, dass die Visualisierung mit impliziten Methoden in eine vielversprechende Richtung zeigt. Vor allem die Ähnlichkeit der Gefäßform zu intraoperativen Ansichten (Spalte 4) und die visuelle Qualität (Spalte 5) wurden auch durch Kommentare der befragten Ärzte hervorgehoben.

Die Resultate in Tabelle 6.1 unterstützen einen zusätzlich für diese Arbeit wichtigen Aspekt. Die modellbasierte Gefäßvisualisierung (Zeile 3 und 4) ist der direkten Visualisierung durch *Surface Rendering* in allen Punkten klar überlegen. Dies spricht für eine Rekonstruktion des Gefäßmodells aus den segmentierten Daten.

7 Zusammenfassung und Ausblick

Sowohl für die präoperative Eingriffsplanung, als auch für die medizinische Ausbildung ist die Visualisierung von Gefäßbäumen von großem Interesse. Hierbei stehen ein Verständnis der relativen Lage zu krankhaften Veränderungen, Topologie und Morphologie der Strukturen im Vordergrund. Neue klinische Fragestellungen, wie die Vermessung des Durchmesserverlaufs entlang eines Gefäßes, Distanzmessungen und quantitative Vergleiche von mehreren Gefäßbäumen, erfordern eine strukturelle Analyse der Gefäßstruktur [GKS+93]. Traditionelle Gefäßvisualisierungsverfahren sind zur Erfüllung dieser Zielstellungen nicht geeignet. Die hier auftretenden Artefakte, bedingt durch Bildrauschen, Partialvolumeneffekt und der begrenzten Auflösung der Bilddaten, motivieren die Rekonstruktion eines Modells der Gefäßstruktur. Mittels Methoden der Bildanalyse kann eine symbolische Beschreibung der Gefäße gewonnen werden. Diese gestattet eine Rekonstruktion der Gefäßoberfläche.

Die Untersuchung bisher existierenden Techniken zur modellbasierten Gefäßvisualisierung hat gezeigt, dass die Generierung einer weichen, organischen Gefäßform durch die Sicherstellung geometrischer Kontinuität an den Verzweigungen das größte Problem darstellt. Sie ist entweder nicht gewährleistet oder erfordert einen beträchtlichen Konstruktionsaufwand. Verschiedene klinische Aufgaben, wie die Definition eines Sicherheitsrandes bei der Tumorresektion oder die Vermessung einzelner Gefäßäste, erfordern jedoch eine Nahansicht der interessierenden Gefäßteile. Hier sollte der Betrachter nicht durch Diskontinuitäten entlang der Gefäßoberfläche abgelenkt werden.

Innerhalb dieser Arbeit wurde einen Gefäßvisualisierungsmethode basierend auf impliziten Oberflächen, genauer auf *Convolution Surfaces*, entwickelt. *Convolution Surfaces* liegt das Prinzip der Faltung zugrunde. Ein Signal wird hierbei mit Hilfe einer Filterfunktion geglättet. Eine ihrer großen Stärken ist die Generierung von weichen, geometrisch kontinuierlichen Übergängen zwischen zwei Objekten **ohne** jeglichen Konstruktionsaufwand. Weiterhin ist die skelettbasierte Modellierung mittels *Convolution Surfaces* in besonderem Maße für die modellbasierte Gefäßvisualisierung geeignet, da die hier zu visualisierende Gefäßstruktur durch das Gefäßskelett und assoziierte Oberflächeninformation beschrieben ist. Das Skelett entspricht dem Signal bei der Faltung. Weiterhin muss eine geeignete Filterfunktion festgelegt werden. Hier wurde die in [BS91] eingeführte Gauß-Funktion als für die Gefäßvisualisierung günstig eruiert.

Erste Visualisierungstests haben gezeigt, dass die Applikation von *Convolution Surfaces* zwar in einer glatten, organischen Gefäßform resultiert, andererseits jedoch folgende Probleme aufwirft:

- Die teilweise wulstige Gefäßform verhindert die korrekte Abbildung eines gegebenen Durchmesserverlaufs.
- *Bulging* führt zu störenden Verdickungen an Verzweigungen, welche fälschlicherweise als pathologische Gefäßveränderungen interpretiert werden könnten.
- Die Überführung der impliziten Beschreibung der Gefäßstruktur in ein polygonales Modell ist mit einer für den klinischen Betrieb inakzeptablen Rechenzeit verbunden.

Als Ursache für das erste Problem konnte die Blendingstärke, d.h. die Stärke der Verschmelzung von Oberflächenanteilen benachbarter Skelettprimitive, identifiziert werden. Diese ist durch die Breite der Filterfunktion kontrollierbar und wurde im Hinblick auf eine korrekte Durchmesserabbildung reduziert. Da die Blendingstärke und die Stärke des Auftretens von *Bulging* eng miteinander verknüpft sind, konnte durch die Filtermodifikation auch letzteres Problem erheblich abgeschwächt werden. Versuche einer vollständigen Vermeidung von *Bulging* durch die in [Blo95b] eingeführte *Combination Surface* haben gezeigt, dass diese im Rahmen der Gefäßvisualisierung nicht geeignet ist. Die Überführung der impliziten Beschreibung der Gefäßstruktur in ein polygonales Modell wurde durch die Verwendung von Hüllkörpern erheblich beschleunigt. Selbst sehr komplexe Gefäßbäume können innerhalb weniger Sekunden in ausgezeichneter Qualität dargestellt werden. Vorher waren hier Wartezeiten bis zu einer knappen Stunde einzuplanen.

Alle in der Einleitung aufgestellten Anforderungen an eine ideale Gefäßvisualisierungsmethode konnten im Rahmen dieser Arbeit erfüllt werden. Die Verwendung von *Convolution Surfaces* zusammen mit der Gauß-Filterfunktion gestattet eine korrekte Durchmesserabbildung durch eine Reduzierung der Filterbreite. Die generierte Gefäßform zeigt weiche Übergänge zwischen einzelnen Gefäßteilen und wirkt organisch. Die Gefäßoberfläche ist vollkommen frei von Diskontinuitäten. Diese Anforderung, sowie die einer einheitlichen Behandlung aller Verzweigungstypen, geschlossene Gefäßenden und die Vermeidung von Strukturen im Inneren des Gefäßmodells werden allein durch den Einsatz von *Convolution Surfaces* garantiert.

Neben der Visualisierung des Gefäßmodells wurde zusätzlich eine kantenbasierte Interaktion möglich gemacht. Zu jedem Gefäßabschnitt sind die korrespondierenden Oberflächendreiecke bekannt. Dies gestattet die Selektion von Teilbäumen und das Hervorheben bzw. Ausblenden benutzerdefinierter Teile der Gefäßstruktur. Die weiterhin etablierte Zuordnung eines Skelettvoxels zu jedem Punkt eines Dreiecks erlaubt die Einfärbung der Gefäße hinsichtlich in einer Gefäßanalyse festgelegter Kriterien.

Die Ergebnisse einer Evaluierung der hier vorgestellten Gefäßvisualisierungsmethode durch Chirurgen und Radiologen zeigen, dass diese im Vergleich mit *Surface Rendering* und einer modellbasierten Visualisierung nach [HPSP01] durchweg die besten Resultate erzielt. Der Unterschied der Bewertungen zwischen den beiden modellbasierten Verfahren ist nicht signifikant, liefert jedoch Indizien dafür, dass die Visualisierung mit *Convolution Surfaces* in eine vielversprechende Richtung zeigt. Vor allem die Ähnlichkeit der Gefäßform zu intraoperativen Ansichten und die visuelle Qualität wurden auch durch Kommentare der befragten Ärzte hervorgehoben.

Bei der Entwicklung einer Gefäßvisualisierung, basierend auf *Convolution Surfaces*, wurden einige Verbesserungsmöglichkeiten deutlich, welche zu weiterführenden Betrachtung anregen. Diese lassen sich wie folgt kategorisieren:

Validierung Die Einschätzung der Korrektheit eines Visualisierungsresultats ist besonders für die spätere Verwendung in klinischen Applikationen von großer Wichtigkeit. Die bisher angewandten Validierungsmethoden lieferten wenig aussagekräftige Resultate. Vor allem die korrekte Abbildung des Gefäßdurchmessers sollte genauer verifiziert werden. Der bisherige visuelle Vergleich zwischen einer Visualisierung durch Kegelstümpfe und der *Convolution Surface* könnte durch die Spezifikation von korrespondierenden Oberflächenpunkten und die Messung der durchschnittlichen Abweichung zu verlässlicheren Ergebnissen führen. Die Voxelisierung der *Convolution Surface* und der anschließende Vergleich mit dem Segmentierungsergebnisses resultieren

nur in einem globalen Maß der Abweichung. Eine Verbesserung ist durch die Bestimmung der lokalen Abweichung zwischen einzelnen Gefäßästen mittels der Hausdorff-Distanz möglich.

Effizienz Die Effizienz der Polygonisierung konnte durch den Einsatz von Hüllkörpern verbessert werden. Die hierbei erzeugte Datenstruktur führt jedoch zu einem erhöhten Speicherbedarf. Polygonisierungstechniken, welche die gegebene Struktur elementweise traversieren [DTG96], machen die Datenstruktur unnötig, da hier naturgemäß jeder Zeit bekannt ist, welcher Abschnitt des Modells gerade bei der Feldfunktionsberechnung betrachtet werden muss.

Weiterhin sollte die bei der Polygonisierung mit dem *Implicit Polygonizer* auftretende hohe Dreieckszahl reduziert werden. Dies wird durch eine direkte Generierung von Oberflächendreiecken nach [AG01] erreicht. Zusätzlich ist eine Anpassung der Dreiecksgröße an die Gefäßdicke und die Gefäßkrümmung wünschenswert. Verschiedene adaptive Methoden wurden hierzu bereits entwickelt.

Interaktion Die in Abschnitt 5.4 vorgestellte Interaktionsmöglichkeit unter OpenInventor ist für Gefäßstrukturen nur begrenzt einsetzbar, da sie bildbasiert ist. Der Benutzer muss die Gefäßstruktur geeignet positionieren, um ein gewünschtes Gebiet ungehindert umfahren zu können. Dies ist bei komplexen Gefäßstrukturen oft nicht möglich, da andere Gefäße die gewünschte Struktur verdecken. Ideal wäre ein Verfahren bei dem der Benutzer den gewünschten Teilbaum durch einen einzigen Mausklick auf die zur Wurzel nächstliegende Gefäßkante selektieren kann. Möglich scheint dies durch den Aufbau des Szenengraphen unter OpenInventor entsprechend dem Gefäßgraphen. Wird ein Knoten im Szenengraphen selektiert, entspricht der dort abgehende Teilbaum dem des Gefäßgraphen. OpenInventor ermöglicht nun die Veränderung der Materialeigenschaften des Teilbaumes oder verhindert dessen Traversierung.

Abbildungsverzeichnis

1.1 Nahansicht einer Verzweigung 2

2.1 Zweidimensionale Schichtdarstellung (*Slicing*) 6
2.2 *Curved Planar Reformation*-Prinzip 7
2.3 *Curved Planar Reformation*-Beispiel 7
2.4 *Maximum Intensity Projection* und *Closest Vessel Projection* 9
2.5 Vergleich von *Maximum Intensity Projection* und *Closest Vessel Projection* . . . 9
2.6 *Surface Rendering* 10
2.7 Gefäßspezifische Visualisierungspipeline 11
2.8 2D-Illustration der Gefäßskelettierung 12
2.9 Graphenanalyse von Lebergefäßsystemen 13
2.10 Gefäßbeschreibung durch Skelett und Gefäßquerschnitte 14
2.11 Gefäßvisualisierung mit Zylindern 15
2.12 Gefäßvisualisierung mit Kegelstümpfen nach Puig 16
2.13 Glättung des Gefäßskeletts 17
2.14 Gefäßvisualisierung mit Kegelstümpfen nach Hahn 18
2.15 Gefäßvisualisierung mit Freiformflächen 18
2.16 Gefäßvisualisierung mit *Subdivision Surfaces* - Schritt 1 19
2.17 Gefäßvisualisierung mit *Subdivision Surfaces* - Schritt 1 20
2.18 Direkte Darstellung des symbolischen Gefäßmodells 20

3.1 Adeninmolekül modelliert mit *Blobs* 24
3.2 *Convolution Surface* des Skeletts einer menschlichen Hand 25
3.3 *Blobby Molecules* 26
3.4 Skelettbasierte Modellierung - Beispiel Seepferdchen 27
3.5 *Distance Surfaces* 28
3.6 *Bulging* entlang der *Union Surfaces* 29
3.7 Prinzip der *Convolution Surfaces* 30
3.8 Superpositions-Eigenschaft der Faltung 30
3.9 *Bulging* an der Verzweigung eines T-Skeletts 31
3.10 Analyse des *Bulging*-Problems 31
3.11 Verminderung von *bulging* durch Separation von Skelettprimitiven 32
3.12 *Union Surface* und *Convolution Surface* einer Verzweigung vierter Ordnung . . . 32
3.13 *Combination Surface* einer Verzweigung vierter Ordnung 33
3.14 Höherdimensionale Skelettprimitive 33
3.15 *Unwanted Blending*-Problem 34
3.16 *Blending*-Kategorien nach Opalach 34
3.17 Kontaktmodellierung nach Opalach 35

3.18 *Local Convolution* . . . 36
3.19 *Local Convolution* und verbesserte *Local Convolution* . . . 36
3.20 Funktionsverlauf des biquadratischen Polynoms . . . 38
3.21 Durchmesservariation mit Hilfe von Profilfunktionen . . . 39
3.22 Kubische Kontrollkurve zur Durchmesservariation . . . 40
3.23 Durchmesservariation von *Convolution Surfaces* nach Jin . . . 40
3.24 Faltung eines Liniensegments nach Bloomenthal . . . 41
3.25 Integrationsfilter an verschiedenen Punkten eines Liniensegments . . . 43
3.26 Funktionsverlauf der inversen kubischen Funktion . . . 43
3.27 Berechnung der Feldfunktion nach Cani . . . 44
3.28 Funktionsverlauf der Filterfunktion nach Hornus . . . 45
3.29 Bestimmung des Faltungsintegrals nach Hornus . . . 45
3.30 Lichttransport in einer Szene . . . 47
3.31 *Ray Tracing* der Verschmelzung dreier *Soft Objects* . . . 48
3.32 *Ray Tracing* Beispiele . . . 49
3.33 Polygonisierung durch *Spatial Sampling* . . . 50
3.34 Polygonisierung durch adaptives *Spatial Sampling* . . . 51
3.35 Polygonisierung durch *Surface Tracking* . . . 52
3.36 *Marching Triangles* . . . 53
3.37 Polygonisierung durch *Surface Fitting* . . . 54
3.38 Visualisierung der impliziten Oberfläche durch *scales* und *Polygonisierung* . . . 54
3.39 Visualisierung der impliziten Oberfläche durch *Section Display* . . . 55
3.40 Visualisierung der impliziten Oberfläche durch *Contour Tracing* . . . 56
3.41 Visualisierung der impliziten Oberfläche durch Partikelsysteme . . . 56

4.1 Skelett dreier Lebergefäßbäume . . . 59
4.2 Lebergefäßbaum modelliert mit einer parametrischen und drei impliziten Methoden. 61
4.3 Durchmesserverlauf der *Convolution Surface* eines isolierten Liniensegments . . . 61
4.4 Verzweigung modelliert durch Kegelstümpfe und *Convolution Surface* . . . 62
4.5 *Blending* entlang einer Bifurkation . . . 63
4.6 *Bulging* an der Wurzel eines Gefäßbaums . . . 63
4.7 *Unwanted Blending* entlang der *vena cava* . . . 64
4.8 Auswirkung der Änderungen von *radius in isolation* und *blobbiness* . . . 65
4.9 Skalarfeld eines Liniensegments gefaltet mit unterschiedlicher Filterbreite . . . 65
4.10 Gauß-Funktionen mit unterschiedlicher Filterbreite . . . 66
4.11 Trifurkation modelliert mit verschiedenen *width coefficients* . . . 67
4.12 Lebergefäßbaum modelliert mit verschiedenen *width coefficients* . . . 67
4.13 Vergleich der Durchmesserabbildung durch Kegelstümpfe und *Convolution Surfaces* 68
4.14 Diskontinuitäten der Normalen bei sehr hohem *width coefficient* . . . 68
4.15 Auswirkung der Modifikation des *width coefficients* auf *bulging* . . . 70
4.16 Separierung von Skelettprimitiven zur Vermeidung von *bulging* . . . 71
4.17 *Convolution Surfaces* vor und nach einer Separierung der Skelettprimitive . . . 71
4.18 Eindellung und *bulge* an der *Convolution Surface* an einer Verzweigung . . . 72
4.19 Ebene durch eine „planare“ Trifurkation bestimmt nach Newell's Methode . . . 72
4.20 *Union Surface*, *Convolution Surface* und *Combination Surface* einer Trifurkation 73
4.21 Diskontinuitäten entlang der *Combination Surface* . . . 74
4.22 Ebene durch eine Trifurkation bestimmt nach Newell's Methode . . . 74

4.23 *bulging* entlang der *Combination Surface* 75
4.24 Diskontinuitäten der Normalen entlang der *Combination Surface* 75
4.25 Die *Convolution Surface* eines Skeletts aus Polygonen. 76
4.26 Untersuchung des *Unwanted Blending*-Problems am Beispiel zweier Liniensegmente. 77
4.27 *Unwanted Blending* entlang der *Convolution Surface* eines S-förmigen Skeletts. . 78
4.28 Artefakte bei der *Improved Local COnvolution* eines S-Skeletts 79
4.29 Vermeidung von Artefakten bei der *Improved Local COnvolution* eines S-Skeletts 80
4.30 Beseitigung von *Unwanted Blending* entlang eines S-Skeletts durch *Local Convolution* 80
4.31 Zwischenresultat der Polygonisierung eines Gefäßbaums 82
4.32 Datenstruktur zur effizienteren Polygonisierung 83
4.33 Unterschied zwischen Gefäßdurchmesser und Skalarfelddurchmesser 84
4.34 Hüllkörper und die jeweils eingefasste Voxelmenge am Beispiel einer künstlichen Trifurkation 86
4.35 Auswirkungen der Größe von Hüllkörpern auf die Korrektheit der Gefäßoberfläche 87
4.36 Erweiterung der Startvoxelauswahl eines voxelbasierten, achsenparallelen Hüllkörpers 89
4.37 Größe der Hüllkörper unter Verwendung der inversen kubischen Funktion 91
4.38 Suche eines Punktes außerhalb der Gefäßoberfläche bei der Startpunktbestimmung für den *Implicit Polygonizer* 92
4.39 *Binary subdivision*-Prozess zur Bestimmung des Startpunktes für den *Implicit Polygonizer* 93
4.40 Farbige Zuordnung der Oberflächendreiecke zu den Kanten einer Trifurkation. . . 95

5.1 OpenInventor-Architektur 100
5.2 OpenInventor-Szenengraph zur Modellierung und Visualisierung eines Obelisken 101
5.3 ILAB-Architektur 102
5.4 Beispielapplikation *Ventricel Volumetry* in ILAB 103
5.5 Einfärbung eines Gefäßbaums nach der Zugehörigkeit von Dreiecken des polygonalen Modells zu Kanten des Gefäßgraphen 113
5.6 Szenengraph einer Gefäßstruktur unter OpenInventor 113
5.7 Selektion eines Teilbaumes unter OpenInventor 115
5.8 Einfärbung der maximalen Teilbäume einer Gefäßstruktur 115
5.9 Lebergefäßbaum und zerebrale Gefäße visualisiert mit *Convolution Surfaces* . . . 116
5.10 Bronchialbaum und Ausgusspräparat einer Schweineleber visualisiert mit *Convolution Surfaces* 116
5.11 Nahansicht einer Verzweigung modelliert mit Kegelstümpfen und *Convolution Surfaces* 118
5.12 Geschlossene Gefäßenden und Vermeidung von Strukturen im Gefäßinneren durch *Convolution Surfaces* 118

6.1 Überlagerung von *Isosurface* und *Convolution Surface* 121
6.2 Voxelisierung der *Convolution Surface* und Vergleich mit dem Segmentierungsresultat 122
6.3 Leberarterie im Kontext der Leber 123
6.4 Visualisierung einer Verzweigung konstruiert mittels *Surface Rendering*, Kegelstümpfen und *Convolution Surfaces* 124

Literaturverzeichnis

[AC02] A. Angelidis und M.-P. Cani: „Adaptive Implicit Modeling using Subdivision Curves and Surfaces as Skeletons“. In: *Solid Modelling and Applications*. 2002.

[AG01] S. Akkouche und E. Galin: „Adaptive Implicit Surface Polygonization using Marching Triangles“. *Computer Graphics Forum*, 20(2):(2001), S. 67–80.

[AJC02] A. Angelidis, P. Jepp und M.-P. Cani: „Implicit Modeling with Skeleton Curves: Controlled Blending in Contact Situations“. In: *Shape Modeling International*. IEEE Computer Society Press, 2002. S. 137–144.

[BBB+97] J. (ed.) Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani, A. Rockwood, B. Wyvill und G. Wyvill: *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[BGSC85] C. Barillot, B. Gibaud, J. Scarabin und J. Coatrieux: „3d reconstruction of cerebral blood vessels“. *IEEE Computer Graphics and Applications*, 5(12):(1985), S. 13–19.

[Bli82] J.F. Blinn: „A Generalization of Algebraic Surface Drawing“. *ACM Transactions on Graphics (TOG)*, 1(3):(1982), S. 235–256.

[Blo88] J. Bloomenthal: „Polygonization of Implicit Surfaces“. *Computer Aided Geometric Design*, 5:(1988), S. 341–355.

[Blo94] J. Bloomenthal: „An implicit surface polygonizer“. S. 324–349.

[Blo95a] J. Bloomenthal: „Bulge Elimination in Implicit Surface Blends“. In: *Implicit Surfaces*. 1995. S. 7–20.

[Blo95b] J. Bloomenthal: *Skeletal Design of Natural Forms*. Dissertation, Univerity of Calgary, Calgary, Alberta, Canada, 1995.

[Blu67] H. Blum: „A tranformation for extracting new descriptors of shape“. In: W.W. Dunn (Hrsg.), *Proc. Symp. Models for the Perception of Speech and Visual Form*. MIT Press, Cambridge, MA, US, 1967. S. 362–380.

[Bou97] P. Bourke: „Modelling the Surface of the Human Cortex from MRI Volumetric Data“, 1997. Http://astronomy.swin.edu.au/ pbourke/modelling/cortex/impl.html.

[BS91] J. Bloomenthal und K. Shoemake: „Convolution Surfaces“. *Computer Graphics*, 25(4):(1991), S. 251–256.

[BW90] J. Bloomenthal und B. Wyvill: „Interactive Techniques for Implicit Modeling“. Bd. 24. 1990. S. 109–116.

[BZ01] H. Biermann und D. Zorin: „Subdivide 2.0“, 2001. `http://mrl.nyu.edu/~biermann/subdivision/`.

[Cas91] C. (POV-Team co-ordinator) et al. Cason: „POV-Ray“, 1991. Erhältlich von http://www.povray.org/download/.

[CH01] M.-P. Cani und S. Hornus: „Subdivision Curve Primitives: a New Solution for Interactive Implicit Modeling“. In: *Shape Modelling International*. 2001.

[Cho99] T.Y. Chow: „What is a Closed-Form Number.“ *American Math. Monthly*, 106:(1999), S. 440–448.

[DKT98] T. DeRose, M. Kass und T. Truong: „Subdivision surfaces in character animation“. In: *Proc. of Computer graphics and interactive techniques*. ACM Press, 1998. S. 85–94.

[DTG96] M. Desbrun, N. Tsingos und M.-P. Gascuel: „Adaptive Sampling of Implicit Surfaces for Interactive Modelling and Animation“. *Computer Graphics Forum*, 15(5):(1996), S. 319–325.

[EDKS94] H.H. Ehricke, K. Donner, W. Koller und W. Strasser: „Visualization of vasculature from volume data“. *Computers and Graphics*, 18(3):(1994), S. 395–406.

[FCGA97] E. Ferley, M.-P. Cani-Gascuel und D. Attali: „Skeletal Reconstruction of Branching Shapes“. *Computer Graphics Forum*, 16(5):(1997), S. 283–293.

[FFKW02] P. Felkel, A.-L. Fuhrman, A. Kanitsar und R. Wegenkittl: „Surface Reconstruction of the Branching Vessels for Augmented Reality Aided Surgery“. *BIOSIGNAL 2002*, 16:(2002), S. 252–254.

[GCD+98] J.-D. Gascuel, M.-P. Cani, M. Desbrun, E. Leroy und C. Mirgon: „Simulating Landslides for Natural Disaster Prevention“. In: *9th Eurographics Workshop on Computer Animation and Simulation (EGCAS'98)*. 1998.

[GKS+93] G. Gerig, T. Koller, G. Szkely, C. Brechbühler und O. Kübler: „Symbolic Description of 3d structures applied to cerebral vessel tree obtained from MR angiography volume data“. In: *Proc. of Information Processing in Medical Imaging*, Bd. 687. Springer, LNCS, 1993. S. 94–111.

[Gla89] A. (ed.) Glassner: *An Introduction to Ray Tracing*. Academic Press, 1989.

[HAC03] S. Hornus, A. Angelidis und M.-P. Cani: „Implicit Modelling Using Subdivision-curves“. *The Visual Computer*, 19(2-3):(2003), S. 94–104.

[Hal01] S.M. Halpine: „Science Visualization and Educational Animation at SIGGRAPH 2001: The Next Big Deal“. 6(8):(2001), S. 39–42.

[Hec86] P.S. Heckbert: „Fun with Gaussians“. In: *SIGGRAPH '86 Advanced Image Processing seminar notes*. 1986.

[HLP03] H.K. Hahn, F. Link und H.-O. Peitgen: „Concepts for Rapid Application Prototyping in Medical Image Analysis and Visualization“. In: *SV - Simulation und Visualisierung*. 2003.

[HPSP01] H.K. Hahn, B. Preim, D. Selle und H.-O. Peitgen: „Visualization and Interaction Techniques for the Exploration of Vascular Structures“. In: *Proc. of IEEE Visualization*. 2001. S. 395–402.

[HSEP00] H. Hahn, D. Selle, C. Evertsz und H.-O. Peitgen: „Interaktive Visualisierung von Gefäßsystemen auf der Basis von Oberflächenprimitiven“. In: *Proc. of Simulation und Visualisierung*. SCS-Verlag, 2000. S. 105–118.

[HSIW96] A. Hilton, A. Stoddart, J. Illingworth und T. Windeatt: „Marching triangles: range image fusion for complex object modelling“. In: *Proc. of International Conference on Image Processing*. 1996.

[JT02] X. Jin und C.-L. Tai: „Convolution Surfaces for Arcs and Quadratic Curves with a Varying Kernel“. *The Visual Computer*, 18(8):(2002), S. 530–546.

[JTFP01] X. Jin, C.-L. Tai, J. Feng und Q. Peng: „Convolution Surfaces for Line Skeletons with Polynomial Weight Distributions“. *Journal of Graphics Tools*, 6(3):(2001), S. 17–28.

[Kan01] Armin Kanitsar: *Advanced Visualization Techniques for Vessel Investigation*. Diplomarbeit, University of Technology Vienna, Institute of Computergraphics and Algorithm, 2001.

[Kaw96] Y. Kawaguchi: „Cellular“. In: *ACM SIGGRAPH 96 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '96*. ACM Press, 1996. S. 48. Bild unter http://www.siggraph.org/artdesign/gallery/S96/brid49.html (Stand: 7.12.2003).

[KB91] C. Kolb und R. Bogart: „Rayshade“, 1991. Erhältlich über ftp von ftp://graphics.stanford.edu/pub/rayshade/.

[KFW+02] A. Kanitsar, D. Fleischmann, R. Wegenkittl, P. Felkel und E. Gröller: „CPR: curved planar reformation“. In: *Proc. of IEEE Visualization*. 2002. S. 37–44.

[KWF+01] A. Kanitsar, R. Wegenkittl, P. Felkel, D. Fleischmann, D. Sandner und E. Gröller: „Computed tomography angiography: a case study of peripheral vessel investigation“. In: *Proc. of IEEE Visualization*. 2001. S. 477–480.

[LC87] W.E. Lorensen und H.E. Cline: „Marching cubes: A high resolution 3D surface construction algorithm“. In: *Proc. of Computer graphics and interactive techniques*. ACM Press, 1987. S. 163–169.

[LvdGRR+99] B.P.F. Lelieveldt, R.J. van der Geest, M. Ramze Rezaee, J.G. Bosch und J.H.C. Reiber: „Anatomical Model Matching with Fuzzy Implicit Surfaces for Segmentation of Thoracic Volume Scans“. *IEEE Transactions on Medical Imaging*, 18(3):(1999), S. 218–230.

[MMD96] Y. Masutani, K. Masamune und T. Dohi: „Region-Growing-Based Feature Extraction Algorithm for Tree-Like Objects“. In: *Proc. of Visualization in Biomedical Computing*, Bd. 1131. Springer, LNCS, 1996. S. 161–171.

[MS98] J. McCormack und A. Sherstyuk: „Creating and Rendering Convolution Surfaces“. *Computer Graphics Forum*, 17(2):(1998), S. 113–120.

[MS03] Cermak M. und V. Skala: *Methods for Implicit Surfaces Polygonization.* Technischer Bericht DCSE/TR-2003-01, University of West Bohemia, Plzen, Czech Republic, 2003.

[Nev82] R. Nevatia: *Machine Perception.* Prentice Hall, 1982.

[NHK+85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa und K. Omura: „Object Modelling by Distribution Function and a Method of Image Generation". *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):(1985), S. 718–725.

[OM93] A. Opalach und S. C. Maddock: „Implicit Surfaces: Appearance, Blending and Consistency". In: *Proc. of Eurographics Workshop on Animation and Simulation.* 1993. S. 233–245.

[OM95] A. Opalach und S. C. Maddock: „An Overview of Implicit Surfaces". In: *Introduction to Modelling and Animation Using Implicit Surfaces.* 1995. S. 1.1–1.13.

[Pou00] W. Poundstone: *Carl Sagan: A Life in the Cosmos.* Owl Publishing Company, 2000.

[PT90] B.A. Payne und A.W. Toga: „Medical Imaging: Surface Mapping Brain Function on 3D Models". *IEEE Computer Graphics and Applications*, 10(5):(1990), S. 33–41.

[PTN97] A. Puig, D. Tost und I. Navazo: „An interactive cerebral blood vessel exploration system". In: *IEEE Visualization.* 1997. S. 443–446.

[Pui98a] A. Puig: *Cerebral Blood Vessels Modelling.* Technischer Bericht LSI-98-21-R, Universitat de Barcelona, Barcelona, Spain, 1998.

[Pui98b] A. Puig: *Visualization of Cerebral Blood Vessels.* Technischer Bericht LSI-98-20-R, Universitat de Barcelona, Barcelona, Spain, 1998.

[RR78] A Ralston und P. Rabinowitz: *First Course in Numerical Analysis.* McGraw Hill College Div, 1978.

[RS01] C. Rezk-Salama: *Volume Rendering Techniques for General Purpose Graphics Hardware.* Dissertation, Universität Erlangen-Nürnberg, Erlangen, Deutschland, 2001.

[SAC+99] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret und J.-D. Gascuel: „Animating Lava Flows". In: *Graphics Interface (GI'99) Proceedings.* 1999. S. 203–210.

[Sch99] M. Schweppe: „ACM SIGGRAPH 99 Electronic art and animation catalog". ACM Press, 1999.

[SH97] B.T. Stander und J.C. Hart: „Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling". *Computer Graphics*, 31(Annual Conference Series):(1997), S. 279–286.

[She94] C.-K. Shene: „Equations of Cylinders and Cones". S. 321–323.

[She97] A. Sherstyuk: *Shells, crabs and seahorses: expanding the modeling power of implicit surfaces.* Technischer Bericht 97/330, Monash University, Melbourne, Australia, 1997.

[She98a] A. Sherstyuk: *Convolution Surfaces in Computer Graphics.* Dissertation, Monash University, Melbourne, Victoria, Australia, 1998.

[She98b] A. Sherstyuk: „Fast ray tracing of implicit surfaces“. *Computer Graphics Forum*, 18(2):(1998), S. 139–147.

[She99a] A. Sherstyuk: „Interactive Shape Design with Convolution Surfaces“. In: *Proceedings of Shape Modeling International.* 1999. S. 56–65.

[She99b] A. Sherstyuk: „Kernel functions in convolution surfaces: a comparative analysis“. *The Visual Computer*, 15(4):(1999), S. 171–182.

[SPSP02] D. Selle, B. Preim, A. Schenk und H.-O. Peitgen: „Analysis of Vasculature for Liver Surgical Planning“. *IEEE Transactions on Medical Imaging*, 21(11):(2002), S. 1344–1357.

[SSPP00] D. Selle, W. Spindler, B. Preim und H.-O. Peitgen: „Mathematical Methods in Medical Image Processing: Analysis of Vascular Structures for Preoperative Planning in Liver Surgery“. In: B. Engquist und W. Schmid (Hrsg.), *Springer's Special Book for the World Mathematical Year 2000: Mathematics Unlimited - 2001 and Beyond.* Springer, 2000. S. 1039–1059.

[ST90] P. Shirley und A.A. Tuchman: „Polygonal Approximation to Direct Scalar Volume Rendering“. In: *Proceedings San Diego Workshop on Volume Visualization, Computer Graphics*, Bd. 24. 1990. S. 63–70.

[Sun02] D. Sunday: „Fast Polygon Area and Newell Normal Computation“. *Journal of Graphics Tools*, 7(2):(2002), S. 9–13.

[TBG95] N. Tsingos, E. Bittar und M.-P. Gascuel: *Implicit Surfaces for Semi-Automatic Medical Organ Reconstruction.* Academic Press, 1995 .

[TGMC03] F. Triquet, L. Grisoni, P. Meseure und C. Chaillou: „Realtime Visualization of Implicit Objects with Contact Control“. *GRAPHITE'2003, International Conference on Computer Graphics and Interactive Techniques in Australia and South East Asia, Sponsored by ACM Siggraph (Melbourne, Australia)*, 1.

[VGdF02] L. Velho, J. Gomes und L.H. de Figueiredo: *Implicit Objects in Computer Graphics.* Springer-Verlag, 2002.

[vOW93] K. van Overveld und B. Wyvill: „Potentials, Polygons and Penguins. An efficient adaptive algorithm for triangulating an equi-potential surface“. In: *Proc. 5th Annual Western Computer Graphics Symposium (SKIGRAPH 93).* 1993. S. 31–62.

[Wer] J. Wernecke: *The Inventor Mentor, Programming Object-Oriented 3D Graphics with Open Inventor^{TM}.*

[WGG99] B. Wyvill, E. Galin und A. Guy: „Extending The CSG Tree. Warping, Blen-

ding and Boolean Operations in an Implicit Surface Modeling System“. *Computer Graphics Forum*, 18(2):(1999), S. 149–158.

[WH94] A.P. Witkin und P.S. Heckbert: „Using Particles to Sample and Control Implicit Surfaces“. *Computer Graphics*, 28(Annual Conference Series):(1994), S. 269–277.

[WMW86] G. Wyvill, C. McPheeters und B. Wyvill: „Data Structure for Soft Objects“. *The Visual Computer*, 2(4):(1986), S. 227–234.

[WvO96] B. Wyvill und K. van Overveld: „Tiling Techniques for Implicit Skeletal Models“. SIGGRAPH Course 11: C1.1–C1.26, 1996.

[WW92] A. Watt und M. Watt: *Advanced Animation and Rendering Techniques*. Addison-Wesley Professional, 1992.

[ZDMP94] A.P. Zijdenbos, B.M. Dawant, R.A. Margolin und A.C. Palmer: „Morphometric Analysis of White Matter Lesions in MR Images: Method and Validation“. *IEEE Transactions on Medical Imaging*, 13(4):(1994), S. 716–724.

[ZJP95] C. Zahlten, H. Jürgens und H.-O. Peitgen: *Visualization in Scientific Computing*. Springer, 1995 S. 41–52.

[Zui95] K.J. Zuiderveld: *Visualisation of Multimodality Medical Volume Data using objekt-oriented methods*. Dissertation, Utrecht University, Utrecht, Netherlands, 1995.

A Anhang

A.1 Gefäßanalysedaten

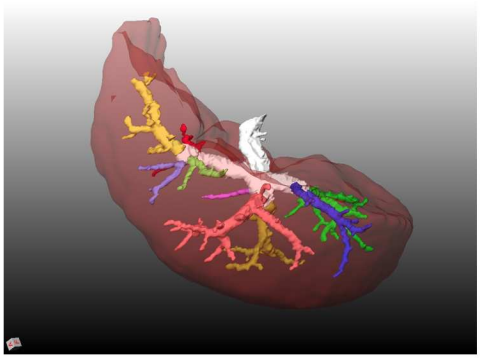
Die in der Gefäßanalyse gewonnenen Daten können sind als Textdatei (*.txt) oder im *Extensible Markup Language* Format (*.xml) verfügbar. Dieser Anhang zeigt einen Auszug aus solch einer Textdatei und beschreibt kurz die enthaltenen Daten. Die Visualisierung aller Gefäßstrukturen innerhalb dieser Arbeit basierte ausschließlich auf diesen Informationen.

```
// Header
Imagesize:                  // Dimension des Volumendatensatzes
VoxelSize:                  // Voxelausdehnung
VoxelToWorldMatrix:         // Matrix für die Transformation der Gefäßstruktur
                            // vom Voxel- ins Weltkoordinatensystem
nodeNum:                    // Anzahl der Knoten des Gefäßgraphen
edgeNum:                    // Anzahl der Kanten des Gefäßgraphen

// Knoten
NodeID:                     // einzigartige Identifikationsnummer
      Koord:                // Voxelkoordinaten
      edgeNum:              // Anzahl inzidierender Kanten
      IDs:                  // Identifikationen der Kanten
// Kanten
EdgeID:                     // einzigartige Identifikationsnummer
      predID:               // Identifikation des Vorgängerknotens
      succID:               // Identifikation des Nachfolgerknotens
      cyc:                  // schließt Kante einen Zyklus
      Hiera:                // Hirarchiestufe
      VoxNum:               // Anzahl der überspannten Voxel
            Koords:         // Voxelkoordinaten der Voxel
            minDist:        // minimaler Radius je Voxel
            maxDist:        // maximaler Radius je Voxel
            LabNum:         // Anzahl der Einfärbemöglichkeiten
                  Labs:     // Färbung nach korrespondierendem Unterbaum
                  Labs:     // Färbung nach Hierarchiestufe
                  Labs:     // Färbung nach Abstand Skelettvoxel - bel. Pkt.
                  Labs:     // Färbung nach Abstand Skelettvoxel - Wurzel

rootNum:                    // Anzahl der enthaltenen Teilbäume
rootIDs:                    // Identifikation der jeweiligen Wurzelknoten
```

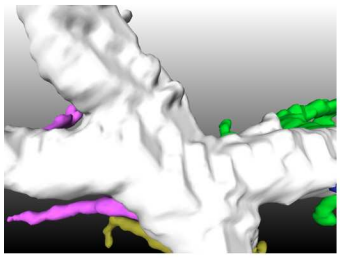
A.2 Evaluierungsbogen

Kriterium	Gefäßbaum 1 (G1)							
	G1-Iso_1	G1-Hahn_1	G1-neu_1	G1-Iso_2	G1-Hahn_2	G1-neu_2	G1-Iso_3	G1-Hahn_3	G1-neu_3
Übersichtlichkeit									
Interpretierbarkeit der räumlichen Verhältnisse									
Ähnlichkeit der Gefäßform im Vergleich zu intraoperativen Ansichten(optional)									
Visuelle Qualität									
Sonstige Bemerkungen									

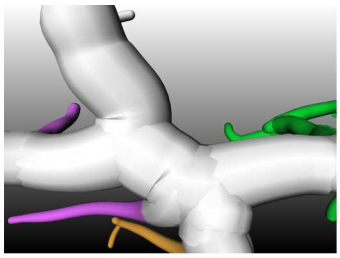
A.2.1 Lebergefäßbaum 1 (G1)



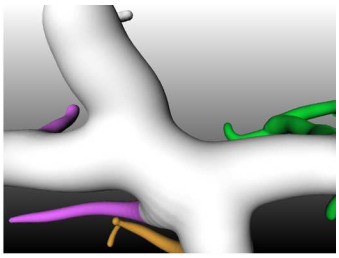
Überblicksdarstellung



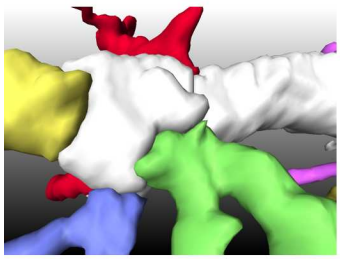
G1-Iso_1



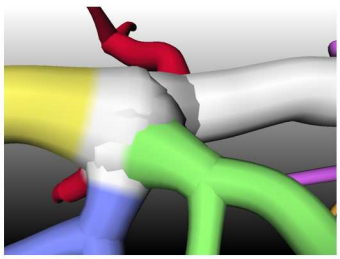
G1-Hahn_1



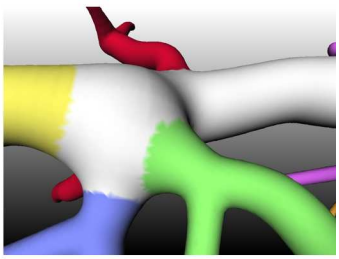
G1-neu_1



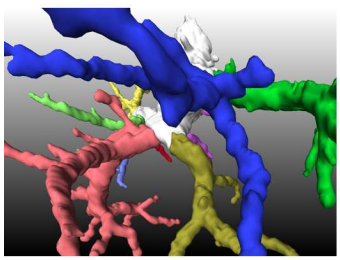
G1-Iso_2



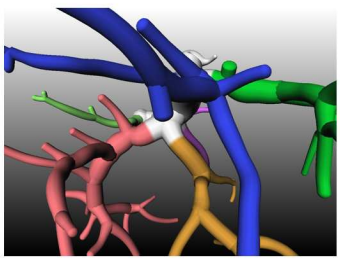
G1-Hahn_2



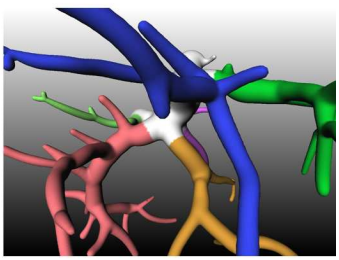
G1-neu_2



G1-Iso_3

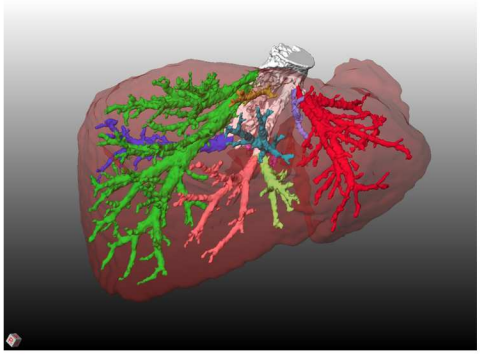


G1-Hahn_3

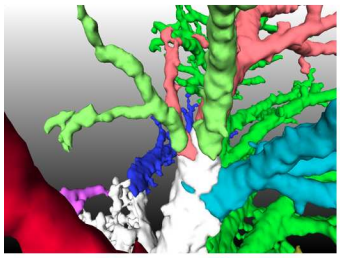


G1-neu_3

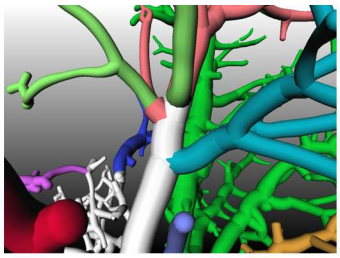
A.2.2 Lebergefäßbaum 2 (G2)



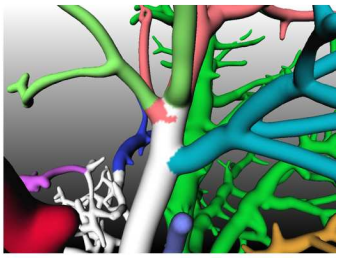
Überblicksdarstellung



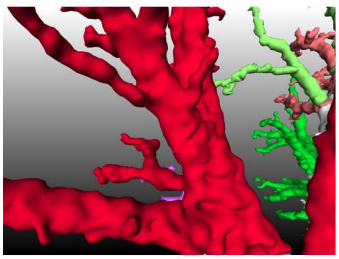
G2-Iso_1



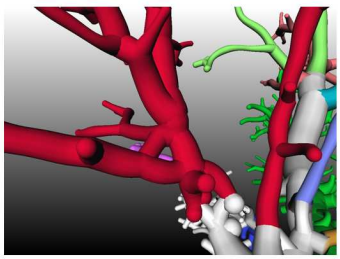
G2-Hahn_1



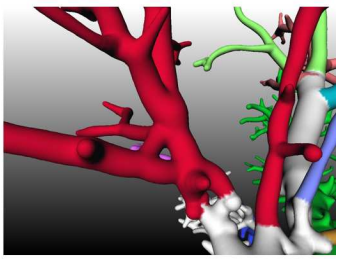
G2-neu_1



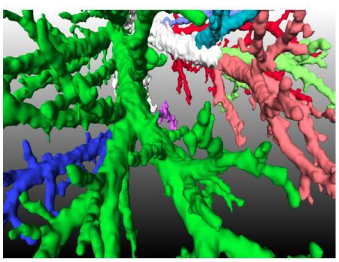
G2-Iso_2



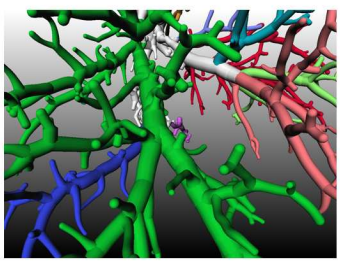
G2-Hahn_2



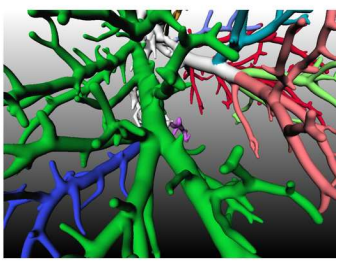
G2-neu_2



G2-Iso_3

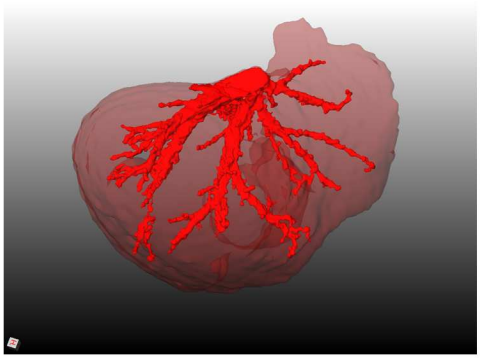


G2-Hahn_3

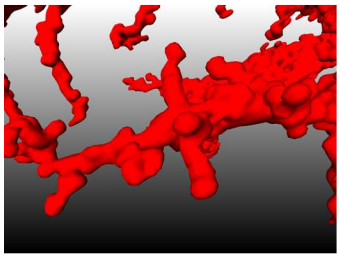


G2-neu_3

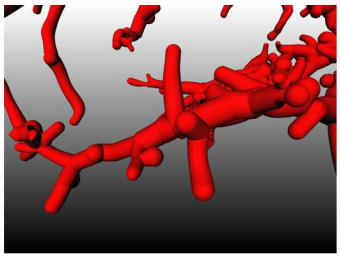
A.2.3 Lebergefäßbaum 3 (G3)



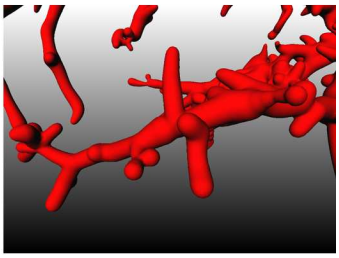
Überblicksdarstellung



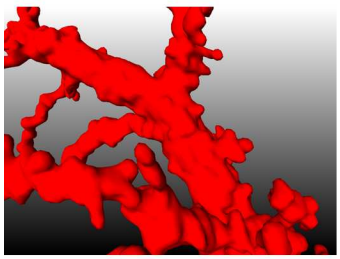
G3-Iso_1



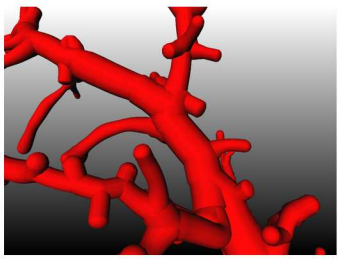
G3-Hahn_1



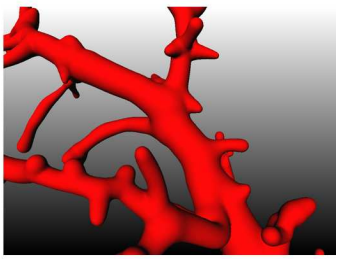
G3-neu_1



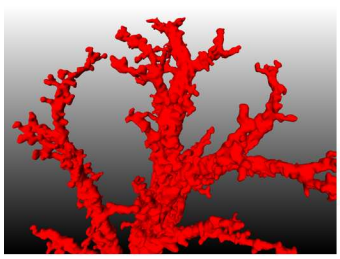
G3-Iso_2



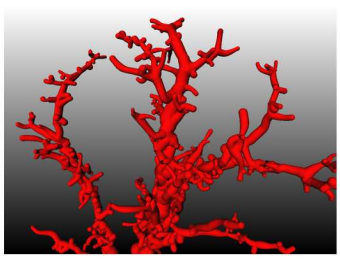
G3-Hahn_2



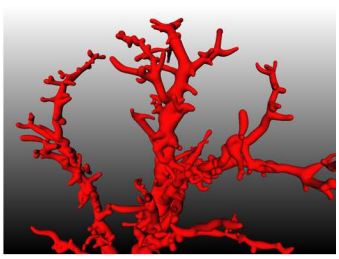
G3-neu_2



G3-Iso_3



G3-Hahn_3



G3-neu_3