Hervorhebung von Herzkranzgefäßen in Volumendatensätzen mit Hilfe von größenbasierten Transferfunktionen

Sebastian Pahnke
Matrikelnummer: 6398832
E-Mail: spahnke@mail.uni-paderborn.de

24. Oktober 2010

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Fakultät für Elektrotechnik, Informatik und Mathematik

Hervorhebung von Herzkranzgefäßen in Volumendatensätzen mit Hilfe von größenbasierten Transferfunktionen

Bachelorarbeit

im Rahmen des Studiengangs Informatik
zur Erlangung des Grades
Bachelor of Science

von

Sebastian Pahnke
Rosenstraße 23
59597 Erwitte

vorgelegt bei

Prof. Dr. Gitta Domik
Jun.-Prof. Dr. Christoph Sorge

Betreuer:

Dipl. Inform. Stephan Arens

Erwitte, den 24. Oktober 2010

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis **iii**

Quelltextverzeichnis **v**

1 Einleitung **1**
1.1 Motivation und Problemstellung 1
1.2 Struktur der Arbeit 2
1.3 Verwandte Arbeiten 2

2 Grundlagen **5**
2.1 Computertomographie 5
2.2 Volumenrendering 7
2.2.1 Volumenrendering Pipeline 7
2.2.2 Ray-Casting 8
2.2.3 Transferfunktionen 9
2.3 Größenbasierte Hervorhebung von Strukturen 10
2.3.1 Scale-Space 10
2.3.2 Größenbasierte Transferfunktionen 12

3 Erkennung von Gefäßen **15**
3.1 Der Filter von Frangi et al. 15
3.2 Der Filter von Pock et al. 18
3.3 Größenabstraktion und Rückprojektion 22

4 Implementation **23**
4.1 VolumeStudio und Plugins 23
4.2 Mathematische Operationen 25
4.2.1 Partielle Ableitungen 26
4.2.2 Gauss-Filter 27
4.2.3 Eigenwerte und Eigenvektoren 28
4.3 Implementation der Filter 29
4.4 Rückprojektion 35

5 Evaluation der Ergebnisse **39**
5.1 Validierung der Verfahren 39
5.2 Anwendung auf reale Datensätze 42

6 Zusammenfassung und Ausblick **47**

Literaturverzeichnis **49**

Abbildungsverzeichnis

2.1 Röntgen-Bild einer Hand. 6
2.2 Drei aufeinanderfolgende Schichtbilder eines CT-Scans des Kopfes. . . . 6
2.3 Aufbau eines CT-Scanners. 7
2.4 Prinzip von Ray-Casting. 9
2.5 Volumenrendering einer Hand. 10
2.6 Mit einem Gauss-Filter weichgezeichnete Kreise. 11

3.1 Richtung der wesentlichen Krümmung bei einer Röhre. 16
3.2 Eigenvektoren und -werte der Hesse-Matrix. 17
3.3 Grafische Veranschaulichung des Verfahrens von Pock et al. 19
3.4 Länge der Gradienten in der Querschnittsfläche einer Röhre. 22

4.1 Oberfläche der Software *VolumeStudio2*. 25

5.1 Die für Validierung verwendeten synthetischen Datensätze. 40
5.2 Extrahierte Maxima der synthetischen Datensätze. 41
5.3 Rückprojektion der synthetischen Datensätze. 42
5.4 CT-Datensatz des Herzens mit hervorgehobenen Herzkranzgefäßen. . . . 43
5.5 Ergebnisse des Filters von Frangi et al. 44
5.6 Ergebnisse des Filters von Pock et al. 45

Quelltextverzeichnis

4.1 Plugin-Grundgerüst. 23
4.2 Berechnung des Gradienten. 26
4.3 Berechnung der HESSE-Matrix. 26
4.4 Implementierung des GAUSS-Filters. 27
4.5 Berechnung von Eigenwerten und Eigenvektoren. 28
4.6 Die Klasse `VesselnessFrangi`. 30
4.7 Der Filter von FRANGI et al. 31
4.8 Die Klasse `VesselRadius`. 31
4.9 Der Filter von POCK et al. 33
4.10 Berechnung der Symmetry Constrainted Medialness. 33
4.11 Berechnung des Adaptive Threshold. 34
4.12 Berechnung des Boundariness-Gradienten. 35
4.13 Die Struktur `VesselCenterline`. 35
4.14 Detektion der Maxima. 36
4.15 Entfernen der Ausreißer. 37
4.16 Rückprojektion. 38

1 Einleitung

> Man kann die Erkenntnisse der Medizin auf eine knappe Formel bringen: Wasser, mäßig genossen, ist unschädlich.
>
> *(Mark Twain)*

1.1 Motivation und Problemstellung

Laut statistischem Bundesamt sind die meisten Todesfälle in Deutschland im Jahr 2008 durch Erkrankungen des Herzens verursacht worden. Angeführt wird die Statistik durch die *chronische ischämische Herzkrankheit* und den dadurch verursachten *akuten Myokardinfarkt*, welche aus einer Verkalkung der Herzkranzgefäße (Arteriosklerose) und der damit verbundenen verminderten Sauerstoffversorgung des Herzmuskels, oder pathologischen Veränderungen der Herzkranzgefäße resultieren. Die Diagnose erfolgt durch bildgebende Verfahren wie *Angiographie*, oder der weniger invasiven *Computertomographie* (CT).

Die entstehenden Volumendaten können dann mit Hilfe effizienter Visualisierungsalgorithmen – wie zum Beispiel *Ray-Casting* – dreidimensional und in Echtzeit dargestellt werden, und dem Kardiologen so ein plastisches Bild der anatomischen Strukturen liefern. Eine große Herausforderung ist nun die semantische Analyse der Volumendaten, um gezielt Strukturen von Interesse hervorzuheben. So ist es wünschenswert, die Herzkranzgefäße vom restlichen Volumen abheben, oder alle anderen Strukturen ausblenden zu können.

Die Schwierigkeiten beim Hervorheben solcher Strukturen sind physikalischer Natur. Während bei einer Magnetresonanztomographie (MRT) die Magnetisierbarkeit des Gewebes gemessen wird, wird bei der Computertomographie die Intensität der vom Gewebe durchgelassenen Röntgenstrahlung, welche man als *Dichte* oder auch als HOUNSFIELD-*Wert*[1] bezeichnet, gemessen. Probleme bei der Hervorhebung treten nun auf, weil sich die HOUNSFIELD-Wertebereiche unterschiedlicher Gewebearten überlappen; wie z. B. die Wertebereiche von Blut und Herzmuskelgewebe. Dies macht es schwierig, Herzgefäße und Herzmuskel eindeutig anhand ihrer Dichtewerte zu unterscheiden. Aus diesem Grund wird für Aufnahmen des Herzens eine Computertomographie-Angiographie (CTA) durchgeführt, bei der dem Patienten vor dem CT-Scan ein Röntgenkontrastmittel injiziert wird, um Blutgefäße vom restlichen Gewebe abzuheben. Dies allein reicht aber nicht aus, da auch die Herzkammern mit Blut gefüllt sind und so den gleichen Dichtewert wie die Blutgefäße haben. Es wurden daher Verfahren entwickelt, die zusätzliche Informationen wie Gradient, Krümmungsverhalten oder Größe zur Unterscheidung verwenden.

[1] Nach Godfrey HOUNSFIELD, einer der Erfinder der Computertomographie [PB07].

Von Correa und Ma wurde unlängst ein effektives Verfahren vorgestellt, um jedem Punkt im Volumen (genannt *Voxel*) die Größe der ihn umgebenden Struktur zuzuordnen [CM08]. Mit Hilfe dieser Information können dann gezielt Bereiche bestimmter Größe hervorgehoben werden; sei es durch Färbung oder durch das Ausblenden aller anderen Bereiche. Dieser Ansatz soll in dieser Arbeit aufgegriffen und auf die Hervorhebung von Herzkranzgefäßen spezialisiert werden. Dazu werden die von Frangi et al. [FFN+98] und Pock et al. [PBB05] vorgeschlagenen Verfahren vorgestellt. Im Gegensatz zum Ansatz von Correa und Ma, in dem die erkannten Größen nur auf einzelne Punkte zutreffen und dann über das Volumen interpoliert werden, werden hier die besonderen Eigenschaften von röhrenartigen Strukturen ausgenutzt, um eine genaue Lokalisierung der Mittellinien der Röhren und eine Abschätzung von deren Radius zu ermöglichen.

1.2 Struktur der Arbeit

Kapitel 2 liefert die für die vorgestellten Verfahren benötigten Grundkenntnisse. Begonnen wird dort mit einem Überblick über die Funktionsweise der Computertomographie. Anschließend wird erläutert, wie die damit gewonnenen Daten dreidimensional mit Hilfe des Volumenrenderings dargestellt werden können. Abgeschlossen wird das Kapitel durch die mathematischen Grundlagen der größenbasierten Hervorhebung von Strukturen. Kapitel 3 stellt zwei konkrete Verfahren zur Hervorhebung von röhrenartigen Strukturen vor und erläutert deren zu Grunde liegende Idee, sowie die mathematische Umsetzung. Die Implementierung der Verfahren wird in Kapitel 4 beschrieben, nachdem zuvor ein grober Überblick über die Software *VolumeStudio2* und die Implementierung von grundlegenden mathematischen Operationen gegeben wurde. Die Implementation wird dann in Kapitel 5 anhand von synthetischen Datensätzen validiert und die Eignung der Verfahren zur Hervorhebung von Herzkranzgefäßen anhand von realen Datensätzen evaluiert.

1.3 Verwandte Arbeiten

Der von Witkin und Lindeberg eingeführte *Scale-Space* bildet die Grundlage für viele Verfahren zur Erkennung von röhrenartigen Strukturen [Wit83, Lin90, Lin91]. Er ermöglicht die Aufspaltung eines Volumens in eine Familie von Volumina, die nur Objekte bestimmter Größe enthalten. Dazu wird das ursprüngliche Volumen – unter Umständen mehrfach – mit einem Gauss-*Filter* der Größe 1 gefaltet (vgl. Abschnitt 2.3), was zu einer sukzessiven Weichzeichnung – genauer *Diffusion* – der Objekte führt. Ausgenutzt wird dabei, dass bestimmte Eigenschaften eines Voxels besondere Charakteristika aufweisen, wenn die Größe der den Voxel umgebenden Struktur genau der Größe σ des Gauss-Filters entspricht[2]; z. B. kann der Laplace-Operator zur Erkennung von kugelförmigen Objekten benutzt werden, da ihre Mittelpunktvoxel dessen Wert minimieren.

[2] Das k-malige Anwenden eines Gauss-Filter der Größe 1 entspricht der einmaligen Anwendung eines Gauss-Filters der Größe k.

Da der Gauss-Filter *isotrop*, d. h. in alle Richtungen gleichmäßig, weichzeichnet, entstehen allerdings Probleme. Zwei nebeneinander liegende Objekte verschmelzen oft schon bei kleinen Werten von σ zu einem Einzigen. *Anisotrope* Diffusionsfilter, wie der von Perona und Malik, welche homogene Regionen im Volumen stärker weichzeichnen als Regionen mit einem hohen Gradienten (z. B. Kanten), können hier eine Lösung bieten [PM90].

Ein anderer Ansatz ist die Ausnutzung der Rotationssymmetrie von Röhren. Insbesondere die *Krümmung*, welche durch die zweite Ableitung, bzw. im Dreidimensionalen durch die Hesse-Matrix, berechnet werden kann, zeigt bei Röhren ein besonderes Verhalten. Durch die Rotationssymmetrie ist die Krümmung in Richtung der Querschnittsfläche größer als in Richtung der Röhre selbst. Diese Richtungen werden durch die Eigenvektoren der Hesse-Matrix ausgedrückt, die Größe der Krümmung durch die zugehörigen Eigenwerte.[3]

Diese Gegebenheiten wurden zum ersten Mal in den Arbeiten von Sato et al. und Lorenz et al. ausgenutzt [SNS+98, LCB+97]. Frangi et al. erweitern diese Ideen, indem sie alle Eigenwerte (anstatt nur zwei) betrachten. Mit ihnen definieren sie die Größen $\mathcal{R}_\mathcal{B}$ und $\mathcal{R}_\mathcal{A}$, welche respektive die Abweichung der den aktuellen Voxel umgebenden Struktur von kugel- und plattenförmigen Objekten beschreiben, sowie die Größe $\mathcal{S}$, welche zur Unterdrückung von Hintergrundvoxeln dient. Damit wird dann eine Funktion $\mathcal{V}_0$ erstellt, die jedem Voxel seine „Gefäßartigkeit“ (engl.: „vesselness“) zuordnet. Dieser Schritt wird mit Hilfe des Scale-Space für mehrere Werte von σ wiederholt und jeweils der maximale Wert von $\mathcal{V}_0$ gewählt (eine genaue Erläuterung folgt in Abschnitt 3.1).

Krissian et al. gehen noch einen Schritt weiter und schlagen ein adaptives Verfahren vor, welches nicht den aktuellen Voxel selbst, sondern eine Umgebung um den Voxel betrachtet [KMA+00]. Sie betrachten ein Gefäß als ein zylinderförmiges Objekt und berechnen für jeden Voxel die Ebene der (hypothetischen) Querschnittsfläche, welche durch die zwei Eigenvektoren mit den absolut größten Eigenwerten aufgespannt wird. Die aktuelle Größe des Scale-Space gibt – bis auf einen konstanten Faktor – den Radius r der Röhre an. Die „Gefäßartigkeit“ wird nun bestimmt, indem der Gradient auf einem Kreis mit Radius r in der Querschnittsebene gemessen und aufsummiert wird. In Analogie zum Verfahren von Frangi et al. wird dies für mehrere Werte von σ wiederholt und der maximale Wert gewählt.

Einen ganz anderen Ansatz wählen Bauer und Bischof [BB08]. Anstatt eines isotropen Diffusionsprozesses im Scale-Space, benutzen sie einen anisotropen Filter, welcher nur einmal auf das ursprüngliche Volumen angewandt wird. Dieser Filter generiert ein Vektorfeld $V(\boldsymbol{x})$ – das „Gradient Vector Flow“-Vektorfeld (kurz GVF) – welches das Energieintegral

$$\iiint_\Omega \mu \left|\nabla V(\boldsymbol{x})\right|^2 + \left|\nabla I(\boldsymbol{x})\right|^2 \left|V(\boldsymbol{x}) - \nabla I(\boldsymbol{x})\right|^2 \, d\boldsymbol{x}$$

minimiert. Die Grundlagen dazu wurden von Xu und Prince eingeführt [XP98]. Das GVF hat den Vorteil, dass es nur einmal berechnet werden muss und bei Röhren die gleichen Charakteristika bzgl. der Gradientenausrichtung aufzeigt, wie ein Gradienten-Vektorfeld, welches im Scale-Space mit der Größe der Röhre berechnet wurde. Auf das

[3] [BVAS+07] enthält eine sehr anschauliche Visualisierung dieser Sachverhalte.

GVF wird dann im nächsten Schritt ein Röhrenerkennungsfilter wie der von Frangi et al. angewandt.

Alle hier vorgestellten Ansätze bieten die Möglichkeit zur Erkennung der Mittellinien von Röhren in einem Volumendatensatz, sowie zur Abschätzung von deren Radius. Damit ist es durch eine *Rückprojektion* möglich, genau die Voxel auszuwählen, die zur Röhre – und damit zum Herzkranzgefäß – gehören. Dies wird im Allgemeinen als *Segmentierung* des Datensatzes bezeichnet und ist ein häufig angewandtes Verfahren zur exklusiven Visualisierung hervorzuhebender Bereiche. Aufgrund der binären Natur der Segmentierung werden allerdings auch „Ausreißer“ mit voller Intensität dargestellt.

Eine Alternative bietet hier die Verwendung einer *Transferfunktion* zur Auswahl der darzustellenden Voxel, wie Correa und Ma in ihrem Papier über größenbasierte Transferfunktionen vorschlagen [CM08]. Der gemessene Wert – in diesem Fall der Radius einer Kugel – wird als Definitionsbereich einer Transferfunktion (ein- oder zweidimensional) verwendet. Durch geschickte Spezifikation der Transferfunktion können dann genau die Voxel von Interesse dargestellt und unter Umständen auch Ausreißer[4] ausgeblendet werden.

In dieser Arbeit wird sowohl das Verfahren von Frangi et al., als auch das Verfahren von Pock et al. vorgestellt. Letzteres verwendet einen ähnlichen Ansatz wie das von Krissian et al., denn auch hier wird der Gradient in einem Kreis um den aktuellen Voxel gemessen. Der Unterschied ist, dass zwei verschiedene Scale-Spaces zur Berechnung der Gradienten und der Hesse-Matrix – und damit der Querschnittsfläche der Röhre – verwendet werden. Diese Vorgehensweise verhindert das schnelle Verschmelzen zweier nebeneinander liegender Röhren. Zudem überprüft das Verfahren die Homogenität innerhalb der Röhre und unterdrückt Hintergrundvoxel durch einen adaptiven Schwellwert.

[4] Für diese liefert die verwendete Metrik im Idealfall nämlich sehr viel kleinere Werte.

2 Grundlagen

> Phantasie ist wichtiger als Wissen,
> denn Wissen ist begrenzt.
>
> *(Albert Einstein)*

Dieses Kapitel liefert die Grundlagen, die für das Verständnis der in den nachfolgenden Kapiteln vorgestellten und evaluierten Verfahren zur Erkennung von Gefäßstrukturen nötig sind. Dazu wird zunächst die Funktionsweise der Computertomographie erläutert, mit der die medizinischen Daten gewonnen werden. Die dreidimensionale Darstellung dieser Daten ist Thema des anschließenden Abschnitts über Volumenrendering. Abgeschlossen wird das Kapitel durch einen Abschnitt über die größenbasierte Hervorhebung von Strukturen, in dem die mathematischen Grundlagen der Verfahren erläutert werden.

2.1 Computertomographie

Der erste große Meilenstein in der medizinischen Bildgebung war sicherlich die Entdeckung der RÖNTGEN-Strahlen im Jahr 1895 durch Wilhelm Konrad RÖNTGEN, wodurch es erstmals möglich wurde, einen Blick in den menschlichen Körper zu werfen ohne diesen aufschneiden zu müssen [PB07]. Sie entstehen, wenn durch ein elektrisches Feld stark beschleunigte Elektronen auf ein Zielmaterial – meistens Wolfram – treffen. Werden diese nun so abgelenkt, dass sie ein Körperteil, z. B. eine Hand, durchdringen, so werden sie durch dort befindliche Strukturen wie Haut, Muskeln und Knochen unterschiedlich stark absorbiert und somit abgeschwächt; Knochen weisen dabei die höchste Absorptionsrate auf. Die wieder austretende Strahlung belichtet dann eine photosensitive Platte hinter dem Körperteil, auf der stark abschwächende Strukturen hell und weniger abschwächende Strukturen dunkel erscheinen (siehe Abbildung 2.1).

Ein Nachteil von RÖNTGEN-Bildern ist die Überlagerung der einzelnen Absorptionswerte entlang der Strahlen, wodurch eine Projektion des dreidimensionalen Körpers auf eine zweidimensionale Fläche stattfindet, durch die Tiefeninformationen verloren gehen. Der zweite große Meilenstein der medizinischen Bildgebung, die *Computertomographie* (kurz CT), geht genau dieses Problem an. Sie wurde 1968 – nach Vorarbeiten durch Allan M. CORMACK – von Godfrey HOUNSFIELD vorgestellt und ermöglichte erstmals die dreidimensionale Darstellung eines Körpers. Im Gegensatz zur RÖNTGEN-Aufnahme, werden bei einem CT-Scan mehrere *Schichtbilder* erzeugt, die jeweils nur eine wenige Millimeter dünne Scheibe des Körpers darstellen. Diese kann man anschließend einzeln betrachten, oder zu einem dreidimensionalen Volumen zusammensetzen. Abbildung 2.2 zeigt drei aufeinanderfolgende Schichtbilder eines Kopfes.

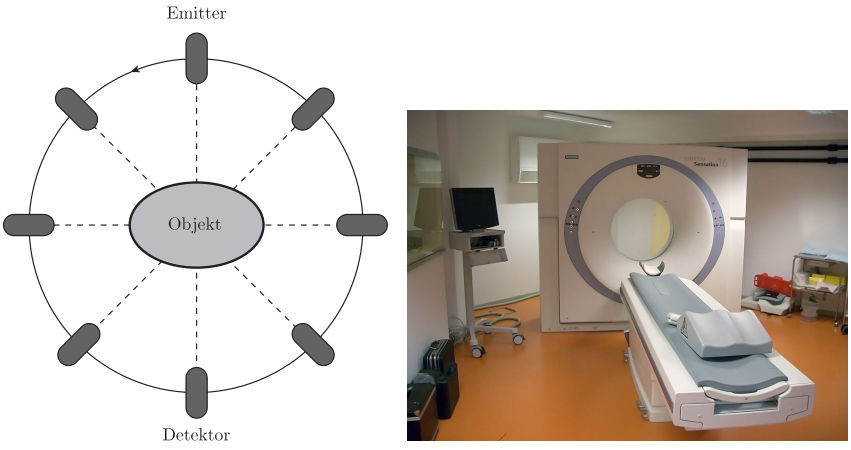
Abbildung 2.1: RÖNTGEN-Bild einer Hand.

(a) (b) (c)

Abbildung 2.2: Drei aufeinanderfolgende Schichtbilder eines CT-Scans des Kopfes.

Ein CT-Scanner besteht im Wesentlichen aus zwei Teilen: einem Tisch, auf dem sich der Patient – bzw. das zu scannende Objekt – befindet und dem sogenannten *Portal* (engl.: „gantry“), einem System aus Emittern und Detektoren, welches bei einem Scan um den Tisch rotiert (vgl. Abbildung 2.3a). Die Emitter schießen währenddessen RÖNTGEN-Strahlen durch den Körper, welche dabei abgeschwächt werden. Ein dem Emitter gegenüber liegender Detektor misst dann die Intensität der durchgelassenen Strahlung. Nach einer vollen Rotation kann aus den ermittelten Intensitäten ein Schichtbild berechnet werden, welches man sich als ein RÖNTGEN-Bild einer dünnen Scheibe des Körpers vorstellen kann. Anschließend wird der Tisch orthogonal zum Portal verschoben und die nächste Schicht kann ermittelt werden.

Die Berechnung der einzelnen Schichtbilder erfolgt mit Hilfe der RADON-*Transformation*, einer Integraltransformation, welche 1917 vom Mathematiker Johann RADON eingeführt wurde. Wird eine mathematische Funktion RADON-transformiert, so wird sie entlang einer Geraden integriert. Den resultierenden Wert nennt man RADON-*Transformierte* der Funktion. Aber auch der umgekehrte Weg ist möglich: die ursprünglichen Funktionswerte können mit einer *Rücktransformation* aus der RADON-Transformierten zurückgewonnen werden. Dieses Prinzip nutzt man bei der Computertomographie aus, um die Schichtbilder



(a) (b)

Abbildung 2.3: (a) Schematischer Aufbau eines CT-Scanners; (b) Foto eines CT-Scanners.

zu berechnen. Betrachtet man nämlich die Funktionsweise eines CT-Scanners genauer, so werden auch dort Integrale (die gemessenen Intensitäten) einer Funktion (das Schichtbild) entlang einer Geraden (der RÖNTGEN-Strahl) ermittelt. Eine RADON-Rücktransformation liefert dann das gewünschte Schichtbild, wobei der Wert in jedem Pixel ein Maß für die „Gewebedichte“ im Körper ist und in HOUNSFIELD-Einheiten angegeben wird.

2.2 Volumenrendering

Durch Stapelung der, mit Hilfe der Computertomographie gewonnenen, zweidimensionalen Schichtbilder, entsteht ein dreidimensionales Gitter, welches man als *Volumendatensatz*, oder einfach nur als *Volumen* bezeichnet. Es besteht aus äquidistanten Messpunkten, den sogenannten *Voxeln*, welche die HOUNSFIELD-Dichtewerte repräsentieren. Die Aufgabe der medizinischen Visualisierung ist es nun, das dreidimensionale Volumen auf eine zweidimensionale Ebene, z. B. einen Bildschirm, zu projizieren. Dies wird u. a. durch das *direkte Volumenrendering*[1] erreicht, welches im Folgenden näher erläutert werden soll.

2.2.1 Volumenrendering Pipeline

Das direkte Volumenrendering lässt sich in aufeinanderfolgende Schritte – die sogenannte *Volumenrendering Pipeline* – unterteilen. Nach [EHK+06] sind das:

- Abtasten (engl.: „sampling“),
- Interpolation,

[1] Es gibt auch das sogenannte *indirekte* Volumenrendering, auf das in diesem Text aber nicht weiter eingegangen wird.

- Klassifikation,
- Beleuchtung und
- Zusammensetzen (engl.: „compositing“).

Beim *Abtasten* werden die Dichtewerte an verschiedenen Stellen des Volumens herausgegriffen. Da sich diese Abtastpunkte (im Folgenden „Samples“ genannt) typischerweise von den Voxeln unterscheiden, werden sie im nachfolgenden Schritt der Pipeline über das Volumen interpoliert.[2] Hierdurch wird aus dem diskreten Volumen ein stetiges Skalarfeld

$$I\colon \mathbb{R}^3 \to \mathbb{R} \tag{2.1}$$

rekonstruiert, welches (stetige) Positionen auf Dichtewerte abbildet und im weiteren Verlauf dieser Arbeit stellvertretend für das Volumen verwendet wird. Die genaue Abtastmethode wird dabei durch den verwendeten Algorithmus festgelegt – in dieser Arbeit wird exemplarisch der populäre Ray-Casting Algorithmus erläutert (siehe Abschnitt 2.2.2).

Der nächste Schritt in der Pipeline ist die *Klassifikation*, durch die die Dichtewerte auf Farb- und Opazitätswerte (häufig RGBA-Werte[3]) abgebildet, und so z. B. für einen Bildschirm darstellbar, bzw. für das menschliche Auge sichtbar gemacht werden. Hierfür setzt man üblicherweise *Transferfunktionen* ein, welche in Abschnitt 2.2.3 näher erläutert werden. Die anschließende optionale *Beleuchtung* kann dazu benutzt werden, die dargestellten Bilder realer wirken zu lassen und die Tiefenwahrnehmung des Betrachters zu verbessern.

Der letzte Schritt der Pipeline ist schließlich die *Zusammensetzung* der klassifizierten Samples zu einem Pixelwert, welcher dann auf der Bildebene angezeigt wird. Dies wird üblicherweise durch gewichtetes Aufsummieren der RGBA-Werte der Samples erreicht. Die genaue Vorgehensweise wird auch hier durch den Algorithmus festgelegt.

2.2.2 Ray-Casting

Der *Ray-Casting* Algorithmus verwendet einen *bildbasierten* Ansatz (engl.: „image-based“), um das Volumen auf die Bildebene zu projizieren. Bildbasiert bedeutet, dass der Algorithmus einmal über alle Pixel der Bildebene iteriert und deren Wert berechnet, weswegen er heutzutage üblicherweise durch ein Shaderprogramm, welches direkt auf der Grafikhardware ausgeführt wird, realisiert wird.

Beim Ray-Casting wird von der Kamera aus ein Strahl durch jeden Pixel der Bildebene ausgesendet, der anschließend durch das Volumen läuft (vgl. Abbildung 2.4). Auf diesem Strahl werden dann solange Samples im Volumen genommen, bis der Strahl das Volumen wieder verlässt. Der Abstand der Samples wird *Sample-Abstand* genannt und ist ein Parameter zur Einstellung der Genauigkeit des Algorithmus. Je geringer der Sample-Abstand ist, desto mehr Samples werden pro Strahl genommen und desto genauer ist

[2] Meist durch eine trilineare Interpolation direkt auf der Grafikhardware.

[3] RGBA (Rot, Grün, Blau, Alpha) ist ein additives Farbmodell, in dem Farben durch Mischen der Grundfarben Rot, Grün und Blau entstehen. Das RGB-Tupel legt das Mischverhältnis fest, während der Alpha-Wert die Durchsichtigkeit der Farbe beschreibt.

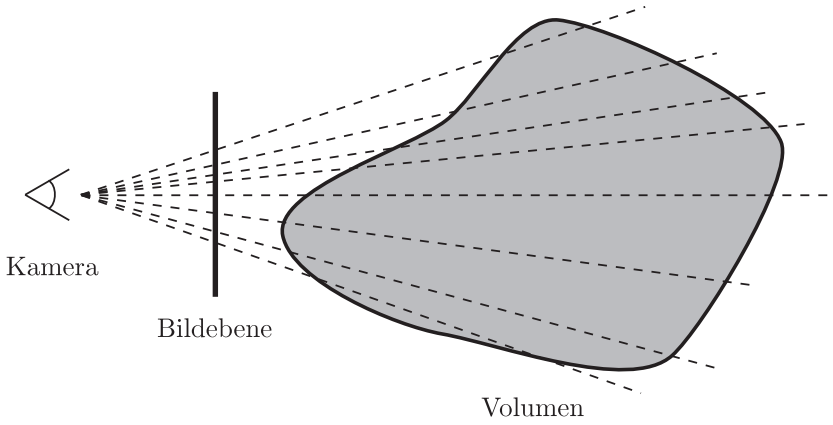


Abbildung 2.4: Prinzip von Ray-Casting.

der berechnete Pixelwert. Die Samples werden, gemäß des direkten Volumenrenderings, mit einer Transferfunktion in Farb- und Opazitätswerte umgewandelt und wie folgt von vorne nach hinten aufsummiert – bzw. zusammengesetzt (vgl. [EHK+06]):

$$C = C + (1 - \alpha)C_n \tag{2.2}$$

$$\alpha = \alpha + (1 - \alpha)\alpha_n. \tag{2.3}$$

Hierbei repräsentiert C den akkumulierten Farb- und α den akkumulierten Opazitätswert, sowie C_n und α_n den Farb- und Opazitätswert des aktuellen Samples. Dem Pixel, durch den der Strahl ausgesandt wurde, wird dann der endgültig berechnete Wert zugewiesen.

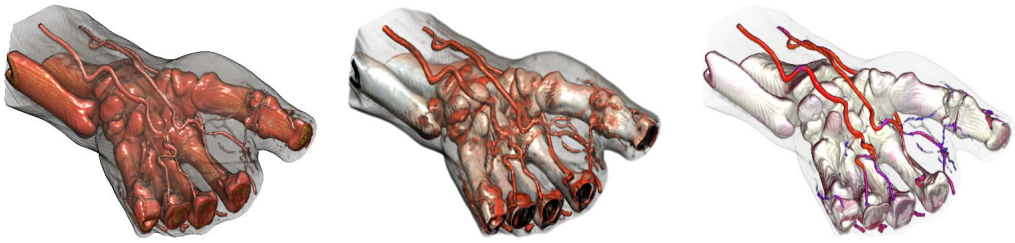
2.2.3 Transferfunktionen

Im Volumen ist jeder Voxelposition ein HOUNSFIELD-Dichtewert zugeordnet, welcher allerdings zunächst gar nichts über die endgültige Darstellung des Voxels aussagt. Eine (eindimensionale) *Transferfunktion* füllt diese Lücke, indem sie jeden Dichtewert auf ein RGBA-Tupel abbildet. Sie ist also nichts anderes als eine Funktion

$$TF\colon H \to [0,1] \times [0,1] \times [0,1] \times [0,1], \tag{2.4}$$

wobei H das Intervall der HOUNSFIELD-Werte repräsentiert. Dadurch ist es bei medizinischen Volumen möglich, verschiedene Gewebearten für den Menschen unterscheidbar zu machen, indem man die Transferfunktion so spezifiziert, dass sie den Dichtewerten jeder Gewebeart ein anderes RGBA-Tupel, d. h. eine andere Farbe, zuweist. Auch ein Blick auf *im* Volumen vorhandene Strukturen ist durch den Alphawert möglich. So wäre es für einen Kardiologen sinnvoll, Herzkranzgefäße in einer anderen Farbe darzustellen, als das Herz selbst, sowie unnötige Strukturen, wie z. B. Rippen und Lunge auszublenden.

Da sich die Dichtewerte verschiedener Gewebearten allerdings teilweise überlappen, ist mit einer eindimensionalen Transferfunktion nicht immer eine genaue Unterscheidung möglich. Mehrdimensionale Transferfunktionen können hier eine Lösung bieten. Bei



(a) (b) (c)

Abbildung 2.5: Volumenrendering einer Hand mit Hilfe unterschiedlicher Transferfunktionen: (a) eindimensional; (b) zweidimensional, Gradient; (c) zweidimensional, Größe (aus [CM08]).

einer mehrdimensionalen Transferfunktion wird einfach der Definitionsbereich erweitert. Hilfreiche Metriken sind z. B. der *Gradient* (Ableitung), mit dem Kanten genauer hervorgehoben werden können, oder die *Größe* von Strukturen im Volumen (vgl. Abschnitt 2.3). Abbildung 2.5 zeigt die Effekte einer eindimensionalen Transferfunktion, einer zweidimensionalen Transferfunktion, dessen zweite Metrik der Gradient ist, und einer zweidimensionalen Transferfunktion, dessen zweite Metrik die Größe ist, am Beispiel einer Hand im Vergleich.

2.3 Größenbasierte Hervorhebung von Strukturen

Wie schon Abbildung 2.5c zeigt, ist die Größe eine sinnvolle Metrik zur Hervorhebung einzelner Strukturen. Die dargestellten Blutgefäße können zwar nicht mit Hilfe ihres Dichtewertes vom restlichen Volumen unterschieden werden (vgl. Abbildung 2.5a), weisen dafür aber eine geringe Größe im Gegensatz zu den anderen Strukturen – wie z. B. Knochen – auf. Im Folgenden soll daher erläutert werden, wie Größe für den Computer erfassbar gemacht und diese Information dann zur Hervorhebung genutzt werden kann.

2.3.1 Scale-Space

Ein Volumen lässt sich – wie auch einfache zweidimensionale Bilder – grundsätzlich in eine Familie von Volumina aufteilen, die nur noch Strukturen enthalten, welche eine bestimmte Größe haben; dies wird *Scale-Space* des Volumens genannt. Er wurde erstmals 1983 von Witkin zur Erkennung von Merkmalen erwähnt und 1990 in der Dissertation von Lindeberg systematisch untersucht [Wit83, Lin90, Lin91].

Gegeben ein Volumen $I\colon \mathbb{R}^3 \to \mathbb{R}$, so ist der Scale-Space definiert als Funktion

$$I^{(\sigma)}\colon \mathbb{R}^3 \times \mathbb{R}^+ \to \mathbb{R}, \tag{2.5}$$

(a) (b) (c)

Abbildung 2.6: Mit einem GAUSS-Filter weichgezeichnete Kreise: (a) Originalbild ($\sigma = 0$); (b) $\sigma = 10$; (c) $\sigma = 30$.

welche durch Faltung[4] (engl.: „convolution“) des Volumens mit einem GAUSS-*Filter*

$$g(\boldsymbol{x}, \sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)^3}}\, e^{-\frac{\|\boldsymbol{x}\|^2}{2\sigma^2}} \tag{2.6}$$

der Größe σ entsteht:

$$I^{(\sigma)}(\boldsymbol{x}) = g(\boldsymbol{x}, \sigma) * I(\boldsymbol{x}). \tag{2.7}$$

Dadurch entsteht eine *isotrope Diffusion*, d. h. eine in alle Richtungen gleichmäßig wirkende Weichzeichnung der Voxel, wodurch die im Volumen enthaltenen Strukturen bei wachsender Größe σ – welche im Folgenden als *Größe des Scale-Space* bezeichnet wird – nach und nach verschwinden; ist $\sigma = 0$, so erhält man das Originalvolumen. Abbildung 2.6 zeigt dies am Beispiel von verschieden großen Kreisen. Je größer σ ist, desto weniger Kreise sind erkennbar.

Bei der größenbasierten Erkennung von Merkmalen werden typischerweise mehrere aufeinanderfolgende Größen des Scale-Space betrachtet. Dies resultiert aus der Tatsache, dass die verwendeten Metriken extremal werden, wenn die Größe des Scale-Space der Größe des Merkmals entspricht. Um den dadurch entstehenden Berechnungsaufwand zu minimieren, kann man sich die Halbgruppenstruktur

$$g(\cdot, \sigma_1) * g(\cdot, \sigma_2) = g(\cdot, \sigma_1 + \sigma_2) \tag{2.8}$$

des GAUSS-Filters zu Nutze machen und den Scale-Space wie folgt iterativ berechnen:

$$I^{(\sigma)}(\boldsymbol{x}) = g(\boldsymbol{x}, 1) * I^{(\sigma-1)}(\boldsymbol{x}), \quad I^{(0)}(\boldsymbol{x}) = I(\boldsymbol{x}). \tag{2.9}$$

[4] Die Faltung zweier Funktionen $f, g\colon \mathbb{R}^3 \to \mathbb{R}$ ist definiert als

$$(f * g)(\boldsymbol{x}) = \int_{\mathbb{R}^3} f(\boldsymbol{\xi}) g(\boldsymbol{x} - \boldsymbol{\xi})\, d\boldsymbol{\xi}.$$

2.3.2 Größenbasierte Transferfunktionen

Correa und Ma beschreiben in ihrer Arbeit über *größenbasierte Transferfunktionen* (kurz SBTF) ein Verfahren, um jedem Voxel im Volumen die Größe der ihn umgebenden Struktur zuzuordnen [CM08]. Dieses lässt sich in folgende drei Schritte unterteilen:

1. Erstellung des Scale-Space,
2. Größenerkennung und
3. Rückprojektion.

Man erhält dadurch ein Skalarfeld, das sogenannte *Scale-Field*

$$S\colon \mathbb{R}^3 \to \mathbb{R}^+, \tag{2.10}$$

welches jeder Position $\boldsymbol{x} = (x\ y\ z)^{\mathrm{T}}$ im Volumen eine Größe t zuordnet und im Anschluss als Definitionsbereich einer Transferfunktion benutzt werden kann. Ein einfaches Beispiel ist eine zweidimensionale Transferfunktion, die Dichtewert und Größe kombiniert und auf ein RGBA-Tupel abbildet (vgl. auch Abbildung 2.5c):

$$TF\colon H \times S \to [0,1] \times [0,1] \times [0,1] \times [0,1]. \tag{2.11}$$

Die *Erstellung des Scale-Space* erfolgt, wie in Abschnitt 2.3.1 beschrieben, durch eine Faltung des Volumens mit einem Gauss-Filter[5] der Größe $\sigma = t$, wobei $t = t_{\min}, \ldots, t_{\max}$. Das Intervall $[t_{\min}, t_{\max}]$ kann beispielsweise der Bereich zwischen der minimalen und maximalen Größe von Gefäßstrukturen sein.

Die *Größenerkennung* erfolgt mit Hilfe des Laplace-Operators

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}, \tag{2.12}$$

welcher auf jeden Voxel $\boldsymbol{x}$ des weichgezeichneten Volumens $I^{(\sigma)}$ angewandt und mit der aktuellen Größe des Scale-Space normalisiert wird:

$$\sigma \Delta I^{(\sigma)} = \sigma \left(\frac{\partial^2 I^{(\sigma)}}{\partial x^2} + \frac{\partial^2 I^{(\sigma)}}{\partial y^2} + \frac{\partial^2 I^{(\sigma)}}{\partial z^2} \right). \tag{2.13}$$

Dadurch entsteht eine Metrik, um *kugelförmige* Strukturen zu erkennen. Ist nämlich ein Voxel $\boldsymbol{x}$ der Mittelpunkt einer Kugel, so nimmt Gleichung (2.13) dort ein Maximum an – und zwar sowohl räumlich gesehen, als auch auf die aktuelle Größe des Scale-Space bezogen –, wenn die Größe des Scale-Space annähernd mit dem Radius der Kugel übereinstimmt. Fasst man alle Maxima zu einer Menge $\{(\boldsymbol{x}_i, t_i)\} \subseteq I^{(\sigma)}$ zusammen, wobei $\boldsymbol{x}_i$ die Position und t_i die Größe der erkannten Kugeln bezeichnet, so erhält man die *Größenabstraktion* (engl.: „scale-abstraction“) des Volumens.

[5] Genauer betrachtet verwenden sie einen modifizierten Gauss-Filter, der ein *anisotropisches* Verhalten hat. Dadurch wird verhindert, dass nah beieinander liegende Strukturen schon bei kleinen Werten von σ zu einer einzigen Struktur verschmelzen.

Die diskreten Punkte der Größenabstraktion müssen nun noch in das stetige Skalarfeld S umgerechnet werden; dies ist die Aufgabe der *Rückprojektion*. Dazu wird angenommen, dass alle Voxel innerhalb einer Kugel mit Radius t und Mittelpunkt $\boldsymbol{x}$ eines erkannten Maximums $(\boldsymbol{x}, t)$ ebenfalls die Größe t haben. Etwaige Zweideutigkeiten durch Überlappung der erkannten Kugeln werden durch eine Interpolation aufgelöst.

Ein Nachteil des Verfahrens ist die Beschränkung auf die Erkennung von kugelförmigen Strukturen im Volumen. Obwohl dadurch einer Vielzahl von Objekten in Volumendatensätzen die korrekte Größe zugeordnet werden kann, so ist das Verfahren bei feinen röhrenartigen Strukturen zu ungenau, da es nicht die besondere Topologie von Röhren beachtet. Im Gegensatz zu einer Kugel, liegen die Maxima bei einer Röhre nämlich auf einer Linie – der sogenannten *Mittellinie* – und sind nicht in einem einzigen Punkt konzentriert. Im nächsten Kapitel werden daher zwei Verfahren zur Erkennung von röhrenartigen Strukturen vorgestellt, die dann den Schritt der Größenerkennung ersetzen.

3 Erkennung von Gefäßen

Probleme kann man niemals mit derselben Denkweise lösen, durch die sie entstanden sind.

(Albert Einstein)

3.1 Der Filter von Frangi et al.

Das Verfahren von FRANGI et al. [FFN+98] dient in erster Linie zur Hervorhebung von Gefäßstrukturen in Volumendatensätzen. Dazu nutzt es aus, dass die Eigenvektoren der HESSE-*Matrix* in Richtung der wesentlichen Krümmung der, den aktuell betrachteten Voxel umgebenden, Struktur zeigen, und die zugehörigen Eigenwerte ein Maß für die Größe dieser Krümmung sind. Die HESSE-Matrix in einem Punkt $\boldsymbol{x} = (x \;\; y \;\; z)^{\mathrm{T}}$ des Volumens I ist dabei wie folgt definiert:

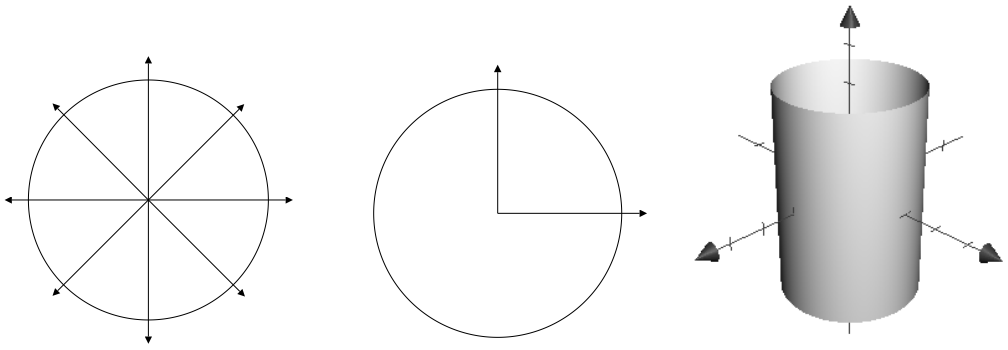
$$\boldsymbol{H}(\boldsymbol{x}) = \left[\frac{\partial^2 I(\boldsymbol{x})}{\partial a \partial b}\right] = \begin{pmatrix} \frac{\partial^2 I(\boldsymbol{x})}{\partial x^2} & \frac{\partial^2 I(\boldsymbol{x})}{\partial x \partial y} & \frac{\partial^2 I(\boldsymbol{x})}{\partial x \partial z} \\ \frac{\partial^2 I(\boldsymbol{x})}{\partial y \partial x} & \frac{\partial^2 I(\boldsymbol{x})}{\partial y^2} & \frac{\partial^2 I(\boldsymbol{x})}{\partial y \partial z} \\ \frac{\partial^2 I(\boldsymbol{x})}{\partial z \partial x} & \frac{\partial^2 I(\boldsymbol{x})}{\partial z \partial y} & \frac{\partial^2 I(\boldsymbol{x})}{\partial z^2} \end{pmatrix} \quad \forall a, b \in \{x, y, z\}\,. \tag{3.1}$$

Aufgrund von

$$\frac{\partial^2 I}{\partial a \partial b} = \frac{\partial^2 I}{\partial b \partial a} \quad \forall a, b \in \{x, y, z\}\,,\; a \neq b, \tag{3.2}$$

ist sie symmetrisch. Da sie zudem nur reelle Werte enthält, bilden die Eigenvektoren ein *Orthonormalsystem*, d. h. ein System von paarweise orthogonalen und normierten Vektoren, und die Eigenwerte sind stets reell.

Aus diesen Eigenschaften kann man nun auf im Volumen vorhandene Strukturen schließen. Dazu seien λ_1, λ_2 und λ_3 die Eigenwerte der HESSE-Matrix, sowie $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ und $\boldsymbol{v}_3$ die zugehörigen Eigenvektoren. Da Blutgefäße röhrenartige Strukturen sind, soll zunächst eine ideale Röhre betrachtet werden. Die Querschnittsfläche einer solchen Röhre ist ein Kreis und weist deshalb ein hohes Maß an Krümmung in *radialer* Richtung auf (siehe Abbildung 3.1a). Diese radial nach außen gerichteten Vektoren können allerdings auch durch zwei geeignete orthogonale Basisvektoren dargestellt werden (siehe Abbildung 3.1b), welche erstaunlicherweise gerade zwei der Eigenvektoren sind – z. B. $\boldsymbol{v}_1$ und $\boldsymbol{v}_2$. Da die Eigenvektoren ein Orthonormalsystem bilden, muss der dritte Eigenvektor $\boldsymbol{v}_3$ in Längsrichtung der Röhre zeigen (siehe Abbildung 3.1c).



(a) (b) (c)

Abbildung 3.1: Richtung der wesentlichen Krümmung bei einer Röhre.

λ_1	λ_2	λ_3	Struktur
H	0	0	plattenförmig
H	H	0	röhrenförmig
H	H	H	kugelförmig

Tabelle 3.1: Verhältnisse der Eigenwerte für verschiedenartige Strukturen. Dabei steht „H“ für einen betragsmäßig großen Wert und „0“ für einen Wert nahe Null.

Aufgrund der verschwindenden Krümmung in Längsrichtung folgt, dass $|\lambda_3| \approx 0$ ist, womit sich folgende Ordnung der Eigenwerte und der zugehörigen Eigenvektoren anbietet:

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \,. \tag{3.3}$$

Eine röhrenartige Struktur lässt sich dann wie folgt charakterisieren:

$$\begin{aligned} \lambda_1 &\approx \lambda_2 \\ |\lambda_2| &\gg |\lambda_3| \\ |\lambda_3| &\approx 0. \end{aligned} \tag{3.4}$$

Diese Überlegungen kann man nun auch auf Platten und Kugeln ausweiten. Die Verhältnisse der Eigenwerte für diese drei Strukturen sind in Tabelle 3.1 zusammengefasst und in Abbildung 3.2 grafisch veranschaulicht.

Um Gefäßstrukturen von anderen Strukturen im Volumen unterscheiden zu können, definieren Frangi et al. zwei Verhältnisse. Das erste Verhältnis misst die Abweichung von kugelartigen Strukturen:

$$\mathcal{R}_\mathcal{B} = \frac{\text{Volume}/(4\pi/3)}{(\text{Largest Cross Section Area}/\pi)^{3/2}} = \frac{|\lambda_3|}{\sqrt{|\lambda_1\lambda_2|}}. \tag{3.5}$$

Da für kugelartige Strukturen $|\lambda_1| \approx |\lambda_2| \approx |\lambda_3|$ gilt, nimmt es in diesem Fall ein Maximum an. Für platten- und röhrenartige Strukturen – bei denen $|\lambda_3| \approx 0$ ist – nimmt es dagegen einen sehr kleinen Wert an.

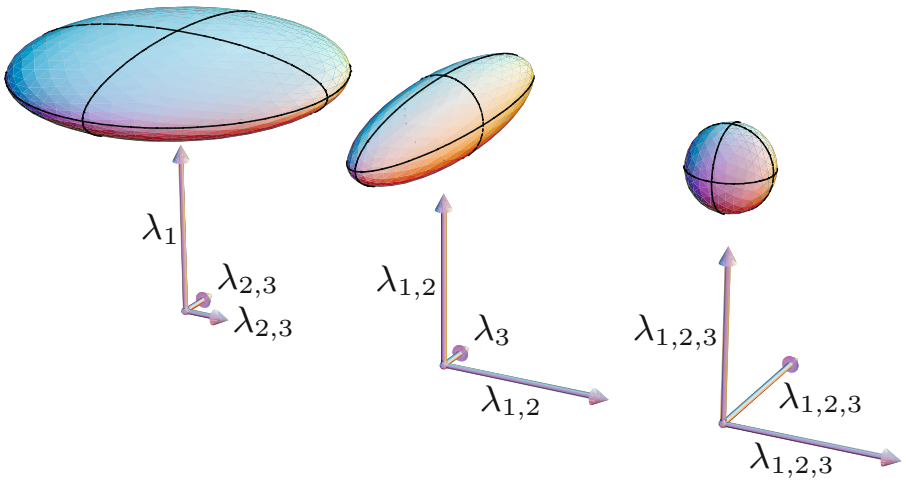


Abbildung 3.2: Eigenvektoren (Richtung) und Eigenwerte (Länge) der HESSE-Matrix für platten-, röhren- und kugelartige Strukturen (aus [BVAS+07]).

Das zweite Verhältnis dient zur Unterscheidung von platten- und röhrenartigen Strukturen:

$$\mathcal{R}_{\mathcal{A}} = \frac{(\text{Largest Cross Section Area})/\pi}{(\text{Largest Axis Semi-length})^2} = \frac{|\lambda_2|}{|\lambda_1|}. \tag{3.6}$$

Da $|\lambda_2|$ bei plattenartigen Strukturen annähernd gleich Null ist, gilt bei diesen $\mathcal{R}_{\mathcal{A}} \approx 0$. Für kugel- und röhrenartige Strukturen nimmt es dagegen ein Maximum an.

Ein Problem bei diesem Ansatz entsteht durch Rauschen, wodurch Hintergrundvoxel – in Regionen ohne Gefäßstrukturen – fälschlicherweise als „röhrenartig“ erkannt werden können. Aus diesem Grund schlagen FRANGI et al. eine Metrik vor, die Hintergrundvoxel von den eigentlichen Strukturen unterscheiden kann. Die Grundlage dafür bietet die FROBENIUS-*Norm* der HESSE-Matrix, welche ein Maß für die „Gesamtkrümmung“ der betrachteten Struktur ist. Die FROBENIUS-Norm hat den Vorteil, dass man sie bei reellen und symmetrischen Matrizen sehr einfach über die Eigenwerte berechnen kann. Dies führt zur Metrik

$$\mathcal{S} = \|\boldsymbol{H}\|_{\mathrm{F}} = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}. \tag{3.7}$$

Sie wird klein für Hintergrundvoxel, da diese – aufgrund des geringen Kontrasts – eine geringe Krümmung und damit kleine Eigenwerte aufweisen; und groß für Voxel, die zu Strukturen im Volumen gehören – und damit einen hohen Kontrast aufweisen.

Alle drei Größen können nun zur sogenannten *Vesselness-Funktion* $\mathcal{V}_0$ zusammengefügt werden. Diese erlaubt eine wahrscheinlichkeitsartige Abschätzung der „Gefäßartigkeit“ eines Voxels. Sie wird nur für solche Voxel maximal, bei denen auch die Größen $\mathcal{R}_{\mathcal{A}}$ und

$\mathcal{S}$ maximal sind, sowie die Größe $\mathcal{R}_\mathcal{B}$ minimal ist – also für röhrenartige Strukturen.

$$\mathcal{V}_0(\boldsymbol{x}, s) = \begin{cases} 0 & \text{falls } \lambda_1 > 0, \text{ oder } \lambda_2 > 0, \\ \left(1 - e^{-\frac{\mathcal{R}_\mathcal{A}^2}{2\alpha^2}}\right)\left(e^{-\frac{\mathcal{R}_\mathcal{B}^2}{2\beta^2}}\right)\left(1 - e^{-\frac{\mathcal{S}^2}{2c^2}}\right) & \text{sonst.} \end{cases} \tag{3.8}$$

Dabei bezeichnet $\boldsymbol{x}$ die Koordinaten des aktuell betrachteten Voxels und s die aktuelle Größe des Scale-Space. Die Werte α, β und c steuern die Gewichtung der einzelnen Metriken. Da Voxel, die zu Strukturen gehören – also insbesondere zu Gefäßen – aufgrund ihrer Kontrastierung in CT-Volumen negative Eigenwerte haben, werden Voxel, deren beiden größten Eigenwerte positiv sind, direkt von der Metrik ausgeschlossen.

Wie schon durch den Parameter s angedeutet, wird die Vesselness-Funktion im Scale-Space ausgewertet. Dazu muss vorher der Scale-Space der Hesse-Matrix definiert werden. Dieser ist aber nichts anderes als die Hesse-Matrix eines weichgezeichneten Volumens. Bezeichnet man dieses wie in Abschnitt 2.3.1 mit $I^{(\sigma)}$, wobei σ die aktuelle Größe des Gauss-Filters ist, so folgt

$$\boldsymbol{H}^{(\sigma)}(\boldsymbol{x}) = \sigma^{2\gamma}\left[\frac{\partial^2 I^{(\sigma)}(\boldsymbol{x})}{\partial a \partial b}\right] \quad \forall a, b \in \{x, y, z\}\,. \tag{3.9}$$

Der Term $\sigma^{2\gamma}$, welcher von Lindeberg eingeführt wurde, ist für die Normalisierung der zweiten Ableitungen zuständig [Lin96, Lin98]. Dadurch ist es möglich bestimmte Größen stärker zu gewichten als andere. Sollen dagegen alle betrachteten Größen gleich behandelt werden – so wie in dieser Arbeit vorausgesetzt –, so muss $\gamma = 1$ gelten; es herrscht dann eine *Größeninvarianz*.

Nun wird $\mathcal{V}_0$ für verschiedene Werte von $\sigma = s$ ausgewertet und jeweils das Maximum gewählt. Die Idee dahinter ist, dass die Vesselness-Funktion für Voxel eines Gefäßes einen maximalen Wert zurückliefert, wenn die Größe des Scale-Space annähernd der Größe dieses Gefäßes entspricht. Dies führt zur finalen Definition der Vesselness-Funktion

$$\mathcal{V}_0(\boldsymbol{x}) = \max_{s_{\min} \leq s \leq s_{\max}} \mathcal{V}_0(\boldsymbol{x}, s), \tag{3.10}$$

wobei $s_{\min}$ und $s_{\max}$ die minimalen und maximalen Größen des Scale-Space sind, die betrachtet werden sollen.

3.2 Der Filter von Pock et al.

Pock et al. [PBB05] schlagen ein adaptives Verfahren zur Erkennung der Mittellinien von röhrenartigen Strukturen im Scale-Space vor, welches nicht nur den aktuellen Voxel selbst, sondern auch eine Umgebung um diesen herum betrachtet. Die Umgebung ist dabei ein Kreis in der von den beiden größten Eigenvektoren[1] aufgespannten Ebene, dessen Radius r die Größe des aktuellen Scale-Space ist. Legt man die gleiche Sortierung der Eigenwerte und Eigenvektoren wie in Abschnitt 3.1 zugrunde, so ist die Ebene folglich durch die Vektoren $\boldsymbol{v}_1$ und $\boldsymbol{v}_2$ festgelegt (vgl. auch Abbildung 3.3).

[1] „Größter“ Eigenvektor steht ab jetzt abkürzend für den Eigenvektor mit dem größten Eigenwert.

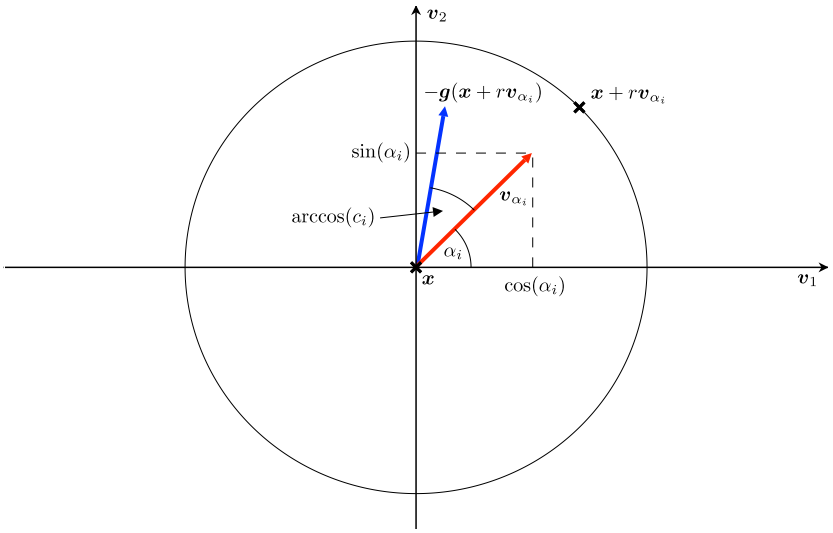


Abbildung 3.3: Grafische Veranschaulichung des Verfahrens von Pock et al.

Auf diesem Kreis werden nun N äquidistante Punkte gewählt. Dazu wird der Winkel $\alpha_i = (2\pi i)/N$, für $i = 0 \dots N-1$ definiert. Die radiale Richtung, also der Vektor vom aktuellen Voxel – der sich an der Stelle $\boldsymbol{x}$ befinde – zum aktuellen Punkt auf dem Kreis ist dann

$$\boldsymbol{v}_{\alpha_i} = \cos(\alpha_i)\boldsymbol{v}_1 + \sin(\alpha_i)\boldsymbol{v}_2, \tag{3.11}$$

und der Punkt auf dem Kreis befindet sich an der Stelle $\boldsymbol{x} + r\boldsymbol{v}_{\alpha_i}$. Da $\boldsymbol{v}_1$ und $\boldsymbol{v}_2$ normiert und orthogonal zu einander sind, gilt

$$\begin{aligned} \|\boldsymbol{v}_{\alpha_i}\|^2 &= \cos^2(\alpha_i)\,\|\boldsymbol{v}_1\|^2 + \sin^2(\alpha_i)\,\|\boldsymbol{v}_2\|^2 + 2\cos(\alpha_i)\sin(\alpha_i)\,\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 \\ &= \cos^2(\alpha_i) + \sin^2(\alpha_i) \\ &= 1, \end{aligned} \tag{3.12}$$

und damit $\|\boldsymbol{v}_{\alpha_i}\| = 1$.

Ist $\boldsymbol{B}(\boldsymbol{x})$ der Gradient – also die Ableitung – im Punkt $\boldsymbol{x}$, so bezeichne $b(\boldsymbol{x}) = \|\boldsymbol{B}(\boldsymbol{x})\|$ dessen Länge und $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{B}(\boldsymbol{x})/\|\boldsymbol{B}(\boldsymbol{x})\|$ dessen Richtung. Betrachtet man nun einen Punkt auf dem Rand einer röhrenartigen Struktur in einem CT-Volumen, dann zeigt der Gradient stets in Richtung der Mittellinie der Röhre. Bei einer idealen Röhre gilt sogar $\boldsymbol{v}_{\alpha_i} = -\boldsymbol{g}(\boldsymbol{x})$, oder äquivalent ausgedrückt

$$-\boldsymbol{g}(\boldsymbol{x}) \cdot \boldsymbol{v}_{\alpha_i} = 1. \tag{3.13}$$

Eine weitere Beobachtung ist, dass $b(\boldsymbol{x})$ extremal wird, wenn $\boldsymbol{x}$ ein Punkt auf dem Rand einer idealen Röhre ist.

Der Vektor $\boldsymbol{x}$ bezeichne wieder die Koordinaten des aktuellen Voxels. Pock et al. schlagen nun eine Metrik vor – die sogenannte *Initial Medialness* –, welche die Zugehörigkeit eines Voxels zur Mittellinie einer Röhre misst. Dazu sei zunächst $N = \lfloor 2\pi r + 1 \rfloor$ die

Anzahl der äquidistanten Punkte auf dem Kreis. Für den i-ten Punkt ist die sogenannte „boundariness“ dann wie folgt definiert:

$$b_i = b(\boldsymbol{x} + r\boldsymbol{v}_{\alpha_i})\, c_i^n. \tag{3.14}$$

Je größer b_i ist, desto wahrscheinlicher ist es, dass sich im Abstand r um den aktuellen Voxel die Kante einer Röhre befindet.

Die Größe b_i setzt sich dabei aus zwei Werten zusammen. Der Erste ist die Länge des Gradienten im i-ten Punkt auf dem Kreis. Wie weiter oben schon festgestellt wurde, ist diese umso größer, je näher der i-te Punkt einer Kante ist, und extremal wenn r genau dem Abstand der Kante von $\boldsymbol{x}$ entspricht. Damit nun aber nicht alle Kanten im Volumen als Röhrenkanten erkannt werden, wird die Länge des Gradienten noch mit der sogenannten „circularity“ c_i multipliziert. Sie ist ein Maß für die Kreisförmigkeit einer Kante, und wie folgt definiert:

$$c_i = \begin{cases} -\boldsymbol{g}(\boldsymbol{x} + r\boldsymbol{v}_{\alpha_i}) \cdot \boldsymbol{v}_{\alpha_i} & \text{falls } -\boldsymbol{g}(\boldsymbol{x} + r\boldsymbol{v}_{\alpha_i}) \cdot \boldsymbol{v}_{\alpha_i} > 0, \\ 0 & \text{sonst.} \end{cases} \tag{3.15}$$

Laut Gleichung (3.13) wird c_i maximal, wenn der Gradient im i-ten Punkt und der Vektor vom Voxel zum i-ten Punkt parallel sind – es sich also um die Kante einer Röhre handelt. Sind sie nicht parallel, so ist $c_i < 1$. Dieser Effekt wird durch den Exponenten n verstärkt, sodass b_i für nicht kreisförmige Kanten schnell klein wird.

Mit dieser Vorarbeit kann nun die Initial Medialness R_0^+ definiert werden. Dazu wird einfach der Durchschnittswert aller b_i gebildet:

$$R_0^+(\boldsymbol{x}, r) = \frac{1}{N} \sum_{i=0}^{N-1} b_i. \tag{3.16}$$

Um eine noch genauere Erkennung von Röhren zu gewährleisten, wird in einem weiteren Schritt auch die Symmetrie betrachtet. Dazu wird die Varianz der b_i errechnet:

$$s^2(\boldsymbol{x}, r) = \frac{1}{N} \sum_{i=0}^{N-1} (b_i - R_0^+(\boldsymbol{x}, r))^2. \tag{3.17}$$

Mit ihr wird eine zweite Metrik – die sogenannte *Symmetry Confidence* $\mathcal{S}$ – erstellt, mit der R_0^+ multiplikativ gewichtet wird.

$$\mathcal{S}(\boldsymbol{x}, r) = 1 - \frac{s^2(\boldsymbol{x}, r)}{R_0^+(\boldsymbol{x}, r)^2}. \tag{3.18}$$

Je geringer die Varianz der b_i ist, desto symmetrischer ist die betrachtete Struktur, und desto größer wird $\mathcal{S}$. Bei einer idealen Röhre gilt $\mathcal{S} = 1$. Dies führt zur finalen Metrik, der *Symmetry Constrainted Medialness*

$$R^+(\boldsymbol{x}, r) = R_0^+(\boldsymbol{x}, r)\, \mathcal{S}(\boldsymbol{x}, r), \tag{3.19}$$

welche die „Röhrenartigkeit“ eines Voxels misst.

Wie anfangs schon erwähnt, verwendet auch dieses Verfahren den Scale-Space zur Berechnung der Metriken. Im Gegensatz zum Ansatz von FRANGI et al. verwendet es allerdings *zwei* verschiedene Scale-Spaces. Dies ist zum einen der Scale-Space der HESSE-Matrix, dessen Größenparameter mit $\sigma_{\boldsymbol{H}}$ bezeichnet wird; und zum anderen der Scale-Space für die Kantenerkennung („boundariness“) – in dem die Gradientenberechnung erfolgt –, dessen Größenparameter mit $\sigma_{\boldsymbol{B}}$ bezeichnet wird. Die beiden Parameter genügen dabei dem Zusammenhang

$$\sigma_{\boldsymbol{B}} = \sigma_{\boldsymbol{H}}^{\eta}, \tag{3.20}$$

wobei $\eta \in [0, 1]$ von der Menge des Rauschens im Volumen abhängig ist. Die eigentlichen Scale-Spaces sind dann definiert als

$$\boldsymbol{H}(\boldsymbol{x}) = \sigma_{\boldsymbol{H}}^{2\gamma} \left[\frac{\partial^2 I^{(\sigma_{\boldsymbol{H}})}(\boldsymbol{x})}{\partial a \partial b} \right] \quad \forall a, b \in \{x, y, z\} \tag{3.21}$$

$$\boldsymbol{B}(\boldsymbol{x}) = \sigma_{\boldsymbol{B}}^{\gamma} \nabla I^{(\sigma_B)}(\boldsymbol{x}), \tag{3.22}$$

wobei $\sigma_{\boldsymbol{H}}^{2\gamma}$ und $\sigma_{\boldsymbol{B}}^{\gamma}$ wieder für die Normalisierung zuständig sind,[2] und

$$\nabla I(\boldsymbol{x}) = \left(\frac{\partial I(\boldsymbol{x})}{\partial x} \;\; \frac{\partial I(\boldsymbol{x})}{\partial y} \;\; \frac{\partial I(\boldsymbol{x})}{\partial z} \right)^{\mathrm{T}} \tag{3.23}$$

den Gradienten im Punkt $\boldsymbol{x}$ des Volumens I bezeichnet.

Zusätzlich zu der in Gleichung (3.19) definierten Symmetry Constrainted Medialness wird, ähnlich wie im Verfahren von FRANGI et al., ein Schwellwert definiert, welcher Hintergrundvoxel unterdrücken soll. Dazu nutzt man aus, dass die Länge der Gradienten in der Querschnittsfläche einer idealen Röhre zur Mittellinie hin immer kleiner wird und auf der Mittellinie schließlich verschwindet (siehe Abbildung 3.4). Dies führt zur *Adaptive Threshold*-Metrik

$$R^{-}(\boldsymbol{x}, r) = \sigma_{\boldsymbol{H}}^{\gamma} \left\| \nabla I^{(\sigma_{\boldsymbol{H}})}(\boldsymbol{x}) \right\|, \tag{3.24}$$

welche die Norm des *zentralen* Gradienten – das ist der Gradient im aktuellen Voxel $\boldsymbol{x}$ –, im Scale-Space der HESSE-Matrix benutzt. Es wird hier derselbe Scale-Space verwendet, da sowohl die HESSE-Matrix als auch der zentrale Gradient Informationen über den aktuell betrachteten Voxel sind, während der Gradient für die Berechnung der „boundariness“ immer in einer Umgebung außerhalb des aktuellen Voxels, und damit auch in einem anderen Scale-Space berechnet wird.

Man kombiniert nun, für $\sigma_{\boldsymbol{H}} = r$, die beiden Metriken R^{+} und R^{-} wie folgt zur Metrik

$$R(\boldsymbol{x}, r) = \begin{cases} R^{+}(\boldsymbol{x}, r) - R^{-}(\boldsymbol{x}, r) & \text{falls } R^{+}(\boldsymbol{x}, r) > R^{-}(\boldsymbol{x}, r), \\ 0 & \text{sonst,} \end{cases} \tag{3.25}$$

welche man dann für verschiedene Werte von r auswertet und schließlich das Maximum wählt:

$$R(\boldsymbol{x}) = \max_{r_{\min} \leq r \leq r_{\max}} R(\boldsymbol{x}, r). \tag{3.26}$$

[2] Es sei hier noch einmal betont, dass in dieser Arbeit $\gamma = 1$ vorausgesetzt wird.

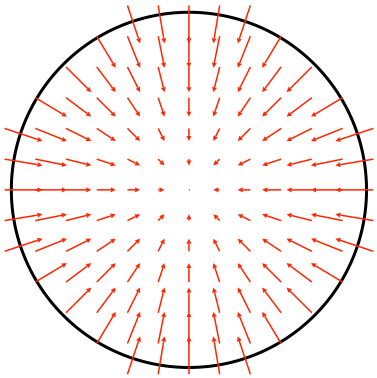


Abbildung 3.4: Länge der Gradienten in der Querschnittsfläche einer Röhre.

3.3 Größenabstraktion und Rückprojektion

Beide soeben vorgestellten Verfahren liefern bisher Metriken, die jedem Voxel im Volumen einen Wert zuweisen, der ein Maß für die Wahrscheinlichkeit ist, dass dieser zu einer röhrenartigen Struktur – und damit zu einem Gefäß – gehört. Um diese Verfahren nun auf den Ansatz von Correa und Ma übertragen zu können, muss aus den Metriken zunächst eine Größenabstraktion berechnet werden und anschließend eine Rückprojektion erfolgen.

Die Größenabstraktion entsteht – ähnlich wie bei der Verwendung des Laplace-Operators – durch Extraktion der lokalen Maxima der jeweiligen Metrik. Wird diese allgemein mit M bezeichnet und sind $\boldsymbol{x}$ ein Voxel und $\boldsymbol{v}_1$ und $\boldsymbol{v}_2$ die größten Eigenvektoren der Hesse-Matrix, so sind nur solche Voxel lokale Maxima, die die Bedingungen

$$M(\boldsymbol{x}) \geq M(\boldsymbol{x} \pm \boldsymbol{v}_1) \quad \text{und} \quad M(\boldsymbol{x}) \geq M(\boldsymbol{x} \pm \boldsymbol{v}_2) \tag{3.27}$$

erfüllen [PSS+03]. Zusammen mit der Größe r, bei der sie gefunden werden, bilden sie dann die Größenabstraktion $\{(\boldsymbol{x}_i, r_i)\} \subseteq I^{(\sigma)}$ des Volumens.

Für jedes gefundene Maximum wird schließlich eine Rückprojektion durchgeführt. Dazu wird allen Voxeln innerhalb einer Kugel mit dem Radius r_i und dem Mittelpunkt $\boldsymbol{x}_i$ die Größe r_i zugewiesen. Können einem Voxel aufgrund von Überlappungen zwei Größen zugeordnet werden, so wird das jeweilige Maximum genommen. Ausreißer, d. h. maximale Voxel deren 26 Nachbarschaftsvoxel selbst keine Maxima sind, müssen hiervon natürlich ausgeschlossen sein.

Diese Vorgehensweise wird besonders dann gute Ergebnisse erzielen, wenn die erkannten Maxima genau auf der Mittellinie der Röhre liegen. In welchem Maß die vorgestellten Verfahren dies erreichen, wird in Kapitel 5 untersucht.

4 Implementation

Die meisten Probleme entstehen bei ihrer Lösung.

(Leonardo da Vinci)

In diesem Kapitel soll kurz die Implementation der Filter erläutert werden. Dazu wird zunächst die Software *VolumeStudio2* mitsamt ihrer Plugin-Architektur vorgestellt. Nach einem Abschnitt über die Implementation der verwendeten mathematischen Operationen folgt dann die Umsetzung der Filter-Algorithmen und der Rückprojektion.

4.1 VolumeStudio und Plugins

VolumeStudio ist eine Software zur Visualisierung von medizinischen Volumendaten, die seit 2006 an der Universität Paderborn entwickelt wird, und nach einer Umstrukturierung der Architektur im Jahr 2010 nun in der zweiten Version (*VolumeStudio2*) vorliegt. Sie ist in der Programmiersprache C++ geschrieben und verwendet die von Nokia bereitgestellte Qt-Bibliothek[1]. *VolumeStudio2* ist vollständig Plugin-basiert, was eine einfache Erweiterung der bestehenden Funktionalität ermöglicht. Neu in dieser Version ist das Pipeline-Konzept, welches eine flexiblere Nutzung der einzelnen Plugins erlaubt, anstatt die Verarbeitung und Verarbeitungsreihenfolge fest vorzuschreiben. So ist es dem Entwickler und dem Anwender möglich neue Kombinationen und Reihenfolgen von Verarbeitungsschritten auszuprobieren, ohne eine Zeile programmieren zu müssen. Die Entwicklungszeit neuer Methoden kann so drastisch gesenkt werden.

Ein Plugin ist dabei eine C++-Klasse, die entweder von der Klasse `ProcessorElement`, oder von der Klasse `ShaderElement` abgeleitet ist. `ShaderElement`-Plugins können dazu genutzt werden, das `RayCaster`-Plugin, welches für das Rendern der Volumendaten zuständig ist, zu erweitern. So sind z. B. alle Transferfunktionen als `ShaderElement`-Plugin implementiert. `ProcessorElement`-Plugins sind dagegen für die Verarbeitung der Volumendaten zuständig. Typische Beispiele sind das `RawLoader`-Plugin, welches Volumendatensätze im Raw-Format von der Festplatte lädt, sowie alle Filter-Plugins (z. B. das `Median`-Plugin). Auch die in dieser Arbeit vorgestellten Verfahren sind als `ProcessorElement`-Plugin (im Folgenden einfach „Plugin“ genannt) implementiert.

Der folgende Quelltext zeigt das Grundgerüst eines Plugins.

```
#ifndef FILTER_H
#define FILTER_H
```

[1] http://qt.nokia.com/

```
#include <stdint.h>

#include <core/pipeline/processorelement.h>

namespace vs2 {

class IntProperty;
class Port;

class Filter : public ProcessorElement
{
  Q_OBJECT

public:
  Filter();
  virtual ~Filter();

protected:
  void update();

private:
  static void indexU32toVoxelU8(PortFunctionContext *context);

  Port *_exampleInputPort;
  Port *_exampleOutputPort;
  IntProperty *_exampleProperty;
  uint8_t *_data;
};

} // namespace vs2

#endif // FILTER_H
```

Quelltext 4.1: Plugin-Grundgerüst.

Die grundlegenden Bestandteile sind die Objektvariablen vom Typ `Port` und `Property`, das Array `_data` und die `update()`-Methode. Die Variablen vom Typ `Port` definieren den Ein- und den Ausgang des Plugins. Bei Filter-Plugins liegt am Eingang typischerweise das Originalvolumen und am Ausgang das gefilterte Volumen. Die Variablen vom Typ `Property` definieren die Parameter des Plugins, welche dann in der grafischen Benutzeroberfläche angepasst werden können. Die eigentlichen Berechnungen des Filters werden in der `update()`-Methode durchgeführt und die Ergebnisse im eindimensionalen Array `_data` vom Typ `uint8_t` – welcher für ein Byte steht – gespeichert. Volumen liegen grundsätzlich als eindimensionales Array vor. Da die Voxel während des Filterns allerdings stets durch ihre drei Koordinaten adressiert werden, müssen diese in einen Array-Index umgerechnet werden. Dies geschieht im Folgenden stets durch das Makro `INDEX(x,y,z)`.

Abbildung 4.1 zeigt die grafische Oberfläche von *VolumeStudio2*. Im oberen Bereich wird die Pipeline definiert mit der ein Datensatz gerendert werden soll. Die Abbildung zeigt

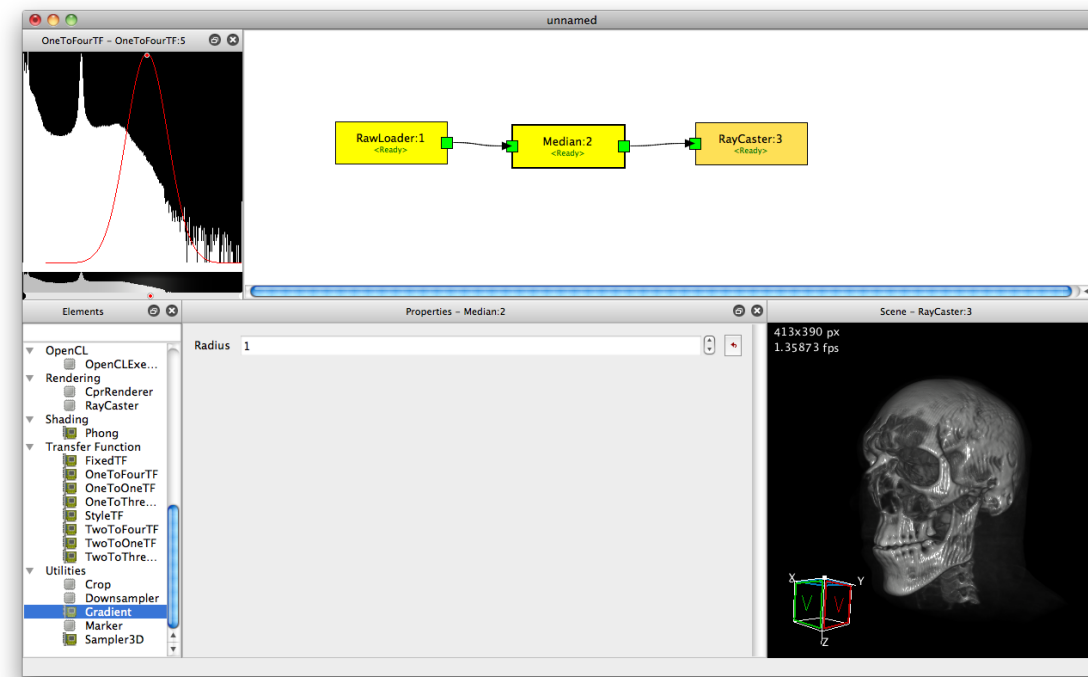


Abbildung 4.1: Oberfläche der Software *VolumeStudio2* mit noch improvisierter Darstellung der Pipeline.

dabei eine einfache Filter-Pipeline: Der Datensatz wird mit Hilfe des `RawLoader`-Plugins geladen und an den Eingangsport des `Median`-Plugins gelegt. Nach der Berechnung liegt das gefilterte Volumen an dessen Ausgangsport, welcher mit dem Eingangsport des `RayCaster`-Plugins verbunden ist. Dieses rendert schließlich das gefilterte Volumen (unten rechts). Die Parameter des `Median`-Plugins können im mittleren Bereich unterhalb der Pipeline, die Transferfunktion oben links angepasst werden. Im unteren linken Bereich werden alle verfügbaren Plugins aufgelistet.

4.2 Mathematische Operationen

Sowohl das Verfahren von Frangi et al. als auch das Verfahren von Pock et al. verwendet mathematische Operationen, wie partielle Ableitungen oder die Faltung, die zunächst nur für stetige Funktionen definiert sind. Da ein Volumendatensatz allerdings aus diskreten Datenwerten besteht, benötigt man für jede dieser Operationen ein diskretes Pendant, um die notwendigen Berechnungen durchführen zu können. Diese, sowie die Berechnung von Eigenwerten und Eigenvektoren, sollen im Folgenden kurz vorgestellt werden.

Dabei wird häufiger die Klasse `NeighborhoodCache` verwendet, welche einen einfachen Zugriff auf die *vollständige Nachbarschaft*, d. h. auf alle 26 Nachbarvoxel des aktuellen Voxels (x, y, z) erlaubt und sich automatisch um die korrekte Behandlung von Randvoxeln kümmert. Durch die Klasse erhält man Zugriff auf ein Array (im Folgenden `nh` genannt),

in dem die Nachbarschaft wie folgt schichtweise (Schichten werden in z-Richtung gezählt) gespeichert ist.

$$
\begin{array}{cccc}
[0] & [1] & [2] & = & (x-1,y-1,z-1) & (x,y-1,z-1) & (x+1,y-1,z-1) \\
[3] & [4] & [5] & = & (x-1,y,z-1) & (x,y,z-1) & (x+1,y,z-1) \\
[6] & [7] & [8] & = & (x-1,y+1,z-1) & (x,y+1,z-1) & (x+1,y+1,z-1) \\
 & & & & & & \\
[9] & [10] & [11] & = & (x-1,y-1,z) & (x,y-1,z) & (x+1,y-1,z) \\
[12] & [13] & [14] & = & (x-1,y,z) & (x,y,z) & (x+1,y,z) \\
[15] & [16] & [17] & = & (x-1,y+1,z) & (x,y+1,z) & (x+1,y+1,z) \\
 & & & & & & \\
[18] & [19] & [20] & = & (x-1,y-1,z+1) & (x,y-1,z+1) & (x+1,y-1,z+1) \\
[21] & [22] & [23] & = & (x-1,y,z+1) & (x,y,z+1) & (x+1,y,z+1) \\
[24] & [25] & [26] & = & (x-1,y+1,z+1) & (x,y+1,z+1) & (x+1,y+1,z+1)
\end{array}
$$

4.2.1 Partielle Ableitungen

Partielle Ableitungen lassen sich gut durch sogenannte *zentrale Differenzen* approximieren, welche in den folgenden Quelltexten zur Berechnung des Gradienten und der Hesse-Matrix eingesetzt werden. Wie schon erwähnt bezeichnet `nh` das Array mit den Nachbarschaftsvoxeln.

```
void gradient(uint8_t *nh, float &dx, float &dy, float &dz)
{
  dx = (nh[14] - nh[12]) / 2.0f;
  dy = (nh[16] - nh[10]) / 2.0f;
  dz = (nh[22] - nh[4])  / 2.0f;
}
```

Quelltext 4.2: Berechnung des Gradienten.

```
void hessian(uint8_t *nh,
             float &dxx, float &dxy, float &dxz,
             float &dyy, float &dyz, float &dzz)
{
  uint8_t current = nh[13];

  // diagonal entries
  dxx = nh[14] + nh[12] - 2.0f*current;
  dyy = nh[16] + nh[10] - 2.0f*current;
  dzz = nh[22] + nh[4]  - 2.0f*current;

  //off-diagonal entries
  dxy = (nh[17] - nh[11] - nh[15] + nh[9]) / 4.0f;
  dxz = (nh[23] - nh[5]  - nh[21] + nh[3]) / 4.0f;
  dyz = (nh[25] - nh[7]  - nh[19] + nh[1]) / 4.0f;
}
```

Quelltext 4.3: Berechnung der Hesse-Matrix.

4.2.2 Gauss-Filter

Das für den Gauss-Filter benötigte Faltungsintegral wird im diskreten Fall durch eine Summe approximiert. Der aktuelle Voxel bekommt dabei den gewichteten Durchschnittswert aller Nachbarschaftsvoxel zugewiesen. Die Gewichtung der einzelnen Voxel wird typischerweise durch eine Faltungsmatrix angegeben. Eine gute Approximation des Gauss-Filters wird durch den *Binomialfilter* erreicht, dessen Faltungsmatrix im zweidimensionalen Fall wie folgt aussieht

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{4.1}$$

Dies bedeutet, dass der aktuelle Voxel mit dem Wert 4 in der Mitte der Matrix, die links, rechts, oben und unten angrenzenden Voxel mit dem Wert 2 und die diagonal angrenzenden Voxel mit dem Wert 1 gewichtet und aufsummiert werden. Wird dieser Wert anschließend durch 16 (die Summe der Werte in der Matrix) geteilt, so erhält man das normierte Endergebnis der Faltung für den aktuellen Voxel. Im dreidimensionalen Fall benutzt man entsprechend eine dreidimensionale Faltungsmatrix, die aus den drei zweidimensionalen Matrizen

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{4.2}$$

aufgebaut ist. Dieses Verfahren ist im folgenden Quelltext umgesetzt, wobei `toBlur` ein Zeiger auf das weichzuzeichnende Volumen ist.

```
void filterVolume(uint8_t *toBlur)
{
  float sum;

  // initialize nh ...
  for (int z = 0; z < _resolution.z; z++)
  {
    for (int y = 0; y < _resolution.y; y++)
    {
      for (int x = 0; x < _resolution.x; x++)
      {
        // current slice
        sum  = nh[13] * 8.0f;
        sum += nh[9]  * 2.0f;
        sum += nh[10] * 4.0f;
        sum += nh[11] * 2.0f;
        sum += nh[12] * 4.0f;
        sum += nh[14] * 4.0f;
        sum += nh[15] * 2.0f;
        sum += nh[16] * 4.0f;
        sum += nh[17] * 2.0f;
```

```
        // previous slice
        sum += nh[0];
        sum += nh[1] * 2.0f;
        sum += nh[2];
        sum += nh[3] * 2.0f;
        sum += nh[4] * 4.0f;
        sum += nh[5] * 2.0f;
        sum += nh[6];
        sum += nh[7] * 2.0f;
        sum += nh[8];

        // next slice like previous slice...

        toBlur[INDEX(x,y,z)] = sum / 64.0f;
      } // x
    } // y
  } // z
}
```

Quelltext 4.4: Implementierung des GAUSS-Filters.

4.2.3 Eigenwerte und Eigenvektoren

Da die HESSE-Matrix reell und symmetrisch ist, bietet sich zur Berechnung ihrer Eigenwerte und Eigenvektoren das JACOBI-Verfahren an. Das ist ein iteratives Verfahren, welches eine reelle und symmetrische Matrix mit Hilfe von Rotationsmatrizen schrittweise auf Diagonalgestalt bringt. Ist diese mit hinreichender Genauigkeit erreicht, so sind die Werte auf der Hauptdiagonalen gerade die Eigenwerte der Matrix. Werden die Rotationen zusätzlich in der gleichen Reihenfolge auf eine Einheitsmatrix angewandt, so enthalten deren Spalten danach die zugehörigen Eigenvektoren. Eine genaue Beschreibung des JACOBI-Verfahrens kann z. B. in [SK09] nachgelesen werden.

```
void eigen(float *m, float *e, float *v1, float *v2, float *v3)
{
  float c, s, t, theta, tau, tmp, tmp2;
  float V[9]; // eigenvector matrix initialized as identity

  // iterations
  for (int it = 0; !isDiagonal(m) && it < 20; it++)
  {
    // sweeps (diagonalizes m)
    for (int p = 0; p < 3; p++)
    {
      for (int q = p+1; q < 3; q++)
      {
        // jacobi rotation (set element m_pq = 0)
        if (abs(m(p,q)) > 0)
        {
          // calculate rotation values
```

```
            theta = (m(q,q) - m(p,p)) / (2 * m(p,q));
            t = 1 / (abs(theta) + sqrt(1 + theta*theta));
            if (theta < 0) t = -t;
            c = 1 / sqrt(1 + t*t);
            s = t * c;
            tau = s / (1 + c);

            // rotate matrix m (and V)
            tmp = m(p,q);
            m(p,q) = 0;
            m(p,p) -= t * tmp;
            m(q,q) += t * tmp;

            for (int r = 0; r < 3; r++)
            {
              if (r != p && r != q)
              {
                tmp  = m(r,p);
                tmp2 = m(r,q);
                m(r,p) = tmp  - s * (tmp2 + tau*tmp);
                m(r,q) = tmp2 + s * (tmp - tau*tmp2);
              }

              tmp  = V(r,p);
              tmp2 = V(r,q);
              V(r,p) = tmp  - s * (tmp2 + tau*tmp);
              V(r,q) = tmp2 + s * (tmp - tau*tmp2);
            }
          } // jacobi rotation
        }
      } // sweeps
    } // iterations
    // write eigenvalues into array e and eigenvectors
    // into v1, v2 and v3 ...
}
```

Quelltext 4.5: Berechnung von Eigenwerten und Eigenvektoren.

Dabei ist `m(i,j)` bzw. `V(i,j)` eine abkürzende Schreibweise für den Zugriff auf die Matrix, die als eindimensionales Array gespeichert ist. Die Funktion `isDiagonal` prüft, ob die Nebendiagonalelemente hinreichend nahe bei Null liegen und gibt in diesem Fall ein `true` zurück.

4.3 Implementation der Filter

Aufgrund seiner einfacheren Struktur wird zuerst die Implementation des Filters von Frangi et al. duch die Klasse `VesselnessFrangi` beschrieben.

```
class VesselnessFrangi : public ProcessorElement
{
  Q_OBJECT

public:
  VesselnessFrangi();
  virtual ~VesselnessFrangi();

protected:
  void update();

private:
  static void indexU32toVoxelU8(PortFunctionContext *context);

  Port *_inputPort;
  Port *_outputPort;

  FloatProperty *_aProperty;
  FloatProperty *_bProperty;
  FloatProperty *_cProperty;
  IntProperty *_rMinProperty;
  IntProperty *_rMaxProperty;

  uint8_t *_data; //output
  tgt::ivec3 _resolution; // resolution of the input volume
  int _dataSize; // actual datasize in bytes

  // filter specific data
  int _r; // current scale value
  uint8_t *_filtered, *_tmpVol; // temporary volume copies
                                // for the scale space

  bool filterVolume(uint8_t *toBlur);
  void hessian(uint8_t *nh, float &dxx, float &dxy, float &dxz,
                            float &dyy, float &dyz, float &dzz);
  bool isDiagonal(const float *m);
  void eigen(float *m, float *e, float *v1, float *v2, float *v3);
};
```

Quelltext 4.6: Die Klasse `VesselnessFrangi`.

Die Variablen vom Typ `FloatProperty` entsprechen dabei den Parametern α, β und c in Gleichung (3.8), während die Variablen `_rMinProperty` und `_rMaxProperty` die minimale und maximale Größe des Scale-Space bezeichnen. Die eigentlichen Berechnungen erfolgen in der `update()`-Methode. Für jede Größe des Scale-Space wird dort einmal über das weichgezeichnete Volumen `_filtered` iteriert und die Hesse-Matrix in jedem Voxel berechnet. Anschließend werden die Eigenwerte ermittelt und – gemäß Gleichung (3.3) – sortiert in das Array `e` geschrieben. Es folgt die Berechnung der Größen $\mathcal{R}_\mathcal{B}$, $\mathcal{R}_\mathcal{A}$ und $\mathcal{S}$, welche dann gemäß Gleichung (3.8) zur Metrik zusammengesetzt werden. Der berechnete Wert wird schließlich in das Array `_data` geschrieben, falls der dort gespeicherte Wert

kleiner ist als der aktuelle. Folgender Quelltext zeigt den relevanten Ausschnitt der `update()`-Methode.

```
for (_r = _rMin; _r <= _rMax; _r++)
{
  // initialize nh ...
  for (int z = 0; z < _resolution.z; z++)
  {
    for (int y = 0; y < _resolution.y; y++)
    {
      for (int x = 0; x < _resolution.x; x++)
      {
        idx = INDEX(x,y,z);
        // compute hessian matrix and eigenvalues
        hessian(nh, dxx, dxy, dxz, dyy, dyz, dzz);
        H(0,0) = _r*_r*dxx; H(0,1) = _r*_r*dxy; H(0,2) = _r*_r*dxz;
        H(1,0) = _r*_r*dxy; H(1,1) = _r*_r*dyy; H(1,2) = _r*_r*dyz;
        H(2,0) = _r*_r*dxz; H(2,1) = _r*_r*dyz; H(2,2) = _r*_r*dzz;
        eigen(H, e, v1, v2, NULL);

        // compute vesselness
        RB = abs(e[2]) / sqrt(abs(e[0] * e[1]));
        RA = abs(e[1]) / abs(e[0]);
        S  = sqrt(e[0]*e[0] + e[1]*e[1] + e[2]*e[2]);

        if (e[0] > 0 || e[1] > 0)
          current = 0;
        else
          current = 255.0f * (1 - exp(-RA*RA/(2*a*a))) *
                    exp(-RB*RB/(2*b*b)) * (1 - exp(-S*S/(2*c*c)));

        if (current < 0) current = 0;
        if (current > 255) current = 255;
        if (current > _data[idx])
          _data[idx] = current
      } // x
    } // y
  } // z
  filterVolume(_filtered)
} // _r
```

Quelltext 4.7: Der Filter von FRANGI et al.

Als nächstes soll die Klasse `VesselRadius` vorgestellt werden, die den Filter von POCK et al. implementiert.

```
class VesselRadius : public ProcessorElement
{
  Q_OBJECT

public:
  VesselRadius();
```

```
  virtual ~VesselRadius();

protected:
  void update();

private:
  static void indexU32toVoxelU8(PortFunctionContext *context);

  Port *_inputPort;
  Port *_outputPort;

  FloatProperty *_etaProperty;
  IntProperty *_nProperty;
  IntProperty *_rMinProperty;
  IntProperty *_rMaxProperty;

  uint8_t *_data; //output
  tgt::ivec3 _resolution; // resolution of the input volume
  int _dataSize; // actual datasize in bytes

  // filter specific data
  int _scaleH, _scaleB, _r; // current scale values
  uint8_t *_filteredH, *_filteredB, *_tmpVol; // temporary volume
                                              // copies for each
                                              // scale space

  bool filterVolume(uint8_t *toBlur);
  void gradient(uint8_t *nh, float &dx, float &dy, float &dz);
  void hessian(uint8_t *nh, float &dxx, float &dxy, float &dxz,
                            float &dyy, float &dyz, float &dzz);
  void computeHessianEigenvectors(uint8_t *nh, float *v1, float *v2);
  void computeBoundariness(uint8_t *nh, float &b, float *g);
  float RPlus(int x, int y, int z, float *v1, float *v2);
  float RMinus(uint8_t *nh);
  bool isDiagonal(const float *m);
  void eigen(float *m, float *e, float *v1, float *v2, float *v3);
};
```

Quelltext 4.8: Die Klasse `VesselRadius`.

Die Variablen `_etaProperty` und `_nProperty` entsprechen den Parametern der Gleichungen (3.20) und (3.14). Die Variablen `_rMinProperty` und `_rMaxProperty` bezeichnen wieder die minimale und maximale Größe des Scale-Space. Da das Verfahren zwei Scale-Spaces verwendet, muss in einem ersten Schritt sichergestellt werden, dass die in Gleichung (3.20) definierte Relation eingehalten wird. Danach wird über das weichgezeichnete Volumen `_filteredH` iteriert und die beiden größten Eigenvektoren `v1` und `v2` mit Hilfe der Methode `computeHessianEigenvectors()` ermittelt. Es folgt die Berechnung der Metrik gemäß Gleichung (3.25). Das Ergebnis wird schließlich in das Array `_data` geschrieben, falls der dort gespeicherte Wert kleiner ist als der aktuelle. Folgender Quelltext zeigt den relevanten Ausschnitt der `update()`-Methode.

```
_scaleB = 0;
for (_r = _rMin; _r <= _rMax; _r++)
{
  // set current scales
  _scaleH = _r;
  int scaleB = pow((float)_scaleH, _eta); // new boundariness scale
  // filter with boundariness-scale if current scale is smaller than
  // the new scale
  while (_scaleB < scaleB)
  {
    filterVolume(_filteredB)
    _scaleB++;
  }

  // initialize nh for volume _filteredH ...
  for (int z = 0; z < _resolution.z; z++)
  {
    for (int y = 0; y < _resolution.y; y++)
    {
      for (int x = 0; x < _resolution.x; x++)
      {
        // calculate eigenvectors
        computeHessianEigenvectors(nh, v1, v2);

        idx = INDEX(x, y, z);
        current = RPlus(x, y, z, v1, v2) - RMinus(nh);
        if (current < 0) current = 0;
        if (current > _data[idx])
          _data[idx] = current
      } // x
    } // y
  } // z
  filterVolume(_filteredH)
} // _r
```

Quelltext 4.9: Der Filter von Pock et al.

Die Logik der Methode `computeHessianEigenvectors()` entspricht der Berechnung der Eigenwerte der Hesse-Matrix im Filter von Frangi et al. und wird deshalb nicht wiederholt. Im Folgenden werden nun noch die Implementierungen der Methoden `RPlus()`, `RMinus()` und `computeBoundariness()` vorgestellt, aber nicht weiter erläutert, da sie eins zu eins Umsetzungen der Gleichungen (3.19), (3.24) und (3.22) aus Abschnitt 3.2 sind.

```
float RPlus(int x, int y, int z, float *v1, float *v2)
{
  N = floor(2.0f * M_PI * _r + 1);
  // initialize nh for volume _filteredB ...
  // initial medialness
  for (int i = 0; i < N; i++)
  {
```

```
    // compute radial direction with eigenvectors of hessian
    a_i = 2.0f * M_PI * i / N;
    cos_ai = cos(a_i);
    sin_ai = sin(a_i);
    // use the two largest eigenvectors
    v_ai[0] = cos_ai * v1[0] + sin_ai * v2[0];
    v_ai[1] = cos_ai * v1[1] + sin_ai * v2[1];
    v_ai[2] = cos_ai * v1[2] + sin_ai * v2[2];

    // compute new coordinates
    x_new = (int)(x + _r*v_ai[0]);
    y_new = (int)(y + _r*v_ai[1]);
    z_new = (int)(z + _r*v_ai[2]);
    // check if still inside volume ...

    // update nh for voxel (x_new,y_new,z_new) ...
    // compute gradient length b and direction g
    computeBoundariness(nh, b, g);

    // compute circularity
    c_i = -g[0]*v_ai[0] - g[1]*v_ai[1] - g[2]*v_ai[2];
    if (c_i > 0) c_i = pow(c_i, _n);
    else c_i = 0;
    b_i[i] = b * c_i;

    sum += b_i[i];
  }
  sum /= N;

  // symmetry confidence
  for (int i = 0; i < N; i++)
    variance += (b_i[i] - sum)*(b_i[i] - sum);
  variance /= N;

  if (sum == 0) return 0;
  return sum * (1 - variance / (sum*sum));
}
```

Quelltext 4.10: Berechnung der Symmetry Constrainted Medialness.

```
float RMinus(uint8_t *nh)
{
  float dx, dy, dz;

  gradient(nh, dx, dy, dz);
  return _scaleH * sqrt(dx*dx + dy*dy + dz*dz);
}
```

Quelltext 4.11: Berechnung des Adaptive Threshold.

```
void computeBoundariness(uint8_t *nh, float &b, float *g)
{
  float dx, dy, dz;

  gradient(nh, dx, dy, dz);
  dx *= _scaleB; dy *= _scaleB; dz *= _scaleB;
  b = sqrt(dx*dx + dy*dy + dz*dz);
  if (b != 0)
  {
    g[0] = dx / b;
    g[1] = dy / b;
    g[2] = dz / b;
  }
}
```

Quelltext 4.12: Berechnung des Boundariness-Gradienten.

4.4 Rückprojektion

Um die in Abschnitt 3.3 beschriebene Vorgehensweise zur Erstellung der Größenabstraktion und Durchführung der Rückprojektion zu implementieren, müssen die Quelltexte 4.7 und 4.9 leicht angepasst werden. Dazu wird die Struktur

```
typedef struct VesselCenterline
{
  float value;
  int radius;
  float v1[3];
  float v2[3];
}VesselCenterline;
```

Quelltext 4.13: Die Struktur `VesselCenterline`.

definiert, mit der der berechnete Wert der jeweiligen Metrik (`value`), die Größe des Scale-Space bei der dieser Wert berechnet wurde (`radius`), sowie die beiden größten Eigenvektoren (`v1` und `v2`) für jeden Voxel im Array `_C` gespeichert werden. Damit können die Zeilen

```
if (current > _data[idx])
  _data[idx] = current
```

durch

```
if (current > _C[idx].value)
{
  _C[idx].value = current;
  _C[idx].radius = _r;
  _C[idx].v1[0] = v1[0]; _C[idx].v2[0] = v2[0];
  _C[idx].v1[1] = v1[1]; _C[idx].v2[1] = v2[1];
```

```
  _C[idx].v1[2] = v1[2]; _C[idx].v2[2] = v2[2];
}
```

ersetzt werden, so dass für jeden Voxel die notwendigen Informationen vorliegen, um die Bedingungen (3.27) in der Methode `detectMaxima()` überprüfen zu können.

```
void detectMaxima()
{
  int x_new, y_new, z_new;
  int idx, idx_new;
  bool isMax;

  for (int z = 0; z < _resolution.z; z++)
  {
    for (int y = 0; y < _resolution.y; y++)
    {
      for (int x = 0; x < _resolution.x; x++)
      {
        // detect maximum
        isMax = true;
        idx = INDEX(x, y, z);

        // first eigenvector positive
        x_new = ceil(x + _C[idx].v1[0]);
        y_new = ceil(y + _C[idx].v1[1]);
        z_new = ceil(z + _C[idx].v1[2]);
        // check if still inside volume
        if (x_new >= 0 && x_new < _resolution.x &&
            y_new >= 0 && y_new < _resolution.y &&
            z_new >= 0 && z_new < _resolution.z)
        {
          idx_new = INDEX(x_new, y_new, z_new);
          isMax &= _C[idx].value > _C[idx_new].value;
        }

        // first eigenvector negative
        x_new = floor(x - _C[idx].v1[0]);
        y_new = floor(y - _C[idx].v1[1]);
        z_new = floor(z - _C[idx].v1[2]);
        // check if still inside volume ...
        // second eigenvector positive and negative ...

        if (isMax)
        {
          _data[idx] = 128;
        }
        else
        {
          _C[idx].value = 0;
        }
      } // x
    } // y
```

```
  } // z
}
```

Quelltext 4.14: Detektion der Maxima.

Wird ein Voxel als Maximum erkannt, so wird ein Wert von 128 an die entsprechende Stelle des Ausgabevolumens geschrieben. Dies ist notwendig für den nächsten Schritt, das sogenannte *Ausdünnen* (engl.: „thinning“), bei dem Ausreißer, d. h. maximale Voxel deren 26 Nachbarschaftsvoxel selbst keine Maxima sind, wieder entfernt werden. Der zugehörige Quelltext lautet wie folgt:

```
void performThinning()
{
  int idx;
  bool isSimple;
  // initialize nh ...

  for (int z = 0; z < _resolution.z; z++)
  {
    for (int y = 0; y < _resolution.y; y++)
    {
      for (int x = 0; x < _resolution.x; x++)
      {
        idx = INDEX(x, y, z);
        // if voxel is max
        if (_data[idx] == 128)
        {
          // update nh ...
          // check if voxel is simple
          isSimple = true;
          for (int i = 0; i < 27; i++)
          {
            // if not current voxel
            if (i != 13 && nh[i] == 128)
            {
              isSimple = false;
              break;
            }
          }
          if (isSimple)
          {
            _C[idx].value = 0;
          }
          _data[idx] = 0;
        }
      } // x
    } // y
  } // z
}
```

Quelltext 4.15: Entfernen der Ausreißer.

Mit dieser Vorarbeit kann nun der Rückprojektionsalgorithmus erläutert werden, der in der Methode `performBackprojection()` implementiert ist. In dieser Methode wird einmal über das Ausgabevolumen iteriert. Wird dabei ein maximaler Voxel entdeckt, so liest die Methode den in `_C` gespeicherten Radius aus. Anschließend wird dieser Radius allen Voxeln innerhalb einer Kugel mit dem ausgelesenen Radius um den maximalen Voxel herum zugewiesen. Hierbei erweist sich die Verwendung von Kugelkoordinaten als hilfreich. Der relevante Ausschnitt der Methode ist in folgenden Quelltext aufgeführt.

```
idx = INDEX(x, y, z);
if (_C[idx].value > 0)
{
  _data[idx] = _C[idx].radius;
  // sphere coordinates
  for (int r = 1; r <= _C[idx].radius; r++)
  {
    for (int i = 0; i < 360; i++)
    {
      for (int j = 0; j <= 180; j++)
      {
        phi = i * M_PI / 180;
        theta = j * M_PI / 180;

        x_new = (int)(x + r * sin(theta) * cos(phi));
        y_new = (int)(y + r * sin(theta) * sin(phi));
        z_new = (int)(z + r * cos(theta));
        // check if still inside volume ...
        idx_new = INDEX(x_new, y_new, z_new);
        if (_data[idx_new] < _C[idx].radius)
          _data[idx_new] = _C[idx].radius;
      }
    }
  }
}
```

Quelltext 4.16: Rückprojektion.

5 Evaluation der Ergebnisse

> Es ist nicht genug zu wissen – man muss auch anwenden. Es ist nicht genug zu wollen – man muss auch tun.
>
> *(Johann Wolfgang von Goethe)*

5.1 Validierung der Verfahren

Um den in den Abschnitten 3.3 und 4.4 beschriebenen Rückprojektionsalgorithmus anwenden zu können, ist es notwendig, dass die bei der Erstellung der Größenabstraktion gefundenen Voxel möglichst auf den Mittellinien der erkannten Röhren liegen. Dies soll im Folgenden, für die in Kapitel 3 vorgestellten Verfahren, anhand von synthetischen Datensätzen untersucht und validiert werden. Die dabei verwendeten Datensätze sind in Abbildung 5.1 dargestellt und stammen aus der öffentlichen Datenbank von Krissian und Farneback.[1] Die Standardparameter wurden – gemäß der Originalarbeiten – auf $\alpha = \beta = 0.5$ und $c = 100$ beim Verfahren von Frangi et al., sowie $\eta = 0$ und $n = 2$ beim Verfahren von Pock et al. festgelegt.

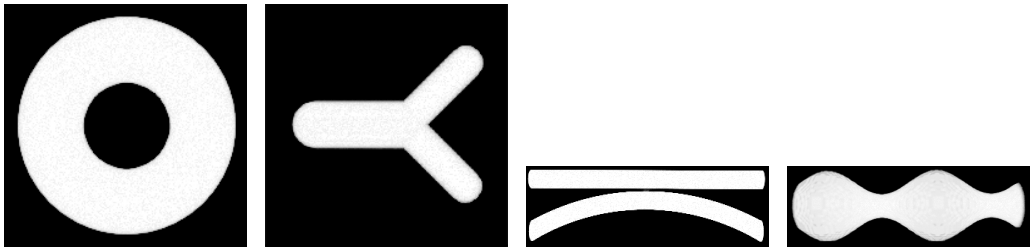
Die Abbildungen 5.2a–5.2d zeigen die Voxel der Größenabstraktion für die verschiedenen Datensätze, nachdem diese mit dem Filter von Pock et al. bearbeitet wurden. Die Größenparameter waren dabei auf $r_{\min} = 1$ und $r_{\max} = 10$ gesetzt. Wie man sieht, stimmen die erkannten Maxima ziemlich genau mit den Mittellinien der ursprünglichen Strukturen überein. Die Erstellung der Größenabstraktion kann somit mit der Extraktion der Mittellinien (engl.: „centerline extraction“) gleichgesetzt werden.

Die Abbildungen 5.2e–5.2h zeigen die Ergebnisse des Filters von Frangi et al., wobei die Größenparameter auf $s_{\min} = s_{\max} = 1$ gesetzt waren. Man erkennt, dass die Maxima nur bei leicht gekrümmten Röhren homogener Größe (tangentiale Röhren und Y-Stück) annähernd mit den Mittellinien übereinstimmen. Sobald der Radius der Röhre variiert oder die Röhre ein hohes Maß an Krümmung aufweist (Torus), werden auch viele Voxel außerhalb der Mittellinie als Maximum erkannt.

Wie anhand der Abbildungen 5.2i–5.2l zu sehen ist, lässt sich dieses Problem auch nicht so leicht beheben. Bei dieser Bilderreihe wurde nur der Parameter β verändert und auf den Wert 0.01 gesetzt.[2] Einzig beim Torus verringert sich dadurch die Streuung der Maxima. Röhren mit variierendem Radius bleiben allerdings problematisch.

[1] http://lmi.bwh.harvard.edu/research/vascular/SyntheticVessels/SyntheticVesselImages.html

[2] Generell veränderten sich die Bilder nur bei Anpassung von β merklich. Alle anderen Parameter schienen keinen großen Einfluss auf das Ergebnis zu haben, weshalb sie im Folgenden auch nicht weiter beachtet werden.



(a) (b) (c) (d)

Abbildung 5.1: Die für Validierung verwendeten synthetischen Datensätze. (a) Torus; (b) Y-Stück; (c) tangentiale Röhren; (d) Röhre variierender Größe.

Die Abbildungen 5.2m–5.2p und 5.2q–5.2t zeigen ein weiteres Problem des Verfahrens im Hinblick auf die Rückprojektion auf. Beide Bilderreihen wurden mit den Größenparametern $s_{\min} = 1$ und $s_{\max} = 10$ erstellt (die erste Reihe mit Standardparametern, die zweite mit $\beta = 0.01$). Hier werden nicht nur viele Maxima außerhalb der Mittellinien erkannt, sondern es fallen zudem auch, mit den vorherigen Einstellungen erkannte Maxima wieder weg, was besonders bei den beiden tangentialen Röhren deutlich wird.

Für eine erfolgreiche Rückprojektion ist, neben der Lokalisierung der Mittellinie einer Röhre, auch eine Abschätzung von deren Radius entlang dieser Mittellinie notwendig, d. h. die Verfahren müssen für verschiedene Größen des Scale-Space angewendet werden. Damit ist das Verfahren von Frangi et al. – nach obiger Ausführung – aber gänzlich ungeeignet für diese Art der Rückprojektion. Einzig das Verfahren von Pock et al. liefert die erforderlichen Ergebnisse.

Da das Verfahren von Frangi et al. ursprünglich zur direkten Hervorhebung von röhrenartigen Strukturen vorgeschlagen wurde, bietet es sich an, jedem Voxel den unveränderten Wert der Metrik (3.10) zuzuweisen, anstatt die Maxima zu extrahieren und eine Rückprojektion durchzuführen. Anstelle der *Größe* als Definitionsbereich einer Transferfunktion wird dann einfach direkt die *Gefäßartigkeit* verwendet.

In Abbildung 5.3 ist ein Vergleich der beiden Verfahren dargestellt. Bei der Erstellung der Bilder wurden die Standardparameter, sowie die Größenparameter $r_{\min} = s_{\min} = 1$ und $r_{\max} = s_{\max} = 10$ verwendet. Die Bilder (a)–(d) zeigen die Ergebnisse des Verfahrens von Pock et al. nach erfolgter Rückprojektion; die Bilder (e)–(h) zeigen die Ergebnisse des Verfahrens von Frangi et al. bei direkter Verwendung der Metrik. Bis auf die Tatsache, dass bei Letzterem eine etwas größere Anzahl an Voxeln erkannt wird, ähneln sich die Ergebnisse doch sehr.

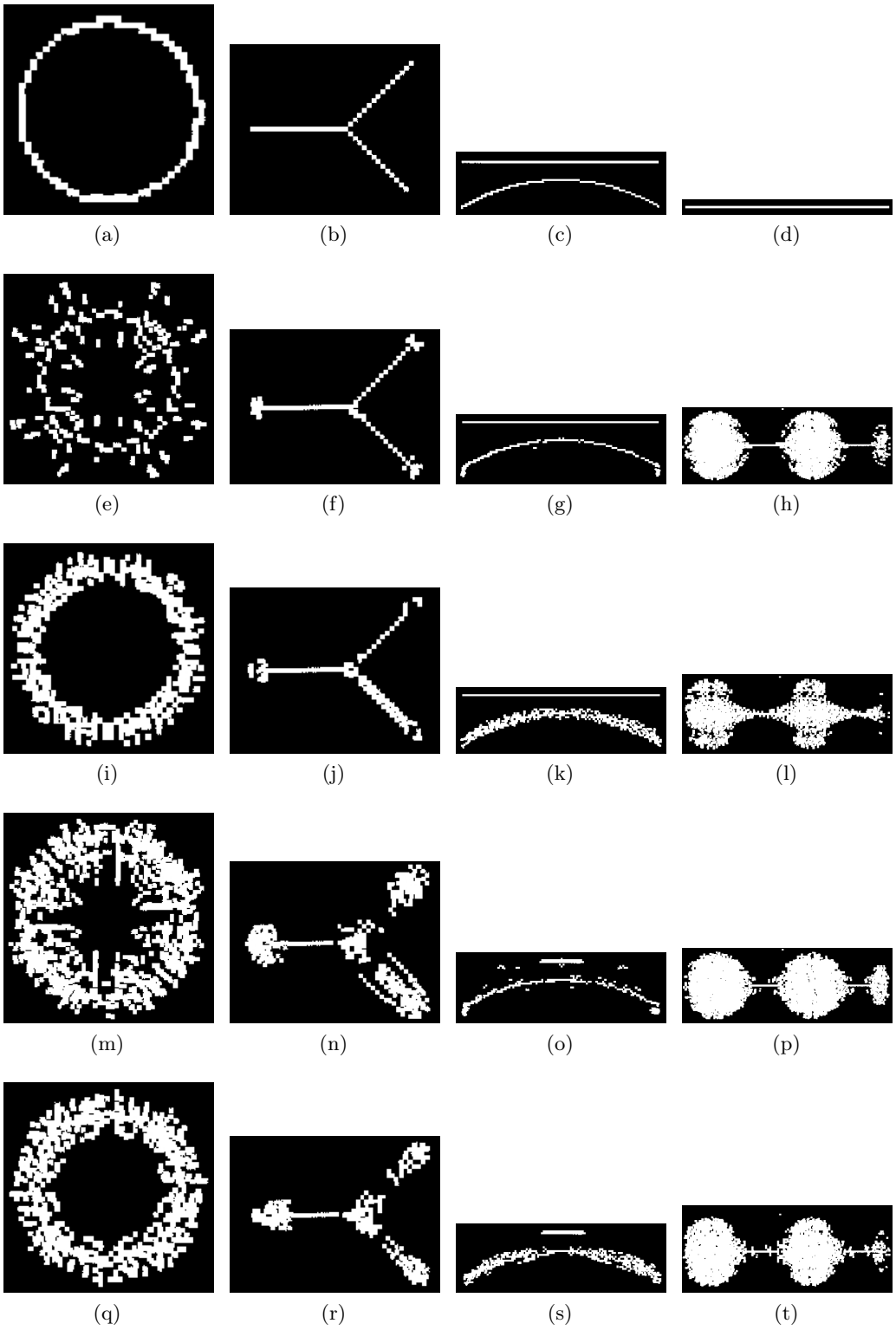


Abbildung 5.2: Extrahierte Maxima der synthetischen Datensätze. (a)–(d): Verfahren von Pock et al.; (e)–(t): Verfahren von Frangi et al. für verschiedene Parameter.

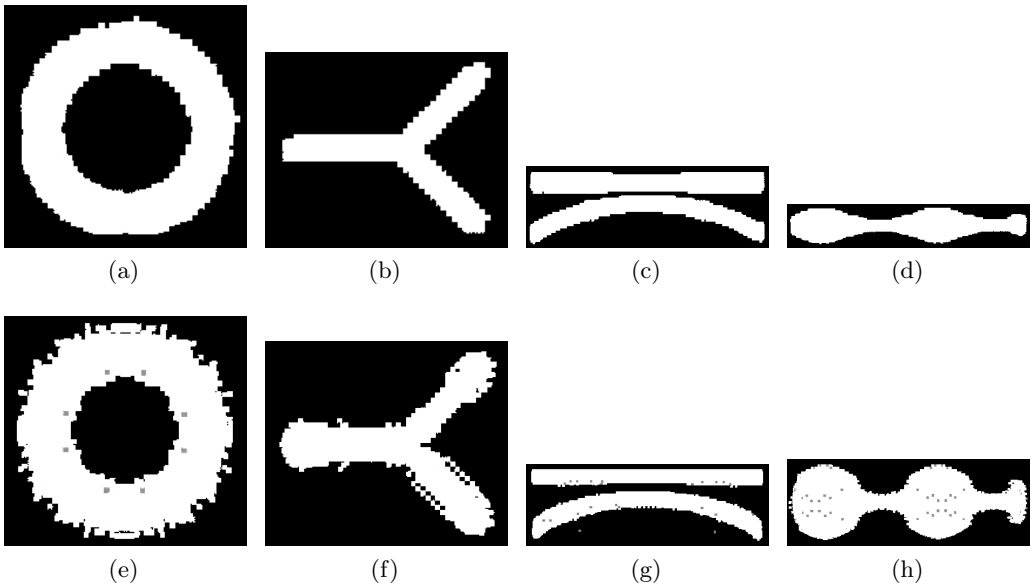


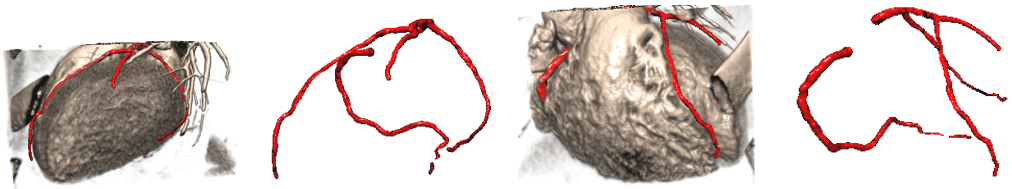
Abbildung 5.3: Rückprojektion der synthetischen Datensätze. (a)–(d): Verfahren von Pock et al. mit vorgeschlagener Rückprojektion; (e)–(h): Verfahren von Frangi et al. mit direkter Verwendung der berechneten Metrik.

5.2 Anwendung auf reale Datensätze

Der Vorteil der synthetischen Datensätze ist die Abwesenheit von nicht röhrenartigen Strukturen, sowie von Bildrauschen, welches typischerweise bei CT-Scans entsteht (u. a. durch leichte Bewegungen des Patienten). Daher soll nun untersucht werden, wie gut die beiden Verfahren bei realen Datensätzen vom menschlichen Herz abschneiden, d. h. wie gut die Herzkranzgefäße herausgestellt werden können.

Um eine Referenz zu haben, wurden die Verfahren auf Datensätze des *Rotterdam Coronary Artery Algorithm Evaluation Framework*[3] [SMW+09] angewandt. Dieses stellt, zusätzlich zu CT-Scans des Herzens, auch eine von Spezialisten erstellte Segmentierung der Koronararterien (Herzkranzarterien) für jeden Datensatz zur Verfügung. Ein Volumenrendering des in diesem Abschnitt verwendeten Datensatzes mit hervorgehobenen Herzkranzgefäßen ist in Abbildung 5.4 aus zwei verschiedenen Perspektiven dargestellt. Abbildung 5.4a zeigt die linke Seite des Herzens mit den beiden Ästen der *Arteria coronaria sinistra* (linke Koronararterie). Abbildung 5.4c zeigt das Herz von vorne. Rechts im Bild ist ein Ast der linken Koronararterie, links im Bild ein Stück der *Arteria coronaria dextra* (rechte Koronararterie) zu sehen, die einmal über die gesamte Unterseite des Herzens verläuft. Die Abbildungen 5.4b und 5.4d zeigen jeweils nur die Koronararterien. Bei der Erstellung der im weiteren Verlauf gezeigten Bilder wurde stets eine Ansicht von vorne gewählt, sowie das Herz ausgeblendet, so dass diese Abbildung 5.4d entsprechen.

[3] http://coronary.bigr.nl/



(a) (b) (c) (d)

Abbildung 5.4: CT-Datensatz des Herzens mit hervorgehobenen Herzkranzgefäßen.

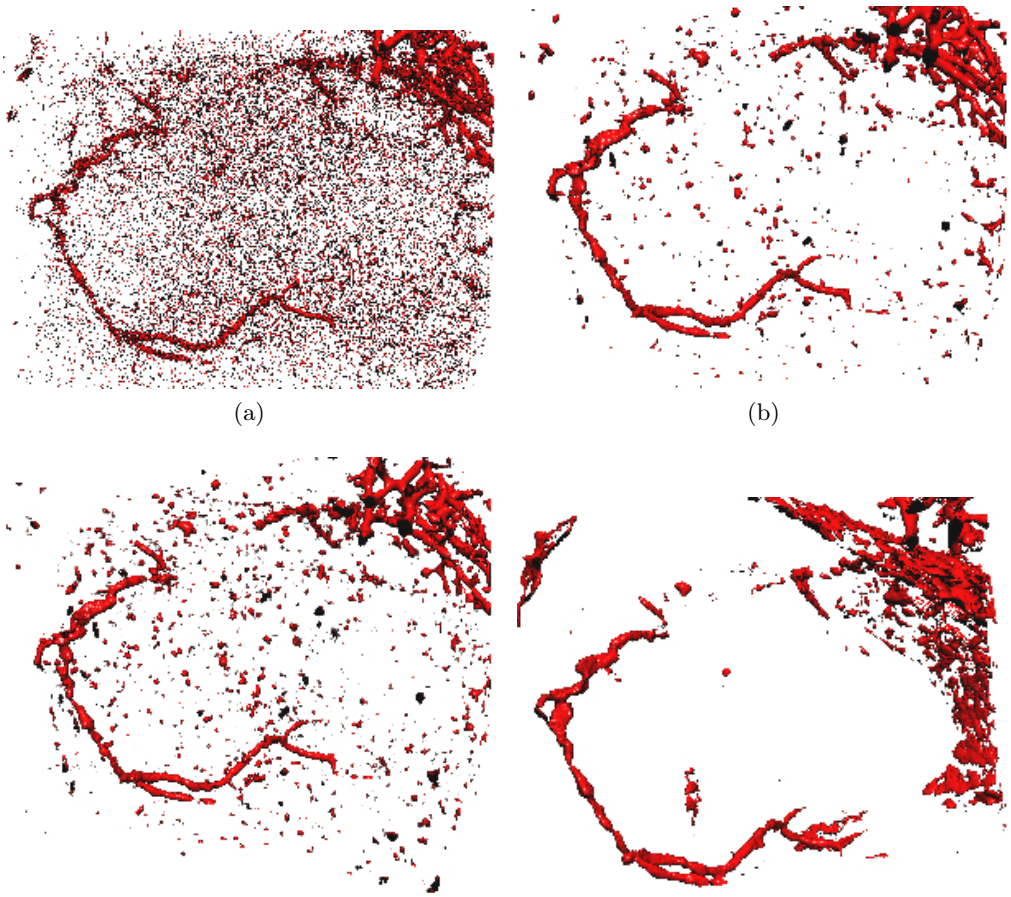
Bei der Evaluation des Filters von Frangi et al. wurde der ursprüngliche Datensatz so zugeschnitten, dass nur der Teil des Herzens, der die Koronararterien enthält, bearbeitet wird. Der resultierende Datensatz hat eine Größe von $284 \times 336 \times 206$ Voxeln. Die Größenparameter des Filters wurden auf $s_{\min} = 1$ und $s_{\max} = 10$ festgelegt. Die Bearbeitung des Datensatzes mit diesen Einstellungen dauerte auf einem Core 2 Duo mit 2.53 GHz ca. 4 min. Abbildung 5.5a zeigt die Ergebnisse des Filters für $\alpha = \beta = 0.5$ und $c = 175$. Die große Menge an Rauschen lässt sich nicht alleine durch den dafür vorgesehenen Parameter c eliminieren, da die leicht sichtbare rechte Koronararterie und die rauschenden Voxel anscheinend den gleichen Wert der Metrik erhalten.

Abhilfe schafft hier ein hinter den Filter geschaltetes Median-Plugin mit Radius 2. Das Ergebnis ist in Abbildung 5.5b dargestellt. In Abbildung 5.5c ist das Ergebnis für die Standardparameter und zusätzlichem Median-Plugin dargestellt. Bis auf eine größere Menge an Rauschen ist kein Unterschied zu 5.5b festzustellen. Die Parameter α und β bieten eine weitere Möglichkeit den Filter anzupassen. Abbildung 5.5d zeigt das Ergebnis für $\alpha = 0.01$, $\beta = 0.5$ und $c = 175$ samt Median-Plugin. Die Menge an Rauschen hat sich zwar deutlich verringert, dafür werden die Gefäße aber eher „platt“ dargestellt. Eine Anpassung von β liefert dagegen keine große Veränderung.

Auffällig ist, dass einzig die rechte Koronararterie in voller Länge erfasst wird, die Äste der linken Koronararterie dagegen nur ansatzweise im oberen rechten Bereich (teilweise verdeckt durch Lungengefäße) zu sehen sind. Auch eine Anpassung der Größenparameter brachte hier keine Verbesserung. Zudem werden die Koronararterien nicht überall mit dem vollen Radius erfasst, so dass Verengungen entstehen, die eigentlich nicht vorhanden sind.

Bei der Evaluation des Filters von Pock et al. wurde der Datensatz ebenfalls zugeschnitten und zusätzlich mit einem Downsampler auf die halbe Größe gebracht, um die Laufzeit zu senken. Die Größenparameter des Filters wurden auf $r_{\min} = 1$ und $r_{\max} = 5$ festgelegt. Die Bearbeitung des $142 \times 168 \times 103$ Voxeln messenden Datensatzes mit diesen Einstellungen dauerte auf einem Core 2 Duo mit 2.53 GHz ca. 2 min ohne Rückprojektion und ca. 5 min mit Rückprojektion. Aufgrund der großen Menge an Rauschen wurde der dafür vorgesehene Parameter η auf 1 gesetzt.

Abbildung 5.6a zeigt die Ergebnisse des Filters für $n = 5$ und ohne Rückprojektion. Zusätzlich zum hohen Anteil an Bildrauschen ist die Mittellinie der rechten Koronararterie

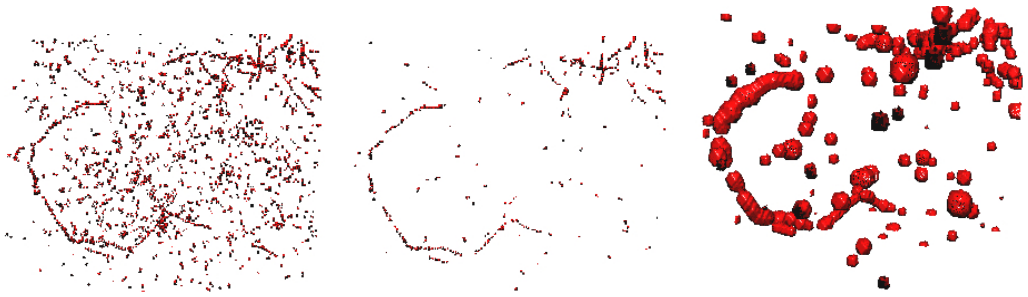


(a) (b) (c) (d)

Abbildung 5.5: Ergebnisse des Filters von Frangi et al. für unterschiedliche Werte der Parameter.

erkennbar. Das Rauschen kann durch Erhöhen des Parameters n auf 10 noch verringert werden. Dies ist in Abbildung 5.6b dargestellt. Da im Gegensatz zu den synthetischen Datensätzen ein hoher Anteil von Rauschen im Bild vorhanden ist, funktioniert die Rückprojektion hier nur bedingt (siehe Abbildung 5.6c, $\eta = 1$, $n = 10$). Weiterhin fällt auch bei diesem Filter auf, dass zwar die rechte Koronararterie annähernd vollständig erfasst und auch die Dicke des Gefäßes ziemlich genau abgeschätzt wird, die linke Koronararterie aber nur ansatzweise vorhanden ist.

Ein großes Problem beider Filter ist die Erkennung von durch Rauschen verursachten Merkmalen. Zwar beruhen die Verfahren auf einer zunehmenden Diffusion des Volumens, wodurch Rauschen effektiv unterdrückt wird, allerdings wird dies erst bei verhältnismäßig großen Strukturen erkennbar sein. Herzkranzgefäße dagegen sind relativ klein, so dass sie gerade in den Größen des Scale-Space erkannt werden, in denen das Rauschen noch stark vorhanden ist. Abhilfe könnten hier Datensätze mit höherer Auflösung schaffen.



(a) (b) (c)

Abbildung 5.6: Ergebnisse des Filters von Pock et al. für unterschiedliche Werte der Parameter. (a) und (b) ohne, (c) mit Rückprojektion.

Dazu wäre aber eine weitaus effektivere Implementation der Filter, z. B. auf der GPU, nötig. Da beide Verfahren mit relativ wenig Aufwand parallelisiert werden können, böte sich hier die Verwendung der Technologie OpenCL[4] an.

Ein weiterer Ansatz wäre die Verwendung eines besseren Thinning-Algorithmus, wie z. B. den von Malandain et al. [MBA93], durch den noch mehr, durch Rauschen entstandene Maxima entfernt werden können.

[4] http://www.khronos.org/opencl/

6 Zusammenfassung und Ausblick

In dieser Arbeit wurden zwei verschiedene Verfahren zur Erkennung von röhrenartigen Strukturen in Volumendatensätzen vorgestellt und versucht, diese auf das Verfahren von Correa und Ma zur Erstellung größenbasierter Transferfunktionen zu übertragen, um Herzkranzgefäße hervorheben zu können.

Dazu wurden zunächst die Grundlagen der medizinischen Bildgebung, sowie des Volumenrenderings erörtert. Es folgte eine Erklärung der grundlegenden mathematischen Begriffe zur Erstellung größenbasierter Transferfunktionen, wie Scale-Space, Größenabstraktion und Rückprojektion. Dabei wurde auch auf die Ziele der Metrik Größe eingegangen. Im Anschluss erfolgte eine Erläuterung der Mathematik der vorgestellten Verfahren von Frangi et al. und Pock et al. Hierbei wurde insbesondere auf die Eigenschaften der Hesse-Matrix und deren Eigenwerten und -vektoren aufmerksam gemacht. In einem nächsten Schritt wurde versucht, die Ergebnisse der beiden Verfahren so zu bearbeiten, dass sie für eine Rückprojektion benutzt werden können. Hierzu ist insbesondere die Erkennung der Maxima und der Ausschluss von Ausreißern notwendig.

In Kapitel 4 wurde schließlich die Implementation der Verfahren vorgestellt. Dazu ist zunächst auf das Plugin-Konzept der Software *VolumeStudio2* eingegangen worden. Dem folgte eine Erklärung, wie die verwendeten mathematischen Operationen in einen diskreten Kontext übertragen und so in der Programmiersprache C++ umgesetzt werden können. Anschließend wurde kurz die Implementation der Verfahren, sowie der Größenabstraktion und Rückprojektion erläutert.

In der Evaluation sind die implementierten Verfahren auf synthetische Datensätze angewandt worden, um zum einen ihre Korrektheit zu validieren und zum anderen ihre Übertragbarkeit auf das Verfahren von Correa und Ma zu ermitteln. Dabei wurde festgestellt, dass nur der Filter von Pock et al. die dazu notwendige Erkennung der Mittellinien von röhrenartigen Strukturen leistet. Im Gegensatz dazu erwies sich der Filter von Frangi et al. als völlig ungeeignet für die vorgestellte Rückprojektion. Allerdings lieferte die direkte Verwendung der berechneten Metrik ein ähnliches Ergebnis, wie der Filter von Pock et al. nach erfolgter Rückprojektion.

Ein großes Problem beider Verfahren wurde bei der Anwendung auf reale CT-Datensätze des Herzens deutlich. Da Herzkranzgefäße nur eine geringe Größe besitzen, werden sie von den Metriken hauptsächlich in den Größen des Scale-Space entdeckt, in denen noch eine große Menge an Rauschen vorhanden ist. Beide Verfahren besitzen zwar Parameter zur Rauschunterdrückung, jedoch beeinflussen diese auch die korrekte Erkennung der Herzkranzgefäße. Beim Filter von Frangi et al. konnte ein Großteil des Rauschens durch einen Median-Filter unterdrückt werden, allerdings wies die Darstellung der Herzkranzgefäße Verengungen auf, die im Originaldatensatz nicht vorhanden waren.

Noch gravierender wirkte sich das Rauschen beim Filter von Pock et al. aus. Die Mittellinien der Herzkranzgefäße werden zwar erkannt und auch die geschätzte Größe stimmt annähernd mit der Realität überein. Allerdings verursachen durch Rauschen erkannte Merkmale nach der Rückprojektion erhebliche Artefakte, die die Qualität der Bilder deutlich mindern.

Abhilfe könnte hier ein verbesserter Thinning-Algorithmus schaffen, der noch mehr Ausreißer unter den erkannten Mittellinien aussortiert. Eine anschließende Rückprojektion sollte dann ein gutes Ergebnis bringen. Auch eine Anpassung der Rückprojektion wäre denkbar. Zudem könnten Datensätze mit höherer Auflösung bzw. höherem Signal-Rausch-Verhältnis zu besseren Ergebnissen führen. Dazu wäre aber eine deutlich effektivere und parallelisierte Implementation der Algorithmen, z. B. mit Hilfe der Technologie OpenCL, notwendig.

Abschließend lässt sich sagen, dass die Größe von Strukturen zwar eine sinnvolle und zudem intuitiv erfassbare Metrik zur Hervorhebung von Merkmalen ist. Allerdings lässt sich dieses Konzept nicht so leicht auf die Hervorhebung von Herzkranzgefäßen spezialisieren. Die beiden vorgestellten Verfahren liefern durchaus annehmbare Ergebnisse für ihren ursprünglichen Verwendungszweck, nämlich die direkte Hervorhebung von Gefäßstrukturen (Frangi et al.) und die Extraktion der Mittellinien von röhrenartigen Strukturen (Pock et al.). Um diese aber effektiv für die Hervorhebung von Röhren anhand ihrer Größe nutzen zu können, ist eine noch intensivere Forschungsarbeit notwendig. Zudem stellen die beiden Verfahren nur einen Bruchteil aller Metriken zur Röhrenerkennung dar, so dass vielleicht ein anderer Algorithmus existiert, der sich leichter übertragen lässt.

Literaturverzeichnis

[BB08] Bauer, Christian ; Bischof, Horst: A Novel Approach for Detection of Tubular Objects and Its Application to Medical Image Analysis. In: *DAGM-Symposium*, Springer-Verlag, 2008, S. 163–172

[BVAS+07] Bennink, H. E. ; Van Assen, H. C. ; Streekstra, G. J. ; Wee, R. T. ; Spaan, J. A. E. ; Romeny, B. M. Ter H.: A novel 3D multi-scale lineness filter for vessel detection. In: *MICCAI'07: Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention*, Springer-Verlag, 2007, S. 436–443

[CM08] Correa, Carlos ; Ma, Kwan-Liu: Size-based Transfer Functions: A New Volume Exploration Technique. In: *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), S. 1380–1387

[EHK+06] Engel, Klaus ; Hadwiger, Markus ; Kniss, Joe ; Rezk-Salama, Christof ; Weiskopf, Daniel: *Real-Time Volume Graphics*. AK Peters, 2006

[FFN+98] Frangi, Alejandro F. ; Frangi, Ro F. ; Niessen, Wiro J. ; Vincken, Koen L. ; Viergever, Max A.: Multiscale Vessel Enhancement Filtering, Springer-Verlag, 1998, S. 130–137

[KMA+00] Krissian, Karl ; Malandain, Greégorie ; Ayache, Nicholas ; Vaillant, Régis ; Trousset, Yves: Model-based detection of tubular structures in 3D images. In: *Comput. Vis. Image Underst.* 80 (2000), Nr. 2, S. 130–171

[LCB+97] Lorenz, Cristian ; Carlsen, I.-C. ; Buzug, Thorsten M. ; Fassnacht, Carola ; Weese, Jürgen: Multi-scale line segmentation with automatic estimation of width, contrast and tangential direction in 2D and 3D medical images. In: *CVRMed-MRCAS '97: Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery*, Springer-Verlag, 1997, S. 233–242

[Lin90] Lindeberg, Tony: Scale-Space for Discrete Signals. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (1990), S. 234–254

[Lin91] Lindeberg, Tony: *Discrete Scale-Space Theory and the Scale-Space Primal Sketch*. Stockholm, Royal Institute of Technology, Diss., 1991

[Lin96] Lindeberg, Tony: Edge Detection and Ridge Detection with Automatic Scale Selection. In: *International Journal of Computer Vision* 30 (1996), S. 465–470

[Lin98] Lindeberg, Tony: Feature Detection with Automatic Scale Selection. In: *International Journal of Computer Vision* 30 (1998), S. 79–116

[MBA93] Malandain, Grégoire ; Betrand, Gilles ; Ayache, Nicholas: Topological segmentation of discrete surfaces. In: *Int. J. Comput. Vision* 10 (1993), Nr. 2, S. 183–197

[PB07] Preim, Bernhard ; Bartz, Dirk: *Visualization in Medicine - Theory, Algorithms, and Applications.* Burlington, USA : Morgan Kaufman Publishers, 2007

[PBB05] Pock, Thomas ; Beichel, Reinhard ; Bischof, Horst: A Novel Robust Tube Detection Filter for 3D Centerline Extraction. In: *18th Symposium on Operating System Principles (SOSP)*, Springer-Verlag, 2005, S. 481–490

[PM90] Perona, P. ; Malik, J.: Scale-Space and Edge Detection Using Anisotropic Diffusion. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 12 (1990), Nr. 7, S. 629–639

[PSS+03] Pizer, Stephen M. ; Siddiqi, Kaleem ; Székely, Gabor ; Damon, James N. ; Zucker, Steven W.: Multiscale Medial Loci and Their Properties. In: *Int. J. Comput. Vision* 55 (2003), Nr. 2-3, S. 155–179

[SK09] Schwarz, Hans R. ; Köckler, Norbert: *Numerische Mathematik.* 7. Vieweg+Teubner, 2009

[SMW+09] Schaap, Michiel ; Metz, Coert T. ; Walsum, Theo van ; Giessen, Alina G. d. ; Weustink, Annick C. ; Mollet, Nico R. ; Bauer, Christian ; Bogunovic, Hrvoje ; Castro, Carlos ; Deng, Xiang ; Dikici, Engin ; O'Donnell, Thomas ; Frenay, Michel ; Friman, Ola ; Hoyos, Marcela H. ; Kitslaar, Pieter H. ; Krissian, Karl ; Kühnel, Caroline ; Luengo-Oroz, Miguel A. ; Orkisz, Maciej ; Smedby Örjan ; Styner, Martin ; Szymczak, Andrzej ; Tek, Hüseyin ; Wang, Chunliang ; Warfield, Simon K. ; Zambal, Sebastian ; Zhang, Yong ; Krestin, Gabriel P. ; Niessen, Wiro J.: Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. In: *Medical Image Analysis* 13 (2009), Nr. 5, S. 701 – 714

[SNS+98] Sato, Yoshinobu ; Nakajima, Shin ; Shiraga, Nobuyuki ; Atsumi, Hideki ; Yoshida, Shigeyuki ; Koller, Thomas ; Gerig, Guido ; Kikinis, Ron: *3D Multi-Scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images.* 1998

[Wit83] Witkin, Andrew P.: Scale-Space Filtering. In: *8th Int. Joint Conf. Artificial Intelligence* Bd. 2, 1983, S. 1019–1022

[XP98] Xu, Chenyang ; Prince, Jerry L.: Snakes, Shapes, and Gradient Vector Flow. In: *IEEE Transactions on Image Processing* 7 (1998), S. 359–369