


Ruprecht-Karls-Universität
Heidelberg

**Diplomarbeit**

# Konzeption und Realisierung eines Datenmodells und Interaktionskomponenten für tubuläre Strukturen in MITK

Abschlussarbeit zur Erlangung des akademischen Grades
Diplom-Informatiker der Medizin (Dipl.-Inform. Med.)

vorgelegt von

**Jasmin Metzger**

Heidelberg, den 13. September 2012

Referent:
Prof. Dr. Rolf Bendl

Koreferent:
Prof. Dr. Hans-Peter Meinzer

Betreuer:
Dipl.-Inform. Med. Marco Nolden

# Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

| Ort, Datum | Unterschrift |
| --- | --- |

# Danksagung

Diese Arbeit entstand am Deutschen Krebsforschungszentrum Heidelberg in der Abteilung für Medizinische und Biologische Informatik. Ich möchte mich bei dem Leiter der Abteilung, Herrn Prof. Dr. Hans-Peter Meinzer, für die Möglichkeit diese Arbeit unter solch angenehmen Arbeitsbedingungen anzufertigen, bedanken. Ebenso möchte ich Herrn Prof. Dr. Rolf Bendl für die Annahme des Themas danken.

Ganz besonderer Dank gilt Herrn Dipl.-Inform. Marco Nolden für die Betreuung meiner Arbeit und für die gute Beratung bei Schwierigkeiten hinsichtlich der Realisierung. Den Kollegen der Abteilung MBI danke ich für die schöne Arbeitsatmosphäre und die vielseitige Unterstützung. Speziell Herrn Andreas Fetzer und Herrn Markus Fangerau, die mir mit Rat und Tat zur Seite standen, möchte ich danken.

Ein großes Dankeschön geht an Diana und Kira für die aufmunternden Worte und das Korrekturlesen.

Sehr dankbar bin ich meinen Eltern, die mir das Studium ermöglicht haben und die mich in jeder Hinsicht auf meinem bisherigen Lebensweg unterstützt haben. Meiner Schwester möchte ich danken, dass sie trotz Hochzeitsstress für mich da war und mich bei der Anfertigung dieser Arbeit unterstützt hat.

Danke an meinen Freund, der mich vor allem am Ende der Arbeit unterstützt, aufgebaut und motiviert, aber auch abgelenkt hat. Zuletzt möchte ich meinen Mitbewohnern, die mir hier in Heidelberg eine tolle “Ersatzfamilie“ sind und allen Freunden, die mich während meines Studiums begleitet haben, danken.

# Zusammenfassung

Eine der zentralen Aufgaben der medizinischen Bildverarbeitung ist es, den Arzt durch neue oder verbesserte Methoden zur Diagnostik und Therapieplanung bei medizinischen Entscheidungs- und Behandlungsprozessen zu unterstützen. Für viele Fragestellungen in der Medizin sind Untersuchungen von Gefäßsystemen oder anderen tubulären Strukturen erforderlich.

Die vorliegende Arbeit beschäftigt sich mit der Konzipierung und der Realisierung eines generischen Datenmodells für tubuläre Strukturen. Für die Darstellung dieser Strukturen wurde eine neue Visualisierungsmethode implementiert. Zudem wurden allgemein verwendbare und benutzerfreundliche Interaktionskomponenten entwickelt, die die Exploration, die Modifikation, die Attributierung sowie die Analyse der modellierten Systeme ermöglichen.

Für den Entwurf des generischen Datenmodells werden Ansätze für eine geeignete symbolische Beschreibung von tubulären Strukturen diskutiert, um daraus eine ausführliche Anforderungsanalyse zu erstellen. Die Realisierung erfolgt auf Basis von ungerichteten Graphen, die mittels Adjazenzlisten verwaltet werden. Das neue Datenmodell erlaubt es alle Gefäßformen, ebenso wie Zyklen, die durch Anastomosen zustande kommen, abzubilden. Durch den generischen Ansatz ist die Einsetzbarkeit für unterschiedliche Anwendungsfälle gewährleistet.

Für die Visualisierung wurde ein eigenes, modellbasiertes Verfahren realisiert, das mittels einfachen Primitiven Oberflächen erzeugt, die die tubulären Strukturen im dreidimensionalen Raum, darstellen. Um eine Navigation durch das Innere der Oberflächen zu ermöglichen, wurde eine Methode entwickelt, die die Fragmente im Inneren der Strukturen entfernt.

Ein besonderer Schwerpunkt dieser Arbeit liegt auf der Konzeption und Realisierung der vielfältigen Interaktionsmuster auf einer dreidimensionalen Graphenstruktur. Hier wurden neben der eigentlichen Interaktionslogik flexible Benutzeroberfläche entwickelt, die für unterschiedliche Anwendungsszenarien verwendet werden können.

Um einen möglichst vielseitiger Einsatz des implementierten Tools zu gewährleisten sind die verwendeten Verfahren und Modelle allgemein und unabhängig von der Bildmodalität und den betrachten tubulären Strukturen realisiert worden. Zudem ist dieses modularisierte Tool im Rahmen des Medical Imaging Interaction Toolkit (MITK) realisiert worden und wird als Open Source zur Verfügung gestellt.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **BGL** | Boost Graph Library |
| **CT** | Computertomographie |
| **ITK** | Insight Segmentation and Registration Toolkit |
| **MBI** | Abteilung für Medizinische und Biologische Informatik des DKFZ |
| **MITK** | Medical Imaging Interaction Toolkit |
| **MRT** | Magnetresonanztomographie |
| **VMTK** | Vascular Modeling Toolkit |
| **VTK** | Visualization Toolkit |
| **XML** | Extensible Markup Language |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1. Einführung

## 1.1. Motivation

Eine der zentralen Aufgaben der medizinischen Bildverarbeitung ist es, den Arzt durch neue oder verbesserte Methoden zur Diagnostik und Therapieplanung bei medizinischen Entscheidungs- und Behandlungsprozessen zu unterstützen.

Für viele Fragestellungen in der Medizin sind Untersuchungen von Gefäßsystemen oder anderen tubulären Strukturen erforderlich. In medizinischen Gebieten wie Chirurgie, Kardiologie und Neurologie spielen sie eine wichtige Rolle. Medizinische Bildgebungsverfahren wie Computertomographie (CT) und Magnetresonanztomographie (MRT) stellen hier zweidimensionale Sichten auf das Untersuchungsgebiet zur Verfügung, dabei setzt die Interpretation der Schichtbildserien ein hohes Maß an Erfahrung voraus. Eine dreidimensionale Visualisierung erleichtert dem Anwender die Interpretation von solchen Datensätzen. Dies kann zu einer Verbesserung der Qualität von Diagnose, Therapie und Therapiekontrolle führen, da eine genaue Kenntnis der Lage und des Verlaufs der Gefäße – z.B. in Relation zu einem Tumor – hierbei eine wichtige Information sind. Hiermit können zum Beispiel Versorgungsgebiete der Gefäße berechnet und bei einer Resektion Nekrosen vermieden werden.

Durch die stetige Verbesserung der Bildaufnahmetechniken erhöht sich die Auflösung der Daten zunehmend. Dies hat zur Folge, dass immer feinere, tubuläre Strukturen, so wie tiefer gelegene Verzweigungen detailliert dargestellt werden können. Durch eine geeignete dreidimensionale Repräsentation wird die räumliche Erfassung dieser Strukturen erleichtert und eine Extraktion von quantitative Größen ermöglicht. Zudem können Operationszeiten durch gezielte computergestützen Operationsplanungen und intra-operative Navigation verkürzt werden. Aber auch die Qualität der medizinischen Eingriffe verbessert sich.

Die Verfügbarkeit solcher dreidimensionalen Datensätze verlangt aber auch eine angemessene Auswertung. Für diese rechnergestützte Auswertung ist es hilfreich durch interaktives Attributieren Differenzierungen der funktionellen und anatomischen Einheiten vorzunehmen. Auch die Exploration solcher röhrenartiger Strukturen ist unabdingbar. Für eine solche Interaktion müssen die tubulären Systeme in adäquater Weise durch eine symbolische Beschreibung repräsentiert werden. Um eine Erleichterung bei der Auswertung von tubulären Strukturen zu erreichen, müssen geeignete Methoden zur Repräsentation, zur Visualisierung und zur Interaktion geschaffen werden.

Für diese Anforderungen existieren nur projektbezogene Ansätze. Ein allgemeines Werkzeug, das neben der Modellierung und der Visualisierung von tubulären Strukturen auch variable Interaktionskomponenten bereitstellt, existiert nicht.

## 1.2. Zielsetzung

Die vorliegende Arbeit beschäftigt sich mit der Konzipierung, sowie der Realisierung eines neuen Datenmodells für tubuläre Strukturen. Für die Darstellung dieser Strukturen soll zudem eine geeignete Visualisierungsmethode entwickelt werden. Im letzten Schritt sollen allgemeine, benutzerfreundliche Interaktionskomponenten entworfen werden, die die Exploration, die Modifikation, die Attributierung sowie die Analyse der modellierten Systeme ermöglichen.

Die verwendeten Verfahren und Modelle sollen möglichst allgemein und unabhängig von den verwendeten Bildmodalitäten oder den betrachteten tubulären Strukturen gehalten sein. Um einen möglichst vielseitiger Einsatz des implementierten Tools zu gewährleisten. Die Implementierung dieses modularisierten Tools für die Modellierung und Interaktion von tubulären Strukturen soll im Rahmen des Medical Imaging Interaction Toolkit (MITK) realisiert und auf diese Weise der wissenschaftlichen Gemeinschaft als Open Source zur Verfügung gestellt werden.

## 1.3. Aufbau der Arbeit

In diesem Abschnitt soll dem Leser ein Überblick über den Aufbau der Arbeit und die Inhalte der einzelnen Kapitel gegeben werden. Die folgenden Kapitel gehören zwar thematisch zusammen und bauen aufeinander auf, können jedoch auch unabhängig voneinander gelesen und verstanden werden.

In *Kapitel 2* werden die wichtigsten Grundlagen beschrieben, die dem Verständnis der Arbeit dienen sollen. Es werden hierbei auf medizinische Anwendungsfälle und ihre anatomischen Besonderheiten eingegangen. Zudem werden die in dieser Arbeit verwendeten Software Toolkits und Bibliotheken vorgestellt.

In *Kapitel 3* werden verwandte Arbeiten beleuchtet und bisherige computergestützte Therapieplanungssystem in MITK betrachtet. Zusätzlich werden Toolkits, die sich mit Gefäßen und tubulären Strukturen im Allgemeinen befassen, vorgestellt.

Die *Kapitel 4-6* sind gleichermaßen aufgebaut: Zu Beginn jedes Kapitels werden die themenbezogenen Grundlagen beschrieben. Anschließend erfolgt eine Anforderungsanalyse, sowie die Beschreibung der Methode, die umgesetzt wurde. Jedes Kapitel schließt mit den resultierenden Ergebnissen der Aufgabe ab.

*Kapitel 4* befasst sich mit der symbolischen Beschreibung von tubulären Strukturen, also dem Datenmodell. *Kapitel 5* beschäftigt sich mit der Visualisierung und *Kapitel 6* beschreibt die Interaktionskomponenten.

In *Kapitel 7* wird der Gesamtansatz diskutiert. Hierbei werden einzelne Aspekte der in dieser Arbeit realisierten Verfahren kritisch beleuchtet und weitere Einsatzmöglichkeiten aufgezeigt. Des Weiteren wird ein Ausblick auf weiterführende Entwicklungsmöglichkeiten gegeben.

# 2. Allgemeine Grundlagen

In diesem Kapitel werden medizinischen und technischen Grundlagen, die für das Grundverständnis der Diplomarbeit wichtig sind, vermittelt.

Dabei werden Anwendungsfälle innerhalb der computergestützten Therapieplanung, sowie deren anatomischen Besonderheiten betrachtet. Des Weiteren werden das MITK als Entwicklungsumfeld dieser Arbeit, sowie die darin verwendeten Software Toolkits und externe Bibliotheken vorgestellt.

Weiterführende Grundlagen und Konzepte, die für das Datenmodell, die Visualisierung und die Interaktion wichtig sind, werden in den jeweiligen themenbasierten Kapiteln behandelt.

## 2.1. Computergestützte Therapieplanungssysteme

Die genaue Analyse von vaskulären Systemen und anderen tubulären Strukturen in unterschiedlichen medizinischen Anwendungsgebieten gewinnt zunehmend an Bedeutung. Kenntnisse der Morphologie und der Struktur solcher Systeme können in der quantitativen Diagnosestellung, Planung von chirurgischen Operationen und Abgleich des Behandlungsergebnisses, sowie für das Beobachten von vaskulären Krankheitsverläufen optimal eingesetzt werden.

### 2.1.1. Gefäßdiagnostik

#### Blutgefäßsystem

Der Blutkreislauf besteht aus dem Herzen und dem kardiovaskulären System und bildet somit das Strömungssystems des Blutes. Es versorgt die Körperzellen mir Sauerstoff und transportiert das entstandene Kohlendioxid ab. Zudem werden Nährstoffe aus dem Verdauungstrakt in einzelne Gewebe transportiert und dort entstanden Stoffwechsel- oder Abbauprodukte werden dann in anderes Gewebe oder Ausscheidungsorgane gebracht.

Dieses vaskuläre System besteht aus unterschiedlichen Gefäßen die anhand ihres Aufbaues und ihrer Funktion unterschieden werden. So wird das Blut das vom Herzen kommt in den *Arterien* weitergeleitet. Da hier eine hohe Fließgeschwindigkeit besteht, müssen die Gefäßwände dicker sein. Der Blutdruck kann damit stabil gehalten werden. Die Arterien verzweigen sich zu den kleineren *Arteriolen*, die die Funktion von Kontrollventilen haben. Über ihre stark muskulären Wände können sie das Gefäß verengen oder weiten, womit zum Beispiel ein Verschluss der Arteriolen möglich ist, um wenig beanspruchte Gewebe und Organe auch weniger zu durchbluten. Die kleinsten Gefäße sind die *Kapillaren*. Sie schließen sich den Arteriolen an und bilden ein feines Netzwerk. Hier erfolgt der

Austausch von Sauerstoff, Kohlendioxid, Nährstoffe, Hormone, Elektrolyte und anderen Stoffe. Für diesen Austausch sind dünne Gefäßwände notwendig, die semipermeable sind. Aus dem venösen Abschnitt der Kapillaren entspringen die *Venolen*, die sich vereinigen und zu den *Venen* zusammenschließen. Sie führen das Blut zurück zum Herzen. Venen besitzen zu dem *Venenklappen* (lat. *Valvula*), die das zurück fließen des Blutes auf Grund der Schwerkraft in die Venolen verhindern.

Bei der computergestützten Darstellung von Gefäßen stellen krankhafte Veränderungen eine besondere Herausforderung dar. So kann es zum Beispiel zu *Aneurysmen*, also zu Ausbuchtungen in der Gefäßwand, kommen. Diese können sehr unterschiedliche Formen ausbilden und führen daher zu einer Gefäßdeformation. Des Weiteren sind *Stenosen* eine weit verbreitete, krankhafte Veränderung von Gefäßen. Durch Ablagerungen in Form von Cholesterin, Kalk, Blutfetten, Bindegewebe und Blutgerinnsel kommt es hierbei zu solchen Verengungen der Gefäße. Des weiteren gibt es besonderen Fälle in denen sich krankhafte *Anastomosen* bilden, also Neubildungen von Verbindungen zwischen zwei Gefäßen. Die so entstehenden Zyklen müssen gesondert bei der Darstellung beachtet werden. Die Vereinigung von Gefäßen wird zudem oftmals in operativen Eingriffen künstlich erzeugt.

### Lebergefäßsystem

Um die Anforderungen an die Interaktionen mit Lebergefäßen innerhalb der Leberoperationsplanung definieren zu können, muss auf die anatomischen Besonderheiten eingegangen werden.

Die Leber (lat. *Hepar*) ist das größte parenchymatöse Organ des Menschen. In ihrer zentralen Rolle des Stoffwechsels produziert sie lebenswichtige Eiweißstoffe und Gallenflüssigkeit, verwertet Nahrungsbestandteile und ist zudem für den Abbau von Stoffwechselprodukten, Medikamenten und Giftstoffe zuständig. Sie liegt größtenteils im rechten Oberbauch und ist mittels mehreren Leberbändern in der Bauchhöhle befestigt. Durch das *Ligamentum falciforme* trennt sich die Leber optisch in einen rechten Leberlappen (lat. *Lobus dexter*) und einen kleineren linken Lappen(lat. *Lobus sinister*). An der Unterseite der Leber befinden sich weiterhin der quadratischen Lappen (lat. *Lobus quadratus*) und der geschwänzte Lappen (lat. *Lobus caudatus*), die jedoch aufgrund der Gefäßversorgung dem linken Lappen zugeordnet werden.

Zwischen diesen beiden Lappen liegt die Leberpforte (lat. *Porta hepatis*). Sie ist Eintrittsstelle der Leberarterie (lat. *Arteria hepatica propria*) und der Pfortader (lat. *Vena portae*), sowie Austrittspunkt der intrahepatische Gallengänge (lat. *Ductus biliferus*). Die Pfortader sammelt das nährstoffreiche, aber sauerstoffarme Blut aus den Venen der unpaaren Bauchorgane und führt es der Leber zu. Hierdurch werden die Nährstoffe aus den Verdauungsorganen, die Abbauprodukte der Milz, sowie die Hormone der Bauchspeicheldrüse transportiert. Dahingegen transportiert die Leberarterie das sauerstoffreiche Blut vom Herzen zur Leber.

Für operative Zwecke ist die Einteilung der Leber in Segmente nach dem französischen Chirurg *Claude Couinaud* von Bedeutung. Nach ihm wird die Leber in acht funktionelle Segmente eingeteilt, die von je einem Ast der Pfortader, der Arterie und des Gallenganges versorgt werden. Der *lobus caudatus* ist Segment I und ist in der inferioren Sicht zu sehen.

Alle andere Segmente sind wie in Abbildung 2.1 zu sehen ist, im Uhrzeigersinn, beginnend mit der linken Spitze der Leber, angeordnet. Später kam die Unterteilung von Segment IV in IVa und IVb hinzu.

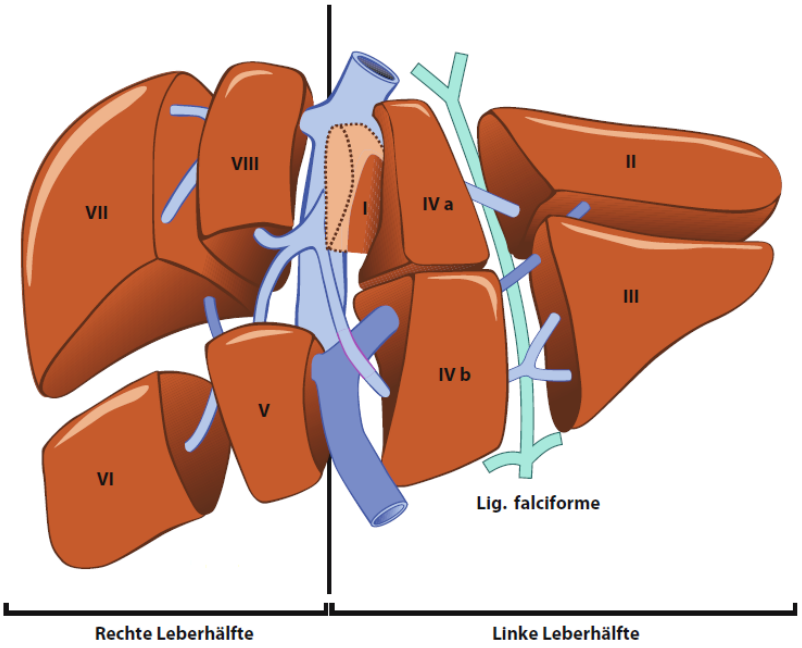


Abbildung 2.1.: Segmentaufteilung der Leber nach C. Couinaud von 1957: Neben den einzelnen Segmenten ist das Lebervenensystem (hellblau) und das Portalvenensystem (dunkelblau) dargestellt. (Quelle: [33])

### Computergestützte Leberoperationsplanung

Auf Grund der ausgeprägte Fähigkeit zur Regeneration der Leber können bei gutartigen oder bösartigen Erkrankungen Teile des Organs entfernt werden. Entfernungen von solitären und begrenzten Tumoren treten hierbei am häufigsten auf. Zudem können auch Abszesse und progrediente Zysten eine Teilresektion bedingen. Je nachdem wo sich der zu entfernenden Teil befindet, gibt es die *typische Resektionen*, die sich nach den anatomischen Begebenheiten richten. So wird bei der *Hemihepatektomie* ein gesamter Leberlappen entnommen. Es ist aber auch möglich einzelne, nebeneinander liegende Segmente, ihrer jeweiligen Blutversorgung entsprechend, zu entfernen. Bei randnahen Gebieten kann auch *atypische Resektionen*, sogenannte *Keilresektionen*, durchgeführt werden, bei denen Verletzungen von größeren intrahepatischen Gefäßen vermieden werden. Wichtig ist es, bei der Entfernung von Teilleberstücken zu berücksichtigen, dass genügend gesundes Leberparenchym übrig bleibt. Bei Patienten deren Leberparenchym nicht angegriffen ist, kann bis zu 80% der Leber entfernt werden.[33]

Für die Entfernung eines Tumors in der Leber muss die Lage in Bezug zu den Gefäßen

innerhalb der Leber berücksichtigt werden. Gefäße, die innerhalb eines Sicherheitsabstandes um den Tumor durch die Leber ziehen, müssen zusammen mit den versorgten Gewebebereichen entfernt werden, da diese ansonsten von der Durchblutung abgeschnitten und damit absterben würden.

Zudem kann im Rahmen der Transplantationschirurgie *Leber-Lebendspende* durchgeführt werden. Hierbei werden einem gesunden Menschen ein Teil der Leber, bei einer Spende von einem Erwachsenen auf einen anderen die rechte Leberhälfte, entnommen und dem leberkranken Patienten transplantiert. Die Entnahme richtet sich hierbei nach den Vorgehensweisen einer Teilresektion.

Da die individuelle Anatomie der Leber von Patient zu Patient stark variiert, ist eine computergestützte Leberoperationsplanung von großem Nutzen. Auf Grundlage von kontrastmittelverstärkten, zweidimensionalen CT-Aufnahmen werden dreidimensionale Modelle erstellt. Diese enthalten dann die dreidimensionale Struktur der Leber, der Gefäßbäume, sowie gegebenenfalls den Tumor. Mit Hilfe dieses Modells können Risiken bei Eingriffen an der Leber berechnet und somit der bestmögliche Resektionsvorschlag geplant werden. Zudem kann das Leberrestvolumen, sowie die Versorgungsgebiete bei der computergestützten Operationsplanung im Vorfeld berechnet und angezeigt werden.

### 2.1.2. Bronchoskopie

#### Tracheobronchialbaum

Das Atmungssystem (*Respiratorisches System*) teilt sich in den oberen Respirationstrakt, mit Nase, Nasennebenhöhlen, sowie Rachenraum und den unteren Bereich, mit Kehlkopf, Luftröhre, Bronchien sowie der Lunge selbst. In der Folge wird sich auf eine Beschreibung der Bronchien und der Lunge konzentriert.

Die Lunge besteht aus zwei Lungenflügeln, die in der Brusthöhle liegen. Im Mediastinum, also im Bereich zwischen den Lungenflügel, liegt, leicht nach links verschoben, das Herz. Auf Grund dieser Asymmetrie ist der linke Lungenflügel etwas kleiner und wird nur in zwei Lappen eingeteilt. Im Gegensatz zu dem rechten Flügel der aus drei Lappen besteht.

Zur Ergänzung der Lappung werden die Lungenflügel noch in *Segmente* eingeteilt. Auf der linken Seite sind es neun Lungensegmente und auf der rechten zehn. Diese Abgrenzung ist nicht durch äußere anatomischen Landmarken zu unterscheiden. Sie definieren sich über die sogenannten *broncho arterielle Einheiten*. Das bedeutet jedes Segment wird jeweils von einem Segmentbronchus und einem Segmentast der Lungenarterie versorgt.

Die sich an der *Bifurcatio trachea* aufteilende Luftröhre (lat. *Trachea*) ist ein muskulöser Schlauch und besteht aus C-förmigen Knorpelspangen, die die strukturelle Stabilität gewährleisten. Beide Hauptbronchien (lat. *Bronchus principalis dexter/sinister*) entspringen dieser Teilung auf der Höhe des 4. und 5. Brustwirbels. Hierdurch ist die Bronchienwand ähnlich aufgebaut wie die Wand der Trachea. Eine weitere Aufteilung erfolgt nach wenigen Zentimetern. Der rechte Bronchus teilt sich in drei kleinere Bronchien. Der linke hingegen in nur zwei. Da dies der Lappenteilung der Lunge gleichkommt, spricht man auch von Lappenbronchien (lat. *Bronchi lobares*). Diese fünf Hauptäste teilen sich wei-

ter in Segmentbronchien (lat. *Bronchi segmentales*), die sich wiederum in immer kleiner werdende Äste verzweigen (siehe Abbildung 2.2).

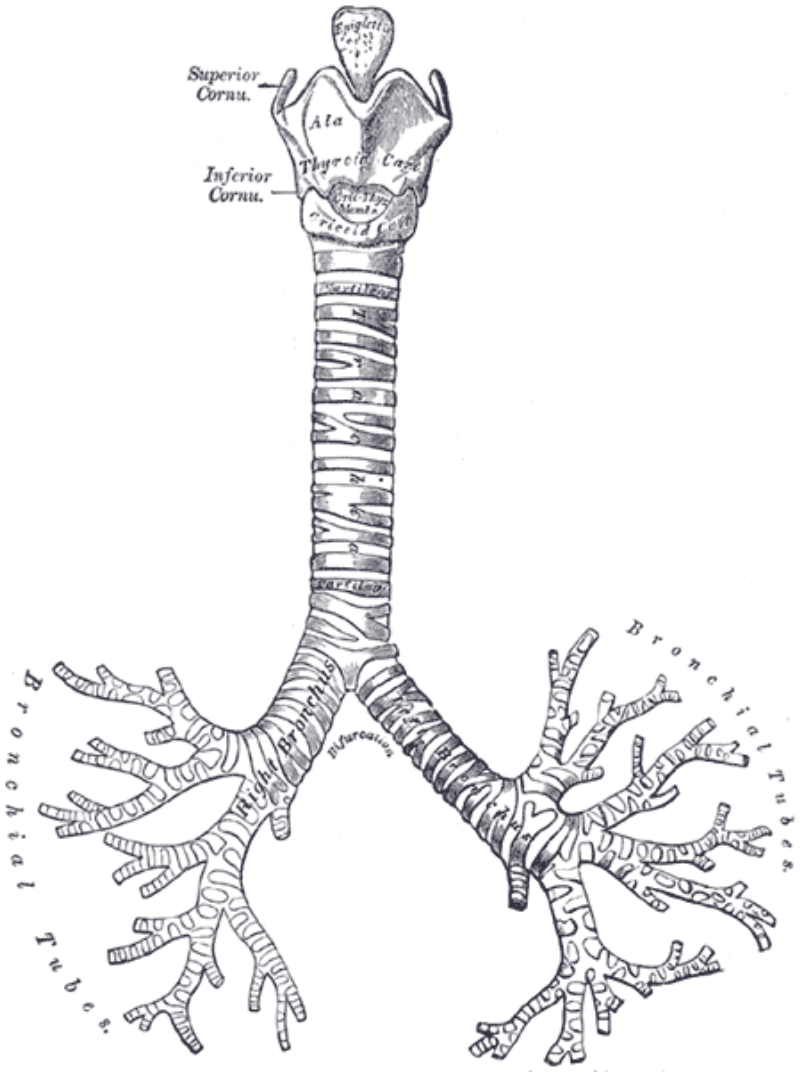


Abbildung 2.2.: Tracheobronchialbaum (Quelle: [15])

Je kleiner die Bronchien werden, desto einfacher und dünnwandiger wird ihr innerer Aufbau. Die Knorpelspangen werden durch Knorpelplättchen ersetzt. In den Bronchiolen (lat. *Bronchioli*), den kleinsten Verzweigungen, verschwinden die Knorpeleinlagerungen vollständig. Sie werden durch Muskelfasern gestützt, die den Luftstrom aktiv regulieren. Das eigentliche atmende Lungengewebe, die Alveolen (lat. *Alveolus*), sind am Ende der Verzweigungsstruktur, mikroskopisch klein zu finden. Hier sind Blut und Luft nur durch die sogenannte *Blut-Luft-Schranke* voneinander getrennt. Der Sauerstoff aus der Alveolarluft kann in das Kapillarblut eintreten und das Kohlendioxid umgekehrt austreten.

Eine Übersicht über die Verzweigungsstrukturen mit Angabe des Durchmesser und ihrer Anzahl ist in Tabelle 2.1 zu sehen.

| Ebene | Bezeichnung | Durchmesser | Anzahl |
|---|---|---|---|
| 0 | Trachea | 13 - 22mm | 1 |
| 1 | Hauptbronchus | 11 - 15mm | 2 |
| 2 | Lappenbronchus | 8 - 12mm | 5 |
| 3a | Segmentbronchus | 4 - 10mm | 18-19 |
| 3b | Rami subsegmentalis | 4 - 10mm | 38 |
| 4 | Primärbronchiol | 0,3 - 5mm | variierend |
| 5 | Endbronchiol | 0,3 - 1mm | variierend |
| 6 | Respiratorischer Bronchiol | 0,4mm | variierend |
| 7 | Alveolargang | 0,25 - 0,5mm | 1,5 - 2 Mio. |
| 8 | Alveole | 0,06 - 0,2mm | ca. 300 Mio. |

Tabelle 2.1.: Verzweigungsstrukturen des Bronchialbaums

### Navigierte Bronchoskopie

Bei der Spiegelung der Bronchien, der sogenannten *Bronchoskopie*, wird ein Endoskop über Mund oder Nase durch die Trachea in die Bronchien der Lunge geschoben. Dieses Verfahren wird für Untersuchungszwecke, aber auch zur Therapie eingesetzt. So können bei diagnostischem Einsatz Lungentumore gesucht, Proben entnommen, Einengungen der Atemwege abgeklärt oder andere Fremdkörper detektiert werden. Der therapeutische Einsatz ermöglicht das Freispülen und Absaugen von Schleimpfropfen aus den Bronchien, eine fiberoptische Intubation, das Einsetzen eines Stents oder die Korrektur der Lage von Beatmungsschläuchen.

Im Gegensatz zur *starren Bronchoskopie* können bei der *flexiblen Bronchoskopie*, durch den Einsatz von flexiblen Instrumenten, die Verletzungsgefahr verringert und die Eindringtiefe erhöht werden. Zudem ist dieses Verfahren schonender für den Patienten, da dieser nicht in Narkose versetzt werden muss. Neben dem diagnostischen Einsatz dient die flexible Bronchoskopie auch der Planung von lokalen Tumor- bzw. Strahlentherapien.

Problematisch ist es den optischen Kanal in tiefer liegende Untersuchungsgebiete zu führen, da mit zunehmender Tiefe sich die Anzahl der Verzweigungen erhöht. Deshalb wird bei der navigierten Bronchoskopie versucht, die aktuelle Position des Bronchoskops relativ zu einem präinterventionell aufgenommenen CT-Bild über ein elektromagnetisches Trackingsystem anzuzeigen. Für die Anzeige der Position ist die dreidimensionale Darstellung des Bronchialbaums, aber auch die Innenansicht der Bronchien unabdingbar. Bei dem Tracking des Bronchoskops stellt die Atembewegung des Patienten allerdings eine wesentliche Herausforderung dar.

## 2.2. Software Toolkits und externe Bibliotheken

In vielen Bereichen der Softwareentwicklung ist die Wiederverwendung bereits implementierter Algorithmen und Methoden ein fester Bestandteil des Entwicklungsprozesses. Um dies zu erleichtern, werden Klassen innerhalb eines Anwendungskontexts zu Bibliotheken oder Toolkits zusammengefasst. Die im Folgenden vorgestellten C++-Softwarebibliotheken sind alle Open Source und plattformunabhängig.

### 2.2.1. Medical Imaging Interaction Toolkit

Das Medical Imaging Interaction Toolkit (MITK)[1] dient der vereinfachten Erstellung von interaktiven, medizinischen Bildverarbeitungsprogrammen. Es ist seit November 2002 die Basis für alle Entwicklungen in der Abteilung für Medizinische und Biologische Informatik (MBI) des Deutschen Krebsforschungszentrums in Heidelberg. Somit ist eine kontinuierlich Weiterentwicklung gegeben. Durch die Kombinierbarkeit und Wiederverwendbarkeit einzelner Module kann die Entwicklungszeit verkürzt werden und höhere Softwarequalität wird erreicht. Der Quellcode richtet sich nach den Grundsätzen der generischen Programmierung und ist plattformunabhängig. Dabei werden mehrere externe Toolkits und Softwarebibliotheken integriert, womit MITK auch die Visualisierungs-Algorithmen des Visualization Toolkit (VTK), sowie die Segmentierungs- und Registrierungs-Algorithmen von Insight Segmentation and Registration Toolkit (ITK) vereint. Es erweitert diese unter anderem um folgende Funktionalitäten [41]:

- Semantische Organisation der Daten in einem Datenbaum
- Vereinfachung und Konsistenz multipler Ansichten auf einen Datensatz
- Erzeugung und Veränderung von Daten mittels generischer Interaktionsmuster
- Realisation komplexer Interaktionen durch hierarchische Zustandsdefinition sowie Undo/Redo-Funktionalität
- Einheitliche Geometrie-Beschreibungen für die Lage von Objekten im Raum und der darzustellenden Bereiche
- Einheitliche Beschreibung sowie Visualisierung und Interaktion von zeitaufgelösten dreidimensionalen Datensätzen (3D+t)

Neben den Algorithmen werden auch Designprinzipien und -konzepte aus VTK und ITK aufgegriffen. Eine genauere Beschreibung der Toolkits erfolgt in Abschnitt 2.2.2 und Abschnitt 2.2.3.

Neben vielen kleineren, flexibel einsetzbaren Komponenten zur Bildverarbeitung und -visualisierung enthält MITK verschiedene komplexe Module und Anwendungen zur interaktiven Segmentierung, sowie zur Registrierung und zur Ansteuerung unterschiedlicher Trackingssystemen für die navigierte Chirugie.

[1] MITK ist unter `http://www.mitk.org` verfügbar.

Das Ziel des Medical Imaging Interaction Toolkits ist es, den Nutzen von externen Toolkits zu vereinen und sie durch Umsetzung komplexer Interaktionsmechanismen und weiterer Bildverarbeitungsalgorithmen zu ergänzen. Durch die Kombination und Wiederverwendung der kontinuierlich weiterentwickelten Komponenten können Anwendungen der interaktiven medizinischen Bildverarbeitung schnell und effizient realisiert werden.

### 2.2.2. Visualization Toolkit

Das Visualization Toolkit (VTK)[2] ist ein Softwaresystem für Computergrafik, Bildverarbeitung, Volume Rendering, sowie Visualisierung. Entstanden ist es 1993 im Rahmen einer Begleitsoftware für das Buch „The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics“, deren Autoren drei Wissenschaftler (Will Schroeder, Ken Martin und Bill Lorensen) von GE Corporate R&D waren. Von diesen wurde das Toolkit weiterentwickelt, was große Aufmerksamkeit erregte. In der Folge entschlossen sich Martin und Lorensen im Jahr 1998 deshalb zur Gründung der eigenen Firma Kitware Inc.[3]

Neben den grafischen Werkzeugen verhilft VTK zur Interaktion mit dreidimensionalen Szenen, sowie zur Anzeige und Belegung der Objekte innerhalb der Szene mit Informationen in Form von Texten oder Markern. Allerdings sind die Interaktionen auf einem unteren Level einzuordnen. Für die Analyse und Auswertung von medizinischen Bilddatensätzen sind sie nicht ausreichend.

### 2.2.3. Insight Segmentation and Registration Toolkit

Das Insight Segmentation and Registration Toolkit (ITK)[4] bietet Algorithmen und Datenstrukturen für Segmentierung und Registrierung in der Bildverarbeitung. Dieses Toolkit wird von der Firma Kitware verwaltet. Das Projekt startete im Jahr 1999 und ist eine von der National Library of Medicine geförderte Entwicklung eines Toolkits. Hauptmotivation des Projektes war die Unterstützung des Visible Human Projects durch die Vereinfachung der Applikationsentwicklung im Bereich der medizinischen Bildverarbeitung.

Der frei verfügbare Quellcode unterstützt Multithreading und richtet sich nach den Grundsätzen der generischen Programmierung. Hierzu gehört auch die ausgeprägte Verwendung von *Templates*. Hierdurch werden Methoden aber auch Klassen zur Verfügung gestellt, die mit verschiedenen Datentypen arbeiten, ohne dass sie für jeden einzelnen Datentyp neu implementiert werden müssen. Ein weiteres Systemkonzept von ITK ist die Verwendung von *Smart-Pointern* (dt. intelligenten Zeigern). Diese gewährleisten eine automatische Speicherverwaltung von Objekten, ähnlich dem *Garbage Collector* in Java. Dieser Mechanismus hält die Anzahl von Referenzen auf ein Objekt und gibt den belegten Speicherplatz automatisch frei, wenn kein Zeiger mehr auf das Objekt weist.

[2] VTK ist unter `http://www.vtk.org` verfügbar.
[3] `http://www.kitware.com`
[4] ITK ist unter `http://www.itk.org` verfügbar.

### 2.2.4. Boost Graph Library

Die Boost Graph Library (BGL) ist eine von vielen frei zugänglichen C++ Bibliotheken, die von der Onlinegemeinschaft Boost unterstützt wird. Die Boost-Gemeinschaft entscheidet ob vorgeschlagene Bibliotheken den Standards entsprechen. Hierbei liegt der Schwerpunkt auf portablen und qualitativ hochwertige Bibliotheken, die mit der C++ Standard Bibliothek zusammenarbeiten, und von diesen dann aufgenommen werden.

Die BGL ist eine solche standardisierte Klassenbibliothek, die durch generische Programmierung die Datenstruktur und Algorithmen der Graphentheorie unterstützt. Sie stellt umfangreiche Schnittstellen für Graphenstrukturen zur Verfügung. Durch die Verwendung von Templates wird eine vom Datentyp unabhängige Programmierung ermöglicht. Währenddessen bleibt die detaillierte Implementierung verborgen. Die BGL ist eine *offene Schnittstelle* insoweit, dass jede Graph-Bibliothek, die diese Schnittstelle implementiert, interoperabel ist [32]. Sie definiert eine Sammlung von Konzepten, welche die Darstellung und Manipulation von Graphen realisieren.

# 3. Stand der Technik

In diesem Kapitel wird eine Übersicht über computergestütze Therapieplanungssysteme und deren wesentlichen Entwicklungen, sowie über den aktuellen Forschungsstand, gegeben. Zudem werden Toolkits, die sich mit der Thematik von tubulären Strukturen auseinandersetzen, vorgestellt und näher betrachtet.

Ein ausführlicher Vergleich von Visualisierungsverfahren für Gefäße wird in Abschnitt 5.1.1 vorgenommen. Diese getrennte Betrachtung dient dem besseren Verständnis der Problematik und ist in die thematischen Auftrennung der Arbeit eingegliedert.

## 3.1. Computergestützte Therapieplanungssysteme

Es gibt nur wenige Forschungsgruppen, die sich mit der Erstellung und Verbesserung von computergestützten Therapieplanungssystemen für die Leberchirugie auseinandersetzen. Beispielsweise sei die Forschungsgruppe der MeVis Medical Solutions AG zu nennen, die sich insbesondere mit der Leberoperationsplanung beschäftigt. In Zusammenarbeit mit der Universität in Magdeburg sind verschiedene Veröffentlichungen und wissenschaftliche Arbeiten, die sich mit den Interaktionstechniken ([17]) und der Visualisierung ([23], [29], [21]) von Gefäßen ([16], [31]) beschäftigen, erschienen. Da es sich bei der Leberoperationsplanung um eine kommerzielle Dienstleistung handelt, die mit internen Produkten realisiert wird, sind die beschriebenen Interaktionskomponenten nicht überprüfbar und können somit nicht direkt mit dieser Arbeit verglichen werden. Jedoch können die beschriebenen Interaktionstechniken als Konzeptvorlagen für eigene Interaktionskomponenten verwendet werden. Hierzu gehören [17]:

Visuelle Analysetechniken

- Selektion von Strukturen
- Multiple 3D-Ansichten: Zusätzliches Übersichtsfenster
- Hervorhebungstechniken oder Abgeschwächte Darstellung
- Annotationen zur Therapieplanung
- Beseitigung von geometrischen Diskontinuität nach der Segmentierung und Skelettierung

Quantitative Analysetechniken

- Darstellung von Radiendiagramme
- Ausgabe quantitativer Parameter
- Markierung der Verzweigungstypen (z.B. Bifurkation, Trifurkation)

Im Bereich der Visualisierung erarbeitet der Lehrstuhl für Visualisierung der Universität in Magdeburg bereits seit einige Jahren neue Verfahren. Ein Vergleich dieser und weiteren aktuellen Visualisierungstechniken wird in Abschnitt 5.1.1 skizziert.

Zudem beschäftigt sich eine Forschungsgruppe des IRCAD in Straßburg mit computergestützen Operationsplanungen ([6], [5] [34]). Diese Gruppe befasst sich in den letzten Jahren aber schwerpunktmäßig mit Augmented Reality Systemen zur intraoperativen Unterstützung, so dass keine aktuellen Veröffentlichung im Bereich von Leberoperationsplanungen vorhanden sind.

Die Forschungsgruppe der Abteilung Medizinische und Biologische Informatik (MBI) des Deutschen Krebsforschungszentrums in Heidelberg beschäftigt sich seit einigen Jahren ebenfalls mit der computergestützten Leberoperationsplanung ([8], [25], [26]). Zudem wurde ein Therapieplanungssystem für die navigierte Bronchoskopie entwickelt ([9], [40], [39]).

Da die Umsetzung der vorliegenden Arbeit in das etablierte MITK der Abteilung MBI einfließen soll, wird eine detaillierte Ist-Analyse der vorhanden Therapieplanungssystemen vorgenommen, mit besonderen Blick auf die Verarbeitung der tubulären Systeme.

### Leberoperationsplanung

Innerhalb der Abteilung MBI wird die Leberoperationsplanung in Teilschritte zerlegt und durch unterschiedliche Module abgedeckt. Diese Teilung der Arbeitsschritte ist in Abbildung 3.1 dargestellt. Die Trennung begünstigt die Optimierung und Verbesserung der einzelnen Module und unterstützt den allgemeinen Ansatz, die Module für unterschiedliche Anwendungsfälle zu verwenden.

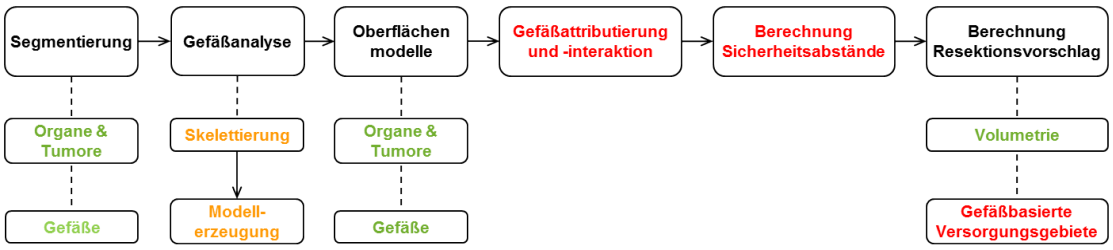


Abbildung 3.1.: Verarbeitungsschritte der Leberoperationsplanung

Im Bereich der Organ- und Tumorsegmentierung werden dem Benutzer vielfältige halb- und vollautomatische Segmentierungstechniken zur Verfügung gestellt. Aber auch manuelle Tools die gegebenenfalls zur Verbesserung des Ergebnisses verwendet werden, sind hiermit abgedeckt.

Für die Gefäßsegmentierung stehen dem Benutzer mehrere Möglichkeiten zur Verfügung: Ein erweitertes Bereichswachstumsverfahren ([12]) oder ein modellbasiertes Ver-

fahren ([37]).

Im Bereich der Gefäßanalyse steht eine einfache Datenstruktur aus früheren Projekten zur Verfügung. Hierbei werden die Gefäße in einer Baumstruktur verwaltet, die jedoch Einschränkungen unterliegt. So können zum Beispiel Zyklen im Gefäßsystem mit dieser Datenstruktur nicht abgebildet werden (siehe Abschnitt 4.1.1).

Weiterhin existiert ein Skelettierungsverfahren, das aus der Segmentierung die Informationen für die Datenstruktur extrahiert. Da hier jedoch, neben den topologischen Informationen, nur Informationen über einen kreisrunden Gefäßquerschnitt weitergeleitet werden, wird von der eigentlichen morphologischen Beschaffenheit der Gefäße abstrahiert ([25]). Dies hat wiederum weitere Einschränkungen des Datenmodells zur Folge.

Auf Grundlage der vorhandenen Datenstruktur werden Oberflächen im dreidimensionalen Raum erzeugt. Die Visualisierung erfolgt mittels einfachen Primitiven (siehe Abschnitt 5.1.1). Diese Visualisierung entspricht der genauen Topologie abstrahiert aber den Gefäßquerschnitt. Da aber vor allem Ersteres bei der Leberoperationsplanung wichtig ist und der genaue Gefäßquerschnitt vernachlässigt werden kann, ist dieses Verfahren hier ausreichend.

Ein weiterer wichtiger Schritt bei der computergestützten Planung ist die Interaktion mit den Gefäßen. Durch den Benutzer können somit weitere wichtige Informationen in das Modell eingebracht werden. Vor allem die Zuordnung der Gefäße zu den Lebersegmenten, aber auch die visuelle, farbliche Unterscheidung der Gefäßtypen ist hierbei dienlich. Über eine solche Attributierung kann auch der Resektionsstatus den einzelnen Gefäßen zugeordnet werden. Eine solche Interaktionskomponente gab es in früheren Softwareprodukten der Abteilung die speziell für die Leberoperationsplanung entwickelt wurden (z.B. LENA ∼2000, OrgaNicer ∼2002, ReLiver ∼2006). Eine Realisierung dieser Interaktionsmöglichkeiten in MITK ist jedoch nicht vorhanden.

Anhand der generierten dreidimensionalen Modelle kann der Arzt nun Resektionsvorschläge planen. Ein nützliches Tool, das in der Vorarbeit zu dieser Diplomarbeit realisiert wurde, ist die deformierbare Ebene. Diese sogenannte Schnittebene kann der Benutzer anhand von anatomischen Gegebenheiten sowie der Lage des Tumors positionieren. Anschließend werden Resektionsvolumen und Leberrestvolumen berechnet und dem Arzt angezeigt. Eine gefäßbasierte Planung, d.h. anhand von berechneten Versorgungsgebieten, ist nicht vorhanden.

### Navigierte Bronchoskopie

Das Ziel der navigierten Bronchoskopie ist es dem Arzt in Echtzeit über Position und Richtung des distalen Ende des Bronchoskops zu informieren [40]. Diese Hilfestellung für den Arzt bei Untersuchungen oder Eingriffen in der Lunge wird wie in Abbildung 3.2 dargestellt.

Die Position des Bronchoskops innerhalb des Volumens wird mittels elektromagnetischem Tracking bestimmt. Diese Information wird mit einem dreidimensionalen Modell der Lunge in Zusammenhang gestellt und angezeigt. Eine besondere Herausforderung stellt dabei die Atembewegung des Patienten dar.

Die Navigation des Arztes durch den Bronchialbaum ist eine der Hauptaufgaben des

Tools. Hierfür wird in der virtuellen Realität ein vordefinierter Pfad dargestellt, an welchem sich der Mediziner orientieren kann. Weicht er von diesem vordefinierten Pfad ab, wird er informiert und an geeigneter Stelle zurückgeführt.

Für die Registrierung mit dem Bilddatensatz muss ein geeignetes Modell des Brochialbaumes erzeugt werden. Dabei kommen die selben Segmentierungs- und Skelettierungsalgorithmen wie bei den Lebergefäßen zum Einsatz.

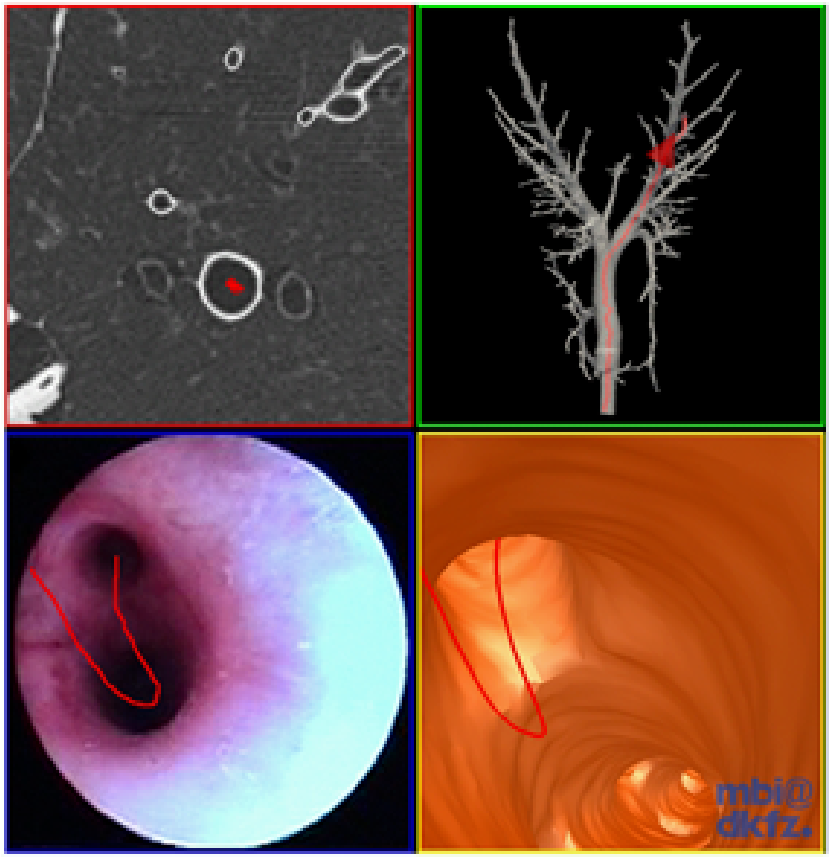


Abbildung 3.2.: Navigierte Bronchoskopie in MITK: 2D Schichtbild an der Position des Instruments (oben links); 3D Ansicht der gesamten Bronchialbaum-Topologie mit transparenter Bronchusoberfläche, rotem Leitpfad und rotem Konus als Repräsentant der Instrumentenspitze (oben rechts); Videobild des Bronchoskops überdeckt von dem virtuellen Pfad zum Zielbronchus (unten links); virtuelle Bronchoskopie an der Stelle des Instruments mit Pfad zum Zielbronchus (unten rechts) (Quelle: [9])

Das Datenmodell wurde nachträglich für diesen Verwendungszweck adaptiert und ist hierfür nur ungenügend. Grund für diese Adaption waren vor allem die Möglichkeit der Exploration der Datenstruktur. Um den Pfad für die Navigation zu planen wird der kürzeste Weg von der Trachea zu dem Zielvolumen gesucht. Ein großer Nachteil dabei

ist, dass sich die Visualisierung des extrahierten Datenmodells nicht für die Navigation innerhalb der Strukturen eignet, da diese Fragmente im Inneren der Strukturen erzeugt. Somit muss auf Grundlage des Segmentierungsergebnisses zusätzlich eine polygone Oberfläche erzeugt werden [39]. Ebenso ist eine spezifischen Attributierung der anatomischen und funktionellen Strukturen der Lunge und des Bronchialbaums nicht möglich. Diese Problematik spiegelt sich in Abbildung 3.3, die den Ablauf der Planung einer solchen navigierten Bronchoskopie abbildet, wider.

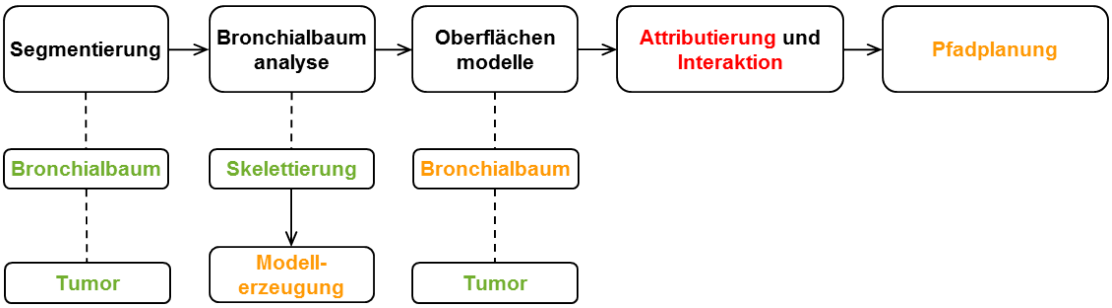


Abbildung 3.3.: Verarbeitungsschritte der navigierte Bronchoskopie

### Fazit

Die hier analysierten Anwendungsfälle zeigen die momentanen Schwächen des MITKs in Bezug auf den Umgang mit tubulären Strukturen, insbesondere deren Repräsentation und die interaktiven Verwendung.

Das Datenmodell, basierend auf einer Baumstruktur, ist für die allgemeine Verwendung nicht mehr ausreichend, da Sonderfälle, wie zum Beispiel durch Anastomosen entstehende Zyklen, nicht abgebildet werden können. Zudem können nur Gefäße mit einem kreisrunden Querschnitt repräsentiert werden. Darüber hinaus werden anatomische und funktionelle Informationen, die nur speziell für die Lebergefäße relevant sind, abgespeichert. Für die allgemeine Verwendung, wie zum Beispiel die Repäsentation von Bronchialbäumen sind andere Informationen relevant und daher nicht abbildbar.

Das aktuelle Verfahren zur Visualisierung von Gefäßen bildet Fragmente im Inneren. Dadurch kann diese Visualisierung nicht für die navigierte Bronchoskopie verwendet werden, da hier eine Navigation durch das Gefäßinnere benötigt wird.

Eine Interaktion mit tubulären Strukturen ist gegenwärtig nicht möglich. In der Vergangenheit konnten zusätzlich Informationen vom Anwender interaktiv zum Datenmodell hinzugefügt werden. Diese Informationen waren allerdings rein auf die Leber fokussiert. Die Explorationsmöglichkeiten auf der alten Datenstruktur sind im Umgang mit tubulären Strukturen im Allgemeinen sinnvoll, sollten hierbei aber auch auf eine allgemeinere Datenstruktur adaptiert werden.

Die genannten Lücken und Einschränkungen sollen mit dieser Arbeit und in zukünftigen Projekten behoben werden. Hierfür werden im folgenden Abschnitt Toolkits, mit

vergleichbaren Ansätzen, näher betrachtet.

## 3.2. Toolkits

Im Bereich von externen Toolkits die mit tubulären Strukturen und Blutgefäßen arbeiten gibt es das TubeTK und Vascular Modeling Toolkit. Diese zwei Toolkits werden hier kurz vorgestellt.

### 3.2.1. TubeTK

Das TubeTK[1] ist speziell für die Bereitstellung von Algorithmen für tubuläre Strukturen von Kitware Inc. entwickelt worden. Neben der Segmentierung von solchen tubulären Strukturen steht vor allem die Registrierung im Vordergrund.

Das TubeTk ist von der Idee, ein Toolkit für tubuläre Strukturen und damit verbunden Funktionalitäten anzubieten, sehr interessant für diese Arbeit. Jedoch beschränken sich die Algorithmen auf die Organregistrierung, sowie der Extraktion von tubulären Strukturen. Darüber hinaus ist auf Grund des bisher erst kurzen Entwicklungszeitraumes wenig über den aktuellen Stand bekannt. Die Ziele, die dieses Toolkit verfolgt, sind vor allem für neurologischen Forschungsgebiete sehr interessant und sollte weiter verfolgt werden.

### 3.2.2. Vascular Modeling Toolkit

Das Vascular Modeling Toolkit (VMTK)[2] ist eine Sammlung von Bibilotheken und Tools zur bildbasierte Modellierungen von Blutgefäßen. Das Toolkit beinhaltet verschiedene Skripte die auf Python Klassen basieren. Diese Skripte interagieren über das integrierte PypeS Framework. Die eigentlichen Algorithmen sind in C++ programmiert und basieren auf VTK und ITK Klassen. Das Toolkit ist plattformunabhängig und frei verfügbar.

Neben einer gradientenbasierten Level-Set Segmentierung der Gefäße, kann der Benutzer interaktiv eine Segmentierung einzelner Äste vornehmen, bei der abgehende Äste nicht berücksichtigt werden. Die Mittellinien können berechnet und grafisch dargestellt werden. Zudem bietet es vielfältige Analysewerkzeuge die mittellinienbasierte, sowie oberflächenbasierte geometrische Größen berechnet. Darunter zum Beispiel Winkelabmessung an Bifurkationen, Entfernungen zur Mittellinie und Oberflächenkrümmung. Neben einer DICOM-Dateneinbindung kann es verschiedene gängige Datentypen von Bildern, Oberflächen und Meshes bearbeiten. In einem bereitgestellten Viewer können diese dann rotiert, verschoben, skalieren und das *volume of interest* verändert werden.

Das VMTK ist in die erweiterbare Anwendung 3D-Slicer eingebunden und bietet hier die Möglichkeiten einer interaktiven Level-Set-Segmentierung. Zudem können die Mittelachsen der Gefäße berechnet werden.

[1] VMTK ist unter `http://public.kitware.com/Wiki/TubeTK`
[2] VMTK ist unter `http://www.vmtk.org/` verfügbar.

Der Schwerpunkt der Funktionalitäten des VMTK liegen auf der Segmentierung und der Mittellinienberechnung. Zudem können mit den vielfältige geometrische Größen auf Grundlage von Oberflächen und der Mittellinie berechnet werden. Für die Ziele der Arbeit, also der Konzepierung eines geeigneten Datenmodell, sowie hierauf basierende Interaktionskomponente, bietet das Toolkit allerdings keine interessanten Ansätze.

### 3.2.3. Übersicht Toolkits

In der nachfolgenden Tabelle 3.1 soll eine kurze Übersicht verschafft werden, welche Bereiche im Umgang mit tubulären Strukturen, von den Toolkits abgedeckt werden.

| **Verfahren** | **TubeTK** | **VMTK** | **MITK** |
|---|---|---|---|
| Vorverarbeitung | + | + | + |
| Segmentierung | ? | + | ++ |
| Skelettierung | + | ++ | + |
| Datenmodell | / | / | - |
| Visualisierung | ? | + | + |
| Registrierung | + | / | / |
| Interaktion | / | / | / |

Tabelle 3.1.: Vergleich der Toolkits: Qualitative Beurteilung der Toolkits anhand von Dokumentation, Veröffentlichungen und eigener exemplarischer Implementierung. Hierbei bedeutet *++*, dass das Verfahren besonders gut und *+* gut abgedeckt ist. Ist ein Verfahren vorhanden, aber bietet große Nachteile so wird das äquivalent mit  und  gekennzeichnet. Wenn ein solches Verfahren nicht realisiert ist, wird das mit */* verdeutlicht. Bei einem *?* kann aufgrund der vorhandenen Dokumentation keine Aussage getroffen werden.

# 4. Datenmodell

## 4.1. Grundlagen

In diesem Abschnitt wird der Begriff der symbolischen Beschreibung eingeführt und die Vorteile, die sich für die Repräsentation der tubulären Strukturen ergeben, werden näher erläutert. Des Weiteren wird auf die Frage, warum sich der Graph als Grundlage der Datenstruktur von tubulären Strukturen besonders eignet, eingegangen. Hierfür werden die Grundzüge der verwendeten Graphentheorie erläutert.

### 4.1.1. Repräsentation der tubulären Strukturen

Eine abstrakte Repräsentation mittels Objekten und den Verbindungen zwischen diesen wird als *symbolische Beschreibung* oder auch „Signal to Symbol“ bezeichnet. Im Falle der tubulären Strukturen wird mit Hilfe dieser Beschreibung die Verzweigungsstruktur dargestellt. Sie sollte den Verlauf der Strukturen, die Lage von Verzweigungs- und Endpunkten, sowie weitere Informationen über Durchmesser und Form abbilden können.

Durch die Anatomie von Gefäßen und des Bronchialbaumes liegt es nahe, dass ein Graph oder Baum als Datenstruktur am Besten geeignet ist. Da der Baum eine Spezialform des Graphen darstellt, bringt er gewisse Einschränkungen in das Modell. Nur ein zusammenhängender Graph ohne Zyklen erfüllt die Anforderungen an einen Baum.

Bei Gefäßen kann es aber zu Anastomosen kommen, die Zyklen erzeugen. Zudem kann es bei der Segmentierung der Strukturen zu lückenhaften und somit nicht verbundenen Teilstücken kommen. Deshalb ist der Graph als allgemeinere Darstellungsart von tubulären Strukturen zu wählen.

Die Endpunkte des Graphen stellen hierbei die Gefäßwurzel und die Spitzen der gefäßartigen Strukturen dar. Alle anderen Knoten und Kanten spiegeln die Verzweigungsstruktur wider. Durch die Verwendung von gerichteten oder attributierten Graphen kann man Durchmesser, sowie funktionelle und weitere anatomische Informationen in das Modell mit einbringen.

### 4.1.2. Erzeugung der Symbolischen Beschreibung

Die Erzeugung einer symbolischen Beschreibung für tubuläre Strukturen in der medizinischen Bildverarbeitung ist ein mehrstufiger Prozess. Daher ist das Ziel dieser Arbeit die Realisierung eines Verfahrens, das auf möglichst viele Objekte anwendbar ist. Hierfür sollen allgemeinere, modularisierte Algorithmen verwendet werden. Um dies zu erreichen, wird der in Abbildung 4.1 dargestellte Arbeitsablauf verwendet.

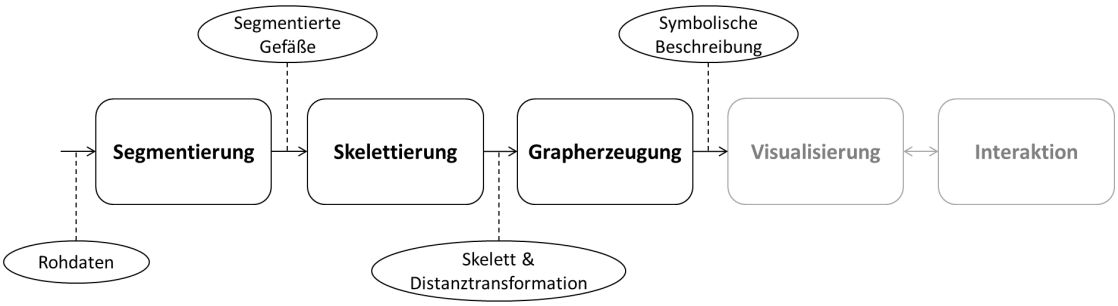


Abbildung 4.1.: Arbeitsablauf zur Erzeugung der symbolischen Beschreibung

Der Arbeitsablauf zur Generierung der symbolischen Repräsentation unterteilt sich in folgende Schritte:

**Segmentierung** Die Aufgabe der Segmentierung ist die Identifikation der Bildpunkte, die zu der betrachteten Struktur gehören. Es handelt sich hierbei um eine binäre Klassifizierung des Bilddatensatzes. Dies kann manuell durch den Benutzer, halbautomatisch oder vollautomatisch durch den Computer erfolgen. Bei der manuellen Segmentierung muss der Benutzer in jeder Bildschicht die Kontur des Objektes nachzeichnen. Zum einen ist dieser Prozess äußerst zeitintensiv. Zum anderen wird er durch die Verfolgung der vielen Verzweigungen, sowie den kleinen Strukturen bei Gefäßbäumen nahezu unmöglich. Dadurch erfolgt die Segmentierung mit Hilfe von halbautomatischen oder vollautomatischen Verfahren.

Neben der Unterteilung nach dem Grad der Benutzerinteraktion können die Segmentierungsverfahren anhand von verwendeten Merkmalskriterien oder nach verwendeten Techniken klassifiziert werden.

Die Verfahren werden in pixel-, kanten- und regionenorientierte Verfahren unterteilt. Zudem unterscheidet man zwischen modellbasierte und texturorientierte Verfahren. [18] [35]

- **Pixelorientierte Verfahren**
  Pixelorientierte (auch punkt- oder histogrammbasierte) Segmentierungsverfahren entscheiden für jeden Bildpunkt mittels eines global definierten Kriteriums des zugrunde liegenden Grauwerthistogramms, ob er der Segmentierung hinzugefügt wird oder nicht. Hierbei wird keine Rücksicht auf räumliche Beziehungen zwischen Bildpunkten genommen. Das bekannteste Beispiel eines solchen Verfahrens ist die Segmentierung mittels eines *Schwellwert* (engl. *Threshold*). Dabei werden alle Bildpunkte mit einem Grauwert über diesem Schwellwert der Segmentierung hinzugefügt und alle darunter liegenden bleiben unberücksichtigt.
- **Kantenorientierte Verfahren**
  Bei kantenorientierten Verfahren wird nicht jedes Pixel einzeln auf sein Homogenitätskriterium geprüft. Viel mehr werden Verhältnisse zwischen verschiedenen Pixelbereichen betrachtet, um Kanten innerhalb des Datensatzes zu

erkennen. Ein Segmentierungsalgorithmus, der dieses Vorgehen nutzt ist die *Wasserscheidentransformation* (engl. *Watershed Transformation*). Die Gradienten innerhalb des Bildes werden als Höhenprofil interpretiert und von den Talsohlen ansteigend bei Erfüllung des Homogenitätskriterium als Region der Segmentierung hinzugefügt. Höhenkämme zwischen zwei Talsohlen bilden sich dabei als Kanten heraus, bis auch ihr Höhenprofil das Merkmal erfüllen.

- **Regionenorientierte Verfahren**
  Entgegen den bereits vorgestellten Verfahren arbeiten regionenorientierte Verfahren nicht global auf dem gesamten Datensatz. Sie beginnen lokal an einem Bildpunkt, dem *Saatpunkt* (engl. *Seed Point*), und prüfen alle benachbarten Bildpunkten auf Ähnlichkeitsmerkmale. Erfüllen sie diese, werden sie der Segmentierung hinzugefügt und der Algorithmus setzt sich an den neu gewonnenen Bildpunkten iterativ fort. Im einfachsten Fall gibt der Benutzer ein Grauwertintervall für die Bestimmung an. Andere Verfahren verwenden einen vom Benutzer angegebenen mittleren Grauwert und Varianz. Das *Bereichswachstumsverfahren* (engl. *Region-Growing*) ist der bekannteste Vertreter dieser Kategorie.

- **Modellbasierte Verfahren**
  Die modellbasierte Verfahren gehören zu den wissensbasierten Methoden. Auf Grundlage eines zuvor definierten Modells, der Form eines Objektes, werden auf dem Bilddatensatz korrespondierende Regionen gesucht.
  Hierbei gibt es sogenannten *Formmodelle* (engl. *Shape Model*), die mittels vorher segmentierten, unterschiedlichen Datensätze erzeugt werden. Dadurch können Information der Varianz der Formen des Objektes entnommen werden. Bei den genannten Anwendungsfällen sind diese Shape Models vor allem bei der Organsegmentierung nützlich, jedoch für Gefäßsegmentierungen nicht anwendbar. In [37] wird ein modellbasierter Ansatz vorgestellt, der speziell für die Segmentierung von Gefäßen geeignet ist.

- **Texturorientiere Verfahren**
  Die Textur eines Objektes stellt seine regelmäßigen und somit seine charakteristische, flächenhafte Verteilung der Bildpunkte dar. Hierzu werden die Grauwerte als strukturelle Beschreibung verwendet, um zusammenhängende Bereiche des Objektes zu finden.

**Skelettierung** Der aus dem Segmentierungsverfahren entstandene Binärbilddatensatz repräsentiert die Menge der Voxel, die zur betrachteten Struktur gehören. Die Informationen, die in der Lage und Größe der Voxel-Haufen enthalten sind, dienen als Grundlage für weitere Analysen. Die volumenbasierte Darstellungsform ist dafür jedoch nicht unmittelbar geeignet (siehe Abschnitt 5.1.1). Deshalb wird eine Repräsentation gewählt, die mittels einer *Skelettierung* bzw. *Mittelachstransformation* erzeugt wird. Hierbei wird das dreidimensionale Objekt unter Erhaltung der Topologie und Geometrie auf eine im Idealfall eindimensionale Linie reduziert. Zudem wird eine Distanztransformation berechnet. Das bedeutet, dass jedem Objektpunkt

der Abstand zum nächsten Hintergrundpunkt zugeordnet wird. Dieser Wert kann als Näherung für den Durchmesser verstanden werden.
Das Verfahren *Topologisches Thinning* ist geeignet für eine Skelettierung solcher gefäßartiger Strukturen. Diese Methode basiert auf der iterativen Identifikation und Löschung von sogenannten *Simple-Points* von der Oberfläche des Objektes. Simple-Points sind Punkte, deren Entfernung die Topologie des Objektes nicht ändert. Das bedeutet, dass die Anzahl der Ecken, der Kanten, der Löcher sowie der Flächen gleich bleiben.[25]

**Analyse und Grapherzeugung** Aus der zuvor erzeugten Linienstruktur und der Distanztransformation wird die symbolische Beschreibung in Form eines Graphen abgeleitet. Dessen Topologie äquivalent zur Topologie des Skelettes ist, welches wiederum topologisch äquivalent zum originalen Objekt, der tubulären Struktur ist.
Diese Transformation in einen ungerichteten Graphen erfolgt auf Basis der vorab berechneten Mittelachsen, in dem zunächst gemäß der Nachbarschaftsbeziehungen der Voxel untereinander Knoten und Kanten im Graphen erzeugt werden. Auch nicht zusammenhängende Abschnitte werden berücksichtigt und vermerkt.[25]

Durch die Aufteilung in diese Einzelschritte und deren separate Betrachtung wird es möglich, sie als eigenständige Probleme zu betrachten und zu lösen. Somit können diese auch einzeln optimiert werden, um in der Folge das beste Endergebnis zu erzielen.
Die durch die Rekonstruktion in Abbildung 4.2 erhaltene symbolische Beschreibung der Anatomie von tubulären Strukturen ist eine wertvolle Grundlage für die computergestützten Therapie - und Operationsplanungen.

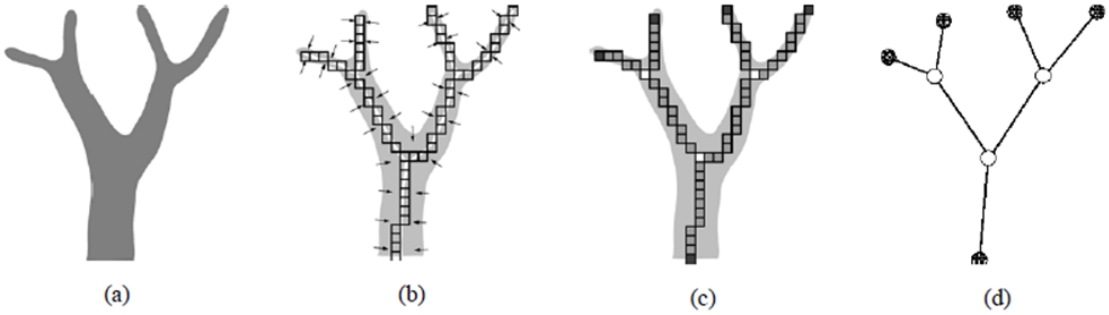


Abbildung 4.2.: Erzeugung der symbolischen Beschreibung: Umwandlung der Segmentierung (a) der tubulären Struktur mit Hilfe von Thinning-Techniken in ein Skelett (b). Aus den nachbearbeiteten Skelett werden Skelettvoxel an Verzweigungen und Endpunkten extrahiert (c) und in einen Graphen (d) überführt. (Quelle: [31])

### 4.1.3. Grundlagen der verwendeten Graphentheorie

Die folgenden Definitionen der Graphentheorie sind zum Verständnis der Datenstruktur grundlegend und notwendig. Die eingeführten Begriffe lehnen sich an [36] an.

- Ein *Graph* $G$ ist ein Paar $G = (V, E)$ aus einer endlichen Menge *Knoten* $V$ (engl. *Vertices*) und einer Menge *Kanten* $E$ (engl. *Edges*) von zweielementigen Teilmengen von V. $V(G)$ bezeichnet die Menge der in einem Graph enthaltenen Knoten und $E(G)$ die Menge der in einem Graph enthaltenen Kanten.

- Eine Kante $e$ heißt *inzident* zu den Knoten $a$ und $b$, wenn eine Kante $e = \{a, b\}$ oder $e = \{b, a\}$ existiert. Zwei Kanten heißen *benachbart* oder *adjazent*, wenn beide inzident zu einem gleichen Knoten sind. Zwei Knoten heißen benachbart oder adjazent, wenn eine Kante existiert, die diese verbindet.

- Der *Grad* $D$ eines Knotens $v$ beschreibt die Anzahl der zu einem Knoten inzidenten Kanten ($D(v)$).

- Ein *Kantenzug* ist eine Folge von Kanten $e_i = \{v_{i-1}, v_i\} \in E, i = 1, ..., n$. Sind die Knoten $v_i$ paarweise verschieden, so liegt ein *Weg* vor. Geschlossene Wege $v_0 = v_n$ sind *Kreise*. Die Knoten $v_0$ und $v_n$ heißen dabei *Anfangs-* und *Endpunkte*.

- Zwei Knoten $a$ und $b$ heißen *verbindbar*, wenn es einen Kantenzug mit Anfangspunkt $a$ und Endpunkt $b$ gibt. Wenn alle Knoten eines Graphen $G$ paarweise verbindbar sind nennt man $G$ *zusammenhängend*.

- Ein *gerichteter Graph* ist ein Paar $(V, E)$ disjunkter Mengen von Knoten und Kanten zusammen mit zwei Funktionen $init : E \to V$ und $ter : E \to V$, die jeder Kante $e \in E$ einen *Anfangsknoten* $v_1 = init(e)$ und einen *Endknoten* $v_2 = ter(e)$ zuordnen. Die Kante $e$ heißt dann von $v_1$ nach $v_2$ *gerichtet*. Man beachte, dass ein gerichteter Graph zwischen zwei Knoten $v_1$ und $v_2$ durchaus mehrere Kanten haben kann. Diese Kante nennt man *Mehrfachkanten*. Haben zwei Mehrfachkanten die gleiche Richtung, so nennt man sie *parallel*. Ist $v_1 = v_2$, so ist e eine *Schlinge*.

- Ein gerichteter Graph $D$ ist eine *Orientierung* eines ungerichteten Graphen $G$, wenn $V(D) = V(G)$ und $E(D) = E(G)$ ist und $\{init(e), ter(e)\} = \{x, y\}$. Ein solcher orientierter Graph entsteht also aus einem ungerichteten Graphen dadurch, dass jede Kante nachträglich gerichtet wird; entweder von x nach y oder von y nach x, aber nicht beides. Orientierte Graphen sind also gerichtete Graphen ohne Mehrfachkanten und Schlingen.

- Ein *Baum* ist ein gerichteter Graph $D$, der zusammenhängend und kreisfrei ist. Jeder Knoten, der nur eine Kante besitzt, heißt *Blatt*. Alle übrigen Knoten werden als *Verzweigungsknoten* bezeichnet. Unter einem *Teilbaum* $T$ von $D$ mit Wurzelkante $e_w = (w, w') \in E$ soll ein Baum $T = (V', E')$ verstanden werden, der w und alle von w' aus erreichbaren Punkte $V' \subseteq V$ enthält, sowie die verbindenden Kanten $E' = \{\{a, b\} \in E | a \in V' \text{ und } b \in E')\}$.

- Ein *gewichteter Graph* ist ein Graph, dem eine oder mehrere Funktionen $W_i(e)$ zugeordnet sind. Diese Funktionen berechnen für eine gegebene Kante e einen Skalarwert, welcher dem Gewicht der Kante entspricht.

- Ist bei $U(F, K)$ $F \subseteq V$ und $K \subseteq E$ so ist $U$ ein *Untergraph* oder *Teilgraph* von $G$. Bei einem induzierten Untergraph existieren alle Kanten von G, deren Knoten in $F$ liegen.

## 4.2. Anforderungen

In den bestehenden Vorarbeiten hat sich gezeigt, dass sich komplexe Gefäßsysteme am besten durch Graphen darstellen lassen [26]. Auf Grundlage der vier Abstraktionsebenen, die nach der klassischen Graphentheorie geordnet sind, soll ein Datenmodell entworfen werden, das für ein System zur Analyse von tubulären Strukturen geeignet ist. Die allgemeinste Form sind ungerichtete Graphen, die die Informationen über Topologie und Morphologie der segmentierten Strukturen abbilden. Werden zudem Annahmen über die Richtung des Blutflusses in das Modell eingebracht, so verwendet man gerichtete Graphen. Eine weitere Abstufung kann erreicht werden, wenn im medizinischen Kontext Anastomosen ignoriert oder ausgeschlossen werden können. Hierbei kann man Zyklen auflösen und azyklische Graphen erzeugen. Aus solchen azyklischen Graphen können wiederum ein oder mehrere hierarchische Bäume erzeugt werden. Wie in Abbildung 4.3 aufgezeigt, steigt bei jeder Abstraktionsstufe das eingebrachte Modellwissen. Die Generalität, sowie die algorithmische Flexibilität nehmen aber ab. Diese Eigenschaft dient der Systematik zur Strukturierung der Modelle und der darauf arbeitenden Algorithmen [26].

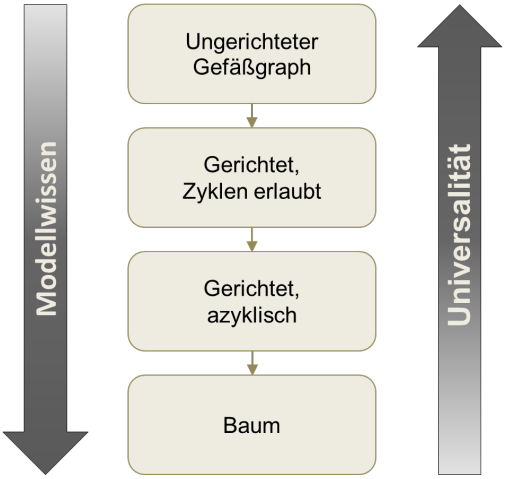


Abbildung 4.3.: Verarbeitung eines Graphen auf Abstraktionsebenen (Quelle: [26])

Auf Grundlage der Vorarbeiten und der dabei gewonnenen Erfahrungen in der Abteilung Medizinische und Biologische Informatik des DKFZ sind eigenständige Anforderungen definiert worden. Zum einen soll das Datenmodell eine allgemeine Repräsentation von tubulären Strukturen sein, die möglichst modular ist. Dies dient dem Einsatz in vielseitigen medizinischen Anwendungsfällen und der Wartbarkeit für die Zukunft. Bisher waren

die alten Baumstrukturen allein für die Leberoperationsplanung ausgelegt. Dadurch traten große Schwierigkeiten auf, um diese für die navigierte Bronchoskopie verwenden zu können. Innerhalb der Abteilung stehen nun neue Anwendungsfällen an, wie zum Beispiel im Bereich der neurologischen Bildverarbeitung. Zudem sollen für die zukünftigen Projekte eine problemlose Verwendung dieser neu entworfenen Datenstruktur möglich sein. Vor allem hier müssen deshalb geeignete Konzepte zur Wiederverwendung und Wartbarkeit entworfen werden.

Die ausführlichen Vorarbeiten der Segmentierungsverfahren sollten als Grundlage genutzt werden. Zudem sollen die aus dem Topologisches Thinning Verfahren gewonnen morphologischen Eigenschaften wie Position und Durchmesser in geeigneter Weise im Graphen abgebildet werden.

Ein sinnvoller Kompromiss zwischen schneller Traversierung, Modifizierbarkeit und Speicherplatzverbrauch soll bedacht werden. Auch die Trennung von topologischen Informationen und den visuellen Informationen sollen gewährleistet werden, da dies ein großer Nachteil bei der Pflege der alten Datenstruktur, war. Somit kann eine inhaltliche Konsistenz erreicht werden.

Um den Datenaustausch mit existierenden und zukünftigen Systemen zu gewährleisten, muss einerseits das bisherige Dateiformat weiterhin geladen, angezeigt und darauf interagiert werden können. Andererseits sollte der resultierende Graph in einem geeigneten standardisierten Format persistent abgespeichert werden können.
Zur besseren Übersicht sind in Tabelle 4.1 die aufgestellten Anforderungen kurz zusammengefasst.

- Datenstruktur auf Basis eines ungerichteten Graphen
- Trennung der topologischen und visuellen Informationen
- Schnelle Traversierung, einfache Modifikation und geringer Speicherplatzbedarf
- Konvertierung der alten Datenstruktur
- Persistente Speicherung des Datenmodells

Tabelle 4.1.: Zusammenfassung der aufgestellten Anforderungen an die Datenstruktur

## 4.3. Methoden und Realisierung

Zur Realisierung der Anforderungen ist die alte Datenstruktur innerhalb des MITK im Detail betrachtet worden. Allerdings ist die Implementierung, der einfachen Repräsentation, bereits veraltet und wurde aus alten Projekten nur übernommen. Zudem ist sie auf die Repräsentation einer Baumstruktur beschränkt. Vor allem aber weist sie bezüglich der Wart- und Erweiterbarkeit große Defizite auf. Deshalb wurde sich dafür entschieden eine komplett neue Datenstruktur aufzubauen und zu implementieren.

Diese neue Datenstruktur beruht auf einem ungerichteten Graphen, so dass nun Spezialfälle wie Zyklen zusätzlich abgedeckt werden können. Gleichzeitig ist die Konvertierung der alten Baumstrukturen in die neue Datenstruktur umgesetzt worden. Diese Konvertierung ermöglicht dem Benutzer die in dieser Arbeit entstandene Interaktionskomponente (siehe Kapitel 6) auch auf älteren Datensätzen zu verwenden. Die komplette Datenstruktur besteht aus Klassen, die selbständig für diese Arbeit implementiert wurden (siehe Abbildung 4.7) und beruhen auf Konzepten des C++ Standards und der Boost Graph Library (BGL). Die Verwendung ist ausdrücklich gekennzeichnet und bewusst eingesetzt um die Datenstruktur modular und wartbar zu halten.

### 4.3.1. Laufzeitrepräsentation

Die Entscheidung für eine Graphenstruktur als Repräsentation wurde bereits während der Analyse der Anforderungen für diese Arbeit gefällt. Nun sind im Rahmen der computergestützten Verarbeitung zwei Arten zur Repräsentation der Topologie eines Graphen gebräuchlich - die *Adjazenzmatrix* und die *Adjazenzliste*.

Um den Graphen $G(V, E)$ in einer Matrix abzubilden, benötigt man eine Matrix der Größe $|V| * |V|$. Die einzelnen Elemente der Matrix sind mit booleschen Werten belegt. Das Feld $a_{i,j}$ wird auf wahr gesetzt, wenn eine Kante $(i, j) \in E$ existiert. Bei ungerichteten Graphen wird lediglich die obere Dreiecksmatrix verwendet. Bei einem gerichteten Graphen wird die gesamte Matrix besetzt (siehe Abbildung 4.4). Der Vorteil von Adjazenzmatrix als Speicherstruktur ist, dass das Einfügen und Entfernen von Kanten eine konstante Zeit in Anspruch nimmt [36].

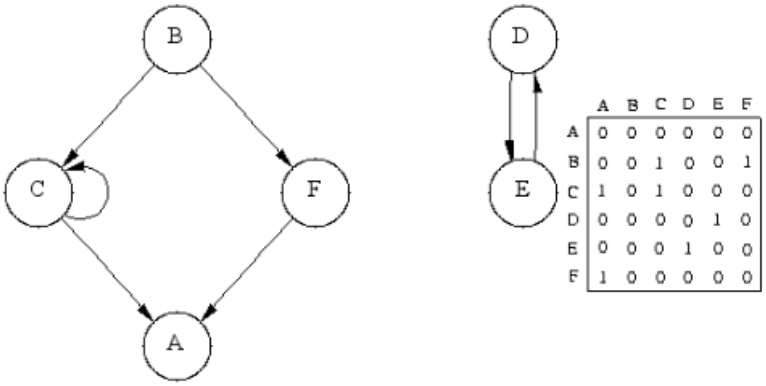


Abbildung 4.4.: Adjazenzmatrix eines gerichteten Graphen (Quelle: [32])

Die Adjazenzliste bietet die Möglichkeit einer dynamischen Datenstruktur. Bei dieser Art der Darstellung von Graphen werden die abgehenden Kanten von jedem Knoten in eine Liste gespeichert (siehe Abbildung 4.5). Somit ist die Basisstruktur die Liste aller Knoten und der maximale Speicherplatz liegt somit bei $O(|V| + |E|)$ [32].

Welche Darstellung am besten geeignet ist, hängt von der Komplexität des Graphen ab. Dieser Komplexitätsgrad $O(V, E)$ lässt sich durch $O(V, E) = |E| / |V|^2$ berechnen.

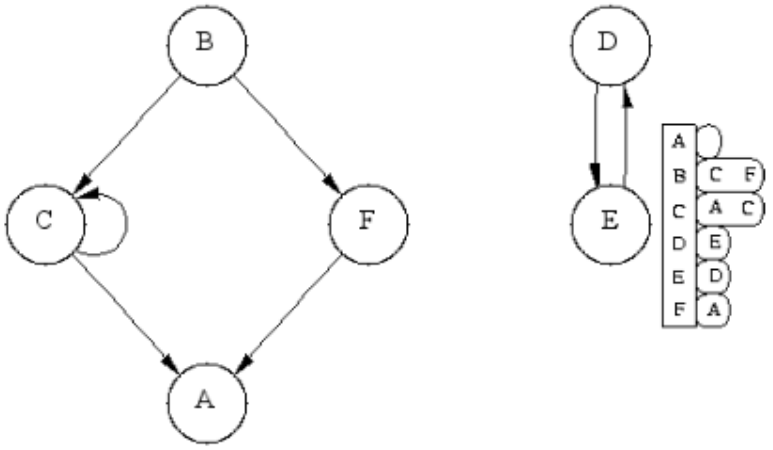


Abbildung 4.5.: Adjazenzliste eines gerichteten Graphen (Quelle: [32])

Ist $O(V, E) \approx 1$ , so ist der Graph sehr dicht, d.h. fast alle möglichen Kanten zwischen den Knoten sind vorhanden. Ein lichter Graph, also ein dünn besetzter, hat einen Komplexitätsgrad $O(V, E) \ll 1$. Daraus ergibt sich, dass Adjazenzlisten vor allem bei lichten Graphen besser geeignet sind, da hier nur der tatsächlich benötigte Speicher belegt wird und diese bei der Traversierung genauso geeignet wie die Matrizen sind. Da man bei Gefäßgraphen, sowie den Brochialbäumen im Mittel von ca. drei Kanten pro Vertex in einem ungerichteten Graph ausgehen kann, ist $O(V, E) = 3 * |V| / |V|^2 = 3/|V|$. Man kann also ab ca. sechs Knoten in einem Graphen diesen als licht bezeichnen. Die vorliegende Datensätze enthalten Graphen, die wiederum aus hunderten bzw. teilweise tausenden Knoten bestehen. Deshalb ist die Adjazenzliste als geeignete Repräsentation der Topologie von den Graphen gewählt worden.

Für die Implementierung des Graphen wird die BGL verwendet, welche neben den grundlegenden Datenstrukturen auch Algorithmen zur Graphenverarbeitung bereitstellt. Die Verwendung dieser externen Bibliothek bietet den Vorteil, dass durch die hohe Templatisierung ein vielseitiger Einsatz möglich ist und diese aktiv gepflegt wird (siehe Abschnitt 2.2.4).

Als Speicherstrukturen werden hier sowohl Adjazenzlisten als auch Adjazenzmatrizen angeboten. Die BGL spricht ihre Kanten und Knoten über sogenannte *Deskriptoren* an. Diese können je nach Graphentyp durch unterschiedliche Datentypen repräsentiert werden. Knoten-Deskriptoren können instanziiert, kopiert und verglichen werden. Das gleiche gilt für Kanten-Deskriptoren, nur dass diese noch zusätzlich auf Quell- und Zielknoten zugreifen können. Der Zugriff auf die Knoten und Kanten des Graphen wird über *Iteratoren* realisiert. Es gibt fünf Arten von Graph-Iteratoren: Iteratoren für Knoten und Kanten, Iteratoren für eingehende und ausgehende Kanten, sowie einen Iterator für benachbarte Knoten. Ein weiteres wichtiges Design der BGL sind die *Property Maps*. Diese

bieten die Möglichkeit den Kanten und Knoten eines Graphen bestimmte Eigenschaften zu assoziieren. Sie realisieren ein Konzept, bei dem ein Satz von Schlüsselobjekten einem Satz von Wertobjekten zugeordnet werden.[32]

Für die Realisierung im Rahmen dieser Arbeit wurde die Klasse `mitkUndirectedGraph` implementiert, die einen allgemeinen ungerichteten Graphen verwaltet. Hierbei erlaubt der Einsatz der objektorientierter C++-Template Mechanismen das Ablegen und Abfragen von beliebigen Informationen an Kanten und Knoten. Je nach Bedarf und Anwendungsfall können passende Datentypen verwendet werden. Die Adjazenzliste wird über die BGL beschrieben und mittels der Datentypenvereinbarungen als Graph definiert.

```
/**
* Creating the graph type
* listS:          Represents the OutEdgeList as a std::list
* vecS:           Represents the VertexList as a std::vector
* undirectedS:    Representation for an undirected graph
* VertexProperty: Defines that all vertex are represented by VertexType
* EdgeProperty:   Defines that all edges  are represented by EdgeType
*/
typedef boost::adjacency_list<
      boost::listS,
      boost::vecS,
      boost::undirectedS,
      boost::property<vertex_properties_t, mitk::VertexType>,
      boost::property<edge_properties_t, mitk::EdgeType>
    > GraphType;
```

Hierbei kann zwischen Listen und Vektoren als Speicherstrukturen für die Kanten und Knoten gewählt werden. Zudem wird der Typ des Graphen übergeben und für die Properties der Kanten und Knoten die Templateparameter gesetzt. Durch die Kapselung der BGL spezifischen Zugriffe und Definitionen stellt die Klasse alle Methoden für die Graphenverarbeitung so zur Verfügung, dass diese auch von Anwendern die keine oder nur wenige Kenntnisse von der BGL haben, verwendet werden können. Knoten und Kanten können über eigens implementierte Funktionen verwaltet werden.

Für die Laufzeitrepräsentation der Morphologie von tubulären Strukturen wird mittels einer konkreten Implementierung der `mitkUndirectedGraph` Klasse sichergestellt, dass für die Darstellung alle relevanten Informationen abgespeichert werden. Die zentral, entworfene Klasse `mitkTubeGraph` mit den Klassen `mitkTubeGraphVertex` und `mitkTubeGraph Edge` als Templateparameter stellt dies sicher. Eine Übersicht der entstanden Klassen und deren Beziehungen zueinander ist in Abbildung 4.7 dargestellt.

Um den Gefäßverlauf in dem ungerichteten Graphen abbilden zu können, sind zwei Möglichkeiten vorhanden: Zum einen kann der gesamten Verlauf der Gefäße allein über Knoten dargestellt werden, wobei Kanten leere Verbindungen sind. Zum anderen können die Knoten nur an Verzweigungsstellen gesetzt werden. Die Information über den Verlauf

der einzelnen Gefäße wird dann in den Kanten abgelegt. Bei der ersten Möglichkeit würde die Anzahl der Knoten und Kanten extrem ansteigen. Dies hätte zur Folge, dass der Rechenaufwand bei einer Exploration des Graphen ansteigen würde, da der Zeitaufwand für Tiefen- und Breitensuche bei $O(|V| + |E|)$ liegt. Zudem betrachtet man ein Gefäß in den meisten Fällen als Ganzes. Die zweite Möglichkeit spiegelt also bei der Graphentraversierung die eigentliche Traversierung der Gefäße wider. Daher ist die Realisierung auf der zweiten Möglichkeit konzipiert und aufgebaut. Somit wird festgelegt, dass die Knoten als Verzweigungspunkte dienen und die Kanten den Gefäßverlauf repräsentieren (siehe Abbildung 4.6).

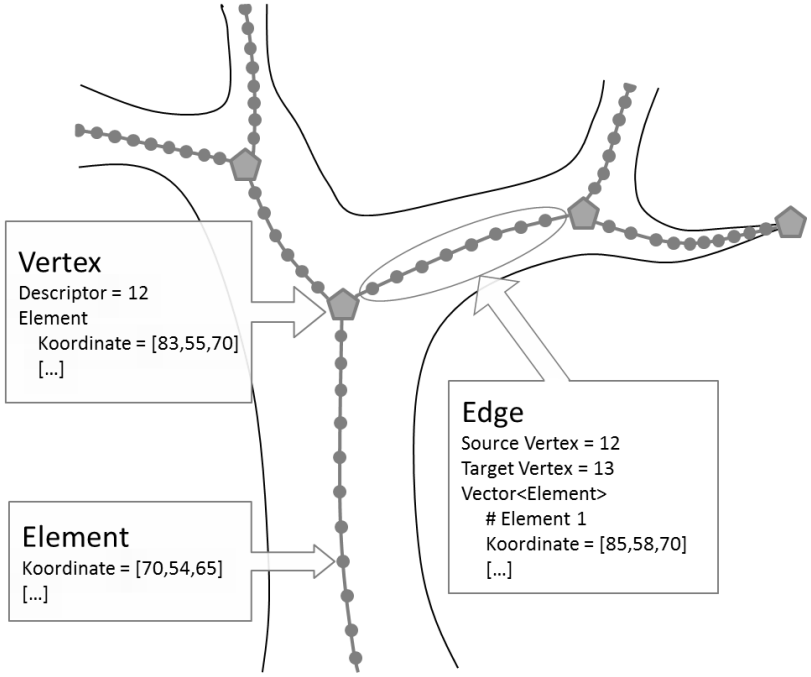


Abbildung 4.6.: Bildhafte Darstellung der Datenstruktur zur Laufzeitrepräsentation

Hierfür verwaltet die neue Klasse `mitkTubeGraphVertex` als Knotenobjekt und somit als Verzweigungs- und Endpunkt eines Gefäßes, nur ein Element. Die Kantenklasse `mitkTubeGraphEdge` hingegen speichert den Gefäßverlauf und hält hierfür einen Vektor von Elementen. Diese Elemente sind in der Klasse `mitkTubeElement` abgebildet und repräsentieren die Informationen von Gefäßabschnitten. Diese ist als rein virtuelle Basisklasse implementiert, welche eine Koordinate als zwingend vorhandenes Attribut verlangt. So kann in einer abgeleiteten Klasse weitere Informationen wie zum Beispiel der Gefäßradius abgelegt werden. Diese Umsetzung wurde bewusst gewählt um möglichst variabel tubuläre Strukturen und ihre Besonderheiten in den Anwendungsfällen (z.B. Aneurysmen) abbilden zu können.

Da das Ergebnis des momentanen verwendeten Skelettierungsverfahren einen kreisrunden Gefäßquerschnitt für die Graphenerzeugung liefert, wurde hierfür beispielhaft eine

abgeleitete Klasse implementiert, die zusätzlich zur Koordinate, den Radius abspeichert.

### 4.3.2. Weitere Informationen

Die Loslösung von den visuellen Informationen von der eigentlichen Datenstruktur ist durch Anhängen einer speziellen Property Klasse an das Datenobjekt realisiert. Die visuellen Informationen, wie zum Beispiel die Farbe oder die Sichtbarkeit der einzelnen Strukturen, werden über die interaktive Attributierungen bestimmt. Deshalb werden sie zusammen mit der resultierenden Eigenschaftsklasse `mitkTubeGraphProperty` im Abschnitt 6.3.3 ausführlich beschrieben.

### 4.3.3. Konvertierung

Bei der Umsetzung der Konvertierung der alten Datenstruktur in die Neue, werden nur die Informationen über den Verlauf und den Gefäßdurchmesser übernommen. Die funktionellen und anatomischen Einordnungen können mittels der neuen Interaktionskomponente vollständig abgebildet und zudem neue Aspekte berücksichtigt werden (siehe Kapitel 6).

Für die Konvertierung wurde ein neuer *Filter* `mitkTubeGraphConverter` implementiert. Ein Filter, im Allgemeinen, bekommt ein Datenobjekt auf dem ein Algorithmus angewendet wird. Zurück wird das veränderte oder ein neues Datenobjekt gegeben. Der Filter `mitkTubeGraphConverter` hat als Eingangsparameter zur Konvertierung einen Baum der alten Struktur und als Rückgabewert ein Objekt der Klasse `mitkTubeGraph`. Über eine Rekursion werden alle Kanten und Knoten des alten Baums abgelaufen und die darin enthalten Information über Lage und Durchmesser der Gefäßelemente in den Graph abgebildet. Hierbei ist es von Vorteil, dass bei beiden Datenmodellen ein einzelnes Gefäß durch seine beiden Verzweigungs- bzw. Endknoten und der Kante mit den Verlaufsinformationen innerhalb eines Vektors abgespeichert werden (siehe Abbildung 4.6). So müssen keine zusätzlichen Kanten und Knoten berechnet werden.

### 4.3.4. Persistenz

Um die erzeugten Graphen über die Dauer eines Programmlaufes hinweg abzulegen und wiederherstellen zu können, ist eine persistente Speicherung notwendig. Hierfür wurde sich für die Auszeichnungssprache Extensible Markup Language (XML) entschieden. Diese ist ein vom WorldWideWeb Konsortium herausgegebene Spezifikation[1], die für den plattform- und implementationsunabhängigen Austausch von Daten, die der Darstellung hierarchisch strukturierter, textueller Information dient, eingesetzt wird.

Die XML-Datei gliedert sich in mehreren Hauptabschnitte, beginnend mit dem Header, der allgemeine Informationen zum Graphen beinhaltet. Nachfolgend werden alle im Graphen existierenden Knoten und alle Kanten mit ihren Attributen gespeichert. Zudem werden die visuellen Informationen der Property Klasse und die funktionellen, sowie anatomischen Gruppierungen der Strukturen abgelegt(siehe Abschnitt 6.3.3). Ein Auszug aus einer XML-Datei ist in Anhang A.1 dargestellt.

[1] `http://www.w3.org/XML/`

Das Erzeugen und das Einlesen einer solchen XML-Datei ist in eigens für dieses neu entstandene Dateiformat implementierten Reader und Writer Klassen umgesetzt worden.

### 4.3.5. Klassenübersicht

Um eine bessere Übersicht der implementierten Klassen zur Darstellung der tubulären Strukturen zu bekommen, ist in Abbildung 4.7 ein vereinfachtes Klassendiagramm abgebildet. Hierbei sind nur Klassen berücksichtigt, die die symbolischen Beschreibung von tubulären Strukturen bilden. Realisierte Klassen zur Speicherung und Konvertierung werden zur Übersichtlichkeit nicht mit einbezogen. In der Folge werden die dargestellten Klassen, zum besseren Verständnis des Diagramms, näher beschrieben:

**mitkUndirectedGraph** Klasse zur Verwaltung eines ungerichteten Graphen. Durch die Templatisierung und der Verwendung des Property-Konzeptes der BGL können den Kanten und Knoten beliebige Klassen zugewiesen werden.

**mitkTubeGraph** Repräsentation von tubulären Strukturen in einem ungerichteten Graphen. Die Knoten werden durch Objekte der Klasse `mitkTubeGraphVertex` und die Kanten durch Objekte der Klasse `mitkTubeGraphEdge` vertreten. Die Gefäße werden über einen `TubeDescriptor` angesprochen. Dieser setzt sich aus den Deskriptoren des Anfangs- und Endknoten einer Kante zusammen. Die Klasse bietet zudem die Möglichkeit Wege zwischen solchen Tubes zu finden. Außerdem kann eine Tube als Wurzel definiert werden. Somit kann dem ungerichtet Graph eine Orientierung gegeben werden, um einen gerichteten Graphen zu erhalten. Das macht eine Exploration in die so definierte Periphery von einer Starttube aus möglich. Des weiteren ist es möglich Teilgraphen zu extrahieren oder zu löschen.

**mitkTubeGraphVertex** Ein Objekt dieser Klasse stellt in einem tubulären System einen Verzweigungs- oder Endpunkt dar. Sie hält ein Objekt der Klasse `mitkTubeElement`.

**mitkTubeGraphEdge** Diese Klasse repräsentiert den Verlauf eines Gefäßes, in dem es die Informationen hierüber in einem Vektor mit Objekten der Klasse `mitkTubeElemet` speichert. Es können beliebig viele Elemente gespeichert werden. Zudem werden Funktionen bereitgestellt, die den gemittelten Durchmesser eines Gefäß, also mit Start-und Endknoten, sowie die Länge eines Gefäßes berechnen.

**mitkTubeElement** Abstrakte Klasse die Informationen über einzelne Gefäßabschnitte bereit stellt. Über virtuell definierte Funktionen wird sicher gestellt, dass jedes Element mindestens mit einer Weltkoordinate abgebildet wird. Mit konkreten Implementierungen dieser Klasse können unterschiedliche Gefäßquerschnitte repräsentiert werden.

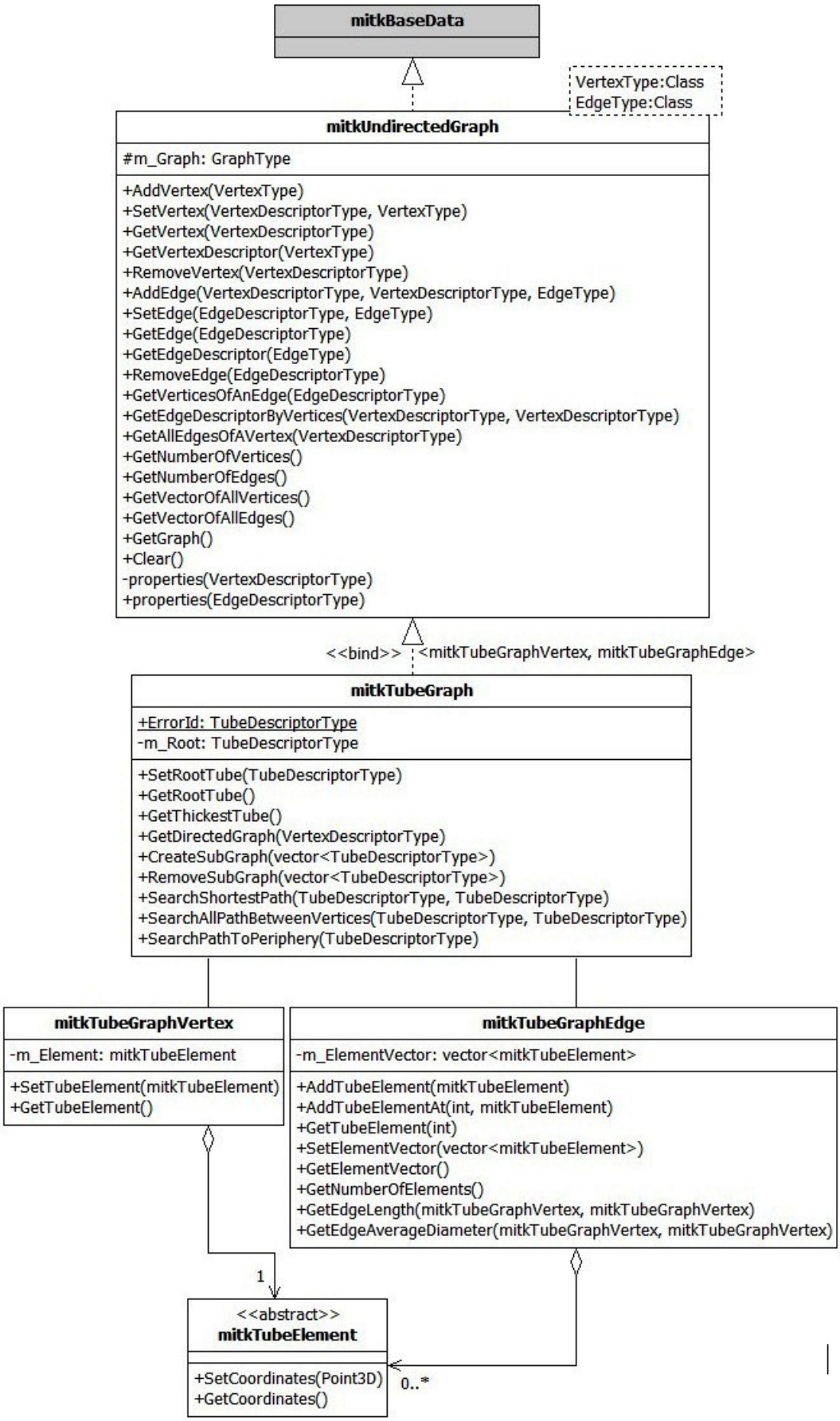




Abbildung 4.7.: Auszug aus dem Klassendiagramm der Datenstruktur

## 4.4. Ergebnisse

Es wurde eine modularisierte Datenstruktur entworfen, die tubuläre Strukturen mittels ungerichteter Graphen repräsentieren. Dabei stand die Wiederverwendbarkeit, aber auch die Wartbarkeit im Vordergrund. Der Einsatz der externen Softwarebibliothek Boost Graph Library unterstützt dies, da sie neben der grundlegenden Datenstruktur für Graphen auch viele templatisierte Algorithmen zur Graphenverarbeitung bietet. Zudem wird sie aktiv gepflegt und kontinuierlich weiterentwickelt.

Mit der Entscheidung den ungerichteten Graphen mit einer Adjazenzliste zu repräsentieren, wird der Speicherplatz gering gehalten. Zudem ist, bei der Verwaltung der vorliegenden, lichten Graphen in doppelt verketteten Listen, eine schnelle Traversierung möglich.

Im Vergleich zur bisher verwendeten, auf Baumstrukturen basierenden, Datenstruktur für Gefäßsysteme in MITK bietet die neue Realisierung die Möglichkeiten, alle Besonderheiten der Gefäßdiagnostik abzubilden. Zyklen, die durch Bildung von Anastomosen im Gefäßsystem entstehen können, können abgebildet werden, ebenso wie Aneurysmen oder andere Abweichungen des kreisrunden Querschnittes idealisierter Gefäße. Letzteres wird durch die konsequente Umsetzung von modular gehaltenen Klassen ermöglicht. So erlauben unterschiedliche konkrete Implementierungen der abstrakten Klasse `mitkTubeElement` die möglichst exakte Beschreibung von Gefäßen. Durch die Wahl verschiedenartiger geometrischer Formen für die konkrete Implementierung können einzelne Gefäßabschnitte individuell modelliert werden.

Des Weiteren ist auch der zugrunde liegende Graph in einer templatetisierten Klasse realisiert worden. Somit kann diese als Datenbasis auch für andere Projekte, die keine tubulären Systeme abbilden, eingesetzt werden. Diese grundlegende Graphenklasse bietet durch die bereitgestellten Klasse eine einfache und konsistente Modifikation. Es können hiermit neue Strukturen, wie zum Beispiel neue Verbindungen hinzugefügt werden. Aber auch die Daten zu bestehende Kanten und Knoten können durch die interne Property Verwaltung konsistent verändert werden. Zudem ist das Löschen logisch angepasst, so dass das Löschen eines Knoten auch die bestehenden, abgehenden Kanten entfernt.

Weiterhin erfolgt mit dem neuen Datenmodell eine strikte Trennung der visuellen Informationen und der Datenstruktur. Die Informationen über die Lage und die Form der Strukturen, sowie die Verbindungen untereinander sind in der Datenstruktur durch die Kanten und Knoten und deren enthaltenen Elemente gespeichert. Die visuellen Informationen und die Zuordnungen zu anatomischen und funktionellen Gruppen sind, im Gegensatz zur alten Struktur, hiervon getrennt und werden in der Property-Klasse verwaltet. Diese Informationen über Sichtbarkeit, Transparenz sowie die Einfärbung sind von der Interaktionskomponente abhängig. Durch das Anhängen der Property-Klasse an die Datenstruktur ist gewährleistet, dass für zukünftige Entwicklungen von Interaktionsmöglichkeiten durch Erweitern dieser Eigenschaftsklasse oder einfaches Austauschen, die Datenstruktur nicht betroffen ist. Diese Trennung folgt damit der logischen Konsistenz.

Die Anforderung der Konvertierung alter Datenstrukturen in die Neue wurden ebenfalls erfüllt. Die alte Struktur wird mittels eines neu geschriebenen Filters rekursiv abgearbeitet. Hierbei werden die Äste der Baumstruktur und die zugehörigen topologischen und morphologischen Eigenschaften in Knoten und Kanten des neuen Graphen abgebildet. Die Information über die anatomische und funktionelle Zuordnungen der einzelnen Gefäße können durch die neue Interaktionskomponente vollständig nachgebildet werden.

Alle Informationen der neuen Datenstruktur und der zugehörigen Eigenschaftsklasse werden mittels der Auszeichnungssprache XML in einem neuen Datenformat persistent gespeichert. Die zugehörigen Lese- und Schreibmechanismen sind in eigens für dieses Format implementierten Klassen realisiert.

# 5. Visualisierung

## 5.1. Grundlagen und Stand der Technik

Um die Anforderungen für eine geeignete visuelle Darstellungsform von tubulären Strukturen zu definieren, müssen zunächst die gängigsten Visualisierungsverfahren näher beleuchtet werden. Hierzu werden diese anhand von verschiedenen Kriterien bewertet und verglichen.

Zudem werden die Konzepte von VTK bezüglich der Visualisierung beschrieben, da diese die Grundlage des Rendering Prozesses in der Entwicklungsumgebung MITK bilden. Ebenso werden in diesem Abschnitt wichtige Klassen und Begriffe für die Visualisierung eingeführt.

### 5.1.1. Visualisierungsverfahren

In diesem Abschnitt wird ein Überblick über den gegenwärtigen Stand der Gefäßvisualisierungen gegeben. Viele der etablierten Verfahren zur Visualisierung baumartiger, tubulärer Strukturen stützen auf der Annahme, dass Gefäße einen kreisförmigen ([1], [13], [11], [21]) oder ellipsoiden ([24]) Querschnitt haben. Diese Annahme ist jedoch eine idealisierte Form der Wirklichkeit. Insbesondere wurde diese allein auf Blutgefäße aufgestellt und fand ausschließlich dort Anwendung. Nun werden die Visualisierungsverfahren aber auch für andere tubuläre Strukturen eingesetzt, wie zum Beispiel bei der Darstellung von Bronchialbäumen. Allerdings weicht hier der Querschnitt der Trachea und Hauptbronchien stark von der kreisrunden Form ab.

Weiterhin können die Querschnitte von Blutgefäßen von der Kreisförmigkeit abweichen. Die Annahme wurde ursprünglich für nicht pathologische Gefäße aufgestellt, wobei es auch bei gesunden Gefäßen durch den äußeren Druck, auch *extravasaler* Druck genannt, zu Verformungen kommen kann. Vor allem die Venen sind wegen ihrer schwächer ausgebauten Muskelschicht, dünnwandiger und haben als Teil des Niederdrucksystems auch einen geringeren inneren (*intravasaler*) Druck. Die Differenz zwischen intra- und extravasalen Druck wird als *transmuraler* Druck bezeichnet. Fällt dieser unter 10 mm Hg, kann es zu Verformungen kommen, was zu einem ellipsoiden Querschnitt führt. Fällt der Druck gegen Null, kommt es zur Kollabierung des Gefäßes und der Querschnitt nimmt die Form einer „8“ an. [27]

Vor allem bei krankhaften Veränderungen im Gefäßsystem ist eine möglichst exakte Darstellung des Segmentierungsergebnisses wichtig. Aneurysmen, also Ausbuchtungen in der Gefäßwand, können sehr unterschiedliche Formen aufweisen und können somit nicht nur mittels Skelett und Radieninformation beschrieben werden. Daneben sind Stenosen eine weit verbreitetere krankhafte Veränderung von Gefäßen. Grund für solche Verengung

der Gefäße ist häufig Plaque, also Ablagerungen in Form von Cholesterin, Kalk, Blutfetten, Bindegewebe und Blutgerinnsel. Diese Ablagerung ist nicht gleichförmig im gesamten Umfang des Gefäßes, daher ist ein kreisförmiger Querschnitt nicht mehr gegeben.

Ein wichtiges Ziel für die Therapieplanung ist eine genaue Darstellung von tubulären Strukturen, wie Gefäße und Bronchialbäume. Zudem wird für die computergestützte Planung ein dreidimensionales Darstellungsverfahren für die Topologie der Struktur benötigt. Für Lagebeziehung einzelner Strukturen muss der Benutzer die Strukturen von allen Seiten sehen können.

### Direktes Volumen Rendering

Das *Direct Volume Rendering* ist eine Möglichkeit zur dreidimensionalen Darstellung eines tomographischen Datensatzes. Hierbei werden die Werte des Datensatzes mit Hilfe von Transferfunktionen auf Farb- und Transparenzwerte abgebildet und mittels Projektion auf die Bildebene direkt dargestellt.

**Maximum Intensity Projection** Die *Maximum Intensity Projection* (dt. Maximumintensitätsprojektion) ist ein verbreitetes Verfahren, bei dem für jedes Pixel, entlang eines Abtaststrahls, das Voxel mit dem höchsten Datenwert ermittelt und dargestellt wird. Auf diese Weise kann es zu einer fehlerhaften Wiedergabe der Tiefeninformation kommen, da kleine Gefäße oder kontrastarme Gefäße fälschlicherweise als durch größere Gefäße verdeckt, dargestellt werden. Bei der *Minimum Intensity Projection* werden anstelle der maximalen Intensitäten die jeweils minimalen ausgewählt. Dieses Verfahren wird zum Beispiel in der Darstellung von Computertomographie-Aufnahmen der Lunge verwendet um die signalarmen Bronchien besser abbilden zu können.[23]

**Closest Vessel Projection** Als Weiterentwicklung von Maximum Intensity Projection löst das *Closest Vessel Projection* die genannten Probleme. Bei diesem Verfahren wird das Voxel dargestellt, das vom Betrachter ausgesehen entlang des Sehstrahls als erstes einen bestimmten Schwellwert überschreitet. Somit können auch kleinere Gefäße, vor größeren Gefäßen, korrekt abgebildet werden. Jedoch ist dieses Verfahren stark abhängig von dem gewählten Schwellwert, so dass keine Gefäße unterdrückt werden, aber auch keine irrelevanten Strukturen mit angezeigt werden.[23]

Die Verfahren des Volume Rendering sind jedoch für die Visualisierung tubulärer Strukturen nur bedingt geeignet, da durch technisch bedingte Probleme, wie zum Beispiel Bildrauschen und Partialvolumeneffekte, die Nutzbarkeit eingeschränkt wird. Auch durch die begrenzte Auflösung kommt es vor allem bei kleineren Strukturen zu Aliasing-Artefakten. Außerdem können mit diesen Verfahren auch andere Strukturen, die in einen ähnlichen Intensitätswerte-Bereich fallen, fälschlicherweise dargestellt werden.

### Modellbasierte Visualisierung

Im Gegensatz zu dem direkten Volumen Rendering verwenden die modellbasierten Verfahren symbolische Beschreibungen als Grundlage der Visualisierung. Diese werden, wie in Abschnitt 4.1.2 beschrieben, mittels Segmentierung und Skelettierung erstellt. Somit liegen Informationen über Länge und Durchmesser, sowie weitere Attribute in der Graphenstruktur vor.

**Einfache Primitive** Ist vor allem die Topologie der Gefäßbäume bei einem Anwendungsfall wichtig, wie zum Beispiel bei der Leberresektion, wird für die Gefäße ein kreisförmiger Querschnitt approximiert. Unter dieser Annahme liegt die Darstellung mittels einfacheren Primitiven mit kreisförmigem Querschnitt nahe. Die früheren Verfahren ([1], [13]) nutzen die Zylinderform zur Visualisierung der Kanten. Die Radien der Zylinder entsprechen hierbei den lokalen Radien der zugehörigen Gefäßabschnitte. Diese Art der Darstellung ist sehr effizient, jedoch lässt sich der Radienverlauf entlang einer Kante mit diesem Visualisierungsverfahren nicht abbilden. Des Weiteren kommt es zu sichtbaren Übergängen an benachbarten Zylindern (siehe Abbildung 5.1).

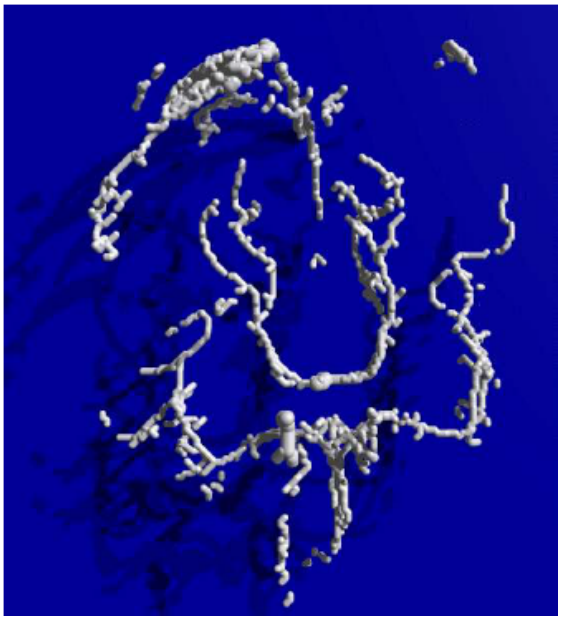

Abbildung 5.1.: Visualisierung mit einfachen Primitiven: Die Darstellung mittels Zylinder erlaubt die Beurteilung der Gefäßtopologie. (Quelle: [13])

Um den Durchmesserverlauf besser darzustellen und die Diskontinuität an den Segmenten zu verringern, werden in neueren Verfahren Kegelstümpfe eingesetzt.[16] Um die Qualität der Darstellung zusätzlich zu erhöhen, werden die Gefäßenden mittels Halbkugeln verschlossen.

Beide Verfahren stellen nur die Topologie dar und nicht die exakte Geometrie der Gefäßbäume, da sie auf der Annahme von kreisförmigen Querschnitten basieren. Des Weiteren kommt es bei Verzweigungen zu überstehenden Strukturen im Inneren der Oberfläche.

**Freiformflächen** Die Erzeugung einer möglichst glatten Gefäßoberfläche ist das Ziel des von *Ehricke et al.* vorgestellten Visualisierungsverfahrens [10]. Hierbei liegen keine Beschränkungen im Hinblick auf den Gefäßquerschnitt vor. Somit können die verwendeten Freiformflächen beliebige Formen besitzen. Die Extraktion des Gefäßmodells resultiert hierbei aus einem durch Spline-Kurven beschriebenen Skeletts. Die Querschnittsinformation wird mittels Voxeln beschrieben, die senkrecht zum Gefäßverlauf den Rand des segmentierten Volumens bilden. Somit soll eine korrekte Abbildung von Aneurysmen und Stenosen möglich sein.

Für eine geometrische Kontinuität muss vor allem bei den Verzweigungen auf das Zusammenfügen der Freiformflächen geachtet werden. Abbildung 5.2(a) bildet das Zusammenfügen an einer solchen Verzweigung schematisch ab. Allerdings werden von den Entwicklern keine genaueren Angaben über die Funktionsweise gemacht. Der einzig aufgeführte Nachweis ist in der Abbildung 5.2(b) zu sehen. Darüber hinaus ist das Verfahren für sehr kleine Äste nicht anwendbar. Angaben über die Rechenkomplexität der Rekonstruktion sind in [10] nicht getroffen worden.

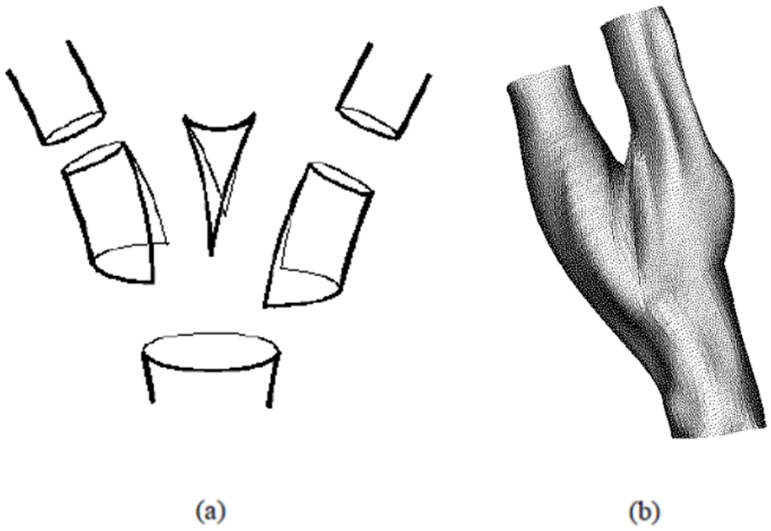


Abbildung 5.2.: Visualisierung mit Freiformflächen: Zusammenfügen von Freiformflächen an einer Verzweigung (a) und Gefäßvisualisierung (b). (Quelle: [10])

**Simplex Meshes** *Simplex Meshes* wurden in [7] als deformierbare Modelle vorgestellt. Sie sind topologisch äquivalent zu Dreiecksnetzen. Ein Punkt in diesem Netz entspricht einem Dreieck im Dreiecksnetz. Jeder Punkt besitzt genau drei Nach-

barpunkte. Kanten entsprechen Verbindungen benachbarter Punkte. Diese Dualität zwischen Simplex Mesh und einem Dreiecksnetz wird in Abbildung 5.3 dargestellt.

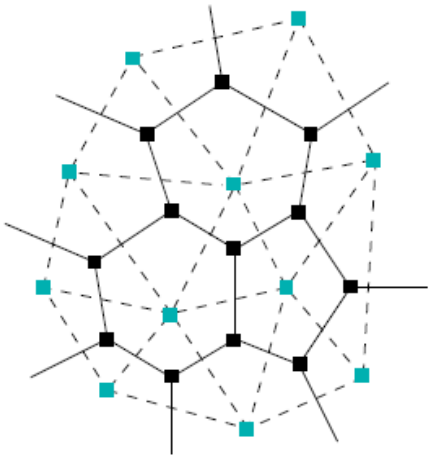

Abbildung 5.3.: Dualität zwischen Simplex Mesh (durchgezogene Linie) und Dreiecksnetz (gestrichelte Linie) (Quelle: [20])

Die Deformation des Simplex Meshes basiert auf einer internen Kraft $F_{int}$ und einer externen Kraft $F_{ext}$. Die interne Kraft sichert die Erhaltung der geometrischen Form des Simplex Meshes. Die externe Kraft passt das Simplex Mesh an das Objekt an.

Diese Verformungseigenschaften wurden von *Bornik et al.* für die Gefäßvisualisierung genutzt [3]. Von einem initialen, zylinderförmigen Simplex Mesh ausgehend, wird dieses durch die aufeinander folgenden Querschnitte erweitert. Die dabei entstehenden Verbindungen müssen aber die Eigenschaften des Simplex Meshes genügen. Deshalb werden zu diesem Zweck zusätzliche Punkte, entlang der neu erzeugten Kante, eingefügt. Nach der Erzeugung des Simplex Meshes für den gesamten Gefäßbaum wird die Darstellungsqualität mit der oben genannten Deformation verbessert.

Wie in der Abbildung 5.4 zu sehen ist, ist das Ergebnis eine geschlossene, glatte Oberfläche bei der Strukturen im Inneren vermieden werden. Jedoch wird in [3] nicht weiter auf die Berechnung der Kräfte $F_{int}$ und $F_{ext}$ eingegangen. Zudem wird keine Genauigkeitsanalyse durchgeführt. Insgesamt ist die Komplexität der erzeugten Netze mit dem Resultat des Marching Cubes-Algorithmus vergleichbar.

**Subdivision Surfaces** Das von *Ed Catmull* und *Jim Clark* entwickelte Konzept der *Subdivision Surfaces* [4] wurde erstmals 1978 vorgestellt. Grundlage des Verfahrens ist ein grobes, polygonales Initialnetz, das mittels mehrerer Iterationen zu beliebig feinen Oberflächennetze aufgeteilt werden kann. Felkel et al. nutzte dies um

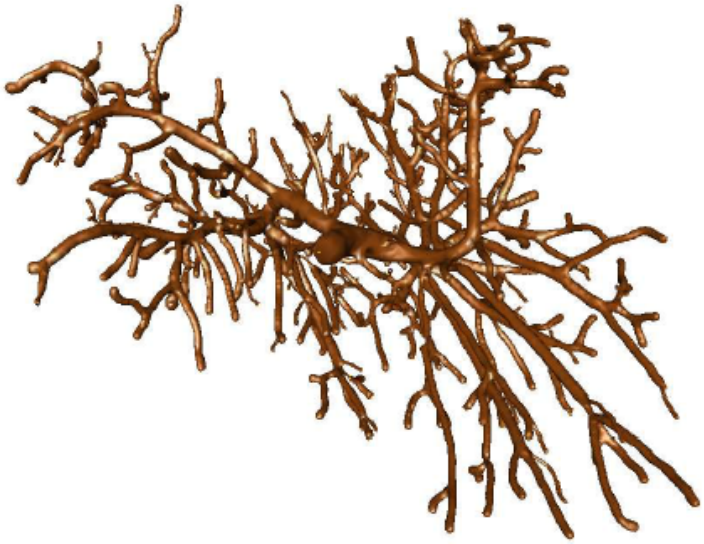

Abbildung 5.4.: Visualisierung einer menschlichen Pfortader durch Simplex Mesh (Quelle: [3])

auf der Basis eines Gefäßmodells eine glatte Oberfläche zu rekonstruieren [11]. Hierbei wird angenommen, dass der Gefäßquerschnitt kreisförmig ist. Diese Querschnitte werden für die Erzeugung des Initialnetzes durch Quadrate approximiert, welche die jeweiligen Kreise vollständig umschließen. Die aufeinander folgenden Quadrate werden so miteinander verbunden, dass eine Verdrehung des Polygonnetzes verhindert wird. Für jeden der so entstandenen Quader wird geprüft, ob ein Seitenast abgeht. Ist dies der Fall, wird der abgehende Gefäßabschnitt mit der entsprechenden Fläche verbunden (siehe Abbildung 5.5). Das so entstandene Basismodell wird mittels eines Subdivision Surfaces-Algorithmus verfeinert und geglättet.

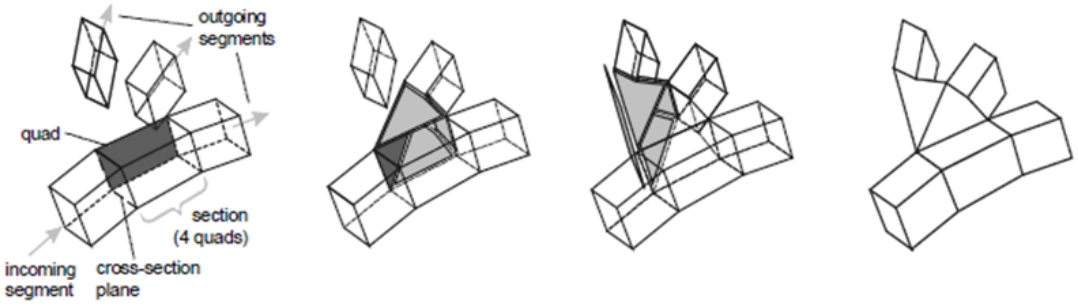


Abbildung 5.5.: Erkennung und Konstruktion einer Trifurkation bei der Visualisierung mittels Sub Division Surfaces. (Quelle: [11])

Die Darstellungsqualität hängt von der gewählten Anzahl der Iterationen ab. Da aber ein Viereck je Iteration durch vier Vierecke ersetzt wird, vervierfacht sich die Komplexität mit jeder Stufe. Die Vorteile des Verfahrens liegen in der schnellen Erzeugung des Oberflächennetzes und der Vermeidung von Geometrien im Inneren der Strukturen. Die Verzweigungen werden organisch und weich dargestellt (siehe Abbildung 5.6). Allerdings wird auch hier von kreisrunden Gefäßen ausgegangen. Zudem kann es zu Abweichungen zwischen den ursprünglichen Zentren der Querschnitte und den Zentren der geglätteten Querschnitte kommen. Damit wird der Gefäßverlauf ebenfalls geglättet bzw. verfälscht. Eingesetzt werden, kann dieses Verfahren vor allem bei Anwendungsfällen bei denen die Genauigkeit nicht im Vordergrund steht, sondern die Topologie und eine skalierbare, hochqualitative Darstellung.

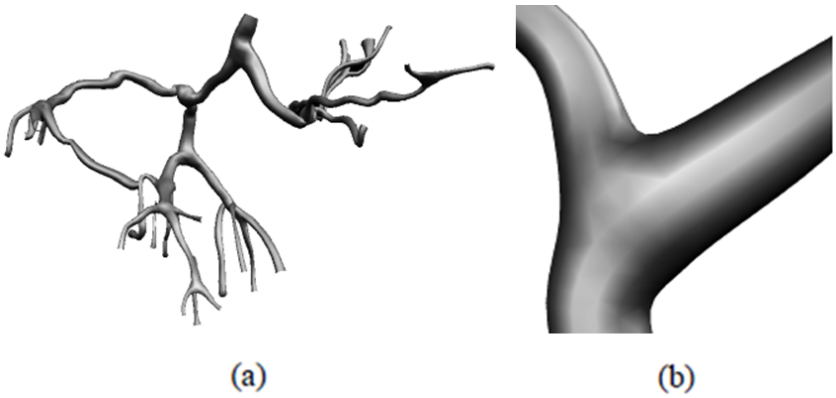


Abbildung 5.6.: Visualisierung mit Sub Division Surfaces: Visualisierung eines Lebergefäßbaumes (a) und eine vergrößerte Darstellung einer Bifurkation (b). (Quelle: [11])

**Convolution Surfaces** *Convolution Surfaces* stellen implizite Beschreibung der Oberfläche dar, die speziell der skelettbasierten Modellierung dienen [2]. Das Verfahren repräsentiert die Oberfläche von komplexen Objekten auf Basis von Skeletten, die durch Liniensegmente, Kurven oder Polygonen beschrieben sein können. Durch die Faltung des Skeletts mit einem Filter wird eine implizite Funktion erzeugt, aus der durch Polygonalisierung eine Oberfläche extrahiert werden kann. Diese Oberfläche hat ein glattes und organisches Erscheinungsbild. Zusätzlich werden benachbarte Objekte automatisch durch weiche Übergänge miteinander verbunden.

Auf Grundlage dieses Verfahrens beschreiben *Oeltze et al.* in [21] ein Verfahren, das Blutgefäße qualitativ hochwertig visualisiert. Hierbei wird der Gefäßbaum durch Centerline-Datenstrukturen beschrieben, die durch Punkte und den dazugehörigen Radieninformationen definiert sind. Hieraus entsteht eine glatte und organisch wirkende Oberfläche (siehe Abbildung 5.7).

Durch die Beschreibung mittels Liniensegmenten können hier nur kreisförmige

Querschnitte dargestellt werden. Zudem kommt es durch zu nah beieinanderliegenden Liniensegmenten an den Verzweigungsstellen zu einer Aufblähung der Gefäßoberfläche – sogenanntem Bulging. Zwar kann dies durch einen weiteren Filter geglättet werden [21], erhöht aber den, ohnehin schon großen, Rechenaufwand.

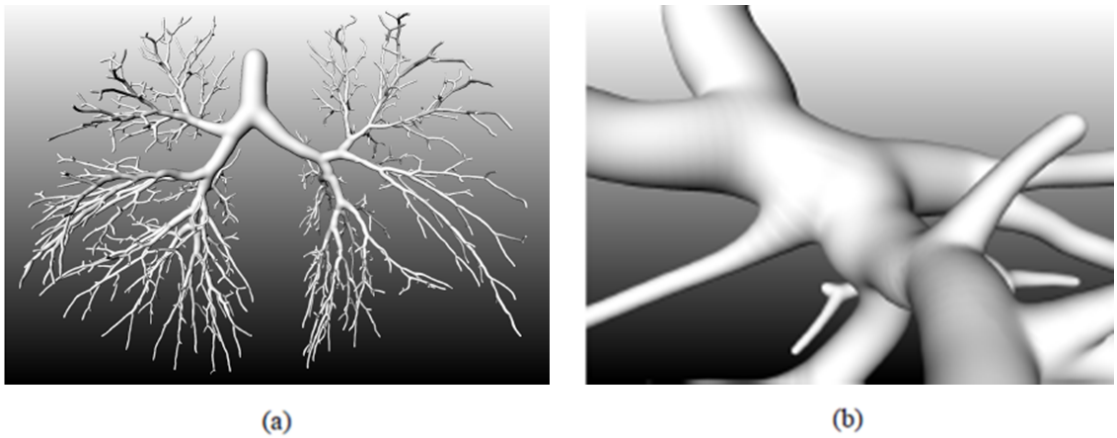


Abbildung 5.7.: Visualisierung mit Convolution Surfaces: Visualisierung eines Bronchialbaumes (a) und eine vergrößerte Darstellung (b) (Quelle: [21])

Die Vorteile der modellbasierten Verfahren liegen in den durch die symbolische Beschreibung geschaffenen analytischen und explorativen Möglichkeiten. Hierdurch sind einfache Berechnungen, wie Pfadlänge und Verzweigungswinkel, sowie das Einfärben und Ausblenden bestimmter Strukturen möglich. Jedoch wird bei den meisten Verfahren von einem kreisförmigen Querschnitt ausgegangen, was zu einer wenig organisch wirkenden Darstellung führt. Auch die entstehenden Diskontinuitäten in der Schattierung und die unnatürliche Darstellung von Verzweigungen sind dabei störend. Visualisierungsverfahren auf Basis von Subdivision Surfaces und Convolution Surfaces versuchen diese Nachteile zu beseitigen.

Generell sind solche Verfahren vor allem für Anwendungsfälle geeignet, bei denen der Schwerpunkt auf die Topologie der Gefäße gelegt wird. Für die Gefäßdiagnostik sind sie aufgrund der Ungenauigkeiten jedoch ungeeignet.

### Voxelbasierte Visualisierung

Die voxelbasierten Visualisierungsverfahren sind im Vergleich zu den im vorherigen Abschnitt beschriebenen modellbasierten Visualisierungsverfahren genauer, da sie ohne jegliche Modellannahme direkt auf Basis der Volumendaten oder dem Segmentierungsergebnis die Oberfläche generieren. Diese Oberfläche wird auch als *Isooberfläche* bezeichnet und stellt eine Fläche dar, die räumlich benachbarte Punkte mit gleichen Werten (*Isowert*) miteinander verbindet.

Es gibt zwei Gruppen von Verfahren zur Extraktion einer Isooberfläche aus Volumendaten. In diesem Abschnitt wird zum einen auf die Gruppe, die explizite Oberflächenre-

präsentationen nutzt eingegangen. Zum anderen wird sich auf die Gruppe, die implizite Beschreibung verwendet, konzentriert.

**Marching Cubes** Der *Marching Cubes* Algorithmus ist das bekannteste voxelbasierte Verfahren. Es erzeugt ein polygonales Oberflächennetz, das eine Isooberfläche aus Volumendaten approximiert [19]. Das Volumen wird dabei in ein Voxelgitter zerlegt und aus den Mittelpunkten von acht benachbarten Voxel wird eine Zelle erstellt. Die Eckpunkte der Zelle sind somit den Datenwerten der zugehörigen Voxel zugeordnet. Für jede Kante einer Zelle muss nun ermittelt werden, ob der gewählte Isowert zwischen den Datenwerten der angrenzenden Voxel liegt. Ist dies der Fall so wird das Volumen an dieser Stelle von der Isooberfläche geschnitten. Der Schnittpunkt wird hierbei über lineare Interpolation festgelegt. Mit Hilfe der Schnittpunkte einer Zelle kann die Teiloberfläche erzeugt werden. Durch die Fortführung dieser Bestimmung über das gesamte Voxelgitter gelangt man zu einem polygonalen Oberflächennetz.

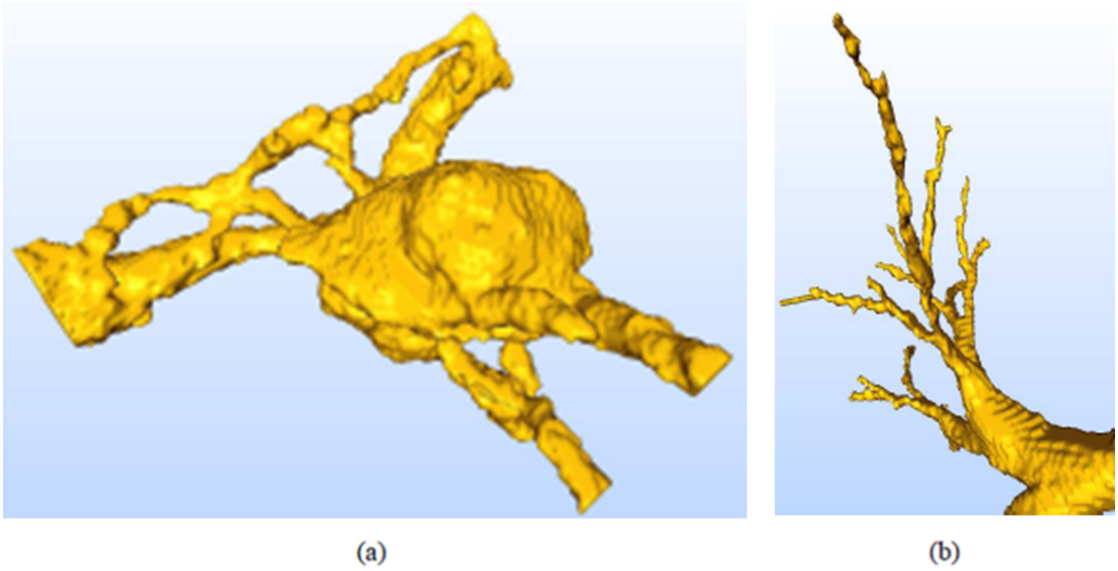


Abbildung 5.8.: Visualisierungen eines Aneurysmas (a) und eines Bronchialbaumes (b) mit Marching Cube: (Quelle: [30])

Das Oberflächennetz repräsentiert die Isooberfläche sehr genau. Die große Anzahl von Polygonen kann hierbei nicht reduziert werden. Ebenso kann die Auflösung des Polygonnetzes an komplexen Krümmungen wenig verändert werden. Hinzu kommt, dass das Verfahren kleinere Gefäße teilweise gar nicht darstellen kann. Gleichzeitig stellt die treppenartige Gesamtdarstellung ein großer Nachteil dar (siehe Abbildung 5.8). Durch eine Glättung kann man dem entgegenkommen, was allerdings zu Volumenverlust führen kann.

**Constrained Elastic Surface Nets** Dieser Algorithmus kann zur Erzeugung ei-

ner glatten Oberfläche aus einem segmentierten Volumendatensatz [14] verwendet werden. Als das *Surface Net* wird die Vernetzung von Punkten auf der Oberfläche des segmentierten Objektes bezeichnet. Bei diesem Verfahren werden wie im Marching Cubes Algorithmus lokale Zellen betrachtet, deren Eckpunkte von den Mittelpunkten von acht benachbarten Voxel repräsentiert werden. Auch hier werden diese Werte miteinander verglichen um zu entscheiden, ob die Zelle als *Surface Cube*, also Zellen, die in der Schnittebene liegen, betrachtet wird oder nicht. Sind alle Werte identisch, so liegt die Zelle innerhalb oder außerhalb des Volumens. Weisen die Eckpunkte unterschiedliche Werte auf, wird im Zentrum dieser Surface Cube ein Knoten erzeugt. Die benachbarten Knoten werden miteinander verbunden und es entsteht das Surface Net.

Auch hier liegt zunächst eine sehr treppenartige Anordnung der Knoten vor, welche jedoch in einem weiteren Schritt elastisch verformt wird, um so annähernd stetige Normalen der Oberfläche zu erhalten. Hierzu wird jeweils zwischen zwei benachbarten Knoten die quadrierte Länge der Verbindung als Energiemaß definiert. Für jeden Knoten gilt es dann, die Summe der Energie für die ausgehenden Verbindungen zu minimieren. Dies erfolgt durch mehrere Iterationen, in denen die Verschiebung der Knoten allerdings durch die Grenze des jeweiligen Surface Cube beschränkt ist, um eine Schrumpfung des Objektes zu vermeiden. Auf der Grundlage eines so erzeugten Netzes wird eine polygonale Oberfläche erzeugt.

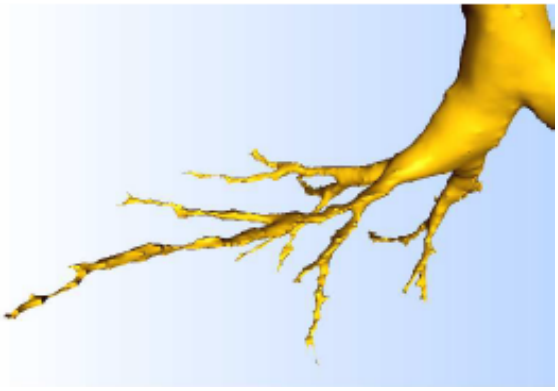

Abbildung 5.9.: Visualisierung eines eines Bronchialbaumes mit Constrained Elastic Surface Nets (Quelle: [30])

Das Ergebnis ist eine glatte Oberfläche, die sehr stark dem Segmentierungsergebnis entspricht. Scharfe Kanten, sowie dünne Strukturen bleiben erhalten. Jedoch werden die dünnen Strukturen meist als Linie abgebildet und haben somit kein Volumen mehr (Abbildung 5.9). Dadurch ist das Verfahren für baumartige Strukturen nur bedingt geeignet.

**MPU Implicits** Das *Multi-level Partition of Unity Implicits* Verfahren approximiert eine implizite Oberfläche aus einer unstrukturierten Punktwolke und wurde

2003 von *Ohtake et al.* in [22] vorgestellt. Durch eine Vielzahl von Parametern kann die Genauigkeit der angenäherten Oberfläche gesteuert werden.

Die Verwendung von MPU Implicits für die Erstellung von Oberflächen für Gefäßbäume wir von *Schumann et al.* in [30] beschrieben. Hierbei wird eine segmentierte Gefäßmaske als Basis verwendet. Aus deren Randvoxeln und Normalenvektoren die benötigte Punktwolke extrahiert werden. Dabei muss vor allem bei sehr kleinen Gefäßen mit einem Durchmesser von nur einem Voxel darauf geachtet werden, dass eine ausreichend große Anzahl von Punkten extrahiert wird.

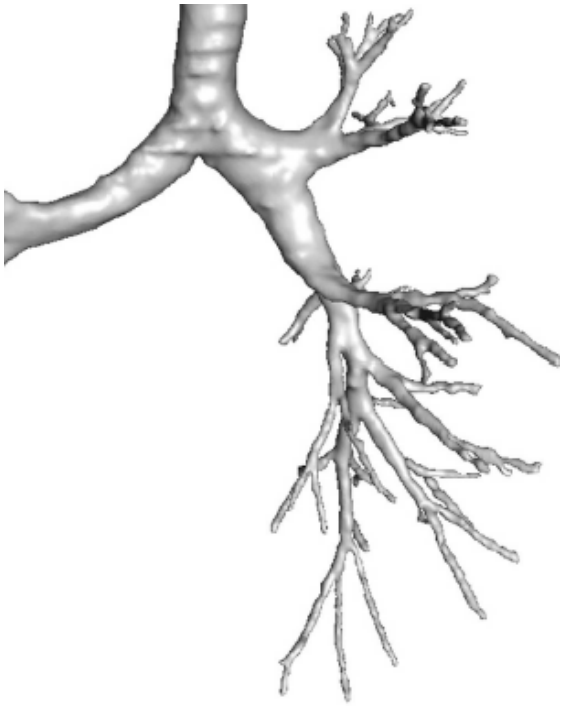

Abbildung 5.10.: Visualisierung eines Bronchialbaumes mit MPU Implicits (Quelle: [30])

Die Vielzahl der zu bestimmende Parameter ist zeitaufwendig und erschwert die Oberflächenerzeugung. Zusätzliche Methoden, die diese Parameter automatisch bestimmen, erleichtern die Anwendung in der Praxis. Das Ergebnis, unter den jeweils eingestellten Parametern, sind artefaktfreie, glatt wirkende Oberflächen (siehe Abbildung 5.10). Das Verfahren kann für die Therapieplanung als auch für die Gefäßdiagnostik genutzt werden. Bei der Therapieplanung wäre allerdings eine Verbindung mit einem Datenmodell notwendig. Der hohe Rechenaufwand bleibt allerdings als Nachteil erhalten. Zu dem steht nicht bei allen Segmentierungsverfahren eine Gefäßmaske zur Verfügung, die somit nachträglich erzeugt werden müsste.

### Zusammenfassung

In diesem Abschnitt wurde eine Vielzahl aktueller und bekannter Visualisierungsverfahren vorgestellt. Die meisten von ihnen basieren auf der Approximation eines kreisförmigen Querschnittes ([16], [11], [13], [21]) und eignen sich somit nur für eine topologischen Darstellung der Gefäße. Allerdings sind diese Verfahren nicht für die Visualisierung von pathologischen oder anatomischen Veränderungen geeignet. Ebenso wenig können sie in der Gefäßdiagnostik eingesetzt werden. Lediglich das Simplex Mesh, das Marching Cube und das MPU Implicits Verfahren können Gefäße mit unterschiedlichen Gefäßquerschnitte darstellen. Hierbei wurde jedoch das Simplex Mesh Verfahren in [3] nur unzureichend beschrieben, um über die Eignung bezüglich der Visualisierung tubulärer Strukturen mit unterschiedlichen Querschnitten Aussagen zu treffen. Bei den anderen Verfahren ([19], [30]) wird eine Maske der Gefäße benötigt. Somit können Segmentierungsergebnisse, die nur eine symbolische Beschreibung extrahieren, nicht direkt visualisiert werden. Die Gefäßdarstellung mittels impliziten Oberflächen ([30]) erzeugen zwar organisch wirkende Oberflächen, sind aber auch mit einem hohen Rechenaufwand behaftet.
In Tabelle 5.1 werden die vorgestellten Verfahren direkt miteinander verglichen.

| **Verfahren** | **Genauigkeit** | | | | **organische Darstellung** | **Verzweigungen** | **Strukturen im Inneren** | **Gefäßenden** |
|---|---|---|---|---|---|---|---|---|
| | T | D | Q | S | | | | |
| Kegelstümpfe | x | x | - | x | - | - | x | x |
| Freiformflächen | x | x | x | - | x | x | - | ? |
| Simplex Meshes | x | x | x | x | x | x | - | x |
| Subdivision Surfaces | x | - | - | x | var | x | - | - |
| Convolution Surfaces | x | x | - | x | var | x | - | x |
| Marching Cubes | x | x | x | x | - | / | - | x |
| Constrained Elastic Surface Nets | x | x | x | | var | / | x | x |
| MPU Implicits | x | x | x | x | var | / | - | x |

Tabelle 5.1.: Vergleich der vorgestellten modellbasierten Verfahren (oberer Abschnitt) und voxelbasierten Verfahren (unterer Abschnitt).
Beim Vergleich der Genauigkeit wird die Darstellungsqualität bzgl. der Topologie (T), des Verlaufs der Durchmesser (D), des reellen Gefäßquerschnitts (Q) und dünneren Strukturen (S) betrachtet. Zudem wird das organische Aussehen der Gefäße und der Verzweigungen bewertet. Weitere Vergleichsfaktoren betrachten, ob das Verfahren Strukturen im Inneren erzeugt und ob die Gefäßenden verschlossen sind.
Ein *x* besagt, dass das Verfahren das Kriterium erfüllt und ein -, dass dies nicht der Fall ist. *var* bedeutet, dass es ein Kriterium zu einem variablen Grad erfüllen kann. Zu Feldern, die mit einem *?* gekennzeichnet sind, können keine Aussagen getroffen werden.

### 5.1.2. Pipeline Konzept in VTK

In VTK gibt es zwei wichtige Konzepte: Das *Graphische Darstellungsmodell* (engl. *Graphics Model*) und die *Visualisierungs Pipeline* (engl. *Visualization Pipeline*). Als Schnittstelle der beiden Bereiche dienen sogenannte *Mapper*.

#### Graphische Darstellungsmodell

Das graphische Darstellungsmodell abstrahiert von der verwendeten Grafikbibliothek (z.B. OpenGL). Durch die Abstraktion wird diese austauschbar und ist robust gegen Standardwechsel. Das Modell besteht aus Klassen und Klassenhierarchien, deren Namen an die Welt des Filmes angelehnt sind. Dabei konvertiert das Modell letztlich grafische Daten in das Bild der Szene, abhängig von Mappern, den Lichtquellen, der Kameraposition, den Darstellern und den Requisiten. Diese *Szene* wird durch die Instanziierung der Klassen gebildet.[28]

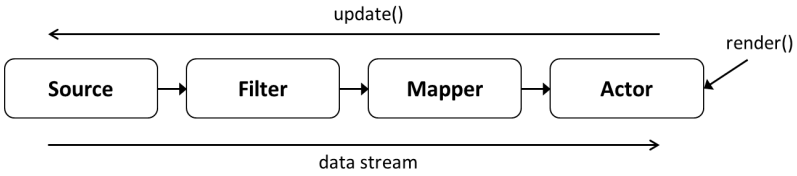


Abbildung 5.11.: VTK Rendering Pipeline

In Abbildung 5.11 wird die VTK Rendering Pipelinekonzept veranschaulicht und in der Folge gängige Objekte innerhalb der Pipeline kurz beschrieben. Die Begriffe *Source* und *Filter* sind in dem folgenden Abschnitt über die Visualisierungs Pipeline näher erläutert.

**vtkRenderer** Verwaltet die Objekt-Aktoren, Kameras und Lichtquellen während des Renderprozesses einer Szene.

**vtkRenderWindow** Grafisches Ausgabefenster, in das ein oder mehrere `vtkRenderer` ihre Daten einzeichnen können.

**vtkRenderWindowInteractor** Realisiert eine Interaktion mit der Szene.

**vtkActor** Repräsentiert ein in der Szene darzustellendes Objekt mit dessen Eigenschaften Position, Transformation, Texturen, Sichtbarkeit, usw.

**vtkProperty** Eigenschaften eines `vtkActor`s wie z.B. Farbe, Transparenz und Lichteigenschaften.

**vtkCamera** Kamera mit den Eigenschaften wie Position, Blickrichtung, usw.

**vtkLight** Lichtquelle mit den Eigenschaften wie Position, Farbe, Helligkeit, usw.

**vtkMapper** Ermöglicht die geometrische Darstellung von Daten und wird für die Visualisierung einem `vtkActor` übergeben.

### Visualisierungs Pipeline

Die *Visualisierungs Pipeline* transformiert Informationen in grafische Daten, so dass andere Pipelines mit ihnen weiter arbeiten oder sie vom System angezeigt werden können (siehe Abbildung 5.12). Eine solche Pipeline besteht aus Datenobjekten und Prozessobjekten. Die *Datenobjekte* repräsentieren Visualisierungsdaten, die mit VTK verarbeitet werden können, wie zum Beispiel Punktmengen, unstrukturierte Gitter, Polygonmodelle, usw. *Prozessobjekte* sind *Filter*, die diese Daten einlesen, verarbeiten und als verändertes Datenobjekt ausgeben. Oder aber sie stellen *Mapper* dar, der aus den Daten geometrische Primitive erstellt. Daten und Filter können beliebig oft hintereinander geschaltet werden, wobei auch die Anzahl der Eingänge und Ausgänge von Filtern beliebig sein können. Am Ende der Pipeline werden die Daten dem Mapper übergeben, der diese für die grafische Ausgabe, also für den *Renderer*, umwandelt.

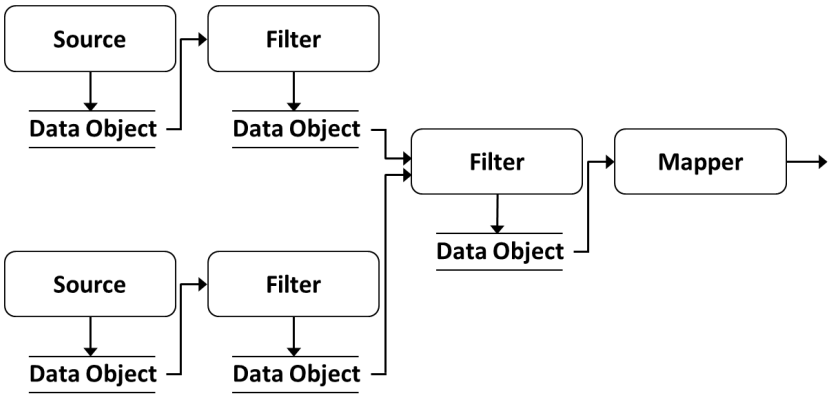


Abbildung 5.12.: VTK Visualisierungs Pipeline: Die Pfeile repräsentieren hierbei den Datenfluss.

## 5.2. Anforderungen

Im Rahmen dieser Arbeit soll ein einfaches und schnell umsetzbares Visualisierungsverfahren für den dreidimensionalen Raum realisiert werden. Auf Grund der in Abschnitt 5.1.1 genannten Vorteile, soll ein modellbasiertes Verfahren verwendet werden. Für die Benutzerinteraktion ist es wichtig, dass die Visualisierung bei Veränderung der Ansicht oder der Farbinformation in Echtzeit funktioniert und es zu keinen größeren Verzögerungen kommt. Außerdem sollte es möglich sein, einzelne Strukturen zu selektieren und auszublenden. Auch sollen die einzelnen Strukturen getrennt voneinander eingefärbt werden können. Das gleichzeitige agieren mit anderen Bildern und Oberflächen muss ebenso berücksichtigt werden.

Trotz der schnellen Reaktion auf die Benutzereingabe ist eine möglichst genaue Wiedergabe der Gefäßsysteme, sowie ein organisches Aussehen das Ziel. Eine weitere Anforderung ist, dass für die navigierten Bronchoskopie ein „Durchfliegen“ des Gefäßinneren

möglich sein sollte und somit Strukturen im Inneren ausgeblendet werden müssen. Das Visualisierungsverfahren soll in die bestehenden Darstellungstechniken des MITK integriert werden.
In Tabelle 5.2 sind die aufgestellten Anforderungen zusammengefasst dargestellt.

- Einfaches und schnell umsetzbares dreidimensionales Visualisierungsverfahren
- Modellbasiertes Verfahren
- Exakte Wiedergabe und organisches Aussehen
- Ausblendung von Fragmenten im Inneren
- Einzelne Strukturen müssen selektiert, eingefärbt und ausgeblendet werden können
- Interaktion mit anderen geöffneten Bildern und Oberflächen innerhalb MITK
- Visualisierung in Echtzeit

Tabelle 5.2.: Zusammenfassung der aufgestellten Anforderungen an die Visualisierung

## 5.3. Methoden und Realisierung

Um die geforderten Anforderungen zu verwirklichen, wurde im Rahmen dieser Arbeit ein modellbasiertes Verfahren, das auf einfachen Primitiven beruht, verwendet. Dieses Verfahren kann schnell und einfach umgesetzt werden und erfüllt hierbei mit zusätzlichen Verarbeitungsschritten die Anforderungen. Hierfür erzeugt das Verfahren aus dem, in Kapitel 4 beschriebenen Datenmodell, eine Oberfläche. Die für diese Arbeit entworfene Implementierung wurde mittels Datenobjekten und Visualisierungstechniken des VTK realisiert und in einem neu entwickelten Mapper umgesetzt. Somit fügt es sich in das Darstellungskonzept von MITK ein, dass die VTK Rendering Pipeline umsetzt.

Da das momentane Skelettierungsverfahren nur Informationen über einen kreisrunden Gefäßquerschnitt liefert, wurde sich in der Folge nur auf die Visualisierung solcher Strukturen beschränkt. Jedoch können anstatt der gewählten Primitiven auch beliebige andere geometrische Formen verwendet werden.

Für die Visualisierung des Datenmodells müssen in einem ersten Schritt die Dimensionen des Gefäßbaumes erfasst werden. Diese Maße werden in einer eigens hierfür implementierten Geometrie-Klasse festgehalten. Die eigentliche Visualisierung erfolgt dann in zwei Generierungsschritten: Dem Erzeugen der Strukturen und dem Schneiden der verbundenen Teilstücke. Diese einzelne Vorgehensweisen, sowie die Verwertung der Farbinformation, aus der gesondert realisierten Property-Klasse (siehe Abschnitt 6.3.3), wird in der Folge beschrieben.

### 5.3.1. Geometrie und Bounding Box

Der Graph als Datenobjekt benötigt eine Geometrie, die die Lage des Objektes in Raum und Zeit beschreibt. Hierfür wird jedem Bildpunkt neben seinem Index in Pixeln eine physikalische Ausdehnung (*Spacing*) zugeordnet. Mit dieser Angabe von Breite, Höhe und Tiefe eines Bildpunktes wird ein Volumen im Raum beschrieben. Um die Lage des gesamten Bildes im Raum festzulegen, muss ein sogenannter *Origin*, also der Bildursprung, festgelegt werden. Diese Angaben werden aus dem Originalbild, aus dem die Gefäße segmentiert wurden, übernommen. Zu dem muss der darzustellende Bereich definiert werden. Hierfür werden alle Gefäße betrachtet, d.h. der gesamte Graph wird traversiert und es werden die Knoten und Kanten, die die äußeren Randpunkte darstellen, gesucht. Somit hat man die Weltkoordinaten dieser Punkte und muss nur noch den Gefäßradius hinzu addieren. Insgesamt spannen die berechneten Werte die sogenannte *Bounding Box* auf. Die übernommenen und berechneten Werte werden in einer eigens implementierten Geometrie-Klasse (`mitkTubeGraphGeometry3D`) gespeichert und dem Datenobjekt zugewiesen.

### 5.3.2. Generierung der Oberflächen

Zunächst werden die im Modell beschriebenen Kanten und Knoten mittels einfachen Primitiven visualisiert. Um den Gefäßverlauf zu beschreiben, wird für jede *Tube*, also Start- und Endknoten sowie der Kante dazwischen, ein Pyramidenstumpf erzeugt. Dieser Pyramidenstumpf wird über den Filter `vtkTubeFilter` von VTK generiert und in ein `vtkPolyData` abgelegt. Für das Erzeugen benötigt dieser die Information über den Verlauf des Gefäß im Raum und den richtigen Durchmesser der Gefäßabschnitte. Diese Information werden mittels Punkte und deren Verbindungen über Linien als Eingabeparameter in den Filter gegeben. Hierfür werden für jedes Element der Kante und der Knoten aus dem Graph ein Punkt (`vtkPoint`) erzeugt und der passende Radius abgespeichert. Diese werden dann in der richtigen Reihenfolge mittels Linien verbunden. Der `vtkTubeFilter` erzeugt nun um diese Linie ein Pyramidenstumpf. Für den gleichmäßigen Verlauf wird der Radienverlauf zwischen zwei Punkten interpoliert. Zudem kann mittels Erhöhung der Anzahl von Seitenflächen der Pyramide ein möglichst runder und damit organischer Eindruck vermittelt werden. Das so generierte polygonale Dataset wird als Eingangsparameter in einem `vtkPolyDataMapper` abgelegt. Dieser wiederum wird einem `vtkActor` zugewiesen (siehe Abschitt 5.1.2). Somit wird eine Tube, also ein Gefäß, durch einen Actor repräsentiert. In der Szene werden alle Darsteller vereint dargestellt.

An den Verzweigungen der tubulären Strukturen kann es vorkommen, dass bei der Vereinigung der Röhren leere Zwischenräume an der Oberfläche entstehen. Diese werden durch Erzeugen von Kugeln an den Verzweigungen geschlossen. Auch die Gefäßenden werden somit abgeschlossen und wirken natürlicher. Diese Kugeln werden als Objekte von `vtkSphereSource` auch als `vtkPolyData`s erzeugt. Auch hier wird für jeden Knoten ein eigener Actor erzeugt und gespeichert.

Die getrennte Speicherung der Röhren und der Kugeln macht es möglich, dass trotz Änderungen der Farbinformation oder der Sichtbarkeit der tubulären Strukturen im Mo-

dell, diese nicht jedes Mal erneut berechnet werden müssen. Nur bei einer Änderung der Datenstruktur, also bei dem Hinzufügen oder dem Entfernen von Kanten und Knoten, sowie Änderungen in den Elementen, die den Gefäßverlauf beschreiben, wird eine Neuberechnung sowie das Schneiden der Strukturen notwendig. Dies wird über Modifizierungs-Flags gesteuert, welche die Klassen `mitkTubeGraph` und `mitkTubeGraphProperty` intern verwalten.

### 5.3.3. Entfernung der inneren Fragmente

Im weiteren Schritt werden die generierten Primitive miteinander geschnitten, um so Strukturen im Inneren, die durch Überlagerungen entstehen, zu vermeiden. Hierfür wird immer ein Knoten mit all seinen abgehenden Kanten betrachtet. Da es aber in VTK keine direkte Möglichkeit gibt zwei Polydatas miteinander zu schneiden, wurde hier ein eigenes Verfahren entwickelt und angewendet.

Bei diesem sogenannten *Clipping* ist es nötig, dass eine implizite Funktion, die einen Zylinder beschreibt, das Gefäß repräsentiert. Da der Radius eines Gefäßes um eine Verzweigung teilweise sehr schnell ab oder zunehmen kann, kann nicht der Radius der Kugel verwendet werden (siehe Abbildung 5.13). Dies kommt zu Stande, da der Knoten und der hier abgelegt Radius für alle abgehenden Gefäße gleich ist. Da bei einer Verzweigung, ein größeres Gefäß meist in zwei dünnere Gefäße teilt, kommt es hier zu Diskrepanzen.

Um für den Zylinder aber einen möglichst genauen Durchmesser $diameter_{cylinder}$ für die Schnittstelle zu erhalten, wird entlang der Elementenliste der Kante das Element gesucht, das eine größere Distanz zum Mittelpunkt der Kugel hat, als ein Punkt auf der Kugel. Das bedeutet, dass es sicher außerhalb der Kugel liegt. Somit kann man das vorherige Element als Bezugspunkt innerhalb der Kugel verwenden. Nimmt man nun den Durchmesser dieses letzten Elementes innerhalb des Kreises $diameter_{in}$ und den Durchmesser des ersten Elementes außerhalb des Kreises $diameter_{out}$ kann der interpolierte Durchmesser wie folgt berechnet werden:

$$diameter_{cylinder} = (1 - i) * diameter_{in} + i * diameter_{out}$$

Wobei der Interpolationsfaktor $i$ über die jeweiligen Entfernungen des letzten Elements innerhalb $distance_{in}$ und des ersten Elements außerhalb $distance_{out}$ der Kugel zur Kugelwand berechnet wird.

$$i = (radius_{sphere} - distance_{in}) \,/\, (distance_{out} - distance_{in})$$

Sonderfälle, wie zum Beispiel, dass das erste Element des Vektors schon außerhalb der Kugel liegt oder der andere Extremfall, dass kein Element außerhalb liegt, werden gesondert betrachtet und berechnet.

Da der generierte Zylinder nun einen entsprechenden Durchmesser hat, er aber nach Definition endlos ist, muss er zunächst geschnitten werden. Dieser Schnitt erfolgt mittels einer selbst definierten Ebene. Hierfür wird der Zylinder zunächst in den Ursprung $[0, 0, 0]$ gelegt. Die Rotation des Zylinders erfolgt um die y-Achse, weshalb der Normalenvektor der Schnittebene auf $[0, 1, 0]$ festgelegt wird. Die Differenz der beiden impliziten Funktionen, also Zylinder und Schnittebene, voneinander wird mit `vtkImpliciteBoolean`

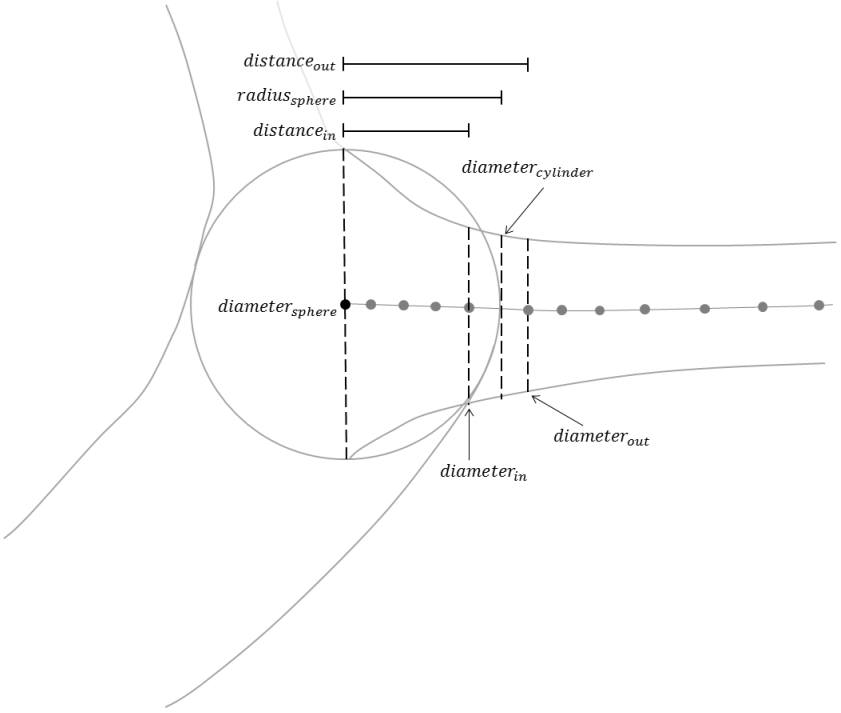


Abbildung 5.13.: Darstellung der Clipping Parameter: Neben den für das Clipping verwendeten Radien (gestichelte Linien), werden die Distanzwerte mit angegeben. In grau dargestellt ist die Kugel als Verbindungsstück und die auf den Gefäßverlauf beruhenden Pyramidenstümpfe.

berechnet. Das Resultat ist ein Zylinder mit richtigem Durchmesser, der aber um die y-Achse rotierend im Ursprung liegt.

Um den geschnittenen Zylinder mit der richtigen Orientierung und an die richtige Stelle des Gefäßes zu bekommen, wird eine Transformationsmatrix ermittelt. Hierfür wird zunächst ein Orientierungsvektor $\vec{o_1}$, der innerhalb des Gefäßes liegt und den Verlauf widerspiegelt, benötigt. Für diesen Zweck wird die Koordinate des ersten Elementes $\vec{e}$ im Vektor genommen und mit dem im Zentrum der Kugel $\vec{c}$ subtrahiert.

$$\vec{o_1} = \vec{e} - \vec{c}$$

Der so entstandene Vektor wird für die weiteren Berechnungen normiert $\vec{n_1} = \vec{o_1} / |\vec{o_1}|$.

Um die weiteren Rotationsvektoren der Matrix zu erhalten, wird ein willkürlich gewählter Vektor $\vec{r}$ auf die imaginäre Ebene (mit dem Orientierungsvektor als Normalenvektor) projiziert.

$$\vec{o_2} = \vec{r} - (\vec{r} * \vec{n_1}) * \vec{n_1}$$

Und man erhält somit ein orthogonaler Vektor $\vec{o_2}$ zu dem bereits berechneten, normierten Orientierungsvektor. Dieser wird wiederum auf die Normalenform $\vec{n_2}$ gebracht.

Der dritte orthogonale Vektor wird mittels des Kreuzproduktes der ersten beiden Orientierungspunkten berechnet und ist gleichzeitig bereits normiert.

$$\vec{o_3} = \vec{n_1} \times \vec{n_2} = \vec{n_3}$$

Für die Translation wird der Zentrumspunkt $\vec{c}$ der Kugel verwendet und da keine Skalierung vorgenommen werden muss, wird die untere Zeile der Matrix mit den Werten der Identitätsmatrix aufgefüllt. Die Matrix setzt sich also wie folgt zusammen:

$$\begin{pmatrix} n_{2_x} & n_{1_x} & n_{3_x} & c_x \\ n_{2_y} & n_{1_y} & n_{3_y} & c_y \\ n_{2_z} & n_{1_z} & n_{3_z} & c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Wird die Transformationsmatrix nun auf den Zylinder angewendet, befindet sich dieser nun wie in Abbildung 5.14 dargestellt, an der Stelle des Pyramidenstumpfes und kann diesen repräsentieren.

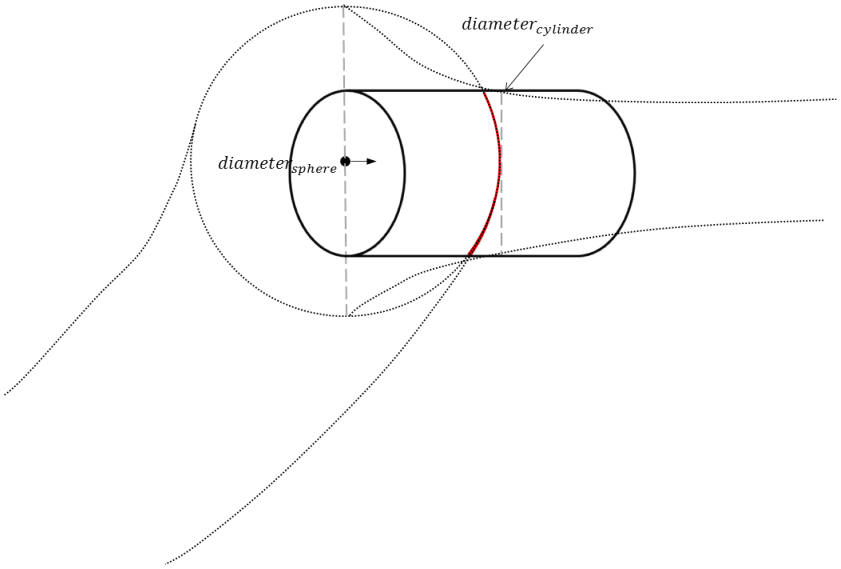


Abbildung 5.14.: Darstellung des Zylinders innerhalb des Gefäßes: Der Zylinder (schwarz) repräsentiert den Gefäßverlauf. Dieser wird für das Schneiden überflüssiger Strukturen im Inneren (rot) verwendet.

Sind alle abzweigenden Gefäße eines Verzweigungspunktes mit einem Zylinder berechnet worden, werden sie mit der Kugel und den jeweilig anderen Pyramidenstümpfen geschnitten. Dieses Clippen wird mit `vtkClipPolyData` realisiert. Hierbei ist die Kugel oder der Pyramidenstumpf der Input und der Zylinder, der eine anderen Tube darstellt, die Funktion die ausgeschnitten werden soll.

Die durch das Clipping veränderten Polydatas werden abgespeichert und gegebenenfalls mit weiteren Strukturen geschnitten. Somit werden bei den Verbindungsstücken

nur noch die Teile der Kugel angezeigt, die die Darstellung abrunden und die Lücken schließen. Die abschließenden Kugeln an den Gefäßenden sind vollständige Halbkugeln.

### 5.3.4. Weitere Informationen zur Visualisierung

Die Information, welche Farben den einzelnen Tubes zugeordnet sind und ob diese überhaupt sichtbar sind, befindet sich in Property-Klasse (siehe Abschnitt 6.3.3). Bei Verbindungsstücken, also den geschnittenen Kugeln, kann es, aufgrund von unterschiedlichen Farbinformationen der abgehende Gefäße, zu Unklarheiten kommen. Hierbei wird ein Mittelwert der Farbe der angrenzenden Gefäße gewählt. Die Sichtbarkeit einer Zwischen- oder Endkugel ändert sich erst, wenn alle abgehenden Gefäße unsichtbar gesetzt werden.

Des Weiteren kann es bei manchen Fällen sinnvoll sein, eine Öffnung der Oberfläche zu erzeugen, um ins Innere der tubulären Strukturen zu navigieren. Hierbei wird eine abschließende Halbkugel bei der Darstellung ausgelassen. Diese Kugel kann der Benutzer selbst bestimmen. Standardmäßig wird die Kugel des dicksten Gefäßes verwendet (siehe Abschnitt 6.3.1: Definition der Wurzel).

Alle Veränderungen an dem Datenobjekt, wie zum Beispiel die Veränderung der Position, Orientierung oder Farbänderungen, resultieren in einer konsistenten Anpassung der Oberflächenelemente.

### 5.3.5. Klassenübersicht

Für die Visualisierung der tubulären Strukturen wurde innerhalb des von MITK bereitgestellten Rendering- Prozesses ein eigener Mapper `mitkTubeGraphVtkMapper3D` implementiert. Dieser Mapper erbt von `mitkVtkMapper3D`, welche die Oberklasse für alle VTK basierten Mapper innerhalb des MITK ist. In ihm wurde das in dieser Arbeit beschrieben Verfahren zur Visualisierung umgesetzt.

Die Informationen zur Visualisierung bezieht dieser Mapper einerseits aus der Datenklasse `mitkTubeGraph`, andererseits aus der zugehörigen Property-Klasse `mitkTubeGraph Property`. Form und Verlauf der Gefäße erhält sie dabei aus den Datenklassen (siehe Abschnitt 4.3.5). Informationen über Farbe und Transparenz eines Gefäßes werden aus der Property-Klasse entnommen. Durch Modifizierungsflags der Property-Klasse und des Graphen wird die Szene nur bei Änderungen der jeweiligen Klasse neu berechnet. Wie in Abbildung 5.15 zu sehen ist, wird in zwei Fälle unterschieden: Zum einen die Neuberechnung der Primitiven und zum anderen in die Aktualisierung der Farb- und Sichtbarkeitinformationen.

Die `mitkTubeGraphGeometry3D` Klasse enthält die Information über Lage und Ausdehnungen des tubulären Systems in Zeit und Raum.

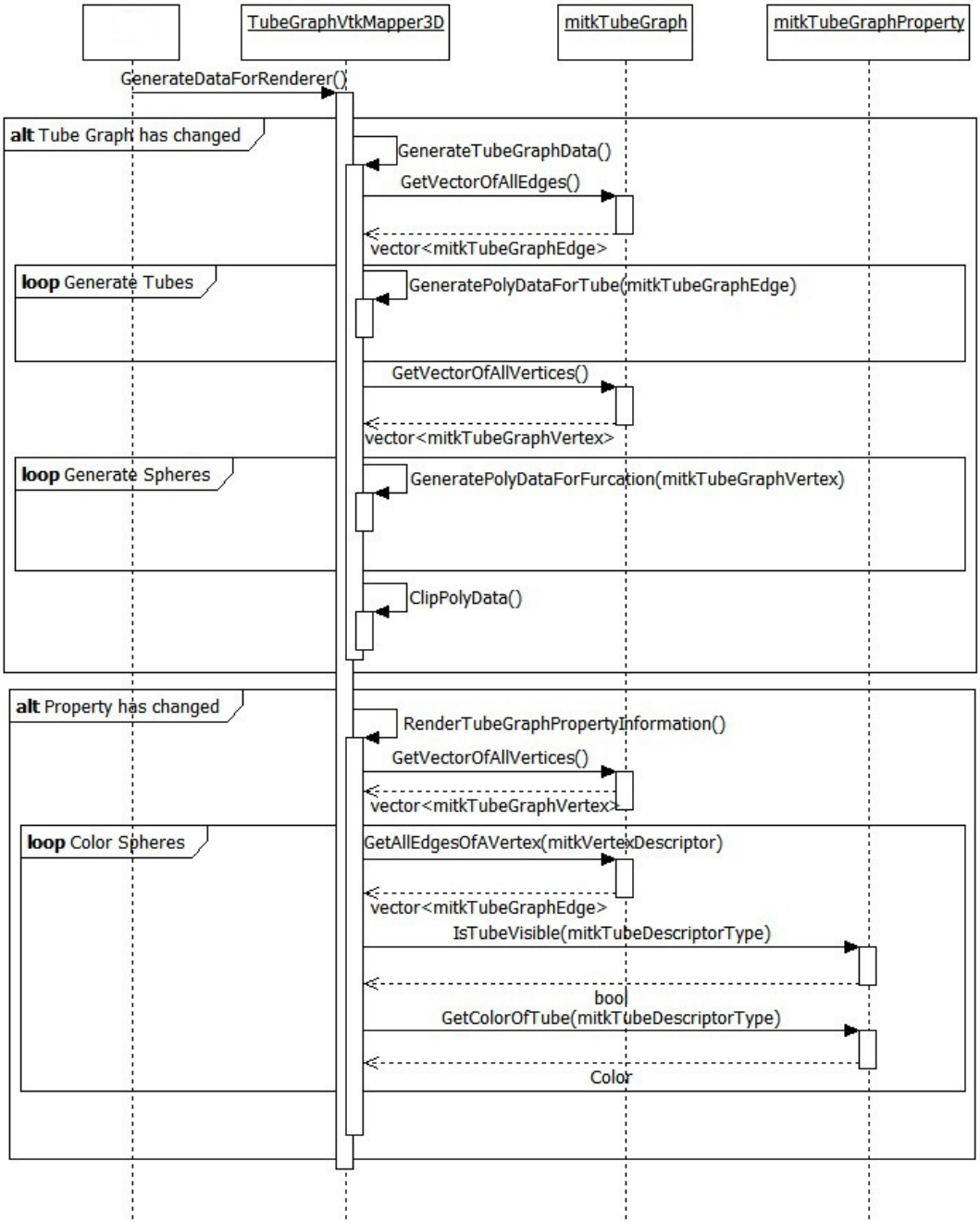


Abbildung 5.15.: Vereinfachtes Sequenzdiagramm des Rendering-Prozesses

## 5.4. Ergebnisse

In dieser Arbeit wurde ein einfaches und schnell realisierbares Verfahren zur Darstellung tubulärer Strukturen im dreidimensionalen Raum entwickelt, das den gestellten Anforderungen genügt.

Auf Grundlage des in Kapitel 4 entwickelten Datenmodells werden einfache Primitive erzeugt, die die Oberfläche der tubulären Systeme darstellen. Hierbei wird eine genaue Darstellung der topologischen Informationen gewährleistet und die Primitive an die morphologischen Eigenschaften angenähert (siehe Abbildung 5.16). Um die Darstellungsqualität zusätzlich zu erhöhen, werden die Gefäßenden mittels Halbkugeln verschlossen. Das organische Aussehen wird leider durch die nicht fließenden Übergänge der Strukturen beeinträchtigt.

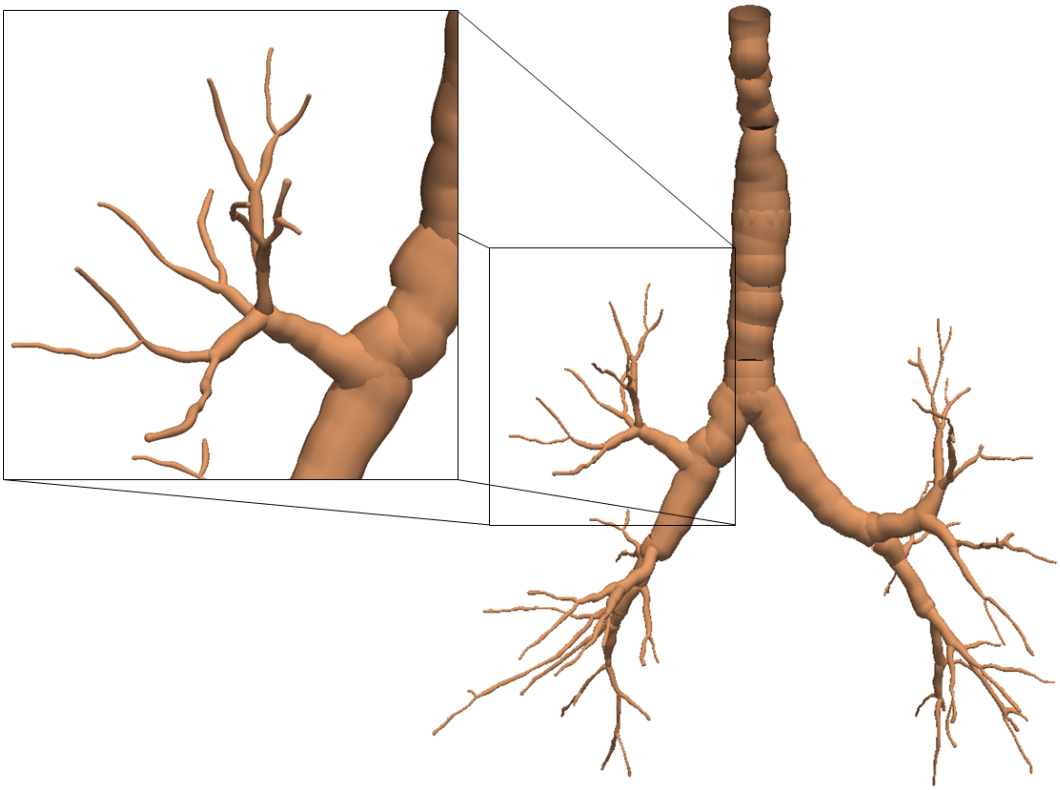

Abbildung 5.16.: Ergebnis des realisierten Visualisierungsverfahren mittels einfachen Primitiven: Visualisierung eines Bronchialbaumes und einer vergrößerten Darstellung.

In einem selbst entwickelten Verfahren werden die erzeugten Oberflächenobjekte miteinander geschnitten, so dass innere Fragmente vermieden werden. Dieser zusätzliche Schritt

ermöglicht das Navigieren innerhalb der tubulären Strukturen und bietet vor allem für die navigierte Bronchoskopie große Vorteile (siehe Abbildung 5.17).

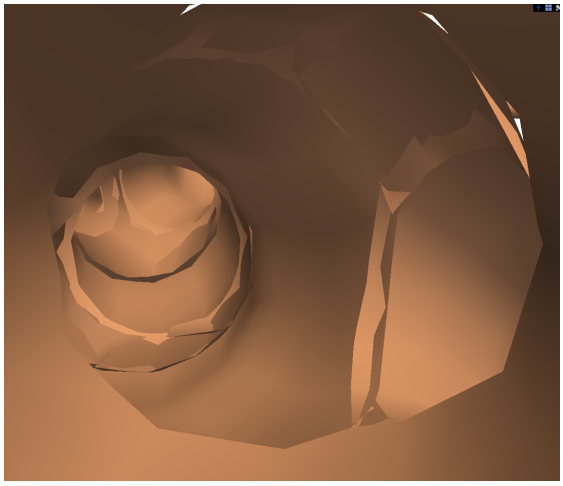

Abbildung 5.17.: Innenansicht der visualisierten Oberflächen: Durch ein selbstentwickeltes Clipping-Verfahren (siehe Abschnitt 5.3.3) werden die Oberflächenfragmente im Inneren entfernt und die Navigation innerhalb der tubulären Strukturen möglich. Die Darstellung der Strukturen von Innen könnte durch ein aufwendigeres Verfahren noch verbessert werden.

Die generierten und geschnittenen Oberflächenstrukturen können einzeln ausgewählt werden. Diese Funktionalität ist Grundlage für die in Kapitel 6 beschriebene Interaktionskomponente. Sie können nicht nur einzeln selektiert, sondern auch eingefärbt und ausgeblendet werden.

Durch das Verwenden der Datenklassen und Renderingkonzepte von VTK, ist das realisierte Verfahren in das Visualisierungskonzept von MITK eingebunden. Das bietet dem Benutzer die Möglichkeiten neben der Visualisierung der tubulären Strukturen andere Bilder und Oberflächen anzeigen zu lassen und mit ihnen zu interagieren. Hierbei können die Interaktionsmöglichkeiten für das dreidimensionale Fenster genutzt werden. So kann die gesamte Szene zum Beispiel rotiert, skaliert und verschoben werden.

Innerhalb der Therapieplanung ist die Darstellung von mehreren Bilddaten zur gleichen Zeit notwendig, um die Gefäße zum Beispiel im Vergleich zur Lage des Tumors oder innerhalb der Organe zu betrachten.

Alle Veränderungen an dem Datenobjekt, wie zum Beispiel die Veränderungen der Datenstruktur oder der Farbinformationen, resultieren in einer konsistenten Anpassung der Oberflächenelementen der tubulären Strukturen in Echtzeit. Dies ist gerade deshalb möglich, da bei jeder Aufforderung zur Aktualisierung geprüft wird, ob die Primitive neu berechnet werden müssen. Dies ist nur der Fall, wenn der Graph an sich modifiziert wurde. Bei Änderungen der visuellen Informationen, also die die Property-Klasse betreffen,

werden nur die Farbinformationen und die Sichtbarkeit der einzelnen tubulären Strukturen angepasst. Durch diese Prüfung können unnötige Berechnungszeiten für das Erstellen und Clippen der Primitiven eingespart werden.

Das Visualisierungsverfahren ist bisher nur für Gefäße mit einem kreisrunden Querschnitt realisiert, da das momentane Verfahren zur Skelettierung nur diese Informationen an die Graphenerzeugung weitergeben. Jedoch bietet die Datenstruktur die Möglichkeit auch andere Querschnitte und Formen von Gefäßen zu zulassen. Diese Informationen können auch in diesem realisierten Visualisierungsverfahren umgesetzt werden. Hierfür müssen die verwendeten Primitive durch geeignete geometrische Repräsentationen ersetzt werden.

# 6. Interaktionen

## 6.1. Grundlagen

Die zu implementierende Interaktionskomponente wird in das MITK eingebunden. Im folgenden Abschnitt wird deshalb das bestehende Interaktionskonzept kurz beschrieben.

### 6.1.1. Interaktionskonzept in MITK

Speziell bei computergestützten Therapie- und Operationsplanung, wird eine komplexe *Mensch-Maschine-Interaktion* benötigt. Hierbei reicht es nicht aus, die Daten aus nur einer Perspektive und in einer Größe beobachten zu können. Vielmehr muss es dem Mediziner möglich sein, die Daten zu skalieren, zu rotieren und zu verschieben, um zu einer möglichst genauen Analyse und Diagnose zu gelangen. Die Eingaben des Benutzers und die folgende Reaktion der Software werden als Interaktion bezeichnet.

Eingaben können durch unterschiedliche Geräte, wie zum Beispiel Maus, Tastatur oder auch Tracking-Systemen erfolgen. Somit muss eine Festlegung von abstrakten Ereignissen, die unabhängig vom Eingabegerät sind, erfolgen. Diese Ereignisse und ihre Reihenfolge müssen dann in einem weiteren Schritt interpretiert werden.

Das Interaktionskonzept in MITK basiert auf der Automatentheorie und ist mit *hierarchischen Zustandsmaschinen* (engl. *State Machine*) umgesetzt. Eine Zustandsmaschine wird benötigt, da die gleiche Interaktion in unterschiedlichen Kontexten verschiedene Aktionen bewirken. Bei dem Einsatz von Zustandsmaschinen kann dies durch den aktuellen Zustand gesteuert werden. Die Vorteile der Verwendung von Zustandsmaschinen liegen in der Zerlegung und Verfeinerung komplexester Systeme in viele kleine Subsysteme. Dies verhilft zum einen zu einer besseren Übersicht und zum anderen ist eine Wiederverwendung möglich. Zum Beispiel ermöglicht das Konzept, dass gleiche Interaktionsschritte bei unterschiedlichen Modulen in unterschiedlichen Kontexten angewendet werden können.

Folgende vier Begriffe sind im Zusammenhang mit Zustandsmaschinen von Bedeutung [38]:

**Ereignis** (engl. *Event*) kann eine Eingabe des Benutzers sein oder ein zeitliches Intervall.

**Aktion** (engl. *Action*) ist das Resultat, das auf ein Ereignis folgt.

**Übergang** (engl. *Transition*) wird durch ein bestimmtes Ereignis ausgelöst und führt von einem akzeptierten Zustand in einen resultierenden. Dabei ist es möglich das dies der gleiche Zustand ist.

**Zustand** (engl. *State*) repräsentiert die Information über die bisherigen Ereignisse.

MITK verwendet deterministische Zustandsmaschinen. Das bedeutet:

- Eine endliche Anzahl an Zuständen
- Ein bestimmtes Ereignis hat aus einem bestimmten Zustand nur ein resultierender Zustand
- Ein Startzustand
- Mehrer Endzustände sind zugelassen

Modelliert sind diese als sogenannte *Mealy*-Zustandsmaschinen. Übergänge und Aktionen hängen hierbei von dem aktuellen Zustand und der Eingabe ab.

Diese Eingaben werden in der Klasse `mitkEventMapper` in abstrakte Ereignisse umgewandelt und an die Klasse `mitkGlobalInteraction` weitergeleitet. Dort werden sie an die aktiven Zustandsmaschinen übergeben. Abschließend wird geprüft, ob der aktuelle Zustand das Event verwerten kann.

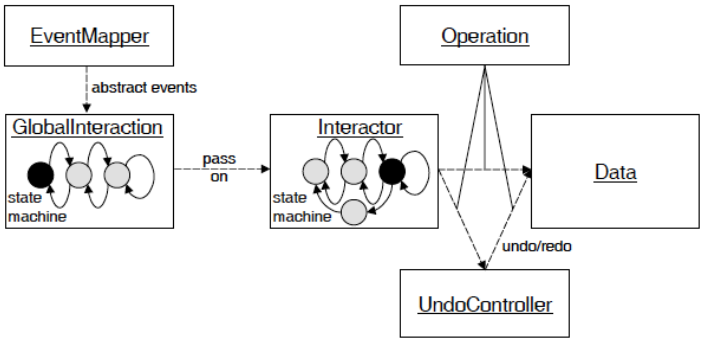


Abbildung 6.1.: Interaktionskonzept des MITK im Überblick (Quelle: [42])

Bei der Durchführung solcher Interaktionen kommt es oft zur fehlerhaften Bedienungen durch den Benutzer. Deshalb wirkt ein Zusammenspiel von Interaktion und Rücknahmemöglichkeit eines Befehls unterstützend. Die Integration eines solchen *Undo/ Redo-Konzeptes* entspricht der Benutzerfreundlichkeit. Bei Bedarf werden die ausgeführten Operationen zusammen mit ihren inversen Operationen dem `mitkUndoController` übermittelt. Dieser speichert diese Ausführungen dann in einer Liste. Bei Aufrufen der Undo-Möglichkeit wird die inverse Operation an das entsprechende Datum versendet und der vorherige Zustand der zugeordneten Zustandsmaschine wird wieder hergestellt (siehe Abbildung 6.1).

## 6.2. Anforderungen

Als Orientierungspunkt für ein interaktives Gefäßanalysetool dienen die bisher verwendeten Werkzeuge für die Analyse der Gefäße innerhalb der Leberoperationsplanungsoftware des DKFZs. Allerdings könnten nur die bekannten Konzepte übernommen werden, da die Realisierungen projektspezifisch umgesetzt wurden und somit die Realisierungen nicht mehr vorhanden sind. Die Konzepte der Benutzerführung und Interaktion sollten nach Möglichkeit übernommen und erweitert werden. Die modulare Organisation der Interaktionskomponenten soll eine Wiederverwendbarkeit für zukünftige Anwendungen ermöglichen.

Im Folgenden wird auf die genauen Anforderungen für ein fortgeschrittenes Werkzeug zu Analyse und Exploration von tubulären Strukturen eingegangen.

### Modifikation

Unter der Modifikation des Graphen versteht man topologieverändernde Aktionen. Diese werden meist durchgeführt, um die Modelltopologie der realen Topologie anzugleichen. Dem Benutzer sollte es möglich sein Äste hinzuzufügen, Teilgraphen zu erzeugen oder Gefäße zu löschen.

Durch die Segmentierung und die anschließende Skelettierung kann es in manchen Fällen zu Graphen mit unterbrochenen Gefäßästen kommen. Diese Lücke zwischen zwei Gefäßen sollte mittels Hinzufügen eines Abschnittes, also einer Kante im Graph, geschlossen werden können. Auch das Erzeugen von neuen Gefäßen, die zum Beispiel durch schlechte Bildqualität nicht segmentiert wurden, aber durch den Benutzer dem Gefäßbaum zugeordnet werden können, soll ermöglicht werden. Hierbei muss interaktiv die symbolische Beschreibung von Hand aufgebaut werden. Diese Funktionalität wäre auch für die präoperative Planung von Bypass-Operationen von Nutzen.

Werden Resektionen von Organen wie zum Beispiel der Leber vorgenommen, sind auch die Teilgefäße in dem zu entfernenden Bereich von Bedeutung. Um diese bei der computergestützten Planung zu berücksichtigen, muss die Möglichkeit geboten sein Gefäßabschnitte interaktiv zu löschen. Auch Artefakte sollen für die weitere Verarbeitung zu löschen sein.

Das Generieren von eigenständigen Teilgraphen aus dem bestehenden Gefäßgraph ist vor allem für das Trennen der beiden venösen Systeme der Leber nützlich. Diese werden durch ihre fließenden Übergänge meist gemeinsam segmentiert.

### Attributierung

Um die Gefäße nach ihren anatomischen, physiologischen und pathologischen Eigenschaften zu klassifizieren, müssen diese mit Merkmalen belegbar sein. Betrachtet man hier bei die bisherigen Anwendungsfälle sind zum Beispiel bei der Leber die Unterscheidung der Gefäßtypen (Arterie, Vene, Hohlvene und Gallengang), die Segmenteinteilung (siehe Kapitel 2.1.1) oder auch der Resektionsstatus der Gefäße mögliche Merkmalsgruppen. Unterschieden werden sollen diese Merkmale über unterschiedliche Einfärbungen der Ge-

fäße. Auch das Ausblenden von Gefäßen mit einem bestimmten Merkmal soll möglich sein.

### Annotation

Über Annotationen können Ärzte Positionen innerhalb eines Gefäßbaumes markieren und benennen. So können Markierungen und kurze Informationen von Stenosen, Aneurysmen, Tumorgewebe oder auch wichtige Verzweigungspunkte bei der Therapieplanung oder aber auch bei der navigierten Behandlung von Nutzen sein. Auch erleichtert der Austausch der Bildatensätze mit entsprechenden Hinweisen die Arbeit, bei einer Behandlung durch unterschiedlichen Ärzten.

### Information

Neben den individuellen Kommentaren gibt es Informationen, die allgemein über den Gefäßgraphen oder eben auch über einer selektierten Struktur dem Benutzer zur Verfügung gestellt werden können.

- Anzahl der Kanten und Knoten
- Durschnittliche Länge, Volumen und Durchmesser
- Durchmesserverlauf
- Hierarchie
- ID der selektieren Kante

Die genannten Anforderungen sollen mittels des in MITK gegebenen Interaktionskonzeptes (siehe Abschnitt 6.1.1) umgesetzt werden. Dabei stehen die Benutzerfreundlichkeit, sowie die Wiederverwendbarkeit der Interaktionskomponente für unterschiedliche Anwendungsfälle im Vordergrund.
Die Ergebnisse der Anforderungsanalyse sind zur besseren Übersicht in Tabelle 6.1 zusammengefasst.

- Integration in das Interaktionskonzept des MITK
- Anwendbarkeit auf unterschiedliche Anwendungsfälle
- Benutzerfreundlichkeit
- Selektion von einzelnen und mehreren Strukturen
- Modifikation des Graphen
- Attributierung der Strukturen
- Hinzufügen von Annotationen
- Bereitstellen von Information

Tabelle 6.1.: Zusammenfassung der aufgestellten Anforderungen an die Interaktionskomponenten

## 6.3. Methoden und Realisierung

### 6.3.1. Selektion und Aktivierung

Um mit den Gefäßen zu interagieren, ist eine Aktivierung von einzelnen oder mehreren Kanten (*Tubes*) notwendig. Das Bereitstellen von verschiedenen Aktivierungsmodi war bereits in der Leberoperationsplanungssoftware ReLiver umgesetzt. Dieses benutzerfreundliche Konzept wurde für die neue Interaktionskomponente übernommen und erweitert. So kann der Benutzer indem er einzelne Tubes selektiert (*Picking*) ganze zusammenhängende Gefäßäste oder auch mehrere einzelne Gefäße aktivieren und damit weiter interagieren. Realisiert wurden diese Auswahlfunktionen in einer Zustandsmaschine mit einer passenden Interactor-Klasse, die auf die Änderung der Zustände reagiert.

#### Picking

Der Benutzer hat zwei Möglichkeiten eine Tube zu selektieren: Er kann mittels einem Doppelklick oder bei Halten der Steuerungstaste (Strg-Taste) mit einem Einfachklick auf die Kante, diese selektieren. Über eine solche Benutzereingabe wird ein Punkt im dreidimensionalen Raum bestimmt und mit der Datenstruktur abgeglichen. Bei diesem Abgleich wird über den gesamten Graphen iteriert und die Tube ausgewählt, welche am nächsten an diesem Punkt liegt. Hierfür wird die Distanz zwischen dem gewählten Punkt und der Koordinate aus allen `mitkTubeElement`s berechnet und analysiert, ob diese geringer als der angegebene Radius ist. So kann die kleinstmögliche Distanz berechnet werden. Es wird über das Element die Tube bestimmt und für den Aktivierungsschritt zur Verfügung gestellt. Ist jedoch keine Tube vorhanden bei der der Punkt innerhalb des Radius liegt, wird keine Struktur selektiert. Diese Vorgehensweise wurde in der neu

implementierten Klasse `mitkTubeGraphPicker` realisiert.

### Graphentraversierung

Für die Interaktion auf der Datenstruktur ist es notwendig mittels geeigneter Algorithmen den gesamten Graphen zu durchsuchen. Um hierbei systematisch vorzugehen gibt es mehrere Verfahren, die von einem Startknoten ausgehend und alle erreichbaren Knoten über die gegebenen Kanten traversieren. Die Grundlage all dieser Verfahren ist ein *Markierungsalgorithmus*. Hierbei werden alle bereits besuchten Knoten und Kanten markiert. Da nur unmarkierte Knoten im Verlauf besucht werden, können Schleifen im Algorithmus ausgeschlossen werden. Die Verfahren laufen so lange bis keine unmarkierten Knoten von dem Startknoten aus erreicht werden können. Somit wird der Algorithmus für endliche Graphen auf endliche Schritte terminiert [36].

Die Suchverfahren unterscheiden sich in der Art und Weise, wie der als nächstes zu besuchende Knoten gewählt wird. Die zwei wesentlichen Varianten sind hierbei die *Breitensuche* und die *Tiefensuche*. Wichtig für die Datenstruktur ist, dass beide Verfahren auf ungerichtete, sowie gerichtete Graphen anwendbar sind.

**Hinzufügen einer Orientierung** Die Breitensuche wird in dieser Arbeit dazu verwendet, dem ungerichteten Graphen eine Orientierung zu geben. Eine Umwandlung des ungerichteten Graphens wird dazu benötigt, den Begriff der Peripherie zu definieren.

Bei der Breitensuche (engl. *Breadth-First-Search*) wird, versucht die Suche möglichst breit anzulegen. Hierbei wird, ausgehend von dem Startknoten, zunächst alle Nachbarn besucht und markiert. Dann wird für alle Knoten immer zunächst der am längsten markierte Knoten betrachtet und auch hier werden alle Nachbarknoten in diese Markierungsfolge aufgenommen [36].

Bei dieser Arbeit wird die von der BGL zur Verfügung gestellte Breitensuche verwendet. Diese `boost::breadth_first_search` Funktion benötigt den zu durchsuchenden Graphen, den Startknoten und ein Objekt der Klasse `default_bfs_visitor` [32]. Dieser Visitor dient dem Benutzer dazu bestimmte Aktionen an unterschiedlichen Ereignisse der Suche durchzuführen. Diese Einstiegspunkte können zum Beispiel vor jeder Betrachtung eines neuen Knotens sein, nach der Verarbeitungen einer Kante oder eines Knotens. Um diese Funktion zu verwenden sind eigene Visitor-Klassen implementiert worden.

Um den ungerichteten Graphen in einen gerichteten Graphen umzuwandeln, wird bei jeder besuchten Kante in der Breitensuche, in einem neuen orientierten Graphen diese Kante mit der abgehenden Richtung abgespeichert. Das Ereignis bei der die Kante hinzugefügt wird, wird als `tree_edge` bezeichnet.

```
class mitkDirectedGraphBfsVisitor: public boost::default_bfs_visitor
{
 void tree_edge(mitk::TubeGraph::EdgeDescriptorType edge,
 const mitk::TubeGraph::GraphType& undirectedGraph)
```

```
 {
  //Get source and target from the discovered edge
  unsigned int numberSource = boost::source(edge, undirectedGraph);
  unsigned int numberTarget = boost::target(edge, undirectedGraph);

  //Generate new vertex descriptor and
  //add them to the new directed graph
  boost::graph_traits<DirectedGraphType>::vertex_descriptor source = numberSource;
  boost::graph_traits<DirectedGraphType>::vertex_descriptor target = numberTarget;

//Add oriented edge to the new directed graph
  boost::add_edge(source, target, directedGraph);
 }
}
```

Als Ergebnis sind alle Knoten und Kanten, die ausgehend von einem Startknoten, zu erreichen sind in einem gerichteten Graphen gespeichert. Das heißt alle Kanten werden abgehend von dem gewählten Startknoten gerichtet.

**Exploration des ungerichteten Graphen** Um ein Weg zwischen zwei Kanten zu finden, werden ausgehend von einem Startknoten zunächst alle erreichbaren Wege aufgezeichnet. In einem zweiten Schritt werden dann die gesuchten Wegen zwischen zwei Kanten daraus extrahiert und weiterverwendet.

Für die Exploration aller Wege von einem Startknoten aus, wird in dieser Arbeit die Tiefensuche verwendet.

Im Gegensatz zur Breitensuche wird bei der Tiefensuche (engl. *Depth-First-Search*) versucht, den am weitesten von dem Startknoten entfernten Knoten, so früh wie möglich zu traversieren. Um möglichst tief in den Graphen zu gehen, wird die Suche immer bei dem zuletzt markiert Knoten fortgesetzt. Bei der Realisierung bietet sich hierbei auch eine rekursive Implementierung an, da der gerade besuchte Knoten ja zur weiteren Suche verwendet wird.

Auch hier wird auf die Algorithmen der BGL zurück gegriffen. Hierbei wird die Funktion `boost::undirected_dfs` angewendet. Neben dem ungerichteten Graphen und dem Startknoten wird ein Standardvisitor übergeben, der den Suchweg mitschreibt. Zu dem werden zwei Maps mit gegeben, die die Markierungsfarbe der einzelnen Kanten und Knoten verwalten. Bei den Suchalgorithmen der BGL werden alle noch nicht besuchten Kanten und Knoten mit weiß gekennzeichnet und alle die noch in der Warteliste stehen mit grau. Die vollständig abgearbeiteten Kanten und Knoten werden mit schwarz markiert.[32]

### Aktivierungsmodi

Die Aktivierungsmodi können direkt von dem Benutzer gewählt werden. Hierüber wird festgelegt ob nur die einzelne, selektierte Tube aktiviert wird oder ob mehrere Tubes

aktiviert werden sollen. Neben dem Modus „None“, also der ausgeschalteten Aktivierung, kann der Benutzer zwischen fünf weiteren Modi wählen, auf die im Folgenden einzeln eingegangen wird. Zudem besteht die Möglichkeit jederzeit alle aktivierten Tubes zu deselektieren.

Für manche Aktivierungsmodi ist die Definition eines Wurzelknotens notwendig. Dieser Begriff wird allerdings innerhalb der Graphenstruktur nicht definiert. Vielmehr stellt er eine Besonderheit der Baumstruktur dar und man muss innerhalb des Graphen einen Knoten gesondert festlegen. Dem Benutzer steht es daher frei, selbst einen Knoten zu wählen, oder auch den berechneten Knoten der innerhalb des dicksten Gefäß am nächstliegend ist, als Wurzelknoten beizubehalten.

**Single** In diesem Modus wird nur die vorher selektierte Tube aktiviert. Wird bei dem Picking keine gültige ID zurück gegeben, wird eine bisher bestehende Aktivierung auf eine Tube aufgelöst.

**Multiple** In deiesem Modus können beliebig viele Gefäße aktiviert werden. Jede neu selektierte Tube wird zu den bereits aktivierten Tubes hinzugenommen. Eine fehlerhafte ID als Rückgabewert des Pickers wird jedoch ignoriert, da bei einem Fehlklick, die bereits aktivierten Tubes nochmals markiert werden müssen.

Wird eine Tube fälschlicherweise markiert, kann der Benutzer diese durch erneutes anklicken, deaktivieren. Der Benutzer kann auch zu jederzeit die gesamte Aktivierung aufheben.

**Path to Root** Dieser Modus kann nur auf einer Baumstruktur angewendet werden, da, wie bereits erwähnt, der Begriff der Wurzel bei Graphen nicht existiert. Jedoch ist es im medizinischen Kontext oft der Fall, dass ein Knoten im Graphen als Anfangspunkt für den Gefäß- oder Bronchialbaum betrachtet werden kann. Somit wird in der Klasse `mitkTubeGraph` ein solcher Vermerk auf die Wurzel abgelegt und kann für diesen Modus abgerufen werden. Auf Grundlage dessen kann der Weg zwischen der selektierten Tube und dem Wurzelknoten mit der beschriebenen Tiefensuche berechnet werden (siehe Abschnitt 6.3.1: Graphentraversierung). Gibt es keine Verbindung zwischen der Wurzel und dem Gefäß, wird nur die selektierte Tube aktiviert.

**Periphery** Auch für diesen Modus ist ein Wurzelknoten notwendig, da nur so der Begriff der Peripherie zustande kommt. Mittels der Breitensuche kann aus dem ungerichteten Graph ein orientierter Graph erzeugt werden. Hierbei geht man von der definierten Wurzel ausgehend mit einer Breitensuche alle Kanten entlang und speichert diese, von der Wurzel weggehend, gerichtet in einem neuen Graphen zwischen (siehe Abschnitt 6.3.1: Graphentraversierung). Somit hat man eine Flussrichtung in das Modell eingebracht und kann alle abgehenden Tubes der selektierten Tube aktivieren. Gibt es keine Verbindung zwischen der Wurzel und dem Gefäß, kann auch keine Richtung angegeben werden, und es wird nur die selektierte Tube aktiviert.

**Point to Point** Ist dieser Modus aktiv, so werden zwischen der zuletzt selektierten Tube und der neu selektierten Tube alle Wege über eine Tiefensuche bestimmt und aktiviert (siehe Abschnitt 6.3.1: Graphentraversierung). Sind die beiden Tubes nicht über mindestens einem Weg miteinander verbunden, so wird nur die neu selektierte Tube aktiviert.

#### Zustandsmaschine

Für die Interaktion mit den Graphen wurde eine Zustandsmaschine entworfen, die auf dem Interaktionskonzept von MITK (siehe 6.1.1) beruht. Bekommt die entworfene Zustandsmaschine eine Eingabe, für die ein Übergang von ihrem aktuellen zu einem neuen Zustand definiert ist, so nimmt sie die Eingabe an, wechselt in den neuen Zustand und führt eine Reihe von Aktionen aus, die dem Zustandsübergang zugeordnet sind. Diese Aktionen werden von der selbst entworfenen Interaktionsklasse `mitkTubeGraphInteractor` ausgeführt. Je nach Aktivierungsmodus wird darauf reagiert. Die Auswirkungen der einzelnen Aktivierungsmodi mit dieser Selektion wurden im vorangehendem Abschnitt näher erläutert.

Wie in Abbildung 6.2 dargestellt ist, reagiert der Startzustand nur auf einen Klick des Benutzers auf eine Tube. Erfolgt ein Doppelklick oder das Anwählen durch das Halten der Steuerung-Taste in Kombination mit Drücken der linken Maustaste im dreidimensionalen Fenster der Anwendung, wird versucht eine Tube zu picken. Im Zustand "Tube Selektion"wird auf die Antwort des `mitkTubeGraphPicker`s gewartet, ob eine Tube getroffen wurde. Ist dies der Fall wird automatisch das Bestätigungsereignis gesendet und die Tube wird selektiert und es erfolgt der Übergang in den Zustand "Tube selektiert". Ist dies nicht der Fall und es wurde keine Tube in der Nähe gefunden, so wird alles zurückgesetzt und wieder den Startzustand erreicht.

Auch das mehrfache Selektieren unterschiedlicher Tubes ist möglich. Hier wird ähnlich vorgegangen. Es wird auf die Benutzereingabe gewartet und geprüft, ob eine Tube getroffen wurde, wenn ja wird diese selektiert. Übergeleitet wird bei beiden Fällen in den Zustand "Tube selektiert".

Alle weiteren Reaktionen auf Interaktionen, welche die tubulären Strukturen betreffen, sind in der `mitkTubeGraphInteractor` Klasse entwickelt worden.

### 6.3.2. Modifikation

**Hinzufügen** in dieser Arbeit wurde das Hinzufügen von neuen tubulären Strukturen sind in dieser Arbeit bereits Verarbeitungsschritte realisiert. So können neue Kanten und Knoten in der Datenstruktur angelegt werden. Über den `mitkTubeGrapPicker` wird auch nicht nur die selektierte Tube zurück gegeben, sondern auch das Elemen ermittelt, dass am nächsten von dem Benutzer gewählten Punkt liegt. Somit ist das Verbinden zweier Tubes und dem daraus folgenden Aufteilen einer Kante in ein neues Verbindungsstück bereits gegeben. Da das Konzept für das Bilden von neueen Strukturen „from the scratch“ in MITK fehlt, wurde die Funktionalität im Rahmen dieser Arbeit nicht umgesetzt.

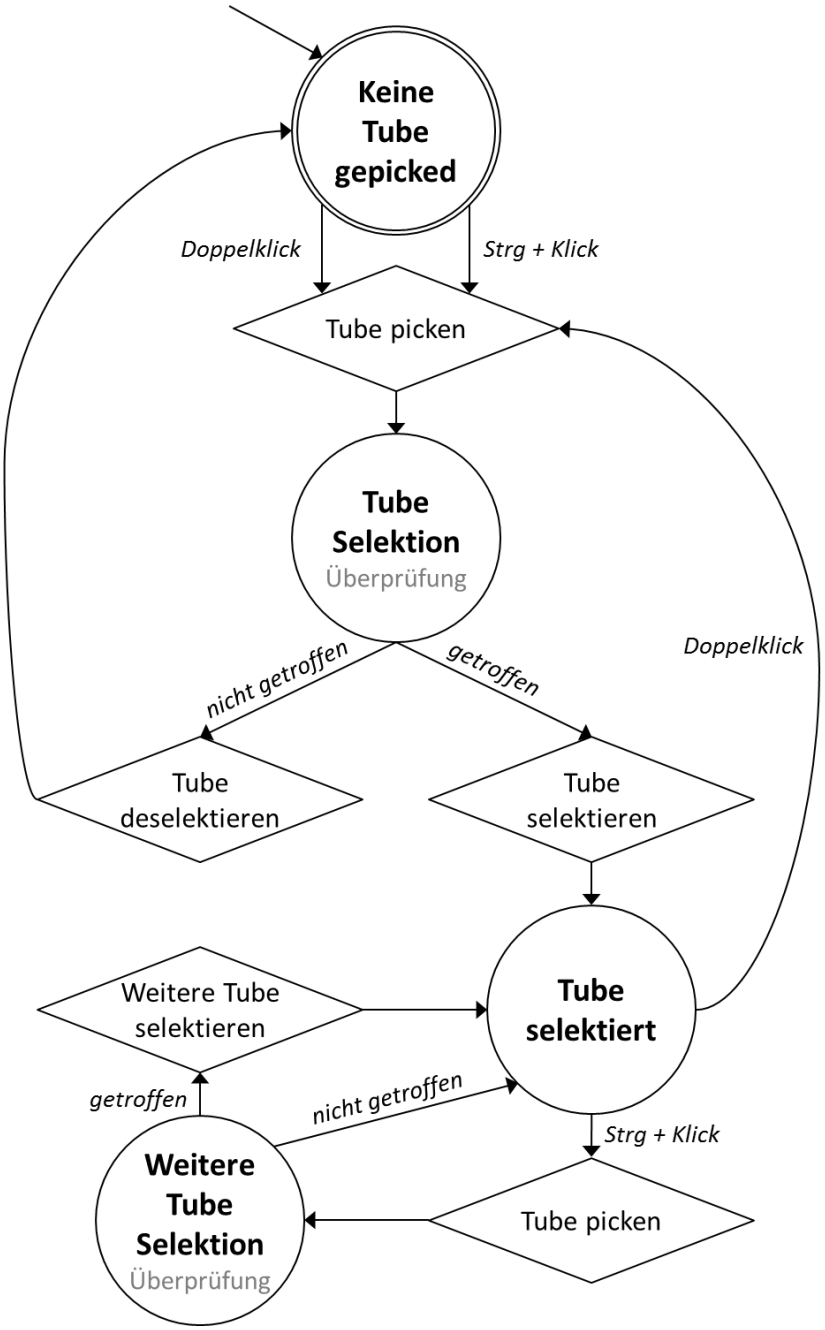


Abbildung 6.2.: Zustandsmaschine für die Selektion: Ausgehend von dem Startzustand (doppelter Kreis) wird auf Benutzereingaben reagiert. Diese Ereignisse (kursive Schrift) und führen über in ein anderen Zustand (Kreis). Hierbei können Aktionen (Rauten) angestoßen werden.

**Entfernen** Über die beschriebenen Selektionsverfahren kann der Benutzer einzelne Tubes oder komplette Gefäßbaumabschnitte aktivieren und danach löschen. Dies geschieht über die von der in dieser Arbeit entwickelten Klasse `mitkUndirectedGraph` bereitgestellten Löschmethoden für Kanten und Knoten. Ein Knoten, der mit einer verbleibenden Kante verbunden ist, bleibt dabei weiterhin bestehen. Mit dieser Funktionalität können fehlerhafte Fragmente der Segmentierung oder Abschnitte gelöscht werden, die nicht von Interesse sind. Das Löschen bedeutet, dass die Tubes nicht nur visuell ausgeblendet sind. Vielmehr werden sie permanent aus dem Graphen gelöscht. Um ein versehentliches Entfernen zu vermeiden, wird der Benutzer in einem Dialog um eine Bestätigung gebeten.

**Extrahieren** Der Benutzer hat die Möglichkeit einen Teilgraphen aus den im Originalgraphen aktivierten Tubes zu extrahieren. Hierbei bleibt der Originalgraph unverändert, d.h. die Teilstruktur wird nicht gelöscht. Dafür wird ein neues Objekt von `mitkTubeGraph` erzeugt, der nur die aktivierten Strukturen abbildet. Bei der Erzeugung werden in diesem neuen Graph Kanten und Knoten angelegt und mit den Informationen aus den `mitkTubeElement` Objekten des Originalgraphen den genauen Verlauf kopiert. Da die BGL bei der Neuerzeugung von Knoten diesen inkrementell Deskriptoren zuteilt und normalerweise weniger Kanten und Knoten in diesem neuen Teilgraphen vorliegen, kommt es zu Differenzen bei den Knotendeskriptoren und somit auch bei den Kantendeskriptoren. Diesem Verhalten wird vorgebeugt, indem in einer Zuweisungsliste die neuen Deskriptoren den alten zugeordnet werden. Anhand dieser Liste können dann auch die Kanten zwischen den zugehörigen Knoten im neuen Teilgraphen erstellt werden. Eine korrekte Darstellung ist somit gesichert.

### 6.3.3. Attributierung

Das Attributieren ist eine Möglichkeit den tubulären Strukturen anatomische und funktionelle Eigenschaften zuzuordnen. Das Ablegen dieser Zuweisungen und die damit verbundene visuellen Informationen sind gesondert zu der Datenstruktur in der `mitkTubeGraphProperty` Klasse realisiert worden. Diese zentrale Klasse wird dem `mitkTubeGraph` direkt zugeordnet und auch persistent gespeichert (siehe Abschnitt 4.3.4). Damit wird gewährleistet das Änderungen, die der Benutzer vorgenommen hat auch nach einem Speichern der tubulären Struktur erhalten bleiben.

#### Property Klasse

Die `mitkTubeGraphProperty` Klasse enthält zusätzliche Informationen zu den tubulären Strukturen. Neben dem Verwalten der aktivierten Tubes, die bei der Visualisierung gesondert eingefärbt werden, speichert sie die Informationen über Einteilung der Gefäße in Gruppierungen. Diese Merkmalsgruppen (engl. *Label Group*) haben unterschiedliche Ausprägungen (engl. *Label*). Diese wiederum zeichnen sich vor allem bei der Visualisierung durch unterschiedliche Einfärbungen aus. Zudem können sie ein- und ausgeblendet werden.

Da einer Tube mehrere Zuordnungen zugewiesen werden können, muss hier eine Reihenfolge definiert werden, wie diese bei der Visualisierung dargestellt werden. Zunächst werden aber die zentralen Bestandteile, die Merkmalsgruppen- und ausprägungen, dieser Klasse erläutert.

**LabelGroup** Eine Festlegung der Merkmalsgruppen `LabelGroup` wird in der Property Klasse über den Datentyp „struct“ definiert. Diese beinhaltet neben der Gruppenbezeichnung, eine Liste mit allen Ausprägungen (`Label`), sowie die Information, ob diese Zuweisung bei der Visualisierung berücksichtigt werden soll. Letzteres spielt eine wichtige Rolle, wenn einer Tube Ausprägungen unterschiedlicher Gruppen zugewiesen wird und über eine Reihenfolge entschieden werden soll, welche Information letztendlich verwendet werden soll.

Jede neu erzeugte `LabelGroup` besitzt standardmäßig die Ausprägung „Undefined“, die allen nicht näher bezeichneten Tubes zugeordnet wird.

**Label** Ein `Label` ist eine Merkmalsausprägung und muss einer LabelGroup zugewiesen sein. Neben der Bezeichnung der Ausprägung, wird die vom Benutzer zugewiesene Farbe, in der die Tubes mit dieser Ausprägung eingefärbt werden sollen, abgespeichert. Zudem wird hier vermerkt, ob der Benutzer die Gefäße mit diesem Label darstellen möchte. Mit dieser Sichtbarkeitseinstellungen kann der Benutzer kurzzeitig bestimmte Strukturen ausblenden. Eine Tube kann nur einem Label einer Gruppe zugewiesen werden. Jedoch kann es aus auch eine weitere Ausprägung einer anderen Gruppierung besitzen.

Um die Attributierung für künftige Anwendungsfälle offen zu halten, wird es dem Benutzer über die Benutzeroberfläche ermöglicht, eigene Merkmalsgruppen mit ihrem Ausprägungen zu definieren. Zudem sind Standardgruppierung, wie zum Beispiel die Einteilung der Lebergefäße in die Segmenteinteilung der Leber (siehe Abschnitt 2.1.1) oder dem Gefäßtyp (Arterie, Vene, Hohlvene, Gallengang), sowie dem Resektionsstatus, hinterlegt.

Da eine Tube mehreren Gruppierungen zugeordnet werden kann, ist in dieser Klasse eine Priorisierung definiert, welche Informationen zur Visualisierung Vorrang haben. Hierbei werden zunächst alle aktivierten Tubes in einer Farbe dargestellt. Ist eine Tube nicht aktiviert, besitzt aber Zuordnungen zu Labels unterschiedlicher Gruppen, so wird die Farbinformation des Labels verwendet, deren LabelGroup in der verwalteten Liste an höherer Stelle steht. Außer das Label ist hierbei auf „Undefined“ gesetzt oder die Gefäße des Labels sind ausgeblendet. Das bedeutet, dass die Information des Labels der nächsten Gruppe ausschlaggebend wird. Ist die Tube weder einem bestimmten Label zugeordnet, noch ist sie aktiviert, so wird diese in der Standardfarbe präsentiert.

### 6.3.4. Annotation

Bei Auffälligkeiten oder bei besonderen Stellen in einer Verzweigungsstruktur von tubulären Strukturen kann es notwendig sein, Vermerke zu hinterlegen. Umgesetzt wurde die Funktionalität mit einer einfachen Liste, die die Bezeichnung, die Lokalisation, also die Tube, sowie besondere Bemerkungen verwaltet. Dem Benutzer wird bei dieser Art

der Interaktion nur der Aktivierungsmodus zur Verfügung gestellt, bei dem er einzelne Tubes aktivieren kann. Diese aktivierte Tube wird für die Lokalisation verwendet. Über ein Benutzereingabefeld können dann die Informationen über Bezeichnung und weitere Bemerkungen eingetragen werden.

Die vermerkten Annotationen werden dem Benutzer in einer Liste bereitgestellt. Eine Visualisierung der Annotationen im dreidimensionalen Fenster, also direkt an der Struktur, ist in der Zukunft geplant.

### 6.3.5. Information

Eine weitere Anforderung an diese Interaktionskomponente ist die Bereitstellung von Informationen. Zum einen werden dem Benutzer zu jeder Zeit Informationen über die aktivierten Strukturen angezeigt. Zum anderen kann sich der Benutzer auch Informationen über den gesamten Graphen ausgeben lassen.

Für die Informationen über die aktivierten Strukturen muss ein Signal an die Ausgabe gesendet werden, wenn sich diese ändern. Dieses Signal wird abgefangen und die Informationen aktualisiert. Auch die Änderungen des gesamten Graphen bewirken eine konsistente Anpassung der Informationen über die Datenstruktur im Allgemeinen. Diese Änderungen können über die Modifikationskomponente (siehe Abschnitt 6.3.2) vorgenommen werden.

Exemplarisch werden in dieser Arbeit die Anzahl der aktivierten Tubes, die momentane Wurzel Tube, die Anzahl der gesamten Kanten und Knoten, sowie Informationen über die Bildgeometrie angezeigt. Die Anzeige von weitere Informationen können mit Hilfe des implementierten Mechanismus jederzeit umgesetzt werden.

### 6.3.6. Klassenübersicht

In dem in Abbildung 6.3 dargestellten Sequenzdiagramm soll mit Hilfe des Picking-Prozesses (siehe Abschnitt 6.3.1), die Interaktion der Datenklassen, der Visualisierungsklasse und der Interaktionsklassen aufgezeigt werden.

Neben den in Abschnitt 4.3.5 und Abschnitt 5.3.5 beschriebenen Klassen des Datenmodells und der Visualisierung, sind bei der Interaktion die Property-Klasse `mitkTubeGraphProperty` und der `mitkTubeGraphInteractor`, die zentralen Klassen. Der Interactor reagiert auf die Benutzereingabe und führt hierzu Aktionen aus. Fügt der Benutzer über die genannten Interaktionskomponenten weitere Informationen über einzelne Gefäße hinzu (z.B. Merkmalsausprägungen, Annotationen,...), werden diese in der Property-Klasse verwaltet. Zudem spielt der `mitkTubeGraphPicker` eine wichtige Rolle, da dieser über das dargestellte Verfahren prüft, ob ein Gefäß von dem Benutzer ausgewählt wurde. Nur über die Selektion und der anschließenden Aktivierung kann der Benutzer die realisierten Interaktionen nutzen.

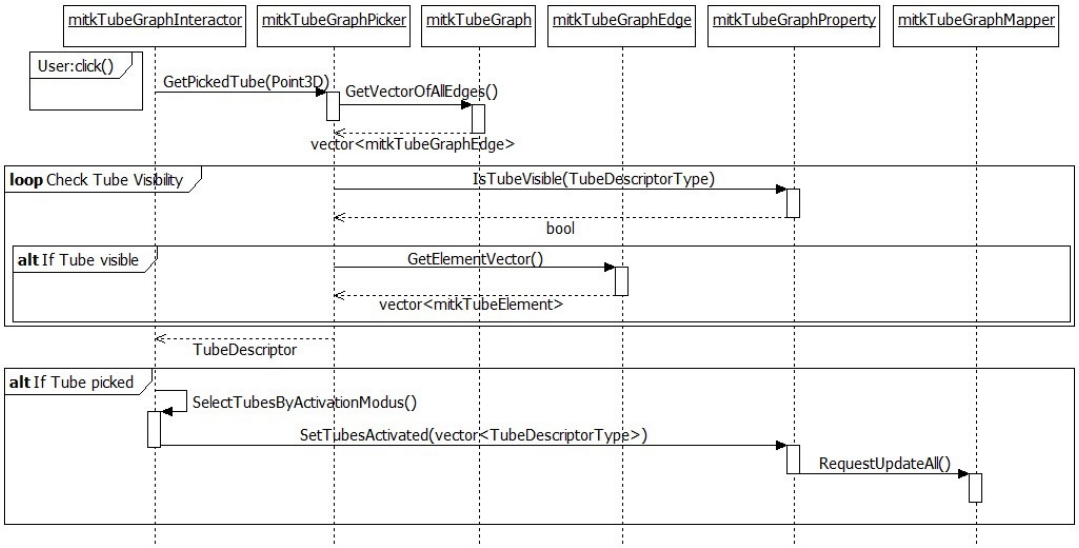


Abbildung 6.3.: Vereinfachtes Sequenzdiagramm des Pickings

## 6.4. Ergebnisse

Die Anforderungen an die Interaktionskomponenten sind in einer grafischen Benutzeroberfläche umgesetzt worden. Diese Oberfläche besteht aus mehreren Bestandteilen, die in der Folge beschrieben werden.

Neben einem festen Bereich, der dem Benutzer jederzeit während der Interaktion auf den tubulären Systemen zur Verfügung steht, kann er weitere Interaktionskomponenten über Reiter auswählen (siehe Abbildung 6.9).

Über ein Auswahlfeld kann der Anwender den Graphen wählen, mit dem er interagieren möchte. Dies ist relevant, wenn mehrere tubuläre Systeme in einer Szene dargestellt werden. Um mit Gefäßen interagieren zu können, muss man einzelne oder komplette Strukturen auswählen. Dies kann der Benutzer über die Wahl eines Aktivierungsmodus und dem selektieren eines Gefäßes. Diese Aktivierungsmöglichkeiten braucht der Benutzer um Gefäße zu markieren auf denen er dann gewünschte Operationen anwenden kann (siehe Abbildung 6.4). Der Anwender kann über das selektieren der Radiobuttons einen der sechs Modi auswählen.
Ist der Modus „*None*“ gewählt, so werden, ungeachtet eines Klicks des Benutzers auf ein Gefäß, keine Strukturen selektiert.
Wird der Modus „*Single*“ verwendet, so wird nur ein Ast aktiviert. Wohingegen es bei dem Modus „*Multiple*“ dem Benutzer frei steht mehrere Strukturen hintereinander auszuwählen.
Ist man im Modus „*Periphery*“ bedeutet das, dass von dem selektierten Ast aus, alle Gefäße, die in die Periphery abgehen, aktiviert werden. Der Begriff der Periphery wird durch

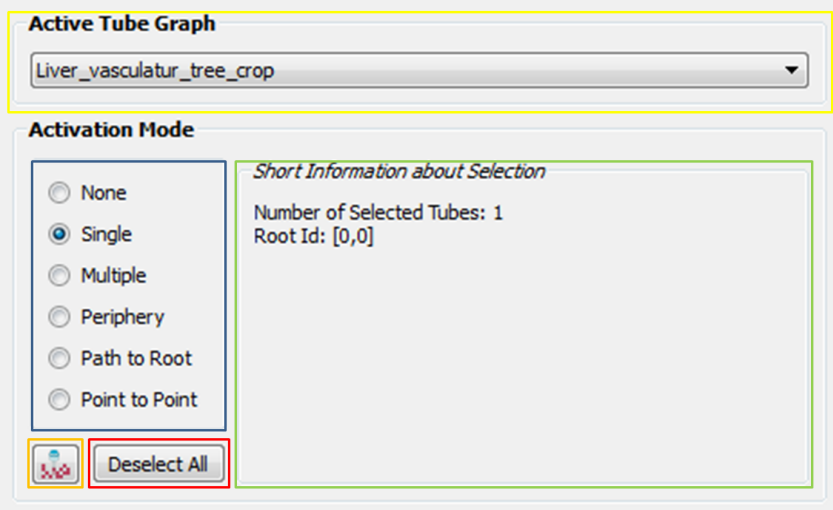


Abbildung 6.4.: Auswahl- und Aktivierungskomponente: In einem Auswahlfeld kann der Benutzer den Graph wählen, auf dem er interagieren möchte (gelb). Neben unterschiedlichen Aktivierungsmodi (blau) kann der Benutzer auch die Wurzel selbst festlegen (orange). Zudem kann er auch seine Aktivierung rückgängig machen (rot). In einem weiteren Feld (grün) werden Information über die aktivierten Strukturen zur Verfügung gestellt.

Bestimmen einer Wurzel definiert. Diese Wurzel kann von dem Benutzer selbstständig gesetzt werden oder es wird die Struktur mit dem dicksten Radius vordefinierte.
Der Modus „*Path to Root*“ ist das Gegenstück. Hier werden von der selektieren Struktur alle Wege zur Wurzel aktiviert.
Bei dem letzten Modus „*Point to Point*“ muss der Benutzer zwei Strukturen hintereinander selektieren. Es werden nun alle Wege die zwischen diesen beiden Punkten existieren aktiviert.
Alle Aktivierungen werden durch den Button „*Deselect All*“ rückgängig gemacht.

In Abbildung 6.5 werden diese Aktivierungsmodi an einem Ausschnitt eines Gefäßbaums beispielhaft demonstriert.
Der Anwender kann nun über Reiter die weiteren Interaktionen auswählen. Mit Hilfe des Attributierungsreiters (siehe Abbildung 6.6) steht dem Benutzer die Möglichkeit zur Verfügung, den aktivierten Strukturen anatomische und funktionellen Attribute zu zuweisen. Er kann hierbei jedem Graph vorgegebene oder selbst definierte Gruppen hinzufügen und wieder entfernen. Die Zuweisung erfolgt durch Auswählen des jeweiligen Attributs. Die Gefäße werden dann in der angezeigten Farbe eingefärbt. Sind solche Zuordnungen getroffen, können diese gruppierten Strukturen auch ausgeblendet werden.

Eine weitere Interaktion ist durch den Annotationsreiter (siehe Abbildung 6.7) gegeben. Hiermit kann der Benutzer einzelnen Gefäßen Annotationen beifügen. Diese Funktionalität ist vor allem im Austausch mit anderen Ärzten sinnvoll. So können wichtige Befunde mit einer Beschreibung abgelegt werden.

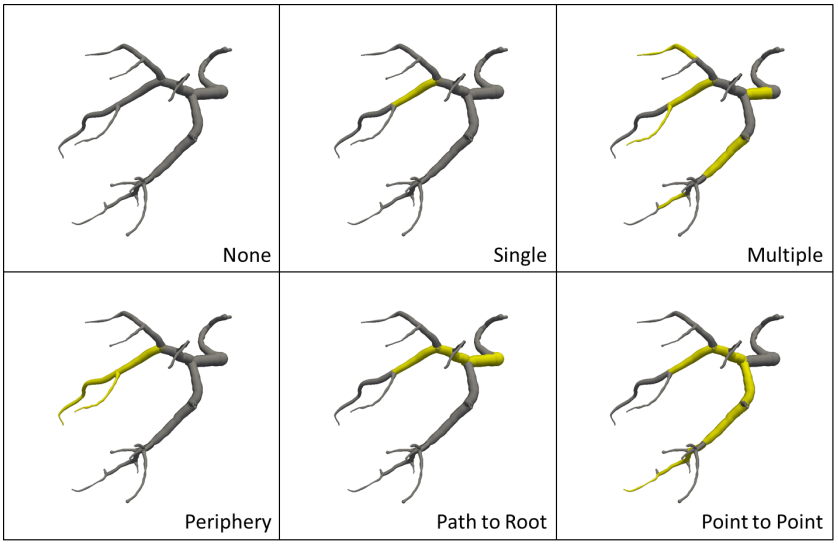


Abbildung 6.5.: Darstellung der Aktivierungsmodi: Darstellung der sechs Aktivierungsmodi anhand einem Teilgefäßbaum (grau). Hierbei ist immer das gleiche Gefäß selektiert. Anhand der unterschiedlichen Modi, werden die Gefäße dann aktiviert (gelb).

Um die Datenstruktur nachträglich zu verändern, kann der Benutzer den Reiter für die Modifikation anwählen. Wie in Abbildung 6.8 gezeigt, stehen dem Benutzer drei Möglichkeiten zur Verfügung. Für die erste Möglichkeit, dem Verbinden von vorhandenen Strukturen, sind bisher jedoch nur die Voraussetzungen geschaffen worden. Die Realisierung erfolgt in zukünftigen Projekten. Die Möglichkeit bestimmte Gefäßabschnitte zu extrahieren ist realisiert. Hierfür muss der Arzt den Abschnitt des Graphen über die Aktivierungsmodi aktivieren und durch drücken den Buttons, wird ein neuer Graph mit diesen Strukturen gebildet. Nun steht es dem Benutzer frei auf diesem Teilgraphen weitere Aktionen vorzunehmen. Des Weiteren besteht die Option Abschnitte aus der Datenstruktur zu Löschen.

Der letzte Reiter stellt dem Anwender Informationen über das gesamte tubuläre System zur Verfügung.

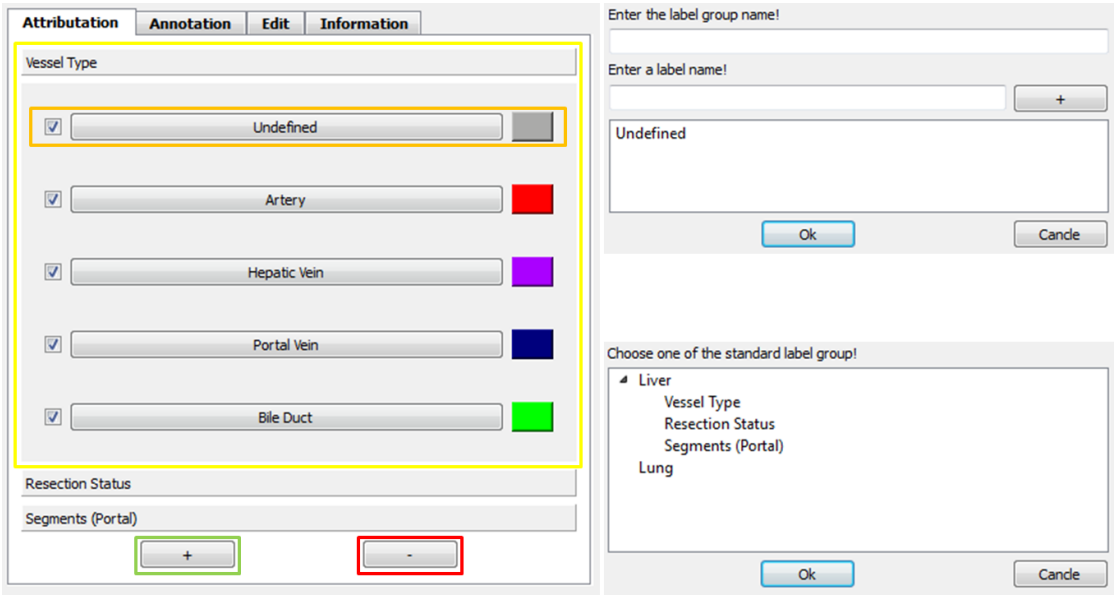


Abbildung 6.6.: Attributierungskomponente: Frei konfigurierbare Zuordnung von anatomischen und funktionellen Attributen zu den tubulären Strukturen. Durch Hinzufügen (grün) von Attributsgruppen (gelb) kann der Benutzer den aktivierten Strukturen Attribute (orange) zuweisen. Hierbei kann er die Farbe und die Sichtbarkeit dieser gruppierten Strukturen steuern. Bestehende Gruppen kann der Anwender wieder entfernen (rot). Zudem sind die Dialogfenster für das Hinzufügen einer Merkmalsgruppe dargestellt.

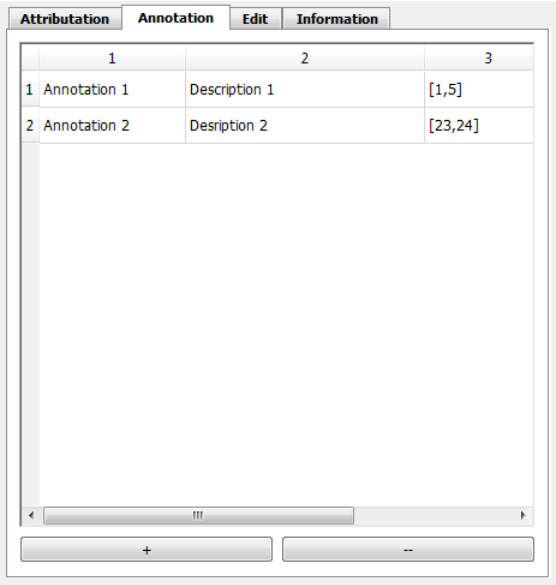


Abbildung 6.7.: Annotationskomponente: Der Benutzer kann einer aktivierten Tube eine Bezeichnung und eine ausführliche Beschreibung hinzufügen. Erstellte Annotationen können auch wieder entfernt werden.


Abbildung 6.8.: Modifikationskomponente: Dem Benutzer wird die Möglichkeiten zur Verfügung gestellt Strukturen hinzuzufügen (grün), zu separieren (gelb) oder zu entfernen (rot).

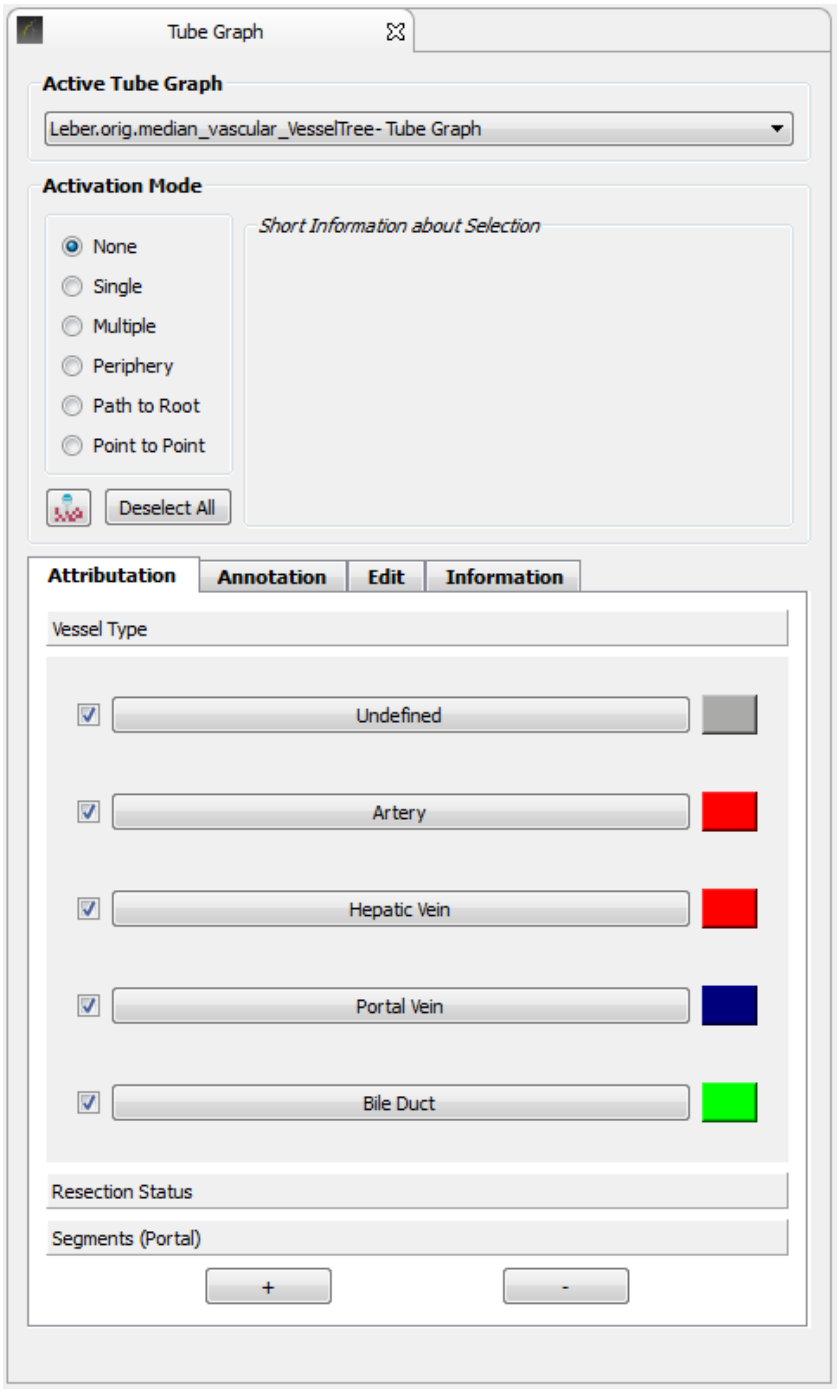


Abbildung 6.9.: Realisierung der Interaktionskomponenten in einer grafischen Benutzeroberfläche

# 7. Diskussion und Ausblick

In dieser Arbeit wurde ein generisches Datenmodell zur symbolischen Beschreibung von tubulären Strukturen realisiert. Zudem wurde ein entsprechendes, modellbasiertes Visualisierungsverfahren eingeführt. Auf Grundlage dieser Implementierungen wurden mehrere Interaktionskomponenten konzipiert und realisiert, sowie in einer grafischen Benutzeroberfläche umgesetzt.

### Datenmodell

Das Datenmodell ist sehr modular gehalten und somit leicht für unterschiedliche Anwendungsfälle einsetzbar. Durch den Einsatz von Templates und abstrakten Klassen ist neben der Wiederverwendbarkeit auch eine eine gute Wartbarkeit der Implementierung gewährleistet. Zudem ist sie für zukünftige Erweiterungen offen.

Der Einsatz der externen Softwarebibliothek Boost Graph Library unterstützt dies, da sie neben der grundlegenden Datenstruktur für Graphen auch viele templatisierte Algorithmen zur Graphenverarbeitung bietet. Zudem wird sie aktiv gepflegt und kontinuierlich weiterentwickelt.

Die neue Datenstruktur, basierend auf einem ungerichteten Graph, erlaubt nun die Darstellung von Zyklen, die durch Anastomosen zustande kommen. Mit dem neuen Datenmodell können nun alle Gefäßformen abgebildet werden, so dass keine Beschränkung mehr auf kreisrunde Querschnitte besteht. Ohne diese Einschränkung können nun in zukünftigen Projekte fortgeschrittene Verfahren zu Skelettierung und Graphenerzeugung realisiert werden. Diese Verfahren können dann Informationen über den tatsächlichen Gqfäßquerschnitt aus der Segmentierung extrahieren und in das Datenmodell einfließen lassen.

Der erste Schritt bei der Ergänzung und Ausweitung des geschaffenen Tools für tubuläre Strukturen innerhalb von MITK wird also die Anpassung des vorhandenen Skelettierungsverfahrens an die neu geschaffenen Möglichkeiten sein. Hierfür sollte ein Einsatz und die Integration des VMTK bedacht werden. Das Toolkit bietet vielseitige Möglichkeiten im Bereich von mittellinienbasierten Berechnungen und Darstellungen.

### Visualisierung

Die Visualisierung der tubulären Strukturen ist, gemäß den Anforderungen an diese Arbeit, momentan nur für die Darstellung von kreisrunder Gefäßquerschnitte realisiert. Aufgrund des vereinfacht abgebildeten Querschnittes ist die Visualisierung primär nicht zur Diagnostik von gefäßformveränderten Krankheiten, wie zum Beispiel Stenosen oder

Ablagerungen von Plaque, geeignet. Um hier exakte Messparameter zu erlangen, muss die genaue Form der Gefäße dargestellt werden. Die Lagebeurteilung der Gefäße zum Beispiel in Relation zu einem Tumor oder die Berechnung von Versorgungsgebieten ist jedoch möglich. Da das Verfahren auf einfachen Primitiven beruht, kann durch die Wahl verschiedenartiger geometrischer Formen, einzelne Gefäßabschnitte individuell modelliert werden.

Ein großer Vorteil des hier präsentierten Verfahrens ist das Entfernen der Fragmente im Inneren der Strukturen. Nun kann auch durch die tubulären Systeme hindurch navigiert werden. Das findet vor allem in der navigierten Bronchoskopie Anwendung. Durch künftige Optimierung des Verfahres können kleinere Fragmente im Inneren mit geschnitten und Löcher vermieden werden, um so ein bessere Darstellungsqualität zu erreichen. Hierbei ist jedoch mit einer erhöhten Laufzeit zu rechnen.

Zudem ist das Verfahren der Visualisierung in strukturierten Einzelschritten unterteilt. Dies ermöglicht es, Berechnungen nur bei Bedarf durchzuführen. So werden zum Beispiel die Strukturen bei visuellen Änderungen, wie Farbe und Sichtbarkeit, nicht jedes Mal neu erzeugt. Nur wenn sich an der Datenstruktur selbst etwas ändert wird dieser Schritt bearbeitet. Da die Strukturen auch unabhängig voneinander erzeugt werden, ist das Verfahren auch parallelisierbar, so dass hier bei Bedarf Zeit eingespart werden kann.

Weitere Ansätze, die die verbesserte Darstellung der tubulären Strukturen betreffen, werden in [17] gegeben. Hier wird zum Beispiel eine multiple dreidimensionale Ansicht vorgeschlagen. Dabei soll dem Benutzer zur dreidimensionalen Szene auch ein korrelierendes Übersichtfenster, der den Graphen in verkleinerter Ansicht darstellt, als Orientierungspunkt angeboten werden. So weiß der Benutzer trotz Überdeckungen, die durch eine Rotation der dreidimensionalen Szene entstehen können, immer an welche Stelle im Gefäßgraph er sich befindet. Ein weiterer Ansatz wäre die komplexe Ansicht der tubulären Strukturen mittels Hervorhebungstechniken und einer Abschwächung von Teilbereichen zu vereinfachen.

### Interaktionen

Die in der grafischen Benutzeroberfläche realisierten Interaktionen bieten dem Anwender Möglichkeiten weitere Informationen über die tubulären Strukturen in das Modell mit einzubringen. Dabei ist es möglich die Strukturen interaktiv in anatomischen und funktionellen Gruppierungen zu unterteilen. Der Benutzer kann hierfür neue Gruppen anlegen oder vordefinierte Gruppen verwenden. So kann zum Beispiel für die Leberoperationsplanung eine Trennungen der Gefäße nach ihrem Gefäßtyp (Arterie, Vene,. . . ) oder nach der Zugehörigkeit zu Lebersegmenten vorgenommen werden. Die eingeteilten Gefäße werden dann farblich gekennzeichnet und können bei Bedarf ausgeblendet werden. Des Weiteren kann der Arzt über Annotationen wichtige Stellen in der Gefäßstruktur markieren.

Eine solche Unterteilung ist nur über Selektionsverfahren möglich. Hierbei bietet die grafische Oberfläche sechs unterschiedliche Aktivierungsmodi. Je nach ausgewähltem Mo-

dus werden einzelne Gefäßkanten, Teilbäume oder beliebige Gefäßgruppierungen aktiviert und für die Interaktionstechniken bereitgestellt. Die angebotenen Selektionsmechanismen decken die Exploration eines tubulären Systems sehr gut ab.

Das interaktive Löschen und Extrahieren von einzelnen Strukturen und Teilgraphen wird dem Benutzer in einer weiteren Interaktionskomponente zur Verfügung gestellt. Das bietet dem Mediziner, zum Beispiel bei einer Leberoperationsplanung, die Möglichkeit zu resezierende Gefäße auch in dem Datenmodell zu Löschen oder zusammenhängende Gefäßsysteme zu trennen. Das Hinzufügen von Strukturen ist in dieser Arbeit nicht realisiert worden, ist jedoch für die Zukunft eine sinnvolle Erweiterung. Da durch fehlerhaften Segmentierungen Verbindungen zwischen zwei Gefäßen nachträglich hinzugefügt werden können. Aber auch das Bilden von komplett neuen Strukturen wäre beispielsweise bei einer präoperativen Planung von Bypass-Operationen nutzbar.

Bei der fehlerhafte Bedienungen der Interaktionskomponenten sollte dem Benutzer die Rücknahme des Befehls möglich sein. Ein solches Undo/Redo-Konzept war kein Gegegenstand der Arbeit, würde aber die Benutzerfreundlichkeit erhöhen. Bisher wird der Anwender, bei nicht umkehrbaren Interaktionen, über ein Dialogfenster informiert und er kann die Aktion gegebenenfalls abbrechen.

Für eine Ausweitung der quantitativen Analysetechniken wie in [17] vorgeschlagen können, neben den bisher angezeigten Informationen der Datenstruktur, auch Radiendiagramme realisiert werden. Über ein solches Diagramm werden dem Benutzer der Radienverlauf der ausgewählten Strukturen visualisiert, um darauf die Vermessungen zu erleichtern. Zudem können weitere quantitative Parameter bei der Anzeige berücksichtigt werden. Das wäre zum Beispiel die Information über Verzweigungstypen, also ob es sich um eine Bifurkation oder Trifurkation handelt, da die Art der Verzweigung direkten Einfluss auf das versorgte Gewebe hat.

Die in dieser Arbeit realisierten Interaktionskomponenten können auf jede tubuläre Verzweigungsstruktur, die durch das Datenmodell abgebildet werden kann, angewendet werden. Sie ist variabel einsetzbar und somit für unterschiedliche, medizinische Anwendungsgebiete geeignet.

Die modulare Implementierung des Datenmodells und die flexibel anwendbaren Interaktionskomponenten erfüllen die Anforderungen von laufenden Projekte wie zum Beispiel der Leberoperationsplanung oder der navigierten Bronchoskopie. Aber sie bilden auch eine gute und zukunftsfähige Grundlage für neue medizinischen Fragestellungen.

# Literaturverzeichnis


[1] Barillot, C. ; Gibaud, B. ; Scarabin, J.-M. M. ; Coatrieux, J.-L. L.: 3d Reconstruction of Cerebral Blood Vessels. In: *IEEE Computer Graphics and Applications* 5 (1985), Nr. 12, S. 13–19

[2] Bloomenthal, J. ; Shoemake, K. : Convolution surfaces. In: *Computer Graphics* 25 (1991), Nr. 4, S. 251–256

[3] Bornik, A. ; Reitinger, B. ; Beichel, R. : Reconstruction and Representation of Tubular Structures using Simplex Meshes. In: *Proceedings of Winter School of Computer Graphics*, 2005, S. 61–65

[4] Catmull, E. ; Clark, J. : Recursively generated B-Spline Surfaces on Arbitrary Topological Meshes. In: *Computer-Aided Design* 10 (1978), Nr. 6, S. 350–355

[5] Charnoz, A. ; Agnus, V. ; Soler, L. : Portal Vein Registration for the Follow-Up of Hepatic Tumours. In: *Proceedings of MICCAI*, 2004, S. 878–886

[6] Charnoz, A. ; Agnus, V. ; Malandain, G. ; Soler, L. ; Tajine, M. : Tree matching applied to vascular system. In: *Proceedings of Graph-Based Representations in Pattern Recognition*. Berlin, Heidelberg : Springer-Verlag, 2005, S. 183–192

[7] Delingette, H. : General Object Reconstruction based on Simplex Meshes. In: *International Journal of Computer Vision* 32 (1999), Nr. 2, S. 111–146

[8] DKFZ Abteilung Medizinische und Biologische Informatik: *Forschungsprojekt der Abteilung: Leberoperationsplanung*. URL: `http://www.dkfz.de/de/mbi/projects/leber.html`, [Stand: 18. August 2012]

[9] DKFZ Abteilung Medizinische und Biologische Informatik: *Forschungsprojekt der Abteilung: Navigierte Bronchoskopie*. URL: `http://www.dkfz.de/de/mbi/projects/bronchoskopie.html`, [Stand: 18. August 2012]

[10] Ehricke, H.-H. ; Donner, K. ; Koller, W. ; Strasser, W. : Visualization of Vasculature from Volume Data. In: *Computers and Graphics* 18 (1994), Nr. 3, S. 395–406

[11] Felkel, P. ; Fuhrmann, A. ; Kanitsar, A. ; Wegenkittl, R. : Surface Reconstruction of the Branching Vessels for Augmented Reality Aided Surgery. In: *Proceedings of BIOSIGNAL* Bd. 16, 2002, S. 252 – 254

[12] Gergel, I. ; Wegner, I. ; Tetzlaff, R. ; Meinzer, H.-P. : Zweistufige Segmentierung des Tracheobronchialbaums mittels iterativen adaptiven Bereichswachstumsverfahren. In: Meinzer, H.-P. (Hrsg.) ; Deserno, T. M. (Hrsg.) ; Handels, H. (Hrsg.) ; Tolxdorff, T. (Hrsg.): *Bildverarbeitung für die Medizin*, Springer, 2009, S. 56–60

[13] Gerig, G. ; Koller, T. ; Székely, G. ; Brechbühler, C. ; Kübler, O. : Symbolic Description of 3-D Structures applied to Cerebral Vessel Tree obtained from MR Angiography Volume Data. In: *Proceedings of Information Processing in Medical Imaging*, Springer Verlag, 94–111

[14] Gibson, S. F. F.: Constrained Elastic Surface Nets: Generating Smooth Surfaces from Binary Segmented Data. In: *Proceedings of MICCAI*, 1998, S. 888–898

[15] Gray, H. : *Anatomy of the Human Body.* Philadelphia : Lea & Febiger, 1918

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

[17] Hintzenstern, V. von: *Interaktionstechniken zur Exploration von Gefäßbäumen für die Therapieplanung*, Otto-von-Guericke-Universität Magdeburg, Institut für Simulation und Graphik der Fakultät für Informatik, Diplomarbeit, 2006

[18] Jähne, B. : *Digitale Bildverarbeitung.* 6. Auflage. Berlin, Heidelberg : Springer, 2005

[19] Lorensen, W. E. ; Cline, H. E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: *Computer Graphics* 21 (1987), Nr. 4, S. 163–169

[20] Montagnat, J. ; Delingette, H. : 4D Deformable Models with Temporal Constraints: Application to 4D Cardiac Image Segmentation. In: *Medical Image Analysis* (2005), S. 87–100

[21] Oeltze, S. ; Preim, B. : Visualization of Vascular Structures: Method, Validation and Evaluation. In: *IEEE Transactions on Medical Imaging* 24 (2005), Nr. 4, S. 540–549

[22] Ohtake, Y. ; Belyaev, A. ; Alexa, M. ; Turk, G. ; Seidel, H.-P. : Multi-level Partition of Unity Implicits. In: *ACM Transactions on Graphics* 22 (2003), Nr. 3, S. 463–470

[23] Preim, B. ; Spindler, W. ; Peitgen, H.-O. : Interaktive medizinische Volumenvisualisierung - ein Überblick. In: *Proceedings of Simulation und Visualisierung*, 2000, S. 69–88

[24] Puig, A. ; Tost, D. ; Navazo, I. : An Interactive Cerebral Blood Vessel Exploration System. In: *IEEE Visualization*, 1997, S. 443–ff.

[25] Schöbinger, M. ; Thorn, M. ; Vetter, M. ; S., C. E. C. ; Hassenpflug, P. ; Wolf, I. ; Meinzer, H.-P. : Robuste Analyse von Gefäßstrukturen auf Basis einer 3D-Skelettierung. In: *Bildverarbeitung für die Medizin 2003*, Springer, 2003, S. 76–80

[26] Schöbinger, M. ; Wolf, I. ; Thorn, M. ; Hasenpflug, P. ; Vetter, M. ; Meinzer, H.-P. : Effiziente Verarbeitung von Gefäßgraphen auf Basis von Open-Source Frameworks. In: *Bildverarbeitung für die Medizin*, Springer, 2004, S. 381–385

[27] Schmidt, R. F. ; Lang, F. ; Heckmann, M. : *Physiologie des Menschen.* 31. Auflage. Berlin, Heidelberg : Springer Verlag, 2011

[28] Schroeder, W. J. ; Avila, L. S. ; Hoffman, W. : Visualizing with VTK: A Tutorial. In: *IEEE Computer Graphics and Applications* 20 (2000), S. 20–27

[29] Schumann, C. ; Oeltze, S. ; Bade, R. ; Preim, B. : Model-free Surface Visualization of Vascular Trees. In: *IEEE Eurographics Symposium on Visualization*, 2007, S. 283–290

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

[31] Selle, D. ; Spindler, W. ; Preim, B. ; Peitgen, H.-O. : Mathematical Methods in Medical Imaging: Analysis of Vascular Structures for Liver Surgery Planning. In: Engquist, B. (Hrsg.) ; Schmid, W. (Hrsg.): *Mathematics unlimited - 2001 and beyond*, Springer Verlag, 2001, S. 1039–1059

[32] Siek, J. G. ; Lee, L.-Q. ; Lumsdaine, A. : *The Boost Graph Library: User Guide and Reference Manual.* Addison-Wesley, 2002

[33] Siewert, J. R. ; Brauer, R. B.: *Basiswissen Chirurgie.* 2. Auflage. Berlin, Heidelberg : Springer Verlag, 2010

[34] Soler, L. ; Delingette, H. ; Malandain, G. ; Montagnat, J. ; Ayache, N. ; Koehl, C. ; Dourthe, O. ; Malassagne, B. ; Smith, M. ; Mutter, D. ; Marescaux, J. : Fully automatic anatomical, pathological, and functional segmentation from CT scans for hepatic surgery. In: *Computer Aided Surgery* 6 (2001), Nr. 3, S. 131–42

[35] Tönnies, K. D.: *Grundlagen der Bildverarbeitung.* München : Pearson Studium, 2005

[36] Turau, V. : *Algorithmische Graphentheorie.* 3. Auflage. München : Oldenbourg, 2009

[37] Wang, X. ; Heimann, T. ; Lo, P. ; Sumkauskaite, M. ; Puderbach, M. ; Bruijne, M. de ; Meinzer, H. P. ; Wegner, I. : Statistical tracking of tree-like tubular structures with efficient branching detection in 3D medical image data. In: *Physics in Medicine and Biology* 57 (2012), Nr. 16, S. 5325–42

[38] Wegner, I. : *Entwurf und Realisation eines generischen Interaktionsmodells mit Undo-Funktionalität für medizinische Bildverarbeitung*, Universität Heidelberg / Hochschule Heilbronn, Diplomarbeit, 2003

[39] Wegner, I. ; Biederer, J. ; Tetzlaff, R. ; Wolf, I. ; Meinzer, H.-P. : Evaluation and Extension of a Navigation System for Bronchoscopy inside Human Lungs. In: *Proceedings of SPIE Medical Imaging: Visualization, Image-Guided Procedures* Bd. 6509, 2007

[40] Wegner, I. ; Vetter, M. ; Schoebinger, M. ; Wolf, I. ; Meinzer, H.-P. : Development of a Navigation System for Endoluminal Brachytherapy in Human Lungs. In: *Proceedings of SPIE Medical Imaging: Visualization, Image-Guided Procedures* Bd. 6141, 2006, S. 23–30

[41] Wolf, I. ; Vetter, M. ; Wegner, I. ; Böttger, T. ; Nolden, M. ; Schöbinger, M. ; Hastenteufel, M. ; Kunert, T. ; Meinzer, H.-P. : The medical imaging interaction toolkit. In: *Medical Image Analysis* 9 (2005), Nr. 6, S. 594–604

[42] Wolf, I. ; Vetter, M. ; Wegner, I. ; Nolden, M. ; Böttger, T. ; Hastenteufel, M. ; Schöbinger, M. ; Kunert, T. ; Meinzer, H.-P. : The medical imaging interaction toolkit (mitk) - a tool facilitating the creation of interactive software by extending vtk and itk. In: Galloway, R. L. (Hrsg.): *Proceedings of SPIE Medical Imaging: Visualization, Image-Guided Procedures, and Display* Bd. 5367, 2004, S. 16–27

# A. Anhang

## Speicherung der Datenstruktur

```
<tube_graph_file file_version="0.1">
    <geometry xx="1" xy="0" xz="0" yx="0" yy="1" yz="0" zx="0" zy="0" zz="1"
        origin_x="0" origin_y="0" origin_z="0" spacing_x="1" spacing_y="1" spacing_z="1" />
    <vertices>
        <vertex vertex_id="0">
            <element element_x="114.971" element_y="108.783" element_z="68" element_diameter="7.3125" />
        </vertex>
        ...
    </vertices>
    <edges>
        <edge edge_id="1" edge_source_id="1" edge_target_id="2">
            <element element_x="113.978" element_y="108.221" element_z="71" element_diameter="6.45514" />
            <element element_x="114.03" element_y="107.658" element_z="72" element_diameter="5.6323" />
            ...
        </edge>
        ...
    </edges>
    <labelgroups>
        <labelgroup labelgroup_name="Vessel Type">
            <label label_name="Undefined" label_visible="1"
                label_color_r="170" label_color_g="170" label_color_b="169" />
            <label label_name="Artery" label_visible="1"
                label_color_r="255" label_color_g="0" label_color_b="0" />
            <label label_name="Hepatic Vein" label_visible="1"
                label_color_r="255" label_color_g="0" label_color_b="0" />
            <label label_name="Portal Vein" label_visible="1"
                label_color_r="0" label_color_g="0" label_color_b="125" />
            <label label_name="Bile Duct" label_visible="1"
                label_color_r="0" label_color_g="255" label_color_b="0" />
        </labelgroup>
        ...
    </labelgroups>
    <attributions>
        <attribut tube_id_1="2" tube_id_2="4" labelgroup_name="Vessel Type" label_name="Artery" />
        <attribut tube_id_1="7" tube_id_2="15" labelgroup_name="Resection Status" label_name="area at risk" />
        <attribut tube_id_1="7" tube_id_2="15" labelgroup_name="Vessel Type" label_name="Portal Vein" />
        <attribut tube_id_1="8" tube_id_2="9" labelgroup_name="Vessel Type" label_name="Portal Vein" />
        ...
    </attributions>
    <annotations>
        <annotation annotation_name="Aneurysm" annotation_description="" tube_id_1="120" tube_id_2="112" />
        ...
    </annotations>
</tube_graph_file>
```

Abbildung A.1.: Auszug aus der XML-Datei für tubuläre Strukturen: Die Struktur des Graphen wird hier auszugsweise abgebildet. Zudem sind beispielshaft Informationen der Property Klasse dargestellt.