

# Echtzeitdeformation und Kollisionserkennung zur virtuellen Operationssimulation

Christian Wienss<sup>(1)</sup>, Gernot Goebbels<sup>(2)</sup>, Igor Nikitin<sup>(3)</sup>, Stefan Müller<sup>(1)</sup>

<sup>(1)</sup> Universität Koblenz	<sup>(2)</sup> fleXilution GmbH	<sup>(3)</sup> Universität zu Köln
Arbeitsgruppe Computergrafik	Gottfried-Hagen-Strasse 60	Mathematisches Institut
Universitätstrasse 1	D-51105 Köln	Weyertal 86-90
D-56070 Koblenz	Tel.: +49/221/98933804	D-50931 Köln
Tel.: +49/261/287-0		Tel.: +49/221/470-2275
cwienss@uni-koblenz.de		

**Zusammenfassung:** Dieses Paper präsentiert einen Ansatz zur Kollisionserkennung und Deformation mit Hilfe der finiten Elemente. Ein echtzeitfähiger Algorithmus zur Berechnung der Eindringtiefe und der Kollisionspunkte wird vorgestellt. Ein Deformationssimulator kann aufgrund dieser Daten eine korrekte Verformung durchführen. Die hohe zeitliche Komplexität dieses Vorgangs wird durch geeignete Ansätze wie z.B. Vorberechnung gesenkt.

**Stichworte:** Elastische Deformation, Virtual Environments, Finite Elemente

## 1 Einleitung

In virtuellen Simulationsumgebungen ist es wichtig, dass sich Objekte so realistisch wie möglich verhalten. Dies ist insbesondere bei der Simulation von chirurgischen Operationen unabdingbar. Hierzu muss nicht nur eine sehr genaue Kollisionserkennung stattfinden, sondern auch ein realistisches Objektverhalten simuliert werden. Zudem kann ein solches System nur zur Anwendung kommen, wenn es echtzeitfähig ist. Die Immersion und die Benutzerführung darf nicht durch Latenzen gestört werden. Dieses Paper setzt sich mit der physikalischen Simulation von Gewebeverhalten unter Kraftereinwirkungen auseinander. Zur haptischen Vermittlung von Kräften und zur korrekten Deformation von flexiblen Materialien an starren Objekten wird eine genaue Kollisionserkennung und -behandlung benötigt. Zur genauen Errechnung und Darstellung der Deformation ist die präzise Bestimmung der Eindringtiefe des flexiblen Objekts in das starre Objekt entscheidend.

## 2 Grundlagen

### 2.1 Materialeigenschaften

Modelldeformationen auf Basis der Finiten Elemente [ZT00] können elementspezifisch definiert werden: jedes Element kann eine andere Materialeigenschaft haben. So können komplexe Modellstrukturen simuliert werden. Da sich auch mit der Simulation von erkrankten

Organen befasst wird, müssen Verhärtungen oder Aufschwemmungen erkennbar und darstellbar sein. Manche Materialien verhalten sich bei geringen Deformationen allerdings wie eine homogene Struktur, auch wenn die Innenstruktur heterogen ist. Dies ist z.B. bei der Simulation des Deformationsverhaltens von Niere und Leber der Fall. Bei der Definition von Material genügt die Angabe der drei Eigenschaften: die Dichte des Materials ( $\text{kg}/\text{m}^3$ ), der Elastizitätsmodul (Pa), und die Reziproke Querkontraktion (Poisson Verhältnis).

## 2.2 Linearität in der Deformation

Es gibt zwei Arten der Nichtlinearität [KNN03], die materialbedingte und die geometrisch bedingte [LL70]. Die materialbedingte Nichtlinearität, hervorgerufen durch Verletzung der Dehnungs-Spannungsrelation, tritt meist unter Extrembelastungen auf, wenn das Material seine elastischen Eigenschaften verliert. Die geometrisch bedingte Nichtlinearität tritt bei starken Verschiebungen der Knoten untereinander auf. *Beispiel:* Gegeben sind zwei nahe beieinander liegende Punkte. Der Eine wird verschoben mit  $x \rightarrow x + \gamma(x)$ , der Andere mit  $(x + \delta x) \rightarrow (x + \delta x) + \gamma(x + \delta x)$ . Zieht man die beiden Positionen voneinander ab, erhält man  $\delta x \rightarrow \delta x + \gamma(x + \delta x) - \gamma(x) = \delta x + \delta\gamma$ . Das Quadrat der Distanz hat sich geändert zu  $\delta x^2 \rightarrow \delta x^2 + 2\delta x * \delta\gamma + \delta\gamma^2$ . Für kleine Deformationen kann man den Term  $\delta\gamma^2$  in der Regel vernachlässigen, so bleibt die Gleichung linear. Bei großen Deformationen, z. B. wenn die Differenz der Deformationen  $\delta\gamma$  in die Größenordnung des Abstands zwischen den beiden Punkten kommt, kann man diesen Term nicht mehr vernachlässigen, die Deformation wird nichtlinear. Diese Eigenschaft ist von der Materialeigenschaft unabhängig und ist die rein geometrische Nichtlinearität. In Folge dessen kann hier die Nichtlinearität vernachlässigt werden. Objekte mit ungleichmäßiger Ausdehnung können so auch dargestellt werden, allerdings nur unter Beschränkung auf sehr kleine Deformationen. Über diese Einschränkungen hinausgehende Anforderungen müssen entweder durch das Hinzufügen von allgemeinen nichtlinearen Methoden [PDN00] [ZC01] [HFL00] oder objektspezifische Näherungen realisiert werden.

## 3 Related Work

### 3.1 Voxelbasiertes Sampling

Zum haptischen Rendering wurde von der Firma Boeing [MPT99] [RPP01] der Voxmap-Pointshell<sup>TM</sup>-Algorithmus veröffentlicht, der einen einfachen, schnellen, nähernden und voxelbasierenden Ansatz mit sechs Freiheitsgraden vorstellt. Voxelbasierte Methoden wurden schon umgesetzt, aber nicht auf Basis von haptischen Schnittstellen [GSF94] [KCY93] [Log96].

In diesem Ansatz werden dynamische Objekte durch ihre Oberflächenpunkte beschrieben, zu jedem Oberflächenpunkt wird die inverse Normale angegeben. Diese Darstellung wird als Punktschale (*point shell*) beschrieben. Die statische Umgebung ist durch die räumliche Be-

legung der statischen Objekte innerhalb einer Voxelumgebung beschrieben, die sogenannte *voxmap*. Für jedes Voxel im Raum wird ein Wert definiert, ob er sich in der Nähe eines statischen Objekts befindet, die Oberfläche beinhaltet oder ob er sich innerhalb dessen befindet. Bei jedem Frame wird jeder Oberflächenpunkt gegen diese *voxmap* abgetastet. Wenn sich nun ein Punkt innerhalb eines Oberflächenvoxels befindet wird die Eindringtiefe als die Distanz  $d$  vom Punkt zur Tangentenebene errechnet. Die aus der Summe der kollidierenden Punkte genäherte Kraft wird an das haptische Interaktionsgerät zurückgeliefert.

### 3.2 Deformation volumetrischer Modelle

Die Veröffentlichung von Bro-Nielsen und Cotin [BNC96] war Pionierarbeit zur Volumensimulation. Unter Volumendatensätzen versteht man Modelle, die sowohl auf der Oberfläche als auch im Inneren des Objekts Punkte besitzen. Diese sind durch Tetraeder miteinander verbunden. So kann Innenstruktur simuliert werden, welche bei Schnittsimulation eine realitätsnahe Situation darstellt. Ein weiterer großer Vorteil liegt in der Zuweisung von Materialeigenschaften. Jedes Element kann eine andere Beschaffenheit besitzen, so dass Strukturen im Inneren von Modellen simuliert werden können.

### 3.3 Deformation mit Interaktionselementen

In Nikitin et al. (2002) [NNF02] wird ein Ansatz beschrieben, der sowohl unter Verwendung von FEM als auch BEM mit Hilfe von Interaktionselementen die Deformationen vorberechnet. Es findet eine Unterteilung in vorberechnete Daten und im Simulator berechnete Daten statt. Das Verfahren basiert auf der linearen Gleichung  $\mathcal{K} * u = f$ , in der  $f$  die bekannte Kraft in den Knoten des Objekts,  $u$  die unbekannte Verformung derselben ist.  $\mathcal{K}$  ist die Steifigkeitsmatrix, welche durch die Materialeigenschaften und die Form des Objekts gegeben ist. Offline wird die sehr dünn besetzte Matrix  $\mathcal{K}$  zur dichtbesetzten Matrix  $\mathcal{K}^{-1}$  invertiert. Zur Darstellung der Deformation reicht es aus, die inverse Steifigkeitsmatrix auf diejenigen Punkte zu beschränken, auf welche Kraft ausgeübt werden kann: die Oberflächenpunkte. Diese auf die Oberflächenpunkte beschränkte Submatrix von  $\mathcal{K}^{-1}$  wird im folgenden als  $(\mathcal{K}^{-1})'$  bezeichnet. Vor der Reduzierung auf die Submatrix muss  $\mathcal{K}$  invertiert werden, um die Kräfte der innen wirkenden Knoten mit einzubeziehen. Hieraus kann online mit  $u = (\mathcal{K}^{-1})' * f$  die Verformung errechnet werden. Es muss nicht die ganze, kaum zu komprimierende, invertierte Matrix gespeichert werden, sondern eine Beschränkung auf diejenigen Oberflächenelemente, auf welche externe Kraft einwirken kann, reicht aus [BNi97]. In diesem Ansatz wird die Kraft durch Interaktionselemente ausgeübt. Es gibt endlich viele physikalisch durchführbare Zustände, in die die Objekte überführt werden können. Diese werden vorberechnet und in einem dafür entworfenen Datenformat gespeichert, so dass diese im Simulator ausgelesen, dargestellt und gegebenenfalls interpoliert werden können.

### 3.4 Deformation mit Kollision

In Klimenko et al. [KNN03] wurde neben den Ergebnissen aus [NNF02] auch ein Ergebnis präsentiert, dass Deformation anhand von Kollision beschreibt. Es basiert auf einem flexiblen BEM-Modell, als starres Objekt dient eine Kugel festgelegter Größe. Die Eindringtiefe lässt sich sehr schnell durch den Abstand der Oberflächenpunkte zum Kugelmittelpunkt bestimmen. Dieser Ansatz lässt sich nur mit einer Kugel umsetzen. Hieraus wird dann die Verschiebung  $u$  berechnet und das Objekt deformiert. Da es sich um eine BEM-Simulation handelt, kann keine Innenstruktur mit anderen Materialeigenschaften simuliert werden.

## 4 Methodik

### 4.1 Wahl der Technik

Die im Folgenden dieses Papers vorgestellten Mechanismen wurden unter dem Hintergrund ausgewählt, echtzeitfähig und realitätsnah zu arbeiten. Ein voxelbasierter Ansatz eröffnet die Möglichkeit, neben der Kollisionserkennung (PointShell-Algorithmus<sup>TM</sup>) auch einen Weg zur Bestimmung der Überlappung der Objekte zu finden (Occupancy Map Method). Hieraus lässt sich ein Ansatz entwickeln, der eine Eindringtiefe berechnen kann (vgl. Abschnitt 4.2). Aufgrund der Struktur der Objekte ist vor dem Inkrafttreten des voxelbasierten Kollisionserkennungsverfahrens zunächst eine sehr schnelle und einfache Verfahrensweise zur Kollisionserkennung erforderlich. Hierzu wurde sich zu der einfachen Abfrage entschieden, ob eine Überschneidung in einer achsenparallelen Ebene der Bounding Boxes stattfindet. Sollte dies der Fall sein, werden wiederum alle Punkte der flexiblen Objekte gegen die Bounding Boxes der starren Objekte getestet. Danach erfolgt das für das Ermitteln der Eindringtiefe wichtige voxelbasierte Verfahren. Andere Verfahren konnten mit der Einfachheit dieses Vorgehens insofern nicht konkurrieren, da sie darauf ausgelegt sind, exakte Kollisionsdaten zu liefern, was im Rahmen dieses Papers im ersten Kollisionserkennungsschritt nicht erforderlich ist. Zur Feinkollisionserkennung wurde ein Ansatz ausgewählt, der dem Voxmap Pointshell<sup>TM</sup>-Algorithmus von Boeing ähnelt (vgl. Abschnitt 3.1). Dieser Ansatz bietet die Erweiterungsmöglichkeit, gleichzeitig eine Kollisionserkennung und eine Eindringtiefe zu errechnen.

### 4.2 Kollisionserkennung

Sind in der Szene nur starre Objekte vorhanden, besteht eine Kollision aus einem Austausch von Energie. In dem Falle, in dem die Objekte flexibel sind, wird Energie nicht nur ausgetauscht, sondern in Form von Deformation auch abgeleitet und gespeichert. Unter Umständen kann von dem Objekt ein Verhalten wie brechen oder reißen erwartet werden. Es wird vorerst von dem Fall ausgegangen, dass es ein beliebig geformtes statisches Objekt und ein beliebig geformtes dynamisches Objekt gibt, welches mit dem statischen Objekt kollidieren kann. Ein Objekt kollidiert mit einem anderen Objekt, wenn sich mindestens ein Punkt

innerhalb oder auf der Oberfläche eines anderen Objekts befindet. Das dynamische Objekt wird zunächst über eine Bounding Box repräsentiert, gegen die auf Kollision getestet wird. Bei Überlappung werden alle Oberflächenpunkte getestet. Die Kollisionserkennung in dieser Arbeit wird durch eine Unterteilung in drei Stufen realisiert (vgl. Abb. 1).

1. Kollisionserkennung zwischen den Bounding Boxes der Objekte.
2. Kollisionserkennung zwischen den Punkten des flexiblen Objekts und der Bounding Box des starren Objekts.
3. Kollisionserkennung durch Interpolation zwischen Gitterpunkten.

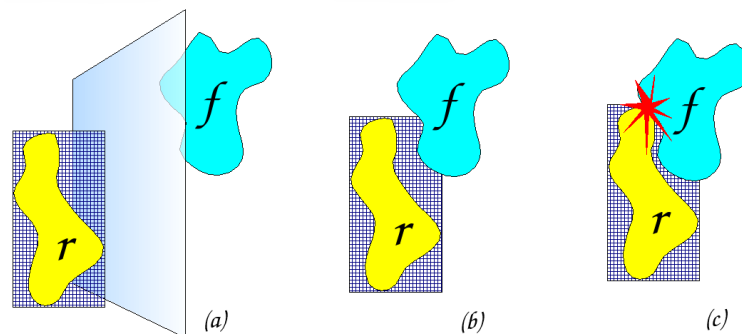


Abbildung 1: Eine separierende Ebene kann gefunden werden (a). Sobald ein Punkt des flexiblen Objekts  $f$  innerhalb der Bounding Box des rigiden Objekts  $r$  liegt, wird der Abstand zur Oberfläche über die Gitterpunktinformationen interpoliert. Als Ergebnis wird der Wert *außerhalb* zurückgeliefert (b) oder die berechnete Eindringtiefe bei Kontakt (c).

Wenn von einer der Stufen zurückgegeben wird, dass keine Kollision stattfindet, wird der Vorgang abgebrochen. Die Punkte **1.** und **2.** werden durch eine Abwandlung des *Separating Plane*-Algorithmus auf der Basis von achsenparallelen Ebenen durchgeführt, da eine achsenparallele, rechteckige Bounding Box konvex ist. Im ersten Fall werden die Grenzen der Bounding Box getestet, im zweiten Fall jeder Punkt des flexiblen Objekts. Solange sich die Bounding Boxes nicht überschneiden, kann keine Kollision stattfinden. In zwei räumlich getrennten Regionen ist die einfache Abfrage ausreichend, ob sich die Bounding Boxes in einer Ebene nicht überschneiden. Hierzu genügt die Abfrage, ob eine Dimension existiert, in der alle Werte der einen Bounding Box größer oder kleiner sind als in der anderen. Die Ebene wird nicht durch ihre Gleichung definiert, ihre Existenz wird nur angenommen. Durch diese nicht iterative Vorgehensweise ist der Algorithmus deterministisch. Sobald eine Überschneidung der Bounding Boxes stattfindet, wird über die Informationen der Gitterpunkte interpoliert und für die betreffenden Punkte die Eindringtiefe berechnet [Wie04]. Das starre Objekt ist in das flexible Objekt eingedrungen, d.h. Punkte des flexiblen Objektes liegen innerhalb des starren Objekts (vgl. Abb. 2). Das starre Objekt ist gefüllt mit Gitterpunkten, in welchen Ort und Abstand des nächsten Oberflächenpunktes gespeichert sind. Anhand dieser

Informationen wird interpoliert, wohin der Oberflächenpunkt des flexiblen Objekts auf die Oberfläche des starren Objektes projiziert wird. Aus dieser Projektion ergibt sich aus der Integration über die Stützstellen ein Verschiebungsfeld (vgl. Abb. 3). Dieses wird als  $u$  in die Gleichung eingesetzt und man erhält die auf das Objekt wirkende Kraft. Aufgrund dieser Information kann man nun die Verschiebung der anderen Punkte des flexiblen Objektes berechnen und darstellen (vgl. Abb. 4).

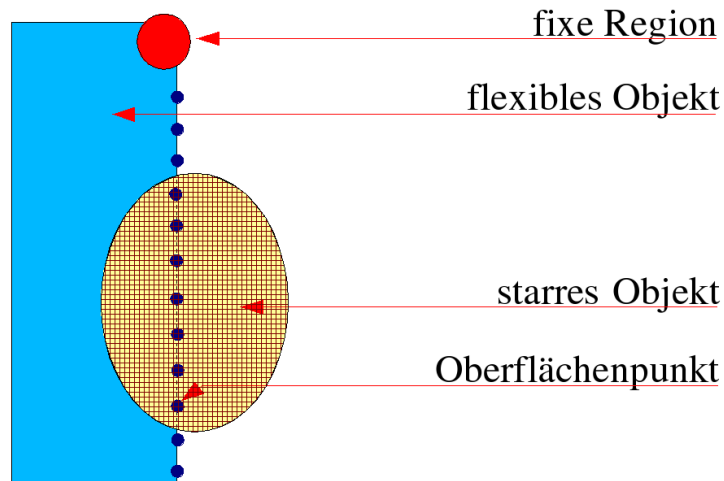


Abbildung 2: Ein starres Objekt dringt in ein flexibles Objekt ein. Eine fixe, von Deformationen nicht betroffene Region bezweckt, dass sich das flexible Objekt nicht verschiebt und ist als Randbedingung zur Lösung des Systems nötig.

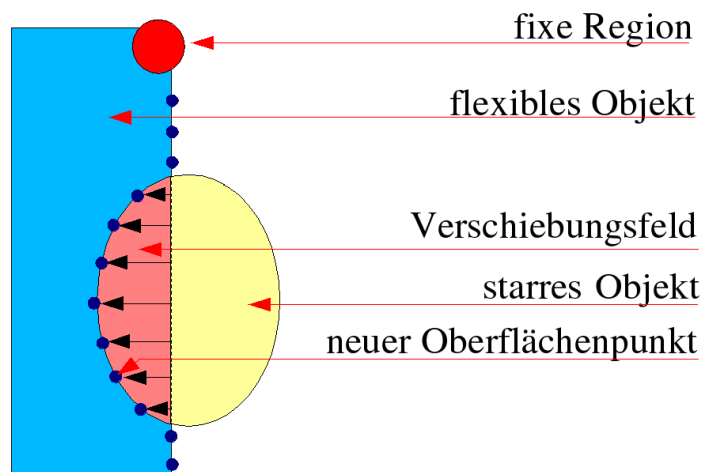


Abbildung 3: Vor der Berechnung der auf das Objekt wirkenden Kraft wird die Größe des Verschiebungsfeldes berechnet. Hierzu werden die Oberflächenpunkte des flexiblen Objekts auf die Oberfläche des starren Objekts projiziert und die Eindringtiefe errechnet.

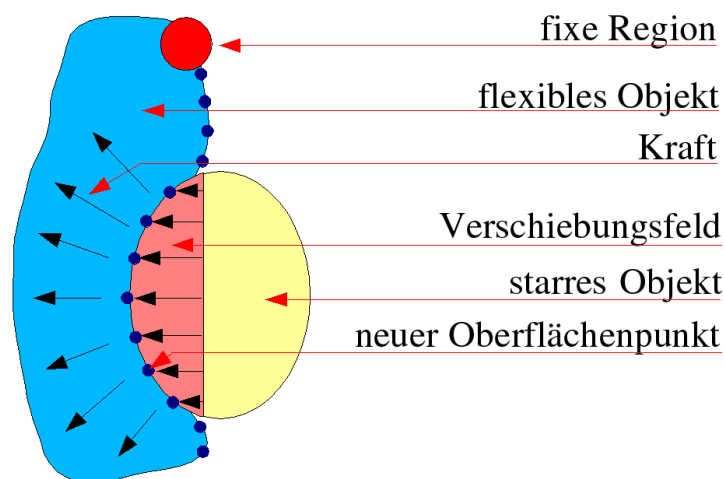


Abbildung 4: Eine mögliche Deformation eines sehr weichen flexiblen Objekts aufgrund des Verschiebungsfeldes.

### 4.3 Deformation: online vs. offline

Zur Lösung von linearen Deformationen existiert die Möglichkeit, die Elastizitätsgleichung online zu lösen und gleichzeitig die graphische Darstellung durchzuführen [NNF02]. Lineare Problemstellungen in der Elastizitätstheorie erfordern die Lösung von großen linearen Systemen der Form  $\mathcal{K}u = f$  mit konstanter Matrix  $\mathcal{K}$  und variabler Kraft  $f$ . So können moderat große Modelle berechnet werden [PDN00] (1.400 Knoten bei 45 fps). Bei der Verwendung von vorberechneten Daten und invertierter Submatrix  $(\mathcal{K}^{-1})'f$  und der Gleichungsdarstellung  $u = (\mathcal{K}^{-1})'f$ , kann eine größere Performanz erreicht werden [NNF02] (10.000 Knoten bei 85 fps). Der Vorteil der online-Lösung liegt in der Möglichkeit der Verwendung von nicht-linearem Objektverhalten [PDN00], welches annähernd in Echtzeit durchführbar ist (1.400 Knoten bei 8 fps). Dafür kann interaktiv die Matrix  $\mathcal{K}$  verändert werden und somit die Randbedingungen und die Topologie des Objekts. So sind z.B. Schnitte im Modell realisierbar. Dieses Vorgehen wurde in aktuellen Arbeiten [JP99][MMP01] schon für offline-Vorberechnungen implementiert. Hierbei wurde ein Algorithmus verwendet, der die schnelle Evaluation von  $(\mathcal{K}^{-1})'$  auf Basis der Boundary Element Methode umsetzt.

## 5 Implementation

### 5.1 Kollisionserkennung

Ein Objekt wird in Form eines Inventorfiles eingelesen und in gleichmäßige Voxel unterteilt. Es wird eine Datei erstellt, in der nicht mehr das Objekt, sondern nur noch die Gitterpunkte der Voxel zeilenweise mit folgenden Informationen gespeichert werden:

1. Das erste Zahlentripel definiert die Position des Gitterpunktes. Jedes Tripel enthält die Koordinaten in folgender Reihenfolge: X-, Y- und Z-Koordinate.

2. Das zweite Zahlentripel definiert die Position des nächsten Oberflächenpunktes.
3. Die Distanz zwischen dem Gitterpunkt und dem Oberflächenpunkt. Dieser Wert wird negativ angegeben, wenn sich der Gitterpunkt außerhalb des Objekts befindet. Ansonsten ist der Wert positiv. Wegen dieser Zusatzinformation und weil sie häufig verwendet wird, wird die Distanz zusätzlich gespeichert.

Zuerst werden die Triangle-Strips aufgelöst und als Dreiecke gespeichert. Aus der Anordnung der Dreiecke im Strip kann erkannt werden, welche Seite des Dreieckes nach außen zeigt. Bei der Speicherung wird dieser Wert als Normale behandelt. Die Entscheidung, ob ein Punkt innerhalb des Objekts liegt, wird zunächst über den Winkel zur Normalen des nächsten Dreieckes entschieden. Liegt im Zweidimensionalen dieser zwischen  $90^\circ$  und  $270^\circ$ , ist der Punkt innerhalb des Objekts. Im Dreidimensionalen liegen die Winkel zwischen Vektoren zwischen  $0^\circ$  und  $180^\circ$ . Winkel zwischen  $90^\circ$  und  $180^\circ$  sind Ergebnisse für Punkte, die innerhalb des Objekts liegen. Nun wird von dem aktuellen Gitterpunkt die Entfernung zu jedem Dreiecksmittelpunkt gemessen und die Kürzeste vermerkt.

Für solche Sonderfälle und für die Situationen, in denen der Gitterpunkt sehr nahe der Oberfläche ist, ist eine unterstützende Methode hinzugefügt worden. Zur Berechnung des Raumwinkels wird wie folgt vorgegangen:

$$\frac{1}{4\pi}\Omega = \frac{1}{4\pi} \sum_i \frac{\vec{s}_i * \vec{r}_i}{|\vec{r}_i|^3}. \quad (1)$$

Hierbei stehen der Vektor  $\vec{r}$  für die Strecke Gitternetzpunkt  $\rightarrow$  Oberflächenpunkt und  $\vec{s}$  für einen flächengewichteten Vektor entlang der Normale des Dreiecks. Die Division durch  $4\pi$  wurde beibehalten, um zu verdeutlichen, dass die Summe aller Vektoren  $\vec{r}$  von einem Gitterpunkt  $\mathcal{P}$  aus gesehen  $4\pi$  ergibt (der Raumwinkel einer Umkugel ist gleich  $4\pi$ ). Das Ergebnis der Gleichung ist 0, wenn  $\mathcal{P}$  außerhalb des Objekts liegt und 1, wenn  $\mathcal{P}$  innerhalb des Objekts liegt.

## 5.2 Deformation: Vorberechnung

Die Deformation mit der Finiten Element Methode basiert auf Objektknoten, die untereinander Kräfte ausüben. Diese Kräfte werden durch eine Matrix  $\mathcal{K}$  von jedem Element auf jedes Element beschrieben. Die dünnbesetzte, symmetrische Matrix  $\mathcal{K}$  besteht aus  $n^2$   $3 \times 3$  Matrizen, wobei  $n$  die Anzahl aller Knotenpunkte darstellt. Die Matrix  $\mathcal{K}$  ist also  $3n \times 3n$  groß. Weniger als 1% der Matrix ist ungleich Null. Diese Matrix kann effizient in Datenstrukturen gespeichert werden, die dem Elementverbund entsprechen. Die diagonalen Einträge  $\mathcal{K}_{ii}$  sind in den Knotenpunkten des Verbundes gespeichert, während die nicht-diagonalen Elemente auf den Kanten gespeichert werden [NNF02]. Jede Beziehung zwischen den Elementen wird durch eine  $3 \times 3$ -Matrix beschrieben: Der Deformationsvektor mit seinen drei Komponenten  $ux, uy, uz$  und die drei Richtungen der Kraft  $fx, fy, fz$ . Jeder dieser Blöcke beinhaltet die proportionalen Koeffizienten der Kraft des  $j$ -ten Elements zur Verschiebung des  $i$ -ten Elementes. Die Vektoren  $u$  und  $f$  aus der Gleichung  $\mathcal{K}u = f$  sind Vektoren der Länge  $3n$  und haben

die Struktur  $(ux_1, uy_1, uz_1, ux_2, uy_2, uz_2\dots)$  und  $(fx_1, fy_1, fz_1, fx_2, fy_2, fz_2\dots)$ . Für Knoten, die nicht benachbart sind, sind diese Blöcke gleich Null.

**Ankerpunkte löschen.** Ein Vorberechnungsprogramm erstellt in den Modellen Anker-elemente, welche in der Deformationsumgebung bisher als Interaktionselemente fungierten. Diese Einträge müssen bearbeitet werden, damit die Matrix später invertiert werden kann. Die original Matrix  $\mathcal{K}$  ist degeneriert und damit nicht invertierbar, da in der Gleichung  $\mathcal{K}u = f$  der Deformationsvektor  $u$  existiert, aber die Kraft  $f$  gleich Null ist. Diese *Deformationen* sind Translationen und Rotationen des gesamten Objekts. Um diese Verschiebungen zu verhindern, werden Regionen des Körpers in einem Teilraum fixiert. Dieser Teilraum kann verschoben und rotiert werden, ohne die Deformation des Objekts zu beeinflussen.

Um die Matrix invertieren zu können, müsste in diesen fixen Punkten die Verschiebung  $u$  auf Null gesetzt werden. Dies würde nicht bedeuten, dass  $f$  in diesen Punkten auch gleich Null wäre. Da die Anzahl der Unbekannten durch das setzen von  $u = 0$  reduziert würde, müsste die Anzahl der nun unabhängigen Kräfte in  $f$  auch verringert werden. Hierzu würden alle Spalten und Zeilen der fixierten Elemente gelöscht. So würden die gelöschten Spalten mit  $u = 0$  multipliziert und hätten keine Bedeutung mehr. Die gelöschten Zeilen ergäben eine Gleichung, welche die Reaktionskräfte in den fixierten Punkten definieren würde.

Die Matrix muss zur weiteren Berechnung invertiert werden. Hierfür müssen bestimmte Randbedingungen erfüllt sein, damit die Matrix nicht degeneriert ist: Translation und infinitesimale Rotationen müssen verhindert werden. Hierzu werden ein oder zwei Regionen des Objekts im Raum fixiert. Diese lassen sich durch Deformation nicht beeinflussen. So kann der Benutzer zwar die Objekte beliebig im Raum verschieben, sämtliche Deformationen des Objekts orientieren sich aber an diesen fixen Interaktionselementen.

**Oberflächenpunkte extrahieren.** Für die online-Deformation sind nur diejenigen Punkte zu Darstellung nötig, auf die Kraft ausgeübt werden kann. So müssen nur diese Punkte gespeichert werden, was das Einlesen beim Start und während der Ausführung des Simulators beschleunigt. Die Information, bei welchen Punkten es sich um Oberflächenelemente handelt, wird in eine Datei geschrieben. Diese wird ausgelesen und so ausgewertet, dass aus der invertierten Matrix  $\mathcal{K}^{-1}$ , die keine Interaktionselemente mehr enthält, eine Submatrix gebildet wird, welche nur noch Oberflächenelemente beinhaltet.

## 6 Ergebnisse

Bei der Berechnung der inneren Gitterstruktur mit der zugehörigen Information des nächsten Oberflächenpunktes können beliebig geformte Objekte bearbeitet werden (vgl. Abb. 5). Auf dieser Grundlage kann die Kollisionserkennung und die daraus resultierende Deformation in Echtzeit berechnet werden (vgl. Abb. 6).

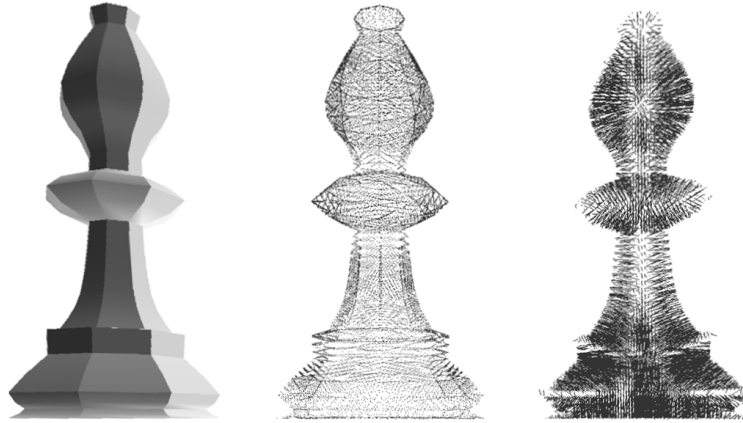


Abbildung 5: Ein konkaves Objekt, hier ein Schach-Läufer, zur Berechnung der Gitterpunktinformation. Links das Originalobjekt, mittig die Oberflächenpunkte, welche den Gitterpunkten am nächsten sind. In der rechten Abbildung sind Verbindungslinien zwischen den Gitterpunkten, die innerhalb des Objekts liegen und den Oberflächenpunkten dargestellt, auf welche bei der Kollision projiziert wird.

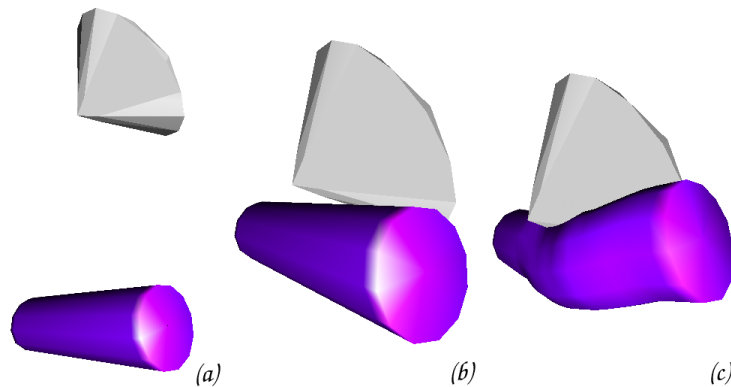


Abbildung 6: Kollisionserkennung und Deformation für ein aus 324 Elementen bestehendes flexibles Objekt und ein auf 8064 Gitterpunkten aufsetzendes starres Objekt. a) Kein Kontakt und Objekt außerhalb der Bounding Box: 75 fps, b) Kein Kontakt und Objekt innerhalb der Bounding Box: 75 fps, c) Kontakt und Deformation: 75 fps.

## 7 Zusammenfassung und Ausblick

In diesem Paper wurde ein echtzeitfähiges System zur Kollisionserkennung und Deformation vorgestellt. Anhand von vorberechneten Gitterpunktinformationen findet zunächst eine Kollisionserkennung statt. Im Falle einer Berührung wird die Eindringtiefe über alle Oberflächenpunkte bestimmt und die Kräfte errechnet, die auf das flexible Objekt ausgeübt werden. Sowohl das flexible als auch das starre Objekt können von beliebiger Form sein. Das System kann sehr hoch auflösende starre Objekte verarbeiten (über 20.000 Punkte), was zu einer sehr genauen Bestimmung des Oberflächenpunktes führt, auf den das flexible Objekt bei Kollision abgebildet wird. Der bei der Interpolation entstehende Fehler ist sehr gering. Die Anzahl der Punkte des flexiblen Objekts ist speicherbedingt limitiert. Bei einer maximal vorzuberechnenden Auflösung von 800 Knotenpunkten ist das System weiterhin echtzeitfähig.

Mit dem vorgestellten Verfahren lässt sich ein für die Operationssimulation wichtiges Objektverhalten in Echtzeit simulieren. Durch die Festlegung auf ein System, welches auf Vorbereitung beruht, können allerdings keine Topologieänderungen simuliert werden.

Die in dieser Arbeit erstellten Inhalte können sinnvoll durch das Ergänzen von einigen Ansätzen erweitert werden. So wäre eine Lösung des Speicherplatzproblems bei den Rechnungen mit der Steifigkeitsmatrix  $\mathcal{K}$  von großem Vorteil für die Verwendung von größeren Objektmodellen. Ausserdem würde eine Kollisionserkennung, die auch Kanten und Flächen berücksichtigt, genauere Kollisions- und Deformationsergebnisse hervorbringen. Für die Berechnung von Objektdurchdringung könnte eine Pfadrückverfolgung notwendig sein.

## Literatur

[ZT00] Zienkiewicz O.C., Taylor R.L.: *The Finite Element Method*, vol. 1, Edition 5, Butterworth-Heinemann, Oxford, 2000.

[net-lexikon] *Das Net-Lexikon*, [www.net-lexikon.de](http://www.net-lexikon.de)

[BNC96] Bro-Nielsen M. und Cotin S.: *Real-Time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation* Eurographics'96, Vol.15, No.3, C57-C66, 1996.

[KNN03] Klimenko S., Nikitina L., Nikitin I.: *Flexible Materials in Avango<sup>TM</sup> Virtual Environment Framework*, Computer Graphics International, 2003. Proceedings, pp. 84- 89., 2003.

[LL70] Landau L.D., Lifschitz E.M.: *Theory of Elasticity*, Volume VII of Theoretical Physics, Moscow, Nauka, 1970.

[PDN00] Picibono G., Delingette H., Nicholas H.-A.: *Non-Linear Anisotropic Elasticity for Real-Time Surgery Simulation*, INRIA Research Report 4028, October, 2000.

- [ZC01] Zhuang Y., Canny J.F.: *Real-Time global deformations*, In Algorithmic and Computational Robotics, pp97-107, Natrick MA, 2001.
- [HFL00] Hirota G., Fisher S., Lin M.: *Simulation of Non-penetrating Elastic Bodies Using Distance Fields*, UNC Technical Report TR00-018, 2000.
- [MPT99] McNeely W.A., Puterbaugh K.D., Troy J.J.: *Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling*, Proceedings of the 26th annual conference on Computer Graphics and interactive techniques, pp: 401 - 408, 1999.
- [RPP01] Renz M., Preusche C., Pötke M., Kriegel H.-P., Hirzinger G.: *Stable Haptic Interaction with Virtual Environments Using an Adapted Voxmap-Pointshell Algorithm*, Eurographics Conference, Birmingham, UK, 2001.
- [GSF94] Garcia-Alonso A., Serrano N., Flaquer J.: *Solving the Collision Detection Problem*, IEEE Computer Graphics and Applications, vol. 14, no. 3, pp. 36-43, 1994.
- [KCY93] Kaufman A., Cohen D., Yagle R.: *Volume Graphics*, IEEE Computer, 26(7), pp. 51-64. 1993.
- [Log96] Logan I.P. et al.: *Virtual Environment Knee Arthroscopy Training System*, Society for Computer Simulation, Simulation Series, vol. 28, no. 4, pp. 17-22, 1996.
- [BNi97] Bro-Nielsen M., PhD: *Fast Finite Elements for Surgery Simulation*, HT Medical Inc., Rockville, Maryland, USA; Department of Mathematical Modelling, Technical University of Denmark, Denmark, 1997.
- [NNF02] Nikitin I., Nikitina L., Frolow P., Goebbels G., Goebel M., Klimenko S., Nielson G.M.: *Real-time simulation of elastic objects in Virtual Environments using finite element method and precomputed Green's Functions*, Eighth Eurographics Workshop on Virtual Environments, pp. 047-052, 2002.
- [Wie04] Wiens C.: *Echtzeitdeformation und Kollisionserkennung zur virtuellen Operations-simulation*, Diplomarbeit, Universität Koblenz, 2004.
- [JP99] James D., Pai D.: *Artdefo - Accurate Real Time Deformable Objects*, Computer Graphics, vol. 33, pp.65-72, 1999.
- [MMP01] Meier U., Monserrat C., Parr N.-C., Garcia F.J., Gil J.A.: *Realtime Simulation of Minimally Invasive Surgery with Cutting Based on Boundary Element Methods*, Lecture Notes in Computer Science, V.2208, p.1263, 2001.
- [Kue03] Kürten S.: *Simulation von elastischem Gewebeverhalten unter Echtzeitsapekten* Diplomarbeit, Universität Bonn, 2003.