Echtzeitinteraktion und -simulation deformierbarer dreidimensionaler Objekte für mikrochirurgische Trainingsmodule

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Dipl.-Inf. Evangelos Sismanidis
aus Stuttgart

Mannheim, 2015

Dekan:	Professor Dr. Heinz Jürgen Müller, Universität Mannheim
Referent:	Professor Dr. Reinhard Männer, Universität Heidelberg
Korreferent:	Professor Dr. Wolfgang Effelsberg, Universität Mannheim

Tag der mündlichen Prüfung: 30. Juli 2015

Für Lala

Zusammenfassung

Virtual-Reality-Simulatoren kommen in vielen Bereichen zum Einsatz. Ein Beispiel dafür ist das Training am Flugsimulator bei der Pilotenausbildung. *MicroSim* ist ein auf virtueller Realität basierender Simulator, an dem abstrakte mikrochirurgische Fertigkeiten sowie komplexe Operationsvorgänge der vaskulären Anastomose erlernt und trainiert werden sollen. Die vorliegende Arbeit trägt zur Entwicklung von *MicroSim* bei.

Das Ziel der Arbeit ist die Entwicklung prototypischer Module für das mikrochirurgische Training an *MicroSim*[1]. Um die Realität in *MicroSim* nachzustellen, müssen virtuelle Nachbildungen der Instrumente und des Gewebes erstellt und in Echtzeit simuliert werden. Durch die gewöhnliche Handhabung echter mikrochirurgischer Instrumente steuert der Benutzer die entsprechenden virtuellen Nachbildungen. Für *MicroSim*, bei dem Gewebe durch den Druck einer Pinzette nicht nur bewegt, sondern auch gegriffen oder gerissen werden kann, existierte bisher kein Algorithmus, der dieses Szenario in einem Echtzeit-Simulator abbildet. Für die Simulation der Anastomose muss zusätzlich die Interaktion zwischen den Blutgefäßen simuliert werden. Im Gegensatz zu der Benutzerinteraktion, bei der ein Festkörper mit einem deformierbaren Körper interagiert, interagieren bei der Kollision zwischen den Blutgefäßen zwei deformierbare Objekte miteinander. Die Schwierigkeit besteht darin, die komplexen Kollisionsflächen exakt zu bestimmen und eine Kraft zu berechnen um die Kollision realistisch aufzulösen.

Für die Darstellung des Gewebes werden Tetraedernetze durch ein *Masse-Feder-Modell* simuliert. Es existiert eine Vielzahl an Ansätzen für die Erstellung einfacher und komplexer Tetraedernetze. Auch die Simulation von Tetraedernetzen ist breit erforscht. Jedoch eignen sich viele Algorithmen der Fachliteratur nur bedingt für die interaktive Simulation. Durch unvorhersehbare Benutzereingaben treten kurzfristig hohe Kräfte im Simulator auf. Dies kann zu hohen numerischen Fehlern während des Integrationsschrittes führen. Für die Interaktion müssen Kollisionen mit dem Gewebe realistisch aufgelöst werden. Algorithmen für die Kollisionserkennung und -auflösung müssen an konkrete Problemstellungen angepasst

[1] Kurze Aufnahmen der Trainingsmodule finden sich auf http://www.sismanidis.de

werden. Für die Erkennung kann auf Algorithmen aus der Fachliteratur zurückgegriffen werden. Im Gegensatz dazu sind Lösungsansätze für die Auflösung der Kollisionen meist nicht direkt übertragbar. Diese müssen sowohl für die realistische Interaktion der Pinzette mit dem Gewebe als auch für die Simulation der Blutgefäße bei der Simulation des Vernähen für *MicroSim* entwickelt werden.

Um die Interaktion zwischen Mensch und Simulator zu implementieren, werden die Anforderungen an *MicroSim* definiert. Dafür werden mikrochirurgische Operationen analysiert, bei denen kleinste Blutgefäße von umliegendem Bindegewebe gesäubert werden, um anschließend vernäht werden zu können. Bei der Erkennung der Kollisionen zwischen Instrumenten und Gewebe werden die Instrumente durch Hüllkörper (*Bounding Volumes*) approximiert. Bei der Kollision wird Kraft auf das Gewebe übertragen. Um die verschiedenen Interaktionen der Instrumente mit dem Gewebe abbilden zu können, wird ein Algorithmus implementiert, der die Kollisionen wie folgt auflöst: Befindet sich Gewebe zwischen mehreren *Bounding Volumes* werden die Teile, die während der Kollisionserkennung in die *Bounding Volumes* eindringen, festgehalten, wodurch das Gewebe gegriffen werden kann. Um das Gewebe durch das feste Zusammenpressen der Pinzette teilen zu können, wird im Inneren der *Bounding Volumes* eine Linie definiert. Überschreitet das Gewebe während der Kollisionserkennung diese Linie wird ein Algorithmus für topologische Veränderungen angewendet. Diese werden durch ein *Remeshing*-Verfahren modelliert, das für das verwendete Simulationsmodell implementiert wird. Um Kollisionen des Tetraedernetzes aufzulösen, wird die benötigte Kraft auf Basis eines numerischen Integrationsverfahrens berechnet.

Die Simulation verwendet das *Velocity Verlet*-Verfahren. Die medizinischen Trainingsmodule, in denen die Algorithmen zum Einsatz kommen, werden abschließend prototypisch implementiert.

Mit den Resultaten der Arbeit können mit Hilfe der Erfahrungswerten von Chirurgen, medizinisch relevante Trainingsmodule in *MicroSim* implementiert werden. Ein Großteil von *MicroSim* wurde in mehreren Publikationen veröffentlicht. *MicroSim* wurde in Kooperation mit der *VRmagic GmbH* entwickelt.

Abstract

Virtual reality simulators are used in many areas, e.g. in the field of pilot training. *MicroSim* is a virtual reality simulator which can be used to train abstract microsurgical skills and teach complex procedures of the vascular anastomosis. The presented work is part of the development of *MicroSim*.

The objective of this dissertation is to develop prototypes of microsurgical training modules for *MicroSim*[2]. In order to simulate reality, virtual models of surgical instruments and soft tissue need to be created. The user controls the virtual models by handling real microsurgical instruments in his accustomed manner. In *MicroSim* soft tissue has not only got to be moved by forceps, but also grabbed, teared or cut precisely. No existing algorithm could be found which handles this scenario for a real time simulator. Additionally, the interaction between blood vessels has to be modelled for the simulation of the anastomosis. In contrast to user interaction, in which a rigid body interacts with a soft body, this case describes two or more soft bodies colliding with each other. The challenge here is, to calculate the complex collision surfaces and the resulting force to respond to the collisions.

In order to represent tissue which is a three dimensional deformable object, a tetrahedral mesh is simulated by a *spring-mass model*. There already exists a vast variety of approaches to generate simple and complex tetrahedral models. The simulation of tetrahedral meshes is also widely examined. Nevertheless, most algorithms are suitable only to a limited extent.

Unpredictable user behaviour can cause higher forces in the simulator. This can lead to larger numerical errors during the integration step. For the interaction, collisions have to be modeled in a realistic way. Algorithms for collision detection and response have to be adjusted for different cases. For collision detection well-known algorithms can be used. When it comes to collision response, on the other hand, the common approaches cannot be directly translated. These algorithms need to be developed for *MicroSim* for the realistic interaction of the instruments

[2]Short recordings of the training modules are located at https://www.sismanidis.de

with the tissue and for the simulation of the micro blood vessels for the anastomosis.

In order to implement the human simulator interaction, the requirement of *MicroSim* is defined by an Analysis of microsurgical operations where micro blood vessels are cleaned from surrounding connective tissues for the anastomosis. Collisions between instruments and tissues are detected by using *Bounding Volumes*. During a collision, a force is transmitted. In order to simulate the different interaction cases between the instruments and the soft tissue a collision algorithm is implemented which responds to the occurring collision as follows: if tissue lies between more than one *Bounding Volume* the parts which penetrate the *Bounding Volume* are held still, which simulates grabbing the tissue. In order to simulate the dissection of tissue, a line in the inner part of the *Bounding Volumes* is defined. If the line is crossed during a collision detection, an algorithm for topological changes is triggered. Topological changes are modelled by a remeshing algorithm. To respond to collisions of the tetrahedral mesh, the required force is calculated based on a numerical integration algorithm.

The implemented simulation uses the *Velocity Verlet* method. Prototypes of surgical training modules have been implemented with the use of the aforementioned algorithms.

With the input of experienced surgeons, the results will be used to implement medical relevant training modules into *MicroSim*. Parts of *MicroSim* have been presented at several conferences. *MicroSim* has been developed in cooperation with *VRmagic GmbH*.

Vorwort

Als ich 2007 das Seminar *Virtual Reality* der Virtual-Patient-Analysis-Gruppe (ViPA) am Lehrstuhl von Professor Männer besuchte, war mir nicht bewusst, welche weitreichenden Folgen diese Entscheidung für mich haben würde. Das Seminar lehrte mich nicht nur C++ und das Benutzen einer 3D-Grafik-Engine, wodurch ich anschließend einen Praktikumsplatz in den Laboren der IBM bekam, sondern auch die Umsetzung physikalischer Gegebenheiten in einer virtuellen Umgebung. Die Möglichkeit mit Hilfe der virtuellen Realität Simulatoren bauen zu können, faszinierte mich fortan. Ich verpflichtete mich im Jahr darauf, meine Diplomarbeit über das Thema *Echtzeitfähige Modellierung elementunabhängiger Schnitte in Tetraedernetzen* zu schreiben. Als mir am Ende meines Diploms eine Forschungsstelle an selbigem Lehrstuhl angeboten wurde, sagte ich zu und begann gemeinsam mit Oliver Schuppe, Nathan Hüsken und der Hilfe der *VRmagic GmbH* die Arbeit an einem mikrochirurgischen Simulator. Der Inhalt der vorliegende Dissertation entstand während dieser Zeit.

Den Algorithmus, den ich während der Diplomarbeit implementierte, konnte für die vorliegende Dissertation in abgeänderter Form verwenden werden. Da die Diplomarbeit nie veröffentlicht wurde, wird an diesen Stellen nicht explizit darauf hingewiesen.

Im Folgenden möchte ich einigen Leuten meinen Dank aussprechen, da diese Arbeit ohne sie nicht möglich gewesen wäre.

Mein Dank gilt vor allem Professor Reinhard Männer für sein Vertrauen und die Betreuung meiner Doktorarbeit, sowie Professor Effelsberg, der sich als Koreferent zur Verfügung gestellt hat.

Außerdem möchte ich meinen Eltern Anthoula und Moschos Sismanidis, meiner Schwester Eleni und Maxim danken, die nicht müde wurden, mich zu motivieren. Ein besonderer Dank gilt Oliver Schuppe, Florian Beier, Nathan Hüsken und der gesamten ViPA-Gruppe für die schöne Zeit am Lehrstuhl und für die Unterstützung. Nicht vergessen möchte ich Stephan Diederich, der mich nicht nur zu dieser

Promotion bewegt hat, sondern darüber hinaus stets ein kritisches Auge auf meine Arbeit hatte. Danke!

Ich danke der VRmagic GmBH, ohne deren Ressourcen diese Arbeit undenkbar gewesen wäre, und allen Mitarbeitern, die mich bei meiner Arbeit unterstützt haben. Ein besonderer Dank gilt Andrea Seeger und Dina Geppert, die mir bei Verwaltungsangelegenheiten immer zur Seite gestanden haben.

Der Firma S & T danke ich, dass sie mir mikrochirurgische Instrumente und Literatur bedingungslos zur Verfügung gestellt haben. Dr. Louis Padilla möchte ich dafür danken, dass er mir Ausschnitte aus seinen mikrochirurgischen Aufnahmen zur Verwendung überlässt.

Inhaltsverzeichnis

KAPITEL 1

Es werden mehr Menschen durch Übung tüchtig als durch Naturanlage.

(Demokrit ca. 460 - 370 v. Chr.*)*

Einleitung

Im Jahr 1921 wurde zum ersten Mal ein Mikroskop bei einem chirurgischen Eingriff eingesetzt. Der Pionier, der damals ein monokulares Mikroskop bei Experimenten an Hasenohren verwendete, war *C. O. Nylén* (Abbildung 1.1). Mit Hilfe eines binokularen Mikroskops führte er noch im selben Jahr Operationen am Ohr des Menschen durch. Die Mikrochirurgie war geboren. *A. Miehlke* schreibt in seinem Werk *Illustrierte Geschichte der Mikrochirurgie* [MT07] über *C. O. Nylén*: „Er wird mit Recht [...] als 'Vater der Mikrochirurgie' bezeichnet."

Fortan wurden die Auflösung und die Handhabung der Mikroskope stetig weiterentwickelt, sodass sie in den 1960er Jahren nach der Ohrenheilkunde auch in weiteren Teilgebieten der Medizin Anwendung fanden. Einige Beispiele dafür sind die Rhinologie, die Ophthalmologie und die Neurochirurgie. Nicht zuletzt diese Verbesserung der Mikroskopie verhalf der Mikrochirurgie, sich mit der Zeit in der Medizin zu etablieren. Die Möglichkeit, präzise im Millimeter- und Submillimeterbereich arbeiten zu können, führte sogar zur Entwicklung neuer Disziplinen, wie z. B. der Mikrogefäßchirurgie. Diese beschäftigt sich unter anderem mit der Wiederanschließung abgetrennter Körperteile an den Blutkreislauf.

Der Anstieg der Komplexität der operativen Eingriffe stellte die Chirurgen vor neue Herausforderungen. Früh wurde erkannt, dass die mentale und körperliche

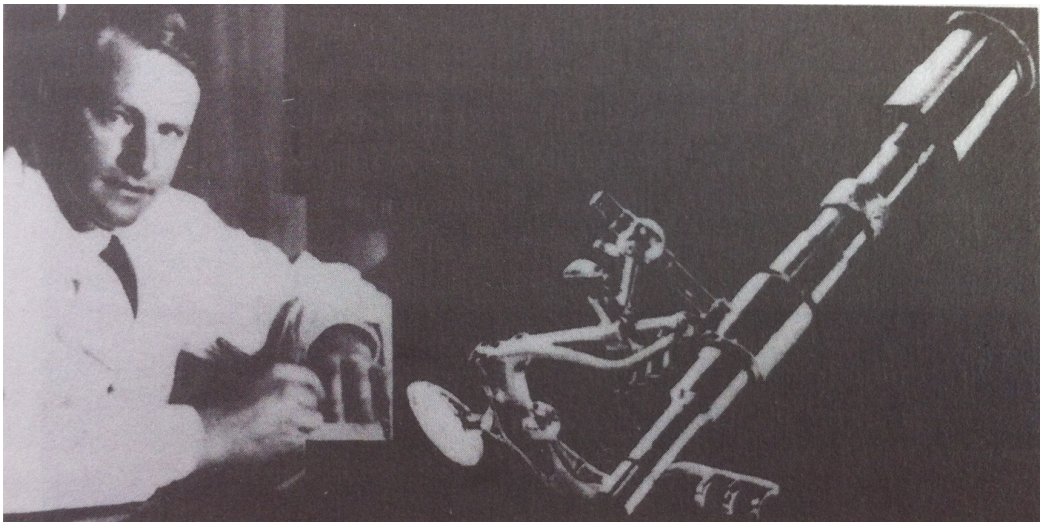


Abbildung 1.1: *C. O. Nylén* und sein monokulares Mikroskop. Quelle: [MT07]

Belastung in der Mikrochirurgie größer ist als in der Allgemeinchirurgie. Für den sicheren Umgang mit dem Mikroskop und dem fachspezifischen Instrumentar bedarf es neben einer geeigneten Ausbildung auch regelmäßigem Training. *H. L. Wullstein*, der die mikrochirurgischen Techniken ab 1949 in Deutschland einsetzte und maßgeblich zur Entwicklung der mikrochirurgischen Mikroskopie beigetragen hat, schreibt diesbezüglich (siehe [MT07]):

> „Wenn das Instrument im Blickfeld des Operationsmikroskops erscheint, so muss die Bewegung der Hand und der Finger der Vergrößerung des Mikroskops sofort angepasst werden, um Schaden am Patienten zu vermeiden. Das erfordert äußerste Disziplin und Training jeder kleinsten Bewegung bis diese sozusagen automatisiert abläuft."

Wullstein beschreibt die ungewohnte Hand-Auge-Koordination beim Arbeiten mit dem Mikroskop. Anders als bei der herkömmlichen Chirurgie, bei der ein operierender Arzt das Operationsfeld und seine Bewegungen direkt observiert, betrachtet er bei einem mikrochirurgischen Eingriff das Operationsfeld und seine Bewegungen der Instrumente indirekt. Ein angehender Chirurg muss deshalb lernen, Hand und Auge bei den Vergrößerungen unter dem Mikroskop zu koordinieren.

Weitere Herausforderungen, die Anfänger überwinden müssen, sind Ermüdungserscheinungen und Tremor. Diese werden zum einen durch die ungewohnte Haltung zum Patienten und zum anderen durch die ungewohnten Bewegungen und die Handhabung der Instrumente hervorgerufen.

Um Ermüdungen entgegenzuwirken gibt *R. Acland* in [AR08] als wichtigste Faktoren die Positionierung zum Mikroskop und eine komfortable Haltung an. In der Fachliteratur werden viele, sich zum Teil widersprechende, Körperhaltungen vorgeschlagen. Inzwischen ist man sich einig, dass jeder einzelne seine persönliche Position experimentell herausfinden sollte (vgl. [WHPK10]).

Des Weiteren ist eine ruhige Hand nicht angeboren, sondern muss erlernt werden. Da in der Mikrochirurgie nur die Fingerspitzen bewegt werden, sollte die restliche Hand auf einem unbeweglichen Untergrund liegen. Gerade in der Mikrogefäßchirurgie sind Eingriffe oft zeitkritisch und finden in einem Bereich von wenigen Millimetern statt. Dies erfordert exzellente feinmotorische Fertigkeiten, die durch spezielle Ausbildungs- und Übungsmöglichkeiten erlangt werden sollten. *Acland* schreibt weiter, dass frühe Erfolge beim Erlernen der Techniken enorm wichtig sind, um das nötige Selbstvertrauen für eine Operation zu entwickeln. Bei Studenten, bei denen die Lernumgebung eine Anspannung hervorruft, können diese Erfolge ausbleiben.

Aus den genannten Gründen resultiert, dass Studenten frühzeitig die Möglichkeit gegeben werden muss, das Arbeiten unter dem Mikroskop regelmäßig üben zu können.

1.1 Praktische Ausbildung in der Mikrogefäßchirurgie

Für die chirurgische Ausbildung galt lange der Leitsatz: “See One, Do One, Teach One“. Sinngemäß kann das mit „Schau einer Operation zu, führe eine durch und bilde den Nächsten darin aus“ übersetzt werden. *Mason et al.* [MS03] stellt fest, dass der Alltag vieler auszubildender Ärzte heute noch so aussieht. Dieser Ansatz wird von *Bergamaschi* [Ber01] für die minimalinvasive Chirurgie als nicht mehr zeitgemäß erklärt. Als Gründe werden unter anderem die wachsende Komplexität der Operationen, die geringeren Arbeitsstunden und der Verlust des taktilen Feedbacks genannt. Diese Kritikpunkte lassen sich auf die Mikrogefäßchirurgie übertragen, da der Chirurg haptische Wahrnehmungen beim Freipräparieren und Vernähen meistens dann spürt, wenn er fehlerhaft handelt.

Heutzutage existieren mikrochirurgische Operationskurse. In diesen können sich Studenten die nötigen allgemeinen und fachspezifischen Fertigkeiten aneignen. In diesen Kursen werden Mediziner im Umgang mit den Originalinstrumenten unter dem Mikroskop geschult. An einem Blatt, einer dünnen Gummi- oder Kunststofffolie werden die ersten Schnitt-, Greif- und Nähversuche unternommen. Ei-

ne Übersicht aktueller Kurse und der damit verbundenen Kosten bietet die Private Akademie der Deutschen Gesellschaft für Gefäßchirurgie und Gefäßmedizin (DGG) (vgl.[Pri]).

Wet Labs

Während die allgemeine Koordination am Mikroskop trainiert werden kann, stellt die Vermittlung der operationsspezifischen Algorithmen und Techniken nach wie vor ein Problem dar. Um diese zu erlernen, werden in sogenannten *Wet Labs* tierische Organe aus dem Schlachthof für das Vernähen von Blutgefäßen oder Nerven verwendet. Sobald die grundlegenden Techniken beherrscht werden, wird das Erlernte an betäubten Tieren vertieft, da lebendiges und totes Gewebe unterschiedliche Eigenschaften aufweisen. Es werden Gefäße, Nerven oder Sehnen getrennt und anschließend wieder vernäht. *S. Remmert und K. Sommer* erklären in [RS95]: „Die eigentliche Mikrogefäßanastomosetechnik erlernt man am besten am vitalen Gewebe.“Lebendige Modelle haben neben den ethisch und moralischen Aspekten den Nachteil, dass sie relativ aufwendig und kostspielig sind, da sie nur einige Male verwendet werden können. Außerdem sind diese nicht ohne Einschränkungen auf die anatomischen Gegebenheiten des Menschen übertragbar. *Acland* schreibt in [AR08]: “[...] experience with the rat fails to simulate the difficulties of access, exposure, and control of unwanted movement that are often encountered in clinical work“. Der Großteil der chirurgischen Ausbildung findet letztlich am Patienten statt.

Dry Labs

Um die Situation zu verbessern, hat sich die Medizin ein Beispiel an Flugsimulatoren genommen. Diese werden seit Jahrzehnten erfolgreich in der Pilotenausbildung eingesetzt. Analog dazu werden in der Medizin vermehrt *Dry Labs* angeboten, in denen die praktische Ausbildung durch Trainingssimulatoren unterstützt wird. Dabei ist gerade die Anzahl der auf virtueller Realität (VR)[1] basierenden Simulatoren, die inzwischen weltweit Anerkennung finden, stark angewachsen (vgl. [Ros08]). Studien wie [RVF+04] und [HS06] belegen, dass diese effektiv genutzt werden können, um die Ausbildung in der Chirurgie zu unterstützen. Erst kürzlich wurden die Ergebnisse einer Studie in [SCN+12] vorgestellt, die belegen, dass VR-Simulatoren die Fertigkeiten der Teilnehmer signifikant verbessern. *Haluck et al.* [HK00] schreibt, dass reale Patienten und gute Mentoren nie komplett ersetzt

[1] Der Begriff *Virtuelle Realität* wird in Kapitel 3.1 definiert

werden können, aber Simulatoren ein essentieller Bestandteil der chirurgischen Ausbildung werden sollten.

VR-Simulatoren

Simulatoren lassen sich in der Chirurgie vielseitig einsetzen: Mediziner in der chirurgischen Ausbildung können die pathologische Anatomie, Operationstechniken und -abläufe, die Benutzung fachspezifischer Instrumente und Hilfssysteme erlernen. Des Weiteren können sie sich motorische Fertigkeiten aneignen und diese perfektionieren. Ein weiterer Vorteil von auf VR-basiertem Training besteht in der Möglichkeit, Studenten objektiv bewerten und Entwicklungen chirurgischer Fertigkeiten exakt messen zu können (vgl. [O'C04]). Nicht zuletzt geben medizinische Simulatoren Chirurgen die Möglichkeit, realitätsnahe Trainingseinheiten zu absolvieren, ohne das Leben von Patienten zu gefährden.

VR-Simulatoren existieren inzwischen für verschiedene Teildisziplinen der Medizin. Am Lehrstuhl für Informatik V der Universität Heidelberg wurden unter anderem *CATHi* für Kathetereingriffe (vgl. [KHVM02]), ein Koloskopie Trainingssimulator (vgl. [Kör03]) und der Augensimulator *Eyesi* für intraokulare Chirurgie [SWH+99] entwickelt. Letzterer wird von der Firma *VRmagic GmbH* inzwischen [VRm] weltweit vertrieben.

1.2 Das Projekt *MicroSim*

Die vorliegende Arbeit ist Teil einer Kooperation zwischen der Forschungsgruppe Virtuelle Patienten Analyse (ViPA) des Lehrstuhls für Informatik V unter der Leitung von *Prof. Dr. R. Männer* und der *VRmagic GmbH*. Das Ziel dieser Kooperation war die Entwicklung des Prototypen *MicroSim*: ein mikrochirurgischer Trainingssimulator basierend auf den Technologien der virtuellen Realität (vgl. [Pre10])[2]. Dieser soll in der Zukunft beim Erlernen mikrochirurgischer Fertigkeiten unterstützen.

Ausschlaggebend für die Entwicklung von *MicroSim* sind die Schwierigkeiten beim Erlernen allgemeiner, mikrochirurgischer und spezieller Fertigkeiten bei der mikrovaskulären Anastomose:

[2]Das Projekt wurde durch das Zentrale Innovationsprogramm Mittelstand vom Bundesministerium für Wirtschaft gefördert [Bun12].

- ungewohnte Hand-Auge-Koordination beim Arbeiten unter dem Mikroskop,
- Ermüdungserscheinungen und Tremor durch die ungewohnte Haltung und Führung der Instrumente,
- Ausbleiben frühzeitiger Lernerfolge durch unregelmäßiges Training und angespannte Trainingsumgebung,
- ethische und moralische Aspekte beim Üben an Tieren und Leichen,
- hohe Kosten und hoher Verschleiß lebendiger Modelle,
- keine direkte Übertragbarkeit der Pathologie und Anatomie von Tieren auf den Menschen,
- Gefährdung der Patienten.

Für die Ausbildung mikrochirurgischer Fertigkeiten an VR-Simulatoren spricht die Tatsache, dass sich die Mikrochirurgie besonders gut für die Übersetzung in die VR eignet, da sowohl in der Realität als auch in der VR ein optisches Gerät für das Betrachten verwendet wird. Des Weiteren wird bei einer Operation in einer sehr kleinen Umgebung gearbeitet, sodass die Haptik im Gegensatz zu anderen chirurgischen Disziplinen von geringerer Bedeutung ist (vgl. *Lannon et al.* [LAB01]). *Lannon et al.* leiten daraus ab, dass eine haptische Benutzerschnittstelle für einen realistischen mikrochirurgischen Trainingssimulator keine Voraussetzung ist. Abbildung 1.2 veranschaulicht die Ähnlichkeit der Haltung und Positionierung der Instrumente eines Chirurgen zum Patienten während einer realen Operation (Abbildung 1.2(a)), mit der Benutzerschnittstelle eines Benutzers von *MicroSim* während eines Trainings (Abbildung 1.2(b)).

MicroSim ist modular aufgebaut und wurde als fächerübergreifende Trainingsumgebung auf der Basis von VR entwickelt. Die vorliegende Arbeit beschäftigt sich mit den Algorithmen für die interaktive Simulation von Blutgefäßen und umliegendem Bindegewebe in Echtzeit (vgl. Abschnitt 1.3).

1.3 Ziel der Arbeit

Das Paradigma der vorliegenden Arbeit findet sich in der Mikrogefäßchirurgie, in der kleinste Blutgefäße freigelegt und miteinander vernäht werden. Im Speziellen werden Algorithmen für die echtzeitfähige Simulation und Interaktion von virtuellen Blutgefäßen und umliegendem Gewebe untersucht.

(a) Arzt während eines mikrochirurgischen Eingriffs.

(b) *MicroSim*-Benutzer während des Trainings.

Abbildung 1.2: Vergleich der Haltung eines Chirurgen mit der Haltung eines Anwenders von *MicroSim*.

Das erste Ziel besteht darin, geeignete Algorithmen zu entwickeln oder existierende zu adaptieren, um Trainingsmodule für die Entwicklung von *MicroSim* implementieren zu können. An diesen soll das Freipräparieren von Blutgefäßen für die vaskuläre Anastomose trainiert werden. Dem Autor sind aktuell keine auf VR basierende Simulatoren bekannt, die dem Training dieser Operationvorgänge dienen.

Das zweite Ziel umfasst die Implementierung eines Algorithmus, der für die Anastomose der virtuellen Blutgefäße die Kollisionen zwischen diesen erkennt und auflöst.

Diese Ziele werden erreicht, indem die Anatomie der Mikrogefäße studiert und sowohl der Algorithmus als auch die damit verbundenen Schwierigkeiten dieses Eingriffs, extrahiert werden. Die Methoden werden im Rahmen der VR und der damit verbundenen Einhaltung der Echtzeit-Randbedingung, wie sie in Abschnitt 3.1 definiert wird, entwickelt und für *MicroSim* implementiert. Die Modelle und Algorithmen, die dafür benötigt werden, schlüsseln sich wie folgt auf:

Deformierbare Objekte

Für die Darstellung der Blutgefäße werden Modelle erstellt. Diese und weitere geometrische Objekte, die für die abstrakten Module benötigt werden, werden im Hinblick auf die Kompatibilität zum Simulationsmodell analytisch modelliert (Abschnitt 4.1).

Simulationsmodell

Die Simulation deformierbarer Objekte in Echtzeit baut auf den Algorithmen aus [THMG04] auf. Diese unterstützen topologische Änderungen der Simulationsobjekte, wodurch Risse und Schnitte des Materials simuliert werden können (Abschnitt 4.2).

Kollisionserkennung

Für die Anastomose müssen Kollisionen zwischen den Blutgefäßen erkannt werden. Es werden verschiedene Teilalgorithmen aus anderen Bereichen der physikalischen Simulation angepasst und implementiert (Abschnitt 4.3).

Benutzereingaben werden in *MicroSim* über die chirurgischen Instrumente in die virtuelle Welt übertragen. Diese werden für die Interaktion des Benutzers mit der Operationsszene verwendet. Um die simulierten Objekte zu bearbeiten, müssen Kollisionen zwischen den Instrumenten und den Objekten erkannt werden. Aus der Fachliteratur bekannte Algorithmen werden angepasst und implementiert.

Werden Kollisionen zwischen einem Instrument und einem Objekt erkannt, muss entschieden werden, wie das Modell vom Benutzer manipuliert werden soll. Das Gewebe kann von der Pinzette bewegt, gegriffen, gerissen oder mit einer Schere präzise geschnitten werden. Der Algorithmus, der entscheidet, welche Kollisionsantwort ausgelöst werden soll, wird in dieser Arbeit entwickelt und implementiert (Abschnitt 4.4).

Kollisionsauflösung

Die Auflösung der einzelnen Kollisionen muss für den Benutzer plausibel erscheinen. Abhängig davon, wie eine Kollision aufgelöst wird, werden die deformierbaren Objekte verdrängt, verformt, geschnitten oder gerissen. Topologische Veränderungen geschnittener oder gerissener Objekte werden durch einen *Remeshing*-Algorithmus modelliert. Dabei werden existierende Tetraeder aus dem Tetraedernetz entfernt und durch neue Tetraeder ersetzt. Aus der Fachliteratur bekannte Algorithmen werden angepasst und in das verwendete Simulationsmodell implementiert (Abschnitt 4.2.1).

1.4 Überblick über die Arbeit

Kapitel 1 bildet die Einführung, Motivation und Zielsetzung der Arbeit. Kapitel 2 führt in die medizinischen Grundlagen ein. Kapitel 3 führt in die technischen Grundlagen ein und gibt den aktuellen Stand der Technik wieder. Es werden bisherige Arbeiten zum Thema mikrochirurgischer Trainingssimulatoren und Algorithmen aus den einzelnen Bereichen, die in Abschnitt 1.3 genannt wurden, vorgestellt. Die Methoden der Arbeit werden in Kapitel 4 erörtert. Kapitel 5 stellt den Funktionsumfang, den Aufbau sowie die Hard- und Software von *MicroSim* vor. Des Weiteren werden die implementierten Trainingsmodule vorgestellt. Kapitel 6 evaluiert die Arbeit abschließend und gibt einen Ausblick auf zukünftige Forschungsfelder.

1.5 Eigene Veröffentlichungen

- Sismanidis, Evangelos: *Triggering different collision response algorithms stated at penetration depths*. In: *IASTED - Modelling, Identification and Control*, April 2013.

- Hüsken, Nathan, Oliver Schuppe, Evangelos Sismanidis und Florian Beier: *MicroSim – A Microsurgical Training Simulator*. Studies in health technology and informatics, 184:205–209, 2013. **Winning Poster Presentation Certificate**.

- Sismanidis, Evangelos: *Real-time simulation of blood vessels and connective tissue for microvascular anastomosis training*. Virtual Reality Workshops (VR), 2012 IEEE, Seiten 109–110, 2012.

- Beier, Florian, Evangelos Sismanidis, A Stadie, K Schmieder, Reinhard Männer und others: *An Aneurysm Clipping Training Module for the Neurosurgical Training Simulator NeuroSim.* Studies in health technology and informatics, 173:42, 2012.

KAPITEL

2

Ich werde nicht schneiden, sogar Steinleidende nicht, sondern werde das den Männern überlassen, die dieses Handwerk ausüben.

(Hippokrates von Kós ca. 460 - 370 v. Chr.*)*

Einführung in die Mikrogefäßchirurgie

Operative Eingriffe an Blutgefäßen werden in vielen therapeutischen Maßnahmen angewendet. Beispiele dafür sind die Replantations- und die plastische Chirurgie. Dabei können abgetrennte Körperteile durch das Vernähen der Blutgefäße wieder an den Blutkreislauf angeschlossen werden. Die Mikrogefäßchirurgie umfasst alle Behandlungen an Blutgefäßen, die einen Durchmesser von bis zu drei Millimetern haben und deshalb nur mit Hilfe von Operationsmikroskopen und Präzisionsinstrumenten durchgeführt werden können. Abschnitt 2.1 gibt einen Überblick über die Arbeitsmittel, die bei einem operativen Eingriff an Mikrogefäßen verwendet werden. In Abschnitt 2.2 wird der Aufbau eines Blutgefäßes am Beispiel einer Arterie erklärt und die anatomischen Unterschiede zwischen Vene und Arterie benannt. Mit diesen Voraussetzungen wird in Abschnitt 2.3 der operative Algorithmus einer vaskulären Anastomose beschrieben.

Das Kapitel stellt keinen Anspruch auf Vollständigkeit. Insbesondere wird auf die Operationstechniken der vaskulären Anastomose eingegangen, die für die vorliegende Arbeit relevant sind. Für eine vollständige Einführung in die Anatomie und die Operationstechniken der Mikrogefäßchirurgie sei auf drei Werke verwiesen: *Acland's Manual for Microvascular Sugery* [AR08]; *Green's Operative Hand*

Surgery [WHPK10]; *Kompendium des mikrovaskulären Gewebetransfers* [RS95]. Auf diesen beruhen die Erkenntnisse des aktuellen Kapitels.

2.1 Equipment

Je nach Fachgebiet variieren das Instrumentar und die Eigenschaften der Mikroskope. Im Folgenden wird der Aufbau eines mikrochirurgischen Operationsmikroskops beschrieben und ein Überblick über die Instrumente gegeben, die für das Freipräparieren und Säubern von Arterien und Venen benötigt werden.

Mikroskop

Heutzutage zeichnen sich mikrochirurgische Mikroskope neben der Möglichkeit, einen Bereich hochauflösend darzustellen, durch sehr gute Belichtungs- und Navigationsmöglichkeiten aus.

(a) *M525 F25* (b) *FL800*

Abbildung 2.1: Operationsmikroskop der Firma *Leica Microsystems GmbH*. Mit freundlicher Genehmigung der *Leica Microsystems GmbH*.

Abbildung 2.1(a) zeigt das Modell *M525 F25* der Firma *Leica Microsystems GmbH* [Leia]. Dieses passt die Lichtintensität des Mikroskops dem Abstand zum Patienten und den Belichtungsradius dem aktuellen Zoom-Faktor an, um Verbrennungen und unnötige Erhitzungen zu vermeiden (vgl. [Leib]). Abbildung 2.1(b) zeigt den Kopf des Operationsmikroskops *FL800*. Rechts sind die Griffe mit den

verschiedenen Navigationsschaltern zu sehen. Viele Mikroskope bieten die Möglichkeit der digitalen Aufnahme eines Eingriffs und die anschließende Übertragung der Daten auf mobile Endgeräte, um eine Operation anschließend mit Kollegen oder Angehörigen besprechen zu können. Modernste Mikroskope können den Blutfluss in den observierten Gefäßen mit Hilfe von Kontrastmittel visualisieren oder die Sicht durch das Mikroskop mit Daten anderer bildgebender Verfahren, z. B. aus der Magnetresonanztomographie (MRT) oder der Computertomographie (CT), überlagern.

Instrumente

In der Mikrogefäßchirurgie kommen neben dem Nadelhalter und dem speziellen Nahtmaterial, die für die Anastomose benötigt werden, spezielle Scheren, Pinzetten und teilweise Blutgefäß-Dilatatoren zum Einsatz. Auf eine Einführung in mikrochirurgische Nadeln, Nadelhalter und Nahtmaterialien wird verzichtet, da diese für die vorliegende Arbeit nicht relevant sind.

Abbildung 2.2: Jeweler's Pinzette der Firma *S&T*.

Das Instrument, das fast durchgängig mit der nicht dominanten Hand geführt wird, ist die Pinzette. Sie wird für die Handhabung des Gewebes und für das Festziehen der Naht verwendet. Der Abstand der Pinzettenspitzen ist dabei ausschlaggebend. Dieser muss beim Zusammendrücken der Pinzette weniger als ein hundertstel Millimeter betragen, um das Nahtmaterial halten zu können. Es gibt nicht nur gerade, sondern auch abgewinkelte Pinzetten. Diese werden benutzt, um verwinkelte Stellen zu erreichen oder Blutgefäße anzuheben. Weiter unterscheiden sich Pinzetten in ihren Größen. Abbildung 2.2 zeigt eine gerade *Jeweler's* Pinzette *No. 5* der Firma *S&T* [S&], wie sie in der Mikrogefäßchirurgie verwendet wird. Diese ist elf Zentimeter lang und ist für Blutgefäße mit einem Durchmesser von fünf Millimetern bis drei Zentimetern geeignet.

Sowohl bei der Trennung von Blutgefäßen als auch bei der Entfernung überflüssigen Gewebes kommen Präzisionsscheren zum Einsatz. Gerade bei der Säuberung

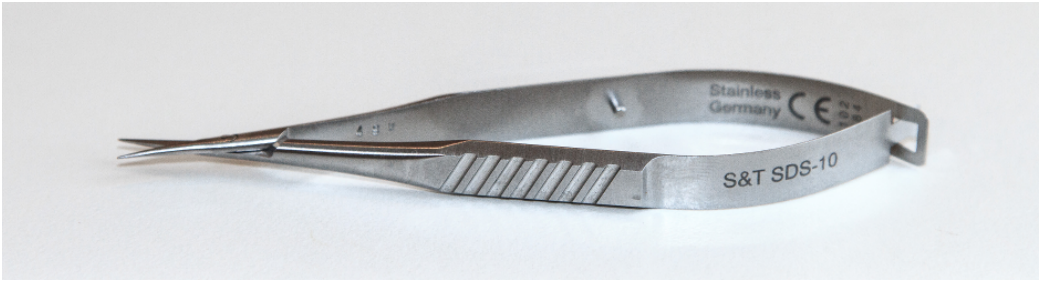


Abbildung 2.3: Adventitia Schere der Firma *S&T*.

eines Blutgefäßes und der damit verbundenen Entfernung des überflüssigen Gewebes an seiner Öffnung, wird eine speziell dafür entwickelte Schere verwendet. Diese wird im Folgenden als Adventitia- oder Mikroschere bezeichnet. Abbildung 2.3 zeigt solch eine Schere. Diese ist ebenfalls für Blutgefäße mit einem Durchmesser von fünf Millimetern bis drei Zentimetern geeignet.

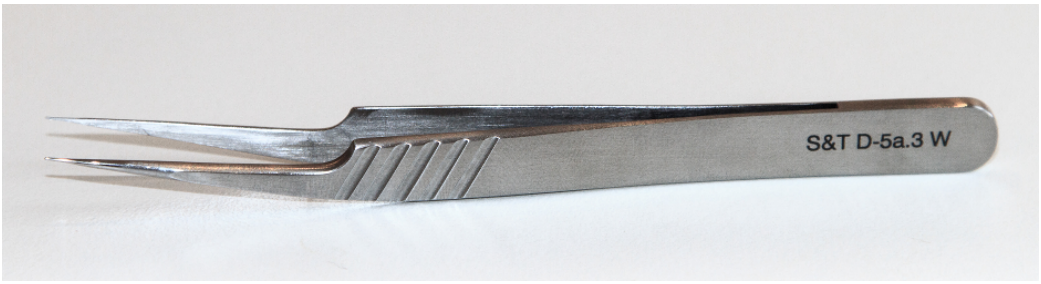


Abbildung 2.4: Blutgefäß Dilatator der Firma *S&T*.

Mit einem Dilatator wird ein Blutgefäß an der Öffnung, an der es vernäht werden soll, geweitet. Ein Dilatator hat Ähnlichkeit mit einer Pinzette (siehe Abbildung 2.4). Seine Enden sind gerade, wodurch sich der geschlossene Dilatator in die Öffnung eines Blutgefäßes einführen lässt. Durch das anschließende Öffnen wird das Gewebe gespreizt. Dies hat in erster Linie den Vorteil, dass sich die Blutgefäßenden zu einer Röhre formen und nicht schlaff zusammenliegen. Dadurch wird dem Chirurgen der anschließende Nahtvorgang erleichtert.

2.2 Anatomie eines Blutgefäßes

Sowohl Arterien als auch Venen bestehen im Wesentlichen aus drei Teilen: Die innere Zellschicht nennt sich *Tunica Interna* oder *Tunica Intima* (kurz: *Intima*). Diese ist von einer dickeren Muskelgewebsschicht umgeben, der *Tunica Media* (kurz: *Media*). Die äußere Schicht besteht aus Bindegewebe und wird *Tunica Ex-*

terna oder *Tunica Adventitia* (kurz: *Adventitia*) bezeichnet. Der von einem Blutgefäß eingeschlossene Innenraum, durch den das Blut fließt, nennt sich *Lumen*. Abbildung 2.5 veranschaulicht den Aufbau eines Blutgefäßes am Beispiel einer Arterie.

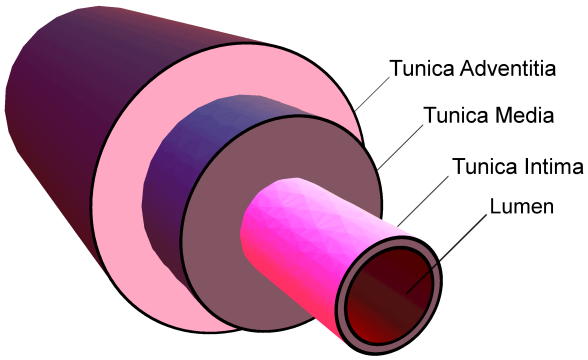


Abbildung 2.5: Querschnitt einer Arterie.

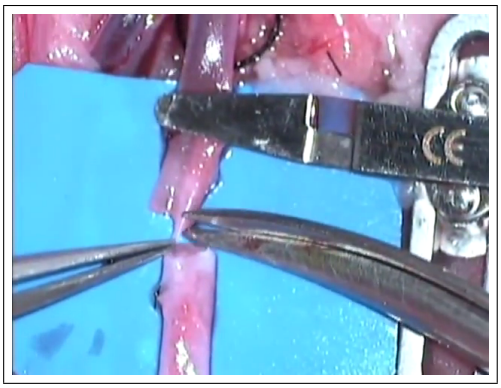
Im Gegensatz zu Arterien besitzen viele der kleineren und mittleren Venen zusätzliche Venenklappen, die das Zurückfließen des Blutes verhindern. Obwohl der Aufbau von Arterien und Venen sehr ähnlich ist, unterscheiden sich die zwei Blutgefäßtypen in einigen Details, die während eines mikrochirurgischen Eingriffs berücksichtigt werden müssen: Während die Adventitia und die Media bei einer Arterie klar voneinander getrennte Schichten bilden, ist der Übergang zwischen diesen bei einer Vene schwieriger auszumachen. Das liegt daran, dass bei Venen in der Adventitia muskelgewebige Elemente vorkommen und in der Media bindegewebige. Erschwerend kommt hinzu, dass die Media dünner ist. Im Gegensatz zu Venen ist die Adventitia bei Arterien füllig und liegt fast lose auf der Media. Weiterhin ist eine Vene empfindlicher als eine Arterie und dadurch leichter verletzbar. Arterien verzeihen Ungenauigkeiten, die beim Vernähen von Blutgefäßen gemacht werden, eher als Venen. Da der Blutfluss in Venen im Allgemeinen langsamer ist als in Arterien, werden Ablagerungen von Blutplättchen und damit die Bildung eines Thrombus begünstigt. [AR08] beschreibt den Umgang mit Venen bei der Operation als schwieriger und empfiehlt den Eingriff an diesen erst vorzunehmen, falls das Handwerk an Arterien beherrscht wird.

2.3 Vaskuläre Anastomose

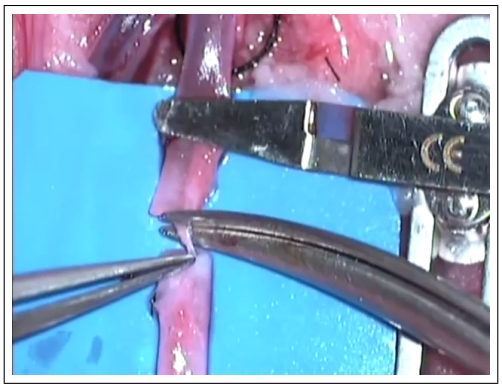
Das Vernähen von Blutgefäßen wird vaskuläre Anastomose genannt. Es existieren drei verschiedene Arten: Die *End-zu-End*-Anastomose beschreibt das Vernähen zweier Blutgefäße an ihren Enden. Analog dazu existieren die *End-zu-Seit-*

Anastomose, bei der das Ende eines Blutgefäßes an der Außenwand eines anderen angenäht wird, und die *Seit-zu-Seit*-Anastomose, bei der die Blutgefäße durch Öffnungen an ihren Außenwänden vernäht werden.

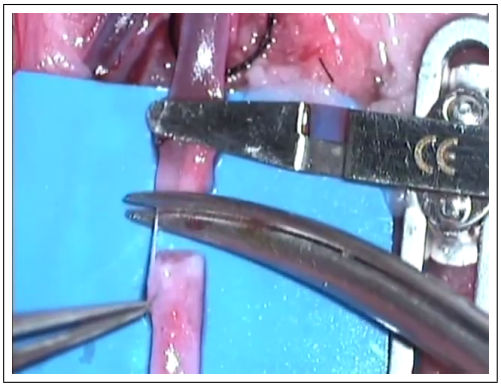
Bevor die Blutgefäße miteinander vernäht werden können, müssen diese freigelegt und entsprechend präpariert werden. Dazu wird die Adventitia nahe der Öffnung eines Blutgefäßes getrimmt. Dafür gibt es mehrere Gründe. Zum einen lässt sich das Ende eines Blutgefäßes besser von dem umliegenden Gewebe abgrenzen, wodurch die Nähte bei der anschließenden Anastomose genauer gesetzt werden können. Zum anderen soll verhindert werden, dass sich während der Anastomose Bindegewebe im Blutgefäß befindet, welches versehentlich eingenäht werden könnte. Da die Adventitia blutgerinnend wirkt, könnte dies zu einer Thrombose und damit zum Scheitern der Anastomose führen (vgl. [TBB+06]).



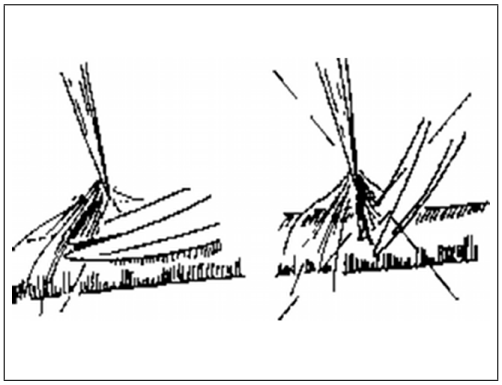
(a) Die Adventitia wird vorsichtig gegriffen.



(b) Die Schere wird nahe der Öffnung des Blutgefäßes justiert.



(c) Die Adventitia wird abgeschnitten.



(d) Die falsche Haltung der Schere kann zu Verletzungen des Blutgefäßes führen. Quelle: [RS95].

Abbildung 2.6: Scharfe Dissektion der Adventitia.

Das Entfernen der Adventitia kann auf verschiedene Weisen erfolgen. Das hängt unter anderem von der Beschaffenheit des Blutgefäßes ab. Wie in Abschnitt 2.2

beschrieben wurde, herrschen zwischen Venen und Arterien anatomische Unterschiede. Je nachdem wie gut sich die Adventitia von der Media abgrenzt, wird sie von Chirurgen zum Teil mit der Pinzette gefasst und zurückgeschoben oder vorsichtig abgerissen. Im Folgenden wird diese Technik als *Stumpfe Methode* bezeichnet. Die zweite gängige Methode besteht darin, die Adventitia vorsichtig mit einer Mikroschere zu trimmen. Dabei wird das Gewebe, welches entfernt werden soll, mit der Pinzette gegriffen, vorsichtig über das Ende des Blutgefäßes gezogen und anschließend mit der Schere abgeschnitten. Diese Technik wird im Folgenden als *Schneidende Methode* bezeichnet (vgl. [LSL98]). Die Abbildungen 2.6(a) - 2.6(c) zeigen das Greifen und Wegschneiden der Adventitia während eines realen operativen Eingriffs. Weiterhin lässt sich in Abbildung 2.6(c) das sehr elastische Verhalten der Adventitia beobachten.

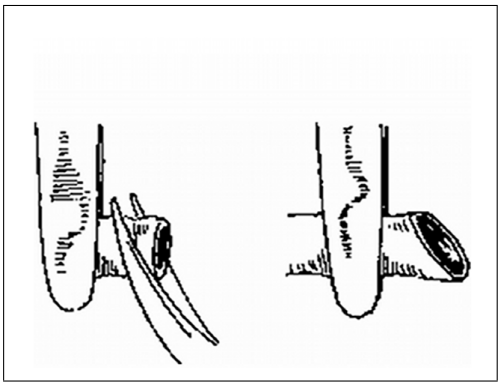
Ist die Adventitia schwer von der Media zu unterscheiden, muss das Blutgefäß in mehreren Schritten schichtweise von der Adventitia bereinigt werden, bis sich die muskelgewebigen Strukturen häufen (vgl. [AR08]).

Die Studie [LSL98] belegt in Bezug auf Arterien, dass die *Schneidende Methode* der *Stumpfen Methode* vorzuziehen ist. Während bei Letzterer das Blutgefäß durch die unpräzisen Bewegungen beschädigt werden kann, hat sich bei der *Schneidenden Methode* der Blutkreislauf nach Beendigung der Anastomose schnell normalisiert.

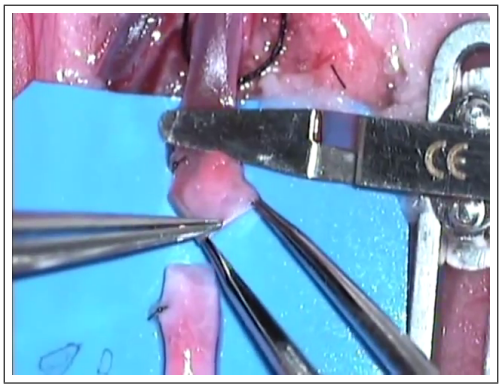
Allgemein wird in [LSL98] festgestellt, dass jede Art von unkontrollierter bzw. unüberlegter Kraftübertragung auf das Blutgefäß vermieden werden sollte, da dies zu einer Veränderung des Gewebes oder des Blutflusses und damit zu Komplikationen bei der Operation führen könnte. Des Weiteren soll der Chirurg das Blutgefäß nicht direkt mit der Pinzette greifen, sondern an der Adventitia oder anderem umliegenden Bindegewebe, um zu vermeiden, dass die Wand des Blutgefäßes eingedrückt wird (vgl. [RS95]). Beim Trimmen der Adventitia muss darauf geachtet werden, dass die Mikroschere oder andere spitze Instrumente nicht gewinkelt zum Blutgefäß geführt werden, sondern, soweit es möglich ist, parallel dazu. Abbildung 2.6(d) veranschaulicht die Haltung der Instrumente während der Operation am Blutgefäß.

Bei den Enden der Blutgefäße muss es sich nicht zwangsläufig um gleich große Öffnungen handeln. Kleinere Größenunterschiede können durch leichtes Ausdehnen des Blutgefäßes mit dem kleineren Durchmesser angepasst werden. Eine weitere Möglichkeit besteht darin, das Blutgefäß schräg anzuschneiden (vgl. [RS95]). Abbildung 2.7(a) skizziert den Präparationsvorgang.

Sobald ausreichend Adventitia an der Öffnung des Gefäßes entfernt wurde, kann die Öffnung mit Hilfe eines Dilatators gespreizt werden (siehe Abbildung 2.7(b)).



(a) Durch diagonales Anschneiden des kleineren Blutgefäßes. Quelle: [RS95].



(b) Durch Ausweiten mit einem Dilatator.

Abbildung 2.7: Anpassung der Größe der Blutgefäßenden.

In [AR08] werden für die Anwendung dieser Technik zwei Gründe genannt: Zum einen hängt die Öffnung nicht schlaff zusammen, sondern nimmt die Form einer Röhre an, wie bereits in Abschnitt 2.1 erklärt wurde, wodurch die Anastomose an dieser erleichtert wird. Zum Anderen wird das Blutgefäß durch den Dilatator paralysiert und verhindert dadurch postoperative Spasmen an der Nahtstelle. Damit ist der Präparationsvorgang abgeschlossen und die Gefäße können miteinander vernäht werden.

Analog dazu werden die Wände der Blutgefäße bei der *End-zu-Seit*- und bei der *Seit-zu-Seit*-Anastomose freigestellt. Unterschiede gibt es beim anschließenden Vernähen. Da das Vernähen von Blutgefäßen nicht Teil dieser Arbeit ist, wird für die vollständige Darstellung des gesamten Operationsvorgangs der vaskulären Anastomose auf [AR08] verwiesen.

KAPITEL

3

Der Anfang ist die Hälfte des Ganzen.

(Aristóteles von Stageira ca. 384 - 322 v. Chr.*)*

Grundlagen und Stand der Technik

Kapitel 3 fasst die technischen Voraussetzungen für das Verständnis von Virtueller Realität und der Simulation deformierbarer Objekte zusammen und gibt den aktuellen Stand der Technik wieder. In Abschnitt 3.1 werden die Begriffe *Virtuelle Realität* und *Echtzeit* definiert. Die verschiedenen Simulationsmodelle werden in Abschnitt 3.2 klassifiziert. Abschnitt 3.3 geht auf die Diskretisierung des Raumes ein und stellt Algorithmen zur Erstellung virtueller Objekte vor. Abschnitt 3.4 behandelt Masse-Feder-Modelle. In Abschnitt 3.5 wird die Diskretisierung der Zeit beschrieben und verschiedene numerische Integrationsverfahren besprochen.

Werden mehrere Objekte simuliert, die sich während der Simulation nicht durchdringen dürfen, müssen Kollisionen zwischen diesen berücksichtigt werden. Dabei wird zwischen Kollisionserkennung (Abschnitt 3.6) und Kollisionsbehandlung (Abschnitt 3.7) unterschieden. Eine Einführung in die Visualisierungskomponente einer Simulation gibt Abschnitt 3.8. Abschließend wird in den Abschnitten 3.9 und 3.10 ein Überblick über vorhandene Simulationsframeworks und chirurgische Trainingssimulatoren gegeben.

Abbildung 3 skizziert den Simulationszyklus beispielhaft und verdeutlicht den Zusammenhang der einzelnen Abschnitte.

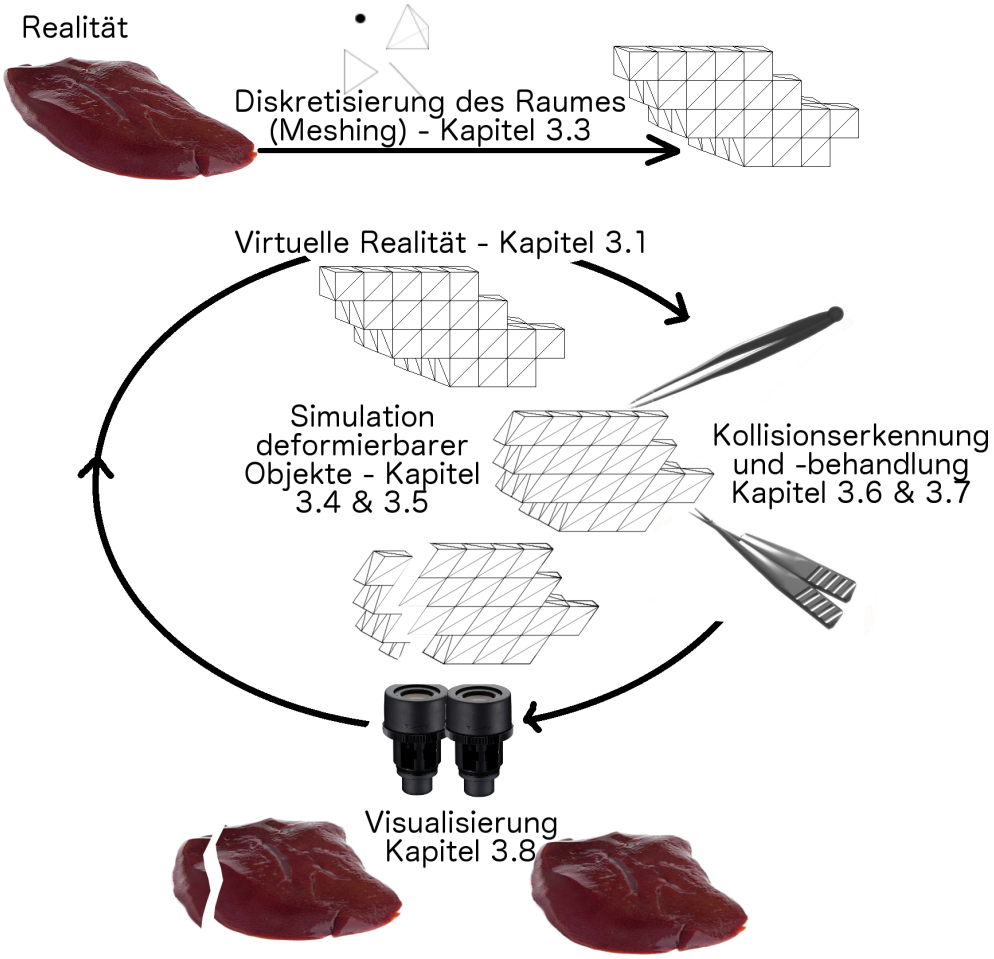


Abbildung 3.1: Schema einer exemplarischen Simulation deformierbarer Objekte. Zu Beginn wird der simulierte Raum diskretisiert. Die Simulation ist eine zeitliche Schleife, in welcher die Veränderungen der Körper auf der Basis von Kräften, welche als interne oder externe Kräfte, z. B. durch die äußere Interaktion des Benutzers, berechnet werden. Die Ausgabe, in diesem Fall das visuelle Feedback, richtet sich nach der Anatomie der menschlichen Sinne, um das Gefühl der Präsenz zu erzeugen (vgl. Abschnitt 3.1).

3.1 Virtuelle Realität

In der Fachliteratur existieren viele Definitionen des Begriffes VR. In [Ste92] wird als Schlüssel für die Definition von VR die Präsenz genannt. [Wag03] definiert

Präsenz als eine „[...] subjektive Empfindung, sich in einer bestimmten Umgebung zu befinden“. Des Weiteren wird die VR als „[...] Einrichtung, die auf eine künstliche Weise das Gefühl der Präsenz in einer bestimmten Umgebung erzeugt“ beschrieben. Erreicht wird dies, indem die Sinne des Menschen mit künstlichen Signalen stimuliert werden, die bestenfalls nicht von realen unterschieden werden können. Daraus folgt, dass sich die VR entsprechend der realen Welt verhalten muss.

Die *Realität* soll für die vorliegende Arbeit vereinfacht als die subjektive Wahrnehmung des Einzelnen von seiner Umwelt, von sich selbst in seiner Umwelt und von der Interaktion zwischen sich und seiner Umwelt verstanden werden. Dementsprechend wird die VR als eine in der Realität künstlich erschaffene Situation definiert, welche die subjektive Wahrnehmung des Einzelnen insofern überzeugt, als dass er diese Situation als real empfindet. Dem Benutzer müssen situationsabhängig in der VR dieselben Interaktionsmöglichkeiten wie in der Realität gegeben sein.

In [BB94] wird VR folgenderweise definiert: „Mit dem Begriff Virtual Reality wird meist die rechnergestützte Generierung eines möglichst perfekten sensorischen Abbildes unserer realen Umwelt assoziiert.“ Der Begriff VR wird für die vorliegende Arbeit ausschließlich auf computergenerierte VR beschränkt.

Echtzeit

Eine Voraussetzung an eine auf VR basierenden Anwendung ist die Einhaltung der Echtzeit. Der Benutzer darf bei der Interaktion mit der Anwendung keine Verzögerung oder Unterbrechung beim Ablauf wahrnehmen. Die Sinne des Menschen müssen mit der Häufigkeit an Informationen beliefert werden, für die sie ausgerichtet sind.

Das menschliche Auge besitzt eine Abtastrate von ca. 50 Hz. Nach dem Abtasttheorem von *H. Nyquist* können damit Frequenzen von bis zu 25 Hz aufgelöst werden. Dies bedeutet, dass die Echtzeit-Anwendung dem Benutzer mindestens 25 Bilder pro Sekunde generieren bzw. der Benutzer spätestens nach jeweils 40 ms ein neues Bild zu sehen bekommen muss (vgl. [gri05]).

In der vorliegenden Arbeit werden keine weiteren Sinne, z. B. der Tastsinn, beansprucht, daher beschränkt sich die Einhaltung der Echtzeit auf das visuelle Feedback. Das heißt, dass die Eingaben in das System durch den Benutzer, die Berechnungen der Simulation und die Visualisierung der Szene „schnell genug“ berechnet werden müssen, sodass alle 40 ms ein neues Bild ausgegeben werden kann. Ist

das gegeben, läuft das System in Echtzeit ab. Die Begriffe Echtzeit-System und Echtzeit-Anwendung werden synonym benutzt.

3.2 Klassifizierung von Simulationsmodellen

Simulationsmodelle können in zwei Gruppen eingeteilt werden: geometrisch und physikalisch basierte Simulationsmodelle (vgl. [Web09]). Deformationen bei geometrisch basierten Modellen basieren auf rein geometrischen Operationen. Physikalisch basierte Modelle unterliegen physikalischen Gesetzmäßigkeiten. Bei einem verschiebungsorientierten Ansatz zum Beispiel, wird die Position eines Elementes während der Simulation direkt gesetzt. Deshalb handelt es sich dabei um ein geometrisch basiertes Simulationsmodell. Wird im Gegensatz zu dem vorangegangenen Beispiel eine Kraft auf ein Element gegeben und daraus die Veränderung des Elementes auf Basis des zweiten newtonschen Gesetzes errechnet, handelt es sich um ein physikalisch basiertes Modell. In diesem Fall handelt es sich um einen kraftbasierter Ansatz (vgl. [gri05]).

Weiter wird in der Literatur zwischen den Methoden der deskriptiven und der physikalischen Modellierung unterschieden (vgl. [Sch01]).

Physikalische und Deskriptive Modellierung

Wurde ein Simulationsmodell auf der Basis physikalischer Modellierung erstellt, spiegeln die Parameter des Models physikalische Größen wider. Diese Größen müssen vorab in der realen Welt gemessen werden. Die Genauigkeit des Modells hängt davon ab, ob physikalische Werte aus der realen Welt extrahiert und wie genau diese gemessen werden können.

Die deskriptive Modellierung beruht im Gegensatz zu der physikalischen Modellierung nicht zwangsläufig auf physikalischen Werten. Die Parameter des Simulationsmodells entsprechen im Normalfall keinen in der realen Welt messbaren Größen. Das Verhalten, welches modelliert werden soll, wird beobachtet und anschließend wird ein Modell erstellt, welches dieses Verhalten approximiert.

Im Allgemeinen können physikalische Modelle beliebig genau gestaltet werden, vorausgesetzt die benötigten Werte sind verfügbar. Der Aufwand steigt mit der Anzahl der Parameter die berücksichtigt werden müssen. Dadurch steigt die Komplexität des Modells und somit auch die Rechenzeit. Im Hinblick auf Echtzeit

werden in der VR deshalb bevorzugt deskriptive oder vereinfachte physikalische Modelle eingesetzt.

Simulationen können in zwei weitere Kategorien eingeteilt werden: Lagrangesche und Eulersche. Lagrangesche Modelle simulieren Objekte, welche aus Punkten mit veränderbaren Positionen und weiteren Eigenschaften bestehen. Eulersche Simulationsmodelle berechnen Objekteigenschaften an stationären Punkten (vgl. [NMK+06]). Eulersche Simulationsmodelle werden bevorzugt bei der Simulation von Flüssigkeiten und Gasen verwendet.

Die vorliegende Arbeit beschränkt sich auf Lagrangesche Simulationsmodelle. Diese können weiter in gitterbasierte und gitterlose Verfahren unterteilt werden. Zu den gitterbasierten zählen kontinuumsmechanische Verfahren, wie die Finite-Elemente-Methode (FEM) und das Masse-Feder-Modell (MFM). Vertreter der gitterlosen Verfahren sind Partikelsysteme. Für einen Überblick über die historische Entwicklung sei auf das Survey [GM97] und für eine umfang- und detailreiche Einführung in den aktuellen Stand der Technik auf die Arbeiten [MSCT08], [NMK+06] und [gri05] verwiesen.

Viele Simulationsmodelle stellen vielversprechende Ergebnisse vor, sind jedoch mit hohen Rechenkosten verbunden. Diese eignen sich daher kaum für Echtzeitsimulationen. Auf der anderen Seite sind auch viele der trivialeren Modelle für interaktive Anwendungen ungeeignet, da sie instabil werden, wenn sie mit undefinierten Benutzereingaben konfrontiert werden. Die Ursache für dieses Problem findet sich in der numerischen Stabilität eines Simulationsmodells. Abschnitt 3.5 geht in Bezug auf numerische Integrationsverfahren darauf ein. Das Integrationsverfahren wirkt sich neben der Qualität der Simulationselemente direkt auf die Stabilität einer Simulation aus (Abschnitt 3.3).

3.3 Diskretisierung des Raumes

Im Folgenden soll der Raum als ein vom Menschen wahrnehmbares Objekt verstanden werden. In diesem Zusammenhang beschreibt die Diskretisierung des Raumes die Modellierung eines Objektes aus „kleineren Teilen“, welche von einem Computer gespeichert, weiterverarbeitet und ausgegeben werden können. Die Art der Diskretisierung hängt mit dem Simulationsmodell und Visualisierungskonzept zusammen. Bei einem MFM wird ein Objekt durch ein Simulationsgitter diskretisiert. In der deutschen Literatur wird auch der englische Begriff *Mesh* anstelle des Simulationsgitters verwendet. *J. Grimm* schreibt in [gri05], dass „die englische Bezeichnung *Mesh* (Netz, Masche)“aufgrund der meist unregelmäßigen Gitterstrukturen bei Simulation deformierbarer Objekte der „intuitivere

Begriff“ ist. Analog dazu wird die Erstellung eines Mesh als *Meshing* bezeichnet. Ein Mesh besteht aus Massepunkten, die als *Knoten* bezeichnet werden, und Verbindungselementen, mit denen physikalische Eigenschaften eines Objektes modelliert werden können. Die Oberfläche des Objektes wird meist durch Dreiecke approximiert, welche die Grundlage für die Visualisierung des Objektes bilden. Die Elemente, aus welchen sich das Mesh zusammensetzt, nennt man Primitive.

Ein bekanntes Meshing-Verfahren zweidimensionaler Objekte ist die Delaunay Triangulierung. Diese setzt voraus, dass ein Objekt durch eine Punktwolke approximiert ist. Die Punkte werden dabei zu einem Mesh aus Dreiecken verbunden. Dabei wird sichergestellt, dass kein vierter Punkt innerhalb des Umkreises eines Dreiecks liegt. Das Verfahren maximiert den kleinsten Innenwinkel des Mesh. Analog dazu wird für dreidimensionale Objekte die Umkugelbedingung für Tetraeder genommen.

Diese und weitere Lösungen für die Diskretisierung von zweidimensionalen und dreidimensionalen Objekten finden sich in der algorithmischen Geometrie, welche sich mit der Verarbeitung geometrischer Daten beschäftigt. Einen guten Überblick geben [BE92] und [BP00].

In der Spiele-Industrie werden dreidimensionale Körper von Grafik-Designern modelliert. Die modellierten Objekte können meist nicht direkt simuliert werden. Zum einen fehlen ihnen oft verschiedene Simulationsparameter (z. B. Masse), zum anderen bestehen sie häufig nur aus der Hülle des dreidimensionalen Objektes oder die Primitive, aus welchen die Objekte erstellt wurden, sind von schlechter Qualität (vgl. [MSCT08]). Mit Qualität sind Form und Größe eines Primitivs gemeint, z. B. seine Innenwinkel. Je nach Simulationsmodell wirkt sich das negativ auf die Genauigkeit, die Rechenzeit und die numerische Stabilität einer Simulation aus (vgl. [She02] und [LJ94]).

Tetraedernetze kommen häufig in physikalisch basierten Simulationen zum Einsatz. In den letzen zwei Jahrzehnten wurden viele Verfahren zur Generierung von Tetraedernetzen veröffentlicht. Viele der Methoden basieren auf der Delaunay Triangulierung. Beispiele hierfür sind [Wat81] und [ACSYD05]. In [MSCT08] wird ein trivialer Algorithmus für die Erstellung von Tetraedernetzen erörtert. Ausgehend von einem geschlossenen dreidimensionalen Dreiecksnetz, das die Oberfläche eines Objektes beschreibt, werden Partikel auf Basis eines einfachen Raytracing-Verfahrens innerhalb des Dreiecksnetzes platziert und anschließend wird ein Tetraedernetz mit Hilfe einer Delaunay Triangulierung erstellt

[MT03] erstellt im ersten Schritt einen Quader, der das geschlossene dreidimensionale Dreiecksnetz eines Objektes beinhaltet. Im zweiten Schritt wird jeder Qua-

der in jeweils fünf Tetraeder unterteilt. Dies ist die Mindestanzahl, durch die ein Quader mit einem Netz aus Tetraedern dargestellt werden kann (vgl. [CDM+02]). Abbildung 3.2 skizziert die Unterteilung eines Quaders durch Tetraeder.

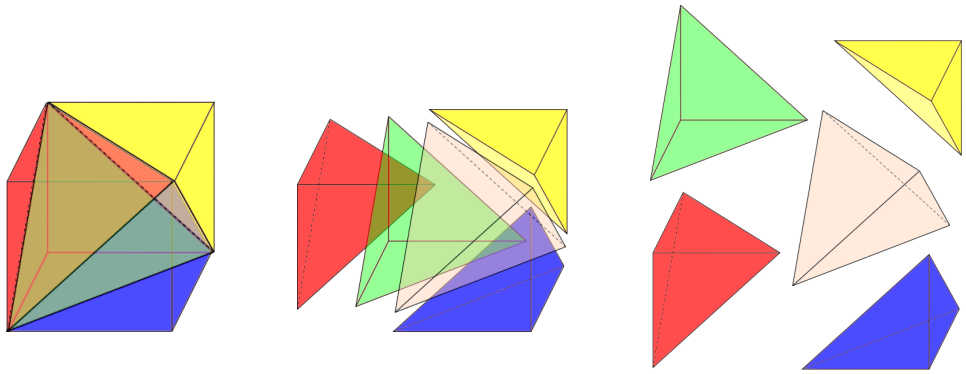


Abbildung 3.2: Minimale Unterteilung eines Quaders in Tetraeder.

Optional kann jeder Tetraeder, abhängig von der Schnittmenge, die er sich mit dem Dreiecksnetz teilt, in einem Zwischenschritt so unterteilt werden, dass er das Dreiecksnetz exakt ausfüllt. Dies wirkt sich jedoch negativ auf die Anzahl und Qualität der Tetraeder aus. Mit Hilfe eines Raytracing Verfahrens werden anschließend alle Tetraeder, die sich außerhalb des ursprünglichen Dreiecksnetzes befinden, gelöscht. Wurde der erwähnte Zwischenschritt nicht durchgeführt, können Zwangsbedingungen für die Tetraederhöhen und Kantenlängen bestimmt werden und durch Krafteinwirkung auf die äußeren Knoten die ursprüngliche Oberfläche iterativ approximiert werden.

In [SWT06] wird ein Verfahren vorgestellt, das auch auf unstrukturierte und unvollständige Dreiecksmengen angewendet werden kann. Als Basis dient ein vorzeichenbehaftetes Distanzfeld. Dabei haben Voxel[1] im Inneren des Objektes eine negative Distanz. Um zu entscheiden, ob sich Voxel innerhalb oder außerhalb befinden, werden verschiedene Verfahren, abhängig von der Vollständigkeit des Netzes vorgeschlagen. Existieren nur wenige Löcher im Netz, kann durch das Verwenden mehrerer Raytraces aus verschiedenen Richtungen entschieden werden, auf welcher Seite sich ein Voxel befindet. Bei stark unstrukturierten Dreiecksmengen werden die Entscheidungen auf der Basis von Wahrscheinlichkeiten berechnet. Anschließend wird das Objekt ähnlich wie in [MT03] unterteilt.

Einfache Körper können durch analytische Algorithmen erstellt werden. Dadurch kann eine sehr hohe Qualität der Tetraeder erreicht werden. Für komplexere Objekte sind solche Methoden ungeeignet, da die Algorithmen aufwendig zu erstellen sind.

[1] Analog zum *Picture Element (Pixel)*, nennt sich ein räumliches Pixel *Volumetric Pixel (Voxel)*. Voxel werden häufig bei der Visualisierung medizinischer Datensätze eingesetzt.

3.4 Masse-Feder-Modell

In medizinischen Simulatoren werden vor allem Simulationsmodelle eingesetzt, die auf dem MFM oder der FEM basieren (vgl. [BHS03]). Die FEM ist im Allgemeinen genauer, aber auch mit höherem Rechenaufwand als das MFM verbunden. Da bei der Echtzeitsimulation das realistische Feedback an den Benutzer entscheidend ist und nicht die exakte Berechnung der Deformation des simulierten Gewebes, eignet sich für die vorliegende Arbeit ein MFM. MFMe sind schnell und in der Lage für viele Materialien realistische Ergebnisse zu erzeugen, unter anderem für viskoelastisches Gewebe (vgl. [BSB+01]). Aus diesen Gründen wurde für die vorliegende Arbeit ein MFM verwendet.

Bei einem MFM handelt es sich um ein deskriptives Simulationsmodell. Das elastische Verhalten der Federn im klassischen MFM beruht auf dem Hookeschen Gesetz. Daher wird es in der Fachliteratur häufig als physikalisches Modell klassifiziert (vgl. [GM97]). Die Federkonstanten bilden jedoch keine analogen Materialkonstanten ab, sondern werden in einem iterativen Prozess angepasst, bis das Materialverhalten bestmöglich abgebildet wird (vgl. [Web09]). In [gri05] wird das MFM treffend als kraftbasiertes, physikalisch orientiertes und deskriptives Modell klassifiziert.

Ausgangspunkt für das MFM ist ein Mesh bestehend aus n Knoten p_i (aus $p_0, \ldots, p_{n-1}$). Die Masse des Objektes wird meist gleichmäßig auf die Knoten verteilt, wodurch jeder Knoten die Masse m besitzt. Weiter besitzt jeder Knoten p_i eine Position $\vec{p}_i$ und eine Geschwindigkeit $\vec{v}_i$. Knoten besitzen neben ihrer aktuellen Position und Masse auch eine Startposition im Mesh, aus der sich die Längen der Federn in Ruhelage berechnen lassen. Diese Längen werden im Folgenden als Nulllängen bezeichnet[2].

Die Summe aller Kräfte, die auf den Knoten wirken, wird als $\vec{F^i}$ bezeichnet. Die physikalischen Strukturen innerhalb des Mesh werden durch masselose Federn s_{ij} beschrieben, die jeweils zwischen den Knoten p_i und p_j wirken. Die Steifigkeit der Feder wird durch die Konstante $k_{s_{ij}}$ festgelegt und die Länge der Feder in Ruhelage beträgt $L^0_{s_{ij}}$. Nach dem Hookeschen Gesetz ist die Federkraft proportional zu ihrer Auslenkung. Dadurch ergibt sich zwischen der Federkraft $F_{s_{ij}}$ und den Positionen $\vec{p}_i$ und $\vec{p}_j$ der Knoten folgender Zusammenhang:

[2] Analog dazu werden Nullflächen in Bezug auf Dreiecke und Nullvolumen in Bezug auf Tetraeder verwendet.

$$\vec{F}^{i}_{s_{ij}} = k_{s_{ij}} \frac{\vec{p}_j - \vec{p}_i}{|\vec{p}_j - \vec{p}_i|}(|\vec{p}_j - \vec{p}_i| - L^0_{s_{ij}}) = -\vec{F}^{j}_{s_{ij}}. \tag{3.1}$$

$\vec{F}^{i}_{s_{ij}}$ ist die Kraft, die s_{ij} auf den Knoten i und $-\vec{F}^{j}_{s_{ij}}$ die Kraft, die s_{ij} auf den Knoten j ausübt. Die Kräfte, die auf i und j wirken, summieren sich zu null auf, wodurch der Impulserhaltungssatz erfüllt wird. Weiter ist aus der Gleichung bereits die erste Schwachstelle eines MFM zu erkennen. Die Konstante $k_{s_{ij}}$ muss vorsichtig gewählt werden, da sie die Kraft, welche auf die Knoten wirkt, direkt beeinflusst und damit maßgeblich für die Dynamik des Systems verantwortlich ist. Abschnitt 3.5 geht explizit auf dieses Problem ein.

Zusätzlich zu den internen Kräften, die ein Knoten durch die Federn, mit denen er verbunden ist, erfährt, können externe Kräfte, z. B. Gravitation, auf einen Knoten wirken. Diese werden mit $\vec{F^i_{ext}}$ zusammengefasst. Berücksichtigt werden muss auch eine der Geschwindigkeit eines Knotens entgegengerichtete Dämpfungskraft $\vec{F^i_d} = d \cdot \vec{v_i}$. Diese existiert in der realen Welt z. B. aufgrund des Luftwiderstandes als Umgebungsdämpfung. Die endgültige Kraft F^i ergibt sich durch Aufsummieren aller Kräfte (vgl. [gri05]):

$$\vec{F^i} = \sum_j \vec{F}^{i}_{s_{ij}} + \vec{F}^{i}_{ext} + \vec{F}^{i}_{d}. \tag{3.2}$$

Um das Verhalten verschiedener Materialien besser zu modellieren wurde das MFM in diversen Arbeiten erweitert. Um mit einem zweidimensionalen strukturierten Gitternetz Kleidung zu simulieren, werden in [PP96] neben den *normalen* Federn für die Elastizität des Materials zusätzliche Federn verwendet. Zum einen wird das Modell um *flexion*-Federn erweitert, wodurch das Gewebe eine Biegesteifigkeit bekommt, und zum anderen werden diagonale *shearing*-Federn eingeführt, wodurch auch Scherungen des Materials simuliert werden können (siehe Abbildung 3.4).

Um steiferes Material zu simulieren, können höhere Steifigkeitskonstanten verwendet werden. Da dies ungünstig ist, wie Abschnitt 3.5 zeigt, wird in [PP96] zusätzlich eine künstliche Korrektur der ausgelenkten Federn bei Überschreitung einer bestimmten Länge vorgeschlagen. [gri05] vergleicht diesen und weitere Ansätze, die auf der Idee der Längenkorrektur ansetzen, und kommt zu dem Ergebnis, dass die Steifigkeit des Materials dadurch nur „leicht erhöht“ werden kann.

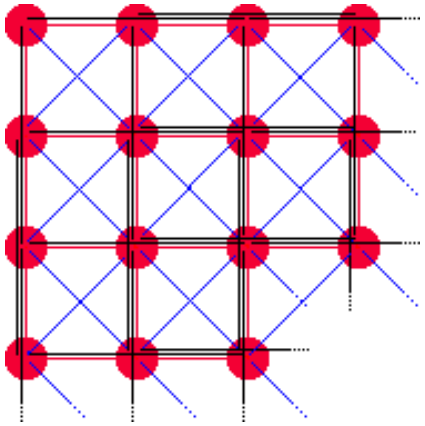


Abbildung 3.3: Verschiede Federtypen nach [PP96]: *normale* (rot); *flexion* (schwarz); *shearing* (blau).

Potentielle Energie

Die in einer eindimensionalen Feder durch Dehnung oder Kompression entstehende potentielle Energie E_L lässt sich durch Integration der Federkraft errechnen. Die Auslenkung der Feder ist mit $y = C(\vec{p}_0, \vec{p}_1) = |\vec{p}_j - \vec{p}_i| - L^0_{s_{ij}}$ gegeben. Daraus folgt:

$$E_L(\vec{p}_0, \vec{p}_1) = -\int_0^y k_{s_{ij}} \cdot y \mathrm{d}y = \frac{1}{2} k_{s_{ij}} y^2. \tag{3.3}$$

In [THMG04] wird die Idee der potentiellen Energie aufgegriffen. Der Ansatz verwendet jedoch nicht nur die in Gleichung 3.3 hergeleitete potentielle Energie einer eindimensionalen Feder zwischen zwei Knoten, sondern verallgemeinert den Ansatz und berechnet für die Zwangsbedingung $C(\vec{p}_0, \ldots, \vec{p}_{n-1}) = 0$ zwischen den n Knoten die potentielle Energie $E(\vec{p}_0, \ldots, \vec{p}_{n-1}) = \frac{1}{2} k C^2$. Die negative Ableitung von E entspricht dann der resultierenden Kraft $F_i(\vec{p}_0, \ldots, \vec{p}_{n-1})$ auf die Knoten:

$$\vec{F}_i(\vec{p}_0, \ldots, \vec{p}_{n-1}) = -\frac{\partial E}{\partial \vec{p}_i} = -kC\frac{\partial C}{\partial \vec{p}_i}. \tag{3.4}$$

Aus Gleichung 3.4 ist zu erkennen, dass die Kraft $\vec{F}_i(\vec{p}_0, \ldots, \vec{p}_{n-1})$ auf den Knoten p_i stets in die Richtung wirkt, in die die potentielle Energie E minimiert wird.

[THMG04] integriert eine Normierung der Kräfte in seinen Ansatz, wodurch Gleichung 3.3 zu Gleichung 3.5 erweitert wird:

$$E_L(\vec{p}_i, \vec{p}_j) = \frac{1}{2} k_D \left(\frac{|\vec{p}_j - \vec{p}_i| - L^0_{s_{ij}}}{L^0_{s_{ij}}} \right)^2. \tag{3.5}$$

Durch Krafteinwirkung verändern sich die Positionen der einzelnen Knoten im Mesh. Die potentielle Energie aus Gleichung 3.5 versucht den anfänglichen Abstand zwischen zwei Knoten zu erhalten und wirkt dadurch längenerhaltend auf das System. Analog dazu entsteht aus der Differenz der ursprünglichen Fläche zwischen drei Knoten und der Fläche, die sich aus den aktuellen Positionen der Knoten ergibt, potentielle Energie, die flächenerhaltend auf das Mesh wirkt:

$$E_A(\vec{p}_i, \vec{p}_j, \vec{p}_k) = \frac{1}{2} k_A \left(\frac{\frac{1}{2}|(\vec{p}_j - \vec{p}_i) \times (\vec{p}_k - \vec{p}_i)| - A_0}{A_0} \right)^2. \tag{3.6}$$

Zuletzt wird in [THMG04] die Veränderung des Volumens eines Tetraeders betrachtet, der durch vier, nicht kollineare Knoten beschrieben wird. Diese Veränderung ergibt sich aus der Differenz zwischen dem aktuellen und dem Anfangsvolumen eines Tetraeders. Das aktuelle Volumen wird auf Basis der aktuellen Position seiner Knoten bestimmt. Nach Gleichung 3.7 kann dadurch die potentielle Energie berechnet werden. Insbesondere bekommen invertierte Tetraeder ein negatives Volumen, wodurch entgegenwirkende Kräfte berechnet werden können. Die Kräfte werden nach Gleichung 3.4 berechnet und wirken sich auf das System volumenerhaltend aus.

$$E_V(\vec{p}_i, \vec{p}_j, \vec{p}_k, \vec{p}_l) = \frac{1}{2} k_V \left(\frac{\frac{1}{6}(\vec{p}_j - \vec{p}_i) \cdot ((\vec{p}_k - \vec{p}_i) \times (\vec{p}_l - \vec{p}_i)) - V_0}{\tilde{V}_0} \right)^2. \tag{3.7}$$

Wird das Modell auf sehr flache Objekte angewendet, kann $V_0 = 0$ nicht ausgeschlossen werden. Daher wird in [THMG04] in diesem speziellen Fall $\tilde{V}_0 = \frac{\sqrt{2}}{\tilde{l}}$ und in den anderen Fällen $\tilde{V}_0 = V_0$ definiert. Dabei ist $\tilde{l}$ die durchschnittliche Kantenlänge der Tetraeder des Mesh.

Des Weiteren wird in [THMG04] eine Dämpfung direkt in das System integriert. Dadurch ergibt sich die Gesamtkraft F_i, durch $\vec{p}_0, \ldots, \vec{p}_{n-1}$ und $\vec{v}_0, \ldots, \vec{v}_{n-1}$ aller Knoten, für die eine Zwangsbedingung in Abhängigkeit zu p_i definiert ist:

$$\vec{F}_i(\vec{p}_0, \ldots, \vec{p}_{n-1}, \vec{v}_0, \ldots, \vec{v}_{n-1}) = \Big(- kC - k_d \sum_{0<j<n} \frac{\partial C}{\partial \vec{p}_j} \vec{v}_j \Big) \frac{\partial C}{\partial \vec{p}_i}. \tag{3.8}$$

Für die vorliegende Arbeit wurden Teile des Algorithmus verwendet (siehe Abschnitt 4.2). Erst kürzlich wurde in [Diz12] ein Verfahren veröffentlicht, bei welchem sich die Erhaltung des Volumens bei hohen Kräften stabiler verhält, indem die Steifigkeitskonstanten benachbarter Tetraeder dynamisch angepasst werden.

Für eine Auflistung bisheriger Anwendungen und die historische Entwicklung von MFMen, beginnend bei der Animation von Gesichtszügen durch verbundene Punkte im Jahr 1981 (vgl. [PB81]), sei auf [Neu09] verwiesen.

3.5 Diskretisierung der Zeit

Veränderungen des Raumes finden im Laufe der Zeit statt. Um diese Veränderungen berechnen zu können, muss die Zeit selbst, welche kontinuierlich ist, diskretisiert werden. In einem Echtzeit-System, bei dem es nur auf das visuelle Feedback an den Benutzer ankommt und ein verschiebungsorientiertes Modell zum Einsatz kommt, reicht es im Allgemeinen, 25 Einheiten pro Sekunde zu verwenden. Dies genügt, um dem menschlichen Auge ein kontinuierliches Bild vermitteln zu können. Die Einheit Δt zwischen einem Zeitpunkt t und dem darauffolgenden Zeitpunkt $t + \Delta t$ wird im Folgenden als Zeitschritt definiert. In einem kraftbasierten Simulationsansatz müssen physikalische Kräfte in Bewegungen umgewandelt werden. Vorausgesetzt, dass sich die Masse über die Zeit nicht ändert, lässt sich die Position $\vec{p}_i$ eines Knotens zum Zeitpunkt t aus dem zweiten Newtonschen Gesetz berechnen:

$$\frac{\partial^2 \vec{p}_i(t)}{\partial t^2} = \frac{\vec{F}_i(t)}{m_i}. \tag{3.9}$$

Bei Gleichung 3.9 handelt es sich um eine Differentialgleichung zweiter Ordnung. Da die Position $\vec{p}_i$ und die Geschwindigkeit $\vec{v}_i$ zum Zeitpunkt t bekannt sind, handelt es sich um ein Anfangswertproblem. Um daraus die Postion $\vec{p}_i(t + \Delta t)$ zu berechnen, werden numerische Integrationsverfahren verwendet. Diese Verfahren können im Allgemeinen nicht auf eine Differentialgleichung (DGL) zweiter Ordnung angewendet werden. Deshalb muss die Gleichung 3.9 erst in zwei DGLen erster Ordnung transformiert werden:

$$\frac{\partial \vec{p}_i(t)}{\partial t} = \vec{v}_i(t), \tag{3.10}$$

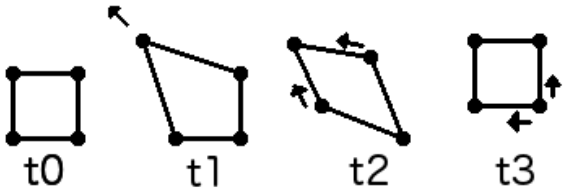
$$\frac{\partial \vec{v}_i(t)}{\partial t} = \frac{\vec{F}_i(t)}{m_i}. \tag{3.11}$$

Numerische Integration

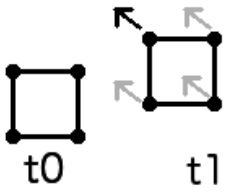
Bei Integrationsverfahren wird zwischen expliziten und impliziten Integrationsverfahren unterschieden. Bei den expliziten Integrationsverfahren breiten sich Kräfte, die zum Zeitpunkt t auf einen Knoten wirken, während Δt nur auf die Knotennachbarn aus, die sich ein Verbindungselement mit dem Knoten teilen (Abbildung 3.4(a)). Im Gegensatz dazu werden bei der impliziten Integration die Auswirkungen in jedem Zeitschritt für das gesamte Mesh berechnet (Abbildung 3.4(b)). Welches Verfahren zum Einsatz kommt, muss anhand der Kriterien: Genauigkeit, Stabilität und Laufzeit entschieden werden. Auf implizite Integrationsverfahren wird nicht eingegangen, da sie für die Simulation von Tetraedernetzen in Echtzeit ungeeignet sind (vgl. [MMDJ01]).

Im Folgenden wird zuerst das klassische Euler-Verfahren vorgestellt. Danach werden die Verlet-Methode und darauf aufbauend die Velocity-Verlet-Methode vorgestellt. Letztere wurde in der vorliegenden Arbeit verwendet.

Alle in dieser Arbeit vorgestellten Verfahren bilden lediglich eine Einführung in numerische Methoden. Für eine ausführliche Übersicht expliziter und impliziter numerischer Lösungsverfahren gewöhnlicher Differentialgleichungen in Bezug auf Simulationsmodelle deformierbarer Objekte sei auf [HES03] verwiesen.



(a) Explizite Integration. Interne Kräfte breiten sich in t auf die Nachbarn eines Knotens aus.



(b) Implizite Integration. Kräfte breiten sich in t auf das gesamte Mesh aus.

Abbildung 3.4: Integrationsverfahren

Euler-Methode

Das einfachste numerische Verfahren zur Lösung von Differentialgleichungen erster Ordnung ist die klassische Euler-Methode. Das Verfahren nimmt an, dass die Kraft in Δt konstant bleibt. Mit der Position $\vec{p_i}(t)$ und der Geschwindigkeit $\vec{v_i}(t)$ kann $\vec{p_i}(t + \Delta t)$ angenähert werden:

$$\vec{p_i}(t + \Delta t) = \vec{p_i}(t) + \Delta t \cdot \vec{v_i}(t). \tag{3.12}$$

Die Geschwindigkeit für den nächsten Zeitschritt $\vec{v_i}(t + \Delta t)$ wird analog zu Gleichung 3.12 abgeschätzt:

$$\vec{v_i}(t + \Delta t) = \vec{v_i}(t) + \Delta t \cdot \frac{\vec{F_i}(t)}{m_i}. \tag{3.13}$$

An den Gleichungen 3.12 und 3.13 lässt sich gut erkennen, dass es sich beim klassischen Euler um ein explizites Integrationsverfahren handelt, da sich die Werte für $t+\Delta t$ direkt aus den Werten zum Zeitpunkt t berechnen lassen. Die Genauigkeit expliziter Integrationsverfahren hängt mit der Länge des gewählten Zeitschrittes zusammen. Je größer der Zeitschritt, desto größer wird auch der numerische Fehler, wie Abbildung 3.5 verdeutlicht.

Um den numerischen Fehler der Euler-Methode zu spezifizieren, wird die Taylor-Entwicklung an der Stelle $t + \Delta t$ angeschaut:

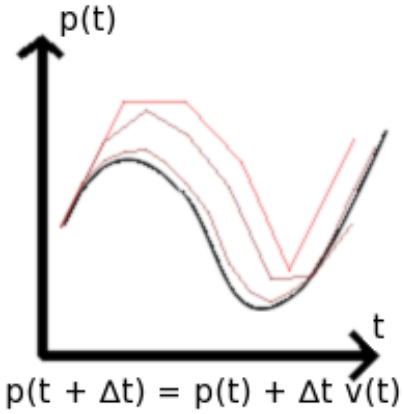


Abbildung 3.5: Der numerische Fehler bei der Euler-Methode ist abhängig von der Länge des gewählten Zeitschritts Δt (vgl. [WB01]).

$$\vec{p}_i(t+\Delta t) = \vec{p}_i(t) + \Delta t \cdot \frac{\partial \vec{p}_i(t)}{\partial t} + \frac{\Delta t^2}{2} \cdot \frac{\partial^2 \vec{p}_i(t)}{\partial t^2} + \cdots + \frac{\Delta t^n}{n!} \cdot \frac{\partial^n \vec{p}_i(t)}{\partial t^n}. \tag{3.14}$$

Werden alle Terme in Gleichung 3.14 außer den ersten beiden außer Acht gelassen, erhält man die Euler-Methode. Der Fehler des Verfahrens ist demnach von der Ordnung $O(\Delta t^2)$. Das Euler-Verfahren ist ungenau und deshalb nur eingeschränkt für interaktive Echtzeit-Simulationen einsetzbar.

Verlet

Ein exakteres numerisches Integrationsverfahren ist die nach dem Erfinder der Methode *Loup Verlet* benannte Verlet-Methode (vgl. [Ver67]). Das Verlet-Verfahren basiert darauf, Informationen aus vergangenen Zeitschritten zu verwenden, um eine bessere Abschätzung geben zu können. Die Methode kann hergeleitet werden, indem zwei Taylor-Entwicklungen dritten Grades an der Position $\vec{p}_i(t)$ vorgenommen werden, und zwar eine zum Zeitpunkt $t+\Delta t$ und eine zum Zeitpunkt $t-\Delta t$.

$$\vec{p}_i(t+\Delta t) = \vec{p}_i(t) + \Delta t \cdot \frac{\partial \vec{p}_i(t)}{\partial t} + \frac{\Delta t^2}{2} \cdot \frac{\partial^2 \vec{p}_i(t)}{\partial t^2} + \frac{\Delta t^3}{3} \cdot \frac{\partial^3 \vec{p}_i(t)}{\partial t^3} + O(\Delta t^4), \tag{3.15}$$

$$\vec{p}_i(t-\Delta t) = \vec{p}_i(t) - \Delta t \cdot \frac{\partial \vec{p}_i(t)}{\partial t} + \frac{\Delta t^2}{2} \cdot \frac{\partial^2 \vec{p}_i(t)}{\partial t^2} - \frac{\Delta t^3}{3} \cdot \frac{\partial^3 \vec{p}_i(t)}{\partial t^3} + O(\Delta t^4). \quad (3.16)$$

Addiert man die Gleichungen 3.15 und 3.16 ergibt sich die Position $\vec{p}_i(t+\Delta t)$ wie folgt:

$$\vec{p}_i(t+\Delta t) = 2\vec{p}_i(t) - \vec{p}_i(t-\Delta t) + \Delta t^2 \cdot \frac{\partial^2 \vec{p}_i(t)}{\partial t^2} + O(\Delta t^4). \quad (3.17)$$

Aufgrund des zweiten Newtonschen Gesetzes (Gleichung 3.9) kann $\Delta t^2 \cdot \frac{\partial^2 \vec{p}_i(t)}{\partial t^2}$ zum Zeitpunkt t bestimmt werden, wodurch die neue Position $\vec{p}_i(t+\Delta t)$ schnell berechnet werden kann. Aus Gleichung 3.17 ist zu erkennen, dass der numerische Fehler nur noch der Ordnung $O(\Delta t^4)$ ist, da die kubischen Ableitungen wegfallen. Die Einfachheit, die Effizienz und die höhere Genauigkeit sind die Vorteile dieses Algorithmus im Vergleich zum Euler-Verfahren.

Ein Nachteil des Algorithmus ist jedoch, dass die Geschwindigkeit durch das Verlet Verfahren nicht direkt berechnet wird. Diese wird abgeschätzt, indem die Positionen $\vec{p}_i(t-\Delta t)$ und $\vec{p}_i(t+\Delta t)$ als konstant vorausgesetzt werden:

$$\vec{v}_i(t+\Delta t) = \frac{\vec{p}_i(t+\Delta t) - \vec{p}_i(t-\Delta t)}{2\Delta t}. \quad (3.18)$$

Der numerische Fehler bei der Berechnung der Geschwindigkeit ist durch Gleichung 3.18 nur noch $O(\Delta t^2)$. Werden die Geschwindigkeiten für keine weiteren Berechnungen in der Simulation benötigt und wird der Zeitschritt klein genug gewählt, erzeugt das Verfahren bei geringem Rechenaufwand sehr genaue Werte.

Velocity Verlet

Das Velocity-Verlet-Verfahren (VVV) ist die Erweiterung des Verlet-Algorithmus und verhält sich durch die Minimierung der Rundungsfehler auch bei größeren Zeitschritten stabil (vgl. [SABW82]). Die Berechnung des VVV wird stufenweise durchgeführt. Dabei wird die Geschwindigkeit zum Zeitpunkt $t+\Delta t$ anhand der Beschleunigung zum Zeitpunkt t und $t+\Delta t$ ermittelt, wodurch die Näherung der

Geschwindigkeit im Vergleich zum Verlet-Verfahren genauer wird. Die Positionen $\vec{p}_i(t + \Delta t)$ können mit den Werten zum Zeitpunkt t berechnet werden, da die Geschwindigkeiten $\vec{v}_i(t + \Delta t)$ erst im nächsten Zeitschritt in die Integrationsrechnung einfließen. Die aktuelle Beschleunigung $\vec{a}_i(t) = \frac{\vec{F}_i(t)}{m_i}$ muss für die Berechnungen in $t + \Delta t$ gespeichert werden:

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + (\vec{a}_i(t) + \vec{a}_i(t + \Delta t))\frac{\Delta t}{2}, \tag{3.19}$$

$$\vec{p}_i(t + \Delta t) = \vec{p}_i(t) + \vec{v}_i(t + \Delta t) \cdot \Delta t + \vec{a}_i(t + \Delta t) \cdot \frac{\Delta t^2}{2}. \tag{3.20}$$

Numerische Stabilität

Implizite Verfahren sind im Gegensatz zu expliziten Integrationsverfahren unbedingt stabil. Unter der Berücksichtigung der Kräfte im Zeitpunkt t, welche sich über das gesamte Mesh ausbreiten, werden die Positionen aller Knoten im Mesh zum Zeitpunkt t berechnet und der Gleichgewichtszustand der Kräfte eingestellt. Bei expliziten Verfahren können durch die diskreten Zeitschritte Fehler im Iterationsschritt entstehen, die sich von einem auf den nächsten Zeitschritt akkumulieren. Dadurch wird ein System instabil und die Positionen der Knoten gehen gegen $\pm\infty$. Verringern sich die Fehler von einem auf den nächsten Zeitschritt ist ein System stabil (vgl.[gri05]).

In [Del98] wird die kritische Steifigkeit k_c eingeführt, ab welcher ein System divergiert. Diese verhält sich zu der Länge des Zeitschrittes und der Masse m_i wie folgt:

$$k_c \approx \frac{m_i}{\pi^2 \Delta t^2} t. \tag{3.21}$$

Folglich kann die Stabilität eines MFMs erhöht werden, indem die Steifigkeitskonstante verkleinert, die Anzahl der Knoten verringert (mehr Masse pro Knoten) oder der Zeitschritt verkleinert wird. Da sich kleinere Zeitschritte jedoch direkt auf die Rechenzeit auswirken und Masse und Steifigkeitskonstanten meist die simulierten Materialkonstanten repräsentieren, ist die Anpassung der Werte nicht

trivial. In [Shi05] werden Angaben für die Wahl der maximalen Schrittlänge und der optimalen Dämpfung gegeben. [BLo05] gibt für die Dämpfung, wie sie in Gleichung 3.2 eingeführt wurde, eine obere und eine untere Grenze an, zwischen der beim Einsatz der Euler-Methode als numerisches Integrationsverfahren ein stabiles System garantiert wird:

$$2\sqrt{m_i k} \leq d_i \leq \frac{|v_i \frac{m_i}{\Delta t} + F_i|}{v_i}. \tag{3.22}$$

In [SGT09] wird ein iteratives Verfahren für eine dynamische Dämpfung vorgestellt, welche unabhängig vom verwendeten Simulationsmodell und Integrationsverfahren ist.

Ein realistisches, stabiles MFM wird nicht zuletzt durch Erfahrung und Anpassung der einzelnen Werte nach dem „Trial & Error “ Prinzip in einem iterativen Prozess erstellt.

3.6 Kollisionserkennung

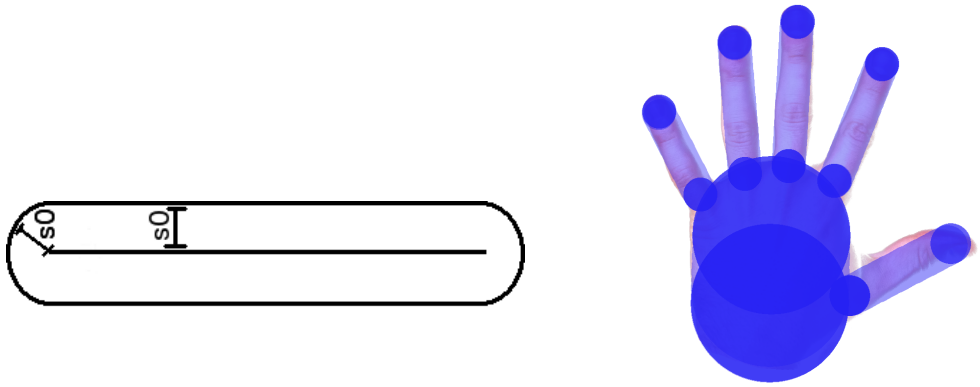
Die Kollisionserkennung (engl. Collision Detection) ist ein geometrisches Problem, welches entscheidet, ob zwei Objekte miteinander kollidieren. Bei der Simulation deformierbarer Objekte muss zusätzlich auf Selbstkollisionen geachtet werden. Dabei entscheidet die Kollisionserkennung, an welcher Stelle und wie tief Objekte ineinander eingedrungen sind. Es wird zwischen diskreter und kontinuierlicher Kollisionserkennung unterschieden. Im Gegensatz zu der diskreten Kollisionserkennung beschäftigt sich die kontinuierliche Kollisionserkennung mit Kollisionen, die zwischen zwei diskreten Zeitpunkten $t - \Delta t$ und t stattfinden. Falls nicht explizit kenntlich gemacht, ist im Folgenden stets die diskrete Kollisionserkennung gemeint.

Bei der diskreten Kollisionserkennung werden Objekte zum Zeitpunkt t auf Überschneidungen getestet. Werden solche erkannt, müssen die für die Auflösung der Kollisionen benötigten Werte berechnet und gespeichert werden. Falls keine Überschneidungen erkannt werden, kann nicht uneingeschränkt davon ausgegangen werden, dass in Δt keine Kollisionen stattgefunden haben. Abhängig von der Länge des Zeitschrittes und der Geschwindigkeit, mit der sich Objekte in der Simulation bewegen, kann es vorkommen, dass sich Objekte weder in $t - \Delta t$ noch in t überschneiden, aber während Δt vollständig durchdringen. Um das zu vermeiden,

muss je nach Rechenzeit die Zeitschrittlänge angepasst oder eine kontinuierliche Kollisionserkennung implementiert werden. Welcher Algorithmus zum Einsatz kommt, muss im Einzelfall entschieden werden.

Veranschaulichen kann man das am Beispiel eines Fußballes, der aufgrund der Gravitation aus einer unbestimmten Höhe auf den Boden fällt. Dieser befindet sich zum Zeitpunkt $t - \Delta t$ über dem Boden und zum Zeitpunkt t unterhalb des Bodens. In diesem Fall muss lediglich erkannt werden, auf welcher Seite der Ebene sich der Ball zum Zeitpunkt t befindet, da der Fußball unterhalb der Bodenebene physikalisch ausgeschlossen werden kann. Werden jedoch mehrere Objekte simuliert, welche alle miteinander kollidieren können, muss ein geeigneter Zeitschritt gewählt werden, um Kollisionen erkennen zu können. Dabei kann die Kollisionserkennung schnell zeitintensiv werden. Werden n Objekte simuliert, muss bei einem trivialen Ansatz das erste Objekt auf Kollisionen mit $n-1$ Objekten getestet werden, das zweite auf Kollisionen mit $n-2$ und so weiter. Die Laufzeit beträgt demnach $O(n^2)$.

Hinzu kommt die Laufzeit der einzelnen Erkennungsalgorithmen, die die exakten Kollisionsinformationen, wie Eindringtiefe und Kollisionspunkt, -linie oder -fläche, berechnen[3]. Diese ist von der Komplexität der Geometrie der einzelnen Objekte abhängig. Handelt es sich dabei um einfache geometrische Körper, z. B. Kugeln, Zylinder, Quader etc., sind die Algorithmen trivial. Je komplexer ein Objekt ist, desto aufwendiger sind die Berechnungen der Kollisionserkennung. Komplexere Körper werden daher durch einfachere Körper approximiert. In diesem Fall nennen sich die einfacheren Körper Hüllkörper. Auch hier wird im Folgenden das gebräuchlichere englische Wort *Bounding Volumes* benutzt. Analog dazu nennen sich Hüllkugeln *Bounding Spheres*, Hüllzylinder *Bounding Cylinders* etc. Im Folgenden wird zum einen auf die geometrischen Eigenschaften von abgerundeten *Bounding Cylinders* eingegangen, sogenannten *Bounding Capsules*. Zum anderen wird die *Axis Aligned Bounding Box (AABB)* vorgestellt. Dabei handelt es sich um einen an den Achsen ausgerichteten Quader. Sowohl AABB als auch *Bounding Capsules* wurden in der vorliegenden Arbeit bei der Kollisionserkennung verwendet.



(a) Der Abstand zur Außenhülle eines BC von seiner Mittelachse ist immer derselbe.

(b) Approximation eines Gegenstandes mit BC.

Abbildung 3.6: Bounding Capsules

Bounding Capsule

Bei einer BC handelt es sich um einen Zylinder, der an seinen beiden Enden jeweils eine Halbkugel besitzt. Der Radius der Halbkugel hat die Länge des Abstandes der Außenhülle des Zylinders zu seiner Mittelachse. Somit hat jeder Punkt auf der Außenhülle der BC denselben Abstand zur Mittelachse. Die Abbildung 3.6(a) veranschaulicht die Eigenschaften und Abbildung 3.6(b) zeigt die Approximation eines Gegenstandes durch BCs. Die Kollisionserkennung lässt sich dadurch so vereinfachen, dass Elemente nur noch auf den Abstand zur Achse des Zylinders getestet werden müssen. Ist der Abstand kleiner als der Radius des Zylinders, liegt eine Kollision vor. Bei der Kollisionserkennung zwischen zwei BCs wird der kürzeste Abstand zweier Linien untersucht. Bei der Kollisionserkennung zwischen einer Dreiecksfläche mit einem BC werden die Dreiecke auf ihren Abstand zur Linie untersucht (vgl. [Eri05]).

Axis Aligned Bounding Boxes

Bei einer AABB handelt es sich im dreidimensionalen Raum um einen Quader. Die Kanten des Quaders sind jeweils parallel zu einer der drei Achsen des Koordinatensystems. Abbildung 3.6 veranschaulicht den Einsatz einer AABB als Approximation eines Gegenstandes im zweidimensionalen Raum. AABB werden

[3]Für eine Einführung in die mathematischen Grundlagen der Kollisionserkennung, die für die vorliegende Arbeit benötigt werden, wird auf Anhang A verwiesen. Behandelt werden baryzentrische Koordinaten, Kollisionen zwischen Punkten und Tetraedern sowie Schnittpunkte zwischen Dreiecken und Linien.

häufig in der *Broad-Phase* (Beschreibung folgt im nächsten Absatz) eingesetzt, da Kollisionen zwischen AABB an ihren Koordinaten abgelesen werden können und wenig Speicherplatz benötigen. Eine AABB ist durch ihre kleinste und ihre größte Koordinate definiert. Das heißt, solange sich zwei AABB entlang einer Achse nicht überschneiden, findet keine Kollision statt.

Abbildung 3.7: Axis Aligned Bounding Box

Broad- und Narrow-Phase

Um die Kollisionserkennung zu beschleunigen, wird diese häufig in mehrere Schritte eingeteilt. Dabei wird zwischen zwei *Phasen* unterschieden, einer „weiten“ und einer „nahen“, aus dem Englischen für *Broad-Phase* und *Narrow-Phase* (vgl. [Eri05]). Während der *Broad-Phase* wird eine Vorauswahl an möglichen Kollisionspartnern getroffen. Dabei kann auf effiziente Algorithmen zurückgegriffen und Objekte, die nicht kollidieren, von der darauffolgenden *Narrow-Phase* ausgeschlossen werden. In der Narrow-Phase werden die Objekte, bei denen Kollisionen nicht ausgeschlossen werden konnten, auf ihre exakten Kollisionsdaten geprüft.

Ein Beispiel für die Broad-Phase ist die Spatial Subdivision (SSD). Dabei wird der Simulationsraum in kleinere Unterräume (Zellen) unterteilt, wodurch die Kollisionserkennung auf Objekte beschränkt werden kann, welche sich in derselben Zelle befinden. Das erste Mal wurde die SSD in der Moleküldynamik bei der Nachbarschaftssuche interagierender Atome verwendet. [Lo66] unterteilt den Raum in dreidimensionale Unterräume gleicher Größe. Durch geschickte Wahl der Größe

muss jedes Atom nur auf Interaktionen mit Atomen aus dem eigenen und aus den Nachbarräumen untersucht werden. [THM+03] unterteilt den $\mathbb{R}^3$ in AABB und fügt Knoten und Tetraeder in eine Hash-Tabelle ein. Der Algorithmus bietet die besten Ergebnisse, wenn Länge, Breite und Höhe der AABB sich an der durchschnittlichen Kantenlänge der Tetraeder orientieren. Der Algorithmus wurde für die vorliegende Arbeit implementiert und für die Erkennung weiterer geometrischer Objekte erweitert, wie in Kapitel 4 beschrieben wird.

Der Stand der Technik der Kollisionserkennung in Bezug auf deformierbare Objekte wird in [TKH+05] ausführlich zusammengefasst. Neben der SSD werden Distanzfelder und Image-Space-Verfahren, bei welchen die Abbildungen der simulierten Objekte auf Kollisionen getestet werden, beschrieben. Als eine der effizientesten Methoden werden *Bounding Volume*-Hierarchien hervorgehoben. Eine umfassende Einführung hierzu findet sich in [ZL03]. Dabei handelt es sich um Datenstrukturen, welche Broad- und Narrow-Phase zusammenfassen. Ein Objekt wird von seinen Primitiven ausgehend, in der Struktur als Blätter definiert, in immer gröbere *Bounding Volumes* geschachtelt, die als Baum implementiert sind. Bei der Kollisionserkennung werden die Äste, deren *Bounding Volumes* kollidieren, rekursiv durchlaufen. *Bounding Volume*-Hierarchien können auch für Selbstkollisionen eines Objektes als Datenstruktur herangezogen werden. Welche Art von *Bounding Volumes* in die Hierarchie implementiert werden, kann frei entschieden werden. Neben einfachen geometrischen Objekten, wie den bereits erwähnten BCs, können z. B. *Object Oriented Bounding Boxes* (OBB) [GLM96] oder allgemein *k-DOP* [KHM+98] verwendet werden. Weiter können AABB in einer Baumstruktur angeordnet werden. [VDB97] beschreibt den Aufbau und die Mechanismen, die übermäßige Überlappungen der *Bounding Boxes* vermeiden.

Bei Kollisionen zwischen deformierbaren Objekten, kann die exakte Kollisionsfläche der Objekte nicht so trivial, wie bei der Kollision mit *Bounding Volumes*, bestimmt werden. Dies ist darauf zurückzuführen, dass es bei einer Kollision bei beiden Kollisionspartnern zu Verformungen kommen kann und die Kollisionsfläche, nicht intuitiv ersichtlich ist. In [HTK+04] wird die Kollisionsfläche der deformierten Objekte auf Basis einer Gewichtungsfunktion berechnet. Dieser Ansatz ist auf Selbstkollisionen anwendbar und wurde in der vorliegenden Arbeit verwendet. Er wird im Detail und mit seinen Erweiterungen in Kapitel 4 beschrieben.

Selbstkollisionen

Das Erkennen von Selbstkollisionen deformierbarer Objekte ist kein triviales Problem. Da Objekte aus mehreren tausend Primitiven bestehen können, muss im

Vorfeld eine geschickte Vorauswahl getroffen werden, welche Regionen eines Objektes für Kollisionen in Frage kommen, um die restlichen Regionen von den weiteren Berechnungen ausschließen zu können. Dieser Ausschluss nennt sich Culling. In [VT94] wird durch einfache Berechnungen der Normalen der Oberflächendreiecke die Krümmung eines Objektes berücksichtigt. Dadurch, als auch durch die Beziehung benachbarter Primitive werden überflüssige Kollisionstests am Beispiel der Simulation von Kleidung vermieden (vgl. [VCMT95]). In [Pro97] wird der Ansatz mit *Bounding Volume*-Hierarchien kombiniert und auch für die Simulation von Kleidung verwendet (vgl. [PP96]).

Für die vorliegende Arbeit wurde der Algorithmus aus [HTK+04] verwendet, der für die Erkennung von Kollisionen zwischen deformierbaren Objekten implementiert wurde.

Kontinuierliche Kollisionserkennung

Methoden für die kontinuierliche Erkennung von Kollisionen, können zum einen rückwirkend verwendet werden, wodurch Kollisionen zwischen Objekten berücksichtigt werden können, welche sich zwischen $t - \Delta t$ und t vollständig durchdrungen haben. Diesen kann in t entgegengewirkt werden. Des Weiteren kommen kontinuierliche Kollisionsalgorithmen vorausschauend zum Einsatz. Dabei wird der folgende Zeitschritt zwischen t und $t + \Delta t$ provisorisch auf Kollisionen untersucht, wodurch diese vermieden werden können. Diese Methode wird vor allem in der Robotik für die Bewegungsplanung eingesetzt. Um Kollisionen kontinuierlich erkennen zu können, müssen die Bewegungen der Objekte abgeschätzt werden. Zusätzlich müssen die Bewegungen der Körper auf Kollisionen überprüft werden. Die Laufzeit ist abhängig von der Länge des Zeitschrittes und der Menge der Elemente, welche auf Kollisionen überprüft werden müssen. Die Implementierung einer SSD kann auch die kontinuierliche Kollisionserkennung beschleunigen. Bei schnellen Bewegungen bzw. zu großen Zeitschritten verteilen sich die Bewegungen der Objekte jedoch über mehrere Zellen, wodurch der Vorteil des Einsatzes einer SSD kleiner wird.

In [TCYM08] werden die Ansätze [VT94] und [Pro97] zur Vorauswahl relevanter Flächen durch die Bündelung der Oberflächennormalen in Kegel für die kontinuierliche Kollisionserkennung erweitert. Dabei wird die Bewegung eines Dreiecks in Δt interpoliert und ein Kegel aus den Normalen des Dreiecks berechnet (*Continuous Normal Cone*).

Kontinuierliche Kollisionserkennung musste in der vorliegenden Arbeit, insbesondere für die Interaktion mit der simulierten Szene implementiert werden, da

ungewollte oder zu schnelle Bewegungen des Benutzers zu Durchdringungen des virtuellen Gewebes führen können.

3.7 Kolissionsbehandlung

Die Kollisionsbehandlung (engl. Collision Response) ist im Gegensatz zur Kollisionserkennung ein physikalisches Problem. Die Kollisionen werden anhand der Informationen der Kollisionserkennung und dem Simulationsmodell entsprechend aufgelöst. Diese wiederum können dann je nach Simulationsmodell zu verschiedenen Reaktionen führen. Material kann brechen oder reißen. Es kann sich verformen, seine Geschwindigkeit und Richtung ändern oder gezielt geschnitten werden. Dabei wird im Folgenden zwischen *Penalty*-Methoden und sogenannten *Constrained Based*-Methoden unterschieden (vgl. [Wit97]).

Bei *Constrained Based*-Methoden sollen im Gegensatz zu *Penalty*-Methoden undefinierte Zustände gar nicht erst zugelassen werden. Aufgrund dieser Eigenschaften werden solche Methoden bevorzugt in der Robotik eingesetzt. Auch für deformierbare Objekte wurden bereits diverse Algorithmen vorgestellt ([BW92], [OTSG09]). *Constrained Based* -Methoden können im Allgemeinen einen sehr hohen Grad an Realität erreichen. Dieser geht aber auf Kosten der Rechenzeit (vgl. [TMOT12]). Für die vorliegende Arbeit werden nur *Penalty*-Methoden berücksichtigt.

Wie der Name bereits suggeriert, handelt es sich bei der *Penalty*-Methode in gewisser Weise um eine Strafe. Im konkreten Fall wird bei einer erkannten Kollision zusätzliche Energie in das System integriert, um eine Kollision nachträglich zu korrigieren. In einer physikalisch basierten Simulation müssen geeignete Rückstellungskräfte berechnet werden. Diese führen im Idealfall im darauffolgenden Zeitschritt zu einem kollisionsfreien Zustand. [MW88] führt die *Penalty*-Methode als eine sehr harte Feder zwischen den kollidierten Objekten in Bezug auf Festkörper ein. Die *Penalty*-Kraft kann z. B. proportional zur Eindringtiefe l_i des Objektes berechnet werden. Mit dem Steifigkeitsfaktor k_i folgt daraus folgende *Penalty*-Energie, welche als weitere potentielle Energie des Knotens p_i interpretiert werden kann:

$$E(p_i) = \frac{k_i \cdot l_i^2}{2}. \tag{3.23}$$

Die aus Gleichung 3.23 resultierende Kraft wird analog zu Gleichung 3.4 berechnet. Eine *Penalty*-Kraft muss groß genug sein, um den restlichen Kräften, die auf einen Knoten wirken, entgegenwirken zu können. Das führt zu einem hohen Steifigkeitsfaktor in Gleichung 3.23.

Penalty-Kräfte, die anhand der Eindringtiefe berechnet wurden, können zu unstetigem Verhalten in der medialen Achse eines Objektes führen. Zu dieser Situation kommt es dann, wenn benachbarte Knoten Kräfte in verschiedene Richtungen erfahren. In [BFA02] wird dieses Problem umgangen, indem Baryzentrischen Koordinaten des exakten Kollisionspunktes errechnet und Impulse anhand dieser auf die Knoten des Dreiecks verteilt werden. In der vorliegenden Arbeit wurden die exakten Kollisionspunkte zwischen Dreiecken und BCs in ähnlicher Form berechnet (siehe Abschnitt 4.4).

Bei Kollisionen mehrerer Objekte an mehreren Stellen können diese durch *Penalty*-Kräfte zum Teil gar nicht mehr oder nur durch sprunghafte Bewegungen aufgelöst werden. Obwohl *Penalty*-Kräfte eine Reihe von Nachteilen haben, bieten sie jedoch für viele Probleme eine ausreichend genaue und schnelle Lösung. Vor allem für Probleme mit einem Schwerpunkt auf eine gute Performanz sind sie geeignet (vgl. [BJ07]). In [TMOT12] wird der Durchschnitt der *Penalty*-Kräfte entlang der Bewegung eines Körpers auf Basis einer kontinuierlichen Kollisionserkennung berechnet. Dadurch können die genannten Nachteile minimiert werden. Der Algorithmus beschränkt sich jedoch auf Kollisionen zwischen Festkörpern und deformierbaren Objekten.

Im Gegensatz zu *Penalty*-Kräften, die Kräfte abschätzen und anschließend in das System integrieren um so Kollisionen aufzulösen, können *Constraint*-Methoden die benötigten Kräfte exakt berechnen, um eine bestimmte Position zu erreichen. *Constraint*-Methoden haben im Allgemeinen den Nachteil, dass sie mehrere Iterationen benötigen um einen kollisionsfreien Gleichgewichtszustand zu erreichen. Daher sind sie für Echtzeitanwendungen nur eingeschränkt einsetzbar.

In [WTF06] wird ein Ansatz für Festkörper vorgestellt, bei dem eine Kraft auf Basis des verwendeten Integrationsschemas in t berechnet wird und anschließend in den Integrationsschritt einfließt, um Zwangsbedingungen in $t + 1$ einzuhalten. Das Verfahren beschränkt sich jedoch auf das Euler-Verfahren und wird nur in Bezug auf Festkörper eingeführt.

In [GBT06] wird dieser Ansatz auf verschiedene Integrationsverfahren erweitert. Des Weiteren wird in [GBT06] das Verfahren auf deformierbare Körper angewendet. Dabei wird für die Knoten, die zum Zeitpunkt t kollidieren, eine Kraft auf Basis des Integrationsverfahren berechnet und im folgenden Integrationsschritt in

das System integriert. Im Folgenden wird die Methode am Beispiel des normalen Verlet-Verfahren vorgestellt:

Soll ein Knoten auf die Position $\vec{p}_i(t + \Delta t)$ bewegt werden, kann die Kraft $\vec{\hat{F}}_i(t)$ berechnet werden, durch die sich der Knoten nach dem Integrationsschritt an die gewünschte Position bewegt. Formt man den Term $\Delta t^2 \cdot \frac{\partial^2 \vec{p}_i(t)}{\partial t^2}$ auf Basis des Zweiten Newtonschen Gesetzes um und addiert zur aktuell wirkenden Kraft $\vec{F}_i(t)$ die zusätzliche Kraft $\vec{\hat{F}}_i(t)$, so erhält man:

$$\vec{p}_i(t + \Delta t) = 2\vec{p}_i(t) - \vec{p}_i(t - \Delta t) + \frac{\Delta t^2}{m_i} \cdot (\vec{F}_i(t) + \vec{\hat{F}}_i(t)). \tag{3.24}$$

Daraus folgt die benötigte Kraft $\vec{\hat{F}}_i(t)$:

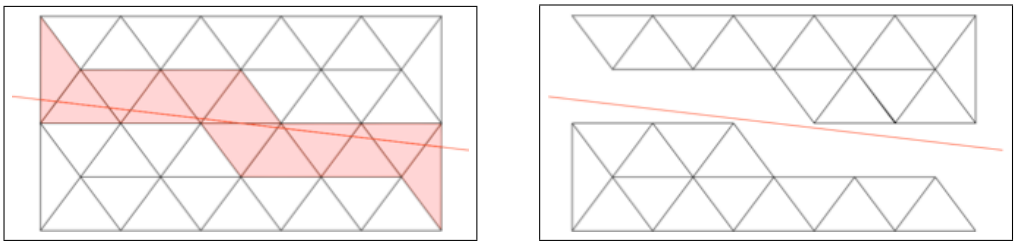
$$\vec{\hat{F}}_i(t) = \frac{m_i}{\Delta t^2}\Big(\vec{p}_i(t + \Delta t) - 2\vec{p}_i(t) + \vec{p}_i(t - \Delta t)\Big) - \vec{F}_i(t). \tag{3.25}$$

Topologische Änderungen

Unter topologischen Veränderungen eines Objektes sind die Spaltung eines Objektes in kleinere oder der Zusammenschluss mehrerer Objekte in ein größeres zu verstehen. Beispiele dafür sind das Zerbrechen eines Glases oder das Verschmelzen zweier Gegenstände. Der Fokus der folgenden Verfahren liegt auf der Genauigkeit und Rechenzeit, die benötigt wird, um die Geometrie eines Objektes zu verändern, sodass die Simulation echtzeitfähig bleibt. Dabei ist nicht jedes Simulationsmodell für die Echtzeit-Simulation topologischer Änderungen geeignet, da manche Simulationsmodelle teure Vorberechnungen benötigen und diese bei topologischen Veränderungen erneut durchgeführt werden müssen.

Bei topologischen Änderungen kann zwischen elementabhängigen und elementunabhängigen unterschieden werden. Topologische Änderungen sind als elementabhängig definiert, wenn das Mesh nur an den vorhandenen, gemeinsamen Grenzen der Primitiven aufgespalten werden kann. Im Gegensatz dazu ist ein elementunabhängiges Verfahren nicht abhängig von seiner topologischen Beschaffenheit.

[BN97] erwähnt zum ersten Mal die Modellierung topologischer Änderungen in Tetraedernetzen durch das Entfernen kollidierender Tetraeder. Abbildung 3.8 skizziert den Algorithmus. Dieses Verfahren ist elementabhängig und bietet eine schnelle und effiziente Lösung. Die Qualität des Schnittes hängt jedoch von der Auflösung des Mesh ab (vgl. [DCA99]). Abhängig von der Anzahl der topologischen Änderungen wird das Mesh während der Simulation gröber, da sich die Anzahl der Elemente verringert.



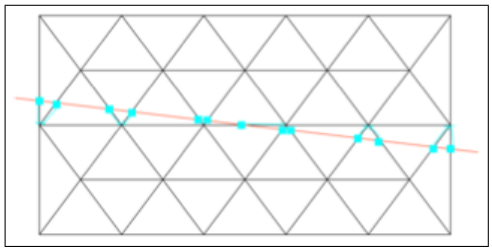
(a) Geschnittene Tetraeder werden erkannt. (b) Tetraeder werden entfernt.

Abbildung 3.8: Topologische Änderungen durch Entfernen von Tetraedern.

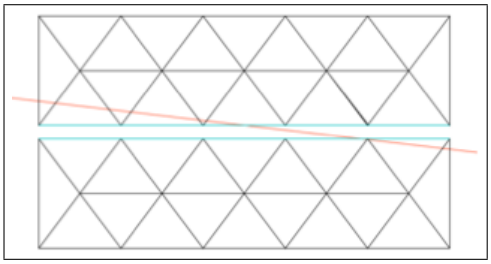
[NvdS00] beschreibt topologische Änderungen durch die Aufspaltung eines Mesh an vorhandenen Elementen. Ein Dreiecksnetz wird entlang der Kanten seiner Dreiecke getrennt (Abbildung 3.9(a) und 3.9(b)). Auch dieses Verfahren ist elementabhängig. Das Netz kann nur entlang des am nächsten gelegenen Primitivs gespalten werden. [NvdS01] führt hier einen *Snapping*-Algorithmus ein, welcher die Primitive, an welchen das Netz gespalten wurde, nachträglich an die genaue Schnittstelle verschiebt (Abbildung 3.9(c)).

[Maz97] und [BMG99] führen topologische Änderungen in Tetraedernetzen durch *Remeshing*-Verfahren ein. Das heißt, dass geschnittene Elemente gelöscht und anschließend neue Elemente erstellt werden, welche den Schnitt darstellen. Dabei ist der Ansatz in [Maz97] bedingt elementunabhängig, da er nur an den exakten Kantenschnittpunkten neue Knoten erstellt und abhängig vom Schnittmuster mit den gegenüberliegenden Knoten verbindet (Abbildung 3.10(a)). Im Gegensatz dazu ist der Ansatz [BMG99] unbedingt elementunabhängig, da Schnitte exakt vom Eintrittspunkt an der Oberfläche eines Tetraeders bis zum Austrittspunkt modelliert werden (Abbildung 3.10(a)). Jedoch ist die Anzahl der Tetraeder, die erstellt werden müssen, um einen Schnitt zu modellieren, im Allgemeinen auch größer.

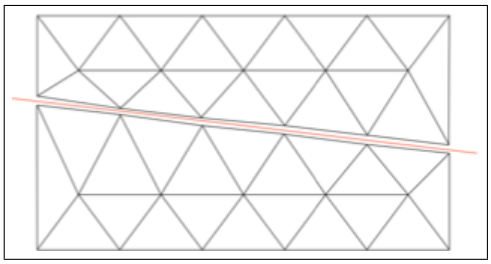
Topologische Änderungen an einem Mesh werden als progressiv oder nicht progressiv eingestuft. Spiegelt ein Mesh in jedem Zeitpunkt des Schneidevorgangs die aktuelle Topologie wider, handelt es sich um einen progressiven Ansatz (vgl. [BGTG04]). Muss erst gewartet werden bis der ganze Schnitt durchgeführt wurde, bevor das Mesh topologisch verändert werden kann, handelt es sich um einen nicht progressiven Ansatz (vgl. [SHGS06]). Als semi progressiv lassen sich Algorith-



(a) Geschnittene Kanten werden erkannt.

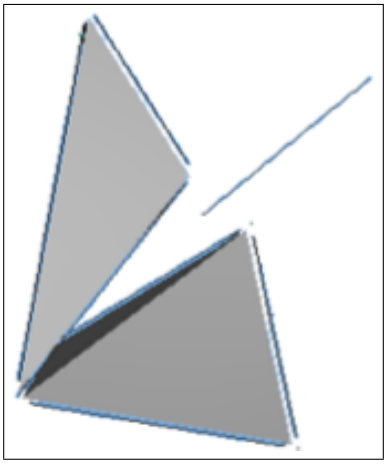


(b) Tetraedernetz wird an Primitiven gespalten.

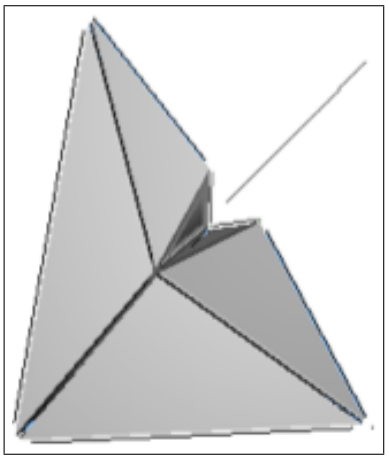


(c) Knoten werden nachträglich an die Schnittkante verschoben.

Abbildung 3.9: Topologische Änderungen durch Aufspalten des Netzes an vorhandenen Knoten und Kanten.



(a) Geschnittene Kanten werden unterteilt und Tetraeder wird mit gegenüberliegendem Knoten unterteilt.



(b) Geschnittene Kanten und Dreiecke werden erkannt. Tetraeder wird exakt unterteilt.

Abbildung 3.10: Topologische Änderungen durch Löschen und Erstellen neuer Tetraeder.

men einordnen, welche die Topologie des Mesh an die aktuelle Schnittbewegung anpassen, sobald ein Tetraeder nicht mehr kollidiert.

In [MK00] wird ein progressiver Ansatz beschrieben. Befindet sich ein Schnittgerät in einem Tetraeder, wird in jedem Zeitschritt t die topologische Änderung an das Schnittgerät angepasst. Das heißt, die Unterteilung aus $t-\Delta t$ wird gelöscht und der Tetraeder wird in t erneut unterteilt. Dies geschieht solange, bis das Schnittgerät den Bereich des ursprünglichen Tetraeders verlassen hat. Der Algorithmus ist nicht echtzeitfähig, da bei dem *Remeshing*-Verfahren zu viele Löschvorgänge benötigt werden (vgl. [SHGS06]). [BGTG04] setzt an diesem Algorithmus an und verkleinert die Löschvorgänge, indem er einen Zustandsautomaten einführt. Weiter werden sechs Schnittmuster verwendet, auf die sich die über drei Tausend Möglichkeiten einen Tetraeder zu schneiden, abbilden lassen. Selbst Zitter- und Rückwärtsbewegungen eines Schnittgeräts werden berücksichtigt, sodass bei unruhiger Hand an der Grenze eines Tetraeders das Mesh nicht ständig neu generiert werden muss.

In [SHGS06] wird ein nicht progressiver Ansatz in Zusammenhang eines Hysteroskopie Simulators vorgestellt. Erst wenn das Schnittgerät das Objekt wieder verlassen hat, kann der Schnitt berechnet und dargestellt werden. Der Ansatz berechnet topologische Änderungen sowohl durch die Unterteilung einzelner Tetraeder als auch durch die Aufspaltung des Tetraedernetzes an Tetraederseiten. In [SHGS06] wird die Masse eines Knoten nach der Unterteilung eines Tetraeders proportional zu den Volumina der neuen Tetraeder verteilt.

In [SDF07] wird von einem geschnittenen Tetraeder eine identische Kopie für die Simulation erstellt. Dabei nehmen Original und Kopie denselben Raum nach dem Schnitt ein. Es wird ein aktiver und passiver Teil eines Tetraeders bestimmt, der durch die Schnittfläche festgelegt wird. Das Mesh wird an den Knoten aufgespalten, an welchen ein aktiver Knoten eines Tetraeders auf einen passiven trifft. Die Kollisionserkennung und Visualisierung des Mesh beschränkt sich auf den aktiven Teil eines Tetraeders.

Auf mögliche Auslöser topologischer Änderungen wird in den vorgestellten Methoden nicht eingegangen. [SSSH11] nimmt zwar topologische Veränderungen an verformten Objekten vor, jedoch wird die Schnittfläche direkt durch eine Klinge festgelegt. Nur [JBB+10] geht durch ihr Modell auf das Umgehen der Oberflächenspannung ein. Dabei wird die Klinge in eine Hülle platziert, wodurch sowohl Druck auf das Mesh ausgeübt als auch Schnitte simuliert werden können, sobald der Druck groß genug ist. Die Methode wird im Zusammenhang mit der Simulation von Voxelnetzen eingeführt.

3.8 Visualisierung

Um die simulierte Szene zu visualisieren, wird eine Grafikpipeline durchlaufen. Das Ziel in der VR dabei ist, dem Benutzer mindestens 25 Bilder pro Sekunde von der Szene, mit welcher er interagiert, zu generieren. Dadurch wird der Prozess vom Benutzer als dynamisch wahrgenommen und Präsenz erzeugt (vgl. Abschnitt 3.1). Dieser Prozess nennt sich Echtzeit-Rendern. Neben anderen Schritten, die in der Grafikpipeline berechnet werden, wird die Farbe der Oberfläche eines Objektes generiert. Ein gängiges Verfahren dabei ist, die Oberfläche durch Dreiecke zu approximieren und durch die Farbe der einzelnen Knoten eines Dreiecks, die Farbe für das gesamte Dreieck zu interpolieren. Da die Visualisierung ein sehr umfangreiches und eigenständiges Forschungsfeld umschreibt, wird der interessierte Leser für eine detaillierte Einführung auf [AMHH08] verwiesen.

3.9 Simulationsframeworks

Die Anzahl der Frameworks, die physikalische Simulationsmodelle für deformierbare Objekte unterstützen, ist in den letzten Jahren stark angewachsen. Es gibt sowohl Open Source als auch Closed Source Frameworks, die unter verschiedenen Lizenzmodellen vertrieben werden. Beispiele für physikalische Open Source Frameworks finden sich in der *Bullet Physics Library* [Bul] und in der *PhysX Engine* von NVidia® [Phy], womit sich sowohl Softkörper als auch Festkörper simulieren lassen. Frameworks, die ihren Fokus speziell auf medizinische Echtzeit-Simulation gelegt haben, sind das GiPSi-Framework [GÇTS04] und das SOFA-Framework [ACF+07]. Letzteres wird aktuell weiterentwickelt. Neben einem kompletten Framework mit verschiedenen Algorithmen für die Simulation und Interaktion von menschlichen Organen, werden anatomische Modelle bereitgestellt.

Viele Unternehmen haben eigene Frameworks, die sie ihren Bedürfnissen angepasst haben. Nicht selten bauen diese auf Open Source Frameworks, wie *OpensceneGraph* [Ope], *Ogre3D* [Ogr] oder die *boost C++ LIBRARIES* [boo] auf.

3.10 Trainingssimulatoren in der Mikrogefäßchirurgie

In [BMLS01] und [BSS02] wird einer der ersten mikrochirurgischen Trainingssimulatoren vorgestellt. Der Benutzer arbeitet mit zwei chirurgischen Pinzetten, welche elektromagnetisch getrackt werden. Blutgefäße bestehen aus zweidimensionalen Primitiven, die von einem MFM simuliert werden. Der Simulator erlaubt das setzen von Nähten zwischen zwei Blutgefäßen. Ein ähnliches Szenario wird in [WSBB08] vorgestellt. Die Benutzereingabe wird um haptisches Feedback erweitert, indem die Pinzetten jeweils auf ein Phantom OMNI montiert werden. Für die Simulation des Fadens wird ein physikalisch basierter Ansatz implementiert.

Dieselben haptischen Eingabegeräte werden in [WBJ+06] für einen Simulator verwendet, der grobe Schnitte auf virtuellen Objekten simuliert. Auf die Verwendung von Originalinstrumenten wird verzichtet. Keiner der hier erwähnten Simulatoren bietet die Möglichkeit, das Entfernen der Adventitia am Ende eines Blutgefäßes bzw. die Dilatation eines Blutgefäßes zu trainieren.

KAPITEL

4

Ich kenne keinen sicheren Weg zum Erfolg, aber einen sicheren Weg zum Misserfolg: Es allen Recht machen zu wollen.

(Platon ca. 427 347 v. Chr.*)*

Methoden

Im Folgenden werden Algorithmen beschrieben, die für die Erstellung der Trainingsmodule benötigt werden.

Anforderungen der Trainingsmodule

Die Anforderungen ergeben sich aus den in Abschnitt 2.3 aufgezeigten Techniken einer vaskulären Anastomose:

- Es müssen Objekte für die Repräsentation des Gewebes erstellt werden.
- Das Gewebeverhalten muss simuliert werden, das heißt, die Objekte müssen sich unter Krafteinwirkung elastisch bzw. plastisch verformen.
- Für die Manipulation der Objekte kommen chirurgische Instrumente zum Einsatz. Mit diesen muss das Gewebe interagieren:
  - Durch die Pinzette kann Gewebe gegriffen, gezogen und gedrückt werden. Unkontrollierte Bewegungen des Benutzers oder hohe Kraftein-

wirkung durch die Pinzette, können zur Dissektion des Gewebes führen.

  - Das Gewebe muss zwischen den Schneiden der Schere geschnitten werden können. Durch die stumpfe Seite der Schere kann das Gewebe verschoben werden.

- Damit Gewebe vernäht werden kann, müssen Kollisionen zwischen den Blutgefäßen erkannt werden und so aufgelöst werden, dass die Gefäße durch den Druck des Fadens aufeinandergedrückt werden.

Spezifikation der Algorithmen

Anhand der Anforderungen können die Algorithmen, die benötigt werden, um eine reale Operationsszene in einem Simulator abbilden zu können, folgendermaßen spezifiziert werden:

1. Die Objekte, die das Gewebe repräsentieren, sind dreidimensional und deformierbar. Sie werden analytisch als Tetraedernetze erstellt (Abschnitt 4.1).

2. Die Grundlage der Gewebesimulation bildet ein MFM. Dieses unterstützt volumetrische Effekte auf Basis von Tetraedernetzen. Weiter unterstützt das Simulationsmodell topologische Änderungen in Echtzeit (Abschnitt 4.2).

3. Kollisionen zwischen verschiedenen Tetraedernetzen werden erkannt und aufgelöst (Abschnitt 4.3).

4. Das Volumen der Instrumente wird durch *Bounding Volumes* angenähert. Es werden Kriterien für das Greifen, das Schieben und das Auslösen topologischer Änderungen des Gewebes definiert (Abschnitt 4.4).

Eine grafische Darstellung der Algorithmen, die benötigt werden, um eine reale Operationsszene im Simulator abbilden zu können, findet sich in Abbildung 4.1.

Alle Algorithmen und Messungen werden auf einem Intel® Core™ i7 CPU mit 2.67 GHz, 6GB Memory und einer NVidia® GeForce GTX 285 Grafikkarte entwickelt und getestet.

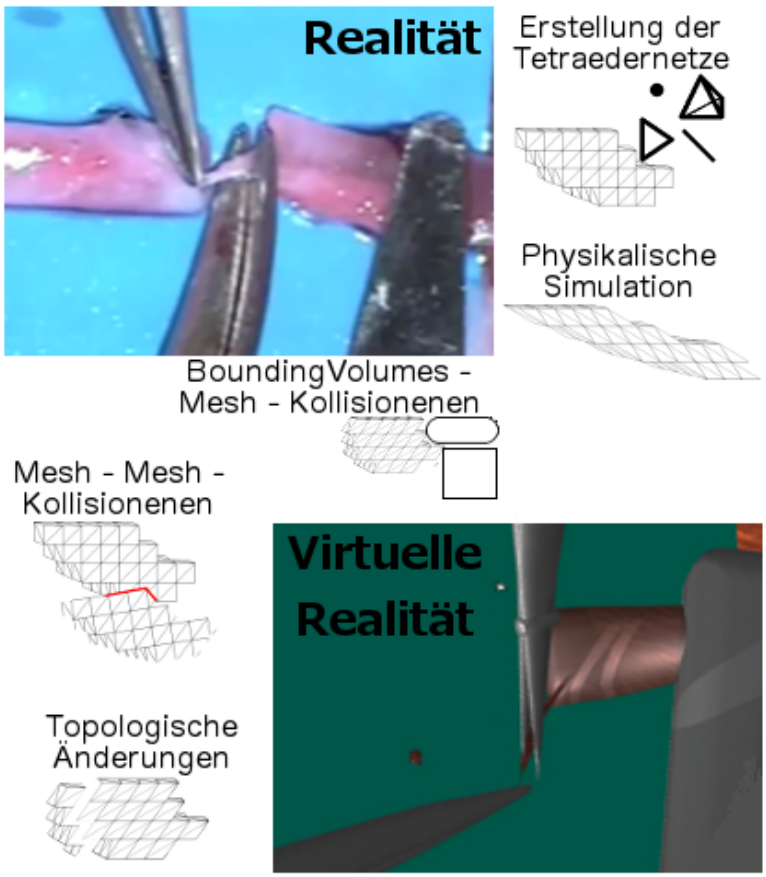


Abbildung 4.1: Aufschlüsselung der Algorithmen zur Abbildung realer Operationsszenen im Simulator.

4.1 Mesh-Erstellung

In der vorliegenden Arbeit stehen Algorithmen für die realitätsnahe Interaktion mit dem Gewebe und nicht die exakte Nachbildung von menschlichem Gewebe im Vordergrund. Daher kann von exakten medizinischen Nachbildungen abgesehen werden.

Es werden Objekte erstellt, die Mikroblutgefäße und umliegendes Bindegewebe darstellen. Tetraedernetze werden auf der Basis von Quadern erstellt. Dabei wird die Tatsache ausgenutzt, dass sich ein Quader in fünf Tetraeder unterteilen lässt (vgl. Abschnitt 3.3).

Abstrakte Objekte

Um aus Quadern Tetraedernetze zu erstellen, werden diese in beliebiger Anzahl in den verschiedenen Koordinatenachsen hintereinander angefügt. Abbildung 4.2 zeigt Beispiele solcher Tetraedernetze. Bei der Implementierung des Algorithmus muss vor allem darauf geachtet werden, dass sich benachbarte Tetraeder gemeinsame Knoten und die Verbindungselemente zwischen den Knoten teilen (vgl. [CDM+02]). Andernfalls wird das Mesh inkonsistent. Dies wirkt sich negativ auf die Simulation aus und führt unter Umständen zum Abbruch der Simulation. Tetraedernetze, die durch die Methode erstellt werden, werden für einfache Formen von Bindegewebe und für abstrakte Aufgaben verwendet.

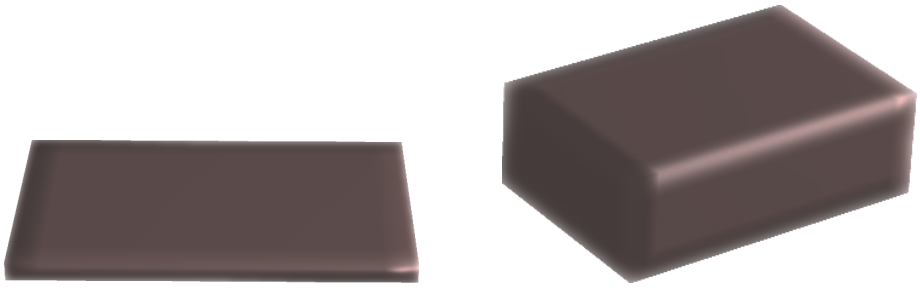


Abbildung 4.2: Erstellung quaderförmiger Tetraedernetze.

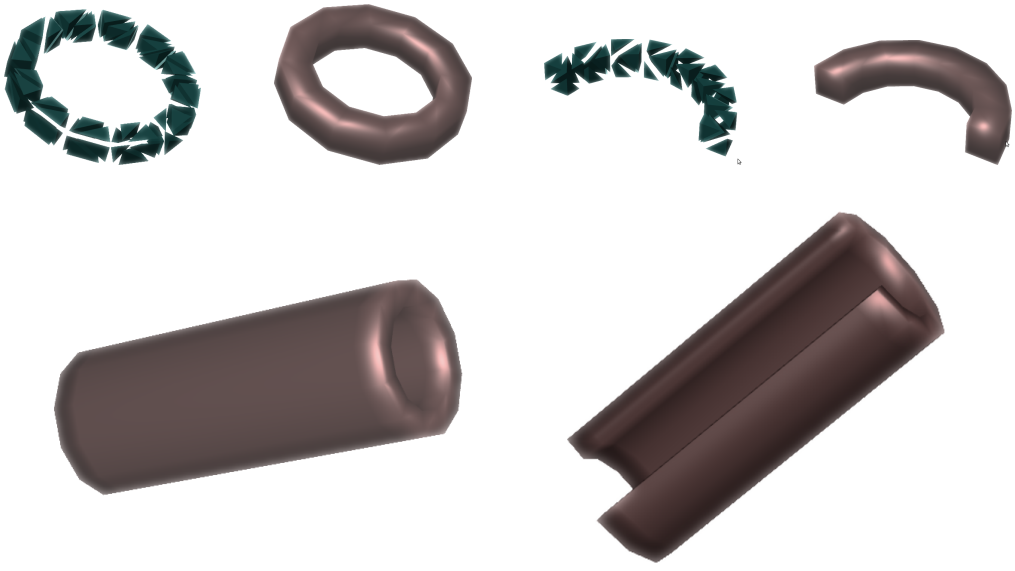


Abbildung 4.3: Offene und geschlossene röhrenförmige Tetraedernetze für die Repräsentation der Media und Adventitia.

Nachbildungen der Blutgefäß

Röhrenförmige Tetraedernetze stellen die virtuellen Objekte der Media und Adventitia dar. Auch hier werden Quader hintereinander angefügt. Neben geschlossenen werden auch offene Objekte generiert (Abbildung 4.3).

Da die Erstellung derartiger Netze trivial ist und auf einfachen mathematischen Rechenoperationen basiert, wird an dieser Stelle auf weitere Erläuterungen verzichtet.

4.2 Simulationsmodell

Die Grundlage des Simulationsmodells, das für die Gewebesimulation eingesetzt wird, bildet der Algorithmus aus [THMG04], der in Abschnitt 3.4 beschrieben wird. In der vorliegenden Arbeit wurden die Algorithmen für die volumen- und längenerhaltenden Kräfte, wie sie in den *VRm Libs*[1] implementiert sind, verwendet:

$$E_L(\vec{p}_i, \vec{p}_j) = \frac{1}{2} k_D \left(|\vec{p}_j \vec{p}_i| - L_0 \right)^2, \tag{4.1}$$

$$E_V(\vec{p}_i, \vec{p}_j, \vec{p}_k, \vec{p}_l) = \frac{1}{2} k_V \left(\frac{1}{6} (\vec{p}_j \vec{p}_i) \cdot ((\vec{p}_k \vec{p}_i) \times (\vec{p}_l \vec{p}_i)) V_0 \right)^2. \tag{4.2}$$

Als Integrationsverfahren wird das in Abschnitt 3.5 beschriebene Velocity-Verlet-Verfahren eingesetzt. Auch dieses ist Teil der *VRm Libs*. Mit diesem Setup kann ein breites Spektrum an verschiedenen Verhaltensmustern modelliert werden, wodurch das heterogene Verhalten von Gewebe abgebildet werden kann. Es kann sowohl steiferes Gewebe als auch elastisches Bindegewebe modelliert werden. Das Modell unterstützt nicht nur elastische Verformungen, sondern auch plastische. Dieser Effekt wird für die Simulation der Dilatation verwendet, bei der sich ein schlaffes Blutgefäß versteift.

[1] Bei den *VRm Libs* handelt es sich um Algorithmen und Datenstrukturen der *VRmagic GmbH*, die für die Implementierung der vorliegenden Arbeit verwendet und erweitert wurden. Abschnitt 5.2 gibt einen Überblick über die *VRm Libs*.

Das Simulationsmodell wurde bereits in “An Aneurysm Clipping Training Module for the neurosurgical Training Simulator NeuroSim“ [BSS+12] verwendet.

4.2.1 Topologische Änderungen

Für das verwendete Simulationsmodell wurde vom Autor ein Algorithmus in die *VRm Libs* implementiert, der topologische Änderungen in Tetraedernetzen darstellt. Dieser basiert auf den in Abschnitt 3.7 vorgestellten Ansätzen und wird für die Interaktion zwischen Instrumenten und Gewebe benötigt (siehe Abschnitt 4.4). Für die vorliegende Arbeit werden Schnittpunkte mit Tetraederseiten nicht berücksichtigt. Diese wurden ebenfalls in die *VRm Libs* implementiert, sind jedoch bei topologischen Änderungen in viskoelastischem Gewebe nicht nötig, da keine exakte Schnittfläche auf dem Gewebe entsteht. Weitere Vorteile beschreibt Abschnitt 4.2.1.

Algorithmus

Der Algorithmus wird anhand eines Tetraedernetzes T und einer Linie L beschrieben. T besteht aus den Tetraedern A_i. T und L befinden sich im dreidimensionalen Raum und interagieren miteinander. Dabei verändert L über die Zeit seine Position, wodurch zwischen den diskreten Zeitpunkten $t - \Delta t$ und t eine Fläche aufgespannt wird, die in zwei Dreiecke unterteilt wird. Die Fläche wird fortan als Schnittfläche bezeichnet, die Dreiecke als Schnittdreiecke. Die Kollisionserkennung wird jeweils in t durchgeführt und berechnet Kollisionen des vergangenen Zeitschrittes Δt. Fanden Kollisionen statt und kollidiert A_i und L zum Zeitpunkt t nicht, kann A_i entlang der Schnittfläche unterteilt werden. Kollidieren L und A_i zum Zeitpunkt t werden die Informationen aus der Kollisionsinformation in eine Tabelle geschrieben und es findet keine Unterteilung in t statt.

Die Tabelle wird in t iterativ abgearbeitet. Steht ein Tetraeder in der Tabelle, fand eine Kollision in $t-2\Delta t$ statt und die Kollision war zum Zeitpunkt $t-\Delta t$ noch nicht vollständig abgeschlossen. Existiert in t kein Kontakt zwischen L und A_i, werden die Informationen in der Tabelle aktualisiert, A_i unterteilt und aus der Tabelle entfernt. Andernfalls werden die Kollisionsinformationen des Tetraeders aktualisiert, jedoch wird A_i nicht unterteilt, sondern erneut in eine Tabelle geschrieben.

Kurz: Kollidiert ein Tetraeder in t, kann er erst im Zeitpunkt $t + n\Delta t$ unterteilt werden, wenn er und L nicht mehr kollidieren. Bei diesem Algorithmus handelt es sich um einen semi-progressiven Ansatz (vgl. Abschnitt 3.7). Abbildung 4.4 gibt einen Überblick über die einzelnen Teilschritte des Algorithmus.

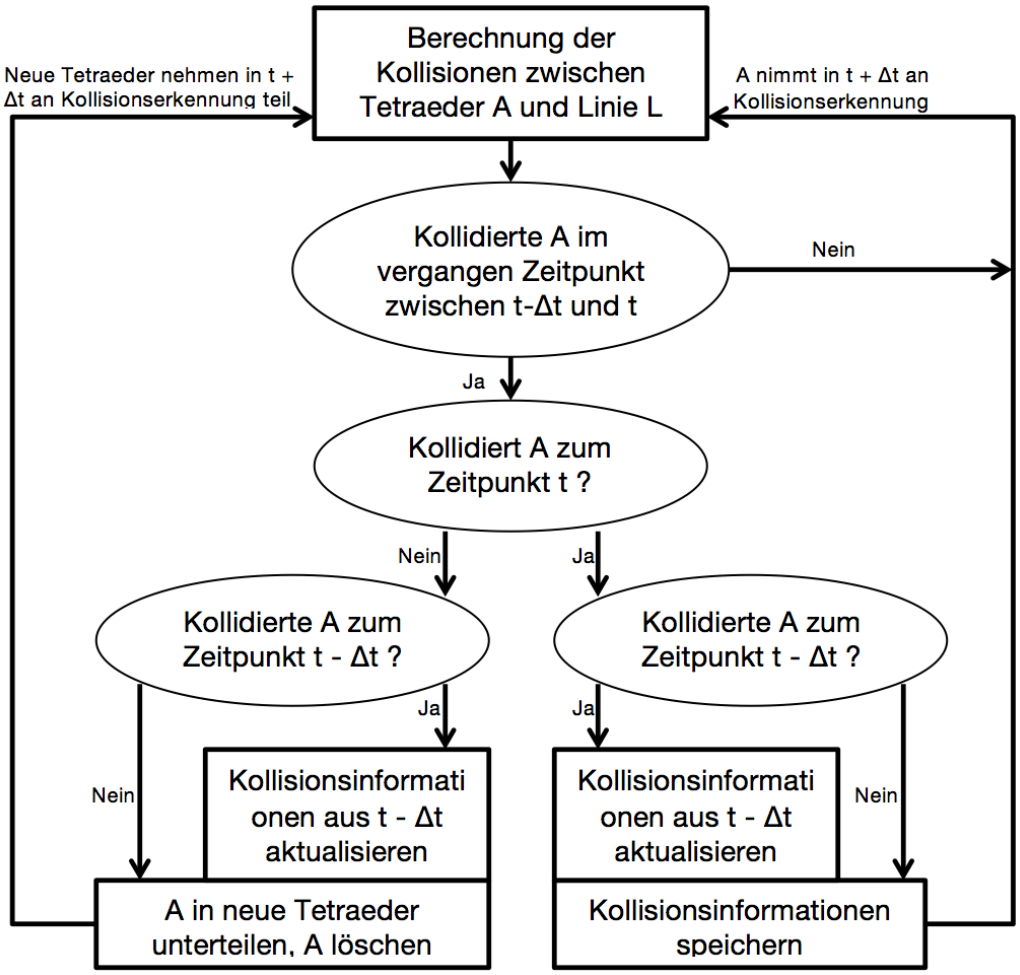


Abbildung 4.4: Übersicht des Algorithmus für topologische Änderungen.

Kollisionserkennung

Die Kollisionserkennung berechnet die Schnittpunkte zwischen den Kanten von A_i und den Schnittdreiecken. Des Weiteren wird überprüft, ob L mit A_i kollidiert. Dazu werden die Schnittpunkte der vier Seiten von A_i mit L berechnet. Demnach lassen sich alle Kollisionsinformationen berechnen, indem Schnittpunkte zwischen Linien und Dreiecken berechnet werden. Eine Einführung in die mathematischen Grundlagen für die Berechnung dieser Schnittpunkte findet sich in A.2. Abbildung 4.5 veranschaulicht die Schnittfläche, die L über die Zeit aufspannt, und die möglichen Kollisionen, die mit A_i auftreten können.

Jeder kollidierende Tetraeder wird mit seinen Kollisionsinformationen in eine Tabelle geschrieben. Listing 4.1 stellt die Struktur, in der die Informationen aus der Kollision gespeichert werden, in Pseudocode dar.

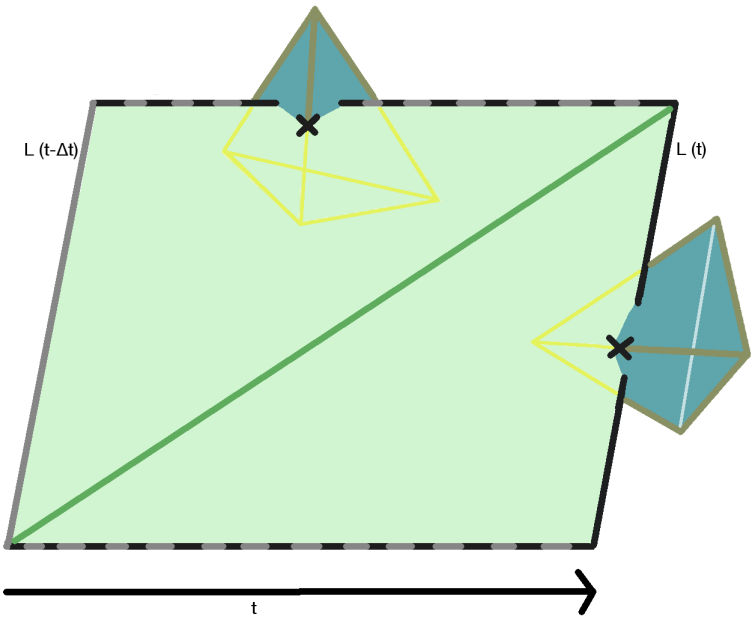


Abbildung 4.5: Schematische Darstellung möglicher Kollisionen zwischen den Tetraedern und einer Fläche, die durch die Linie L zwischen den Zeitpunkten $t - \Delta t$ und t aufgespannt wird.

```
struct Collision {
    struct EdgeCollision {
        node *p0, *p1; //Kantenknoten
        vector3<float> collision_point;
    };

    //Kanten des Tetraeders, die zwischen t−Δt und t kollidierten
    list<EdgeCollision> edge_collisions_list;
    //Tetraeder kollidiert zum Zeitpunkt t
    bool active;
};
```

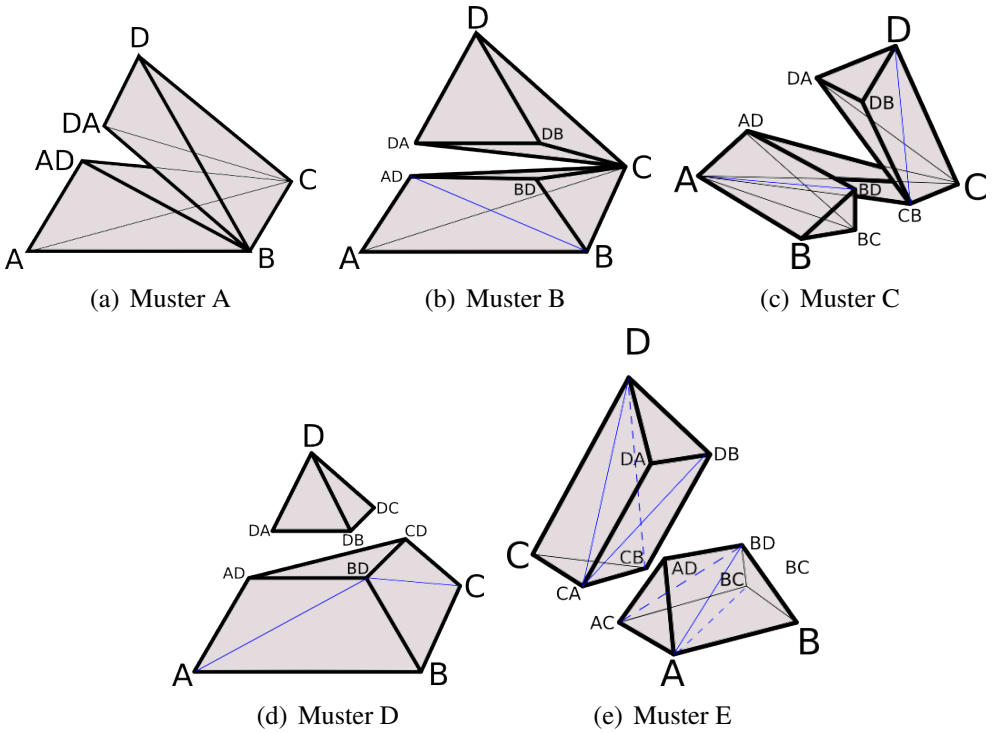
Listing 4.1: Struktur der zu jedem kollidierten Tetraeder gesammelten Informationen.

Klassifizierung der Unterteilungsmuster

Alle Varianten, auf die ein Tetraeder geschnitten werden kann, werden auf ein passendes Schnittmuster abgebildet. Die gewählten Schnittmuster basieren auf den Ansätzen von [Maz97] und [SHGS06]. Im Gegensatz zu [SHGS06] werden bei der Unterteilung der Tetraeder weniger Primitive benötigt. Auf die zusätzlichen

Knoten, die auf jeder durchschnittenen Oberfläche erstellt werden, wird in der Implementierung verzichtet. Diese werden in [SHGS06] dazu verwendet, um Tetraeder mit besseren Innenwinkeln erstellen zu können. Dadurch enstehen jedoch mehrere Tetraeder mit kleineren Volumen.

Für jede Unterteilung muss der geschnittene Tetraeder, die geschnittenen Kanten und Oberflächendreiecke gelöscht werden. Es werden neue Knoten an den Schnittpunkten und neue Tetraeder, Kanten und Oberflächendreiecke erstellt. Abbildung 4.6 zeigt die Schnittmuster der Implementierung. Die mit blau gekennzeichneten Kanten sind variable Kanten. In bestimmten Fällen wird die Diagonale zwischen den beiden anderen Knoten in dem Viereck auf dieser Seite erstellt. Abschnitt 4.2.1 geht gesondert auf dieses Problem ein.



(a) Muster A (b) Muster B (c) Muster C

(d) Muster D (e) Muster E

Abbildung 4.6: Muster für die Unterteilung geschnittener Tetraeder.

Für die verwendeten Schnittmuster mussten die exakten Unterteilungen der Tetraeder, Kanten und Oberflächendreiecke herausgearbeitet werden. Diese werden für jedes Schnittmuster detailliert im Anhang B beschrieben.

Generierung neuer Knoten

Bevor ein Tetraeder unterteilt werden kann, werden neue Knoten an den Kantenschnittpunkten generiert. Bei der Generierung muss auf Folgendes geachtet werden: Kanten werden in einem Mesh in der Regel von mehreren Tetraedern genutzt. Deshalb muss bei der Erstellung eines Knotens darauf geachtet werden, dass am Kollisionspunkt einer Kante nicht von jedem Tetraeder, der die Kante nutzt, neue Knoten generiert werden, sondern nur ein einziges Mal. Um Knoten nicht mehrfach zu erstellen, wird folgender Ansatz für die Propagierung gewählt: Sowohl die Endknoten der unterteilten Kanten als auch die an den Schnittpunkten der Kanten erstellten Knoten werden in einer Liste gespeichert. Bevor weitere Knoten erstellt werden, wird in dieser Liste nachgesehen, ob die passenden Knoten bereits generiert wurden. Ist dies der Fall, werden die entsprechenden Knoten verwendet und es wird von der Generierung neuer Knoten abgesehen.

Wie in Abschnitt 3.4 beschrieben, besitzen Knoten eine Startposition. Damit deformierte Objekte unterteilt werden können, muss auch für neu erstellte Knoten eine Startposition errechnet werden. Dadurch wird das Volumen des geschnittenen Tetraeders und die Länge der unterteilten Kanten erhalten. Im Folgenden wird beschrieben, wie die Startposition neu generierter Knoten bestimmt wird. Anhand dieser Position werden die Nullvolumen und Nulllängen der Tetraeder und Kanten berechnet.

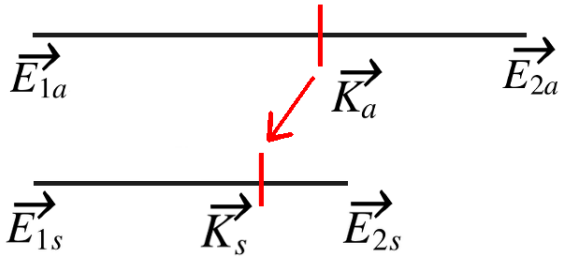


Abbildung 4.7: Startposition und aktuelle Position eines Schnittpunktes.

Kollidiert die Kante eines Tetraeders, entstehen am Schnittpunkt zwei neue Knoten N_1 und N_2, an denen die Kante geteilt wird. Ist das Netz deformiert, variieren Startposition und aktuelle Position der neu erstellten Knoten. In Abbildung 4.7 stellt $\vec{K}_a$ den Schnittpunkt einer deformierten Kante $\overline{\vec{E}_{1a}\vec{E}_{2a}}$ dar. An der Stelle $\vec{K}_a$ muss ein neuer Knoten generiert werden. Die aktuelle Position kann direkt von der Kollisionserkennung übernommen werden. Die Startposition $\vec{K}_s$ muss auf die undeformierte Kante $\overline{\vec{E}_{1s}\vec{E}_{2s}}$ zurückgerechnet werden.

Seien $\vec{E_{1a}}$ und $\vec{E_{2a}}$ die aktuelle Position und $\vec{E_{1s}}$ und $\vec{E_{2s}}$ die Startposition, an dem Endknoten der Kante, und $\vec{K}a$ sei der Punkt, an dem die Kante durchschnitten wurde. Die Startposition der neuen Knoten $\vec{N1}s$ und $\vec{N2}s$ lässt sich über die aktuelle und die Nulllänge der Kante berechnen. Voraussetzung ist, dass zwei Knoten, die über eine Kante miteinander verbunden sind, nicht dieselbe Position einnehmen können, deshalb folgt aus $|\vec{E_{2a}} - \vec{E_{1a}} \neq 0|$:

$$\vec{N_{1s}} = \vec{N_{2s}} = \vec{E_{1s}} + \frac{\vec{K_a} - \vec{E_{1a}}}{|\vec{E_{2a}} - \vec{E_{1a}}|} \cdot (\vec{E_{2s}} - \vec{E_{1s}}). \tag{4.3}$$

Abbildung eines Schnittes auf ein passendes Schnittmuster

Anhand der Anzahl der durchschnittenen Kanten wird entschieden, nach welchem Muster ein Tetraeder unterteilt wird. Aus der Kollisionserkennung sind die Endknoten der durchschnittenen Kanten bekannt. Im folgenden Schritt müssen die einzelnen Endknoten auf das entsprechende Schnittmuster abgebildet werden. Dies erfolgt anhand logischer Operationen. In Abbildung 4.8 wird das Abbilden der einzelnen Knoten bei einem Schnitt über zwei Kanten dargestellt.

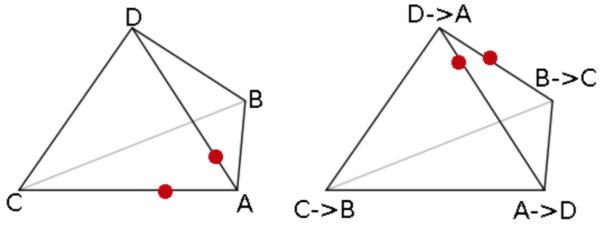


Abbildung 4.8: Abbilden eines Schnittes auf ein passendes Schnittmuster. Der Schnitt wird auf Muster B aus Abbildung 4.6(b) abgebildet.

Der Knoten A erscheint in der Kollisionserkennung zweimal, einmal als Endknoten der durchschnittenen Kante (A, C) und einmal als Endknoten der Kante (A, D). Aus dieser Information lässt sich A auf den Knoten D des Schnittmusters abbilden. D und B werden passend auf A oder B abgebildet. Bei diesem Beispiel ist es aufgrund der Symmetrie des Tetraeders egal, ob D auf A und C auf B oder D auf B und C auf A abgebildet wird. In der Abbildung wurde erstere Variante gewählt.

Auswirkungen der Tetraedernachbarn

Nachdem die neuen Knoten generiert und der Tetraeder auf ein passendes Schnittmuster abgebildet wurde, kann der Tetraeder unterteilt werden. Analog zu der Erstellung neuer Knoten muss bei der Erstellung neuer Kanten darauf geachtet werden, dass Kanten die von mehreren Tetraedern genutzt werden, nur von einem Tetraeder neu erstellt werden. Dazu wird überprüft ob eine Kante zwischen zwei Knoten, wie sie von einem Tetraeder benötigt wird, bereits existiert. Besonders zu beachten sind Schnitte über zwei oder mehr Kanten. Hier kommt es vor, dass eine Tetraederseite in ein Dreieck und Viereck geteilt wird. Das Viereck muss nun wiederum in zwei Dreiecke unterteilt werden, welche bei der Generierung der neuen Tetraeder die Seiten dieser darstellen. Bei der Unterteilung des Vierecks kann zwischen zwei Möglichkeiten entschieden werden. Wird nur ein Tetraeder geschnitten, kann eine der beiden Möglichkeiten gewählt werden. Da eine Tetraederseite jedoch von zwei Tetraedern genutzt werden kann, muss überprüft werden, ob der Nachbartetraeder bereits unterteilt wurde und die Kantenlegung dadurch festgelegt ist.

Zur Veranschaulichung sei in Abbildung 4.9 ein Ausschnitt eines Tetraeders T1 dargestellt, bei dem die Kanten (A, D) und (B, D) durchschnitten wurden. Diese Kanten werden unterteilt. Da in einem Tetraedernetz keine viereckigen Flächen existieren, muss eine zusätzliche Kante erstellt werden. Diese kann sowohl zwischen A und P2 wie auch zwischen B und P1 liegen. Wird die Seite ABD von einem zweiten Tetraeder T2 genutzt und wurde bei diesem bereits eine Kante zwischen A und P2 gelegt, muss T1 bei seiner Unterteilung die bereits vorhandene Kante nutzen.

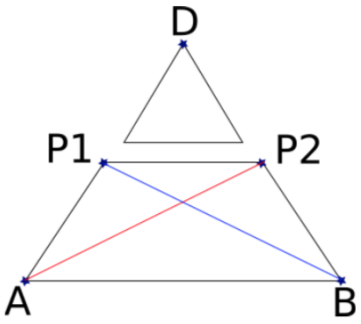


Abbildung 4.9: Mögliche Kanten zwischen P1 und B oder zwischen P2 und A.

Wurde T2 nicht vor T1 unterteilt, wird die Kantenlegung von T1 bestimmt und T2 übernimmt die Kante, wie sie bei der Unterteilung von T1 erstellt wurde. Dieser Fall tritt bei den Schnittmustern B, C, D und E aus Abbildung 4.6 auf.

Unterteilung der Primitive

Der letzte Schritt in dem Algorithmus ist die Unterteilung der einzelnen Primitive. Unterteilt werden Tetraeder, Kanten und Oberflächendreiecke. Damit das Volumen des Simulationsgitters während der Simulation erhalten bleibt, muss für jeden neu erstellten Tetraeder das Nullvolumen berechnet werden. Dasselbe gilt für die Unterteilung der Kanten. Bei diesen wird die Nulllänge errechnet. Bei der Bestimmung der Nullgrößen von Tetraedern und Kanten sind die in Abschnitt 4.2.1 berechneten Startpositionen der Knoten von entscheidender Bedeutung, da die Nullgrößen anhand dieser direkt berechnet werden können. Abschließend werden die an der Oberfläche getroffenen Dreiecke unterteilt und an der Schnittfläche neue Dreiecke erstellt.

Ausnahme bei der Unterteilung

Generell lassen sich alle Schnitte eines Tetraeders nach den beschriebenen Verfahren unterteilen. Eine Ausnahme hierbei bildet das Schnittmuster D aus Abbildung 4.6(d). Es kommt vor, dass die Kanten an den drei relevanten Seiten des Tetraeders durch die Nachbarn bereits so festgelegt wurden, dass eine Unterteilung des Tetraeders anhand der bestehenden Kanten nicht möglich ist. Abbildung 4.10 veranschaulicht das Problem an einer Skizze des betroffenen Teils des Schnittmusters. Tritt der Fall ein, wird ein weiterer Knoten im Inneren des Tetraeders erstellt, um den Tetraeder unterteilen zu können. Abbildung 4.10 stellt die Unterteilung des Schnittmusters D, wie es standardmäßig und bei der eben beschriebenen Ausnahme unterteilt wird, grafisch dar. Die standardmäßige Unterteilung benötigt drei Tetraeder weniger im Vergleich zu dem speziellen Fall.

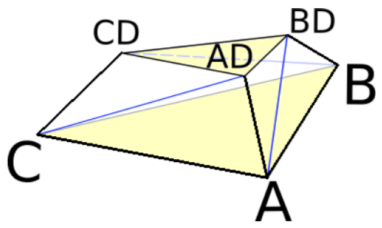


Abbildung 4.10: Keine Tetraederunterteilung möglich, da jeweils Kanten zwischen (A, BD), (B, CD) (C, AD) gelegt wurden. Dasselbe Problem tritt auf bei der Kantenlegung: (A, CD), (C, BD) (B, CD).

Analyse des Algorithmus

Listing 4.2 stellt für die Laufzeitbestimmung relevante Stellen des implementierten Algorithmus in Pseudocode dar. Die Anzahl der aktiven Tetraedern in $t - \Delta t$ wird mit n beschrieben, k steht für die Anzahl der in t kollidierten Tetraeder.

Die Abschnitte des Algorithmus, die in konstanter Zeit c ablaufen, können vernachlässigt werden. In der Implementierung wurde eine *map* aus der C++ Standard Library verwendet. Da binäre Suchbäume logarithmische Zugriffszeiten bieten, hat die Suche in einer der beiden Tabellen des Algorithmus jeweils eine Laufzeit von O(log(n)). Durch die Schleife über alle kollidierten Tetraeder resultiert eine Laufzeit von k · (O(log(n)) + O(log(k))). Da Tetraeder, die im Zeitpunkt $t-\Delta t$ aktiv waren, durch die kontinuierliche Kollisionserkennung in t wieder kollidieren, folgt $n \leq k$. Somit kann die erwartete Laufzeit auf O(k · log(k)) vereinfacht werden.

```
for every collided tetrahedron A do
  check if A in last_timestep_collision_list   O(log(n))
    for every link L of A
      if NOT L subdivieded                     O(log(k/2))
        subdivide L
      determine cutpattern and subdivide A     c
```

Listing 4.2: Analyse des Algorithmus

4.3 Mesh-Mesh-Kollisionen

Da sich die Form deformierbarer Objekte während der Simulation laufend verändert, kann diese nicht ohne Einschränkungen durch *Bounding Volumes* approximiert werden. Eine gängige Form, um Kollisionen zwischen Tetraedernetzen zu erkennen, ist die Berechnung der n Knoten eines Tetraedernetzes, die in einen der m Tetraeder eines anderen Tetraedernetzes eingedrungen sind. Der triviale Ansatz hat die Komplexität $O(n * m)$. Daraus ergibt sich die erste Schwierigkeit der Kollisionserkennung zwischen deformierbaren Objekten: Effizienz der Kollisionserkennung.

Bei der Kollision zwischen zwei deformierbaren Objekten ist der Ansatz nicht intuitiv wie bei der Kollision zwischen einem Festkörper und deformierbaren Objekten. Festkörper werden in der Simulation als unelastisch betrachtet. Das vereinfacht die Kollisionsbehandlung, da die Kollisionsfläche zwischen Festkörper und Softkörper exakt ermittelt werden kann. Im Unterschied zu Kollisionen mit Festkörpern, verformen sich bei Kollisionen zwischen deformierbaren Objekten alle

Objekte, die an der Kollision teilnehmen. Dabei sind Deformationen nicht nur von der Kraftübertragung der aktuellen Kollisionen abhängig, sondern auch von internen Kräften der jeweiligen Körper. Die exakte Bestimmung der Kollisionsfläche ist somit eine weitere Hürde.

Die Kollisionsauflösung muss nicht nur effizient, sondern auch exakt sein, um Vibrationseffekte an den Oberflächen der Objekte zu vermeiden. Diese entstehen, wenn einzelne Knoten bei der Kollisionsbehandlung zu weit aus dem Netz gedrückt werden und anschließend nicht auf der Kollisionsfläche liegen. Durch die Überbrückung des entstandenen Abstands zwischen den beiden Objekten und das erneute Eindringen und Rausdrücken kommt es zu dem benannten Effekt.

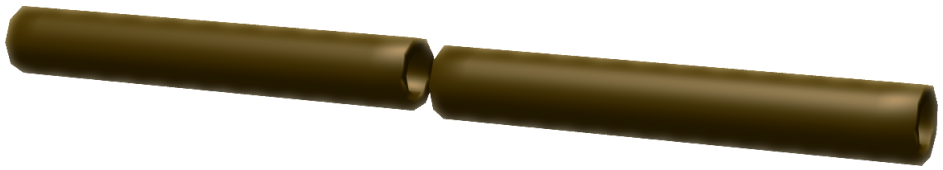
Kollisionserkennung

Im Folgenden werden zwei identische Blutgefäße A und B simuliert. A und B bestehen aus jeweils n Knoten und m Tetraeder. Das heißt, dass die Knoten p_A^i mit den Tetraedern q_B^j und die Knoten p_B^i mit den Tetraedern q_A^j auf Überschneidungen geprüft werden müssen. Dabei ist $0 \leq i \leq n$ und $0 \leq j \leq m$. Es folgt die Laufzeit $O(\frac{n}{2} \cdot \frac{m}{2})$. Da die Anzahl der Knoten im Tetraedernetz von der Anzahl der Tetraeder abhängt, lässt sich die Laufzeit vereinfacht mit $O(m^2)$ angeben.

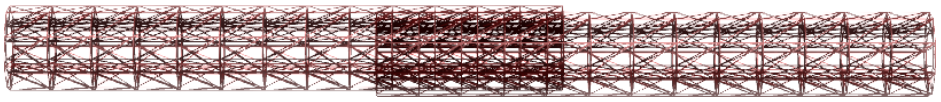
Um die echtzeitfähigkeit der Kollisionserkennung zu untersuchen, wird ein Versuch aufgebaut, bei dem sich zwei identische Tetraedernetze durch konstante Krafteinwirkung aufeinander zubewegen (Abbildung 4.11(a)). Die Tetraedernetze bestehen aus jeweils 252 Knoten und 702 Tetraedern. Durch die Krafteinwirkung kommt es zu Kollisionen der Tetraedernetze (Abbildung 4.11(b)). Diese werden vorerst nicht aufgelöst. Es wird die Zeit gemessen, die benötigt wird, um zu erkennen, ob sich ein Knoten in einem Tetraeder des anderen Mesh befindet.

Abbildung 4.12 misst die Zeit, die benötigt wird, um alle Kollisionen zu erkennen. Dadurch, dass die Auswahl möglicher Kollisionspartner vorab nicht untersucht und eingeschränkt wird, werden in jedem Zeitschritt alle Knoten auf Kollisionen mit allen Tetraedern überprüft. Dadurch ist die Anzahl der Kollisionstests abhängig von der Größe der Tetraedernetze. Schon bei vergleichsweise kleinen Meshes ist allein durch die Algorithmen für die Simulation und für die Kollisionserkennung zwischen den Meshes keine Echtzeit zu erreichen.

Um die Berechnungen für die Kollisionserkennung zu minimieren, wurde der *Spatial Hashing*-Algorithmus, der in [THM+03] veröffentlicht wurde, implementiert. Dazu wird die Position jedes Knoten durch eine selbstdefinierte Zellgröße der SSD geteilt und die einzelnen Komponenten der Position werden auf die



(a) Darstellung des gerenderten Mesh.



(b) Darstellung der Knoten und Tetraederkanten.

Abbildung 4.11: Mesh-Mesh-Kollisionserkennung. Versuchsaufbau: Zwei identische Tetraedernetze liegen sich marginal versetzt gegenüber. Die Meshes bewegen sich durch konstante Krafteinwirkung aufeinander zu (a). Dabei tunneln sie sich (b).

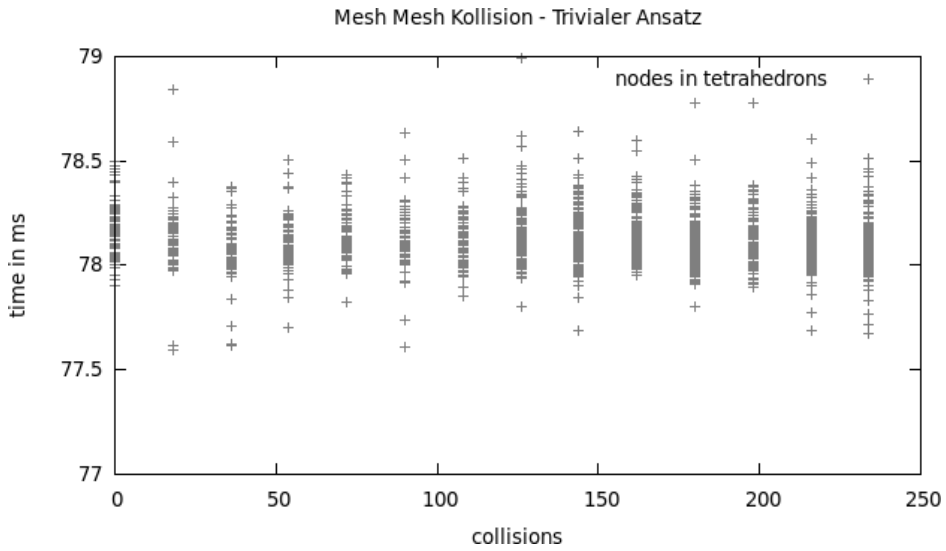


Abbildung 4.12: Test-Szenario: Mesh-Mesh-Kollisionen ohne Broad-Phase.

nächste ganze Zahl gerundet. Eine Funktion $h : R^3 \rightarrow R$ berechnet aus der gerundeten Postion einen eindimensionalen Wert. Dieser Wert dient als Schlüssel für die Hash-Tabelle, in der die Knoten gespeichert werden. Es wurde die in [THM+03] als Beispiel erwähnte Hashfunktion implementiert, n beschreibt die Größe der Tabelle:

$$hash(p_i^x, p_i^y, p_i^z) = (73856093 p_i^x \; XOR \; 19349663 p_i^y \; XOR \; 83492791 p_i^z) \mod n. \tag{4.4}$$

Im nächsten Schritt werden die AABB der Tetraeder durchlaufen, dabei wird für jeden diskreten Wert, zwischen dem Minimum und dem Maximum der AABB der Hash-Wert berechnet und mit den Einträgen in der Tabelle verglichen. Ein Treffer in der Hash-Tabelle sagt aus, dass Kollisionen zwischen dem Tetraeder und dem Knoten möglich sind. In diesem Fall wird der Tetraeder und die Knoten, die denselben Hash-Wert haben, für die Narrow-Phase der Kollisionserkennung vorgemerkt.

Abbildungen 4.13 zeigt die Ergebnisse für das obige Szenario, dieses Mal mit Verwendung des *Spatial Hashing*-Algorithmus.

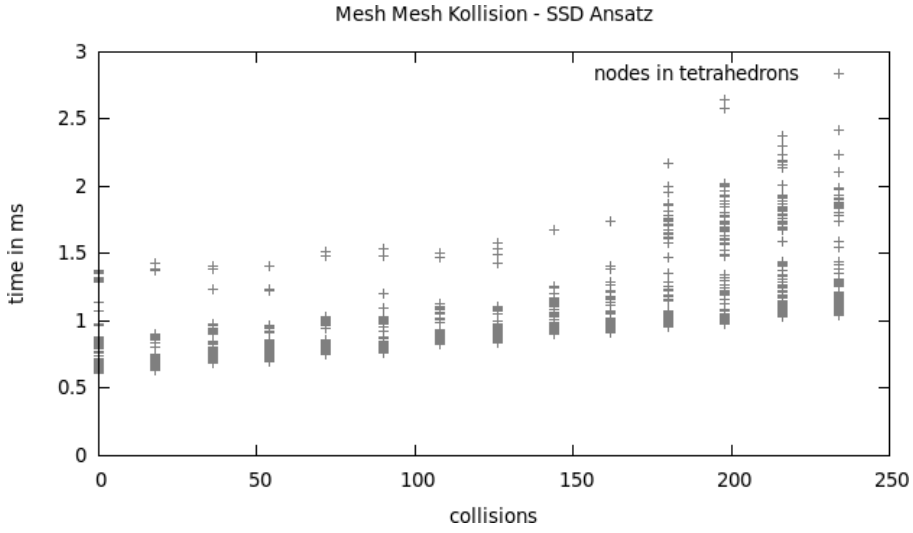
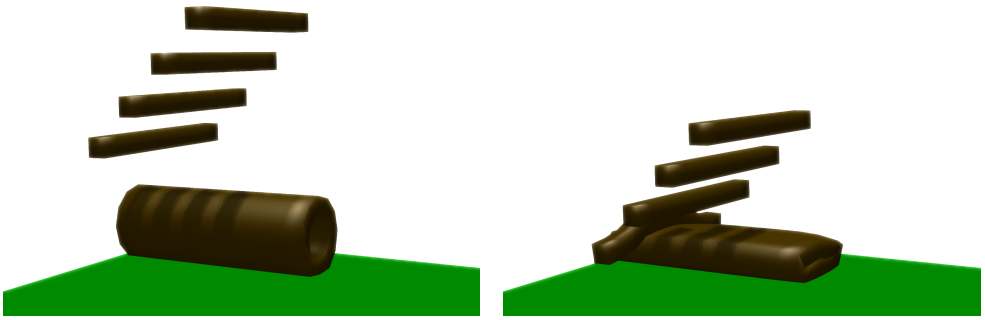


Abbildung 4.13: Test-Szenario: Mesh-Mesh-Kollisionen mit Spatial Hashing Algorithmus nach[THM+03].

Die folgende Kollisionsbehandlung setzt exaktes Wissen über eine Kollision voraus. Da sich bei einer Kollisionen zwischen zwei Tetraedernetzen beide verformen, können die Knoten des einen Mesh nicht ohne Weiteres auf die aktuelle Oberfläche des andren Mesh „verschoben“ werden. Um die folgenden Methoden

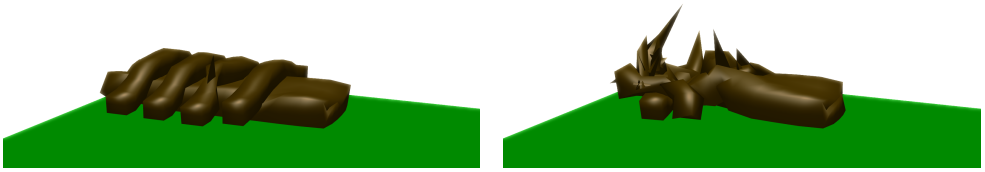
zu erklären, wurde ein Testszenario erstellt, das aus fünf Tetraedernetzen besteht. Vier davon sind quaderförmig und das fünfte wird durch die dreidimensionale Hülle eines Zylinders beschrieben. Durch Einwirkung von Gravitation fallen die Netze aufeinander und kollidieren an ihren Oberflächen und mit der Ebene. Beim Zylinder kommt es im Inneren zusätzlich zu Selbstkollisionen. Abbildung 4.14 zeigt den Aufbau des Szenarios.



(a) (b)

Abbildung 4.14: Test-Szenario – Mesh-Mesh-Kollisionen

Als trivialer Ansatz wurde bei der Kollision zwischen zwei Tetraedernetzen die Kollisionsfläche auf halbem Weg zwischen Oberfläche und eingedrungenem Knoten berechnet. Für die Kollisionsantwort wurden lineare *Penalty*-Kräfte verwendet (Abbildung 4.15).



(a) (b)

Abbildung 4.15: Test-Szenario: Mesh-Mesh-Kollisionen

Dieser Ansatz scheitert daran, dass die Kollisionsfläche für benachbarte Knoten variieren und somit starke interne Kräfte entstehen. Die ungenaue Kollisionsbehandlung auf Basis einer *Penalty*-Methode verstärken diesen Effekt, wodurch der numerische Fehler während des Integrationsschrittes immer größer wird und die Simulation aufgrund der divergierenden Kräfte schließlich abbricht.

In [HTK+04] wurde ein Algorithmus veröffentlich, der nicht nur den kürzesten Weg eines kollidierten Knotens zu der am nächstgelegenen Oberfläche berücksichtigt, sondern auch die umliegende Topologie des Knotens. Es werden die anliegenden Tetraederkanten eines Knotens betrachtet. Es werden die exakten Schnittpunkte dieser Kanten mit den Oberflächendreiecken berechnet und anhand des Abstandes zwischen dem Kollisionspunkt der Kante und der aktuellen Position des kollidierten Knotens durch die Gewichtungsfunktion $\omega(\vec{t_j}, \vec{p_i})$ unterschiedlich gewichtet. Das ist wie folgt definiert (Gleichungen 4.5, Gleichungen 4.6, Gleichungen 4.7 und 4.8 aus [HTK+04]):

$$\omega(\vec{t_j}, \vec{p_i}) = \frac{1}{\|\vec{t_j} - \vec{p_i}\|^2}. \tag{4.5}$$

Dabei ist $\vec{t_j}$ durch den Schnittpunkt der Tetraederkante mit der Oberfläche des Mesh und $\vec{p_i}$ durch die Position des kollidierten Knotens definiert. Mit der Normalen $\vec{n_i}$ des Oberflächendreiecks, das mit der Tetraederkante t_j kollidiert, kann die Eindringtiefe von p_i berechnet werden:

$$l(\vec{cp}) = \frac{\sum_{i=1}^{k}(\omega(\vec{p_i}, \vec{cp}) \cdot (\vec{p_i} - \vec{cp}) \cdot \vec{n_i})}{\sum_{i=1}^{k} \omega(\vec{p_i}, \vec{cp})}. \tag{4.6}$$

Weiter werden alle Normalen der Oberflächendreiecke, die an der Kollision teilnehmen gewichtet:

$$\vec{\hat{r}}(\vec{p_i}) = \frac{\sum_{i=1}^{k}(\omega(\vec{p_i}, \vec{cp}) \cdot \vec{n_i})}{\sum_{i=1}^{k} \omega(\vec{p_i}, \vec{cp})}. \tag{4.7}$$

Dadurch wird der normierte Vektor $\vec{r}(\vec{p_i})$ bestimmt, der die Richtung für die Kollisionsbehandlung bestimmt.

$$\vec{r}(\vec{p_i}) = \frac{\hat{r}(\vec{p_i})}{\|\hat{r}(\vec{p_i})\|}. \tag{4.8}$$

Durch die Integration des Algorithmus aus [HTK+04] in die Phase der Kollisionserkennung wird die Simulation stabil. Die Simulation bricht nicht ab, und Ungenauigkeiten können ausgeglichen werden. Trotzdem kommt es aufgrund der linearen *Penalty*-Kräfte, die für die Kollisionsbehandlung im trivialen Ansatz verwendet wurden, gelegentlich zu starken Vibrationseffekten des Mesh. Um diesem Effekt entgegenzuwirken, wurden die Kollisionsbehandlung untersucht und die Rückstellungskräfte mit Hinblick auf das eingesetzte Integrationsverfahren implementiert.

Kollisionsbehandlung

Die Kollisionsantwort wurde nach der Idee des in [GBT06] vorgestellten Verfahrens implementiert. Als Basis dafür dient das numerische Integrationsverfahren, das in der Simulation benutzt wird. Voraussetzung ist, dass die kollisionsfreie Position von p_i bekannt ist. In der vorliegenden Arbeit wurde das VVV verwendet. Um die Kraft zu berechnen, die benötigt wird, um die gewünschte Position für einen Knoten zu erreichen, wird Gleichung 3.19 in die Gleichung 3.20 substituiert (Die Gleichungen des VVV werden hier aus Gründen der Übersicht ein weiteres Mal genannt):

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + (\vec{a}_i(t) + \vec{a}_i(t + \Delta t))\frac{\Delta t}{2}, \tag{3.19}$$

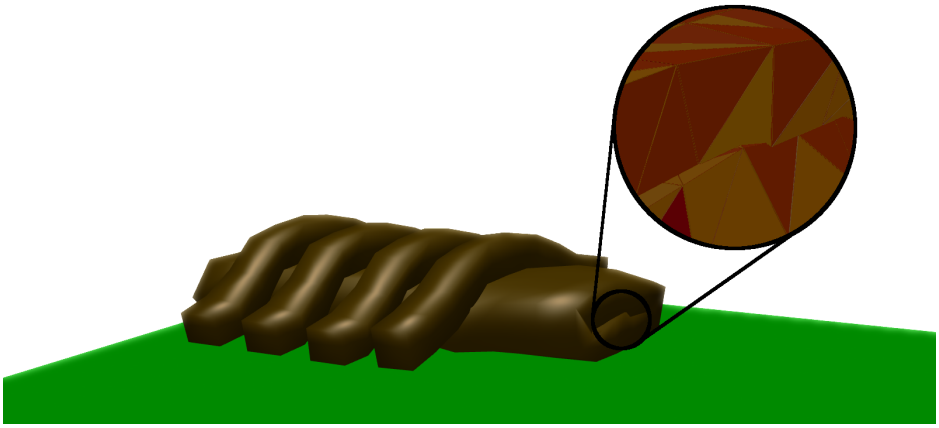
$$\vec{p}_i(t + \Delta t) = \vec{p}_i(t) + \vec{v}_i(t + \Delta t) \cdot \Delta t + \vec{a}_i(t + \Delta t) \cdot \frac{\Delta t^2}{2}. \tag{3.20}$$

Daraus kann nun die erforderliche Kraft berechnet werden, die zusätzlich auf den Knoten wirken muss:

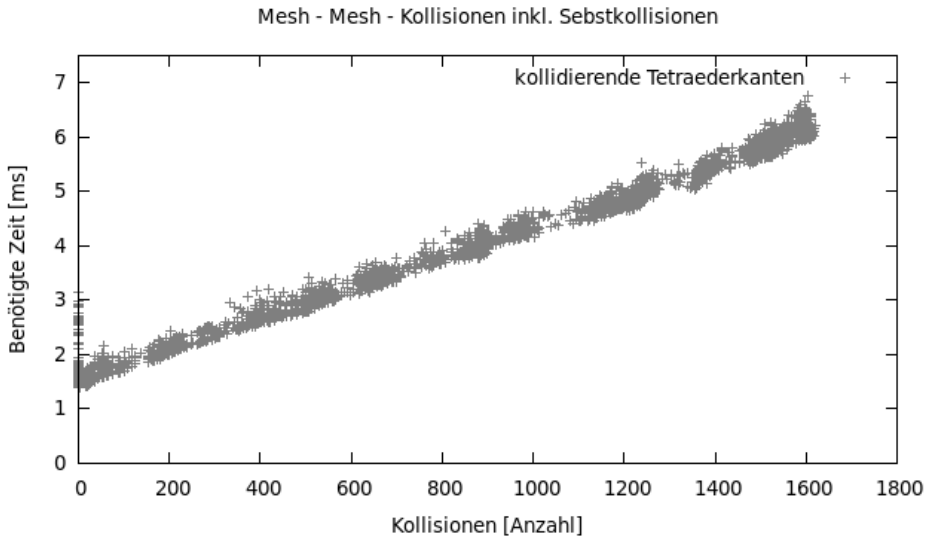
$$\vec{p}_i\,(t + \Delta t) = \vec{p}_i\,(t) + \left(\vec{v}_i\,(t) + (\vec{a}_i\,(t) + \vec{a}_i\,(t + \Delta t)) \cdot \frac{\Delta t}{2}\right) \cdot \Delta t + \vec{a}_i\,(t + \Delta t) \cdot \frac{\Delta t^2}{2}, \tag{4.9}$$

$$\vec{a}_i\,(t+\Delta t) = \left(\frac{\vec{p}_i\,(t+\Delta t) - \vec{p}_i\,(t)}{\Delta t} - \vec{v}_i\,(t)\right) \cdot \frac{1}{\Delta t} - \frac{\vec{a}_i\,(t)}{2}. \tag{4.10}$$

Für die Berechnung muss zum Zeitpunkt $t + \Delta t$ die Position $\vec{p}_i\,(t)$, die Geschwindigkeit $\vec{v}_i\,(t)$ und die Kraft $\vec{F}_i\,(t)$ aus t gespeichert werden.



(a)



(b)

Abbildung 4.16: Test-Szenario: Mesh-Mesh-Kollisionen

In Abbildung 4.16(a) sieht man Artefakte an der Oberfläche. Diese treten auf, wenn sich kein Knoten in einem Tetraeder befindet, jedoch Tetraederkanten Überschneidungen aufweisen. An sich können die Artefakte für die Simulation einer vaskulären Anastomose vernachlässigt werden, da sie bei den Vergrößerungen während einer Operation nicht ins Auge fallen. Des Weiteren sind die Grenzen der Gefäße auch im Realen nicht unbedingt eindeutig voneinander abgrenzbar.

Der Algorithmus wurde in "Real-time simulation of blood vessels and connective tissue for microvascular anastomosis training"[Sis12] veröffentlicht.

Problematisch werden die Artefakte jedoch, falls sich ein Knoten in der Nähe eines Blutgefäßes weiter über die Endung des Blutgefäßes schiebt, sodass der Knoten eingebunden wird. Bei elastischem Gewebe kommt es dabei zu Sprüngen. Wirken extrem hohe Kräfte, verhaken sich die Tetraedernetze sogar und die Kollisionen können von der Kollisionsbehandlung nicht ohne Weiteres aufgelöst werden.

Das hängt mit der Topologie der Blutgefäße zusammen. Da die Blutgefäßwand aus einer Schicht von Tetraedern besteht, scheitert der Algorithmus beim Versuch, tief eingedrungene Knoten korrekt auf das Blutgefäßende aufzulösen, sodass die Enden aufeinander liegen. Um diesem Problem entgegenzuwirken, wurde folgender Algorithmus implementiert, der die Gegebenheit ausnutzt, dass bei der *End-zu-End*-Anastomose die Blutgefäße durch Klammern befestigt sind. Ihre Enden, welche miteinander vernäht werden sollen, liegen sich gegenüber. Dadurch können Kollisionen nur an den Enden der Blutgefäße auftreten. Die Positionen der Knoten, die sich an den Enden des Blutgefäßes befinden, und die Normalen der Oberflächendreiecke, die sich an der vorderen Schicht des Blutgefäßes befinden, werden gemittelt. Die Position des Knoten definiert zusammen mit der Normalen eine Ebene. Kommt es nun zu Kollisionen zwischen den Blutgefäßen. Werden die Knoten aller kollidierenden Tetraederkanten überprüft, ob sie sich vor oder hinter dieser definierten Ebenen befinden. Befinden sie sich dahinter, wird die Kollision aufgelöst. Dadurch vereinfacht sich die Kollisionserkennung und -auflösung auf die Berechnung zwischen Positionen und Ebene.

Die in dem aktuellen Abschnitt beschriebenen Methoden sind dafür ausgelegt, mit einer Fadensimulation, wie sie in [Hüs11] beschrieben wird, zu interagieren. In Abschnitt 5.2.2 werden aktuelle Forschungsergebnisse aus der Arbeit [Hüs13] von N. Hüsken vorgestellt. Diese verwenden den in diesem Abschnitt vorgestellten Algorithmus für das Erkennen und Behandeln der Kollisionen zwischen den einzelnen Blutgefäßen.

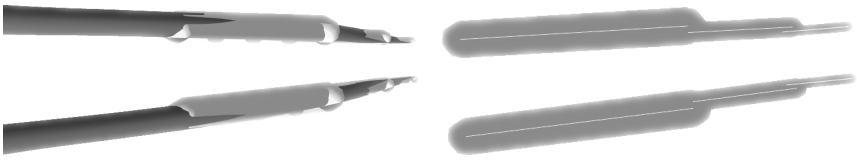
4.4 Instrument-Gewebe-Kollisionen

Im Folgenden werden Methoden für die Modellierung der Interaktion zwischen Gewebe und Instrumenten beschrieben. Bei den Instrumenten, die während der Operation zum Einsatz kommen, handelt es sich um Pinzetten, Mikroscheren und Dilatatoren.

Eine Mikroschere hat zwei scharfe Schneiden. Mit diesen wird Gewebe durchtrennt. Je nach Beschaffenheit des Gewebes bzw. der Schärfe der Kanten kann es vor dem Überwinden der Oberflächenspannung bei einem Schnitt zu Deformationen des Gewebes kommen. Die Rückseite einer Mikroschere ist dagegen stumpf. Bei Kollisionen mit ihr verformt sich das Gewebe, das führt jedoch meist nicht zu topologischen Veränderungen.

Im Gegensatz zu einer Schere wird die Pinzette dafür benutzt, Gewebe zu greifen. Dadurch wird Gewebe positioniert, um vernäht oder geschnitten zu werden. Abgetrennte Teile können aus der Wunde entfernt werden. Des Weiteren wird die Pinzette dazu benutzt, Gewebe zu manipulieren. Wird sie fest zusammengedrückt, kann sprödes Gewebe durchtrennt werden. Beim Freipräparieren wird die leicht geöffnete Pinzette dazu verwendet, die Adventitia entlang der Media nach hinten zu verschieben.

Abhängig von einem Instrument und seiner Bewegung, kann Gewebe bewegt oder topologisch verändert werden. Um die genannten Eigenschaften zu modellieren, werden Modelle der Instrumente mit BCs approximiert (Abbildung 4.18(a) und 4.17(a)). Auf der medialen Achse der BCs wird einer Linie definiert, die den Auslöser für topologische Änderungen darstellt. Diese Linie sei fortan als *Line Of No Return (LONOR)* definiert. In den BCs der Instrumente in den Abbildungen 4.17(b) und 4.18(b) sind diese Linien veranschaulicht.



(a) Mit Pinzettenmodell. (b) Ohne Pinzettenmodell.

Abbildung 4.17: Bounding Capsules der Pinzette

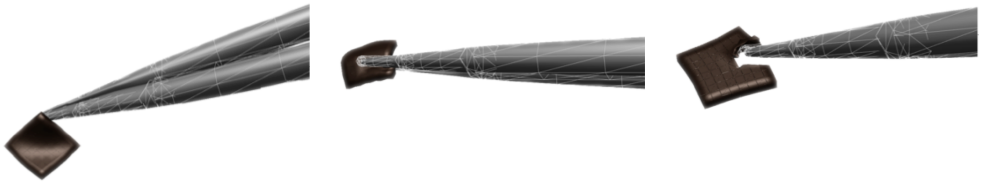
Für die Broad-Phase der Kollisionserkennung wird der *Spatial Subdivision* Algorithmus, der in Abschnitt 4.3 für die Kollisionserkennung zwischen den Tetraedernetzen verwendet wird, benutzt. Dazu werden die AABB der BCs diskretisiert und auf Kollisionen mit den Oberflächendreiecken der Tetraedernetze untersucht. Bei

(a) Mit Scherenmodell. (b) Ohne Scherenmodell

Abbildung 4.18: Bounding Capsules der Instrumente

einer Übereinstimmung einer BC mit einem Wert in der Hash-Tabelle werden die exakten Kollisionen berechnet.

Werden Kollisionen erkannt, muss zwischen drei verschiedenen Möglichkeiten für die Manipulation der Tetraedernetze entschieden werden: ein- bzw. wegdrücken (bewegen) (Abbildung 4.19(a)), greifen (Abbildung 4.19(b)) und topologische Änderungen (trennen) (Abbildung 4.19(c)). Um die Methoden im Folgenden aufzuzeigen, werden zwei BCs $BC1$ und $BC2$ mit jeweils einer *LONOR* bei der Interaktion mit einem Tetraedernetz X beschrieben. Dafür besteht das Netz im Folgenden aus einem Tetraeder mit vier Knoten, sechs Kanten und vier Oberflächendreiecken.



(a) Bewegen. (b) Greifen. (c) Trennen.

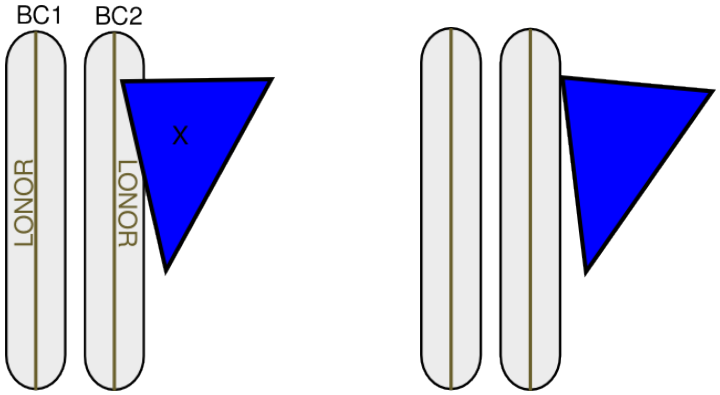
Abbildung 4.19: Drei Möglichkeiten der Kollisionsbehandlung

Obwohl die folgenden Methoden auf dreidimensionale Objekte angewendet werden, sind die Skizzen des Kapitels aus Gründen der Übersichtlichkeit zweidimensional.

4.4.1 Bewegen

X dringt im Zeitpunkt t in $BC2$ ein, ist aber nicht in Kontakt mit der *LONOR* von $BC2$ und hat diese auch nicht zwischen dem Zeitschritt $t - \Delta t$ und t passiert. Die Kollisionsoberfläche auf der BC wird errechnet. Für die Knoten von X werden Rückstellkräfte berechnet. Diese wurden mit der selben Methode aus Abschnitt 4.3 berechnet, welche die Kraft für das Erreichen der exakten Zielposition eines Knoten auf der Basis des verwendeten Integrationsverfahrens herleitet. In Abbil-

dung 4.20(a) wird die Kollision zwischen X und $BC2$ veranschaulicht und in Abbildung 4.20(b) die Position von X nach Beendigung des Integrationsschrittes.



(a) Kollisionen zwischen X und $BC2$ werden erkannt.

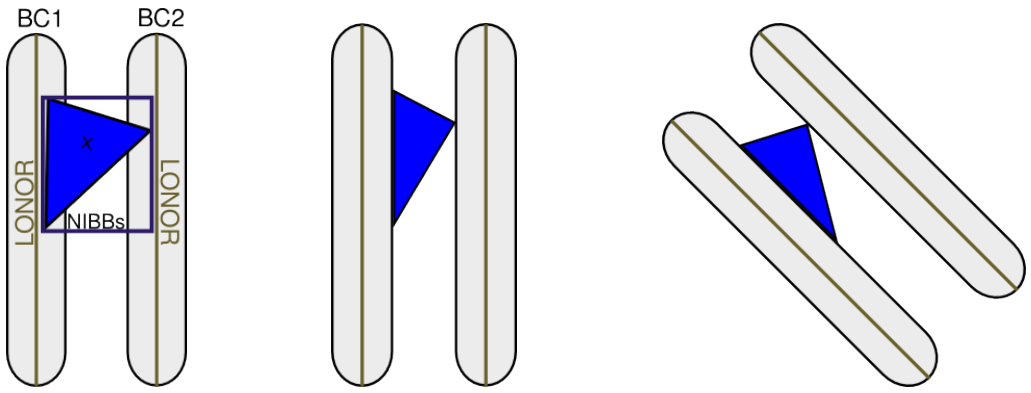
(b) Kollisionen zwischen X und $BC2$ werden aufgelöst.

Abbildung 4.20: Kollisionsauflösung: Bewegen.

4.4.2 Greifen

Der Tetraeder X ist in beide BCs $BC1$ und $BC2$ eingedrungen, aber ist im Zeitpunkt t nicht in Kontakt mit der $LONOR$ und hat diese in Δt nicht passiert. Im Folgenden muss X in einen gültigen Zustand außerhalb von $BC1$ und $BC2$ gebracht werden.

Um das Greifen von X zu simulieren, muss zusätzlich Haftreibung in das Modell implementiert werden. Um diese zu integrieren wird eine *Bounding Box* zwischen den $LONOR$s von $BC1$ und $BC2$ erstellt (Abbildung 4.21(a)). Alle Knoten, die sich innerhalb der Box befinden, werden gespeichert. Anschließend werden alle Knoten der kollidierenden Oberflächendreiecke, wie in Abschnitt 4.4.1 beschrieben wurde, im Zeitpunkt t verschoben (Abbildung 4.21(b)). Im folgenden Zeitschritt $t + \Delta t$ bekommen die gespeicherten Knoten eine zusätzliche Kraft, die relativ zu der Bewegung der BCs berechnet wird. Das führt dazu, dass sich das Gewebe relativ zu den Bewegungen der BCs bewegt (Abbildung 4.21(c)). Entfernen sich $BC1$ und $BC2$ voneinander gleitet das Gewebe langsam raus und wird nicht mehr gegriffen.



(a) Detection (b) Response (c) Friction

Abbildung 4.21: Kollisionsauflösung: Greifen.

4.4.3 Trennen.

X ist in beide BCs eingedrungen und hat die *LONOR* eines oder beider BCs zwischen $t\Delta t$ und t passiert oder befindet sich zum Zeitpunkt t in Kontakt mit der *LONOR*. Durch den Kontakt oder durch das Passieren mit einer *LONOR* soll ein Algorithmus ausgelöst werden, der das Tetraedernetz topologisch verändert.

Dabei muss zwischen folgenden Fällen unterschieden werden: Entweder X wurde von $BC2$ über die *LONOR* von $BC1$ gedrückt oder der Tetraeder hat die *LONOR* von $BC1$ aufgrund der eigenen Bewegung überschritten. In den meisten Fällen ist beides der Fall, da der Tetraeder durch die Interaktion mit beiden BCs interagiert und sich die Knoten dadurch sowohl durch externe als auch durch interne Kräfte bewegen. Im Folgenden wird davon ausgegangen, dass X die *LONOR* von $BC1$ passiert hat, indem $BC2$ eine Kraft auf X ausgeübt hat. Dabei müssen drei Fälle berücksichtigt werden (Die Illustrationen sind im Folgenden aufgrund der Übersichtlichkeit in Vogelperspektive gezeichnet):

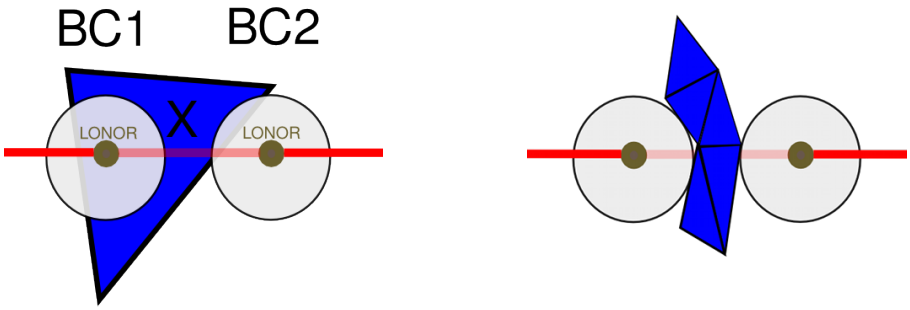
1. X ist im Zeitpunkt t in Kontakt mit der *LONOR* von $BC1$ (Abbildung 4.22(a).

2. X ist im Zeitpunkt t nicht in Kontakt mit der *LONOR* von $BC1$ hat diese aber in Δt passiert (Abbildung 4.23(a)).

3. X kollidiert sowohl mit der *LONOR* von $BC1$ als auch mit der von $BC2$. Die beiden BCs stehen selbst auch in Kontakt miteinander (Abbildung 4.24(a)).

Fall 1:

X ist im Zeitpunkt t in Kontakt mit der *LONOR* von $BC1$. Da das Tetraedernetz nicht zusätzlich in Kontakt mit der *LONOR* von $BC2$ ist, reißt X bis zu der Stelle, die die *LONOR* bereits passiert hatte ein. Abbildung 4.22(a) veranschaulicht die Kollision und Abbildung 4.22(b) den Tetraeder nach der topologischen Veränderung. Für die vorliegende Arbeit wurde der *Remeshing*-Algorithmus verwendet, der in 4.2 beschrieben wurde. Der Algorithmus für die topologischen Änderungen ist auswechselbar.



(a) X ist in Kontakt mit der *LONOR* von $BC1$.

(b) Ein Algorithmus für topologische Änderungen wird ausgelöst. Das Mesh wird unterteilt. Hier: Remeshing.

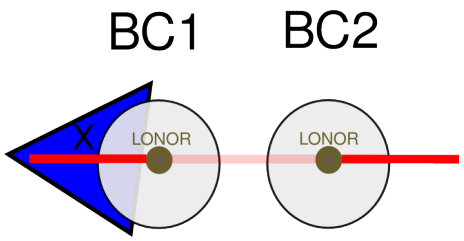
Abbildung 4.22: Fall 1 – Tetraederkollision mit der *LONOR*.

Bevor das Mesh verändert werden kann, muss die Kollisionsfläche festgelegt werden. Eine triviale kontinuierliche Kollisionserkennung, bei der die Kollisionsfläche durch die *LONOR* zum Zeitpunkt $t\Delta t$ und t festgelegt wird, scheitert, wenn sich der Tetraeder durch eine eigene Bewegung in Δt über die *LONOR* bewegt hat.

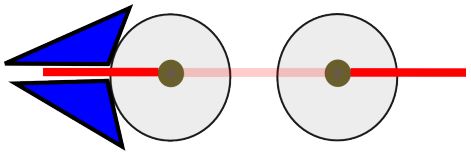
Um selbst diese Kollisionen zu erkennen, wird die Fläche, die durch die beiden *LONOR*s aufgespannt wird, gewählt. Dabei nimmt die Fläche, die sich zwischen den *LONOR*s befindet, nicht an der Kollisionserkennung teil. Die Fläche ist in der Abbildung 4.22(a) dunkelrot gekennzeichnet. Dabei werden Kollisionen mit den Teilen erkannt, die sich im Zeitpunkt $t\Delta t$ noch zwischen $BC1$ und $BC2$ befunden haben.

Fall 2:

Im Zeitpunkt t steht die *LONOR* von $BC1$ nicht in Kontakt mit X. Dadurch würden topologische Veränderungen durch die diskrete Kollisionserkennung nicht ausgelöst werden (Abbildung 4.23(a)). Zwei Fälle in Δt können die Situation zum Zeitpunkt t hervorrufen: Zum einen kann sich $BC1$ mit einer sehr hohen Geschwindigkeit bewegt haben, ohne dass X sich selber bewegt hat. Hier würde die Fläche, welche die *LONOR* in Δt aufspannt, ausreichen, um die Kollisionen zu erkennen. Zum Anderen kann X durch eine Kollision mit $BC2$ eine so hohe Kraft erfahren haben, dass X die *LONOR* passiert hat.



(a) Kollisionserkennung: X hat keinen Kontakt mit der *LONOR* in t.



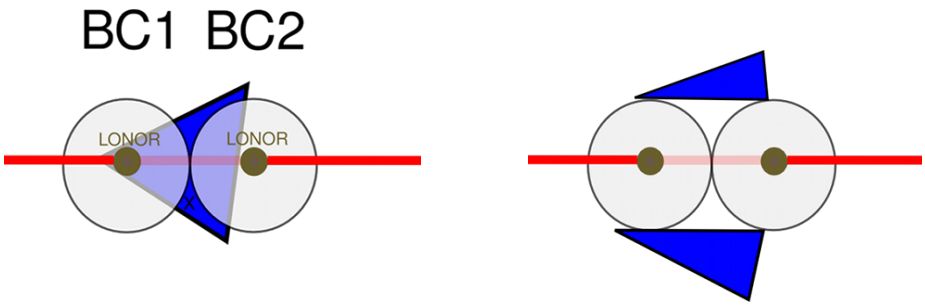
(b) Kollisionsantwort: X wird in zwei Teile geteilt.

Abbildung 4.23: Fall 2 – Vollständige Durchdringung der LONOR in Δt.

Um zu erkennen, ob X die *LONOR* in Δt passiert hat, werden die Geschwindigkeiten der Knoten von X überprüft. Zeigen diese in die Richtung der *LONOR*, wird Fall 1 ausgelöst und X erfährt eine Kraft und bewegt sich aus $BC1$. Zeigen sie in die entgegengesetzte Richtung, wird X unterteilt (Abbildung 4.23(b)). Die Kollisionsfläche ist dabei analog zu der in Fall 1.

Fall 3:

Je nachdem, wie „scharf “ das Instrument oder wie elastisch bzw. brüchig das simulierte Material ist, soll das Tetraedernetz komplett getrennt werden. wenn das Instrument geschlossen ist. Um diesen Fall zu berücksichtigen, wird das Tetraedernetz, sobald sich $BC1$ und $BC2$ berühren, entlang der Kollisionsfläche, die von den *LONOR*s von $BC1$ und $BC2$ aufgespannt werden, getrennt.



(a) X kollidiert mit $BC1$ und $BC2$. Diese berühren sich.

(b) X wird getrennt.

Abbildung 4.24: Fall 3 – Bounding Capsules berühren sich und kollidieren mit X.

Die Distanz der BCs, die dabei den aktuellen Fall 3 auslöst, ist variabel. Dadurch kann in der Simulation Gewebe von leicht brüchig bis gummiartiges Material modelliert werden. Abbildung 4.24(a) skizziert die Kollision, welche den aktuellen Fall auslöst und Abbildung 4.24(b) die Szene nachdem das Tetraedernetz getrennt und der Tetraeder aus den BCs geschoben wurde.

Der Algorithmus wurde in "Triggering Different Collision Response Algorithms stated at penetration depths"[Sis13] veröffentlicht.

KAPITEL

5

Kein Ding entsteht planlos, sondern aus Sinn und unter Notwendigkeit.

(Leukipp ca. 450 v. Chr. - 370 v. Chr.*)*

MicroSim – Ein mikrochirurgischer Trainingssimulator

MicroSim wurde von der Forschungsgruppe ViPA des Lehrstuhls für Informatik V der Universität Heidelberg gemeinsam mit der *VRmagic GmbH* konzipiert und entwickelt. Im Folgenden wird der prototypische Aufbau von *MicroSim* vorgestellt. Abschnitt 5.1 beschreibt die Hardware und die Benutzerschnittstelle von *MicroSim*. In Abschnitt 5.2 wird die Software von *MicroSim* vorgestellt. Anschließend werden die *VRm Libs* und der Aufbau des *MicroSim*-Framework vorgestellt. Abschnitt 5.2.2 stellt die vom Autor implementierten prototypischen Trainingsmodule vor. Diese basieren auf den Erfahrungen, die bei der Analyse von mikrochirurgischem Videomaterial, bei der Observation von mikrochirurgischen Operationen und während des Gespräches mit Chirurgen gemacht wurden.

Bei *MicroSim* handelt es sich um eine auf VR basierende Trainingsumgebung, die fächerübergreifend in der mikrochirurgischen Ausbildung eingesetzt werden soll. Dabei werden zwei Ziele verfolgt: Zum einen sollen allgemeine Fertigkeiten wie die Hand-Augen-Koordination geschult, zum anderen spezifische mikrochirurgische Operationstechniken trainiert werden.

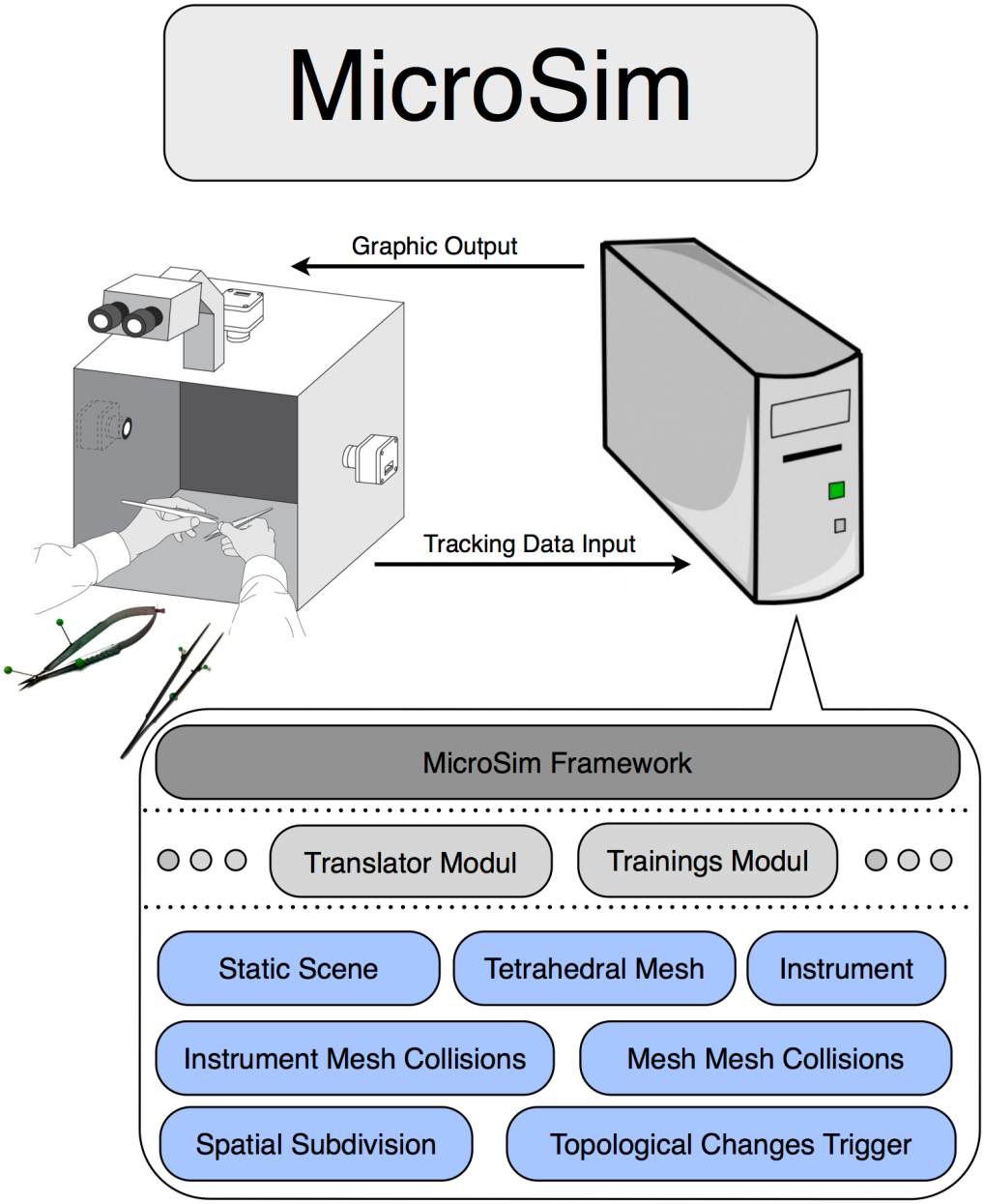
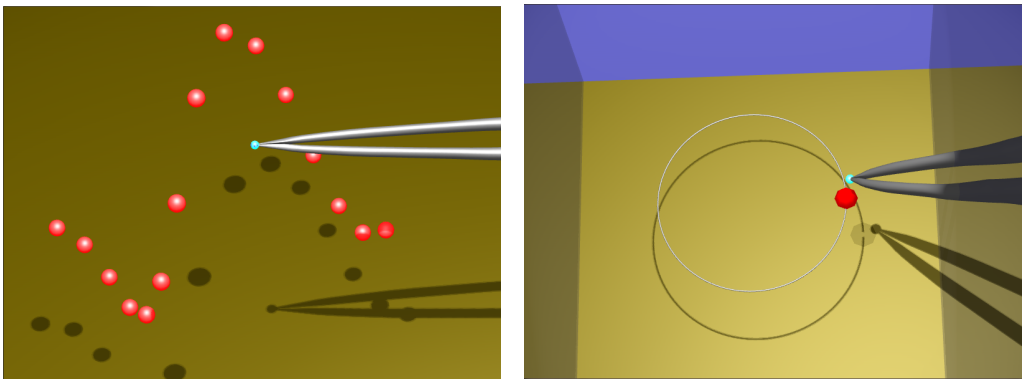


Abbildung 5.1: *MicroSim* – Überblick: Die Benutzerschnittstelle besteht aus echten mikrochirurgischen Instrumenten, die in einer quaderförmigen Box getrackt werden. Die Informationen des Trackings werden an einen PC übertragen, der die Simulation berechnet. Die gerenderte Simulationsszene wird über ein stereoskopisches Display ausgegeben. Die Basis der Software bildet das *MicroSim*-Framework, das mit Hilfe der *VRm Platform* erstellt wurde. Für die einzelnen Trainingsmodule werden verschiedene Objekte, z. B. deformierbare Objekte und Instrumente, und die dazugehörigen Simulationsalgorithmen geladen.

Zur Schulung der allgemeinen Fertigkeiten wurden abstrakte Trainingsmodule von der *VRmagic GmbH* entwickelt und in den Simulator implementiert. In diesen muss der Benutzer z. B. virtuelle Kugelobjekte mit der Pinzette in einem vorgegebenem Winkel treffen und in den Kugeln verharren, bis diese ihre Farbe wechseln (Abbildung 5.2(a)). Zusätzlich kann ein vorgegebener Pfad nach dem Vorbild „Der Heiße Draht “ mit einem Instrument abgefahren werden (Abbildung 5.2(b)).



(a) Abstraktes Modul 1. (b) Abstraktes Modul 2.

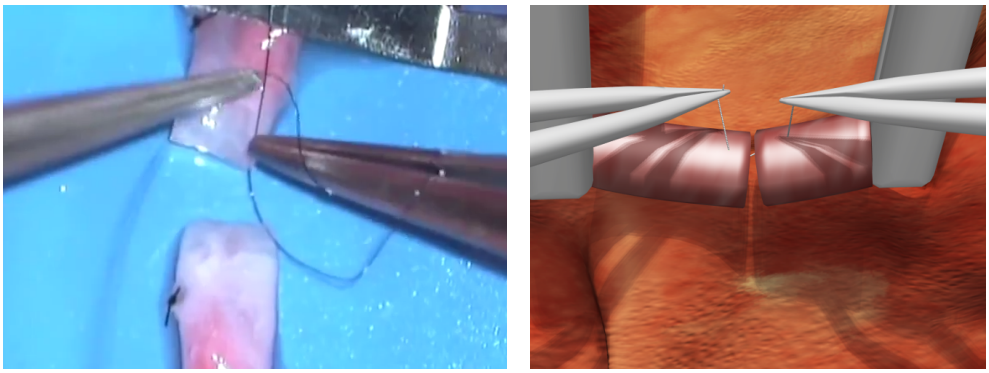
Abbildung 5.2: MicroSim – Abstrakte Trainingsmodule

Für das Training spezifischer Operationstechniken werden Trainingsmodule für das Freipräparieren von Blutgefäßen entwickelt, die in Abschnitt 5.2.2 vorgestellt werden. Ergänzt werden die Module durch die Arbeit [Hüs13] von *N. Hüsken*, die Methoden für die Simulation mikrochirurgischen Nahtmaterials und dessen Interaktion mit Tetraedernetzen behandelt. Mit diesen Methoden werden Trainingsmodule erstellt, an denen Nahtvorgänge trainiert werden können. Abbildung 5.3 demonstriert die Ähnlichkeit einer möglichen Szene aus *MicroSim* im Vergleich zu einem Bild einer realen mikrochirurgischen Operation.

5.1 *MicroSim*-Hardware

Die Benutzerschnittstelle von Trainingssimulatoren kann fächerübergreifend oder fachspezifisch konzipiert werden. Zum Beispiel wird bei *Eyesi Surgical* der Firma *VRmagic GmbH* das Auge eines Modellkopfes für die Eintrittsstelle der Instrumente verwendet. Dadurch ist der Simulator *Eyesi Surgical* äußerlich als Trainingssimulator für Operationstechniken in der Augenheilkunde erkennbar. Die Benutzerschnittstelle von *MicroSim* wurde im Gegensatz dazu fächerübergreifend konzipiert. Die Hülle des Simulators ist keinem menschlichen Körperteil nachempfunden, sondern neutral konzipiert. Diese wird im Folgenden auch als

Trackingbox bezeichnet, da in dieser die Position und Ausrichtung der Instrumente festgestellt wird.



(a) Mikrochirurgische Operation. (b) Training an MicroSim.

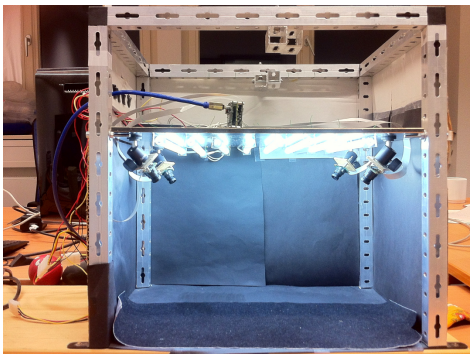
Abbildung 5.3: Vergleich der Sicht einer realen Operation (5.3(a)) mit der Sicht auf einer von *MicroSim* generierten Operationsszene (5.3(b)).

Der Innenraum der Trackingbox ist zur Benutzerseite offen und hohl, sodass er dem Benutzer uneingeschränkten Zugang bietet. Des Weiteren besteht die Benutzerschnittstelle aus Instrumenten, die in einer mikrochirurgischen Operation verwendet werden. Der Benutzer kann die Instrumente im Innenraum der Box frei bewegen, da auf ein Force-Feedback-System verzichtet wurde. Die Gründe dafür wurden in Abschnitt 1.2 genannt.

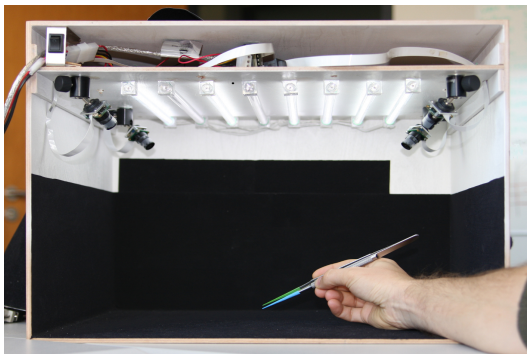
Durch die freie Benutzung der Instrumente und die Tatsache, dass taktiles Feedback bei mikrochirurgischen Operationen meist nur bei fehlerhaftem Handeln zu spüren ist, kann durch den gewählten Aufbau von *MicroSim* ein hoher Grad an Immersion erreicht werden. Für die vorliegende Arbeit wurde der erste Prototyp der Trackingbox (Abbildung 5.4(a)), der von *O. Schuppe* gebaut wurde, verwendet. Der Innenraum ist 42 cm breit, 35 cm tief und 27 cm hoch. Abbildung 5.4(b) zeigt ein aktuelles Modell der Trackingbox.

Die Bewegungen der Instrumente werden mit Hilfe eines optischen Trackingsystems registriert. Es wird eine Multisensor-Kamera der *VRmagic GmbH* verwendet. Die Sensoren der Kamera werden an den oberen vier Ecken der Box befestigt, um ein möglichst großes Tracking-Volumen erreichen zu können. Die Kamera besitzt ein Field Programmable Gate Array (FPGA) für die Vorverarbeitung und ist mittels *USB* an einen Computer angeschlossen. Der Innenraum der Box ist mit schwarzem Fleece ausgekleidet, um die Instrumente besser segmentieren zu können. Die Beleuchtung findet über Kaltkathodenröhren statt, die an der Decke der Box angebracht sind.

Für das Tracking der Instrumente wurde ein markerbasiertes Tracking von *O. Schuppe* entwickelt und implementiert [Sch12]. In Abbildung 5.4(c) ist im oberen Teil des Bildes die verwendete Pinzette mit Markern zu sehen, wie sie für die vorliegende Arbeit verwendet wurde. Auf derselben Abbildung ist eine weitere Pinzette abgebildet, die für ein markerloses Tracking bestimmt ist. Diese wird für Operationen benötigt, bei denen Instrumente auf engstem Raum zusammenarbeiten müssen. Bei diesen Operationen kann es zu Kollisionen der Pinzettenmarker des markerbasierten Trackings kommen. Diese Kollisionen wirken sich für den Benutzer negativ auf die Intensität der Immersion im Simulator aus.



(a) Erster Prototyp der Trackingbox.



(b) Aktuelle Version der Trackingbox.

(c) Benutzerschnittstelle: Originalinstrumente mit und ohne Marker.

Abbildung 5.4: *MicroSim* – Die Hardware

Da das markerlose Trackingverfahren während der Ausarbeitung der vorliegenden Arbeit noch in der Entwicklung war, wurde das markerbasierte Tracking verwendet. Das markerbasierte Trackingverfahren konnte vom Autor problemlos für das Tracking weiterer Instrumente verwenden werden, wie z. B. kleinere Pinzetten (Abbildung 5.5(a)) und Mikroscheren (Abbildung 5.5(b)).

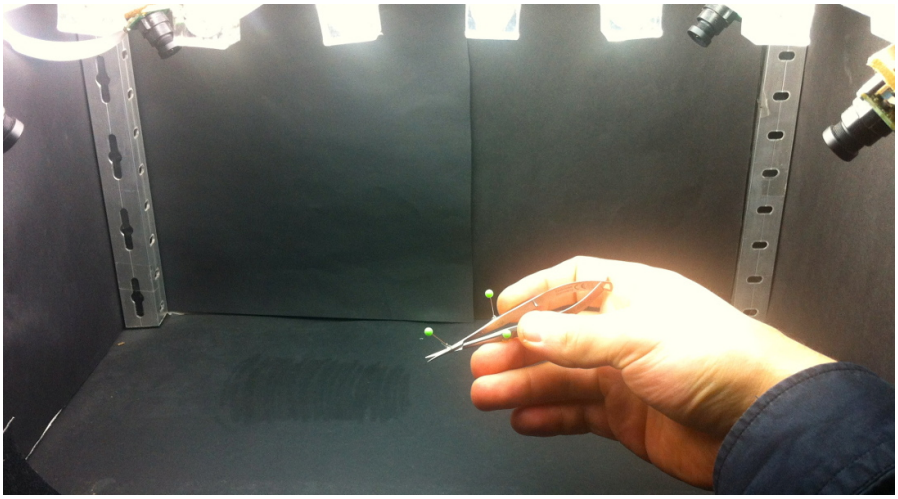
Auf der Box ist ein stereoskopisches Display befestigt. Durch dieses werden die gerenderten Bilder der Simulation vom Benutzer dreidimensional wahrgenommen. Für den Benutzer entsteht der Eindruck, er würde durch ein Mikroskop schauen. Die Ähnlichkeit der Haltung, die ein Benutzer von *MicroSim* im Ver-

gleich zu einem Chirurgen bei einem mikrochirurgischen Setup hat, wurde in Abbildung 1.2 veranschaulicht.

Für die vorliegende Arbeit fanden die Berechnungen auf einem Intel® Core™ i7 CPU mit 2.67 GHz, 6 GB Memory und einer NVidia® GeForce GTX 285 Grafikkarte statt. *MicroSim* ist jedoch nicht auf diese Rechnerarchitektur beschränkt, sondern läuft auf allen Rechnern, die mit den *VRm Libs* kompatibel sind.

(a) Mikropinzette mit Markern.

(b) Mikroschere mit Markern.



(c) Schere in der Trackingbox.

Abbildung 5.5: Implementierung des Trackings für weitere Instrumente.

5.2 *MicroSim*-Software

MicroSim wurde mit Hilfe der *VRm Platform* erstellt.

VRm Platform

Die *VRm Platform* ist ein Framework, mit dem das Grundgerüst eines Simulators schnell erstellt werden kann (vgl. [Bei12]). Dahinter verbirgt sich die Idee,

dass sich viele grundlegende Eigenschaften medizinischer Trainingssimulatoren wiederholen. Beispielsweise benötigen alle Ein- und Ausgabesysteme ein Bewertungssystem. Nach dem Baukastenprinzip lassen sich verschiedene Funktionalitäten je nach Anforderungen des Simulators als Plug-in laden. Dadurch wird die Logik vom restlichen Aufbau der Trainingsmodule getrennt. Einen weiteren Vorteil, den die *VRm Platform* durch Ihren Einsatz in verschiedenen Simulatoren bietet, ist die simulatorübergreifende Fehlerbehebung.

Ursprünglich von Stephan Diederich am Lehrstuhl für Informatik V der Universität Heidelberg initiiert, wird die *VRm Platform* von der *VRmagic GmbH* weiterentwickelt. Auf Basis der *VRm Platform* wurden bereits die Simulatoren *Eyesi Indirekt* und *NeuroSim* gebaut. Für eine vollständige Einführung in die *VRm Platform* sei auf [Bei12] verwiesen. In Abschnitt 5.2 werden die *VRm Libs* vorgestellt. Diese wurden sowohl für den Bau *VRm Platform* als auch für die Methoden der Arbeit benutzt, die in Kapitel 4 hergeleitet wurden.

VRm Libs

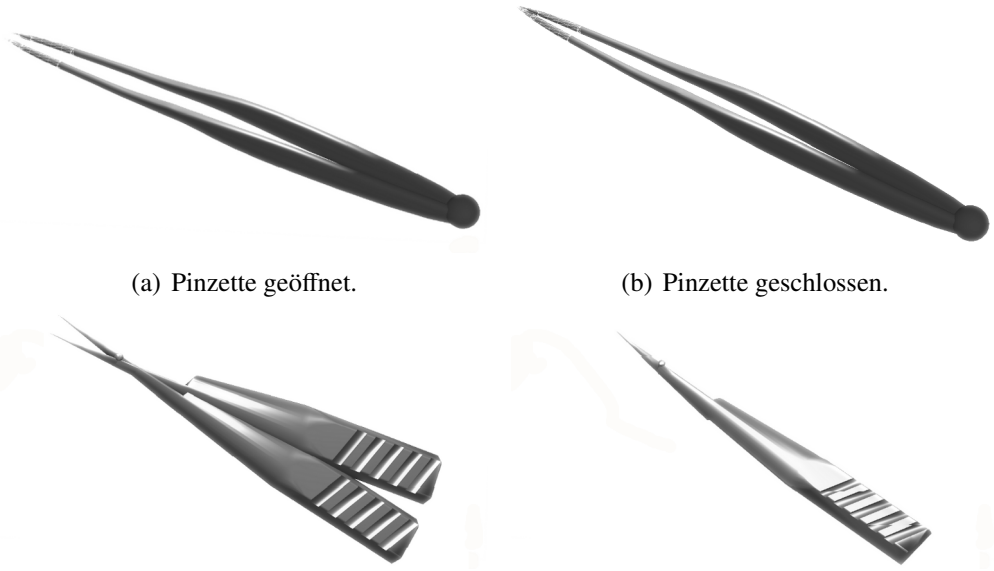
Die *VRm Libs* wurden am Lehrstuhl für Informatik V entwickelt und seit 2001 in Kooperation mit der *VRmagic GmbH* konstant erweitert. Ursprünglich entstand die Bibliothek für die Umsetzung des Trainingssimulators *Eyesi Surgical*. Die *VRm Libs* sind fast ausschließlich in C++ programmiert und verwenden für verschiedene Funktionen externe Bibliotheken. Beispiele dafür sind die *Qt* Bibliothek [Plc] für das Graphical User Interface (GUI), die *boost C++ LIBRARIES* [boo], die einen hohen Standard effizienter Algorithmen für viele Anwendungsfälle bereitstellen, sowie *OpenGL* [The] für das Rendern der simulierten Szenen.

Die *VRm Libs* bieten Algorithmen und Datenstrukturen für die verschiedenen Bausteine beim Bau eines Simulators. Beispiele dafür sind Algorithmen für die Anbindung und Einstellung externer Hardware. In der vorliegenden Arbeit kamen Algorithmen und Datenstrukturen für die Erstellung, Simulation und Speicherung deformierbarer Objekte zum Einsatz. Weiterhin wurden Kollisionsalgorithmen einfacher geometrischer Objekte verwendet.

Für einen besseren Überblick sei auf [Sch01] und [Wag03] verwiesen, in denen die *VRm Libs* das erste Mal vorgestellt wurden. Durch die Entwicklung weiterer Module, bzw. neuer Simulatoren, werden die *VRm Libs* stetig weiterentwickelt (vgl. [Jak09], [Web09], [Han09], [Bei12]).

5.2.1 *MicroSim*-Framework

Für eine Einführung in die grundlegenden Plug-ins der *VRm Platform*, z. B. für Grafikausgaben, die Interaktion mit den Eingabegeräten oder die Kommunikation zwischen den Modulen, wird auf [Bei12] verwiesen. Im Folgenden werden die für die vorliegende Arbeit relevanten Plug-ins vorgestellt:



(a) Pinzette geöffnet. (b) Pinzette geschlossen.

(c) Schere geöffnet. (d) Schere geschlossen.

Abbildung 5.6: *MicroSim* – Virtuelle Modelle der Instrumente

Plug-in für virtuelle Objekte

Alle virtuellen Objekte, die während der Simulation zum Einsatz kommen, werden in diesem Modul festgelegt. Das Objekt wird so benannt, dass es eindeutig identifizierbar ist und von einzelnen Trainingsmodulen dynamisch geladen werden kann. Es können die Position des Objektes im Szenegraphen und Eigenschaften für das Rendern festgelegt werden, z. B. können Dateien angegeben werden, aus denen dreidimensionale Modelle der Objekte oder Texturen geladen werden können.

Die virtuellen Nachbildungen der Instrumente, mit denen das simulierte Gewebe manipuliert wird, wurden von der *VRmagic GmbH* mit Cinema 4D modelliert und in den Simulator integriert. Das Tracking erkennt die Position der Marker und bestimmt daraufhin die Position und die Öffnungen der Instrumente. Abbildung 5.6 zeigt die virtuellen Modelle in geschlossenem und geöffnetem Zustand.

Translator Plug-in

Das Translator Plug-in abstrahiert Eingabegeräte von den gesteuerten Elementen. Dadurch können beide Ebenen unabhängig voneinander programmiert werden. So konnte z. B. das markerbasierte Tracking problemlos für weitere Instrumente verwendet werden (vgl. Abschnitt 5.1).

Trainingsmodul Plug-in

Das Trainingsmodul Plug-in bildet den Ausgangspunkt der Simulation. In diesem Plug-in werden alle weiteren Plug-ins für die Simulation geladen. Zu Beginn des Trainingsmoduls werden die verschiedenen virtuellen Objekte, die für die Trainingseinheit benötigt werden, aktiviert. Den Kern des Trainingsmoduls bildet die *process()*-Methode, die regelmäßig von der *VRm Platform* aufgerufen wird. Hier finden die Berechnungen für die Simulation der Objekte statt. Jedes Trainingsmodul kann in mehrere Level unterteilt werden. Wie bei Computerspielen wird der Schwierigkeitsgrad für das erfolgreiche Beenden der Trainingssimulation zunehmend schwieriger.

5.2.2 Trainingsmodule

Im Folgenden wird der Aufbau und die Funktionsweise der Trainingsmodule beschrieben, die mit Hilfe der in Kapitel 4 vorgestellten Methoden implementiert wurden. Als Grundlage für die Trainingsmodule dienen die in Kapitel 2 beschriebenen medizinischen Grundlagen.

Es wurden fünf verschiedene Trainingsmodule implementiert. Das erste Trainingsmodul hat keinen direkten medizinischen Hintergrund. In diesem kann die allgemeine Koordination trainiert werden. Für das Entfernen der Adventitia wurden *Trim Modul 1 + 2* implementiert. *Dilatation-Modul* beschreibt ein Modul für das Erlernen der Dilatation eines Blutgefäßes. Im *Diagonal-Modul* werden ungleich große Öffnungen der Blutgefäße durch diagonales Anschneiden angeglichen.

Die Simulationsschleifen aller Trainingsmodule ähneln sich. Zu Beginn des Simulationsschrittes wird die SSD gefüllt, um die Kollisionserkennung zu beschleunigen. Die Position, die aus den Bewegungen der realen Instrumente resultieren, werden auf die virtuellen Instrumente angewendet. Dabei wird die absolute Bewegung der Instrumente in mehreren Schritten iteriert. Die Kraft, die auf das Mesh übertragen wird, kann dadurch gestaffelt werden. Dadurch werden Kollisionen

großer Bewegungen registriert und die Simulation wird stabiler, da numerische Fehler verkleinert werden. Auf Basis der internen und externen Kräfte werden in der Simulationsschleife die Deformationskräfte berechnet und der Integrationsschritt durchgeführt. Der größte Unterschied zwischen den Trainingsmodulen sind die Kollisionserkennungs- und -behandlungsalgorithmen, die aufgrund der unterschiedlichen Instrumente und virtuellen Objekte, die in den Modulen eingesetzt werden, variieren. Listing 5.1 zeigt die Simulationsschleife in Pseudocode.

```
LOAD SIMULATION SCENE
BEGIN PROCESS
  BEGIN TIME STEP
    FILL SSD
    ITERATION LOOP
      SET ITERATED POSITIONS OF INSTRUMENTS
      CALCULATE DEFORMATION FORCES
      INTEGRATION STEP
      COLLISION DETECTION AND RESPONSE
    END ITERATION LOOP
  END TIME STEP
  UPDATE VISUAL SCENE
END PROCESS
```

Listing 5.1: Mesh-Mesh-Kollisionen

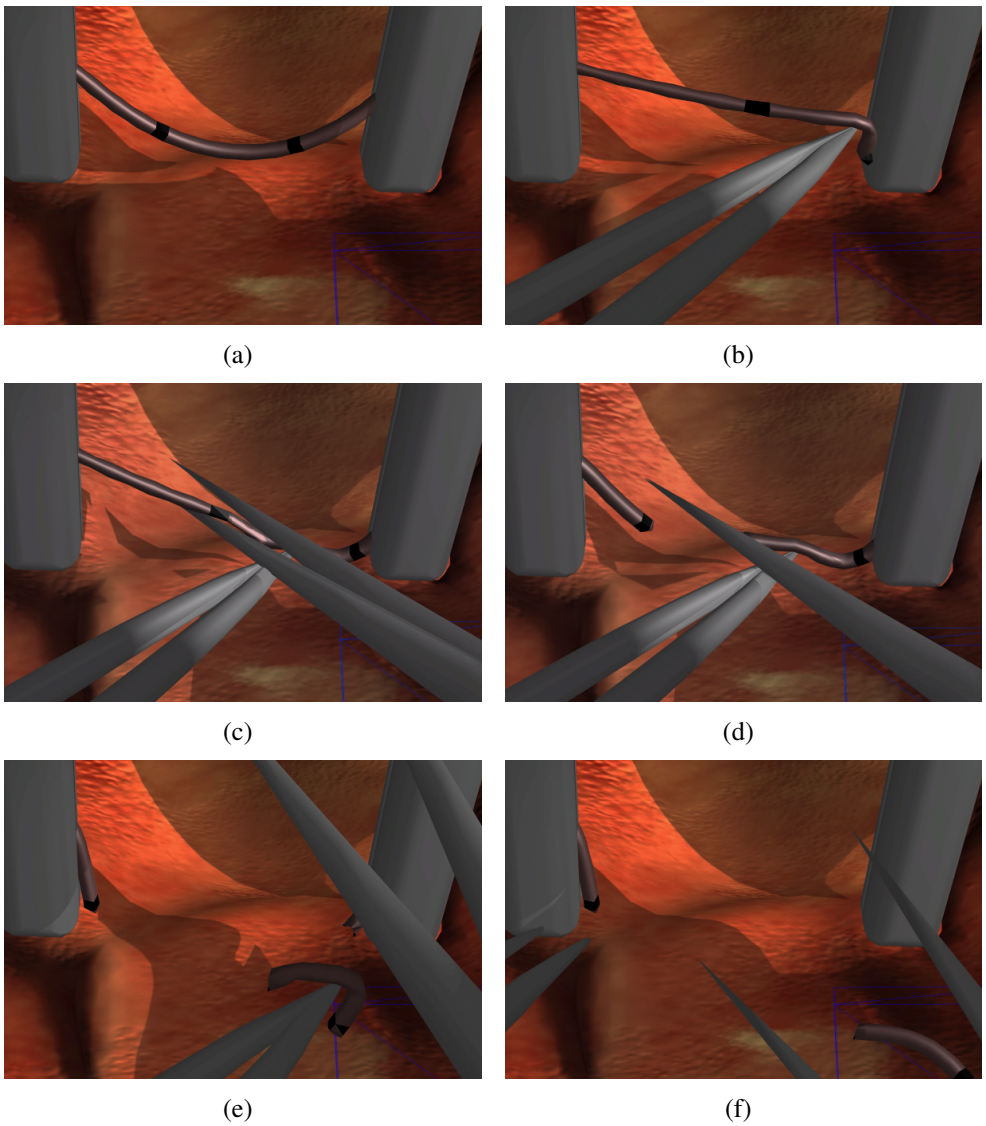
Allgemeine Koordination

In diesem Modul soll die allgemeine Koordination beim Arbeiten unter starken Vergrößerungen geschult werden. Der Benutzer macht sich mit dem Instrumentar vertraut. Dabei sollen abstrakte Aufgaben mit verschiedenen Instrumenten gelöst werden.

Als Beispiel wird ein Mesh zwischen zwei Klammern gespannt. Gravitationskräfte wirken auf das Mesh (Abbildung 5.7(a)). Dem Benutzer stehen Pinzette und Schere zur Verfügung. Die Pinzette wird mit der nicht dominierenden Hand geführt. Die andere Hand bewegt die Schere. Die Aufgabe besteht darin, das Mesh an den gekennzeichneten Stellen mit der Mikroschere zu schneiden.

Die Eigenschaften des Gewebes können beliebig eingestellt werden. Bei steifem Material kann das Tetraedernetz durch festes Zudrücken abgezwickt werden. Bei elastischem Material, welches der Eigenschaft von Bindegewebe näher kommt, funktioniert diese Methode nicht immer. In diesem Fall, muss die Schere zu Hilfe genommen werden und ein präziser Schnitt gesetzt werden. Die Pinzette kann in diesem Fall dazu benutzt werden, das Mesh zu justieren. Abbildung 5.7 illustriert

den Vorgang. Der Benutzer greift das Mesh mit der Pinzette und fixiert es (Abbildung 5.7(b)). Anschließend wird das Mesh an den gekennzeichneten Stellen geschnitten (Abbildung 5.7(c) und 5.7(d)). Gelingt das, wird das Mesh mit der Pinzette an die markierte Stelle transportiert (Abbildung 5.7(e) und 5.7(f)).



(a) (b) (c) (d) (e) (f)

Abbildung 5.7: *Basis-Modul* 1 – Allgemeine Koordination. Ein gespanntes Mesh wird vom Benutzer geschnitten, gegriffen und an die gekennzeichnete Stelle bewegt und fallengelassen.

Trimmen der Adventitia

Die folgenden Module dieses Abschnittes beschäftigen sich mit dem Beseitigen der Adventitia rund um das Ende des Blutgefäßes. Dabei werden zwei Möglichkeiten betrachtet.

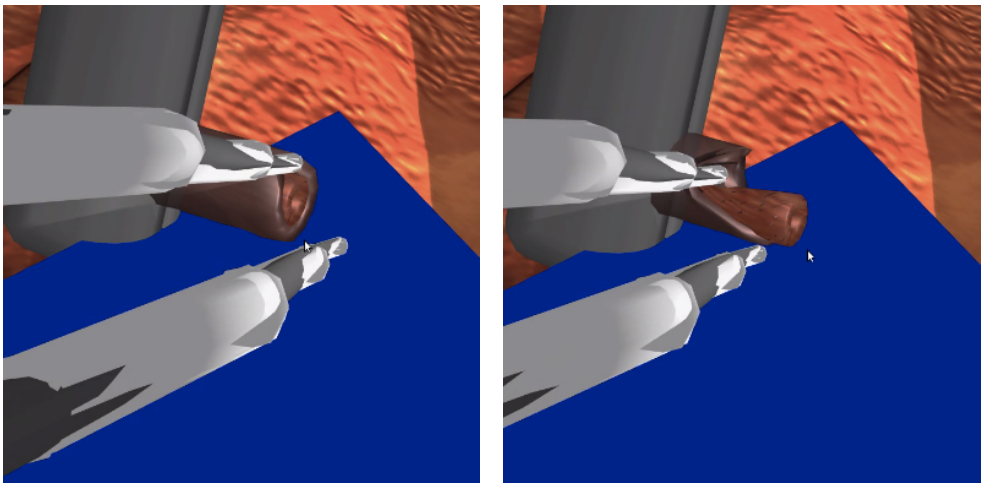


Abbildung 5.8: *Trimm-Modul* 1 – Säubern des Blutgefäßes. Die Adventitia wird mit einer Pinzette nach hinten weggeschoben.

(1) Die Adventitia, die lose auf der Media liegt oder leicht mit dieser verbunden ist, wird durch eine Pinzette beiseite geschoben. Der Benutzer hat die Möglichkeit, zwei Pinzetten zu verwenden. Sowohl Media als auch Adventitia sind dabei an einer Klammer fixiert. Abbildung 5.8 verdeutlicht den Vorgang. Die Adventitia wird dabei mit sehr kleinen Steifigkeitskonstanten modelliert. Das hat den Effekt, dass die Erhaltungskräfte verschwindend gering sind, sodass die Adventitia verformt bleibt.

(2) Die Adventitia wird über den Rand des Blutgefäßendes gezogen und anschließend mit einer Schere getrennt. Wurde nicht genug Adventitia getrimmt, kann der Vorgang wiederholt werden. Die Steifigkeitskonstanten der Tetraederkanten werden erhöht, die der Tetraeder verringert. Das hat den Effekt, dass das Gewebe äußerst elastisch ist und trotzdem hohe Rückstellkräfte bildet. Die Adventitia wurde mit Constraints, ähnlich der Kollisionsantwort, an die Media gekoppelt. Dadurch ist die Adventitia nicht zu lösen und muss abgeschnitten werden. Abbildung 5.9 zeigt den Vorgang. Diese kann mit Abbildung 2.6 des realen mikrochirurgischen Eingriffs verglichen werden.

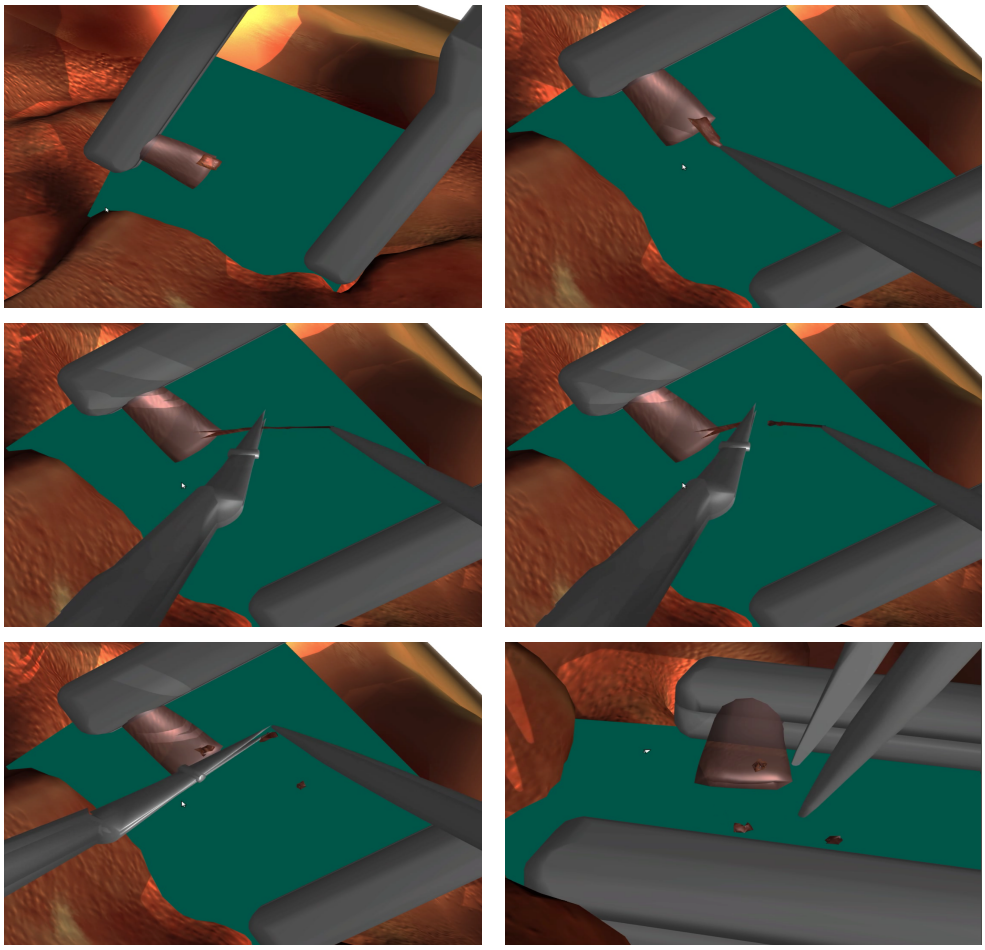
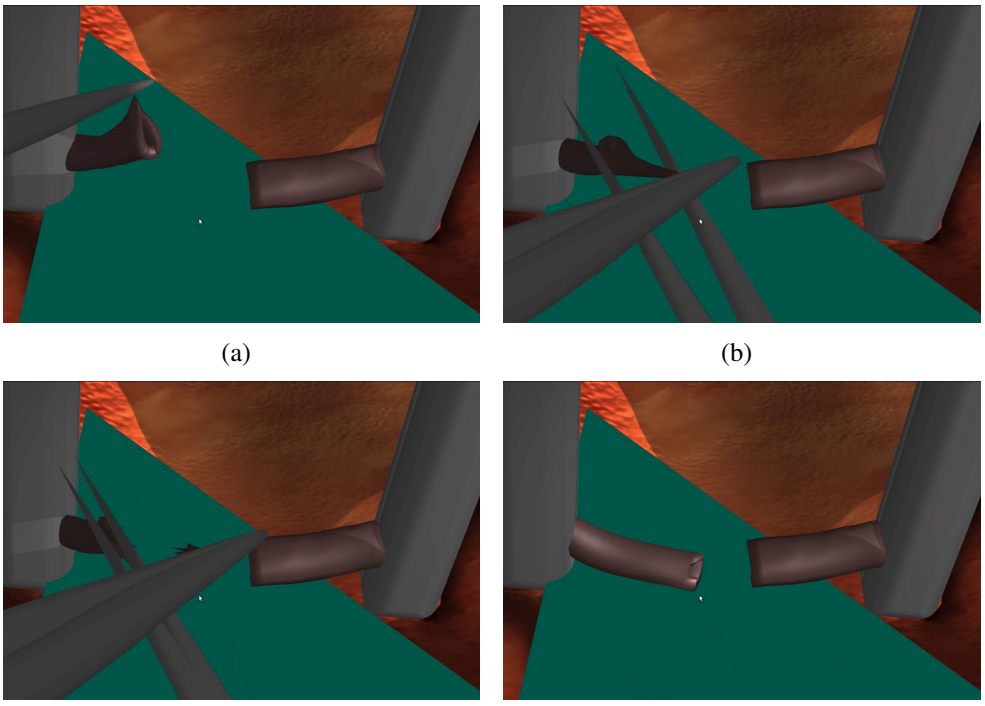


Abbildung 5.9: *Trimm-Modul* 2 – Überstehendes Bindegewebe wird von der Pinzette gegriffen, über das Blutgefäßende gezogen und vorsichtig abgeschnitten.

Diagonales Anschneiden

Sind die Öffnungen zweier Blutgefäßenden ungleich groß und müssen miteinander vernäht werden, kann das Blutgefäß mit dem geringeren Durchmesser diagonal angeschnitten werden. Es wurden zwei Blutgefäße mit unterschiedlichen Durchmessern erstellt und an den Klammern fixiert. Das kleinere Blutgefäß wird gegriffen und diagonal angeschnitten.



(a) (b)

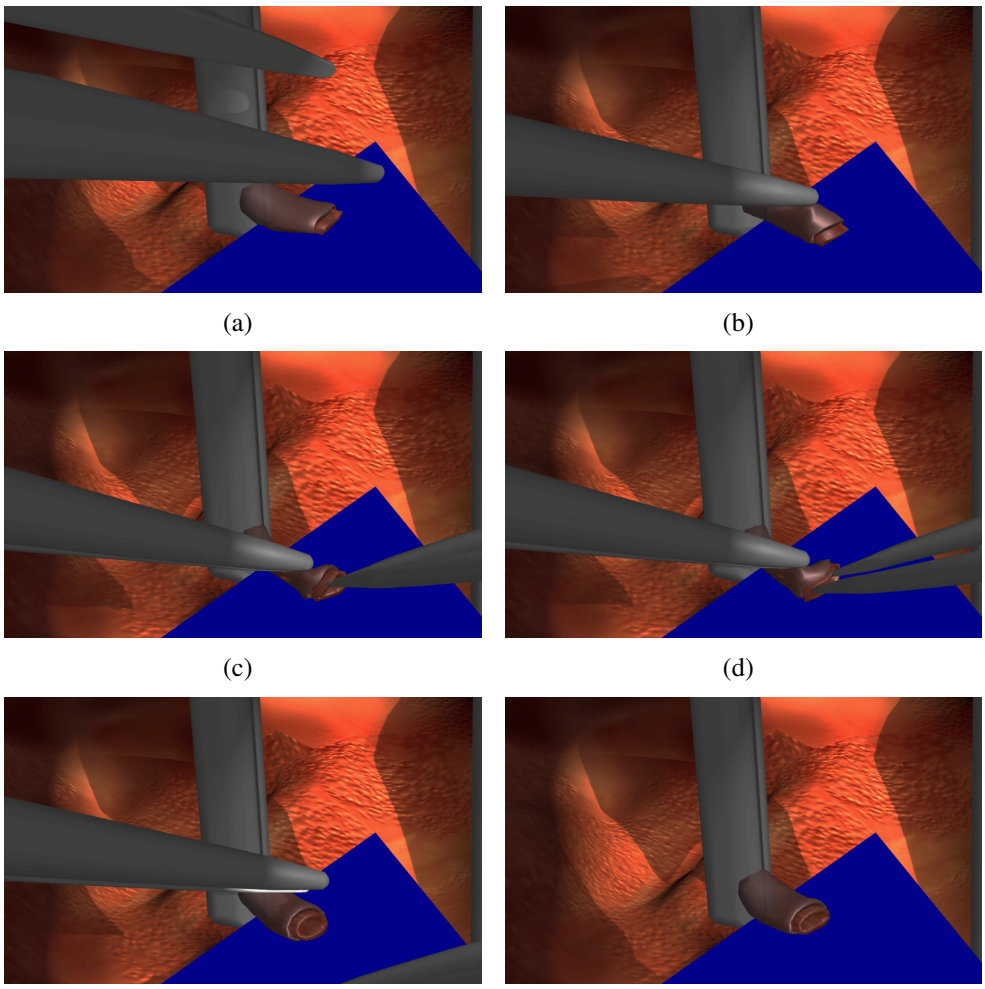
(c) (d)

Abbildung 5.10: *Diagonal-Modul* – Anpassung der Größe der Blutgefäßenden

Dilatation

Das folgende Modul wurde auf Basis der medizinischen Grundlagen, welche in Abschnitt 2.3 in Bezug auf die Dilatation von Blutgefäßen beschrieben wurde, konzipiert und implementiert. Die Media wurde erstellt und an der Klammer fixiert. Die Media ist von einem Mesh umgeben, welches die Adventitia darstellt (Abbildung 5.11(a)). An dieser wird das Blutgefäß gegriffen, sodass mit der dominanten Hand der Dilatator in die Öffnung des Blutgefäßes eingeführt werden kann (Abbildungen 5.11(b) und 5.11(c)). Anschließend wird der Dilatator leicht gespreizt ((Abbildung 5.11(d))). Durch die BCs werden die Tetraeder und Tetraederkanten an der Stelle gedehnt. Damit der Effekt des paralysierten Blutgefäßes

simuliert wird, werden die Steifigkeitskonstanten der gedehnten Primitive erhöht. Dadurch wird das Gewebe an der Stelle steifer und die Öffnung bleibt erhalten (Abbildungen 5.11(e) und 5.11(f)). Der Vorgang kann mit Abbildung 2.7(b) eines realen mikrochirurgischen Eingriffs verglichen werden.



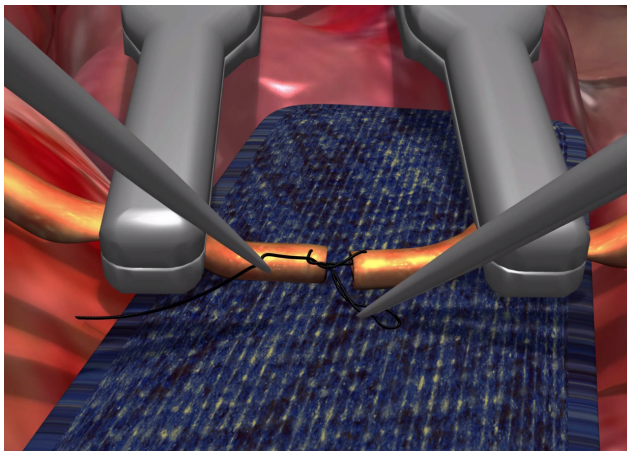
(a) (b) (c) (d) (e) (f)

Abbildung 5.11: Dilatationsmodul – Die Adventitia wird von der Pinzette gegriffen. Der Dilatator wird vorsichtig in das Blutgefäß eingeführt und das Blutgefäß wird gespreizt.

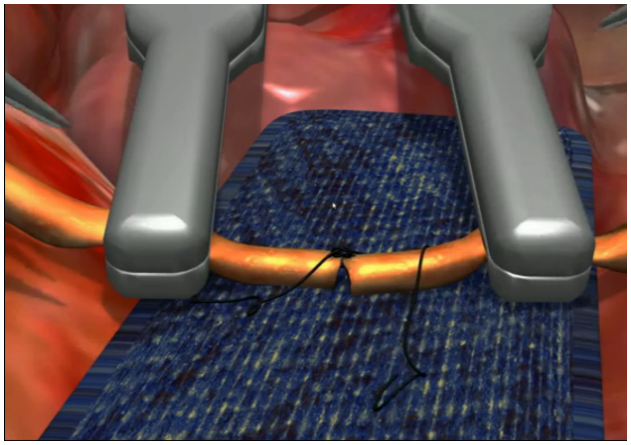
Nähen

Das in diesem Abschnitt vorgestellte Trainingsmodul zeigt aktuelle Ergebnisse aus der Dissertation von N. Hüsken [Hüs13], die sich mit der Simulation des Nahtmaterials auseinandersetzt. Das Modul wird an dieser Stelle erwähnt, da es die in

Abschnitt 4.3 beschriebenen Kollisionsalgorithmen verwendet. Diese werden für die Erkennung und Auflösung der Kollisionen zwischen den Blutgefäßen verwendet. Für mehr Informationen wird auf die Quelle [Hüs13] verwiesen.



(a) h



(b) h

Abbildung 5.12: *Anastomose-Modul* – Aktuelle Ergebnisse von N. Hüsken aus [Hüs13], in die die Kollisionsalgorithmen der vorliegenden Arbeit einfließen. Abbildungen mit freundlicher Genehmigung von N. Hüsken.

Der Trainingssimulator *MicroSim* wurde in "*MicroSim* – A microsurgical Trainingsimulator" [HSSB13] veröffentlicht.

KAPITEL

6

Mehr als die Vergangenheit interessiert mich die Zukunft, denn in ihr gedenke ich zu leben.

(Albert Einstein 1879 - 1955*)*

Schlussfolgerung und Ausblick

Der Einzug von VR-Simulatoren in die medizinische Ausbildung nimmt stetig zu. Zurückzuführen ist das vor allem auf den technischen Fortschritt der letzten Jahre, mit dem sich Umgebungen realisieren lassen, die sich kaum noch von der Realität unterscheiden. Die vorliegende Arbeit trägt zu der Entwicklung des auf VR basierenden Trainingssimulators *MicroSim* bei. *MicroSim* kann sowohl für das Training von fächerübergreifenden Fertigkeiten (z. B. motorischen) als auch für das Erlernen spezifischer mikrochirurgischer Fähigkeiten (z. B. das Vernähen von Mikroblutgefäßen) eingesetzt werden. Durch die Verwendung realer Operationsinstrumente kann der Benutzer mit der virtuellen Szene interagieren.

Ziel der Arbeit war die Entwicklung prototypischer Module für das mikrochirurgische Training mit *MicroSim*. Es wurden Fertigkeiten, die sich ein Chirurg während seiner praktischen Ausbildung aneignen muss, analysiert (Kapitel 2). Darauf basierend wurden fünf Module implementiert. Am *Basis-Modul* kann die Koordination unter starken Vergrößerungen und das Setzen präziser Schnitte mit einer Mikroschere trainiert werden. Dieses Modul zielt darauf ab, die ungewohnte Hand-Auge-Koordination unter starken Vergrößerungen zu trainieren. Um das Säubern des Blutgefäßes von der Adventitia zu trainieren, wurden zwei Module implementiert, *Trimm-Modul* 1 und *Trimm-Modul* 2. In Ersterem kann die Adventitia, die auf der Media aufliegt, vorsichtig nach hinten geschoben werden. In *Trimm-*

Modul 2 kann überflüssiges Gewebe mit der Pinzette über die Öffnung der Media gezogen und mit der Mikroschere abgeschnitten werden. Für die abschließende Behandlung der Blutgefäße wurden die Module *Dilatation-Modul* und *Diagonal-Modul* implementiert. Ersteres wird für das Spreizen eines Blutgefäßes verwendet. Letzteres beschäftigt sich mit dem Angleichen zwei ungleich großer Blutgefäßöffnungen. Nach bestem Wissen des Autors ist *MicroSim* der erste Simulator, an dem eine Dilatation oder das diagonale Anschneiden eines Blutgefäßes trainiert werden kann. Anhand der genannten Module können sämtliche Operationsvorgänge, die ein Mikrochirurg vor dem Vernähen der Blutgefäße eventuell durchführen muss, trainiert werden.

Es wurden zum einen Algorithmen der umfangreichen Softwarebibliothek der *VRmagic GmbH* kombiniert und erweitert, zum anderen neue Algorithmen entwickelt und in die Bibliothek implementiert. Den Schwerpunkt bildet die Implementierung der Interaktion des Benutzers mit der virtuellen Operationsszene.

Um volumetrische Effekte im Simulator abzubilden, wurden die virtuellen Nachbildungen der Blutgefäße und des Bindegewebes analytisch mit Tetraedernetzen modelliert. Diese werden von einem MFM simuliert. Als Integrationsverfahren wird das VVV eingesetzt. Durch das Verstellen einzelner Parameter des Simulationsmodells können verschiedene physikalische Effekte erzeugt werden. Für das *Dilatation-Modul* werden die Federlängen und die Volumina der Tetraeder beim Spreizen des Mesh dynamisch angepasst, wodurch plastische Verformungen simuliert werden, die für das Paralysieren der Blutgefäßöffnungen nach der Dilatation benötigt werden.

Die virtuellen Nachbildungen der Instrumente wurden von der *VRmagic GmbH* in Cinema 4D modelliert. Sie werden durch BCs approximiert, wodurch auf effiziente Kollisionserkennungsalgorithmen zurückgegriffen werden kann. In der Broad-Phase der Simulation wird eine SSD implementiert, um die Anzahl der Objekte, die möglicherweise kollidieren, für weitere Berechnungen zu minimieren. Die Kräfte für die Kollisionsbehandlung werden auf Basis des zugrundeliegenden Integrationsverfahrens berechnet. Im Gegensatz zu anderen Ansätzen, z. B. *Penalty*-Kräften, führt das eingesetzte Verfahren zu einer stabilen Auflösung der Kollisionen deformierbarer Objekte.

Eine Herausforderung für die Implementierung der Trainingsmodule stellt die Interaktion zwischen mikrochirurgischen Instrumenten (z. B. Pinzette) und dem Gewebe dar. Hierfür wurde ein Algorithmus entwickelt, der anhand des Drucks, der auf ein Objekt ausgeübt wird, und der Beschaffenheit des Instrumentes entscheidet, wie eine Kollision aufgelöst wird. Ist der Druck hoch genug, führt dies zu einer topologischen Änderung des Objektes: Steifes Gewebe bricht leichter, elastisches verformt sich und ist schwieriger zu trennen.

Die Grundlage des Algorithmus bildet eine Linie, die im Inneren der BCs als LONOR definiert wird. Dabei wurde die Gegebenheit ausgenutzt, dass Elemente in der virtuellen Welt nicht ohne Weiteres miteinander interagieren, sondern Kollisionen zwischen diesen erst erkannt und anschließend plausibel aufgelöst werden müssen. Die LONOR bestimmt somit indirekt den Druck, ab wann das Mesh getrennt wird. Dadurch wird das Gewebe wie in der Realität nicht ohne Widerstand, sondern erst nach Überwindung der Oberflächenspannung von der Pinzette oder der Mikroschere durchtrennt. Dabei kann die Position der LONOR so angepasst werden, dass sowohl stumpfe Objekte, wie Pinzetten, als auch scharfkantige Objekte, wie Scheren, simuliert werden können. Der Algorithmus verwendet einfache geometrische Elemente, wodurch auf schnelle Kollisionserkennungsalgorithmen zurückgegriffen werden kann. Dadurch wird die Schnelligkeit des Algorithmus nur durch die verwendete Tetraederanzahl bestimmt.

Der Algorithmus basiert nicht auf exakten physikalischen Parametern, sondern nutzt die vorgegebenen Beschränkungen der VR gezielt aus. Aufgrund dieser Vorgaben müssen einzelne Parameter, wie die Position der LONOR, für die Instrumente, die in der Simulation verwendet werden, der Realität iterativ angenähert werden. Der Algorithmus kann dennoch für diverse Instrumente eingesetzt werden, da es sich erstens bei Bindegewebe um sehr heterogenes Material handelt, dessen Verformung nicht exakt vorherzusehen ist; zweitens kommt es bei der Echtzeitsimulation auf das realistische Feedback an den Benutzer an und nicht auf die exakte Berechnung der Modifikationen.

Es wurde ein *Remeshing*-Algorithmus implementiert, der die Trennung des Mesh ermöglicht. Algorithmen der Fachliteratur wurden für das verwendete Simulationsmodell angepasst. Dafür wurde die Unterteilung der Tetraeder erarbeitet und der Arbeit als Anhang beigefügt. Fortan können topologische Änderungen eines Tetraedernetzes modelliert werden, indem die Tetraeder, die an der Riss- bzw. Schnittkante liegen, unterteilt werden. Die Unterteilung der einzelnen Primitive des Mesh wird auf Basis der Startpositionen der einzelnen Knoten berechnet. Dadurch kann auch gedehntes Material realistisch geschnitten und das anschließende Zurückfedern des Gewebes simuliert werden.

Der Fokus wurde auf echtzeitfähige Algorithmen gerichtet. Da in allen implementierten Modulen die Echtzeitanforderungen auf der verwendeten Hardware eingehalten werden konnten, wurde keiner der Algorithmen parallelisiert. Da die Interaktion mit der Szene während dieser Arbeit im Vordergrund stand, wurde kein sehr komplexer Aufbau für die einzelnen Simulationsmodule gewählt. Sollten komplexere Module höhere Anforderungen an die Rechenzeit stellen, können verschiedene Algorithmen parallelisiert werden. Beispiele hierfür sind das Einfügen der Primitiven in die SSD und die Kollisionserkennung zwischen Knoten und

Tetraedern bzw. zwischen Schnittgeräten und Tetraedern. Diese Algorithmen sind die, die bei einer hohen Anzahl an Kollisionen die meiste Rechenzeit benötigen.

Die Trainingsmodule wurden auf Basis der Erkenntnis aus mikrochirurgischen Lernvideos, der Fachliteratur, der Observation realer Eingriffe und den bisherigen Erfahrungen, die am Lehrstuhl mit dem Bau anderer chirurgischer Simulatoren gesammelt wurden, konzipiert und implementiert. Der nächste Schritt besteht darin, gemeinsam mit einem Chirurgen den trainingsrelevanten Inhalt der Module zu validieren und das Verhalten der simulierten Objekte durch iteratives Anpassen der Parameter der Realität weiter anzunähern. Des Weiteren müssen im Austausch mit Fachärzten Kriterien erforscht werden, um die Leistung der *MicroSim*-Nutzer objektiv bewerten zu können. Diese Kriterien könnten anschließend in das *MicroSim*-Bewertungssystem integriert werden.

Weiterhin sollen die Hintergrundszene des Simulators sowie die Modelle der Instrumente mit Hilfe von Grafikern realistischer gestaltet werden. Auch die deformierbaren anatomischen Objekte werden durch die Verwendung geeigneter Texturen einen lebendigeren Charakter erhalten. Abbildung 5.12 zeigt, wie die Echtheit der Szene verbessert werden kann, wenn professionelle Designer den Simulationshintergrund modellieren.

ANHANG

Mathematische Grundlagen der Kollisionserkennung

Die hier beschriebenen und weitere mathematische Grundlagen in Bezug auf die Kollsionserkennung finden sich in [Eri05].

A.1 Baryzentrische Koordinaten

Baryzentrische Koordinaten werden verwendet, um die Position eines Punktes im n-dimensionalen Raum in Abhängigkeit eines n-Simplex zu beschreiben. Im Folgenden wird die Herleitung beispielhaft an einer Dreiecksfläche (2-Simplex) gezeigt. Gegeben sei ein Dreieck D mit den Eckpunkten $\vec{A}$, $\vec{B}$, $\vec{C}$. Jeder Punkt $\vec{P}$, der innerhalb von D liegt, kann durch die Parameter s und t wie folgt beschrieben werden:

$$\vec{P} = \vec{A} + s(\vec{B} - \vec{A}) + t(\vec{C} - \vec{A}) \quad \text{(A.1)}$$

$$\vec{P} = (1 - s - t)\,\vec{A} + s\,\vec{B} + t\,\vec{C} \quad \text{(A.2)}$$

Sei nun $1 - s - t = r$, dann beschreibt das Tripel (r, s, t) die baryzentrischen Koordinaten des Punktes $\vec{P}$. Die baryzentrischen Koordinaten der Punkte $\vec{A}, \vec{B}, \vec{C}$ sind dazu entsprechend $(1, 0, 0)$, $(0, 1, 0)$ und $(0, 0, 1)$. Daraus folgt, dass der Punkt $\vec{P}$ nur dann innerhalb des Dreiecks liegt, wenn $0 \leq r, s, t \leq 1$. Für r, s, t gilt immer: $r + s + t = 1$.

Für $\vec{v_0}=\vec{B} - \vec{A}$, $\vec{v_1}=\vec{C} - \vec{A}$ und $\vec{v_2}=\vec{P} - \vec{A}$ folgt aus der Gleichung A.1:

$$s\,\vec{v_0} + t\,\vec{v_1}=\vec{v_2}\,. \quad \text{(A.3)}$$

Darauf wird jeweils die Skalarmultiplikation mit $\vec{v_0}$ und $\vec{v_1}$ angewendet, woraus ein 2×2 lineares Gleichungssystem entsteht:

$$s(\vec{v_0} \cdot \vec{v_0}) + t(\vec{v_1} \cdot \vec{v_0}) = \vec{v_2} \cdot \vec{v_0} \quad \text{(A.4)}$$

$$s(\vec{v_0} \cdot \vec{v_1}) + t(\vec{v_1} \cdot \vec{v_1}) = \vec{v_2} \cdot \vec{v_1} \quad \text{(A.5)}$$

Die Lösung des linearen Gleichungssystems ergibt die baryzentrischen Koordinaten des Punktes $\vec{P}$:

$$r = 1 - s - t \quad \text{(A.6)}$$

$$s = \frac{(\vec{v_1} \cdot \vec{v_1}) \cdot (\vec{v_2} \cdot \vec{v_0}) - (\vec{v_0} \cdot \vec{v_1}) \cdot (\vec{v_2} \cdot \vec{v_1})}{(\vec{v_0} \cdot \vec{v_0}) \cdot (\vec{v_1} \cdot \vec{v_1}) - (\vec{v_0} \cdot \vec{v_1}) \cdot (\vec{v_0} \cdot \vec{v_1})} \quad \text{(A.7)}$$

$$t = \frac{(\vec{v_0} \cdot \vec{v_0}) \cdot (\vec{v_2} \cdot \vec{v_1}) - (\vec{v_0} \cdot \vec{v_1}) \cdot (\vec{v_2} \cdot \vec{v_0})}{(\vec{v_0} \cdot \vec{v_0}) \cdot (\vec{v_1} \cdot \vec{v_1}) - (\vec{v_0} \cdot \vec{v_1}) \cdot (\vec{v_0} \cdot \vec{v_1})} \quad \text{(A.8)}$$

A.2 Kollision zwischen Linie und Dreieck

Die Kollision wird nach [Eri05] eingeführt. Im Folgenden sei D das Dreieck bestehend aus den Punkten $\vec{A}$, $\vec{B}$ und $\vec{C}$, wobei sich jeder Punkt auf der Fläche des Dreiecks mit $\vec{T}\ (r,s,t) = r\ \vec{A} + s\ \vec{B} + t\ \vec{C}$ beschreiben lässt. Die baryzentrischen Koordinaten des Punktes sind damit (r,s,t). Analog lässt sich $\vec{T}$, wie in Abschnitt A.1 gezeigt wurde, auch als $\vec{T}\ (s,t) = \vec{A} + s(\vec{B} - \vec{A}) + t(\vec{C} - \vec{A})$, wobei $r = 1 - s - t$, schreiben. $\overline{PQ}$ sei die Linie zwischen den Punkten $\vec{P}$ und $\vec{Q}$. Der Schnittpunkt lässt sich mit $\vec{A} + s(\vec{B} - \vec{A}) + t(\vec{C} - \vec{A}) = \vec{P} + l(\vec{Q} - \vec{P})$ mit $0 \leq l \leq 1$ beschreiben. Daraus folgt:

$$(\vec{P} - \vec{Q})l + (\vec{B} - \vec{A})s + (\vec{C} - \vec{A})t = \vec{P} - \vec{A} \tag{A.9}$$

Dieses 3×3 lineares Gleichungssystem lässt sich als

$$\left[(\vec{P} - \vec{Q})(\vec{B} - \vec{A})(\vec{C} - \vec{A})\right] \begin{bmatrix} l \\ s \\ t \end{bmatrix} = \left[(\vec{P} - \vec{A})\right] \tag{A.10}$$

schreiben. Aus der Cramerschen Regel folgt für s, t, l:

$$l = \frac{det\left[(\vec{P} - \vec{A})(\vec{B} - \vec{A})(\vec{C} - \vec{A})\right]}{det\left[(\vec{P} - \vec{Q})(\vec{B} - \vec{A})(\vec{C} - \vec{A})\right]} \tag{A.11}$$

$$s = \frac{det\left[(\vec{P} - \vec{Q})(\vec{P} - \vec{A})(\vec{C} - \vec{A})\right]}{det\left[(\vec{P} - \vec{Q})(\vec{B} - \vec{A})(\vec{C} - \vec{A})\right]} \tag{A.12}$$

$$t = \frac{det\left[(\vec{P} - \vec{Q})(\vec{B} - \vec{A})(\vec{P} - \vec{A})\right]}{det\left[(\vec{P} - \vec{Q})(\vec{B} - \vec{A})(\vec{C} - \vec{A})\right]} \tag{A.13}$$

Aus $det\left[\vec{x}\vec{y}\vec{z}\right] = \vec{x} \cdot (\vec{y} \times \vec{z})$ lassen sich s, t, l wie folgt vereinfachen:

$$l = (\vec{P} - \vec{A}) \cdot \frac{\vec{n}}{d} \tag{A.14}$$

$$s = (\vec{C} - \vec{A}) \cdot \frac{\vec{e}}{d} \tag{A.15}$$

$$t = -(\vec{B} - \vec{A}) \cdot \frac{\vec{e}}{d} \tag{A.16}$$

wobei:

$$\vec{n} = (\vec{B} - \vec{A}) \times (\vec{C} - \vec{A}) \tag{A.17}$$

$$d = (\vec{P} - \vec{Q}) \cdot \vec{n} \tag{A.18}$$

$$\vec{e} = (\vec{P} - \vec{Q}) \times (\vec{P} - \vec{A}) \tag{A.19}$$

$\vec{n}$ ist hierbei die Normale des Dreiecks. Ist $d < 0$, schneidet die Linie das Dreieck nicht. Falls $d = 0$, verläuft die Linie parallel zur Fläche des Dreiecks. Ist $d > 0$, können die baryzentrische Koordinaten des Schnittpunktes aus den Gleichungen A.15 und A.16 berechnet werden. r folgt aus s und t.

A.3 Punkt Tetraeder

Es gibt mehrere verschiedene Wege, um zu testen, ob ein Punkt $\vec{P}$ innerhalb oder ausserhalb eines Tetraeders T liegt. Die Normalen $\vec{n}_{ABC}$, $\vec{n}_{ACD}$, $\vec{n}_{ADB}$, $\vec{n}_{BCD}$ der vier Oberflächendreiecke, die vom Tetraeder "wegzeigen", seien bekannt. Mit Hilfe des Skalarproduktes lässt sich effektiv feststellen, ob ein Punkt innerhalb eines Tetraeders liegt. Für die folgenden Gleichungen gilt:

$$\vec{n}_{ABC} \cdot \vec{AP} = t_{ABC} \tag{A.20}$$

$$\vec{n}_{ACD} \cdot \vec{AP} = t_{ACD} \tag{A.21}$$

$$\vec{n}_{ADB} \cdot \vec{AP} = t_{ADB} \tag{A.22}$$

$$\vec{n}_{BCD} \cdot \vec{BP} = t_{BCD} \tag{A.23}$$

Ist t_{ABC}, t_{ACD}, t_{ADB} oder $t_{BCD} < 0$, liegt $\vec{P}$ nicht im Tetraeder. Andernfalls liegt $\vec{P}$ innerhalb des Tetraeders.

ANHANG

B

Schnittmuster

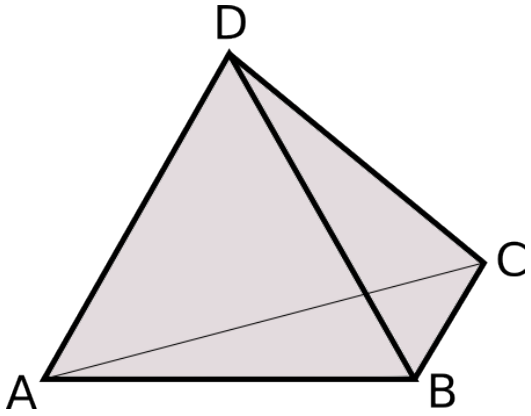


Abbildung B.1: Tetraeder

Im Folgenden werden die konkreten Schnittmuster aufgelistet. Es wird von einem Tetraeder T mit den Endknoten A, B, C und D ausgegangen (Abbildung B.1). Bei der Knotenbenennung repräsentieren Knoten mit zwei Buchstaben die auf einer getrennten Kante generierten Knoten, wobei der erste Buchstabe den am nächsten gelegenen Endknoten der Kante repräsentiert.

B.1 Muster A – Aufspalten an einer Kante

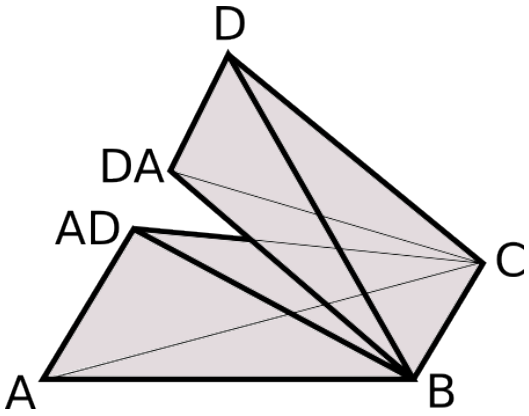


Abbildung B.2: Muster A

Knoten erstellen:

AD, DA

Kanten löschen:

(A, D)

Kanten erstellen:

1. (A, AD) 2. (DA, D) 3. (B, AD)
4. (C, AD) 5. (B, DA) 6. (C, DA)

Oberflächendreiecke löschen:

(A, B, D), (C, A, D)

Oberflächendreiecke erstellen:

1. (C, A, AD) 2. (C, DA, D) 3. (A, B, AD)
4. (DA, B, D) 5. (C, AD, B) 6. (B, DA, C)

Tetraeder löschen:

(A, B, C, D)

Tetraeder erstellen:

1. (A, B, C, AD) 2. (DA, B, C, D)

B.2 Muster B – Aufspalten an zwei Kanten

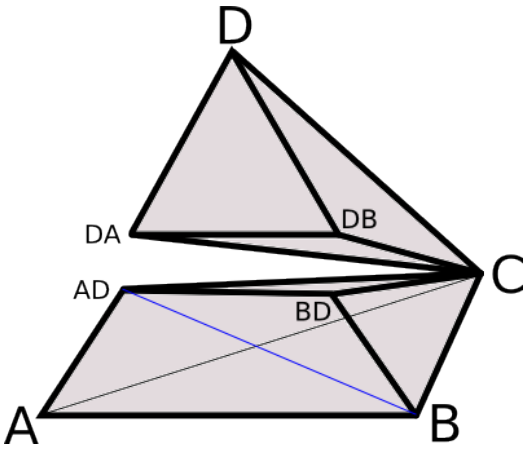


Abbildung B.3: Muster B

Knoten erstellen:

AD, DA, BD, DB

Kanten löschen:

(A, D), (B, D)

Kanten erstellen:

1. (A, AD)	2. (DA, D)	3. (B, BD)
4. (DB, D)	5. (C, AD)	6. (C, DA)
7. (C, BD)	8. (C, DB)	9. (AD, BD)
10. (DA, DB)	11. (B, AD) oder (A, BD)	

Oberflächendreiecke löschen:

(A, B, D), (B, C, D), (C, A, D)

Oberflächendreiecke erstellen:

1. (DA, DB, D)	2. (DB, C, D)	3. (C, DA, D)
4. (C, DB, DA)	5. (AD, BD, C)	6. (C, A, AD)
7. (B, C, BD)		
Kante (B, AD):	8. (A, B, AD)	9. (B, BD, AD)
Kante (A, BD):	8. (A, B, BD)	9. (A, BD, AD)

Tetraeder löschen:

(A, B, C, D)

Tetraeder erstellen:

1. (AD, BD, C, D)		
Kante (B, AD):	2. (AD, B, A, C)	3. (AD, B, BD, C)
Kante (A, BD):	2. (A, B, C, BD)	3. (A, BD, C, AD)

B.3 Muster C – Aufspalten an drei Kanten

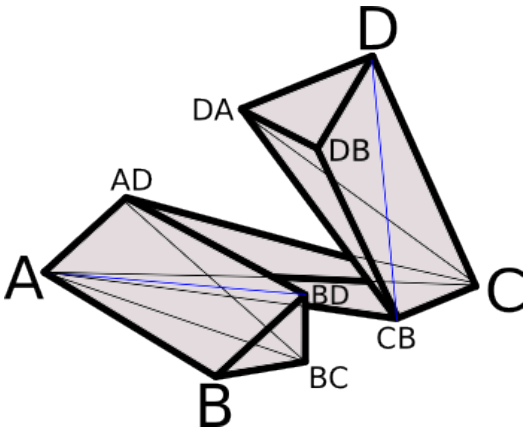


Abbildung B.4: Muster C

Knoten erstellen:

AD, DA, BD, DB, BC, CB

Kanten löschen:

(A, D), (B, D), (B, C)

Kanten erstellen:

1. (A, AD)
2. (DA, D)
3. (B, BD)
4. (DB, D)
5. (B, BC)
6. (CB, C)
7. (AD, BD)
8. (DA, DB)
9. (BD, BC)
10. (DB, CB)
11. (AD, BC)
12. (DA, CB)
13. (A, BC)
14. (A, CB)
15. (C, AD)
16. (C, DA)
17. (A, BD) oder (B, AD)
18. (D, CB) oder (C, DB)

Oberflächendreiecke löschen:

(A, B, D), (B, C, D), (C, A, D), (A, B, C)

Oberflächendreiecke erstellen:

1. (A, BC, AD)	2. (AD, BC, BD)	3. (B, BC, BD)
4. (A, B, BC)	5. (C, A, AD)	6. (A, AD, CB)
7. (AD, CB, C)	8. (A, C, CB)	9. (C, DA, D)
10. (D, DA, DB)	11. (C, DA, CB)	12. (DA, DB, D)
Kante (A, BD)	13. (A, B, BD)	14. (A, BD, AD)
Kante (B, AD)	13. (A, B, AD)	14. (B, BD, AD)
Kante (D, CB)	15. (DB, CB, D)	16. (CB, C, D)
Kante (C, DB)	15. (DB, CB, C)	16. (DB, C, D)

Tetraeder löschen:

(A, B, C, D)

Tetraeder erstellen:

1. (A, CB, C, DA)		
Kante (A, BD)	2. (A, B, BC, BD)	3. (A, BD, BC, AD)
Kante (B, AD)	2. (A, B, BC, AD)	3. (B, AD, BC, BD)
Kante (D, CB)	4. (DA, D, CB, DB)	5. (DA, D, CB, C)
Kante (C, DB)	4. (C, DB, D, DA)	5. (C, CB, DA, DB)

B.4 Muster D – Teilen an drei Kanten

Knoten erstellen:

AD, DA, BD, DB, CD, DC

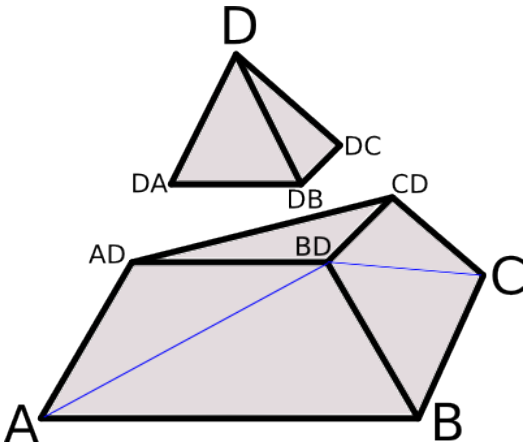


Abbildung B.5: Muster D

Kanten löschen:

(A, D), (B, D), (C, D)

Kanten erstellen:

1. (A, AD)	2. (DA, D)	3. (B, BD)
4. (DB, D)	5. (C, CD)	6. (DC, D)
7. (AD, BD)	8. (BD, CD)	9.(CD, AD)
10. (DA, DB)	11. (DB, DC)	12. (DC, DA)

13. (A, BD) oder (B, AD)
14. (B, CD) oder (C, BD)
15. (C, AD) oder (A, CD)

Oberflächendreiecke löschen:

(A, B, D), (B, C, D), (C, A, D)

Oberflächendreiecke erstellen:

1. (DA, DB, D)	2. (DB, DC, D)	3. (DC, DA, D)
4. (DA, DB, DC)	5. (AD, BD, CD)	
Kante (A, BD):	6. (A, B, BD)	7. (A, BD, AD)
Kante (B, AD):	6. A, B, AD)	7. (B, BD, AD)
Kante (BD, C):	8. (BD, C, B)	9. (BD, C, CD)
Kante (CD, B):	8. (CD, B, C)	9. (CD, B, BD)
Kante (A, CD):	10. (A, CD, AD)	11. (BD, C, CD)
Kante (C, AD):	10. (C, AD, A)	11. (C, AD, CD)

Tetraeder löschen:

(A, B, C, D)

Tetraeder erstellen:

1. (DA, DB, DC, D)	
Kante (A, BD) und (C, BD):	2.(A, B, C, BD)
Kante (B, AD) und (B, CD):	2. (B, AD, BD, CD)
Kante (A, BD) und (B, CD) und (A, CD):	2. (B, CD, A, BD)
Kante (B, AD) und (C, BD) und (C, AD):	2. (B, AD, BD, C)
Kante (B, CD) und (A, CD):	3.(A, B, C, CD)
Kante (C, AD) und (C, BD):	3.(C, AD, BD, CD)
Kante (B, CD) und (C, AD) und (B, AD):	3.(C, CD, B, AD)
Kante (C, BD) und (A, CD) und (A, BD):	3.(C, CD, BD, A)
Kante (C, AD) und (B, AD):	4.(A, B, C, AD)
Kante (A, BD) und (A, CD):	4.(A, AD, BD, CD)
Kante (C, AD) und (A, BD) und (C, BD):	4.(A, AD, BD, C)
Kante (A, CD) und (B, AD) und (B, CD):	4.(A, AD, B, CD)

Falls die Kanten (A, BD), (B, CD) und (C, AD) oder (A, CD), (C, BD) und (B, AD) festgelegt wurden, können die Tetraeder aus den Punkten 2, 3 und 4 nicht erstellt werden. In diesem Fall wird im Mittelpunkt des Tetraeders ein weiterer Knoten $INSIDE$ erstellt und jeweils eine Kante von A, B, C, AD, BD, CD zu $INSIDE$. Die Knoten jedes Oberflächendreiecks der unteren Teilhälfte des Tetraeders bilden mit $INSIDE$ jeweils einen neuen Tetraeder. Insgesamt werden fünf Tetraeder erstellt.

B.5 Muster E – Teilen an vier Kanten

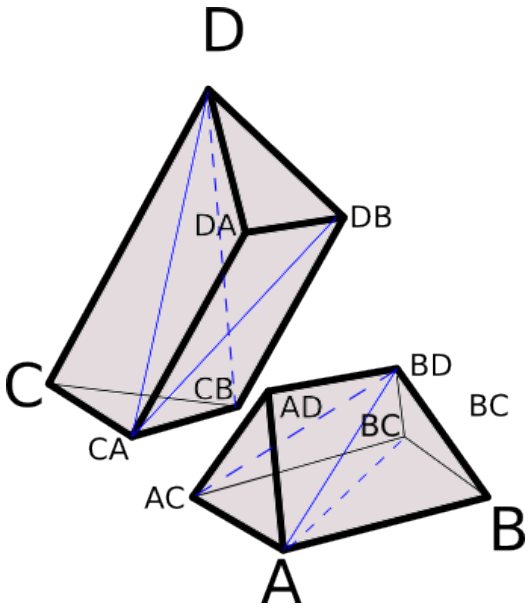


Abbildung B.6: Muster E

Knoten erstellen:

AD, DA, BD, DB, AC, CA, BC, CB

Kanten löschen:

(A, D), (B, D), (A, C), (B, C)

Kanten erstellen:

1. (A, AD)	2. (DA, D)	3. (B, BD)
4. (DB, D)	5. (B, BC)	6.(CB, C)
7. (A, AC)	8. (CA, C)	9. (AD, BD)
10. (BD, BC)	11. (BC, AC)	12. (AC, AD)
13. (DA, DB)	14. (DB, CB)	15. (CB, CA)
16. (CA, DA)	17. (A, BD) oder (B, AD)	18. (A, BC) oder (B, AC)
19. (AC, BD) oder (BC, AD)	20. (CA, D) oder (DA, C)	21. (CA, DB) oder (CB, DA)
22. (CB, D) oder (C, DB)		

Oberflächendreiecke löschen:

(A, B, D), (B, C, D), (C, A, D), (A, B, C)

Oberflächendreiecke erstellen:

1. (DA, DB, D)	2. (CA, CB, C)	3. (A, AD, AC)
4. (B, BD, BC)		
Kante (A, BD)	5. (A, B, BD)	6. (A, BD, AD)
Kante (B, AD)	5. (A, B, AD)	6. (B, BD, AD)
Kante (A, BC)	7. (A, B, BC)	8. (A, BC, AC)
Kante (B, AC)	7. (A, B, AC)	8. (B, BC, AC)
Kante (AC, BD)	9. (AC, BD, AD)	10. (AC, BC, BD)
Kante (AD, BC)	9. (AC, BC, AD)	10. (AD, BC, BD)
Kante (CA, DB)	11. (CA, DB, CB)	12. (CA, DB, DA)
Kante (CB, DA)	11. (CB, DA, DB)	12. (CB, DA, CA)
Kante (CA, D)	13. (CA, D, C)	14. (CA, D, DA)
Kante (C, DA)	13. (C, DA, CA)	14. (C, DA, D)
Kante (CB, D)	15. (CB, D, DB)	16. (CB, D, C)
Kante (C, DB)	15. (C, DB, D)	16. (C, DB, CB)

Tetraeder löschen:

(A, B, C, D)

Tetraeder erstellen:

Kante (A, BD), (A, BC) und (AC, BD):
1. (A, B, BC, BD) 2. (A, AC, AD, BD) 3. (AC, BC, BD, A)
Kante (B, AD), (B, AC) und (AC, BD):
1. (A, B, AC, AD) 2. (B, BD, AD, AC) 3. (B, BC, BD, AC)
Kante (A, BD), (B, AC) und (AC, BD):
1. (A, AC, AD, BD) 2. (A, B, BD, AC) 3. (AC, BC, B, BD)
Kante (B, AD), (A, BC) und (BC, AD):
1. (A, B, AD, BC) 2. (B, BC, BD, AD) 3. (AC, AD, BC, A)
Kante (CA, D), (CB, D) und (CA, DB):
4. (C, CA, CB, D) 5. (CA, DB, DA, D) 6. (CB, DB, D, CA)
Kante (C, DA), (C, DB) und (CA, DB):
4. (C, DA, CA, DB) 5. (C, DA, D, DB) 6. (C, CA, CB, DB)
Kante (AC, D), (C, BD) und (CA, DB):
4. (CA, DA, D, DB) 5. (CA, DB, C, D) 6. (C, CB, CA, DB)
Kante (AD, BC), (A, BC) und (CB, DA):
4. (CA, CB, C, DA) 5. (C, DA, D, CB) 6. (DB, CB, DA, D)

Abkürzungsverzeichnis

AABB Axis Aligned Bounding Box

AMA Australian Medical Association

BC Bounding Capsule

DEM Deformation-Energy-Minimization

DGG Private Akademie der Deutschen Gesellschaft für Gefäßchirurgie und Gefäßmedizin

DGL Differentialgleichung

FEM Finite-Elemente-Methode

FPGA Field Programmable Gate Array

GUI Graphical User Interface

MFM Masse-Feder-Modell

MMVR Medicine Meets Virutal Reality

LONOR Line Of No Return

SLE Simulating Learning Enironment

SSD Spatial Subdivision

ViPA Virtuelle Patienten Analyse

VR Virtuelle Realität

VVV Velocity-Verlet-Verfahren

Abbildungsverzeichnis

Literaturverzeichnis

[ACF+07] Allard, Jérémie, Stéphane Cotin, François Faure, Pierre-Jean Bensoussan, François Poyer, Christian Duriez, Hervé Delingette und Laurent Grisoni: *SOFA an Open Source Framework for Medical Simulation*. In: *Medicine Meets Virtual Reality (MMVR'15)*, Long Beach, USA, 2007.

[ACSYD05] Alliez, Pierre, David Cohen-Steiner, Mariette Yvinec und Mathieu Desbrun: *Variational tetrahedral meshing*. In: *Acm Transactions on Graphics*, Seiten 617–625. INRIA, Vandoeuvre Les Nancy, France, 2005.

[AMHH08] Akenine-Möller, Tomas, Eric Haines und Natty Hoffman: *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[AR08] Acland, Robert und Sabapathy S Raja: *Acland's Practice Manual for Microvascular Surgery*. The Indian Society for Surgery of the Hand (ISSH), third Auflage, 2008.

[BB94] Bullinger, Hans-Jörg und Wilhelm Bauer: *Strategische Dimensionen der Virtual Reality*. In: Warnecke, H J und H J Bullinger (Herausgeber): *Virtual Reality '94*, Seiten 13–26. Springer Berlin Heidelberg, 1994.

[BE92] Bern, Marshall und David Eppstein: *Mesh generation and optimal triangulation*. Computing in Euclidean geometry, 1(1):23–90, 1992.

[Bei12] Beier, Florian: *Entwicklung eines neurochirurgischen Trainingssimulators für intrakranielle Eingriffe*. Doktorarbeit, Universität Mannheim, 2012.

[Ber01] Bergamaschi, Roberto: *Farewell to ßee one, do one, teach one"?* Surgical endoscopy, 15:637–637, 2001.

[BFA02] Bridson, Robert, Ronald Fedkiw und John Anderson: *Robust treatment of collisions, contact and friction for cloth animation*. In: *Acm Transactions on Graphics*, Seiten 594–603, New York, NY, USA, 2002. Stanford Univ, Stanford, CA 94305 USA.

[BGTG04] Bielser, Daniel, Pascal Glardon, Matthias Teschner und Markus H. Gross: *A state machine for real-time cutting of tetrahedral meshes*. Journal of Graphical Models, 66:398–417, 2004.

[BHS03] Bianchi, G, Matthias Harders und Gabor Szekely: *Mesh topology identification for mass-spring models*. Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003, Seiten 50–58, 2003.

[BJ07] Barbic, Jernej und Doug James: *Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models*. In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Seiten 171–180, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[BLo05] Bhasin, Y, A Liu und others: *Bounds for damping that guarantee stability in mass-spring systems*. Studies in health technology and informatics, 119:55, 2005.

[BMG99] Bielser, Daniel, Volker A. Maiwald und Markus H. Gross: *Interactive Cuts through 3-Dimensional Soft Tissue*. EUROGRAPHICS, 18(3):C31–C38, 1999.

[BMLS01] Brown, Joel, Kevin Montgomery, Jean-Claude Latombe und Michael Stephanides: *A Microsurgery Simulation System*. In: Niessen, Wiro und Max Viergever (Herausgeber): *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001*, Seiten 137–144. Springer, Springer Berlin / Heidelberg, 2001.

[BN97] Bro-Nielsen, Morten: *Finite Element Modeling in Surgery Simulation*. Proceedings of the IEEE, 86(3):490–503, 1997.

[boo] boost C++ LIBRARIES. Website (aufgerufen am 03.02.2013): http://www.boost.org.

[BP00] Bern, Marshall und Paul Plassmann: *Mesh Generation*. In: *Handbook of Computational Geometry. Elsevier Science*, Seiten 291–332, 2000.

[BSB+01] Brown, Joel, Stephen Sorkin, C Bruyns, Jean-Claude Latombe, Kevin Montgomery und Michael Stephanides: *Real-time simulation of deformable objects: Tools and application*. In: *Computer Animation, 2001. The Fourteenth Conference on Computer Animation. Proceedings*, Seiten 228–258. IEEE, 2001.

[BSS02] Brown, Joel, Stephen Sorkin und Michael Stephanides: *Algorithmic tools for real-time microsurgery simulation.* Medical Image Analysis, 6(3):289–300, 2002.

[BSS+12] Beier, Florian, Evangelos Sismanidis, A Stadie, K Schmieder, Reinhard Männer und others: *An Aneurysm Clipping Training Module for the Neurosurgical Training Simulator NeuroSim.* Studies in health technology and informatics, 173:42, 2012.

[Bul] Bullet Physics Library. Website (aufgerufen am 03.02.2013): http://bulletphysics.org/wordpress/.

[Bun12] Bundesministerium für Wirtschaft und Technologie: *Zentrales Innovationsprogramm Mittelstand (ZIM)*. Website (aufgerufen am 12.12.2012): http://www.zim-bmwi.de/download/infomaterial/broschuere_zim.pdf, Oktober 2012.

[BW92] Baraff, David und Andrew Witkin: *Dynamic simulation of non-penetrating flexible bodies*. SIGGRAPH Comput. Graph., 26(2):303–308, 1992.

[CDM+02] Cutler, Barbara, Julie Dorsey, Leonard McMillan, Matthias Müller und Robert Jagnow: *A procedural approach to authoring solid models*. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, Seiten 302–311, New York, NY, USA, 2002. ACM.

[DCA99] Delingette, Hervé, Stéphane Cotin und Nicholas Ayache: *A Hybrid Elastic Model allowing Real-Time Cutting, Deformations and Force-Feedback for Surgery Training and Simulation*. Computer Animation, 1999, Proceedings, Seiten 70–81, 1999.

[Del98] Delingette, Hervé: *Toward realistic soft-tissue modeling in medical simulation*. Proceedings of the IEEE, 86(3):512–523, 1998.

[Diz12] DIZIOL, RAPHAEL: *Simulation inkompressibler deformierbarer Körper*. Doktorarbeit, Karlsruher Institut für Technologie, Karlsruhe, 2012.

[Eri05] ERICSON, CHRISTER: *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[GBT06] GISSLER, MARC, MARKUS BECKER und MATTHIAS TESCHNER: *Local constraint methods for deformable objects*. In: *Proc. of the 3rd Workshop in VR Interactions and Physical Simulation (VRIPHYS)*, Seiten 1–8, 2006.

[GÇTS04] GOKTEKIN, T, M CENK ÇAVUSOGLU, F TENDICK und S SASTRY: *GiPSi: An open source/open architecture software development framework for surgical simulation*. Medical Simulation, Seiten 240–248, 2004.

[GLM96] GOTTSCHALK, S, M C LIN und DINESH MANOCHA: *OBBTree: a hierarchical structure for rapid interference detection*. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, Seiten 171–180. ACM, 1996.

[GM97] GIBSON, S F F und B MIRTICH: *A survey of deformable modeling in computer graphics*. Technical Report, 9, 1997.

[gri05] GRIMM, JOHANNES: *Interaktive Echtzeitmodellierung von biologischem Gewebe für Virtuelle Realitäten in der medizinischen Ausbildung*. Doktorarbeit, Universität Mannheim, 2005.

[Han09] HANDEL, HOLGER: *Algorithms for Building High-Accurate Optical Tracking Systems*. Doktorarbeit, Universität Mannheim, 2009.

[HES03] HAUTH, M, O ETZMUSS und WOLFGANG STRASSER: *Analysis of numerical methods for the simulation of deformable models*. The Visual Computer, 19(7):581–600, 2003.

[HK00] HALUCK, RANDY S. und T M KRUMMEL: *Computers and virtual reality for surgical education in the 21st century*. Archives of surgery, 135(7):786, 2000.

[HS06] HAQUE, S. und S. SRINIVASAN: *A meta-analysis of the training effectiveness of virtual reality surgical simulators*. Information Techno-

logy in Biomedicine, IEEE Transactions on, 10(1):51–58, 2006.

[HSSB13] Hüsken, Nathan, Oliver Schuppe, Evangelos Sismanidis und Florian Beier: *MicroSim – A Microsurgical Training Simulator*. Studies in health technology and informatics, 184:205–209, 2013.

[HTK+04] Heidelberger, Bruno, Matthias Teschner, Richard Keiser, Matthias Müller und Markus H. Gross: *Consistent penetration depth estimation for deformable collision response*. In: *In Proceedings of Vision, Modeling, Visualization VMV'04*, Seiten 339–346. Citeseer, 2004.

[Hüs11] Hüsken, Nathan: *Realtime Simulation of Stiff Threads Using Large Timesteps*. In: *VRIPHYS*, Seiten 1–9, 2011.

[Hüs13] Hüsken, Nathan: *Realtime Simulation of Stiff Threads for microsurgery training simulation*. Doktorarbeit, Combined Faculties for the Natural Sciences and for Mathematics, Heidelberg, 2013.

[Jak09] Jakubik, Ole: *Simulation der Phakoemulsifikation im Augenoperationssimulator Eyesi*. Doktorarbeit, Universität Mannheim, 2009.

[JBB+10] Jerábková, Lenka, Guillaume Bousquet, S Barbier, François Faure und Jérémie Allard: *Volumetric modeling and interactive cutting of deformable bodies*. Progress in biophysics and molecular biology, 103(2):217–224, 2010.

[KHM+98] Klosowski, J T, M Held, J S B Mitchell, H Sowizral und K Zikan: *Efficient collision detection using bounding volume hierarchies of k-DOPs*. Visualization and Computer Graphics, IEEE Transactions on, 4(1):21–36, 1998.

[KHVM02] Kornmesser, Ulrike, Jürgen Hesser, Wolfram Voelker und Reinhard Männer: *CATHI - Training on virtual patients for catheter interventions*. Biomedizinische Technik / Supplement, 47(1,1):121–123, 2002.

[Kör03] Körner, Olaf: *Entwicklung eines computergesteuerten Trainingssimulators für die Koloskopie mit aktivem Force-Feedback*. Doktorarbeit, Universität Mannheim, Dezember 2003.

[LAB01] Lannon, Declan A, Jo-Anne Atkins und Peter EM Butler: *Non-vital, prosthetic, and virtual reality models of microsurgical trai-*

ning. Microsurgery, 21(8):389–393, 2001.

[Leia] LEICA MICROSYSTEMS: . Website (aufgerufen am 16.12.2012): http://www.leica-microsystems.com/home/.

[Leib] LEICA MICROSYSTEMS: *Leica M525 F25*. Website (aufgerufen am 15.12.2015): http://www.leica-microsystems.com/fileadmin/downloads/Leica%20M525%20F50/Brochures/Leica_M525_F50_bro_de.pdf.

[LJ94] LIU, A und B JOE: *Relationship between tetrahedron shape measures*. BIT Numerical Mathematics, 34:268–287, 1994.

[Lo66] LEVINTHAL, C und OTHERS: *Molecular model-building by computer*. WH Freeman and Company, 1966.

[LSL98] LOHMAN, R, M SIEMIONOW und G LISTER: *Advantages of sharp adventitial dissection for microvascular anastomoses*. Annals of plastic surgery, 40(6):577–585, 1998.

[Maz97] MAZURA, ANDREAS: *Virtuelles Schneiden in Volumendaten*. Doktorarbeit, Universität Karlsruhe, 1997.

[MK00] MOR, ANDREW B. und TAKEO KANADE: *Modifying soft tissue models: progressive cutting with minimal new element creation*. CVR Med., 19:598–607, 2000.

[MMDJ01] MÜLLER, MATTHIAS, LEONARD MCMILLAN, JULIE DORSEY und ROBERT JAGNOW: *Real-time simulation of deformation and fracture of stiff materials*. In: *Proceedings of the Eurographic workshop on Computer animation and simulation*, Seiten 113–124, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[MS03] MASON, WTM und P W STRIKE: *Short Communication See one, do one, teach one-is this still how it works? A comparison of the medical and nursing professions in the teaching of practical procedures**. Medical Teacher, 25(6):664–666, 2003.

[MSCT08] MÜLLER, MATTHIAS, JOS STAM, DOUGH JAMES CORNELL und NILS THUREY: *Real Time Physics Class Notes* . In: *Multiple values selected*, Seiten 88:1–88:90, New York, NY, USA, 2008. ACM.

[MT03] MÜLLER, MATTHIAS und MATTHIAS TESCHNER: *Volumetric Meshes for Real-Time Medical Simulations*. In: *Proc. BVM '03*, Seiten 279–283, 2003.

[MT07] MIEHLKE, ADOLF und ULRICH TRÖHLER: *Illustrierte Geschichte der Mikrochirurgie: Die historische Entwicklung in den verschiedenen operativen Disziplinen*. Voltmedia, Paderborn, 2007.

[MW88] MOORE, MATTHEW und JANE WILHELMS: *Collision Detection and Response for Computer Animation*. In: *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, Seiten 289–298, New York, NY, USA, 1988. ACM.

[Neu09] NEUMANN, JENS: *Verfahren zur ad hoc-Modellierung und -Simulation räumlicher Feder-Masse-Systeme für den Einsatz in Virtual Reality-basierten Handhabungssimulation*. Berichte aus dem Produktionstechnischen Zentrum Berlin. Fraunhofer-IRB-Verlag, 2009.

[NMK+06] NEALEN, A, MATTHIAS MÜLLER, RICHARD KEISER, E BOXERMAN und M CARLSON: *Physically based deformable models in computer graphics*. In: *Computer Graphics Forum*, Seiten 809–836. Wiley Online Library, 2006.

[NvdS00] NIENHUYS, HAN-WEN und A FRANK VAN DER STAPPEN: *Combining finite element deformation with cutting for surgery simulations*. In: SOUSA, A DE und J C TORRES (Herausgeber): *EuroGraphics Short Presentations*, Seiten 43–52, 2000.

[NvdS01] NIENHUYS, HAN-WEN und A FRANK VAN DER STAPPEN: *Supporting cuts and finite element deformation in interactive surgery simulation*. Technischer Bericht UU-CS-2001-16, 2001.

[O'C04] O'CONNOR, JAMES P.: *Objective assessment of technical skills in surgery: surgical skills and facts may be best taught in non-clinical training modules*. BMJ (Clinical research ed.), 328(7436), Februar 2004.

[Ogr] OGRE3D. Website (aufgerufen am 03.02.2013): http://www.ogre3d.org.

[Ope] OPENSCENEGRAPH. Website (aufgerufen am 03.02.2013): http://www.openscenegraph.org/projects/osg.

[OTSG09] Otaduy, M A, Rasmus Tamstorf, Denis Steinemann und Markus H. Gross: *Implicit contact handling for deformable objects*. In: *Computer Graphics Forum*, Seiten 559–568. Wiley Online Library, 2009.

[PB81] Platt, Stephen M und Norman I Badler: *Animating facial expressions*. SIGGRAPH Comput. Graph., 15(3):245–252, 1981.

[Phy] PhysX Engine. Website (aufgerufen am 03.02.2013): http://www.nvidia.de/object/physx_new_de.html.

[Plc] Plc, Digia: *QT*. Website (aufgerufen am 22.12.2015): http://qt.digia.com.

[PP96] Provot, Xavier und Xavier Provot: *Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior*. In: *In Graphics Interface*, Seiten 147–154, 1996.

[Pre10] Pressestelle Kommunikation und Marketing Universität Heidelberg: *Simulator für Mikrochirurgie: Training für angehende Mediziner*. Website (aufgerufen am 12.12.2012): http://www.uni-heidelberg.de/presse/news2010/pm20100729_mikrochirurgie.html, 2010.

[Pri] Private Akademie der Deuchen Gesellschaft für Gefässchirurgie und Gefässmedizin: *Chirurgie Workshops* . Website (aufgerufen am 17.10.2012): http://www.dgg-akademie.de/kurse-der-akademie.html.

[Pro97] Provot, X: *Collision and self-collision handling in cloth model dedicated to design garments*. In: *Graphics interface*, Seiten 177–189. Eurographics Association, 1997.

[Ros08] Rosen, Kathleen R: *The history of medical simulation*. Journal of Critical Care, 23(2):157–166, 2008.

[RS95] Remmert, S. und K. Sommer: *Kompendium des mikrovaskulären Gewebetransfers*. Ethicon, Hamburg-Norderstedt, 1995.

[RVF+04] Rossi, Juliana V., Dinesh Verma, G. Y. Fujii, R.R. Lakhanpal, S.L. Wu, M.S. Humayun und Eugene De Juan Jr.: *Virtual Vitreoretinal Surgical Simulator as a Training Tool*. Retina (Philadelphia, Pa.), 24(2):231, 2004.

[S&] S&T AG. Website (aufgerufen am 16.12.2012): http://s-and-t.net.

[SABW82] Swope, William C., Hans C. Andersen, P. Berenc und Kent R. Wilson: *A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to small water Clusters*. Chemical Physics, 76(1), 1982.

[Sch01] Schill, Markus: *Biomechanical Soft Tissue Modeling - Technique Implementation and Applications*. Doktorarbeit, Universität Mannheim, 2001.

[Sch12] Schuppe, Oliver: *An optical tracking system for a microsurgical training simulator.* Studies in health technology and informatics, 173:445–449, 2012.

[SCN+12] Sigounas, V Y, P W Callas, C Nicholas, J E Adams, D J Bertges, A C Stanley, G Steinthorsson und M A Ricci: *Evaluation of Simulation-Based Training Model on Vascular Anastomotic Skills for Surgical Residents*. Simulation in Healthcare, 2012.

[SDF07] Sifakis, Eftychios, Kevin G. Der und Ronald Fedkiw: *Arbitraty Cutting of Deformable Tetrahedralized Objects*. Eurographics / ACM SIGGRAPH Symposium on Computer Animation, Seiten 73–80, 2007.

[SGT09] Schmedding, Ruediger, Marc Gissler und Matthias Teschner: *Optimized damping for dynamic simulations*. In: *Proceedings of the 2009 Spring Conference on Computer Graphics*, Seiten 189–196, New York, NY, USA, 2009. ACM.

[She02] Shewchuk, Jonathan Richard: *What is a Good Linear Element? Imterpolation, Conditioning, and Quality Measures*. International Meshing Roundtable, 11:115–126, 2002.

[SHGS06] Steinemann, Denis, Matthias Harders, Markus H. Gross und Gabor Szekely: *Hybrid Cutting of Deformable Solids*. Virtual Reality Conference, Seiten 35–42, 2006.

[Shi05] Shinya, Mikio: *Theories for Mass-Spring Simulation in Computer Graphics: Stability, Costs and Improvements*. IEICE - Trans. Inf. Syst., E88-D(4):767–774, 2005.

[Sis12] Sismanidis, Evangelos: *Real-time simulation of blood vessels and connective tissue for microvascular anastomosis training*. Virtual Reality Workshops (VR), 2012 IEEE, Seiten 109–110, 2012.

[Sis13] Sismanidis, Evangelos: *Triggering different collision response algorithms stated at penetration depths*. In: *IASTED - Modelling, Identification and Control*, April 2013.

[SSSH11] Seiler, Martin, Denis Steinemann, Jonas Spillmann und Matthias Harders: *Robust interactive cutting based on an adaptive octree simulation mesh*. Vis. Comput., 27(6-8):519–529, 2011.

[Ste92] Steuer, Jonathan: *Defining virtual reality: Dimensions determining telepresence*. Journal of communication, 42(4):73–93, 1992.

[SWH+99] Schill, Markus, Clemens Wagner, Marc Hennen, Hans-Joachim Bender und Reinhard Männer: *EyeSi - A Simulator for Intra-ocular Surgery.* In: *MICCAI*, Seiten 1166–1174. Springer, 1999.

[SWT06] Spillmann, Jonas, Michael Wagner und Matthias Teschner: *Robust tetrahedral meshing of triangle soups*. In: *In Proc. Vision, Modeling, Visualization (VMV*, Seiten 9–16, 2006.

[TBB+06] Thorne, Charles H., Scott P. Bartlett, Robert W. Beasley, Sherrell J. Aston, Geoffrey C. Gurtner und Scott L. Spear: *Grabb and Smith's Plastic Surgery*. Lippincott Williams and Wilkins, 6th Auflage, 2006.

[TCYM08] Tang, M, S Curtis, S E Yoon und Dinesh Manocha: *Interactive continuous collision detection between deformable models using connectivity-based culling*. In: *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, Seiten 25–36. ACM, 2008.

[The] The Industry's Foundation for High Performance Graphics,: *OpenGL*. Website (aufgerufen am 15.12.2015): http://www.opengl.org/.

[THM+03] Teschner, Matthias, Bruno Heidelberger, Matthias Müller, Danat Pomeranets und Markus H. Gross: *Optimized Spatial Hashing for Collision Detection of Deformable Objects*. Proc. of Vision, Moddling , Visualization, Seiten 47–54, 2003.

[THMG04] Teschner, Matthias, Bruno Heidelberger, Matthias Müller und Markus H. Gross: *A versatile and robust model for geometrically complex deformable solids*. CGI '04 Proceedings of the Computer Graphics Internationa, Seiten 312–319, 2004.

[TKH+05] Teschner, Matthias, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, Marie-Paule Cani, François Faure, Nadia Magnenat Thalmann, Wolfgang Strasser und Pascal Volino: *Collision Detection for Deformable Objects*. In: *Eurographics State-of-the-Art Report (EG-STAR)*, Seiten 61–81. Eurographics Association, 2005.

[TMOT12] Tang, Min, Dinesh Manocha, Miguel A. Otaduy und Ruofeng Tong: *Continuous penalty forces*. ACM Trans. Graph., 31(4):107:1–107:9, 2012.

[VCMT95] Volino, Pascal, Martin Courchesne und Nadia Magnenat Thalmann: *Versatile and efficient techniques for simulating cloth and other deformable objects*. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, Seiten 137–144, New York, NY, USA, 1995. ACM.

[VDB97] Van Den Bergen, G: *Efficient collision detection of complex deformable models using AABB trees*. Journal of Graphics Tools, 2(4):1–13, 1997.

[Ver67] Verlet, Loup: *Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules*. Phys. Rev., 159:98–103, 1967.

[VRm] VRmagic GmbH: *VRmagic GmbH*. Website (aufgerufen am 16.10.2012): http://www.vrmagic.com/de/home.

[VT94] Volino, Pascal und N M Thalmann: *Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity*. In: *Computer Graphics Forum*, Seiten 155–166. Wiley Online Library, 1994.

[Wag03] Wagner, Clemens: *Virtuelle Realitäten für die chirurgische Ausbildung: Strukturen, Algorithmen und ihre Anwendung*. Doktorarbeit, Universität Mannheim, 2003.

[Wat81] WATSON, D F: *Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes*. The computer journal, 24(2):167–172, 1981.

[WB01] WITKIN, A und DAVID BARAFF: *Physically Based Modeling Differential Equation Basics*. online]. California, Pixar. Available from: http://www. pixar. com/companyinfo/research/pbm2001/notesc. pdf [Accessed 5 July 2005], 2001.

[WBJ+06] WANG, P, A A BECKER, I A JONES, A T GLOVER, S D BENFORD, C M GREENHALGH und M VLOEBERGHS: *A virtual reality surgery simulation of cutting and retraction in neurosurgery with force-feedback*. Computer Methods and Programs in Biomedicine, 84(1):11–18, 2006.

[Web09] WEBER, KATHRIN: *Interaktive Echtzeitsimulation deformierbarer Oberflächen für Trainingssysteme in der Augenchirurgie*. Doktorarbeit, Universität Mannheim, 2009.

[WHPK10] WOLFE, SCOTT W, ROBERT N HOTCHKISS, WILLIAM C PEDERSON und SCOTT H KOZIN: *Green's Operative Hand Surgery*. Churchill Livingstone, 6 Auflage, 2010.

[Wit97] WITKIN, A: *Physically Based Modeling: Principles and Practice Constrained Dynamics*. SIGGRAPH Course notes, Seiten 11–21, 1997.

[WSBB08] WANG, FEI, EILEEN SU, ETIENNE BURDET und HANNES BLEULER: *Development of a microsurgery training system*. Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, 2008:1935–1938, 2008.

[WTF06] WEINSTEIN, RACHEL, JOSEPH TERAN und RONALD FEDKIW: *Dynamic Simulation of Articulated Rigid Bodieswith Contact and Collision*. Visualization and Computer Graphics, IEEE Transactions on, 12(3):365–374, Januar 2006.

[ZL03] ZACHMANN, G und E LANGETEPE: *Geometric data structures for computer graphics*. Tutorial at ACM SIGGRAPH, 16:1–54, 2003.