Impulsbasierte Dynamiksimulation von Mehrkörpersystemen in der virtuellen Realität

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Jan Bender

aus Karlsruhe

Tag der mündlichen Prüfung:	7. Februar 2007
Erster Gutachter:	Prof. Dr. rer. nat. A. Schmitt
Zweiter Gutachter:	Prof. Dr.-Ing. R. Dillmann

Danksagung

Diese Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Betriebs- und Dialogsysteme der Fakultät für Informatik an der Universität Karlsruhe entstanden.

An dieser Stelle möchte ich Herrn Prof. Dr. A. Schmitt für die Betreuung meiner Arbeit herzlich danken. Außerdem danke ich ihm dafür, dass er immer Zeit für mich hatte und mich von Anfang an bei meiner Forschung unterstützt hat. Die unzähligen fachlichen Diskussionen mit ihm haben mir bei meiner Arbeit sehr geholfen.

Herrn Prof. Dr. R. Dillmann danke ich für die spontane Übernahme des Korreferats und für die hilfreichen Anregungen zur Gestaltung meiner Dissertation.

Bei meinen Kollegen bedanke ich mich für die gute Atmosphäre am Institut und die fachlichen Gespräche. Matthias Baas, Dieter Finkenzeller und Bertrand Klimmek möchte ich außerdem für die Durchsicht und die Korrekturvorschläge meiner Arbeit danken.

Ein ganz besonderer Dank geht an meine Mutter Beate Bender, die mir meine Ausbildung ermöglicht und mich bei meiner Promotion immer unterstützt hat. Darüber hinaus waren ihre Korrekturvorschläge eine große Hilfe.

Schließlich möchte ich mich bei meiner Verlobten Kirsten Lehnig ganz herzlich für das Korrekturlesen und die Verbesserungsvorschläge sowie für ihre unendliche Geduld mit mir bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Die dynamische Simulation von Mehrkörpersystemen hat bereits viele Anwendungsbereiche, wie z. B. die Simulation von Robotern oder die Erstellung von realistischen Animationsfilmen. Die Anforderungen an ein Simulationsverfahren hängen von dem jeweiligen Anwendungsgebiet ab. Bei der Simulation eines Roboters ist es wichtig, dass die Ergebnisse möglichst genau sind. Genaue Simulationsergebnisse erlauben Rückschlüsse auf das Verhalten eines realen Roboters. Die dynamische Simulation bietet dann die Möglichkeit, das Verhalten und die Eigenschaften eines Roboters zu untersuchen, bevor dieser gebaut wird. Auf diese Weise können Kosten bei der Entwicklung eingespart werden. Im Bereich der Computeranimation reicht es meistens aus, wenn die Ergebnisse der dynamischen Simulation für den Betrachter plausibel sind. Die Geschwindigkeit des verwendeten Verfahrens ist in diesem Bereich wesentlich wichtiger als die Genauigkeit.

Die Dynamiksimulation gewinnt jedoch auch im Bereich der virtuellen Realität immer mehr an Bedeutung. Der Begriff der virtuellen Realität bezeichnet ein computergeneriertes, dreidimensionales Modell, das eine reale Umgebung widerspiegelt. Mit Hilfe der Dynamiksimulation können physikalische Gesetze in die virtuelle Welt abgebildet werden. Dadurch kann ein Benutzer intuitiv mit der virtuellen Umgebung interagieren und der Grad der Immersion des Benutzers erhöht sich. Außerdem können komplexe Maschinen in dieser virtuellen Umgebung simuliert und ihr Verhalten untersucht werden.

1.1 Zielsetzung

Der Anwendungsbereich der virtuellen Realität stellt besondere Anforderungen an ein Verfahren für die dynamische Simulation von Mehrkörpersystemen. Diese werden im Folgenden näher erläutert:

- **Geschwindigkeit:** Sowohl einfache als auch komplexe Modelle müssen in der virtuellen Realität in Echtzeit simuliert werden können. Andernfalls gerät die Darstellung der virtuellen Welt ins Stocken und der realistische Eindruck geht für den Benutzer verloren. Daher benötigt man ein Verfahren, das sehr schnell arbeitet.

- **Genauigkeit:** Wenn das Verhalten eines realen Modells in der virtuellen Welt untersucht werden soll, wird ein äquivalentes virtuelles Modell benötigt. Die Abbildung der physikalischen Eigenschaften und des dynamischen Verhaltens dieses Modells in die virtuelle Welt ist die Aufgabe der Dynamiksimulation. Die Ergebnisse einer solchen Untersuchung lassen allerdings nur dann Rückschlüsse auf die Realität zu, wenn die Simulationsergebnisse entsprechend genau sind. Aus diesem Grund ist es wichtig, dass der Grad der Genauigkeit einer Simulation vorhersagbar ist.

- **Verschiedene Gelenkarten:** Das dynamische Simulationsverfahren soll verschiedene Arten von mechanischen Gelenken unterstützen. Ein mechanisches Gelenk verbindet zwei Körper und reduziert deren Freiheitsgrade. Beispiele für solche Gelenke sind Kugel-, Dreh-, Kardan- und Schienengelenke. Außer den einfachen mechanischen Gelenken werden zusätzlich Gelenke benötigt, die Kräfte auf die Körper ausüben, wie z. B. Motoren und Federn. Schließlich gibt es noch spezielle Gelenktypen, die nur die Geschwindigkeiten der Körper beeinflussen. Gelenke dieser Art können zur Interaktion verwendet werden und sind daher in einer virtuellen Umgebung sehr nützlich.

- **Behandlung von Kollisionen:** Wenn zwei Körper miteinander kollidieren oder permanent Kontakt miteinander haben, muss die dynamische Simulation eine Durchdringung der Körper verhindern. Im Fall einer Kollision muss außerdem ein Rückstoß der Körper simuliert werden. Falls zwei Körper Kontakt miteinander haben, tritt entweder dynamische oder statische Reibung auf. Daher muss Reibung bei der Behandlung von Kollisionen und Kontakten berücksichtigt werden.

- **Ergebnisse zu jeder Zeit:** Insbesondere in Anwendungen der virtuellen Realität kann die Einhaltung einer bestimmten Bildwiederholrate wichtiger sein als die Genauigkeit der Simulationsergebnisse. Eine Anwendung soll in einem solchen Fall nicht auf die Ergebnisse der dynamischen Simulation warten müssen, sondern die Simulation jederzeit abbrechen können und ein vorläufiges Ergebnis erhalten. Ein Algorithmus, der diese Anforderung erfüllt, bezeichnet man auch als *Anytime-Algorithmus*.

- **Stabilität:** Die dynamische Simulation eines Modells muss über einen längeren Zeitraum stabil laufen. Die Einwirkung von äußeren Kräften, wie z. B. bei der Interaktion des Benutzers mit dem Modell, darf das dynamische Modell

nicht zerstören. Wenn ein Simulationsschritt vorzeitig abgebrochen werden muss, um sofort ein Ergebnis zu bekommen, ist die Erhaltung der Stabilität besonders wichtig.

- **Einfache Implementierung und Erweiterbarkeit:** Eine weitere Anforderung an das Verfahren ist, dass es möglichst einfach zu implementieren sein soll. Wenn das Verfahren diese Anforderung erfüllt, dann kann es problemlos in bestehende Anwendungen der virtuellen Realität integriert werden. Außerdem soll das Verfahren erweiterbar sein, damit z. B. neue Gelenktypen hinzugefügt werden können.

Bisher existiert kein Verfahren, das alle genannten Anforderungen erfüllt. Das Ziel dieser Arbeit ist es daher, ein solches Verfahren für die dynamische Simulation von Mehrkörpersystemen zu entwickeln. Um dies zu erreichen, wird ein neuer impulsbasierter Ansatz verwendet, der in dieser Arbeit vorgestellt wird.

1.2 Aufbau der Arbeit

Kapitel 2 gibt einen Überblick über die verwandten Arbeiten. Es werden dabei Arbeiten zu den folgenden Themen näher betrachtet: dynamische Simulation von Gelenken, Kollisionserkennung und Kollisionsauflösung.

In Kapitel 3 wird zunächst die Simulation eines freien Systems erläutert. Anschließend werden Gelenke durch Zwangsbedingungen für die Körper definiert. Es wird gezeigt, wie ein Gelenk ausschließlich mit Hilfe von Impulsen dynamisch simuliert werden kann. Außerdem wird erklärt, wie man die benötigten Impulse für viele verschiedene Gelenktypen in einer einheitlichen Form berechnet. In Modellen mit mehreren Gelenken gibt es im Allgemeinen Abhängigkeiten zwischen den Gelenken. Es werden zwei verschiedene Verfahren mit unterschiedlichen Eigenschaften vorgestellt, um die Impulse für solche Modelle zu bestimmen. Diese beiden Verfahren werden dann anhand von Messergebnissen miteinander verglichen. Danach werden Verfahren hergeleitet, mit denen ein höherer Genauigkeitsgrad erreicht werden kann. Am Ende des Kapitels folgt ein ausführlicher Vergleich mit Verfahren der klassischen Mechanik.

Das Kapitel 4 beschäftigt sich mit der Kollisionserkennung. Zunächst werden die Anforderungen für ein Kollisionserkennungsverfahren in der dynamischen Simulation definiert. Anschließend werden verschiedene, bekannte Verfahren hinsichtlich dieser Anforderungen miteinander verglichen. Es werden zwei geeignete Verfahren ausgewählt und so erweitert, dass sie alle Anforderungen erfüllen. Am Ende des Kapitels werden Messergebnisse zu beiden Verfahren präsentiert.

Die Auflösung von Kollisionen und bleibenden Kontakten mit Reibung wird in Kapitel 5 behandelt. Kollisionen und Kontakte werden mit Hilfe von Zwangsbe-

dingungen aufgelöst. Dadurch sind exakte Ergebnisse möglich. Die Bedingungen werden durch die iterative Berechnung von Impulsen erfüllt, da Impulse sehr einfach und schnell bestimmt werden können. Anschließend werden Messergebnisse zur Genauigkeit und Geschwindigkeit der Kollisionsauflösung präsentiert. Am Ende des Kapitels wird ein Vergleich des Verfahrens mit anderen Methoden anhand der Anforderungen aus Abschnitt 1.1 durchgeführt.

In Kapitel 6 wird die Simulationsumgebung vorgestellt, die während dieser Arbeit entwickelt wurde. Diese Umgebung besteht aus einem Werkzeug zur einfachen Erstellung von dynamischen Modellen und dem Simulator. Im Simulator wurden alle beschriebenen Verfahren umgesetzt und getestet.

Das letzte Kapitel fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf mögliche Erweiterungen.

Kapitel 2

Verwandte Arbeiten

Die Bewegung eines freien Starrkörpers, der durch äußere Kräfte beschleunigt wird, kann mit Hilfe von Newtons zweitem Gesetz

$$\mathbf{F} = m\,\mathbf{a}$$

und der Euler-Gleichung (siehe Anhang D.2) simuliert werden. Ein Mehrkörpersystem ist zu einem Zeitpunkt definiert durch die Lage $\mathbf{x}$ aller Körper und deren Geschwindigkeiten $\mathbf{v}$. Der Vektor $\mathbf{x}$ beschreibt für jeden Körper die aktuelle Position und Rotation und der Vektor $\mathbf{v}$ die Geschwindigkeiten der Schwerpunkte und die Winkelgeschwindigkeiten. Die Bewegungsfreiheit eines Körpers kann durch Zwangsbedingungen eingeschränkt sein. Es existieren zwei verschiedene Arten von Zwangsbedingungen:

- **Holonome Zwangsbedingungen** beschränken die Bewegung der Körper in Form einer impliziten Funktion

  $$\mathbf{C}(\mathbf{x}, t) = 0. \tag{2.1}$$

  Diese Funktion darf von der aktuellen Lage $\mathbf{x}$ der Körper und von der Zeit t abhängig sein.

  Ein Beispiel für eine holonome Bedingung ist die Funktion $\mathbf{C}(\mathbf{s}_1, \mathbf{s}_2) = \mathbf{s}_1 - \mathbf{s}_2 = \mathbf{0}$. Diese Zwangsbedingung fordert, dass der Abstand zwischen den Schwerpunkten $\mathbf{s}_1$ und $\mathbf{s}_2$ von zwei Körpern immer Null ist.

  Die holonomen Bedingungen haben die wichtige Eigenschaft, dass sie die Freiheitsgrade des Mehrkörpersystems reduzieren. Jede skalare Gleichung entfernt dabei einen Freiheitsgrad des Systems. Im obigen Beispiel werden demnach drei Freiheitsgrade durch die Funktion $\mathbf{C}(\mathbf{s}_1, \mathbf{s}_2)$ entfernt. Durch holonome Zwangsbedingungen lassen sich alle Arten von mechanischen Gelenken beschreiben.

- **Nichtholonome Zwangsbedingungen** sind alle Bedingungen, die sich nicht durch eine Funktion in der Form von Gleichung 2.1 beschreiben lassen. Die beiden wichtigsten Vertreter dieser Klasse sind Bedingungen, die zusätzlich von den Geschwindigkeiten der Körper abhängen, und Bedingungen, die sich nur in Form einer Ungleichung formulieren lassen. Die erste Art kann durch eine implizite Funktion der Form

  $$\mathbf{C}(\mathbf{x}, \mathbf{v}, t) = 0$$

  beschrieben werden.

  Ein Beispiel für eine solche Funktion ist $\mathbf{C}(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1 - \mathbf{v}_2 = \mathbf{0}$. Diese nichtholonome Bedingung sorgt dafür, dass die Schwerpunktgeschwindigkeiten $\mathbf{v}_1$ und $\mathbf{v}_2$ von zwei Körpern immer gleich sind.

  Durch Funktionen in Form von Ungleichungen lassen sich Zwangsbedingungen formulieren, die verhindern, dass sich Körper gegenseitig durchdringen. Daher werden solche Bedingungen bei der Auflösung von Kollisionen eingesetzt.

  Bei einer nichtholonomen Bedingung existiert kein direkter Zusammenhang zwischen ihrer Gleichung bzw. Ungleichung und den Freiheitsgraden des Systems.

Eine Zwangsbedingung, die nicht explizit von der Zeit abhängt, bezeichnet man als *skleronom*. Andernfalls nennt man sie *rheonom*.

Die dynamische Simulation von Mehrkörpersystemen besteht im Allgemeinen wegen der unterschiedlichen Zwangsbedingungen aus drei Teilen. Der erste Teil ist die Simulation von Starrkörpern, die durch Gelenke miteinander verbunden sind. In diesem Teil werden alle holonomen Zwangsbedingungen und oft auch Bedingungen für die Geschwindigkeiten behandelt. Nichtholonome Zwangsbedingungen in Form einer Ungleichung werden in einem Mehrkörpersystem benötigt, um Kollisionen zwischen Körpern zu simulieren. Die Simulation von Kollisionen erfordert eine Kollisionserkennung und eine Kollisionsauflösung. Dies sind die anderen beiden Teile der dynamischen Simulation. Die Kollisionserkennung wird benötigt, um zu bestimmen, wann und wo eine Kollision stattfindet. Die Aufgabe der Kollisionsauflösung ist, im Fall einer Kollision einen Rückstoß der betroffenen Körper mit Reibung zu simulieren.

2.1 Simulation von Gelenken

Die dynamische Simulation von Mehrkörpersystemen mit Gelenken wird bereits seit vielen Jahren erforscht. Es existieren verschiedene Ansätze, um die Zwangsbedingungen der Gelenke zu erfüllen. Drei der wichtigsten Verfahren zur Behandlung

von Zwangsbedingungen sind die Penalty-Methode, die Verwendung von reduzierten Koordinaten und die Lagrange-Faktoren-Methode. Diese Verfahren werden im Folgenden vorgestellt.

2.1.1 Penalty-Methode

Die Penalty-Methode kann Zwangsbedingungen in Form einer impliziten Funktion behandeln. Diese Funktion darf abhängig von der Lage der Körper $\mathbf{x}$, den Geschwindigkeiten $\mathbf{v}$ und der Zeit t sein. Wenn eine Zwangsbedingung nicht erfüllt ist, dann wird dem Mehrkörpersystem eine zusätzliche Kraft hinzugefügt, um die Lage bzw. die Geschwindigkeit zu korrigieren. Der Betrag der Kraft hängt davon ab, wie stark die Bedingung verletzt wurde. Außerdem ist sie der Richtung der Verletzung entgegengesetzt.

In [dJB94] und [Wag01] wird diese Kraft für holonome Zwangsbedingungen mit der Gleichung

$$\mathbf{F}_{Penalty} = -\alpha\,\mathbf{J}^T\,(\Omega^2\,\mathbf{C} + 2\,\Omega\,\mu\,\dot{\mathbf{C}} + \ddot{\mathbf{C}})$$

berechnet, wobei $\mathbf{J}$ die Jacobi-Matrix der Bedingung $\mathbf{C}$ ist. Die Werte α, Ω und μ sind konstante Parameter. Die Kraft $\mathbf{F}_{Penalty}$ ist genau dann ungleich Null, wenn die Bedingung $\mathbf{C}$ nicht erfüllt wird. In diesem Fall wird sie auf die entsprechenden Körper angewendet. Die Ableitungen der Zwangsbedingung nach der Zeit dienen der Stabilisierung des Verfahrens. Die konstanten Parameter α, Ω und μ können als die Stärke, die Eigenfrequenz und die Dämpfung einer Feder interpretiert werden. Nichtholonome Zwangsbedingungen der Form $\mathbf{C}(\mathbf{x}, \mathbf{v}, t) = 0$ können ebenfalls mit der Penalty-Methode simuliert werden. Die Kraft für eine solche Bedingung wird mit der Gleichung

$$\mathbf{F}_{Penalty} = -\alpha\,\left(\frac{d\mathbf{C}}{d\mathbf{v}}\right)^T\,(\mu\,\mathbf{C} + \dot{\mathbf{C}})$$

berechnet [dJB94, Wag01].

In einem Simulationsschritt mit der Penalty-Methode werden zunächst alle Kräfte $\mathbf{F}_{Penalty}$ für die Zwangsbedingungen berechnet. Anschließend werden diese zu den externen Kräften, die auf die Körper wirken, addiert. Schließlich kann die Bewegungsgleichung für die Körper gelöst werden. Die Penalty-Methode ist sehr einfach zu implementieren und hat einen geringen Berechnungsaufwand. Allerdings werden die Zwangsbedingungen mit dieser Methode nur näherungsweise erfüllt. Eine exakte Lösung wäre nur mit einem unendlich großen Wert α möglich. Die Simulation hängt stark von den Parametern α, Ω und μ ab, aber eine geeignete Wahl dieser Parameter ist schwierig. Der Wert von α muss sehr groß sein, damit die Zwangsbedingungen möglichst exakt erfüllt werden. Dies kann allerdings zu sogenannten *steifen Differentialgleichungen* führen, welche nur mit sehr kleinen Schrittweiten oder mit speziellen Verfahren stabil gelöst werden können [PFTV92]. Diese spe-

ziellen Verfahren sind wesentlich ineffizienter und ungenauer als die Standardverfahren [Wag01]. Deshalb sollten steife Differentialgleichungen vermieden werden.

2.1.2 Reduzierte Koordinaten

Durch holonome Zwangsbedingungen werden die Freiheitsgrade eines Mehrkörpersystems permanent reduziert. Die Methode der reduzierten Koordinaten nutzt diese Eigenschaft der holonomen Bedingungen aus. Wenn in einem freien Mehrkörpersystem mit n Freiheitsgraden m voneinander unabhängige Zwangsbedingungen definiert werden, dann reduziert sich die Anzahl der Freiheitsgrade auf $n-m$. Das bedeutet, dass der Zustand des Systems durch $n-m$ unabhängige Koordinaten q_i bestimmt ist. Diese unabhängigen Koordinaten werden auch als generalisierte oder reduzierte Koordinaten bezeichnet. Die n Koordinaten des freien Systems können als Funktion der reduzierten Koordinaten

$$x_i = x_i(q_1, \ldots, q_{n-m}) \tag{2.2}$$

mit $i \in [1, \ldots, n]$ beschrieben werden.

Dies lässt sich anhand eines Beispiels verdeutlichen. Abbildung 2.1 zeigt ein Partikel mit der Position $\mathbf{x} \in \mathbb{R}^2$, das sich um den Ursprung des Koordinatensystems dreht. Die Kreisbahn, auf der es sich bewegt, hat den Radius r. Das Partikel hat

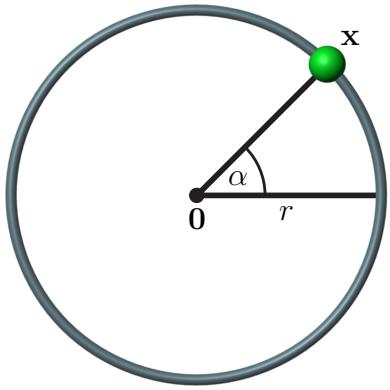


Abbildung 2.1: Partikel, das sich auf einer Kreisbahn bewegt

nur einen Freiheitsgrad. Demnach kann sein Bewegungszustand durch einen einzigen Parameter beschrieben werden. Eine geeignete Wahl für diesen Parameter ist der Winkel α zwischen der x-Achse und dem Vektor vom Nullpunkt zu $\mathbf{x}$. Die Position des Partikels wird folgendermaßen durch den Winkel bestimmt:

$$\mathbf{x} = r \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}. \tag{2.3}$$

Der Winkel α ist durch keine Bedingung beschränkt und damit frei. Die Gleichung 2.3 definiert bei freier Wahl des Parameters α alle zulässigen Positionen $\mathbf{x}$ für das Partikel.

Im Allgemeinen besteht die Methode der reduzierten Koordinaten aus den folgenden Schritten. Als Erstes müssen $n - m$ unabhängige Koordinaten q_i gefunden werden, die den Bewegungszustand des Mehrkörpersystems eindeutig beschreiben. Anschließend werden die allgemeinen Koordinaten x_i mit Gleichung 2.2 durch die reduzierten Koordinaten q_i ersetzt. Danach werden die $n-m$ Bewegungsgleichungen bezüglich der Koordinaten q_i aufgestellt. Das daraus resultierende System von gewöhnlichen Differentialgleichungen wird mit Hilfe von numerischen Integrationsverfahren gelöst.

Die Bewegungsgleichungen können mit dem Lagrange-Formalismus aufgestellt werden. Dieser wird z. B. in [GPS06] beschrieben. Die Lagrange-Funktion $L = E_{kin} - E_{pot}$ beschreibt die Differenz der gesamten kinetischen und potentiellen Energie des Systems. Wenn diese Funktion in die Euler-Lagrange-Differentialgleichungen eingesetzt wird, erhält man die Lagrange-Gleichungen

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0, \qquad \text{für } i = 1, \ldots, n-m.$$

Diese Gleichungen bilden ein System gewöhnlicher Differentialgleichungen zweiter Ordnung. Diese Vorgehensweise ist auch als die Lagrangesche Methode zweiter Art bekannt und hat einen Berechnungsaufwand von $O(m^4)$.

Das rekursive Newton-Euler-Verfahren hat einen geringeren Aufwand und ist daher für die dynamische Simulation von Modellen mit Baumstruktur besser geeignet [FO00]. Das Verfahren arbeitet in zwei Phasen. In der ersten Phase werden die kinematischen Parameter der Körper abhängig von den äußeren Kräften bestimmt. Die Parameter eines Körpers im Baum hängen nur von denen seines Vorgängers ab. Der Baum wird daher von der Wurzel bis hin zu den Blättern durchlaufen und dabei werden alle Parameter berechnet. In der zweiten Phase wird der Baum rückwärts durchlaufen. Bei diesem Durchlauf werden für alle Körper die inneren Kräfte und Momente bestimmt, die jeweils von denen des nachfolgenden Körpers im Baum abhängen. Das Newton-Euler-Verfahren wird in [FO00] ausführlich beschrieben. Außerdem werden in dieser Arbeit noch weitere Verfahren vorgestellt, die auf dem Prinzip des Newton-Euler-Verfahrens basieren. Das schnellste dieser Verfahren ist das von Roy Featherstone aus [Fea87], das mit einem linearen Zeitaufwand arbeitet.

Die Methode der reduzierten Koordinaten hat den Vorteil, dass die Zwangsbedingungen der Gelenke durch die Parametrisierung nicht explizit in den Bewegungsgleichungen auftauchen. Jeder Zustand des Systems, der in den reduzierten Koordinaten angegeben wird, erfüllt automatisch alle Zwangsbedingungen. Allgemeine Koordinaten, die die Bedingungen erfüllen, werden mit der Gleichung 2.2 aus den reduzierten Koordinaten bestimmt. Die Methode der reduzierten Koordinaten ist

ein effizientes Simulationsverfahren, da anstatt n nur $n-m$ Differentialgleichungen gelöst werden müssen.

Für ein System mit beliebigen Zwangsbedingungen ist es schwer, eine Parametrisierung zu finden. Wenn eine solche Parametrisierung gefunden werden kann, dann ist der Zeitaufwand für die Bestimmung der Beschleunigung der m reduzierten Koordinaten $O(m^3)$ [Bar96]. Mehrkörpersysteme, die keine geschlossenen kinematischen Ketten enthalten, sind einfach parametrisierbar und die Beschleunigung der m Koordinaten kann in linearer Zeit mit dem Verfahren von Roy Featherstone [Fea87] berechnet werden. Allerdings ist die Implementierung dieses Verfahrens kompliziert [Mir96b]. Ein weiterer Nachteil des Verfahrens ist, dass es ausschließlich holonome Zwangsbedingungen unterstützt. Bei der Methode der reduzierten Koordinaten muss die Parametrisierung jedes Mal neu bestimmt werden, wenn sich die Anzahl der Zwangsbedingungen während der Simulation ändert. Außerdem können geschlossene kinematische Ketten nur mit speziellen Verfahren simuliert werden [FO00]. Die Probleme, die bei der Simulation von geschlossenen Ketten auftreten, werden in [Wit77] ausführlich besprochen.

2.1.3 Lagrange-Faktoren-Methode

Die Lagrange-Faktoren-Methode simuliert Zwangsbedingungen mit Hilfe von internen Kräften, die die Bewegungsfreiheit der Körper einschränken. Diese Methode wird häufig für die dynamische Simulation eingesetzt, da sie flexibler und einfacher zu implementieren ist als die Methode der reduzierten Koordinaten und da sie genauer ist als die Penalty-Methode. Eine der ersten Arbeiten in der Computergraphik, die die Lagrange-Faktoren-Methode zur Simulation verbundener Starrkörper eingesetzt hat, ist die von Ronen Barzel und Alan H. Barr [BB88]. Diese Arbeit beschreibt die physikalisch basierte Modellierung mit Zwangsbedingungen. Der Ansatz mit internen Kräften ist dem impulsbasierten Ansatz dieser Arbeit am ähnlichsten. Daher wird die Lagrange-Faktoren-Methode im Folgenden genauer betrachtet.

Der Zustand eines Starrkörpers ist durch einen Lagevektor $\mathbf{x}$ und einen Geschwindigkeitsvektor $\mathbf{v}$ definiert. Der Lagevektor beinhaltet die Position des Schwerpunkts und eine Quaternion zur Beschreibung der Rotation. Daher hat $\mathbf{x}$ die Dimension sieben. Der Geschwindigkeitsvektor $\mathbf{v}$ besteht aus der Schwerpunktgeschwindigkeit und der Winkelgeschwindigkeit des Körpers und hat die Dimension sechs. Alternativ zum Vektor $\mathbf{v}$ kann der Bewegungszustand auch durch einen Vektor $\mathbf{p}$ beschrieben werden, der sich aus dem Impuls und dem Drehimpuls des Körpers zusammensetzt. Die Beschreibung mit dem Vektor $\mathbf{p}$ hat einen entscheidenden Vorteil. Die Winkelgeschwindigkeit eines Körpers kann sich auch dann ändern, wenn keine äußeren Kräfte wirken. Der Drehimpuls des Körpers ist dagegen in diesem Fall konstant. Alle externen Kräfte und Drehmomente, die auf einen

Körper wirken, können aufsummiert und in einem sechsdimensionalen Vektor $\mathbf{F}$ zusammengefasst werden. Die Wirkung einer Kraft auf einen Körper wird durch die Matrix

$$\mathbf{M}(t) = \begin{pmatrix} m\,\mathbf{E}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{J}(t) \end{pmatrix} \in \mathbb{R}^{6\times 6} \tag{2.4}$$

bestimmt, die die Masseneigenschaften des Körpers widerspiegelt. Diese Matrix setzt sich aus der Masse m, die mit der dreidimensionalen Einheitsmatrix $\mathbf{E}_3$ multipliziert wird, und dem Trägheitstensor $\mathbf{J}$ in Weltkoordinaten zusammen. Die Vektoren $\mathbf{x}$, $\mathbf{v}$, $\mathbf{p}$ und $\mathbf{F}$ für ein System mit n Körpern können jeweils zu Vektoren der Dimension $6n$ bzw. $7n$ zusammengefasst werden. Analog kann eine Massenmatrix $\mathbf{M} \in \mathbb{R}^{6n\times 6n}$ für das Gesamtsystem mit den Matrizen $\mathbf{M}_i$ der einzelnen Körper auf der Diagonalen gebildet werden.

Mit Newtons zweitem Gesetz kann die Bewegungsgleichung in Form einer gewöhnlichen Differentialgleichung zweiter Ordnung definiert werden:

$$\mathbf{F} = \frac{d^2}{d^2t}(\mathbf{M}(t)\,\mathbf{x}(t)).$$

Diese Gleichung kann in zwei gewöhnliche Differentialgleichungen erster Ordnung umgeformt werden:

$$\begin{aligned} \frac{d}{dt}\mathbf{p}(t) &= \mathbf{F} \\ \frac{d}{dt}\mathbf{x}(t) &= \mathbf{M}^{-1}(t)\,\mathbf{p}(t) = \mathbf{v}(t). \end{aligned}$$

Durch numerische Integration dieser beiden Gleichungen kann ausgehend von einem bekannten Zustand des Systems zum Zeitpunkt t_0 der Zustand zu einem Zeitpunkt $t_0 + h$ bestimmt werden.

Die Lagrange-Faktoren-Methode unterstützt ohne eine Sonderbehandlung holonome Zwangsbedingungen und Bedingungen, die zusätzlich von den Geschwindigkeiten der Körper abhängen. Die Zwangsbedingungen werden mit Hilfe von internen Kräften erfüllt, die die Beschleunigungen der Körper verändern. Gelenke müssen daher als eine Bedingung für die Beschleunigungen der Körper definiert werden. Dies wird erreicht, indem die holonomen Zwangsbedingungen zweimal und die Geschwindigkeitsbedingungen einmal abgeleitet werden. In [Wag01] werden diese Ableitungen ausführlich beschrieben. Die abgeleiteten Bedingungen können in die allgemeine Form

$$\mathbf{J}(\mathbf{x}, \mathbf{v}, t)\,\dot{\mathbf{v}} + \mathbf{c}(\mathbf{x}, \mathbf{v}, t) = 0 \tag{2.5}$$

gebracht werden, wobei $\mathbf{J}(\mathbf{x}, \mathbf{v}, t)$ die sogenannte Jacobimatrix der Bedingung $\mathbf{C}$ ist. Das Problem mit den bisher beschriebenen Zwangsbedingungen ist, dass $\dot{\mathbf{x}} \neq \mathbf{v}$ gilt, da der Lagevektor nicht die gleiche Dimension hat wie der Geschwindigkeitsvektor. Dieses Problem wird in [WGW90] durch die Verwendung von sogenannten

Konnektoren gelöst. Ein Konnektor ist ein Punkt, der fest mit einem Körper verbunden ist. Seine Position ist demnach abhängig von der Lage $\mathbf{x}$ des Körpers, kann aber mit einem dreidimensionalen Vektor beschrieben werden, genau wie die Geschwindigkeit des Konnektors. Die Zwangsbedingungen der Gelenke werden mit Hilfe der Konnektoren beschrieben und sind damit nicht mehr direkt abhängig von dem Zustandsvektor $\mathbf{x}$. In [Wag01] wird dieses Konnektoren-Konzept für Geschwindigkeitsbedingungen erweitert. Um holonome Zwangsbedingungen mit k Konnektoren $\mathbf{a}_i$ in die allgemeine Form zu bringen, werden die Ableitungen der Bedingungen nach den Konnektoren benötigt. Mit den folgenden Gleichungen wird die allgemeine Form bestimmt:

$$J_{i,j} = \sum_{l=1}^{k} \frac{\partial C_i}{\partial \mathbf{a}_l} \frac{\partial \dot{\mathbf{a}}_l}{\partial v_j}, \qquad c_i = \sum_{l=1}^{k} \frac{\partial \dot{C}_i}{\partial \mathbf{a}_l} \dot{\mathbf{a}}_l + \sum_{l=1}^{k} \frac{\partial C_i}{\partial \mathbf{a}_l} \left(\frac{\partial \dot{\mathbf{a}}_l}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{a}}_l}{\partial t} \right) + \frac{\partial \dot{C}_i}{\partial t}.$$

Die allgemeine Form einer Geschwindigkeitsbedingung wird dagegen folgendermaßen ermittelt:

$$J_{i,j} = \sum_{l=1}^{k} \frac{\partial C_i}{\partial \dot{\mathbf{a}}_l} \frac{\partial \dot{\mathbf{a}}_l}{\partial v_j}, \qquad c_i = \sum_{l=1}^{k} \frac{\partial C_i}{\partial \mathbf{a}_l} \dot{\mathbf{a}}_l + \sum_{l=1}^{k} \frac{\partial C_i}{\partial \dot{\mathbf{a}}_l} \left(\frac{\partial \dot{\mathbf{a}}_l}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{a}}_l}{\partial t} \right) + \frac{\partial C_i}{\partial t}.$$

Die benötigten Ableitungsterme werden in [Wag01] für verschiedene Gelenke vorgestellt.

Im folgenden werden zwei Verfahren zur Bestimmung der internen Kräfte der Gelenke präsentiert. Beide Verfahren können geschlossene kinematische Ketten nur durch eine Sonderbehandlung verarbeiten, wie sie z. B. in [Bar96] vorgestellt wird. Im Folgenden werden daher nur Modelle ohne geschlossene Ketten betrachtet.

2.1.3.1 Standardverfahren

Zwangsbedingungen werden mit Hilfe von Zwangskräften $\mathbf{F}_c$ simuliert. Diese werden wie folgt in die Bewegungsgleichung eingefügt:

$$\dot{\mathbf{v}} = \mathbf{M}^{-1}(\mathbf{F}_{ext} + \mathbf{F}_c). \tag{2.6}$$

Setzt man diese Bewegungsgleichung in die Gleichung 2.5 ein, die die allgemeine Form für Zwangsbedingungen beschreibt, dann erhält man das folgende lineare Gleichungssystem:

$$\mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{F}_c = -\mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{F}_{ext} - \mathbf{c}. \tag{2.7}$$

Dieses Gleichungssystem ist in dieser Form nicht lösbar, denn es ist im Allgemeinen unterbestimmt. Berücksichtigt man das Prinzip von d'Alembert, welches in [GPS06] ausführlich behandelt wird, dann bekommt man die folgende Gleichung für die Zwangskräfte:

$$\mathbf{F}_c = \mathbf{J}^T \boldsymbol{\lambda},$$

wobei $\boldsymbol{\lambda}$ die sogenannten Lagrange-Faktoren sind. Die Zwangskräfte wirken demnach immer in die Richtungen, in die das System durch die Zwangsbedingungen eingeschränkt ist, und die Lagrange-Faktoren bestimmen ihre Stärke. Setzt man die Gleichung der Zwangskräfte in die Gleichung 2.7 ein, dann erhält man das Gleichungssystem

$$\begin{aligned} \mathbf{A} &= \mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{J}^T, \quad \mathbf{b} = -\mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{F}_{ext} - \mathbf{c} \\ \mathbf{A}\,\boldsymbol{\lambda} &= \mathbf{b} \end{aligned} \tag{2.8}$$

für die Lagrange-Faktoren. Wenn die Matrix $\mathbf{A}$ regulär ist, kann das System für $\boldsymbol{\lambda}$ gelöst werden. Dadurch lassen sich die Zwangskräfte $\mathbf{F}_c$ bestimmen, die wiederum in Gleichung 2.6 eingesetzt werden. Mit Hilfe numerischer Integration können anschließend die Geschwindigkeiten und Positionen aller Körper für den nächsten Zeitschritt bestimmt werden. Die Berechnung der Lagrange-Faktoren hat einen Berechnungs- und Speicheraufwand von $O(m^3)$, wobei m die Anzahl der Zwangsbedingungen ist.

2.1.3.2 Verfahren von Baraff

David Baraff beschreibt in [Bar96] ein Verfahren, das die Lagrange-Faktoren in linearer Zeit berechnet. Das Gleichungssystem 2.8 zur Bestimmung der Lagrange-Faktoren wird dabei zunächst in die folgende Form gebracht:

$$\underbrace{\begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ -\mathbf{J} & \mathbf{0} \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{b} \end{pmatrix}.$$

Die Matrix $\mathbf{H}$ hat den Vorteil, dass sie immer dünn besetzt ist. Das bedeutet, dass die Matrix nur wenige Einträge hat, die von Null verschieden sind. Durch eine Tiefensuche im Verbindungsgraph des Modells kann die Matrix so umgeformt werden, dass sich die Einträge eines Knotens im Graph immer nach denen seiner Kindknoten in der Matrix befinden. Die Matrix hat nach der Umformung die Eigenschaft, dass bei ihrer Faktorisierung keine weiteren von Null verschiedenen Einträge hinzukommen. Die Faktorisierung ist demnach ebenso dünn besetzt wie die Matrix selbst. Zum Faktorisieren der Matrix wird die sogenannte $\mathbf{LDL}^T$-Faktorisierung verwendet. Dabei ist $\mathbf{L}$ die untere Blockdreiecksmatrix, deren Diagonalelemente Einheitsmatrizen sind, und $\mathbf{D}$ ist die Blockdiagonalmatrix. Wenn bei der Faktorisierung die Struktur der Matrix berücksichtigt wird, kann sie in $O(m)$ durchgeführt werden. Die Lagrange-Faktoren können mit dieser Faktorisierung ebenfalls in $O(m)$ bestimmt werden. Die Geschwindigkeiten und Positionen der Körper werden wie beim Standardverfahren mit Hilfe von numerischer Integration berechnet.

2.1.3.3 Stabilisierung

Die Lagrange-Faktoren-Methode berechnet Zwangskräfte, die verhindern, dass die Zwangsbedingungen durch externe Kräfte verletzt werden. Wenn eine Zwangsbedingung auf eine andere Weise verletzt wird, kann dies mit der Methode nicht korrigiert werden. Eine solche Verletzung der Bedingungen kann z. B. durch numerische Fehler bei den Berechnungen auftreten. Die numerischen Fehler summieren sich in jedem Zeitschritt auf und die Gelenke driften auseinander. Das Problem besteht darin, dass für eine holonome Zwangsbedingung $\mathbf{C}$ gefordert wird, dass die zweite Ableitung Null ist. Dies gilt allerdings allgemein für alle Gleichungen der Form

$$\mathbf{C} = \mathbf{k}_1\, t + \mathbf{k}_2$$

und nicht nur für die zu erfüllende Bedingung $\mathbf{C} = \mathbf{0}$. Das bedeutet, bestehende Positions- und Geschwindigkeitsfehler werden nicht korrigiert. Aus diesem Grund wird bei der Lagrange-Faktoren-Methode ein zusätzliches Stabilisierungsverfahren benötigt.

Am häufigsten wird das Verfahren von Baumgarte eingesetzt, das in [Bau72] beschrieben ist. Bei diesem Verfahren wird die Gleichung $\ddot{\mathbf{C}} = \mathbf{0}$ der holonomen Bedingungen durch

$$\ddot{\mathbf{C}} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} = \mathbf{0}$$

ersetzt, wobei α und β konstante Parameter sind. Dadurch werden die Positions- und Geschwindigkeitsfehler mit einer Gewichtung berücksichtigt. Für Geschwindigkeitsbedingungen wird die Gleichung

$$\dot{\mathbf{C}} + \gamma\mathbf{C} = \mathbf{0}$$

anstatt $\dot{\mathbf{C}} = \mathbf{0}$ verwendet. Der konstante Parameter γ bestimmt dabei, wie stark Geschwindigkeitsfehler berücksichtigt werden.

Die Stabilisierungsterme können durch Addition in die allgemeine Form einer Zwangsbedingung eingefügt werden. Alternativ werden in [WW90] die Stabilisierungsterme als zusätzliche Kräfte in die Bewegungsgleichung eingefügt. Die Bestimmung geeigneter Parameter für die Baumgarte-Stabilisierung ist nicht einfach. Die Probleme bei der Bestimmung der Parameter werden in [ACPR95] diskutiert. Außerdem wird in dieser Arbeit eine verbesserte Stabilisierung vorgeschlagen.

2.1.4 Weitere Verfahren

In [Wag01] wird ein Überblick über weitere Lösungsverfahren gegeben, die bisher noch nicht erwähnt wurden. Ein neueres Verfahren wird in [WTF06] von Rachel Weinstein et al. vorgestellt. Dieses Verfahren verwendet Impulse, um Zwangsbedingungen von Gelenken und Geschwindigkeitsbedingungen zu lösen. Die Vorgehensweise zur Bestimmung der Impulse gleicht der in [BBS03] und [BFS05]. Der

einzige Unterschied ist, dass in [WTF06] eine nichtlineare Gleichung iterativ gelöst wird, um einen Impuls zu bestimmen und in [BBS03] sowie [BFS05] eine lineare Gleichung. Das Verfahren mit linearen Gleichungen hat unter anderem den Vorteil, dass mehrere Impulse gleichzeitig mit einem linearen Gleichungssystem bestimmt werden können.

Ein anderes neues Verfahren wird in [MHHR06] von Matthias Müller et al. präsentiert. Dieses Verfahren verwendet weder Kräfte noch Impulse zum Lösen der Zwangsbedingungen, sondern arbeitet direkt mit den Positionen der Körper. In einem Simulationsschritt werden zunächst alle Positionen für den nächsten Schritt bestimmt. Danach werden diese Positionen so verändert, dass sie die gegebenen Zwangsbedingungen erfüllen. Auf diese Weise können sowohl Gelenke als auch Kollisionen simuliert werden. Allerdings lässt diese Vorgehensweise nur plausible und keine genauen Ergebnisse zu. Für genaue Simulationen ist das Verfahren ungeeignet.

2.2 Kollisionserkennung

In der dynamischen Simulation existiert zu jedem Starrkörper, der mit anderen Körpern kollidieren kann, ein zugehöriges Kollisionsobjekt. Dieses Objekt besteht aus einem dreidimensionalen Modell, das den Starrkörper bei der Kollisionserkennung repräsentiert. Dieses Modell muss vor der Durchführung einer Kollisionserkennung an die aktuelle Position des Körpers transformiert werden. Es wird benötigt, um Kollisionen zwischen den Körpern zu erkennen und die Kontaktgeometrie zu bestimmen. Das Verfahren zur Kollisionserkennung ist stark abhängig von der Art der Modelle, die verwendet werden. Eine Einteilung der Modelle in verschiedene Klassen und speziell angepasste Verfahren für diese Klassen werden im Abschnitt 2.2.1 vorgestellt.

Die meisten Verfahren für die Kollisionserkennung arbeiten in zwei Phasen. Es ist sehr zeitaufwendig, für jedes Paar von Kollisionsobjekten zu überprüfen, ob eine Kollision vorliegt. Dieses Vorgehen hat einen quadratischen Aufwand. Deshalb werden in der ersten Phase, der sogenannten *Broad Phase*, schnelle Methoden eingesetzt, um die Anzahl der Kollisionstests zu reduzieren. In der zweiten Phase, die auch als *Narrow Phase* bezeichnet wird, müssen die Paare von Objekten, bei denen in der ersten Phase eine Kollision nicht ausgeschlossen werden konnte, genau überprüft werden. Durch die Aufteilung in zwei Phasen wird die Kollisionserkennung im Allgemeinen erheblich beschleunigt.

Da die dynamische Simulation in diskreten Zeitschritten abläuft, wird auch die Kollisionserkennung nur an diskreten Zeitpunkten durchgeführt. Dies kann unter Umständen dazu führen, dass Kollisionen nicht erkannt werden. Wenn zum Beispiel ein kleiner Körper mit einer hohen Geschwindigkeit durch eine dünne Wand fliegt,

kann es passieren, dass er sich zu einem Zeitpunkt vor der Wand befindet und im nächsten Simulationsschritt vollständig hinter der Wand. Die Kollision zwischen Körper und Wand wird in diesem Fall nicht erkannt und kann somit auch nicht aufgelöst werden. Das führt zu einer fehlerhaften Simulation. Es gibt verschiedene Ansätze, um dieses Problem zu lösen. Einige davon werden in Abschnitt 2.2.4 beschrieben.

2.2.1 Repräsentation der Kollisionsobjekte

Bei der Kollisionserkennung in der dynamischen Simulation wird jedes Kollisionsobjekt durch ein dreidimensionales Modell repräsentiert. Diese Modelle wurden von Ming C. Lin und Stefan Gottschalk in [LG98] in verschiedene Klassen unterteilt (siehe Abbildung 2.2). In ihrer Arbeit geben sie einen Überblick über unterschied-

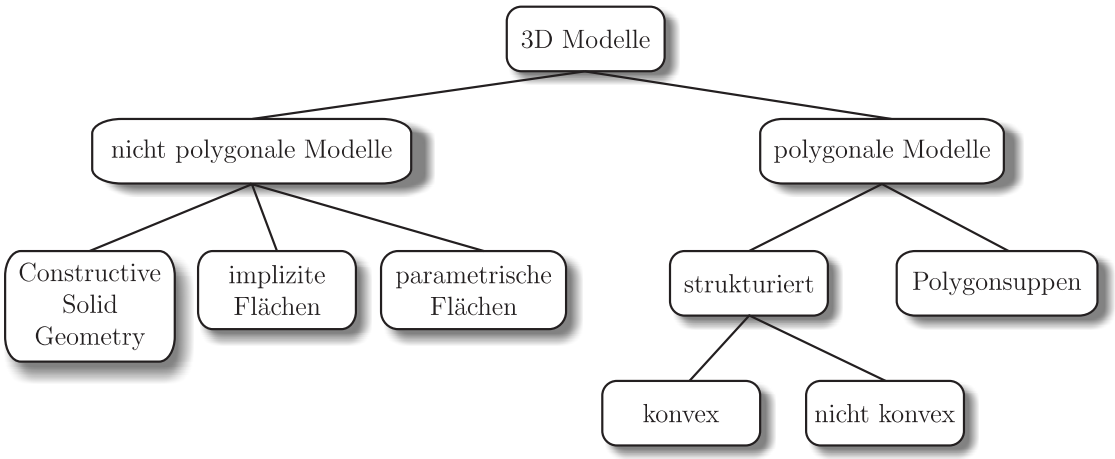


Abbildung 2.2: Klassifizierung von 3D Modellen in unterschiedliche Kategorien

liche Verfahren zur Kollisionserkennung. Für jede Klasse von Modellen gibt es speziell angepasste Algorithmen.

Grundsätzlich wird zwischen polygonalen und nicht polygonalen Modellen unterschieden. Die nicht polygonalen Modelle lassen sich weiter unterteilen in implizite Flächen, parametrische Flächen und Modelle, die mit der Technik der *Constructive Solid Geometry* (CSG) [Hof89] erzeugt wurden. Polygonale Modelle werden unterteilt in Polygonsuppen und strukturierte Modelle. Eine Polygonsuppe ist eine Menge von Polygonen, die beliebig angeordnet sein können und damit unstrukturiert sind. Bei den strukturierten Modellen bilden die Polygone einen Körper, der eine geschlossene Oberfläche hat. Dieses Polyeder kann entweder konvex oder nicht konvex sein. Viele schnelle Algorithmen zur Kollisionserkennung arbeiten mit konvexen Polyedern, da für diese Art der Körper viele Optimierungen mög-

lich sind. Es wird im Folgenden auf diese Algorithmen näher eingegangen (siehe Abschnitt 2.2.3).

Implizite Flächen werden durch eine implizite Funktion definiert. Die Oberfläche eines Kollisionsobjektes kann durch die Funktion $f(x, y, z) = 0$ beschrieben werden. Das Äußere des Objektes ist dann durch $f(x, y, z) > 0$ und das Innere durch $f(x, y, z) < 0$ definiert. Wenn die Funktion f ein Polynom zweiten Grades für x, y und z bildet, dann spricht man von einer *Quadrik*. Mit einer Quadrik können zum Beispiel Kugeln, Kegel und Zylinder beschrieben werden. Es gibt verschiedene Verfahren für die Kollisionserkennung bei Quadriken [SJ91, DLLP03].

Eine parametrische Fläche wird durch eine Funktion beschrieben, die einen zweidimensionalen Parameterbereich in den dreidimensionalen Raum abbildet: $f : \mathbb{R}^2 \mapsto \mathbb{R}^3$. Eine wichtige Klasse der parametrischen Flächen bilden die sogenannten *Non-Uniform Rational B-Splines* (NURBS) [PT95], die oft im CAD-Bereich zum Einsatz kommen. Die Kollisionserkennung für solche Flächen wird unter anderem in [LM97] diskutiert.

Bei der CSG-Technik werden komplexe Modelle aus Primitiven erzeugt. Primitive, die dabei verwendet werden, sind z. B. Würfel, Kugeln, Zylinder und Kegel. Ein komplexer Körper wird konstruiert, indem Primitive mit Hilfe von Mengenoperationen (Vereinigung, Differenz und Schnitt) miteinander verknüpft werden [Hof89]. Zwei CSG-Objekte haben genau dann eine Kollision, wenn die Schnittmenge der beiden Objekte nicht leer ist. Um zu überprüfen, ob die Schnittmenge leer ist, kann für das resultierende Objekt eine Oberflächendarstellung (Boundary Representation) berechnet werden. Eine schnelle und genaue Berechnung dieser Oberflächendarstellung ist allerdings schwierig [Hof89, KKM97].

2.2.2 Reduktion der Kollisionstests

Die Kollisionserkennung muss in jedem Zeitschritt alle Kollisionen zwischen den Körpern in der simulierten Szene ermitteln. Dafür muss ein genauer Kollisionstest für jedes Paar von Körpern durchgeführt werden, bei dem der Abstand der beiden Körper berechnet wird. Dies ist sehr zeitaufwendig, da die Körper eine komplexe Geometrie haben können. Es ist sinnvoll, die Anzahl dieser Kollisionstests zu reduzieren. Die beiden wichtigsten Techniken dafür sind die Verwendung von Hüllkörpern und der Einsatz von Zellrasterverfahren.

2.2.2.1 Hüllkörper

Ein Hüllkörper ist im Allgemeinen ein einfaches geometrisches Objekt, das das Kollisionsobjekt annähert und vollkommen einschließt. Verschiedene Hüllkörper, die bei der Kollisionserkennung zum Einsatz kommen, sind: Kugeln [Qui94,Hub95], achsenorientierte Quader [CLMP95], am Objekt ausgerichtete Quader [Got00] und

sogenannte k-DOPs. Ein k-DOP ist ein diskret orientiertes Polyeder, das aus k Flächen besteht [HKM96, KHM+98, Klo98].

Der Kollisionstest für die Hüllkörper ist relativ einfach und kann daher sehr schnell durchgeführt werden. Wenn zwei Hüllkörper sich nicht durchdringen, dann haben auch die zugehörigen Körper keine Kollision. Nur im Fall einer Durchdringung muss ein genauer Test für die Kollisionsobjekte durchgeführt werden. Durch dieses Vorgehen kann die Kollisionserkennung erheblich beschleunigt werden. Je besser ein Hüllkörper sein zugehöriges Kollisionsobjekt annähert, um so eher kann ein genauer Kollisionstest eingespart werden. Im Allgemeinen lassen sich mit komplexeren Hüllkörpern die zugehörigen Kollisionsobjekte besser approximieren. Allerdings sind die Kollisionstests für komplexere Hüllkörper zeitaufwendiger.

Oft werden Hierarchien von Hüllkörpern verwendet, um ein Objekt in verschiedenen Auflösungsstufen anzunähern. Ein Kollisionstest mit einer feineren Auflösungsstufe wird nur dann durchgeführt, wenn eine Kollision durch den Test mit der gröberen Auflösung nicht ausgeschlossen werden konnte. Durch eine Hierarchie von Hüllkörpern kann ein Kollisionsobjekt beliebig genau approximiert werden. Ein Vergleich verschiedener Hüllkörperhierarchien befindet sich in [Zac00].

Eine Kollision zwischen zwei Kugeln liegt vor, wenn der Abstand der Mittelpunkte kleiner ist als die Summe der Radien. Bei achsenorientierten Quadern kann der sogenannte *Sweep-And-Prune*-Algorithmus verwendet werden, um alle Überlappungen in einer Szene zu bestimmen [Bar92]. Bei diesem Algorithmus werden alle Quader auf die drei Achsen des Koordinatensystems projiziert. Auf den Achsen entsteht durch die Projektion für jeden Hüllkörper ein Intervall. Die Anfangs- und Endpunkte der Intervalle werden anschließend auf den Achsen sortiert. Zum Schluss wird jede Achse einmal abgelaufen, um die Überlappungen der Intervalle zu bestimmen. Genau die Quader, deren Intervalle auf allen drei Achsen überlappen, durchdringen sich gegenseitig. Die sortierten Listen der Intervalle werden gespeichert und beim nächsten Durchlauf werden nur die Änderungen durch Sortieren mit Einfügen vorgenommen. Wenn sich zwischen zwei Durchläufen nicht viel in der Szene ändert, dann müssen wenige Elemente in der Liste umsortiert werden. Dadurch ergibt sich ein Erwartungswert von $O(n)$ für den Zeitaufwand.

Eine Überlappung von zwei Quadern, die an ihrem jeweiligen Kollisionsobjekt ausgerichtet sind, wird mit einem Kollisionstest, der auf dem sogenannten *Separating Axis Theorem* [Got00] basiert, erkannt. Bei diesem Test wird eine Ebene gesucht, die die beiden Quader vollständig voneinander trennt. Wenn beide Quader auf die Normale einer Ebene projiziert werden, entstehen auf der Normalen zwei Intervalle. Die Ebene trennt die beiden Quader, wenn sich die Intervalle auf der Normalen nicht überlappen. Die Normale der Ebene ist dann eine sogenannte *Trennachse*. Das verwendete Theorem sagt aus, dass sich zwei konvexe Polyeder genau dann nicht schneiden, wenn eine solche Achse existiert. Die gesuchte Achse ist entweder senkrecht zu einer Fläche der Polyeder oder sie ist zu jeweils einer Kante von

beiden Polyedern senkrecht. Dies bedeutet, dass alle Flächennormalen und die Kreuzprodukte von jeweils zwei Kanten überprüft werden müssen. Da ein Quader symmetrisch ist, müssen höchstens 15 Tests durchgeführt werden. Wird bei keinem Test eine Trennachse gefunden, dann schneiden sich die Quader.

Beim Einsatz von k-DOPs als Hüllkörper werden die Polyeder oft so gewählt, dass sie alle die gleiche Anzahl k an Flächen haben und die jeweils gegenüberliegenden Flächen parallel zueinander und fest ausgerichtet sind. In diesem Fall kann ein Schnitt der Hüllkörper durch einen Überlappungstest der Intervalle (analog zum Test bei achsenorientierten Quadern) auf den $k/2$ Achsen durchgeführt werden, die durch die Flächennormalen bestimmt sind. Allerdings muss ein Polyeder bei Rotation des zugehörigen Kollisionsobjektes neu bestimmt werden. Wenn die k-DOPs frei gewählt werden und an den Kollisionsobjekten ausgerichtet sind, muss beim Kollisionstest, analog zu den am Objekt ausgerichteten Quadern, eine Trennachse gesucht werden.

2.2.2.2 Zellrasterverfahren

Bei diesem Verfahren wird eine Szene durch ein Raster in einzelne Zellen aufgeteilt. Jedes Kollisionsobjekt wird den Zellen zugewiesen, die das Objekt zumindest teilweise enthalten. Die Kollisionserkennung muss nur für Objekte, die sich in der gleichen Zelle befinden, durchgeführt werden. Wenn sich ein Objekt bewegt, muss erneut überprüft werden, welche Zellen es schneidet. Dabei kann Kohärenz ausgenutzt werden, indem zunächst die benachbarten Zellen überprüft werden. Die Schwierigkeit des Verfahrens liegt in der Wahl der Zellgröße. Sind die Zellen zu groß, dann beinhaltet eine Zelle viele Kollisionsobjekte und nur wenige Kollisionstests können eingespart werden. Bei einer zu kleinen Zellgröße gibt es sehr viele Zellen und viele Objekte werden mehreren Zellen zugewiesen. Außerdem wechseln die Objekte oft die Zellen. Dadurch steigt der Verwaltungsaufwand enorm.

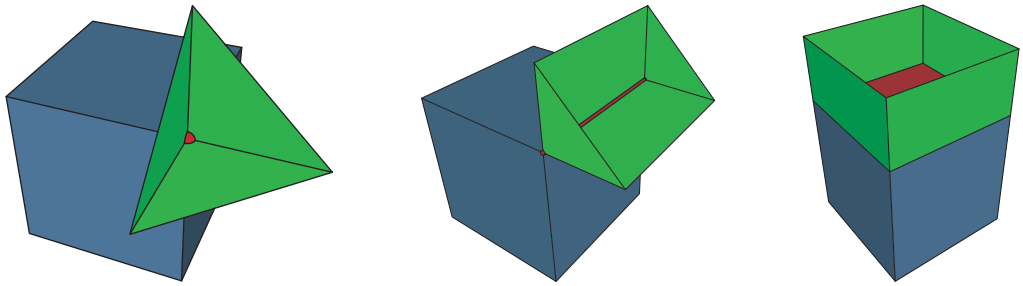
2.2.3 Bestimmung der nächsten Punkte

Für die exakte Bestimmung der nächsten Punkte von zwei Körpern existieren verschiedene Verfahren. Die beiden wichtigsten sind der Algorithmus von Ming C. Lin und John F. Canny [LC91, Lin93] und der Algorithmus von Elmer G. Gilbert, Daniel W. Johnson und Sathiya S. Keerth [GJK88]. Diese beiden Verfahren werden im Folgenden als *Lin-Canny-Algorithmus* und *GJK-Algorithmus* bezeichnet.

2.2.3.1 Der Lin-Canny-Algorithmus

Der Lin-Canny-Algorithmus verwendet Voronoi-Regionen, um die nächsten Punkte von zwei konvexen Polyedern zu bestimmen. Jedes Polyeder besteht aus Punkten,

Kanten und Flächen, den sogenannten *Elementen* des Polyeders. Die Voronoi-Region eines dieser Elemente ist die Menge aller Punkte außerhalb des Körpers, deren Distanz zu dem Element kleiner ist als zu allen anderen Elementen des Körpers. In einem Vorverarbeitungsschritt werden diese Regionen für alle Punkte, Kanten und Flächen ermittelt. Die Voronoi-Regionen der Elemente sind in Abbildung 2.3 dargestellt. Die Region eines Punktes wird durch die Ebenen, die



(a) Punkt (b) Kante (c) Fläche

Abbildung 2.3: Voronoi-Regionen der einzelnen Elemente

senkrecht zu den anliegenden Kanten stehen und den Punkt beinhalten, eingeschlossen. Bei einer Kante wird die Voronoi-Region durch vier Ebenen begrenzt. Zwei dieser Ebenen stehen senkrecht zu den anliegenden Flächen und gehen durch die Kante selbst. Die anderen beiden Ebenen beinhalten jeweils einen Endpunkt der Kante und stehen senkrecht zu der Kante. Die Voronoi-Region einer Fläche wird von den Ebenen begrenzt, die senkrecht zur Fläche stehen und durch die anliegenden Kanten gehen.

Mit den Voronoi-Regionen können die nächsten Elemente von zwei konvexen Polyedern bestimmt werden und damit auch ihre nächsten Punkte. Der Algorithmus beginnt mit zwei beliebigen Elementen der Polyeder. Für diese Elemente werden die nächsten Punkte bestimmt. Anschließend wird überprüft, ob der nächste Punkt des ersten Elements in der Voronoi-Region des zweiten Elements liegt und umgekehrt. Ist dies der Fall, dann handelt es sich bei den Punkten um die nächsten Punkte der beiden Polyeder. Eine Voronoi-Region wird durch Ebenen begrenzt, deren Normalen in das Innere der Region zeigen. Ein Punkt befindet sich genau dann innerhalb der Region, wenn er auf der positiven Seite jeder Ebene der Region liegt. Wenn beim Überprüfen der ermittelten nächsten Punkte ein Punkt auf der negativen Seite einer Ebene liegt, dann existieren Punkte in den Polyedern, die einen kleineren Abstand haben. Die Suche nach den nächsten Elementen wird in diesem Fall mit dem Element fortgesetzt, das an diese Ebene angrenzt. Das Verfahren konvergiert in linearer Zeit gegen die korrekte Lösung, wenn sich die Polyeder nicht gegenseitig durchdringen.

Im Fall einer Durchdringung kann es zu einer Endlosschleife kommen. In den Arbeiten von Jonathan D. Cohen und Brian V. Mirtich [CML+94, Mir98] werden so-

genannte Pseudo-Voronoi-Regionen für das Innere eines Polyeders bestimmt, um dieses Problem zu lösen. Wenn sich die Polyeder zwischen zwei Tests nur wenig bewegen, dann ist es sinnvoll, den Algorithmus mit den zuletzt bestimmten nächsten Elementen zu beginnen. Dadurch wird die zeitliche und geometrische Kohärenz ausgenutzt und es ergibt sich ein Erwartungswert für den Berechnungsaufwand von $O(1)$.

2.2.3.2 Der GJK-Algorithmus

Der GJK-Algorithmus arbeitet mit Objekten, die aus konvexen Polyedern zusammengesetzt sind. Außerdem kann jedes Objekt kugelförmig erweitert werden. Damit lässt sich zum Beispiel eine Kugel als Erweiterung eines Punktes beschreiben. Der Abstand zwischen zwei Körpern wird mit Hilfe der sogenannten *Minkowski-Differenz* bestimmt. Diese ist für die Objekte A und B wie folgt definiert:

$$A - B := \{a - b : \ a \in A, \ b \in B\} .$$

Zur Bestimmung des Abstands der Objekte A und B wird der nächste Punkt der Minkowski-Differenz $A - B$ zum Ursprung ermittelt. Für den Abstand gilt:

$$d(A, B) = \min\{|\mathbf{x}| : \ \mathbf{x} \in A - B\}.$$

Liegt der Ursprung in $A-B$, dann ist der Abstand Null und die Objekte schneiden sich.

Die Minkowski-Differenz von zwei konvexen Polyedern ergibt wieder ein konvexes Polyeder. Dieses wird allerdings beim GJK-Algorithmus nicht explizit bestimmt. Stattdessen wird der nächste Punkt zum Ursprung mit Hilfe sogenannter *Simplexe* angenähert. Ein Simplex ist die konvexe Hülle einer affin unabhängigen Punktmenge. Im dreidimensionalen Raum kann ein Simplex aus bis zu vier Punkten bestehen. Der GJK-Algorithmus beginnt mit einem beliebigen Punkt aus $A - B$ und bestimmt in jedem Schritt ein neues Simplex, das näher am Ursprung liegt als das vorherige. Für Polyeder konvergiert der Algorithmus in einer endlichen Anzahl von Schritten gegen die korrekte Lösung [GJK88]. Eine ausführliche Beschreibung des GJK-Algorithmus befindet sich in [vdB04]. Außerdem wird in diesem Buch eine Erweiterung des GJK-Algorithmus vorgestellt, mit der die exakte Eindringtiefe zweier sich schneidender Körper bestimmt werden kann.

2.2.4 Diskrete und kontinuierliche Kollisionserkennung

Die meisten Verfahren zur Kollisionserkennung arbeiten in diskreten Schritten. Wenn sich die Kollisionsobjekte zwischen den Schritten weit bewegen, dann werden unter Umständen Kollisionen, die zwischen den einzelnen Schritten passieren, nicht erkannt. Bei der dynamischen Simulation kann dieses Problem auftreten,

wenn sich relativ kleine Körper sehr schnell bewegen. Solche Simulationen werden oft mit einer sehr kleinen Zeitschrittweite durchgeführt, um dem Problem entgegen zu wirken. Allerdings kann auch bei einer kleinen Schrittweite nicht garantiert werden, dass keine Kollision verpasst wird. Meistens wird eine Kollision von einem diskreten Verfahren erst dann erkannt, wenn sich zwei Körper bereits durchdringen. Der letzte Simulationsschritt muss in diesem Fall rückgängig gemacht werden. Anschließend wird der genaue Zeitpunkt der Kollision, z. B. mit einem Intervallhalbierungsverfahren, ermittelt.

Brian V. Mirtich verwendet in seiner Arbeit Hüllkörper, die die Kollisionsobjekte während eines gesamten Zeitschritts einschließen [MC94,Mir96b]. Zur Bestimmung der Hüllkörper wird die ballistische Bewegung der Körper berechnet. Nur wenn sich zwei Hüllkörper schneiden, kann in dem Zeitschritt eine Kollision zwischen den zugehörigen Körpern passieren. In diesem Fall wird eine untere Schranke für den Kollisionszeitpunkt berechnet. Diese muss aktualisiert werden, wenn einer der Körper mit einem anderen kollidiert, da sich dann die ballistische Bewegung ändert.

Bei den Verfahren der kontinuierlichen Kollisionserkennung wird die Bewegung eines Körpers zwischen den Zeitschritten angenähert. Auf den kontinuierlichen Bahnen der Körper erfolgt dann die Kollisionserkennung. Es wurden verschiedene Techniken entwickelt, um die kontinuierliche Bewegung eines Körpers zu approximieren [KR03, RKC00, RKC02]. Die Approximation wird bei diesen Verfahren mit Hilfe von Interpolation vorgenommen. Dabei wird davon ausgegangen, dass sich die Körper frei im Raum bewegen. Stephane Redon stellt in [RKLM04] ein kontinuierliches Verfahren vor, das speziell für Körper entwickelt wurde, die durch Gelenke miteinander verbunden sind. Die kontinuierliche Kollisionserkennung hat den Vorteil, dass keine Kollision verpasst wird. Außerdem können die kontinuierlichen Verfahren den genauen Zeitpunkt einer Kollision relativ schnell ermitteln. Diese Verfahren haben aber den Nachteil, dass sie einen wesentlich höheren Berechnungsaufwand gegenüber den diskreten Verfahren haben und daher langsamer sind.

2.3 Kollisionsauflösung

Das Ziel der Kollisionsauflösung in der dynamischen Simulation ist, Durchdringungen zwischen Körpern zu verhindern. Im Falle einer Kollision muss außerdem abhängig von der Geschwindigkeit des Aufpralls und der Elastizität der Körper ein Rückstoß simuliert werden. Wenn zwei Körper dagegen permanent in Kontakt sind, muss ausschließlich eine Durchdringung verhindert werden. In beiden Fällen tritt im Allgemeinen Reibung zwischen den Körpern auf. Die Simulation von Reibung basiert bei den meisten Verfahren auf dem Reibungsgesetz von Coulomb.

Für eine Simulation in Echtzeit muss ein Verfahren zur Kollisionsauflösung sehr

schnell arbeiten. Es soll ebenfalls über eine hohe physikalische Genauigkeit verfügen. Durch diese beiden Anforderungen haben sich zwei verschiedene Ansätze für die Behandlung von Kollisionen durchgesetzt. Der erste Ansatz verwendet Zwangsbedingungen, um Durchdringungen zwischen Körpern zu vermeiden. Dagegen arbeitet der zweite Ansatz ausschließlich mit Impulsen. Diese beiden Ansätze werden im Folgenden näher erläutert.

2.3.1 Kollisionsauflösung mit Zwangsbedingungen

Bei diesem Ansatz wird für jeden Kontakt, der zwischen zwei Körpern auftritt, eine Zwangsbedingung formuliert. Diese Bedingung sagt aus, dass sich die Körper in diesem Kontaktpunkt nicht durchdringen dürfen. Anschließend müssen Kräfte bestimmt werden, die dafür sorgen, dass diese Zwangsbedingungen erfüllt werden. In [Ste00] wird ein ausführlicher Überblick der Techniken für die Kollisionsauflösung und die Kontaktbehandlung mit Reibung gegeben.

2.3.1.1 Penalty-Verfahren

Die sogenannten *Penalty-Verfahren* berechnen die gesuchten Kräfte mit Hilfe eines Federmodells. Matthew Moore und Jane Wilhelms beschreiben in [MW88b], wie bleibende Kontakte auf diese Weise behandelt werden können. In jedem Kontaktpunkt wird eine Feder zwischen den beiden zugehörigen Körpern eingefügt. Abhängig von der Tiefe der Durchdringung wirkt dann eine Federkraft auf die Körper und zieht sie auseinander. Das bedeutet, dass bei diesem Verfahren erst dann eine Kraft wirkt, wenn ein fehlerhafter Zustand im System auftritt. Die Größe der Kraft ist abhängig von der Größe des Fehlers. Nachdem ein Kontakt aufgelöst ist, wird die zugehörige Feder wieder aus dem System entfernt.

Mit dieser Vorgehensweise können genaue Ergebnisse nur erreicht werden, wenn diese Federn sehr steif sind. Andernfalls können sich die Körper stark durchdringen, bevor sie auseinander gezogen werden. Bei der Verwendung von steifen Federn müssen bei der numerischen Integration allerdings sehr kleine Zeitschrittweiten verwendet werden, sonst wird das System instabil [PB88, MW88b]. Der Vorteil des Penalty-Verfahrens ist, dass es leicht zu implementieren und sehr schnell ist. Aus diesen Gründen wird das Verfahren meistens nur dann eingesetzt, wenn eine sehr schnelle Kollisionsauflösung benötigt wird und ein plausibles Simulationsergebnis ausreichend ist.

2.3.1.2 Analytische Verfahren

Analytische Verfahren unterscheiden zwischen Kollisionen und bleibenden Kontakten [Bar89]. Kollisionen haben eine unendlich kleine Zeitdauer. Sie werden daher

von den meisten Verfahren mit Kraftstößen bzw. Impulsen aufgelöst. Dagegen werden bei vielen analytischen Verfahren für bleibende Kontakte Kräfte in Normalenrichtung berechnet, die eine Durchdringung der Körper während des Zeitintervalls des Kontakts verhindern. Nur wenn die Körper in Richtung der Kontaktnormalen aufeinander zu beschleunigen, dürfen diese Kontaktkräfte wirken. Andernfalls trennen sich die Körper von selbst. Das Problem der Bestimmung dieser Kräfte wird oft als lineares Komplementaritätsproblem [CPS92] formuliert. Dies geht auf die Arbeiten von Per Lötstedt zurück [Löt82, Löt84].

David Baraff präsentiert in [Bar89] ein Verfahren für Polyeder, bei dem die Kontaktkräfte analytisch berechnet werden. Kollisionen zwischen zwei Körpern können in mehreren Punkten gleichzeitig auftreten und werden mit Hilfe von Impulsen aufgelöst. Die Bestimmung der Kontaktkräfte wird zunächst als quadratisches Programmierproblem [GT00] formuliert. Im Allgemeinen ist quadratisches Programmieren NP-hart. Daher wird dieses Problem mit Hilfe eines heuristischen Ansatzes gelöst. Dieses Verfahren wird in [Bar90] für gekrümmte Flächen erweitert. Die Simulation von Kontakten mit Reibung wird in [Bar91] behandelt. Reibung wird dabei mit dem Gesetz von Coulomb simuliert. Das in [Bar91] beschriebene Verfahren verwendet bei der Kontaktbehandlung sowohl Kräfte als auch Impulse.

In [Bar94] stellt David Baraff eine neue Methode zur Berechnung der Kräfte in Kontaktpunkten mit Reibung vor. Diese Methode ist einfacher, schneller und robuster als frühere Verfahren. Im Gegensatz zu vorherigen Arbeiten wird das Problem dabei nicht in ein Optimierungsproblem umgewandelt. Das Verfahren ist für den reibungsfreien Fall äquivalent zu dem Algorithmus, der in [CD68] beschrieben wird. Dabei muss für jeden Kontaktpunkt i die folgende Bedingung für die relative Beschleunigung a_i und die Kontaktkraft f_i in Normalenrichtung erfüllt werden:

$$a_i \geq 0, \quad f_i \geq 0 \quad \text{und} \quad f_i\, a_i = 0. \tag{2.9}$$

Diese Bedingung sagt aus, dass sich die Körper entweder von alleine auseinander bewegen oder eine Kraft f_i in Richtung der Kontaktnormalen angewendet werden muss, die dafür sorgt, dass die relative Beschleunigung der Körper im Kontaktpunkt Null ist. Dadurch wird eine Durchdringung verhindert. Am Anfang werden alle f_i auf Null gesetzt. Anschließend werden die Kräfte für die Kontaktpunkte nacheinander bestimmt, so dass sie die Bedingung 2.9 erfüllen. Wenn im j-ten Schritt $a_j < 0$ gilt, dann muss eine Kraft f_j bestimmt werden, so dass die Beschleunigung Null wird. Anschließend muss für alle Kontakte $1 \leq i \leq j-1$ überprüft werden, ob die Bedingung 2.9 noch erfüllt ist. Gegebenenfalls müssen die Kräfte für einige Kontakte angepasst werden. Auf diese Weise werden alle Kontaktkräfte bestimmt. Die Simulation von dynamischer und statischer Reibung kann durch eine Modifikation des beschriebenen Algorithmus durchgeführt werden.

Katsuaki Kawachi et al. erweitern in [KSK97] das Verfahren zur Bestimmung der Kollisionsimpulse aus [Bar89]. Durch diese Erweiterung ist es möglich, Kollisionen mit dynamischer und statischer Reibung mit Hilfe von Impulsen aufzulösen. Die

Simulation von Reibung basiert auf dem Gesetz von Coulomb. Wenn zwei Körper zu einem Zeitpunkt mehrere Kontaktpunkte haben, werden die auftretenden Kollisionen gleichzeitig behandelt. Die vorgestellte Erweiterung wurde in [KSK97] zunächst nur für den zweidimensionalen Fall beschrieben. Basierend auf dieser Arbeit wurde dann in [KSK98] der dreidimensionale Fall behandelt.

Verfahren, wie sie zum Beispiel in [Löt82] oder [PG96] vorgestellt werden, formulieren in jedem Zeitschritt ein lineares Komplementaritätsproblem und bestimmen damit die Kräfte zur Auflösung. Diese wirken dann im folgenden Zeitschritt auf die Körper. Das Problem bei solchen Verfahren ist, dass bei der Bestimmung der Kontaktkräfte mit Reibung unter Umständen keine eindeutige oder gar keine Lösung gefunden werden kann [MW88a, Bar93]. Das wurde bereits von Painlevé im Jahr 1895 entdeckt. David E. Stewart und Jeff C. Trinkle stellen in [ST96] und [ST97] einen Ansatz vor, um dieses Problem bei der Auflösung von unelastischen Kollisionen mit Reibung zu lösen. Die grundlegende Idee dieses Ansatzes ist, das Integral der Kräfte über einen Zeitschritt zu bestimmen. Dabei wird der Reibungskegel durch ein Polyeder angenähert. In [APS98] wird gezeigt, wie die Berechnung ohne Annäherung des Reibungskegels durchgeführt wird. Die resultierenden Impulse werden mit Hilfe eines nichtlinearen Komplementaritätsproblems ermittelt. Es wird gelöst, indem für eine Reihe von linearen Komplementaritätsproblemen mit Lemkes Algorithmus eine Lösung bestimmt wird. Jörg Sauer und Elmar Schömer verwenden in ihren Arbeiten [SS98a,SS98b] ebenfalls diesen Ansatz, um permanente Kontakte mit Reibung zu simulieren. Für die Auflösung von Kollisionen setzen sie eine impulsbasierte Methode ein.

Paul G. Kry und Dinesh K. Pai verwenden reduzierte Koordinaten, um einen einzelnen kontinuierlichen Kontakt zwischen glatten Oberflächen zu behandeln [KP03]. Die Formulierung mit reduzierten Koordinaten ist sehr komplex, aber sie ermöglicht die Wahl von größeren Schrittweiten bei einer Simulation mit plausiblen Ergebnissen. Das Reibungsmodell von Coulomb wurde ebenfalls in die Formulierung integriert.

Victor J. Milenkovic simuliert in [Mil96] eine hohe Anzahl an Körpern, die miteinander interagieren, durch die Definition von Zwangsbedingungen für die Positionen der Körper. Es wird dabei eine vereinfachte Physik verwendet, die er als *positionsbasierte Physik* bezeichnet. Durch diese Vereinfachung ist es möglich, sehr schnell eine plausible Bewegung für viele Körper zu berechnen. Die Simulation erfolgt durch Minimierung von Energie mit Hilfe von linearer Programmierung. In [MS01] wird für jeden Körper vor dem Simulationsschritt eine Zielposition berechnet, die der Körper ohne Kollision erreichen würde. Ein Optimierungsalgorithmus bewegt die Körper dann so nah wie möglich an die Zielposition unter der Bedingung, dass keine Durchdringung erlaubt ist. Die Körper bewegen sich nach den Gesetzen von Newton, kollidieren elastisch und haben Reibung. Dadurch ist eine realistischere Simulation möglich als mit der positionsbasierten Physik. Die neuen Positio-

nen und Beschleunigungen der Körper werden mit quadratischem Programmieren bestimmt. Das Verfahren eignet sich für physikalisch plausible Animationen, ist aber nicht für genaue Simulationen gedacht. Schmidl und Milenkovic verwenden in [SM04] Impulse, um Kollisionen und Kontakte mit Reibung aufzulösen. Die Impulse für die Kollisionen und die bleibenden Kontakte werden jeweils mit einem quadratischen Programm bestimmt. Die Simulation wird durch Einfrieren von Körpern, die sich über einen gewissen Zeitraum nicht bewegen, beschleunigt. Für die eingefrorenen Körper müssen keine Kollisionen oder Kontakte aufgelöst werden. Durch eine Kollision kann ein eingefrorener Körper an Geschwindigkeit gewinnen und muss daher wieder aktiviert werden.

In [KEP05] werden Durchdringungen von Körpern verhindert, indem ihre Geschwindigkeiten auf einen zulässigen Bereich beschränkt werden. Dabei wird jeder Körper als ein unabhängiges System betrachtet. Die zulässigen Geschwindigkeiten und die Reibung zwischen den Körpern werden mit zwei konvexen, quadratischen Programmen pro Körper beschrieben. Die beiden quadratischen Programme sind linear zur Anzahl der Kontaktpunkte. Mit diesem Verfahren sind plausible Simulationen von vielen, nicht konvexen Körpern möglich.

2.3.2 Impulsbasierte Kollisionsauflösung

Der impulsbasierte Ansatz zur Behandlung von Kollisionen und Kontakten verwendet keine Bedingungen. Wenn zwei Körper Kontakt miteinander haben, dann wird in den nächsten Punkten der beiden Körper ein Impuls angewendet, der verhindert, dass sich die Körper durchdringen. Im Fall einer Kollision simuliert dieser Impuls den Rückstoß. Die Bestimmung solcher Impulse wird z. B. in [CR98] ausführlich besprochen. Haben zwei Körper in mehreren Punkten Kontakt, dann werden die Kollisionen in den Punkten zeitlich nacheinander aufgelöst. Dadurch muss für zwei Körper immer nur ein Kollisionsimpuls berechnet werden. Impulse verändern die Geschwindigkeiten der Körper sofort. Daher kann bei der Auflösung einer Kollision eine neue Kollision entstehen. Die Auflösung geschieht daher in einer Schleife, bis alle Kollisionen behandelt wurden.

Moore und Wilhelms präsentieren in ihrer Arbeit eine analytische Methode, um die Impulse für die Kollisionsauflösung zu bestimmen [MW88b]. Dagegen werden bleibende Kontakte durch ein Penalty-Verfahren aufgelöst (siehe oben). Bei ihrer Methode werden Kollisionen in nur einem einzigen Punkt aufgelöst. Kollisionen, die gleichzeitig in mehreren Punkten stattfinden, werden als eine Serie von Kollisionen betrachtet und nacheinander aufgelöst. Auf die gleiche Weise werden Kollisionen von James K. Hahn in [Hah88] behandelt. Die Durchdringung von zwei Körpern im Fall eines bleibenden Kontaktes wird in dieser Arbeit durch eine Serie von Kollisionen verhindert. Dies stellt allerdings nur eine Annäherung an die

exakte Lösung dar. Dynamische und statische Reibung wird nach dem Gesetz von Coulomb simuliert.

Das Verfahren von Moore und Wilhelms und das von Hahn bilden damit die Anfänge der impulsbasierten Methoden. Der Begriff der impulsbasierten Kollisionsauflösung wurde allerdings hauptsächlich durch die Arbeiten von Brian V. Mirtich und John F. Canny geprägt [MC94,MC95,Mir96b]. Bei ihrem Verfahren werden sowohl Kollisionen als auch bleibende Kontakte mit Hilfe von Impulsen aufgelöst. Für zwei Körper wird genau ein Kontaktpunkt bei der Auflösung berücksichtigt. Wenn mehrere Kontaktpunkte existieren, dann müssen diese nacheinander behandelt werden. Das Modell von Coulomb wird bei der Auflösung verwendet, um dynamische und statische Reibung zu simulieren. Die Simulation eines Rückstoßes wird unter Verwendung von Stronges Hypothese durchgeführt. Dabei wird eine Kollision in eine Kompressions- und eine Rückstoßphase unterteilt. In der Kompressionsphase werden die relative Tangentialgeschwindigkeit der Körper im Kontaktpunkt und die Arbeit in Richtung der Normale integriert, bis die relative Geschwindigkeit der Körper in Normalenrichtung Null ist. Dann ist der Punkt der maximalen Kompression erreicht und es beginnt die Rückstoßphase. Am Ende der ersten Phase ist die Arbeit, die in Normalenrichtung geleistet wurde, bekannt. Mit Stronges Hypothese kann dadurch die Arbeit in Normalenrichtung für die gesamte Kollision berechnet werden. Sie dient daher als Parameter für die Integration der relativen Punktgeschwindigkeit in der zweiten Phase. Bei der Integration wird dynamische und statische Reibung berücksichtigt. Im Fall eines bleibenden Kontakts werden ständig Impulse berechnet, um eine Durchdringung der Körper zu verhindern. Da die Auflösung in nur einem Kontaktpunkt geschieht, fangen die Körper an zu vibrieren. Diese Vibration kann z. B. dazu führen, dass ein Körper trotz statischer Reibung auf einer schiefen Ebene zu rutschen anfängt. Dieses Problem wird mit Hilfe von sogenannten *Mikrokollisionen* gelöst. Eine Mikrokollision liegt vor, wenn die relative Geschwindigkeit eines Kontaktpunktes in Normalenrichtung innerhalb eines Toleranzbereichs liegt. In [Mir95] werden die Vorteile der impulsbasierten Methode und der Simulation mit Zwangsbedingungen untersucht. Außerdem wird die Kombination der beiden Ansätze diskutiert, welche in [Mir96b] umgesetzt wurde.

Eran Guendelman et al. unterscheiden in [GBF03,Gue06] zwischen Kollisionen und bleibenden Kontakten durch eine Veränderung des Ablaufs eines Simulationsschrittes. Dadurch benötigen sie keine Toleranz für die Kontaktpunktgeschwindigkeit. Ein Simulationsschritt wird in die folgenden Phasen eingeteilt. Am Anfang wird die Kollisionserkennung durchgeführt. Dabei wird für jedes Paar von kollidierenden Körpern ein Kontaktpunkt bestimmt. Anschließend werden alle Kollisionen mit Hilfe von Impulsen aufgelöst. Dann werden die Geschwindigkeiten der Körper nach dem Zeitschritt berechnet. Anhand der neuen Geschwindigkeiten kann bestimmt werden, ob sich zwei Körper während des Zeitschritts durchdringen werden. In diesem Fall wird die relative Geschwindigkeit der Körper in dem Kontaktpunkt durch einen Impuls angepasst und damit die Durchdringung verhindert. Der Vorteil

dieses veränderten Ablaufs ist, dass keine Vibration bei der Simulation bleibender Kontakte auftritt und damit keine Sonderbehandlung nötig ist. Durch die beschriebene Vorgehensweise ist eine Simulation von vielen Objekten möglich, die plausible Ergebnisse liefert. Um die Simulation weiter zu beschleunigen, wird die Methode der Schockfortpflanzung vorgeschlagen. Dabei wird ein Graph aufgestellt, der die Abhängigkeiten der Kontakte beschreibt. Die Kollisionen in diesem Kontaktgraph werden Ebene für Ebene aufgelöst. Nachdem die Auflösung auf einer Ebene des Graphen beendet ist, wird die Masse von allen zugehörigen Körpern auf unendlich gesetzt. Dadurch werden Objekte in bereits abgearbeiteten Ebenen nicht mehr von Impulsen beeinflusst. Wenn die Kollisionsauflösung auf allen Ebenen durchgeführt wurde, werden die ursprünglichen Massen wieder hergestellt. Die Methode der Schockfortpflanzung kann bei der Auflösung von Kollisionen oder Kontakten nach wenigen Iterationen angewendet werden. Das Verfahren von Eran Guendelman et al. wird von Rachel L. Weinstein et al. in [WTF06] eingesetzt, um Kollisionen und Kontakte zwischen Körpern aufzulösen, die durch Gelenke miteinander verbunden sind.

Kapitel 3

Dynamische Simulation eines Mehrkörpersystems

Ein Mehrkörpersystem kann aus statischen und dynamischen Körpern bestehen. Statische Körper sind unbeweglich und ändern daher weder ihre Position noch ihre Geschwindigkeit. Die Bewegung dynamischer Körper wird durch Kräfte beeinflusst. Dabei muss man zwischen externen und internen Kräften unterscheiden. Externe Kräfte sind Kräfte, die von außen auf das System als Ganzes wirken, wie z. B. Gravitation. Interne Kräfte wirken dagegen zwischen einzelnen Körpern des Systems, z. B. wenn diese untereinander mit Gelenken verbunden sind. Im Folgenden wird davon ausgegangen, dass die Summe der externen Kräfte $\mathbf{F}_{ext}$ und die Summe der externen Drehmomente $\boldsymbol{\tau}_{ext}$, die auf einen Körper wirken, während eines Simulationsschrittes konstant sind.

3.1 Bewegung eines freien Starrkörpers

Ein freier Körper wird durch keinen anderen Körper in seiner Bewegung beeinflusst. Er ist weder durch Gelenke mit einem anderen Körper verbunden, noch kollidiert er mit seiner Umgebung. Das heißt, dass keine internen Kräfte auf diesen Körper wirken und sein Bewegungszustand ausschließlich von externen Kräften verändert wird. Ein solcher Körper hat sechs Freiheitsgrade: drei translatorische und drei rotatorische.

3.1.1 Eigenschaften eines Starrkörpers

In der dynamischen Simulation wird ein Starrkörper beschrieben durch

- seine Masse m,

- die Position seines Schwerpunktes $\mathbf{s}(t)$,
- die Geschwindigkeit des Schwerpunktes $\mathbf{v}(t)$,
- seinen Trägheitstensor (in lokalen Koordinaten) $\mathbf{J}_l$,
- seine Rotationsmatrix $\mathbf{R}(t)$ und
- seine Winkelgeschwindigkeit $\boldsymbol{\omega}(t)$.

Die Masse und der Trägheitstensor in lokalen Koordinaten sind konstant. Der Trägheitstensor $\mathbf{J}$ eines Körpers mit konstanter Masse und konstanter Dichte kann durch Integration bestimmt werden (siehe Anhang D.1). Die resultierende Matrix ist reell, symmetrisch sowie positiv definit [Mir96b] und kann daher durch eine Hauptachsentransformation in Diagonalform gebracht werden [GPS06], d. h. $\mathbf{J}$ hat die Form:

$$\mathbf{J} = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}.$$

Das Koordinatensystem eines Körpers wird am Anfang der Simulation so gewählt, dass der Trägheitstensor $\mathbf{J}_l$ in diesem System Diagonalform hat und der Nullpunkt im Schwerpunkt des Körpers liegt. Dies hat den Vorteil, dass bei Berechnungen mit dem Trägheitstensor in lokalen Koordinaten vereinfachte Rechenoperationen verwendet werden können, die schneller sind. Bei vielen Berechnungen wird der inverse Trägheitstensor benötigt. In lokalen Koordinaten ist dieser konstant und kann vorberechnet werden:

$$\mathbf{J}_l^{-1} = \begin{pmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{pmatrix}.$$

Der Trägheitstensor $\mathbf{J}$ in Weltkoordinaten und der inverse Tensor $\mathbf{J}^{-1}$ werden mit Hilfe der Rotationsmatrix $\mathbf{R}$ des Körpers bestimmt:

$$\begin{aligned} \mathbf{J} &= \mathbf{R}^T \mathbf{J}_l \mathbf{R} \\ \mathbf{J}^{-1} &= \mathbf{R}^T \mathbf{J}_l^{-1} \mathbf{R}. \end{aligned}$$

Die Rotationsmatrix $\mathbf{R}$ beschreibt die aktuelle Rotation eines Körpers um seinen Schwerpunkt. Die Zeilenvektoren der Matrix bilden eine Orthonormalbasis des lokalen Koordinatensystems des Körpers, dies bedeutet, dass $\mathbf{R}^T = \mathbf{R}^{-1}$ gilt. Ein Punkt $\mathbf{a}_l$ in diesem lokalen Koordinatensystem hat im Weltkoordinatensystem die Position

$$\mathbf{a} = \mathbf{a}_l \mathbf{R} + \mathbf{s}. \tag{3.1}$$

3.1.2 Simulationsschritt

In einem Simulationsschritt von $t = t_0$ bis $t = t_0 + h$ müssen für jeden Körper in der Simulation die Lage und die Geschwindigkeit zum Zeitpunkt t_0 bekannt sein. Anschließend werden abhängig von der Zeitschrittweite h und den externen Kräften und Drehmomenten die Positionen und Geschwindigkeiten der Körper für den Zeitpunkt $t_0 + h$ berechnet.

3.1.2.1 Translation

Newtons zweites Gesetz beschreibt, wie die Bewegung eines Körpers durch Kräfte, die auf ihn wirken, verändert wird. Die Beschleunigung eines freien Körpers mit konstanter Masse m wird durch die externen Kräfte und seine Masse bestimmt:

$$\dot{\mathbf{v}} = \frac{1}{m}\,\mathbf{F}_{ext}.$$

Da die Summe der externen Kräfte $\mathbf{F}_{ext}$, die auf den Körper wirken, während eines Simulationsschrittes als konstant angenommen wird, ist auch die Beschleunigung des Körpers in diesem Zeitraum konstant. Die Änderung der Geschwindigkeit $\mathbf{v}$ während eines Zeitschrittes der Länge h wird durch Integration der Beschleunigung berechnet:

$$\mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + \int_0^h \dot{\mathbf{v}}\; dt = \mathbf{v}(t_0) + \frac{1}{m}\,\mathbf{F}_{ext}\,h. \tag{3.2}$$

Die Position des Schwerpunktes zum Zeitpunkt $t_0 + h$ ergibt sich durch eine weitere Integration:

$$\begin{aligned} \mathbf{s}(t_0 + h) &= \mathbf{s}(t_0) + \int_0^h \mathbf{v}(t_0) + \frac{1}{m}\,\mathbf{F}_{ext}\,t\; dt \\ &= \mathbf{s}(t_0) + \mathbf{v}(t_0)\,h + \frac{1}{2m}\,\mathbf{F}_{ext}\,h^2. \end{aligned} \tag{3.3}$$

Die Änderung der Position und der Geschwindigkeit des Schwerpunktes können direkt aus den externen Kräften berechnet und müssen nicht mit Hilfe von numerischer Integration bestimmt werden.

3.1.2.2 Rotation

Die Rotation und die Winkelgeschwindigkeit eines Körpers werden durch die externen Drehmomente, die auf den Körper wirken, beeinflusst. Die Änderung der Winkelgeschwindigkeit während eines Simulationsschrittes wird durch die Euler-Gleichung (siehe Anhang D.2)

$$\dot{\boldsymbol{\omega}}(t) = \mathbf{J}^{-1}\left(\boldsymbol{\tau}_{ext} - (\boldsymbol{\omega}(t) \times (\mathbf{J}\,\boldsymbol{\omega}(t)))\right) \tag{3.4}$$

beschrieben. Diese Differentialgleichung wird zunächst in das lokale Koordinatensystem des Körpers transformiert, da der Trägheitstensor in diesem System konstant ist. Anschließend wird die Gleichung mit Hilfe numerischer Verfahren gelöst (siehe Anhang B).

Die Rotationsmatrix eines Körpers am Ende eines Simulationsschrittes kann auf verschiedene Arten berechnet werden. Entweder wird eine Differentialgleichung für die Rotationsmatrix aufgestellt oder es wird eine Einheitsquaternion [Sho85] zur Berechnung der Rotation zum Zeitpunkt $t_0 + h$ verwendet, die anschließend in eine Rotationsmatrix konvertiert werden kann (siehe Anhang A). Die Beschreibung einer Rotation mit Euler-Winkeln hat die geringste Redundanz. Allerdings können einige Berechnungen nicht oder nur schwer mit diesen Winkeln durchgeführt werden [Sho85]. Daher werden sie im Folgenden nicht weiter betrachtet.

Rotationsmatrix Die Richtung eines Vektors $\mathbf{r}$ nach einem Simulationsschritt kann durch numerisches Lösen der Differentialgleichung

$$\dot{\mathbf{r}}(t) = \boldsymbol{\omega}(t) \times \mathbf{r}(t) \tag{3.5}$$

bestimmt werden. Wenn die Differentialgleichung 3.5 im lokalen Koordinatensystem eines Körpers für die Einheitsvektoren $\mathbf{e}_x(t_0) = (1,0,0)^T$, $\mathbf{e}_y(t_0) = (0,1,0)^T$ und $\mathbf{e}_z(t_0) = (0,0,1)^T$ gelöst wird, dann ergibt sich die Änderung der Rotation des Körpers durch die folgende Rotationsmatrix:

$$\triangle\mathbf{R} = \begin{pmatrix} \mathbf{e}_x(t_0+h)^T \\ \mathbf{e}_y(t_0+h)^T \\ \mathbf{e}_z(t_0+h)^T \end{pmatrix}.$$

Die Rotationsmatrix des Körpers für den Zeitpunkt t_0+h in Weltkoordinaten wird durch die Multiplikation der Matrix $\triangle\mathbf{R}$ mit der Rotationsmatrix vom Anfang des Simulationsschrittes bestimmt:

$$\mathbf{R}(t_0+h) = \triangle\mathbf{R} \cdot \mathbf{R}(t_0). \tag{3.6}$$

Im Allgemeinen weicht die Lösung beim numerischen Integrieren einer Differentialgleichung von der exakten Lösung leicht ab. Dies kann dazu führen, dass die resultierende Rotationsmatrix nicht mehr orthonormal ist. Aus diesem Grund müssen die Zeilenvektoren der Matrix $\mathbf{R}(t_0+h)$ nach dem Simulationsschritt orthogonalisiert und normiert werden (siehe Anhang C.3).

Quaternion Alternativ zur Beschreibung der Drehung eines Körpers mit einer Rotationsmatrix kann eine Einheitsquaternion verwendet werden. Eine Einheitsquaternion ist ein Vektor mit vier Elementen, der auf die Länge Eins normiert ist.

Die Drehung eines Körpers lässt sich während eines Simulationsschrittes aus den folgenden Gründen besser durch eine Einheitsquaternion repräsentieren als durch eine Rotationsmatrix. Eine Rotationsmatrix benötigt neun Elemente, um eine Drehung mit drei Freiheitsgraden zu beschreiben. Eine Einheitsquaternion benötigt dafür nur vier Elemente. Quaternionen haben somit deutlich weniger Redundanz. Dadurch wirken sich Ungenauigkeiten, die beim numerischen Integrieren auftreten, nicht so gravierend auf die Quaternionen aus, wie auf die Rotationsmatrizen. Nach der numerischen Integration müssen die Zeilenvektoren einer Rotationsmatrix orthogonalisiert und normiert werden. Eine Einheitsquaternion muss dagegen lediglich normiert werden, was wesentlich effizienter ist. Ein weiterer Vorteil der Quaternionen ist, dass sich die Verkettung von Rotationen mit ihnen günstiger berechnen lässt als mit Rotationsmatrizen (siehe Anhang A). Allerdings ist die Drehung von Punkten mit einer Quaternion rechenintensiver. Daher wird in der Simulation eine Quaternion verwendet, um die Rotation eines Körpers zu beschreiben, und nach jedem Simulationsschritt wird diese Quaternion in eine Rotationsmatrix umgewandelt, um Punkte schnell drehen zu können.

Die Änderung der Einheitsquaternion $\mathbf{q}$ eines Körpers während der Simulation wird durch die folgende Differentialgleichung beschrieben:

$$\dot{\mathbf{q}}(t) = \frac{1}{2}\,\boldsymbol{\omega}(t) \cdot \mathbf{q}(t). \tag{3.7}$$

Die Multiplikation $\boldsymbol{\omega}(t) \cdot \mathbf{q}(t)$ ist dabei eine Kurzschreibweise für die Multiplikation der Quaternion $(0,\ \omega_x(t),\ \omega_y(t),\ \omega_z(t))$ mit $\mathbf{q}(t)$. Durch numerisches Lösen der Differentialgleichung wird die Quaternion für den Zeitpunkt $t_0 + h$ bestimmt. Die resultierende Quaternion $\mathbf{q}(t_0 + h)$ muss anschließend noch normiert werden, denn im Allgemeinen hat $\mathbf{q}(t_0+h)$ wegen Ungenauigkeiten beim numerischen Integrieren nicht mehr die Länge Eins.

3.2 Simulation eines Gelenks

Bei der dynamischen Simulation eines freien Körpers wirken ausschließlich externe Kräfte auf den Körper. Wenn ein Körper mit einem anderen durch ein Gelenk verbunden ist, wirken zusätzlich interne Kräfte zwischen den Körpern, die das Gelenk zusammenhalten. In diesem Abschnitt wird erklärt, wie diese Kräfte bzw. die entsprechenden Impulse für verschiedene Gelenktypen berechnet werden.

Ein Gelenk verbindet zwei Körper, von denen mindestens einer dynamisch ist, durch eine Zwangsbedingung, die während der Simulation stets erfüllt sein muss. Dadurch werden die Freiheitsgrade der verbundenen Körper reduziert. Für jeden Freiheitsgrad, der aus dem System entfernt wird, wird eine skalare Gleichung benötigt. Es wird davon ausgegangen, dass die Zwangsbedingungen aller Gelenke des Systems zu einem bestimmten Zeitpunkt t_0 erfüllt sind. Damit die Bedingungen

nach einem Simulationsschritt mit der Zeitschrittweite h ebenfalls erfüllt werden, wird für jedes Gelenk ein Impuls berechnet, der die Geschwindigkeiten der verbundenen Körper entsprechend verändert.

Im Folgenden wird zunächst die allgemeine Vorgehensweise bei einem Simulationsschritt mit Gelenken beschrieben. Dann werden einige grundlegende Gleichungen und Matrizen eingeführt, die zur Berechnung der Impulse benötigt werden. Anschließend werden sechs Grundgelenke vorgestellt. Durch Kombination dieser Grundgelenke können dann weitere Gelenktypen realisiert werden. Außerdem wird die Simulation von Servomotoren und Federn erklärt. Diese Gelenktypen werden gesondert behandelt, da sie dem Mehrkörpersystem externe Drehmomente und Kräfte hinzufügen. Am Ende des Abschnitts werden Gelenke mit Geschwindigkeitsbedingungen vorgestellt, die für die Interaktion verwendet werden können.

3.2.1 Simulationsschritt mit einem Gelenk

Gelenke können *Positionsbedingungen* und *Geschwindigkeitsbedingungen* definieren. Eine Positionsbedingung (im Folgenden auch *Gelenkbedingung* genannt) beschränkt die Translation und die Rotation der verbundenen Körper, während eine Geschwindigkeitsbedingung eine Beschränkung für die Geschwindigkeiten der Körper beschreibt. In Abbildung 3.1 wird der Ablauf eines Simulationsschrittes vom Zeitpunkt t_0 nach $t_0 + h$ mit einem Gelenk, das beide Arten von Bedingungen definiert, veranschaulicht. Am Anfang des Simulationsschrittes werden Impulse für

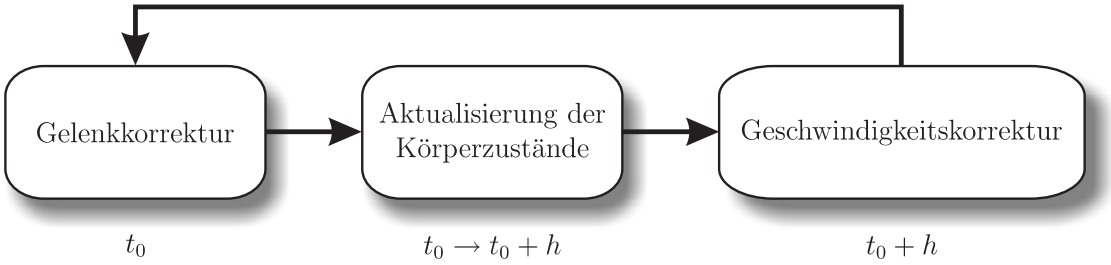


Abbildung 3.1: Ablauf eines Simulationsschrittes mit Gelenken

das Gelenk bestimmt und angewendet, so dass die Positionsbedingung des Gelenks nach dem Simulationsschritt aufgrund der veränderten Geschwindigkeiten der Körper erfüllt wird. Dieser Vorgang wird im Folgenden als *Gelenkkorrektur* bezeichnet. Anschließend werden die Körperzustände für den Zeitpunkt $t_0 + h$ berechnet. Dafür wird ein Simulationsschritt durchgeführt, wobei die verbundenen Körper als freie Körper (siehe Abschnitt 3.1) behandelt werden. Zum Schluss werden Impulse berechnet, die die Geschwindigkeiten der Körper so verändern, dass die Geschwindigkeitsbedingung des Gelenks erfüllt wird. Im Folgenden wird dies als *Geschwindigkeitskorrektur* bezeichnet.

Die Gelenkkorrektur wird in dieser Arbeit wie folgt durchgeführt (siehe Abbildung 3.2). Zunächst wird der Zustand des Gelenks nach dem Simulationsschritt

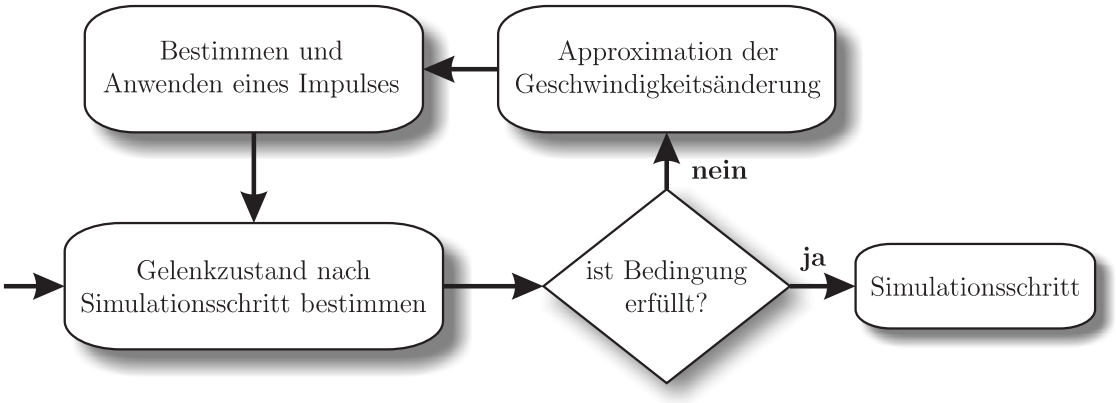


Abbildung 3.2: Ablauf der Gelenkkorrektur

bestimmt. Anschließend wird überprüft, ob dieser Zustand die Positionsbedingung erfüllt. Wenn dies nicht der Fall ist, wird ein Impuls berechnet, der die Geschwindigkeiten der verbundenen Körper so ändert, dass die Bedingung nach einem Simulationsschritt erfüllt wird. Da die Punkte eines Körpers im Allgemeinen eine nichtlineare Bewegung durchführen, wird in [WTF06] eine nichtlineare Gleichung aufgestellt, um den gesuchten Impuls zu berechnen. Diese Gleichung wird dann mit Newton-Iteration gelöst. In dieser Arbeit wird eine Vereinfachung verwendet. Es wird approximiert, wie sich die Geschwindigkeiten der verbundenen Körper ändern müssen, damit die Positionsbedingung erfüllt wird. Der Impuls, der diese Geschwindigkeitsänderung bewirkt, kann mit einer linearen Gleichung direkt bestimmt werden. Da es sich bei dem berechneten Impuls nur um eine Annäherung handelt, wird die Positionsbedingung nach Anwenden des Impulses im Normalfall nicht genau erfüllt, aber der Fehler reduziert. In einer iterativen Schleife werden dann weitere Impulse bestimmt, bis die Positionsbedingung mit einer vorgegebenen Genauigkeit erfüllt ist. Anschließend kann ein Simulationsschritt durchgeführt werden.

Durch die Approximation der Geschwindigkeitsänderung ergibt sich eine lineare Gleichung. Eine lineare Gleichung hat den Vorteil, dass für sie sehr schnell eine exakte Lösung bestimmt werden kann. Außerdem können mehrere Gleichungen als lineares Gleichungssystem zusammengefasst und gemeinsam gelöst werden (siehe Abschnitt 3.3.2). Wegen der Approximation muss ein Korrekturimpuls iterativ bestimmt werden. Es wurden verschiedene Simulationen mit einer relativ großen Zeitschrittweite von 0,04 s durchgeführt. Dies entspricht 25 Bildern pro Sekunde und ist in etwa die Bildwiederholrate, die nötig ist, um dem menschlichen Auge eine kontinuierliche Bewegung vorzutäuschen. Bei den Simulationen wurde gefordert, dass die Bedingung des Gelenks innerhalb einer sehr kleinen Toleranz von 10^{-6} m

erfüllt wird. Nur wenn die Körper sehr hohe Geschwindigkeiten hatten, wurden bei den Simulationen mehr als zwei Iterationen benötigt, um den gesuchten Impuls zu bestimmen.

Die Geschwindigkeitskorrektur wird nach dem Simulationsschritt durchgeführt. Im Gegensatz zur Gelenkkorrektur kann hier die Geschwindigkeitsänderung, die zur Erfüllung der Bedingung benötigt wird, exakt bestimmt werden. Der zugehörige Impuls kann anschließend durch Lösen einer linearen Gleichung berechnet werden. Nach Anwenden des Impulses ist die Geschwindigkeitsbedingung sofort erfüllt.

3.2.2 Grundlagen

Die Geschwindigkeit eines Punktes **a**, der fest mit einem Körper k verbunden ist, lässt sich durch die Schwerpunkt- und die Winkelgeschwindigkeit des Körpers bestimmen. Die Punktgeschwindigkeit von **a** ist

$$\mathbf{u}_a = \mathbf{v}_k + \boldsymbol{\omega}_k \times \mathbf{r}_{as}, \tag{3.8}$$

wobei $\mathbf{r}_{as} = \mathbf{a} - \mathbf{s}_k$ der Ortsvektor vom Schwerpunkt des Körpers zu **a** ist. Wenn an einem beliebigen Punkt **b** des Körpers k ein Impuls **p** angewendet wird, ändert sich die Punktgeschwindigkeit von **a** (siehe Abbildung 3.3). Die Geschwindigkeitsände-

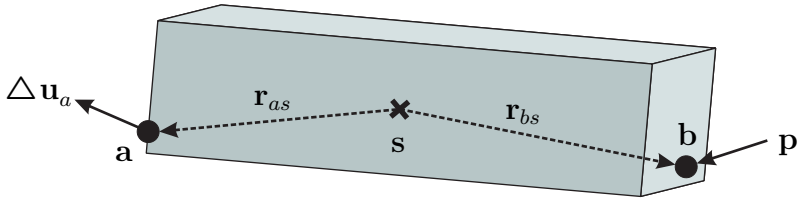


Abbildung 3.3: Änderung der Punktgeschwindigkeit bei Anwenden eines Impulses

rung $\triangle\mathbf{u}_a$ des Punktes **a** kann bestimmt werden, indem zunächst die Änderung der Schwerpunktgeschwindigkeit des Körpers

$$\triangle\mathbf{v}_k = \frac{1}{m_k}\,\mathbf{p}$$

berechnet wird, und anschließend die Änderung der Winkelgeschwindigkeit

$$\triangle\boldsymbol{\omega}_k = \mathbf{J}_k^{-1}\,(\mathbf{r}_{bs} \times \mathbf{p}).$$

Dabei ist $\mathbf{r}_{bs}$ der Ortsvektor vom Schwerpunkt des Körpers zum Punkt **b**. Setzt man die Werte $\triangle\mathbf{v}_k$ und $\triangle\boldsymbol{\omega}_k$ in Gleichung 3.8 ein, so ergibt sich die Änderung

der Punktgeschwindigkeit

$$\begin{aligned}\triangle \mathbf{u}_a &= \triangle \mathbf{v}_k + \triangle \boldsymbol{\omega}_k \times \mathbf{r}_{as} \\ &= \frac{1}{m_k}\,\mathbf{p} + \left(\mathbf{J}_k^{-1}\,(\mathbf{r}_{bs} \times \mathbf{p})\right) \times \mathbf{r}_{as} \\ &= \frac{1}{m_k}\,\mathbf{p} - \mathbf{r}_{as} \times \left(\mathbf{J}_k^{-1}\,\mathbf{r}_{bs}^*\,\mathbf{p}\right) \\ &= \frac{1}{m_k}\,\mathbf{p} - \mathbf{r}_{as}^*\,\mathbf{J}_k^{-1}\,\mathbf{r}_{bs}^*\,\mathbf{p},\end{aligned}$$

wobei $\mathbf{r}_{as}^*$ und $\mathbf{r}_{bs}^*$ die Kreuzproduktmatrizen der Ortsvektoren (siehe Anhang C.1) sind. Durch eine Umformung mit der Einheitsmatrix $\mathbf{E}_3 \in \mathbb{R}^3$ wird die Matrix

$$\mathbf{K}_{a,b}(t) := \begin{cases} \frac{1}{m_k}\,\mathbf{E}_3 - \mathbf{r}_{as}^*(t)\,\mathbf{J}_k^{-1}(t)\,\mathbf{r}_{bs}^*(t) & \text{falls Körper } k \text{ dynamisch ist} \\ \mathbf{0} & \text{sonst} \end{cases} \tag{3.9}$$

definiert. Mit dieser Matrix kann die Änderung der Geschwindigkeit eines Körperpunktes $\mathbf{a}$ berechnet werden, wenn ein Impuls in einem Punkt $\mathbf{b}$ des Körpers wirkt:

$$\triangle \mathbf{u}_a(t) = \mathbf{K}_{a,b}(t)\,\mathbf{p}.$$

Wenn ein Drehimpuls $\mathbf{l}$ auf einen dynamischen Körper angewendet wird, dann ändert sich die Winkelgeschwindigkeit dieses Körpers (siehe Abbildung 3.4). Diese

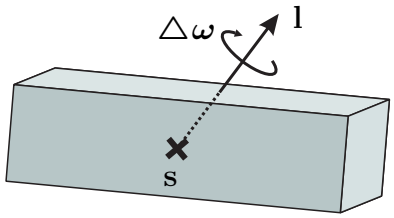


Abbildung 3.4: Änderung der Winkelgeschwindigkeit bei Anwenden eines Drehimpulses

Geschwindigkeitsänderung kann mit Hilfe der Inversen des Trägheitstensors bestimmt werden. Damit dynamische und statische Körper einheitlich behandelt werden können, wird die folgende Matrix definiert:

$$\mathbf{L}_k(t) := \begin{cases} \mathbf{J}_k^{-1}(t) & \text{falls Körper } k \text{ dynamisch ist} \\ \mathbf{0} & \text{sonst.} \end{cases} \tag{3.10}$$

Die Winkelgeschwindigkeit eines Körpers k ändert sich um

$$\triangle \boldsymbol{\omega}_k(t) = \mathbf{L}_k(t)\,\mathbf{l},$$

wenn ein Drehimpuls $\mathbf{l}$ auf den Körper wirkt.

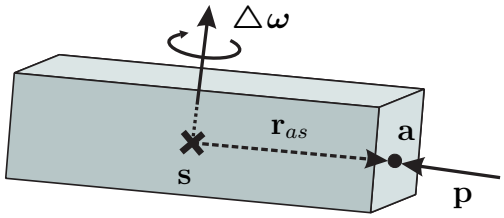


Abbildung 3.5: Änderung der Winkelgeschwindigkeit bei Anwenden eines Impulses

Wird an einem Punkt **a** eines Körpers ein Impuls **p** angewendet, dann wird die Winkelgeschwindigkeit des Körpers durch den Drehimpuls

$$\mathbf{l} = \mathbf{r}_{as} \times \mathbf{p}$$

verändert, wobei $\mathbf{r}_{as}$ der Ortsvektor vom Schwerpunkt des Körpers zu **a** ist. Auf diese Weise lässt sich zu einem Impuls der zugehörige Drehimpuls bestimmen. Dadurch lässt sich die Matrix

$$\mathbf{W}_{k,a}(t) := \begin{cases} \mathbf{J}_k^{-1}(t)\,\mathbf{r}_{as}^*(t) & \text{falls Körper } k \text{ dynamisch ist} \\ \mathbf{0} & \text{sonst} \end{cases} \tag{3.11}$$

definieren. Mit dieser Matrix kann die Änderung der Winkelgeschwindigkeit eines Körpers k berechnet werden, wenn in einem Punkt **a** dieses Körpers ein Impuls **p** wirkt (siehe Abbildung 3.5):

$$\triangle\boldsymbol{\omega}_k(t) = \mathbf{W}_{k,a}(t)\,\mathbf{p}.$$

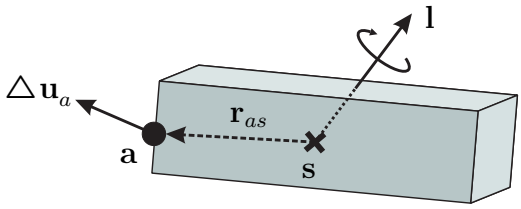


Abbildung 3.6: Änderung der Punktgeschwindigkeit bei Anwenden eines Drehimpulses

Umgekehrt kann die Änderung der Geschwindigkeit eines Körperpunktes **a** bestimmt werden, wenn ein Drehimpuls **l** auf den zugehörigen Körper wirkt (siehe Abbildung 3.6). Dafür wird die folgende Matrix definiert:

$$\mathbf{U}_{a,k}(t) := \begin{cases} -\mathbf{r}_{as}^*(t)\,\mathbf{J}_k^{-1}(t) & \text{falls Körper } k \text{ dynamisch ist} \\ \mathbf{0} & \text{sonst.} \end{cases} \tag{3.12}$$

Durch Multiplikation dieser Matrix mit dem Drehimpuls ergibt sich die Änderung der Punktgeschwindigkeit:

$$\triangle\mathbf{u}_a(t) = \mathbf{U}_{a,k}(t)\,\mathbf{l}.$$

3.2.3 Grundgelenke

In dieser Arbeit wird zwischen zwei Arten von Grundgelenken unterschieden: Gelenke mit Translationsbedingungen und Gelenke mit Rotationsbedingungen. Die Translationsbedingungen entfernen ausschließlich Translationsfreiheitsgrade und die Rotationsbedingungen nur Rotationsfreiheitsgrade. Jedes Grundgelenk hat eine Gelenk- und eine Geschwindigkeitsbedingung. Die Gelenkbedingung entfernt die entsprechenden Freiheitsgrade, während die Geschwindigkeitsbedingung bei den vorgestellten Grundgelenken dafür sorgt, dass die zu den entfernten Freiheitsgraden zugehörigen, relativen Geschwindigkeiten Null sind. Die Geschwindigkeitskorrektur bei den Grundgelenken dient zur Verbesserung der Genauigkeit der Simulation. Dies lässt sich mit Hilfe eines Beispiels veranschaulichen. Angenommen zwei Körper sind in einem Punkt miteinander verbunden und auf die Körper wirken keine externen Kräfte oder Drehmomente. Wenn die Position von einem Körper verschoben wird, bricht die Verbindung auf. Die Gelenkkorrektur berechnet daraufhin einen Impuls für beide Körper, um die Gelenkbedingung wieder zu erfüllen. Nach dem Simulationsschritt ist die Gelenkbedingung durch den Impuls erfüllt. Allerdings haben die Körper durch die Wirkung des Impulses in dem gemeinsamen Punkt jetzt unterschiedliche Geschwindigkeiten. Dies bedeutet, dass die Verbindung im nächsten Simulationsschritt erneut aufbricht und wieder eine Gelenkkorrektur benötigt wird. Die zweite Gelenkkorrektur sorgt dafür, dass die beiden Körper im gemeinsamen Punkt die gleiche Geschwindigkeit haben. Allerdings haben die beiden Körper zwischen den beiden Simulationsschritten keine korrekten Geschwindigkeiten. Der Fehler bei den Geschwindigkeiten kann zu fehlerhaften Ergebnissen bei der Behandlung von Kollisionen und Kontakten (siehe Kapitel 5) führen. Durch die Definition einer Geschwindigkeitsbedingung für den gemeinsamen Punkt kann dies verhindert werden. Wird die Bedingung aufgestellt, dass die Körper im gemeinsamen Punkt die gleiche Geschwindigkeit haben müssen, dann wird diese Bedingung nach jedem Simulationsschritt durch eine Geschwindigkeitskorrektur erfüllt. Dadurch haben die beiden Körper nach jedem Simulationsschritt korrekte Geschwindigkeiten und ein erneutes Aufbrechen der Verbindung wird verhindert. Im Allgemeinen wirken externe Kräfte auf die Körper in der Simulation. Dadurch muss in jedem Simulationsschritt eine Gelenkkorrektur vorgenommen werden. Wenn es bei der Simulation nicht so sehr auf die Genauigkeit der Ergebnisse ankommt, sondern Schnelligkeit gefragt ist, kann auf die Geschwindigkeitskorrektur verzichtet werden.

3.2.3.1 Gelenke mit Translationsbedingungen

Kugelgelenk Ein Kugelgelenk (siehe Abbildung 3.7) verbindet zwei Körper in einem Punkt. Die beiden Körper haben damit einen gemeinsamen Punkt, um den sie rotieren können. Das bedeutet, dass das Gelenk drei Translationsfreiheitsgrade

des Systems entfernt. Für die Simulation wird an der Position des Kugelgelenks

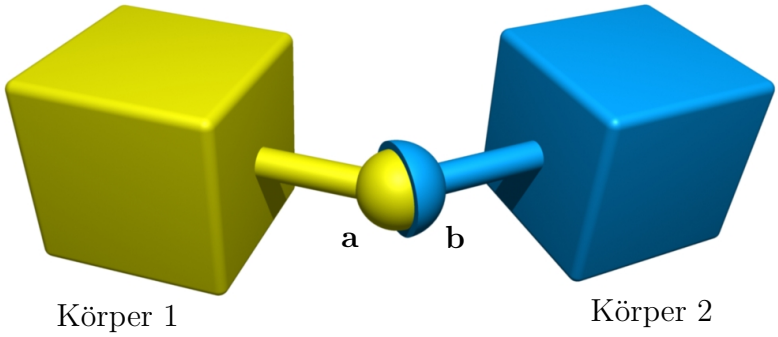


Abbildung 3.7: Kugelgelenk

ein Punkt **a** zum ersten Körper und ein Punkt **b** zum zweiten Körper hinzugefügt. Diese Punkte werden als *Gelenkpunkte* bezeichnet. Sie sind fest mit dem jeweiligen Körper verbunden und bewegen sich mit ihm. Gelenkpunkte werden in lokalen Koordinaten des jeweiligen Körpers gespeichert und in jedem Simulationsschritt einmal mit Gleichung 3.1 in Weltkoordinaten transformiert. Dadurch lässt sich ein numerischer Drift der Punkte verhindern.

Der Abstand der beiden Punkte des Kugelgelenks muss während der Simulation immer Null sein[1] bzw. innerhalb eines Toleranzbereichs liegen. Dies ist die Gelenkbedingung des Kugelgelenks. Die Gelenkpunkte müssen also die Bedingung

$$|\mathbf{a}(t) - \mathbf{b}(t)| \leq \varepsilon_d$$

erfüllen, wobei ε_d die maximal zulässige Distanz zwischen den beiden Punkten ist. Wenn auf die beiden Körper des Gelenks keine inneren Kräfte wirken, dann driften die Gelenkpunkte im Allgemeinen auseinander (siehe Abbildung 3.8). Am Ende eines Simulationsschrittes haben die beiden Punkte dann einen Abstand, der durch den Vektor

$$\mathbf{d}(t_0 + h) = \mathbf{b}(t_0 + h) - \mathbf{a}(t_0 + h)$$

beschrieben wird. Die Position eines Gelenkpunktes **a** am Ende des Simulationsschrittes kann berechnet werden, indem zunächst die Differentialgleichung 3.5 für den Ortsvektor $\mathbf{r}_{sa}(t_0) = \mathbf{a}(t_0) - \mathbf{s}(t_0)$ gelöst wird. Dadurch erhält man die Richtung des Vektors zum Zeitpunkt $t_0 + h$. Mit Gleichung 3.3 kann die neue Position des Schwerpunktes bestimmt werden. Die Position des Gelenkpunktes am Ende des Simulationsschrittes ist dann $\mathbf{a}(t_0 + h) = \mathbf{r}_{sa}(t_0 + h) + \mathbf{s}(t_0 + h)$. Damit die

[1] Wenn man fordert, dass der Abstand einen bestimmten Wert d annimmt anstatt Null, dann kann man ein Distanzgelenk simulieren. Dieses sorgt dafür, dass die beiden Gelenkpunkte immer den gleichen Abstand d zueinander halten. Auf die gleiche Weise kann ein bestimmter Abstand bei jedem Gelenk mit Translationsbedingungen eingefügt werden. Bei den restlichen Gelenken in diesem Abschnitt wird es daher nicht mehr gesondert erwähnt.

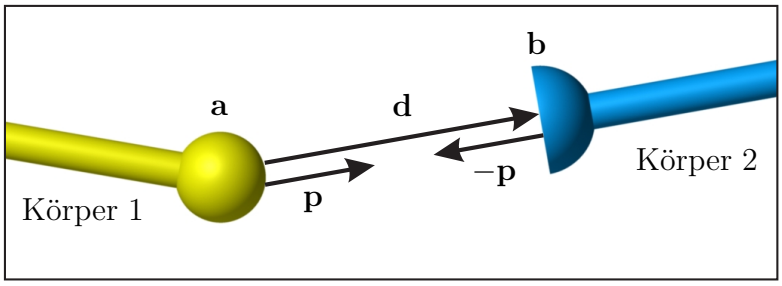


Abbildung 3.8: Gelenkkorrektur

Gelenkpunkte während des Simulationsschrittes nicht auseinander driften, wird ein Impuls $\mathbf{p}_d$ berechnet. Dieser Impuls wird am Anfang des Simulationsschrittes positiv auf den ersten Punkt und negativ auf den zweiten Punkt angewendet. Durch die Geschwindigkeitsänderung, die die beiden entgegengesetzten Impulse bewirken, muss der Abstand $\mathbf{d}(t_0 + h)$ in der Zeitspanne h verschwinden. Da die beiden Impulse $\mathbf{p}_d$ und $-\mathbf{p}_d$ den gleichen Betrag, aber entgegengesetzte Richtungen haben, ist die Impulserhaltung des Systems gewährleistet. Im Allgemeinen ist die Bewegung der beiden Gelenkpunkte relativ zueinander nichtlinear. Um die Impulse dennoch mit einer linearen Gleichung bestimmen zu können, wird eine Vereinfachung verwendet. Es wird ein Paar von Impulsen $\mathbf{p}$ und $-\mathbf{p}$ bestimmt, die die relative Punktgeschwindigkeit der beiden Gelenkpunkte um $\mathbf{d}(t_0 + h)/h$ verändern. Der Impuls $\mathbf{p}$ entspricht nur dann dem gesuchten Impuls $\mathbf{p}_d$, wenn die relative Bewegung der beiden Punkte linear ist. Andernfalls wird durch $\mathbf{p}$ und $-\mathbf{p}$ die Distanz zwischen den beiden Gelenkpunkten zum Zeitpunkt $t_0 + h$ reduziert (Beweis: siehe Anhang D.3), aber nicht vollständig eliminiert. Im Falle einer nichtlinearen Bewegung werden Impulse in einer iterativen Schleife bestimmt, die den Abstand $|\mathbf{d}(t_0 + h)|$ schrittweise verringern. In jedem Iterationsschritt muss der Wert $\mathbf{d}(t_0 + h)$ aktualisiert werden. Die Schleife endet, wenn der Abstand der Gelenkpunkte nach dem Simulationsschritt kleiner als der Toleranzwert ε_d ist.

Die Änderung der Geschwindigkeit eines Gelenkpunktes, wenn an seiner Position ein Impuls auf den Körper angewendet wird, lässt sich mit Hilfe der in Abschnitt 3.2.2 definierten Matrix $\mathbf{K}$ beschreiben. Die Impulse, die die relative Punktgeschwindigkeit der beiden Gelenkpunkte um $\mathbf{d}(t_0 + h)/h$ verändern, werden durch das Lösen der folgenden Gleichung bestimmt:

$$\mathbf{K}_{a,a}(t_0)\,\mathbf{p} - \mathbf{K}_{b,b}(t_0)\,(-\mathbf{p}) = \frac{1}{h}\mathbf{d}(t_0 + h). \tag{3.13}$$

Die Matrix $\mathbf{K}(t_0) := \mathbf{K}_{a,a}(t_0) + \mathbf{K}_{b,b}(t_0)$ ist konstant zum Zeitpunkt t_0, regulär, symmetrisch und positiv definit (Beweis in [Mir96b]). Da die Matrix regulär ist, kann die Gleichung durch Invertieren von $\mathbf{K}(t_0)$ gelöst werden, um den Impuls $\mathbf{p}$ zu bestimmen:

$$\mathbf{p} = \frac{1}{h}\,\mathbf{K}(t_0)^{-1}\,\mathbf{d}(t_0 + h).$$

Wenn die Impulse $\mathbf{p}$ und $-\mathbf{p}$ zum Zeitpunkt t_0 auf die Gelenkpunkte angewendet werden, ändern sich die Geschwindigkeiten der Körper sofort. Dadurch ergibt sich ein neuer Abstand $\mathbf{d}(t_0+h)$. Ist für diesen Abstand die Bedingung $|\mathbf{d}(t_0+h)| \leq \varepsilon_d$ erfüllt, dann wird ein Simulationsschritt durchgeführt, bei dem die Körper als freie Körper behandelt werden (siehe Abschnitt 3.1). Die geänderten Geschwindigkeiten der Körper führen dazu, dass die Gelenkbedingung zum Zeitpunkt $t_0 + h$ erfüllt ist.

Die zweite Bedingung, die für ein Kugelgelenk erfüllt sein muss, ist die Geschwindigkeitsbedingung. Diese fordert, dass die Geschwindigkeitsdifferenz der beiden Gelenkpunkte immer Null sein bzw. innerhalb eines Toleranzbereichs liegen muss. Die Punkte müssen während der Simulation die Bedingung

$$|\mathbf{u}_a(t) - \mathbf{u}_b(t)| \leq \varepsilon_v$$

erfüllen, wobei ε_v die maximal zulässige Geschwindigkeitsdifferenz ist. Im Allgemeinen sind die Geschwindigkeiten der beiden Gelenkpunkte nach einem Simulationsschritt nicht gleich. Deshalb muss ein Impuls $\mathbf{p}_v$ berechnet werden, der dies korrigiert. Dieser Impuls muss die relative Punktgeschwindigkeit $\triangle\mathbf{u} = \mathbf{u}_b - \mathbf{u}_a$ zum Zeitpunkt $t_0 + h$ eliminieren und somit die Gleichung

$$\mathbf{K}(t_0 + h)\,\mathbf{p}_v = \triangle\mathbf{u}(t_0 + h) \tag{3.14}$$

erfüllen. Der gesuchte Impuls kann durch Invertieren der Matrix $\mathbf{K}(t_0 + h)$ bestimmt werden. Wenn der berechnete Impuls positiv auf den Punkt $\mathbf{a}$ und negativ auf den Punkt $\mathbf{b}$ angewendet wird, ändern sich die Punktgeschwindigkeiten und die Geschwindigkeitsbedingung wird erfüllt. Die Berechnung der Impulse bei der Geschwindigkeitskorrektur muss nicht iterativ durchgeführt werden, da die Geschwindigkeitsdifferenz der Gelenkpunkte und die entsprechenden Impulse genau berechnet werden können. Damit ist ein Simulationsschritt für zwei Körper, die mit einem Kugelgelenk verbunden sind, abgeschlossen, da für den Zeitpunkt t_0+h alle Bedingungen des Gelenks erfüllt sind.

Die Matrix $\mathbf{K}(t)$ ist konstant für einen Zeitpunkt, daher muss ihre Inverse nur einmal pro Simulationsschritt berechnet werden. Mit der Inversen kann ein Impuls bestimmt werden, um die Geschwindigkeitsbedingung zu erfüllen und im nächsten Simulationsschritt wird mit der gleichen Matrix der Impuls für die Gelenkbedingung berechnet. Außerdem ist die Matrix $\mathbf{K}(t)$ symmetrisch, deshalb kann die Inverse schneller bestimmt werden als bei allgemeinen Matrizen (siehe Anhang C).

Geradengelenk Ein Geradengelenk (siehe Abbildung 3.9) ist ein Kugelgelenk, das sich auf einer Geraden frei bewegen darf. Dieses Gelenk entfernt zwei Translationsfreiheitsgrade des Systems. Am Anfang der Simulation wird das Gelenk definiert durch einen gemeinsamen Punkt der beiden Körper und einen normierten Vektor $\mathbf{x}$, der die Richtung der Geraden bestimmt. Die Gerade wird mit dem

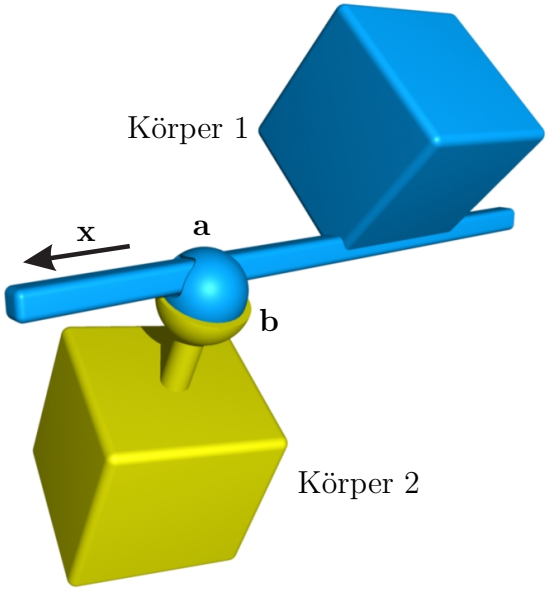


Abbildung 3.9: Geradengelenk

ersten Körper fest verbunden und bewegt sich während der Simulation mit ihm. Um dies zu erreichen, wird der gemeinsame Punkt als Aufpunkt der Geraden $\mathbf{a}_0$ und der Vektor $\mathbf{x}$ in das lokale Koordinatensystem des Körpers transformiert und so gespeichert. In jedem Simulationsschritt wird dann die Gerade für die Berechnungen einmal in Weltkoordinaten transformiert. Zum zweiten Körper wird nur der gemeinsame Punkt als Gelenkpunkt $\mathbf{b}$ hinzugefügt. Während der Simulation muss für jeden Zeitpunkt t gelten, dass der Gelenkpunkt $\mathbf{b}(t)$ des zweiten Körpers auf der Geraden $\mathbf{a}_0(t) + \lambda\mathbf{x}(t)$, $\lambda \in \mathbb{R}$ des ersten Körpers liegt.

Die Gelenkbedingung des Geradengelenks kann aufgestellt werden, indem der Abstandsvektor der Gelenkpunkte $\mathbf{a}_0$ und $\mathbf{b}$ auf eine Ebene senkrecht zu $\mathbf{x}$ projiziert wird. Dazu werden zunächst zwei linear unabhängige Vektoren $\mathbf{y}$ und $\mathbf{z}$ bestimmt, die diese Ebene aufspannen. Sei $\mathbf{w} \in \mathbb{R}^3$ ein beliebiger normierter Vektor, für den $|\mathbf{x} \cdot \mathbf{w}| \neq 1$ gilt. Das heißt, dass $\mathbf{x}$ und $\mathbf{w}$ linear unabhängig sind. Dann ergeben sich die beiden Vektoren, die die gesuchte Ebene aufspannen, folgendermaßen:

$$\begin{aligned} \mathbf{y} &= \mathbf{x} \times \mathbf{w} \\ \mathbf{z} &= \mathbf{x} \times \mathbf{y}. \end{aligned}$$

Mit diesen beiden Vektoren wird die Projektionsmatrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{y}^T \\ \mathbf{z}^T \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

definiert, mit der der dreidimensionale Abstandsvektor der beiden Gelenkpunkte

auf die Ebene abgebildet werden kann. Auf diese Weise lässt sich die Gelenkbedingung des Geradengelenks aufstellen:

$$|\mathbf{P}\,(\mathbf{a}_0(t) - \mathbf{b}(t))| \leq \varepsilon_d.$$

Um die Gelenkbedingung während der Simulation zu erfüllen, wird in jedem Simulationsschritt ein Impuls $\mathbf{p}_d$ berechnet, der verhindert, dass die beiden auf die Ebene projizierten Gelenkpunkte auseinander driften. Dieser Impuls muss an der aktuellen Verbindungsstelle des Gelenks auf beide Körper in entgegengesetzte Richtungen angewendet werden. Dafür muss der Lotpunkt $\mathbf{a}$ des Gelenkpunktes $\mathbf{b}$ auf der Geraden $\mathbf{a}_0(t) + \lambda\mathbf{x}(t)$ bestimmt werden:

$$\mathbf{a} = \mathbf{a}_0 + ((\mathbf{b} - \mathbf{a}_0)\,\mathbf{x})\,\mathbf{x}.$$

Berechnet man die Projektionsmatrix $\mathbf{P}$ für den Zeitpunkt t_0 und projiziert mit ihr die Gleichung 3.13 auf die Ebene, dann ergibt sich die Gleichung für den Impuls im Koordinatensystem der Ebene $\mathbf{p}'$, der den Abstand zwischen den Punkten $\mathbf{a}$ und $\mathbf{b}$ reduziert:

$$\mathbf{P}\,(\mathbf{K}_{a,a}(t_0) + \mathbf{K}_{b,b}(t_0))\,\mathbf{P}^T\,\mathbf{p}' = \frac{1}{h}\,\mathbf{P}\,\mathbf{d}(t_0 + h). \tag{3.15}$$

Die Matrix $\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T$ ist konstant zum Zeitpunkt t_0, regulär, symmetrisch und positiv definit.

Beweis: Da $\mathbf{P}$ und $\mathbf{K}(t_0)$ konstant sind, ist auch die Matrix $\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T$ konstant. Außerdem ist die Matrix symmetrisch, da $\mathbf{K}(t_0)$ symmetrisch ist und somit $\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T = (\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T)^T$ gilt. Sei $\mathbf{w} \in \mathbb{R}^2$ ein beliebiger Vektor für den $\mathbf{w} \neq \mathbf{0}$ gilt, dann gilt auch $\mathbf{P}^T\,\mathbf{w} \neq \mathbf{0}$. Für den Vektor $\mathbf{w}$ ist

$$\mathbf{w}^T\,\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T\,\mathbf{w} = (\mathbf{P}^T\,\mathbf{w})^T\,\mathbf{K}(t_0)\,(\mathbf{P}^T\,\mathbf{w})$$

größer als Null, da die Matrix $\mathbf{K}(t_0)$ positiv definit ist. Damit ist auch $\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T$ positiv definit und folglich regulär. □

Die Gleichung 3.15 kann daher durch Invertierung der Matrix $\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T$ nach $\mathbf{p}'$ aufgelöst werden. In einer iterativen Schleife werden Impulse durch Lösen dieser Gleichung bestimmt. Nachdem ein Impuls in einem Iterationsschritt berechnet wurde, wird er mit Hilfe der Projektionsmatrix in das Weltkoordinatensystem transformiert:

$$\mathbf{p} = \mathbf{P}^T\,\mathbf{p}'.$$

Anschließend wird der Impuls $\mathbf{p}$ zum Zeitpunkt t_0 positiv auf $\mathbf{a}$ und negativ auf $\mathbf{b}$ angewendet. Die Schleife endet, wenn die Bedingung $|\mathbf{P}\,(\mathbf{a}_0(t) - \mathbf{b}(t))| = |\mathbf{d}(t_0 + h)| \leq \varepsilon_d$ erfüllt ist. Nach einem Simulationsschritt wird durch die Wirkung der Impulse die Gelenkbedingung erfüllt.

Die Geschwindigkeitsbedingung des Geradengelenks kann durch Projektion der Geschwindigkeitsdifferenz zwischen dem aktuellen Lotpunkt **a** und dem Gelenkpunkt **b** auf die Ebene senkrecht zu **x** aufgestellt werden:

$$|\mathbf{P}(\mathbf{u}_a(t) - \mathbf{u}_b(t))| \leq \varepsilon_v.$$

Da die Projektionsmatrix abhängig von der aktuellen Lage des Richtungsvektors **x** in Weltkoordinaten ist, muss sie für den Zeitpunkt t_0+h neu berechnet werden. Die gleiche Projektionsmatrix kann für die Geschwindigkeitsbedingung und die Gelenkbedingung des nächsten Simulationsschrittes verwendet werden, da sie für einen Zeitpunkt konstant ist. Dadurch muss sie in jedem Simulationsschritt nur einmal bestimmt werden. Das gleiche gilt für den Lotpunkt **a**, der für jeden Zeitpunkt einmal neu berechnet werden muss. Um den Impuls $\mathbf{p}_v$ zu berechnen, der zum Erfüllen der Geschwindigkeitsbedingung benötigt wird, wird die Gleichung 3.14 für den Lotpunkt **a** und den Gelenkpunkt **b** aufgestellt und auf die Ebene projiziert:

$$\mathbf{P}\,\mathbf{K}(t_0 + h)\,\mathbf{P}^T\,\mathbf{p}'_v = \mathbf{P}\,\triangle\mathbf{u}(t_0 + h). \tag{3.16}$$

Anschließend kann der zweidimensionale Impuls $\mathbf{p}'_v$ durch Invertieren der Matrix $\mathbf{P}\,\mathbf{K}(t_0+h)\,\mathbf{P}^T$ berechnet und mit Hilfe der Projektionsmatrix in Weltkoordinaten transformiert werden. Die Inverse der Matrix muss wie beim Kugelgelenk in jedem Simulationsschritt nur einmal bestimmt werden.

Ebenengelenk Ein Ebenengelenk (siehe Abbildung 3.10) ist ein Kugelgelenk, das sich in einer Ebene frei bewegen kann. Ein solches Gelenk entfernt nur einen Translationsfreiheitsgrad des Systems. Zu Beginn der Simulation wird das Gelenk definiert durch einen gemeinsamen Punkt der beiden verbundenen Körper und die Normale $\mathbf{n} = \mathbf{x} \times \mathbf{y}$ der Ebene. Im Folgenden wird davon ausgegangen, dass diese Normale in normierter Form vorliegt. Die Ebene wird mit dem ersten Körper fest verbunden, indem der gemeinsame Punkt als Aufpunkt $\mathbf{a}_0$ der Ebene und die Normale **n** in lokalen Koordinaten des Körpers gespeichert werden. In jedem Simulationsschritt muss die Ebene dann einmal in das Weltkoordinatensystem zurück transformiert werden. Außerdem wird der gemeinsame Punkt als Gelenkpunkt **b** zum zweiten Körper hinzugefügt. Der Punkt **b** des zweiten Körpers muss sich während der Simulation stets in der Ebene $n_x\,x + n_y\,y + n_z\,z - \mathbf{a}_0\,\mathbf{n} = 0$ des ersten Körpers befinden.

Der Abstand der beiden Gelenkpunkte $\mathbf{a}_0$ und **b** muss auf die Normale der Ebene projiziert werden, um die Gelenkbedingung des Ebenengelenks zu formulieren. Mit der Normalen lässt sich die Projektionsmatrix

$$\mathbf{P} = \left(\mathbf{n}^T\right) \in \mathbb{R}^{1\times 3}$$

definieren. Das Produkt dieser Projektionsmatrix mit dem Abstandsvektor der beiden Gelenkpunkte ergibt dann die Gelenkbedingung:

$$|\mathbf{P}(\mathbf{a}_0(t) - \mathbf{b}(t))| \leq \varepsilon_d.$$

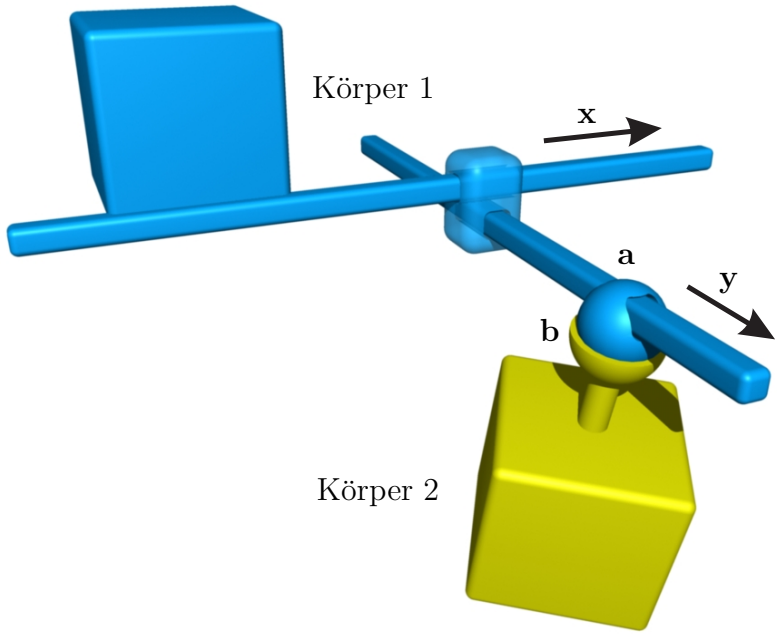


Abbildung 3.10: Ebenengelenk

Die Gelenkbedingung ist die gleiche wie beim Geradengelenk mit einer anderen Projektionsmatrix. Die beiden auf die Normale projizierten Gelenkpunkte dürfen während der Simulation nicht auseinander driften. Der Impuls, der dies verhindert, muss an der aktuellen Verbindungsstelle des Gelenks in entgegengesetzte Richtungen auf die beiden Körper wirken. Dafür muss zunächst der Lotpunkt $\mathbf{a}$ des Gelenkpunktes $\mathbf{b}$ auf der Ebene des ersten Körpers berechnet werden:

$$\mathbf{a} = \mathbf{b} - ((\mathbf{b} - \mathbf{a}_0)\ \mathbf{n})\ \mathbf{n}.$$

Die Gelenkbedingung nach dem Simulationsschritt wird erfüllt, indem iterativ Impulse durch Lösen der Gleichung 3.15 bestimmt werden. Dabei wird die Projektionsmatrix des Ebenengelenks verwendet. Jeder Impuls wird nach der Berechnung in Weltkoordinaten transformiert und auf die Punkte $\mathbf{a}$ und $\mathbf{b}$ in entgegengesetzte Richtungen angewendet.

Die Geschwindigkeitsbedingung des Ebenengelenks ergibt sich durch die Projektion der Geschwindigkeitsdifferenz zwischen dem aktuellen Lotpunkt $\mathbf{a}$ und dem Gelenkpunkt $\mathbf{b}$:

$$|\mathbf{P}(\mathbf{u}_a(t) - \mathbf{u}_b(t))| \leq \varepsilon_v.$$

Da dies, abgesehen von der Projektionsmatrix, die gleiche Bedingung wie die des Geradengelenks ist, kann der Impuls $\mathbf{p}'_v$, der die Geschwindigkeitsdifferenz der beiden Punkte auf der projizierten Gerade eliminiert, durch Lösen der Gleichung 3.16 bestimmt werden. Der berechnete Impuls wird in Weltkoordinaten transformiert, bevor er auf die beiden Punkte entsprechend angewendet wird.

Abschließend lässt sich sagen, dass die Impulse für alle Gelenke mit Translationsbedingungen durch Lösen der Gleichungen 3.15 und 3.16 berechnet werden können, wenn als Projektionsmatrix des Kugelgelenks die dreidimensionale Einheitsmatrix verwendet wird.

3.2.3.2 Gelenke mit Rotationsbedingungen

Translationsgelenk Zwei Körper, die mit einem Translationsgelenk (siehe Abbildung 3.11) verbunden sind, dürfen sich relativ zueinander nicht drehen. Das

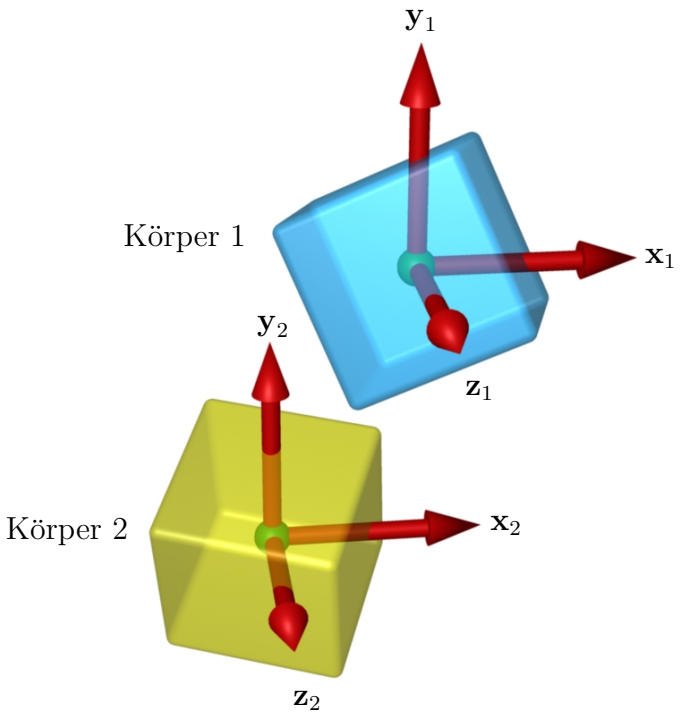


Abbildung 3.11: Translationsgelenk

bedeutet, dass während der Simulation die Bedingungen $\mathbf{x}_1 = \mathbf{x}_2$, $\mathbf{y}_1 = \mathbf{y}_2$ und $\mathbf{z}_1 = \mathbf{z}_2$ erfüllt sein müssen. Das Gelenk entfernt somit drei Rotationsfreiheitsgrade des Systems. Zu Beginn der Simulation werden die Quaternionen $\mathbf{q}_1(0)$ und $\mathbf{q}_2(0)$, die die Rotation der beiden Körper beschreiben, gespeichert. Die Änderung der relativen Drehung der beiden Körper zum Zeitpunkt t kann folgendermaßen beschrieben werden:

$$\triangle\mathbf{q}(t) = \left(\mathbf{q}_2(0)^{-1} \cdot \mathbf{q}_2(t)\right)^{-1} \cdot \left(\mathbf{q}_1(0)^{-1} \cdot \mathbf{q}_1(t)\right).$$

Die Quaternion $\triangle\mathbf{q}(t)$ wird in eine entsprechende Rotationsachse $\mathbf{x}(t)$ mit zugehörigem Winkel $\alpha(t)$ konvertiert (siehe Anhang A.2). Da die relative Rotation

zwischen den Körpern des Gelenks während der Simulation immer konstant sein muss, ist die Gelenkbedingung folgendermaßen definiert:

$$|\alpha(t) \cdot \mathbf{x}(t)| \leq \varepsilon_d.$$

Wenn auf die beiden Körper des Gelenks keine internen Kräfte bzw. Drehmomente wirken, dann wird die Gelenkbedingung im Allgemeinen nicht erfüllt. Dadurch ergibt sich am Ende eines Simulationsschrittes eine Änderung der relativen Rotation, die durch die Quaternion $\triangle\mathbf{q}(t_0 + h)$ beschrieben wird. Diese wird berechnet, indem die Quaternionen $\mathbf{q}_1(t_0 + h)$ und $\mathbf{q}_2(t_0 + h)$ durch Integrieren der Differentialgleichung 3.7 bestimmt werden. Die Quaternion $\triangle\mathbf{q}(t_0+h)$ wird anschließend in die Rotationsachse $\mathbf{x}(t_0 + h)$ mit dem Winkel $\alpha(t_0 + h)$ konvertiert. Damit die Gelenkbedingung zum Zeitpunkt $t_0 + h$ erfüllt wird, muss ein Drehimpuls berechnet werden, der die Rotation

$$\mathbf{d}(t_0 + h) = \alpha(t_0 + h) \cdot \mathbf{x}(t_0 + h)$$

in der Zeitspanne h eliminiert. Anstatt den gesuchten Drehimpuls mit einer nichtlinearen Gleichung zu berechnen, wird ein Paar von Drehimpulsen $\mathbf{l}$ und $-\mathbf{l}$ bestimmt, die die Drehung $\mathbf{d}(t_0 + h)$ innerhalb des Simulationsschrittes reduzieren. Die relative Winkelgeschwindigkeit der beiden Körper muss sich durch die Wirkung der beiden entgegengesetzten Drehimpulse um $\mathbf{d}(t_0 + h)/h$ ändern. Mit Hilfe der Matrix $\mathbf{L}$, die in Abschnitt 3.2.2 definiert wurde, kann die Gleichung für den Drehimpuls $\mathbf{l}$ aufgestellt werden:

$$(\mathbf{L}_1(t_0) + \mathbf{L}_2(t_0))\,\mathbf{l} = \frac{1}{h}\,\mathbf{d}(t_0 + h). \tag{3.17}$$

Der Trägheitstensor eines Körpers sowie seine Inverse sind konstant zum Zeitpunkt t_0, regulär, symmetrisch und positiv definit. Daraus folgt, dass auch die Matrix $\mathbf{L}(t_0) = \mathbf{L}_1(t_0) + \mathbf{L}_2(t_0)$ diese Eigenschaften hat. Der Drehimpuls $\mathbf{l}$ kann daher durch Invertieren der Matrix $\mathbf{L}(t_0)$ berechnet werden:

$$\mathbf{l} = \frac{1}{h}\,\mathbf{L}(t_0)^{-1}\,\mathbf{d}(t_0 + h). \tag{3.18}$$

Die Impulse $\mathbf{l}$ und $-\mathbf{l}$ werden zum Zeitpunkt t_0 auf die beiden Körper angewendet und bewirken eine sofortige Änderung der Winkelgeschwindigkeiten. Dadurch verkleinert sich die relative Rotation $\mathbf{d}(t_0 + h)$. In einer iterativen Schleife werden mit Gleichung 3.18 weitere Drehimpulse bestimmt und angewendet, bis die Bedingung $|\mathbf{d}(t_0 + h)| \leq \varepsilon_d$ gilt. Anschließend wird ein Simulationsschritt durchgeführt, bei dem die beiden Körper als freie Körper behandelt werden (siehe Abschnitt 3.1). Die geänderten Winkelgeschwindigkeiten bewirken, dass die Gelenkbedingung nach dem Simulationsschritt erfüllt ist.

Die Differenz der Winkelgeschwindigkeiten der beiden Körper muss während der Simulation Null sein bzw. innerhalb eines Toleranzbereichs liegen. Die Geschwindigkeitsbedingung des Translationsgelenks ist definiert durch

$$|\boldsymbol{\omega}_1(t) - \boldsymbol{\omega}_2(t)| \leq \varepsilon_v,$$

wobei ε_v die maximal zulässige Geschwindigkeitsdifferenz ist. Die Geschwindigkeitsbedingung ist nach einem Simulationsschritt im Allgemeinen nicht erfüllt. Aus diesem Grund wird ein Drehimpuls $\mathbf{l}_v$ bestimmt, der dies korrigiert. Dieser Drehimpuls muss bewirken, dass die Differenz der Winkelgeschwindigkeiten $\triangle\boldsymbol{\omega} = \boldsymbol{\omega}_2 - \boldsymbol{\omega}_1$ zum Zeitpunkt $t_0 + h$ Null wird und muss daher die Gleichung

$$\mathbf{L}(t_0 + h)\,\mathbf{l}_v = \triangle\boldsymbol{\omega}(t_0 + h) \tag{3.19}$$

erfüllen. Durch Invertieren der Matrix $\mathbf{L}(t_0+h)$ lässt sich der gesuchte Drehimpuls exakt berechnen. Wenn der Drehimpuls in entgegengesetzte Richtungen auf die beiden Körper angewendet wird, ändern sich die Winkelgeschwindigkeiten und die Geschwindigkeitsbedingung wird erfüllt. Da die Matrix $\mathbf{L}(t)$ konstant ist für einen Zeitpunkt, muss ihre Inverse nur einmal in jedem Simulationsschritt berechnet werden.

Richtungsgelenk Zwei Körper, die mit einem Richtungsgelenk (siehe Abbildung 3.12) verbunden sind, dürfen sich relativ zueinander beliebig verschieben, aber nur um eine gemeinsame Achse drehen. Daher muss während der Simulation

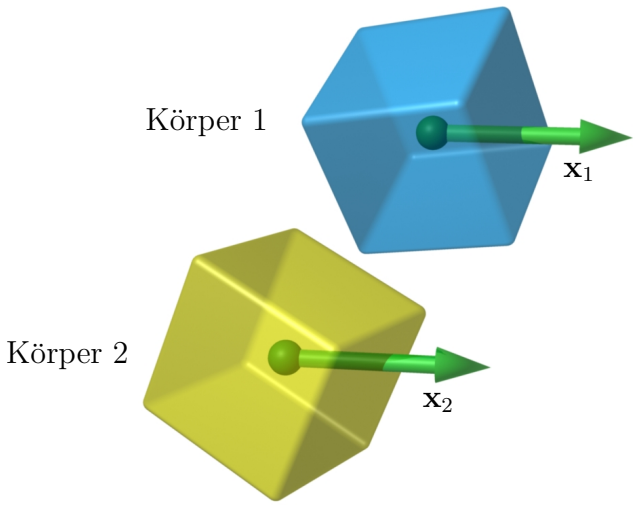


Abbildung 3.12: Richtungsgelenk

stets $\mathbf{x}_1 = \mathbf{x}_2$ gelten. Dieses Gelenk entfernt zwei Rotationsfreiheitsgrade des Systems. Ein Richtungsgelenk wird am Anfang der Simulation durch eine gemeinsame Rotationsachse der beiden verbundenen Körper definiert. Für die Simulation wird

in Richtung der Rotationsachse der normierte Vektor $\mathbf{x}_1$ zum ersten Körper und der normierte Vektor $\mathbf{x}_2$ zum zweiten Körper hinzugefügt. Diese Vektoren werden als *Gelenkvektoren* bezeichnet. Gelenkvektoren sind fest mit dem jeweiligen Körper verbunden. Ein solcher Vektor wird im lokalen Koordinatensystem seines zugehörigen Körpers gespeichert und in jedem Simulationsschritt einmal in Weltkoordinaten transformiert. Auf diese Weise lässt sich ein numerischer Drift des Vektors vermeiden.

Die Gelenkbedingung des Richtungsgelenks fordert, dass die Richtungen der beiden Gelenkvektoren immer gleich sein müssen. Dies ist der Fall, wenn das Kreuzprodukt der Vektoren die Länge Null hat und die Vektoren nicht in entgegengesetzte Richtungen zeigen. Die Gelenkbedingung muss zweidimensional sein, da das Gelenk zwei Freiheitsgrade des Systems entfernt. Wenn die Gelenkbedingung erfüllt ist und damit $\mathbf{x}_1 = \mathbf{x}_2$ gilt, kann eine Ebene senkrecht zu den Gelenkvektoren der beiden Körper bestimmt werden. Die zweidimensionale Gelenkbedingung ergibt sich, wenn das Kreuzprodukt auf diese Ebene projiziert wird. Um die Projektionsmatrix zu bestimmen, werden zwei linear unabhängige Vektoren $\mathbf{y}$ und $\mathbf{z}$ benötigt, die diese Ebene aufspannen. Es sei $\mathbf{w} \in \mathbb{R}^3$ ein beliebiger normierter Vektor, für den $\mathbf{w} \cdot \mathbf{x}_1 \neq 1$ gilt. Die Vektoren

$$\begin{aligned} \mathbf{y} &= \mathbf{x}_1 \times \mathbf{w} \\ \mathbf{z} &= \mathbf{x}_1 \times \mathbf{y} \end{aligned}$$

spannen die gesuchte Ebene auf. Mit der Projektionsmatrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{y}^T \\ \mathbf{z}^T \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

kann das Kreuzprodukt in die Ebene projiziert werden. Damit ergibt sich für die Gelenkbedingung des Richtungsgelenks:

$$|\mathbf{P}\ (\mathbf{x}_1(t) \times \mathbf{x}_2(t))| \leq \varepsilon_d.$$

Wenn die Gelenkbedingung am Anfang des Simulationsschrittes erfüllt ist, dann gilt $\mathbf{x}_1(t_0) = \mathbf{x}_2(t_0)$. Zu diesem Zeitpunkt kann die Projektionsmatrix $\mathbf{P}$ bestimmt werden. Der Fehler, der sich ergibt, wenn ein Simulationsschritt ohne Berücksichtigung der Gelenkbedingung durchgeführt wird, kann durch den Term

$$\mathbf{d}(t_0 + h) = \arccos\left(\mathbf{x}_1(t_0 + h) \cdot \mathbf{x}_2(t_0 + h)\right)\ \left(\mathbf{x}_1(t_0 + h) \times \mathbf{x}_2(t_0 + h)\right)$$

beschrieben werden[2]. Dieser Fehlerterm ist das Produkt des Winkels zwischen den beiden Gelenkvektoren und der zugehörigen, normierten Drehachse. Wenn die Gelenkvektoren genau entgegengesetzte Richtungen haben, dann wird das Kreuzprodukt Null und der Winkel hat einen Wert von 180 Grad. In diesem Fall kann eine

[2]Die Vektoren $\mathbf{x}_1$ und $\mathbf{x}_2$ sind zum Zeitpunkt $t_0 + h$ unter Umständen durch numerische Ungenauigkeiten nicht mehr normiert. Daher müssen sie vor der Berechnung des Fehlers $\mathbf{d}(t_0+h)$ normiert werden.

beliebige Drehachse gewählt werden. Der Fehler kann mit einem Drehimpuls $\mathbf{l}$, der zum Zeitpunkt t_0 in entgegengesetzte Richtungen auf die beiden Körper wirkt, reduziert werden. Dieser Drehimpuls wird bestimmt, indem zunächst die Gleichung 3.17 mit der Projektionsmatrix in die Ebene projiziert wird:

$$\mathbf{P}\left(\mathbf{L}_1(t_0) + \mathbf{L}_2(t_0)\right)\mathbf{P}^T\,\mathbf{l}' = \frac{1}{h}\,\mathbf{P}\,\mathbf{d}(t_0 + h). \tag{3.20}$$

Die Matrix $\mathbf{P}\,\mathbf{L}(t_0)\,\mathbf{P}^T$ ist zum Zeitpunkt t_0 konstant, symmetrisch, positiv definit und damit auch regulär. Der Beweis dieser Eigenschaften funktioniert analog zu dem Beweis für die Matrix $\mathbf{P}\,\mathbf{K}(t_0)\,\mathbf{P}^T$. Die Gleichung 3.20 kann daher durch Invertieren der Matrix $\mathbf{P}\,\mathbf{L}(t_0)\,\mathbf{P}^T$ für den Drehimpuls $\mathbf{l}'$ in der Ebene gelöst werden. Anschließend wird der Drehimpuls in das Weltkoordinatensystem transformiert:

$$\mathbf{l} = \mathbf{P}^T\,\mathbf{l}'.$$

Wenn zum Zeitpunkt t_0 der Drehimpuls $\mathbf{l}$ auf den ersten Körper und der Drehimpuls $-\mathbf{l}$ auf den zweiten Körper wirkt, dann ändern sich die Winkelgeschwindigkeiten der beiden Körper. Dies bewirkt, dass der Fehler $\mathbf{d}(t_0 + h)$ kleiner wird. Die Gelenkbedingung am Ende des Simulationsschrittes wird durch iteratives Berechnen von Drehimpulsen mit Gleichung 3.20 erfüllt.

Wird die Differenz der Winkelgeschwindigkeiten der beiden Körper in die Ebene senkrecht zu der gemeinsamen Rotationsachse des Gelenks projiziert, so ergibt sich die Geschwindigkeitsbedingung:

$$|\mathbf{P}(\boldsymbol{\omega}_1(t) - \boldsymbol{\omega}_2(t))| \leq \varepsilon_v.$$

Bevor ein Drehimpuls zur Korrektur der Geschwindigkeitsdifferenz $\mathbf{P}\,\triangle\boldsymbol{\omega}(t_0 + h)$ berechnet werden kann, muss zunächst die Projektionsmatrix $\mathbf{P}$ für den Zeitpunkt $t_0 + h$ neu bestimmt werden. Anschließend kann die Gleichung 3.19 auf die Ebene senkrecht zu der Rotationsachse projiziert werden:

$$\mathbf{P}\,\mathbf{L}(t_0 + h)\,\mathbf{P}^T\,\mathbf{l}'_v = \mathbf{P}\,\triangle\boldsymbol{\omega}(t_0 + h). \tag{3.21}$$

Die resultierende Gleichung wird nach dem Drehimpuls $\mathbf{l}'_v$ im Koordinatensystem der Ebene aufgelöst. Dieser Drehimpuls wird dann in das Weltkoordinatensystem transformiert und auf die beiden Körper des Gelenks in entgegengesetzte Richtungen angewendet. Durch die Wirkung des Drehimpulses wird die Geschwindigkeitsbedingung erfüllt.

Doppelrotationsgelenk Zwei Körper, die durch ein Doppelrotationsgelenk (siehe Abbildung 3.13) verbunden sind, dürfen sich relativ zueinander nur um zwei linear unabhängige Achsen drehen. Dieses Gelenk entfernt einen Rotationsfreiheitsgrad des Systems und wird am Anfang der Simulation durch die beiden normierten,

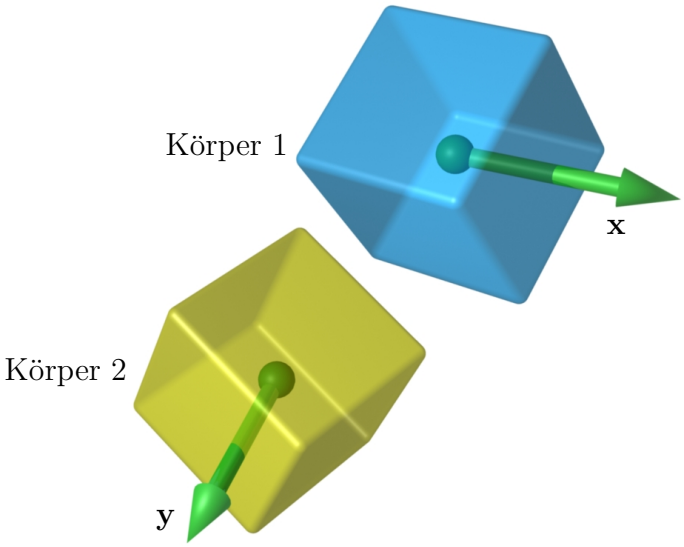


Abbildung 3.13: Doppelrotationsgelenk

linear unabhängigen Achsen $\mathbf{x}$ und $\mathbf{y}$ definiert. Eine der beiden Achsen wird als Gelenkvektor zum ersten Körper hinzugefügt und die andere Achse zum zweiten Körper. Die Gelenkbedingung des Doppelrotationsgelenks ist, dass der Winkel

$$\alpha(t) = \arccos(\mathbf{x}(t) \cdot \mathbf{y}(t))$$

zwischen den beiden Gelenkvektoren konstant bleiben muss. Das bedeutet, dass die Differenz zwischen dem aktuellen Winkel $\alpha(t)$ und dem Winkel zu Beginn der Simulation $\alpha(0)$ Null bzw. kleiner als ein Toleranzwert sein muss:

$$|\alpha(t) - \alpha(0)| \leq \varepsilon_d.$$

Die Richtungen der Gelenkvektoren $\mathbf{x}(t_0+h)$ und $\mathbf{y}(t_0+h)$ am Ende eines Simulationsschrittes können durch Lösen der Differentialgleichung 3.5 bestimmt werden. Wenn während der Simulation keine internen Kräfte bzw. Impulse auf die Körper des Gelenks wirken, dann entsteht zum Zeitpunkt $t_0 + h$ ein Fehler, der durch den Term $\alpha(t_0+h)-\alpha(0)$ beschrieben wird. Ein Drehimpuls muss zum Zeitpunkt t_0 in entgegengesetzte Richtungen auf die beiden Körper angewendet werden, um diesen Fehler zu korrigieren. Die Richtung $\mathbf{z}$ des Drehimpulses muss senkrecht zu den beiden Vektoren $\mathbf{x}$ und $\mathbf{y}$ zum Zeitpunkt t_0 sein:

$$\mathbf{z} = \mathbf{x} \times \mathbf{y}.$$

Mit der Richtung des Drehimpulses kann die Projektionsmatrix

$$\mathbf{P} = \left(\mathbf{z}^T\right) \in \mathbb{R}^{1\times 3}$$

definiert werden. Das Produkt des Fehlerterms mit dem Vektor $\mathbf{z}$ ergibt den dreidimensionalen Fehler

$$\mathbf{d}(t_0 + h) = (\alpha(t_0 + h) - \alpha(0))\,\mathbf{z}.$$

Mit diesem kann die Gleichung 3.20 für den Drehimpuls aufgestellt werden, der den Fehler reduziert. Durch Lösen dieser Gleichung ergibt sich die Stärke des Drehimpulses l'. Der Drehimpuls $\mathbf{l} = l' \cdot \mathbf{z}$ wird positiv auf den ersten Körper und negativ auf den zweiten Körper angewendet. Durch iterative Bestimmung von Drehimpulsen mit Gleichung 3.20 wird der Fehler $|\mathbf{d}(t_0 + h)|$ schrittweise verkleinert, bis die Gelenkbedingung nach dem Simulationsschritt erfüllt ist.

Die Geschwindigkeitsbedingung des Doppelrotationsgelenks ergibt sich, wenn die relative Winkelgeschwindigkeit der beiden verbundenen Körper mit der aktuellen Projektionsmatrix auf die Achse senkrecht zu $\mathbf{x}$ und $\mathbf{y}$ projiziert wird:

$$|\mathbf{P}(\boldsymbol{\omega}_1(t) - \boldsymbol{\omega}_2(t))| \leq \varepsilon_v.$$

Durch Lösen der Gleichung 3.21 ergibt sich die Stärke des Drehimpulses l'_v, der die Differenz der beiden Winkelgeschwindigkeiten eliminiert. Wendet man den Drehimpuls $\mathbf{l}_v = l'_v \cdot \mathbf{z}$ auf beide Körper in entgegengesetzte Richtungen an, dann wird die Geschwindigkeitsbedingung des Gelenks erfüllt.

Zum Schluss lässt sich noch anmerken, dass die Drehimpulse für alle Gelenke mit Rotationsbedingungen mit Hilfe der Gleichungen 3.20 und 3.21 berechnet werden können, wenn beim Translationsgelenk die Einheitsmatrix als Projektionsmatrix verwendet wird.

3.2.4 Kombinierte Gelenke

Im letzten Abschnitt wurden sechs Gelenke mit Translations- und Rotationsbedingungen vorgestellt. Durch Kombination mehrerer dieser Grundgelenke können neue Gelenktypen realisiert werden. Ein solches kombiniertes Gelenk kann verschiedene Translations- und Rotationsfreiheitsgrade entfernen. Bisher wurden ausschließlich Mehrkörpersysteme mit höchstens einem Gelenk diskutiert. Durch ein kombiniertes Gelenk entsteht ein System mit mehreren Grundgelenken. Die Simulation solcher Systeme wird ausführlich in Abschnitt 3.3 beschrieben. In diesem Abschnitt werden die wichtigsten kombinierten Gelenke präsentiert.

Drehgelenk Ein Drehgelenk (siehe Abbildung 3.14) erlaubt den beiden verbundenen Körpern eine relative Drehbewegung um eine gemeinsame Achse $\mathbf{x}$. Außerdem haben die beiden Körper einen gemeinsamen Punkt $\mathbf{a}$. Dies bedeutet, dass die beiden Körper um die Gerade $\mathbf{a} + \lambda\,\mathbf{x}$ rotieren können. Dieser Gelenktyp kann durch die Kombination eines Kugelgelenks mit einem Richtungsgelenk simuliert

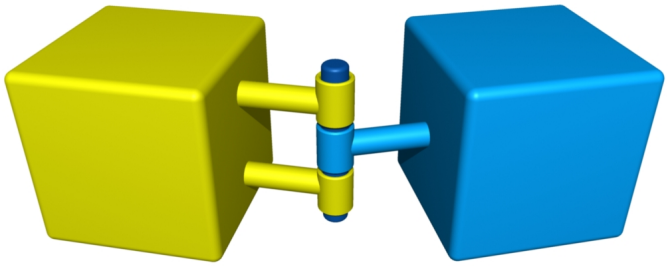


Abbildung 3.14: Drehgelenk

werden. Das Kugelgelenk verhindert, dass die Körper in dem gemeinsamen Punkt auseinander driften und das Richtungsgelenk sorgt dafür, dass die Körper nur um die gemeinsame Achse rotieren. Damit entfernt das Drehgelenk drei Translations- und zwei Rotationsfreiheitsgrade des Systems.

Eine weitere Möglichkeit für die Simulation eines Drehgelenks besteht darin, ein Kugelgelenk mit einem Geradengelenk zu kombinieren. Die beiden Gelenke definieren am Anfang der Simulation jeweils einen gemeinsamen Punkt der beiden Körper. Der Punkt des Kugelgelenks wird an die Position $\mathbf{a}$ gesetzt und der Punkt des Geradengelenks auf eine beliebige andere Position auf der Geraden $\mathbf{a} + \lambda\,\mathbf{x}$. Außerdem muss der Vektor des Geradengelenks die Richtung $\mathbf{x}$ haben. Während der Simulation haben die Punkte der beiden Gelenke immer den gleichen Abstand, da das Kugelgelenk verhindert, dass sich die verbundenen Körper auf der Geraden des Geradengelenks bewegen. Daher kann in diesem Fall auf die Lotpunktberechnung beim Geradengelenk verzichtet werden. Da die beiden Punkte unterschiedliche Positionen auf der Geraden haben, können sich die verbundenen Körper relativ zueinander nur um die Achse $\mathbf{x}$ drehen.

Durch die Kombination von zwei Grundgelenken konnte ein weiterer Gelenktyp realisiert werden. Allerdings gibt es verschiedene Kombinationsmöglichkeiten, die zu dem gleichen Ergebnis führen.

Schienendrehgelenk Zwei Körper, die mit einem Schienendrehgelenk verbunden sind (siehe Abbildung 3.15), können sich auf einer gemeinsamen Gerade bewegen und nur um diese Gerade rotieren. Dieser Gelenktyp kann durch die Kombination eines Geradengelenks mit einem Richtungsgelenk simuliert werden. Das Geradengelenk definiert die Schiene, auf der sich die beiden Körper bewegen dürfen. Das Richtungsgelenk ist an dieser Schiene ausgerichtet. Dadurch können die Körper des Gelenks relativ zueinander nur um die Schiene rotieren. Das Schienendrehgelenk entfernt zwei Translations- und zwei Rotationsfreiheitsgrade des Systems.

Schienengelenk Abbildung 3.16 zeigt ein Schienengelenk. Dieser Gelenktyp er-

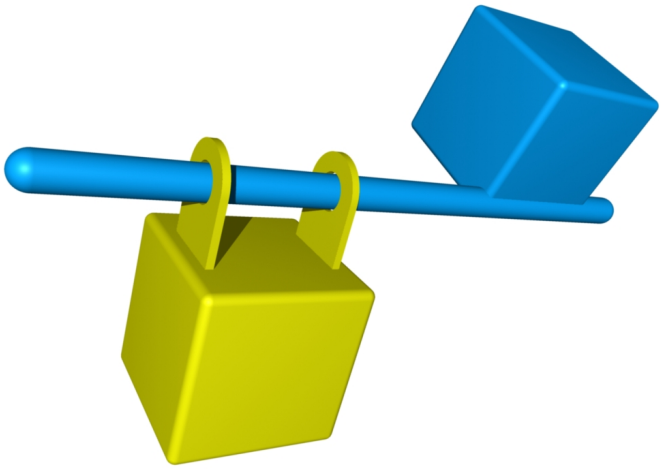


Abbildung 3.15: Schienendrehgelenk

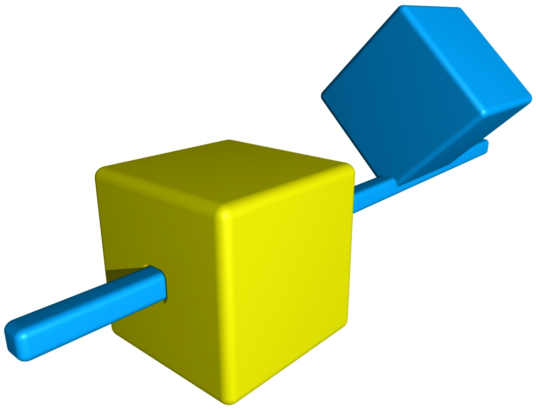


Abbildung 3.16: Schienengelenk

laubt den beiden verbundenen Körpern nur, sich relativ zueinander auf einer Geraden zu bewegen. Die relative Drehung der Körper muss während der Simulation konstant bleiben. Ein Schienengelenk kann simuliert werden, indem ein Geradengelenk mit einem Translationsgelenk kombiniert wird. Das Geradengelenk ermöglicht den beiden Körpern, sich frei auf einer Schiene zu bewegen, während das Translationsgelenk eine relative Rotation der Körper verhindert. Dadurch werden zwei Translations- und drei Rotationsfreiheitsgrade aus dem System entfernt.

Kardangelenk Zwei Körper, die mit einem Kardangelenk verbunden sind (siehe Abbildung 3.17), haben einen gemeinsamen Punkt und können sich um zwei linear unabhängige Achsen drehen. In dem gemeinsamen Punkt dürfen sich die Körper relativ zueinander nicht verschieben. Dies kann mit Hilfe eines Kugelgelenks simuliert werden. Wenn das Kugelgelenk mit einem Doppelrotationsgelenk, das den

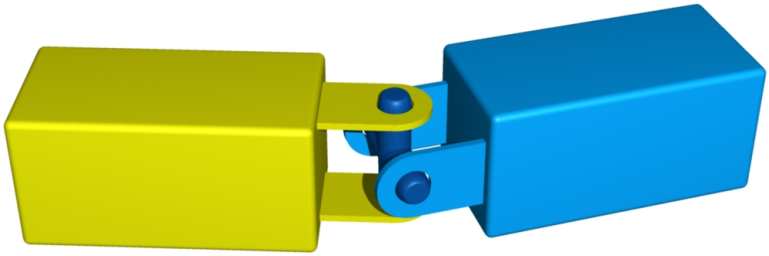


Abbildung 3.17: Kardangelenk

beiden Körpern eine Rotation um zwei verschiedene Achsen erlaubt, kombiniert wird, dann ergibt sich ein Kardangelenk. Dieser Gelenktyp reduziert die Freiheitsgrade des Systems um drei Translations- und einen Rotationsfreiheitsgrad.

Fixierung Mit einer Fixierung (siehe Abbildung 3.18) lässt sich ein Körper an einem anderen befestigen. Die Fixierung verhindert, dass sich die beiden Körper

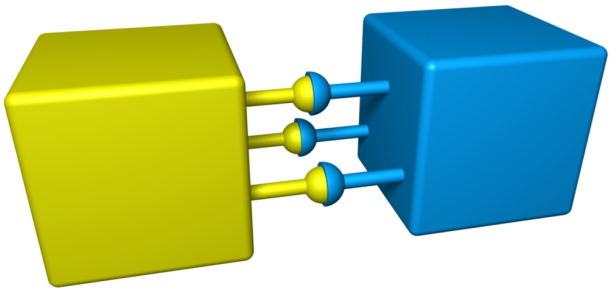


Abbildung 3.18: Fixierung

relativ zueinander drehen oder verschieben. Ein solches Gelenk kann z. B. durch eine Kombination von einem Kugelgelenk und einem Translationsgelenk simuliert werden. Das Kugelgelenk sorgt dafür, dass die relative Translation der beiden Körper konstant bleibt, und das Translationsgelenk verhindert eine relative Rotation. Damit entfernt eine Fixierung drei Translations- und drei Rotationsfreiheitsgrade des Systems.

3.2.5 Servomotor

Ein Motor ist ein Gelenk, das zwei Starrkörper verbindet, und außerdem eine Kraft auf beide Körper ausübt. Diese Kraft wird am Anfang eines Simulationsschrittes zu den externen Kräften der jeweiligen Körper addiert und bleibt während des Simulationsschrittes konstant.

In dieser Arbeit wird zwischen zwei Arten von Motoren unterschieden: Drehmotoren und Linearmotoren. Die erste Art von Motoren verbindet zwei Starrkörper mit

einem Drehgelenk (siehe Abbildung 3.19). Wenn ein Drehmoment $\boldsymbol{\tau}$ in Richtung

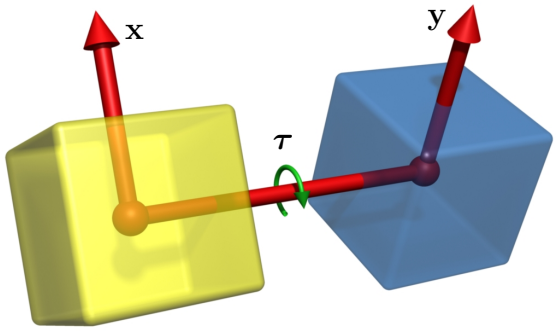


Abbildung 3.19: Drehmotor

der Drehachse positiv auf den ersten Körper und negativ auf den zweiten Körper angewendet wird, beschleunigen die beiden Körper ihre Rotationsbewegung. Da die Drehmomente in entgegengesetzte Richtungen wirken, ist die Drehimpulserhaltung des Systems gewährleistet. Am Anfang der Simulation wird ein Vektor senkrecht zur Drehachse bestimmt. Dieser wird dann als Gelenkvektor zum ersten und zweiten Körper hinzugefügt. Die resultierenden Gelenkvektoren **x** und **y** können während der Simulation dazu verwendet werden, den aktuellen Winkel zwischen den beiden Körpern zu bestimmen.

Ein Linearmotor verbindet zwei Körper durch ein Schienengelenk (siehe Abbildung 3.20). Die beiden Starrkörper werden angetrieben, indem eine Kraft **F** in

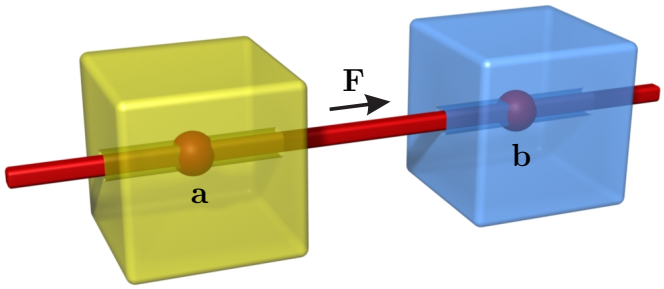


Abbildung 3.20: Linearmotor

Richtung der Schiene positiv auf den ersten Körper und negativ auf den zweiten Körper wirkt. Zu Beginn der Simulation wird ein beliebiger Punkt auf der Schiene zu beiden Körpern als Gelenkpunkt hinzugefügt. Mit Hilfe dieser Punkte **a** und **b** kann während der Simulation die relative Position der beiden Körper bestimmt werden.

Ein Servomotor benötigt eine Regelung, damit er einen Winkel, eine Position oder eine Geschwindigkeit ansteuern kann. Im Gegensatz zu einer Steuerung, existiert bei einer Regelung ein geschlossener Regelkreis (siehe Abbildung 3.21). In diesem

Regelkreis wird ständig der aktuelle Istwert gemessen und mit dem gewünschten Sollwert verglichen. Die Differenz zwischen dem Sollwert und dem Istwert ergibt die Regelabweichung e. Der Regler berechnet abhängig von der Regelabweichung eine Stellgröße sg für das zu regelnde System, die dafür sorgt, dass der gewünschte Sollwert erreicht wird. Die Regelstrecke steht für das zu regelnde System. Die Stellgröße ändert den Istwert nicht sofort, sondern wirkt sich mit einer Verzögerung auf den Istwert aus. Außerdem können Störungen auftreten, die beeinflussen, wie schnell der Sollwert erreicht wird. Die Aufgabe der Regelung ist es, die Werte des Systems trotz Verzögerung und Störungen an die vorgegebenen Sollwerte anzupassen und diese Werte durch ständiges Nachregeln zu halten.

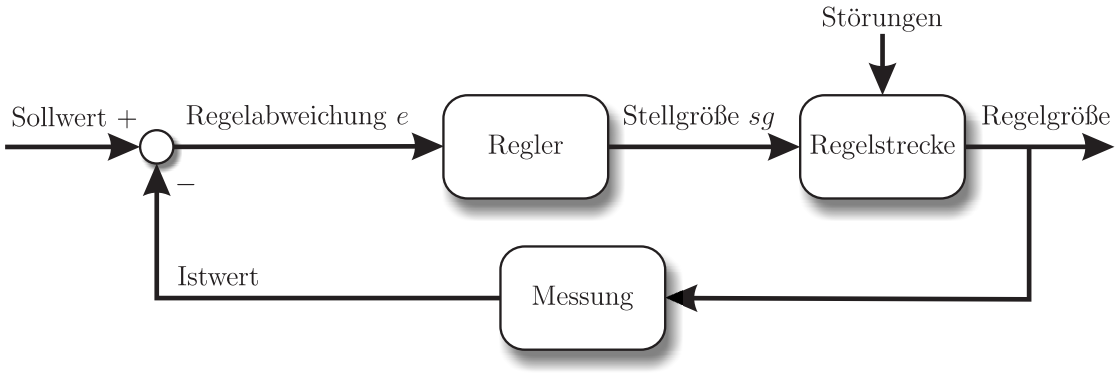


Abbildung 3.21: Regelkreis

In dieser Arbeit wurde ein PID-Regler für die Regelung der Motoren verwendet. PID-Regler steht für Proportional-Integral-Differential-Regler. Das Verhalten des PID-Reglers kann mit Hilfe von drei Parametern festgelegt werden. Wenn $e(t)$ die Differenz zwischen Soll- und Istwert ist, dann berechnet sich die Stellgröße sg folgendermaßen:

$$sg = K_r \cdot \left(\underbrace{e(t)}_{\text{Proportionalteil}} + \underbrace{\frac{1}{T_N} \int_0^t e(x)dx}_{\text{Integralteil}} + \underbrace{T_V \frac{d\,e(t)}{dt}}_{\text{Differentialteil}} \right) \tag{3.22}$$

Die Stellgröße ist abhängig von dem Parameter K_R für den Proportionalteil, der Nachstellzeit T_N für den Integralteil und der Vorhaltezeit T_V für den Differentialteil. In der Praxis hat es sich als vorteilhaft erwiesen, mit dem Parameter K_r die gesamte Stellgröße anzupassen, anstatt nur den Proportionalteil. Deswegen wurde dieser Parameter ausgeklammert. Die drei Teile des PID-Reglers haben die folgende Bedeutung:

- Der **Proportionalteil** sorgt dafür, dass die Kraft, die angewendet wird, um den Sollwert zu erreichen, proportional zur Regelabweichung ist. Je kleiner die Differenz zwischen Soll- und Istwert ist, umso kleiner wird die Kraft.

Allerdings kehrt sich die Richtung der Kraft erst um, wenn der Sollwert überschritten wurde. Dadurch kommt es zu einem Überschwingen. Dieses Problem tritt vor allem bei Systemen auf, die in ihrer Regelstrecke keine oder nur wenig Dämpfung haben. Ein einfacher P-Regler reicht deshalb in den meisten Fällen nicht aus.

- Man benötigt eine ausreichende Dämpfung, um das Überschwingen zu verhindern. Bei Systemen, die von sich aus nur eine geringe Dämpfung haben, muss man eine künstliche Dämpfung verwenden. Das wird durch den **Differentialteil** des PID-Reglers erreicht. Verringert sich die Differenz zwischen Soll- und Istwert, dann wird der Differentialteil negativ und eine Kraft wirkt dem Proportionalteil entgegen, bevor der Sollwert erreicht wird. Die Stärke der Dämpfung ist abhängig von der Geschwindigkeit, mit der sich der Istwert auf den Sollwert zu bewegt, sowie vom Parameter T_V.

- Der **Integralteil** summiert die Regelabweichungen $e(t)$ über die Zeit auf. Die Dauer der Abweichung geht daher in die Berechnung der Stellgröße mit ein. Durch den Integralteil kann der Sollwert exakt erreicht werden. In der Praxis wird nicht der vollständige Zeitraum von 0 bis t integriert, wie in Gleichung 3.22 angegeben, denn nur die Regelabweichungen bis zu einem bestimmten Zeitpunkt in der Vergangenheit sind interessant.

Für die Implementierung des PID-Reglers wurde die Formel für die Stellgröße durch eine Diskretisierung vereinfacht:

$$sg = K_r \cdot \left(e(k) + \frac{h}{T_N} \sum_{i=0}^{k-1} e(i) + \frac{T_V}{h} \left(e(k) - e(k-1) \right) \right) \tag{3.23}$$

Die Größe h gibt dabei die Zeitschrittweite an. Das Integral wurde durch eine Summe der letzten k Regelabweichungen angenähert und der Differentialquotient durch eine Differenz der letzten beiden Werte.

Die optimale Einstellung der drei Parameter des PID-Reglers für eine bestimmte Regelstrecke ist nicht trivial. Es gibt verschiedene Anforderungen an den Regler, die bei der Wahl der Parameter berücksichtigt werden müssen:

- Der Sollwert soll möglichst schnell erreicht werden.

- Dabei soll kein Überschwingen auftreten.

- Wenn der Sollwert erreicht ist, soll dieser so genau wie möglich gehalten werden.

- Die Regelung soll möglichst unempfindlich gegenüber Störungen sein.

Es existieren verschiedene Verfahren zur Bestimmung der Parameter für einen PID-Regler. Die bekanntesten Verfahren sind die von Ziegler und Nichols [Lun06] und von Chien, Hrones und Reswick [RZ02].

3.2.6 Gelenke mit Geschwindigkeitsbedingungen

Die sechs Grundgelenke sowie alle Gelenke, die aus Kombinationen von Grundgelenken erzeugt wurden, stellen für die verbundenen Körper jeweils eine Zwangsbedingung für die relative Position bzw. Rotation und für die Geschwindigkeiten der Körper auf. Gelenke, die nur die Geschwindigkeiten der Körper beeinflussen, haben nur eine Geschwindigkeitsbedingung. Diese kann, wie bei den Grundgelenken, durch die Differenz der Geschwindigkeiten der Gelenkpunkte oder die Differenz der Winkelgeschwindigkeiten definiert werden. Mit Hilfe von Projektionsmatrizen können Geschwindigkeitsbedingungen für jeden translatorischen und jeden rotatorischen Freiheitsgrad aufgestellt werden. Die resultierenden Zwangsbedingungen werden während der Geschwindigkeitskorrektur erfüllt, indem Korrekturimpulse mit den Gleichungen 3.16 und 3.21 berechnet und angewendet werden. Durch Kombination von translatorischen und rotatorischen Bedingungen für die Geschwindigkeiten der Körper kann jede mögliche Geschwindigkeitsbedingung simuliert werden.

Gelenke mit Zwangsbedingungen für die Geschwindigkeiten sind sehr nützlich, wenn die Bewegung eines Körpers mit Hilfe eines anderen Körpers gesteuert werden soll. Wenn z. B. der Anwender Einfluss auf die laufende Simulation nehmen will, indem er einen Körper bewegt, ist die Verwendung eines solchen Gelenks sinnvoll. Um einen Körper interaktiv mit Hilfe der Maus zu bewegen, wird an der aktuellen Mausposition ein Starrkörper k_h hilfsweise eingefügt, der sich mit der Maus bewegt und auf den keine externen Kräfte wirken. Dieser Körper wird mit dem Starrkörper k_m, der manipuliert werden soll, durch ein Gelenk mit der Geschwindigkeitsbedingung:

$$|\mathbf{u}_a(t) - \mathbf{u}_b(t)| \leq \varepsilon_v$$

verbunden. Dabei ist $\mathbf{a}$ der Punkt, an dem der Körper k_m gezogen wird, und $\mathbf{b}$ die Position des Mauszeigers in dreidimensionalen Koordinaten. Die Geschwindigkeit des Mauszeigers kann durch $(\mathbf{b}(t) - \mathbf{b}(t-h))/h$ approximiert werden. Durch die Geschwindigkeitsbedingung folgt der Körper k_m dem Mauszeiger, solange er nicht durch ein anderes Gelenk oder eine Kollision daran gehindert wird. Dies kann mit einem Gelenk mit Bedingungen für die Position oder Rotation nicht umgesetzt werden, da die Positionsbedingung zwischen dem Hilfskörper und dem Körper, der gezogen wird, nicht in jedem Fall erfüllt werden kann.

3.2.7 Feder

Eine Feder (siehe Abbildung 3.22) wird in dieser Arbeit als Gelenk behandelt, da sie zwei Körper durch eine Kraft verbindet, die zwischen den Körpern wirkt. Durch eine Feder wird keine Bedingung aufgestellt, die während der Simulation erfüllt sein muss. Eine Feder ist jeweils an einem Punkt der beiden Körper befestigt.

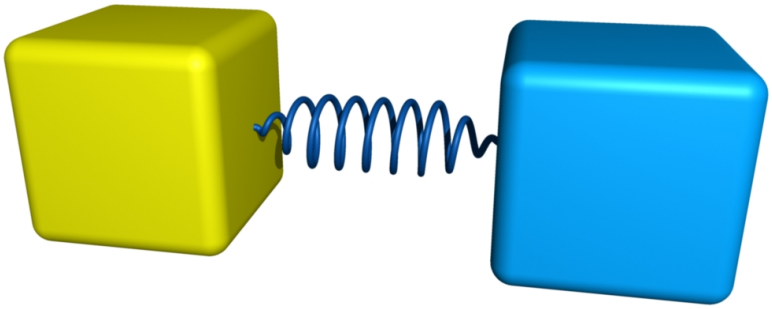


Abbildung 3.22: Feder

Seien $\mathbf{a}$ und $\mathbf{b}$ diese beiden Gelenkpunkte. Die Stärke der Federkraft wird durch den Abstand dieser beiden Gelenkpunkte $\mathbf{d} = \mathbf{b} - \mathbf{a}$, die Länge l, die die Feder in Ruhelage hat, und die Federkonstante D bestimmt:

$$\mathbf{F}_{Feder} = D\left(|\mathbf{d}| - l\right)\hat{\mathbf{d}},$$

wobei $\hat{\mathbf{d}}$ der Einheitsvektor in Richtung des Abstandsvektors $\mathbf{d}$ ist. Die Reibung der Feder kann simuliert werden, indem eine weitere Kraft berechnet wird, die proportional zur Relativgeschwindigkeit der beiden Gelenkpunkte, wirkt:

$$\mathbf{F}_{Reibung} = -\mu_F(\mathbf{u}_b - \mathbf{u}_a)$$

Die resultierende Kraft $\mathbf{F} = \mathbf{F}_{Feder} + \mathbf{F}_{Reibung}$ wird zu den externen Kräften der Körper hinzu addiert[3]. Außerdem ergibt sich ein externes Drehmoment für die beiden Körper, da die Kraft in den Gelenkpunkten wirkt, die im Allgemeinen nicht in den Schwerpunkten der Körper liegen. Das Drehmoment kann folgendermaßen berechnet werden:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F},$$

wobei $\mathbf{r}$ der Vektor vom Schwerpunkt des jeweiligen Körpers zum zugehörigen Gelenkpunkt ist.

[3] In Kapitel 3 wurde die Annahme gemacht, dass alle externen Kräfte während eines Simulationsschrittes konstant sind. Federkräfte bilden eine Ausnahme, da sie kontinuierlich sind. Aus diesem Grund werden die Kräfte entweder nur an diskreten Zeitpunkten berechnet und dazwischen als konstant angenommen oder sie gehen als kontinuierliche Kräfte in die Gleichungen 3.2 und 3.3 ein. Im zweiten Fall müssen die Gleichungen dann durch numerische Integration gelöst werden.

3.3 Systeme von Gelenken

Im letzten Abschnitt wurde die Simulation eines Gelenks beschrieben, das zwei Körper verbindet. Dieser Abschnitt beschäftigt sich mit der Simulation von Mehrkörpersystemen mit mehreren Gelenken. Wenn ein Körper durch mehrere Gelenke mit anderen Körpern verbunden ist, dann beeinflussen sich die Impulse, die während der Gelenkkorrektur bzw. Geschwindigkeitskorrektur für die einzelnen Gelenke berechnet werden, gegenseitig. Dies lässt sich mit Hilfe eines Beispiels leicht veranschaulichen. Abbildung 3.23 zeigt den Gelenkzustand eines Doppelpendels während der Gelenkkorrektur. Für das linke Kugelgelenk wurden Impulse berech-

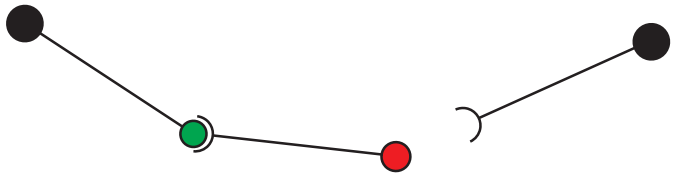


Abbildung 3.23: Doppelpendel während der Gelenkkorrektur

net, so dass die Gelenkpunkte nach einem Simulationsschritt die gleiche Position haben. Werden nun im Rahmen der Gelenkkorrektur Impulse berechnet und angewendet, um das rechte Gelenk zu korrigieren, dann ändern sich durch die Impulse auch die Geschwindigkeiten des mittleren Körpers. Diese Geschwindigkeitsänderung bewirkt, dass die Punkte des linken Gelenks auseinander driften. Im Folgenden werden zwei Verfahren vorgestellt, um dieses Problem zu lösen. Das erste Verfahren arbeitet iterativ und wird im Folgenden *iteratives Verfahren* genannt. Das zweite Verfahren verwendet lineare Gleichungssysteme, um die Struktur des simulierten Mehrkörpersystems zu beschreiben. Dieses Verfahren wird im Folgenden als *LGS-Verfahren* bezeichnet.

3.3.1 Iteratives Verfahren

Beim iterativen Verfahren werden die Gelenke in einem Mehrkörpersystem einzeln betrachtet. Die Abhängigkeiten im Modell, die dadurch entstehen, dass sich die Impulse von verschiedenen Gelenken gegenseitig beeinflussen, werden nicht explizit berücksichtigt.

3.3.1.1 Gelenkkorrektur

Bei der Gelenkkorrektur werden alle Gelenke, die im folgenden Simulationsschritt von t_0 nach t_0+h auseinander brechen würden, in einer iterativen Schleife korrigiert (siehe Algorithmus 3.1).

jointCorrection()

Eingabe: Gelenke $\{0, ..., n\}$

```
repeat
  korrigiert := false
  for i = 0 to n
    Gelenkzustand des i-ten Gelenks nach einem Simulationsschritt bestimmen
    if Gelenkbedingung für i-tes Gelenk nicht erfüllt
      Berechnung eines Korrekturimpulses
      Anwenden des Impulses zum Zeitpunkt t_0
      korrigiert := true
  end for
until korrigiert = false
```

Algorithmus 3.1: Gelenkkorrektur mit dem iterativen Verfahren

Für jedes Gelenk wird zunächst sein Zustand nach dem Simulationsschritt bestimmt. Anhand des Gelenkzustands wird überprüft, ob die jeweilige Gelenkbedingung nach dem Zeitschritt erfüllt wird. Ist dies der Fall, kann mit dem nächsten Gelenk fortgefahren werden. Andernfalls muss ein Impuls bestimmt und zum Zeitpunkt t_0 angewendet werden, der den Fehler nach dem Simulationsschritt reduziert. Wenn alle Gelenke abgearbeitet wurden und dabei keine Korrektur vorgenommen werden musste, dann kann mit dem Simulationsschritt fortgefahren werden. Ansonsten beginnt die Gelenkkorrektur für alle Gelenke von vorne und wird solange fortgesetzt, bis alle Gelenkbedingungen für den Zeitpunkt $t_0 + h$ erfüllt sind. Diese Vorgehensweise konvergiert zu der physikalisch korrekten Lösung, wenn die Zeitschrittweite gegen Null geht. Der Beweis dafür befindet sich in [SBP05b].

Aufgrund der Abhängigkeiten im Modell ist es im Allgemeinen nicht sinnvoll, für jedes Gelenk iterativ einen Impuls zu bestimmen, der dafür sorgt, dass die Gelenkbedingung erfüllt wird (vgl. Abschnitt 3.2.1), und erst dann mit dem nächsten Gelenk fortzufahren. Daher wird bei dem iterativen Verfahren in einer Iteration nur ein Impuls je Gelenk berechnet und angewendet.

3.3.1.2 Geschwindigkeitskorrektur

Nach dem Simulationsschritt müssen die Geschwindigkeitsbedingungen der Gelenke erfüllt werden. Dies wird ebenfalls in einer iterativen Schleife für alle Gelenke gemacht (siehe Algorithmus 3.2). In der Schleife wird für jedes Gelenk bestimmt, ob die zugehörige Geschwindigkeitsbedingung erfüllt ist. Wenn die Bedingung für ein Gelenk nicht erfüllt ist, dann muss ein Impuls bestimmt und angewendet werden, der die Geschwindigkeiten der verbundenen Körper entsprechend korrigiert. Musste in einer Iteration für kein Gelenk eine Korrektur vorgenommen werden,

dann sind alle Bedingungen erfüllt und die Schleife endet. Andernfalls wird mit der Geschwindigkeitskorrektur für alle Gelenke fortgefahren.

---

velocityCorrection()

Eingabe: Gelenke $\{0, ..., n\}$

```
repeat
   korrigiert := false
   for i = 0 to n
      if Geschwindigkeitsbedingung für i-tes Gelenk nicht erfüllt
         Berechnung eines Korrekturimpulses
         Anwenden des Impulses zum Zeitpunkt t_0 + h
         korrigiert := true
   end for
until korrigiert = false
```

Algorithmus 3.2: Geschwindigkeitskorrektur mit dem iterativen Verfahren

3.3.2 LGS-Verfahren

Bei den Grundgelenken wird zwischen Gelenken mit Translationsbedingungen und Gelenken mit Rotationsbedingungen unterschieden. Gelenke mit Translationsbedingungen schränken ausschließlich die translatorischen Freiheitsgrade ein und können mit Hilfe eines Impulses $\mathbf{p}$ korrigiert werden. Dagegen schränken Gelenke mit Rotationsbedingungen nur die rotatorischen Freiheitsgrade der Körper ein und können mit Hilfe eines Drehimpulses $\mathbf{l}$ korrigiert werden. Gelenke, die sowohl translatorische als auch rotatorische Freiheitsgrade entfernen, können als Kombination von Grundgelenken gebildet werden (siehe Abschnitt 3.2.4) und werden deshalb im Folgenden nicht mehr gesondert betrachtet. Dies bedeutet, dass ein Mehrkörpersystem ausschließlich Grundgelenke beinhaltet, die jeweils maximal drei Freiheitsgrade des Systems entfernen (siehe Abschnitt 3.2.3). Gelenke, die weniger als drei Freiheitsgrade entfernen, können durch eine Projektion in den zwei- bzw. eindimensionalen Raum gebildet werden. Im Folgenden definiert die Menge $T = \{1, \ldots, m'\}$ die Indizes aller Grundgelenke mit Translationsbedingungen und die Menge $R = \{m'+1, \ldots, m\}$ die Indizes aller Gelenke mit Rotationsbedingungen in einem Mehrkörpersystem mit m Gelenken.

Im Gegensatz zum iterativen Verfahren werden beim LGS-Verfahren die Abhängigkeiten zwischen den Gelenken berücksichtigt. Diese Abhängigkeiten können mit Hilfe eines linearen Gleichungssystems beschrieben werden:

$$\mathbf{M} \cdot \mathbf{x} = \triangle \mathbf{v}.$$

Die Matrix $\mathbf{M}$ spiegelt dabei die Struktur des simulierten Mehrkörpersystems wider. Die Dimension dieser Matrix ist abhängig von den Gelenken, für die Korrekturimpulse bestimmt werden müssen. Ein Gelenk, das f Freiheitsgrade entfernt, benötigt f Zeilen bzw. Spalten in der Matrix. Summiert man die Freiheitsgrade f_i von allen m Gelenken im System auf, so erhält man die Dimension $n = \sum_{i=1}^{m} f_i$ der Matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ des Gleichungssystems. Der Vektor $\triangle\mathbf{v} = (\triangle\mathbf{v}_1, \ldots, \triangle\mathbf{v}_m)^T \in \mathbb{R}^n$ beinhaltet die zu eliminierenden Geschwindigkeitsdifferenzen aller m Gelenke. Je nachdem, ob es sich bei dem i-ten Gelenk um ein Grundgelenk mit Translations- oder Rotationsbedingungen handelt, ist der Vektor $\triangle\mathbf{v}_i$ eine Differenz von Punktgeschwindigkeiten oder eine Differenz von Winkelgeschwindigkeiten:

$$\triangle\mathbf{v}_i = \begin{cases} \triangle\mathbf{u}_i & \text{falls } i \in T \\ \triangle\boldsymbol{\omega}_i & \text{falls } i \in R. \end{cases}$$

In einer entsprechenden Form wird der Vektor mit den Korrekturimpulsen der Gelenke $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)^T \in \mathbb{R}^n$ aus Impulsen und Drehimpulsen zusammengesetzt:

$$\mathbf{x}_i = \begin{cases} \mathbf{p}_i & \text{falls } i \in T \\ \mathbf{l}_i & \text{falls } i \in R. \end{cases}$$

Durch Lösen des linearen Gleichungssystems werden die Korrekturimpulse $\mathbf{x}$, die die Geschwindigkeiten um $\triangle\mathbf{v}$ ändern, berechnet. Im Unterschied zum iterativen Verfahren werden hier die Impulse für alle Gelenke auf einmal bestimmt.

Die Matrix $\mathbf{M}$ ist eine Blockmatrix. Das heißt, sie wird aus kleineren Matrizen bzw. Blöcken zusammengesetzt. Der Block $\mathbf{M}_{i,j}$ beschreibt dabei, wie die Geschwindigkeiten des i-ten Gelenks von dem Korrekturimpuls des j-ten Gelenks abhängen. Im Fall eines Diagonalblocks $\mathbf{M}_{i,i}$ bedeutet dies, dass die bereits bekannten Matrizen aus den Gleichungen 3.15 bzw. 3.20 verwendet werden können, je nachdem ob es sich um ein Gelenk mit Translations- oder Rotationsbedingungen handelt. Die Dimension eines Blocks $\mathbf{M}_{i,j}$ ist abhängig von der Anzahl der Freiheitsgrade, die die beiden beteiligten Gelenke aus dem System entfernen. Beseitigt das i-te Gelenk f_i Freiheitsgrade und das j-te Gelenk f_j Freiheitsgrade, dann gilt: $\mathbf{M}_{i,j} \in \mathbb{R}^{f_i \times f_j}$. Wenn zwei Gelenke keinen gemeinsamen Körper haben, dann sind die beiden zugehörigen Matrixblöcke $\mathbf{M}_{i,j}$ und $\mathbf{M}_{j,i}$ Nullmatrizen. Die meisten Modelle haben daher ein dünnbesetztes Gleichungssystem zur Bestimmung der Korrekturimpulse.

Im Folgenden wird zunächst der allgemeine Ablauf eines Simulationsschrittes mit dem LGS-Verfahren erklärt. Dann wird beschrieben, wie die gesuchten Matrixblöcke ohne Projektion (also im dreidimensionalen Raum) bestimmt werden. Damit können die Grundgelenke simuliert werden, die drei Freiheitsgrade entfernen. Anschließend wird gezeigt, wie man mit Hilfe von Projektionsmatrizen die Matrixblöcke von den Grundgelenken bildet, die weniger als drei Freiheitsgrade entfernen.

3.3.2.1 Ablauf eines Simulationsschrittes

Bevor ein Simulationsschritt durchgeführt wird, werden die Impulse für die Gelenkkorrektur berechnet (siehe Abbildung 3.24). Dazu wird der Zustand aller Gelenke

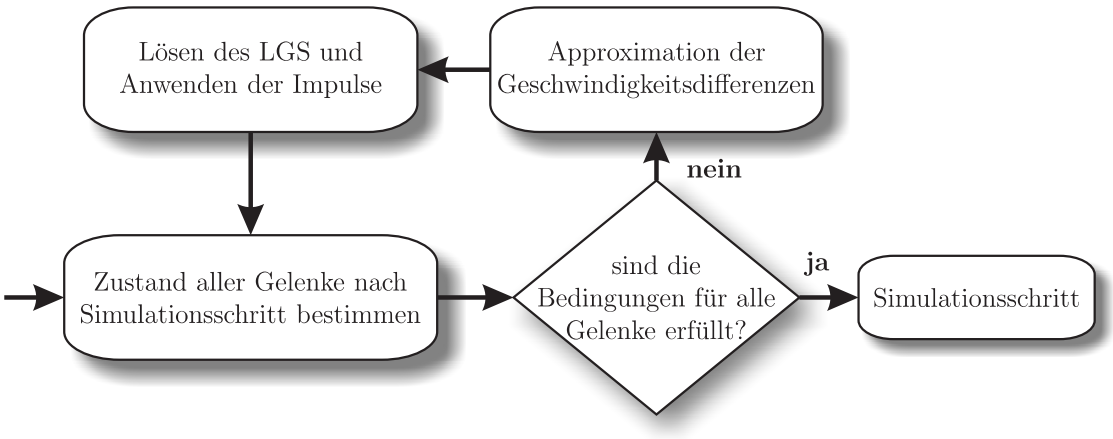


Abbildung 3.24: Ablauf der Gelenkkorrektur mit einem linearen Gleichungssystem

nach diesem Simulationsschritt bestimmt. Anhand des Zustands wird dann überprüft, ob alle Gelenkbedingungen erfüllt sind. Ist dies nicht der Fall, dann wird für jedes Gelenk eine Geschwindigkeitsänderung für die verbundenen Körper approximiert, mit der der Gelenkzustand nach dem Simulationsschritt die geforderte Bedingung erfüllt bzw. den aufgetretenen Fehler reduziert. Anschließend wird ein lineares Gleichungssystem für die Impulse aufgestellt, die exakt diese Geschwindigkeitsänderungen bewirken. Durch das Lösen des Gleichungssystems werden alle Korrekturimpulse bestimmt. Die Gelenkkorrektur wird so lange fortgesetzt, bis die Gelenkbedingungen nach dem Simulationsschritt erfüllt werden.

Nach dem Simulationsschritt erfolgt die Geschwindigkeitskorrektur. Da die benötigten Geschwindigkeitsänderungen hier exakt bestimmt werden können und das lineare Gleichungssystem sämtliche Abhängigkeiten im simulierten Modell berücksichtigt, werden die Korrekturimpulse durch einmaliges Lösen des Gleichungssystems genau berechnet. Im Gegensatz zum iterativen Verfahren werden damit alle Geschwindigkeitsbedingungen der Gelenke in einem einzigen Schritt erfüllt.

In einem Mehrkörpersystem, das ausschließlich Grundgelenke bzw. kombinierte Gelenke beinhaltet, hat die Matrix des linearen Gleichungssystems bei der Gelenk- und der Geschwindigkeitskorrektur die gleiche Form. Außerdem ist sie konstant für einen Zeitpunkt. Deswegen muss sie in jedem Zeitschritt nur einmal neu bestimmt werden und kann dann für die Geschwindigkeitskorrektur und für die Gelenkkorrektur des nächsten Simulationsschrittes verwendet werden. Wenn in dem Mehrkörpersystem zusätzlich Gelenke mit Geschwindigkeitsbedingungen definiert sind, muss das lineare Gleichungssystem der Geschwindigkeitskorrektur um diese Gelen-

ke erweitert werden. Entfernt man die Zeilen und Spalten, die zu diesen Gelenken gehören, aus der Matrix des Gleichungssystems, dann erhält man die Matrix für die nächste Gelenkkorrektur. Gelenke mit Geschwindigkeitsbedingungen werden daher im Folgenden nicht mehr gesondert betrachtet.

3.3.2.2 Bestimmung der Matrixblöcke

Wenn der Matrixblock $\mathbf{M}_{i,j}$ für die Gelenke i und j bestimmt werden soll, muss zunächst überprüft werden, um welche Art von Gelenken es sich handelt. Abhängig von der Art der Gelenke wird dann die folgende Matrix bestimmt:

$$\mathbf{N}_{i,j}(k,\mathbf{a},\mathbf{b}) = \begin{cases} \mathbf{K}_{a,b} & \text{falls } i,j \in T \\ \mathbf{L}_k & \text{falls } i,j \in R \\ \mathbf{W}_{k,b} & \text{falls } i \in R \wedge j \in T \\ \mathbf{U}_{a,k} & \text{falls } i \in T \wedge j \in R. \end{cases}$$

Dabei wird für eine einfachere Schreibweise angenommen, dass jeder Gelenktyp zwei Gelenkpunkte $\mathbf{a}$ und $\mathbf{b}$ hat, die als Parameter übergeben werden. Gelenke mit Rotationsbedingungen benötigen keine Gelenkpunkte, aber da in der Matrix $\mathbf{N}_{i,j}(k,\mathbf{a},\mathbf{b})$ ein Gelenkpunkt nur dann verwendet wird, wenn das zugehörige Grundgelenk nur Translationsbedingungen hat, stellt dies kein Problem dar. Die Wechselwirkungen zwischen den verschiedenen Gelenken können mit Hilfe der vorgestellten Matrix in einer einheitlichen Form beschrieben werden.

Mit der Matrix $\mathbf{N}_{i,j}(k,\mathbf{a},\mathbf{b})$ kann ein lineares Gleichungssystem $\tilde{\mathbf{M}} \cdot \tilde{\mathbf{x}} = \triangle\tilde{\mathbf{v}}$ für dreidimensionale Grundgelenke aufgestellt werden. Ein Block $\tilde{\mathbf{M}}_{i,j}$ der Matrix des Gleichungssystems beschreibt, wie sich der Impuls bzw. Drehimpuls des j-ten Gelenks auf die Geschwindigkeiten des i-ten Gelenks auswirkt. Die durch den Impuls $\tilde{\mathbf{x}}_j$ bewirkte Geschwindigkeitsänderung im Gelenk i wird im Folgenden durch den Vektor $\triangle\tilde{\mathbf{v}}_{i,j}$ dargestellt. Außerdem bezeichnen die Indizes k_{i_1} und k_{i_2} den ersten bzw. den zweiten Körper des i-ten Gelenks. Bei den Abhängigkeiten der Gelenke muss zwischen sieben Fällen unterschieden werden:

1. Fall: $k_{i_1} = k_{j_1} \ \wedge \ k_{i_2} \neq k_{j_2}$

   Die beiden Gelenke haben einen gemeinsamen Körper und zwar den jeweils ersten Körper der beiden Gelenke. In diesem Fall wirkt der Impuls des j-ten Gelenks $\tilde{\mathbf{x}}_j$ positiv auf den gemeinsamen Körper. Dadurch ergibt sich die folgende Änderung der relativen Geschwindigkeit des i-ten Gelenks:

   $$\mathbf{N}_{i,j}(k_{i_1},\mathbf{a}_i,\mathbf{a}_j) \cdot \tilde{\mathbf{x}}_j = \triangle\tilde{\mathbf{v}}_{i,j}.$$

2. Fall: $k_{i_2} = k_{j_2} \ \wedge \ k_{i_1} \neq k_{j_1}$

Der gemeinsame Körper der beiden Gelenke ist in diesem Fall der jeweils zweite Körper. Der Impuls $\tilde{\mathbf{x}}_j$ des zweiten Gelenks wirkt daher negativ auf diesen Körper. Dies hat zur Folge, dass sich die relative Geschwindigkeit wie folgt verändert:

$$-\mathbf{N}_{i,j}(k_{i_2}, \mathbf{b}_i, \mathbf{b}_j) \cdot (-\tilde{\mathbf{x}}_j) = \triangle\tilde{\mathbf{v}}_{i,j}.$$

3. Fall: $k_{i_1} = k_{j_1} \ \wedge \ k_{i_2} = k_{j_2}$

   Die beiden Gelenke verbinden die gleichen Körper. Dies bedeutet, dass sie zwei gemeinsame Körper haben. Deswegen müssen beide Impulse des j-ten Gelenks $\tilde{\mathbf{x}}_j$ und $-\tilde{\mathbf{x}}_j$ bei der Berechnung der relativen Geschwindigkeitsänderung im i-ten Gelenk berücksichtigt werden:

$$\mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{a}_j) \cdot \tilde{\mathbf{x}}_j - \mathbf{N}_{i,j}(k_{i_2}, \mathbf{b}_i, \mathbf{b}_j) \cdot (-\tilde{\mathbf{x}}_j) = \triangle\tilde{\mathbf{v}}_{i,j}.$$

4. Fall: $k_{i_1} = k_{j_2} \ \wedge \ k_{i_2} \neq k_{j_1}$

   In diesem Fall sind der erste Körper des i-ten Gelenks und der zweite Körper des j-ten Gelenks identisch. Das heißt, dass der Impuls $-\tilde{\mathbf{x}}_j$ auf den gemeinsamen Körper wirkt:

$$\mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{b}_j) \cdot (-\tilde{\mathbf{x}}_j) = \triangle\tilde{\mathbf{v}}_{i,j}.$$

5. Fall: $k_{i_2} = k_{j_1} \ \wedge \ k_{i_1} \neq k_{j_2}$

   Der gemeinsame Körper der beiden Gelenke ist der zweite Körper des i-ten Gelenks bzw. der erste Körper des j-ten Gelenks. Die relative Geschwindigkeit im Gelenk i verändert sich durch den Impuls $\tilde{\mathbf{x}}_j$ wie folgt:

$$-\mathbf{N}_{i,j}(k_{i_1}, \mathbf{b}_i, \mathbf{a}_j) \cdot \tilde{\mathbf{x}}_j = \triangle\tilde{\mathbf{v}}_{i,j}.$$

6. Fall: $k_{i_1} = k_{j_2} \ \wedge \ k_{i_2} = k_{j_1}$

   Die beiden Gelenke verbinden die gleichen Körper über Kreuz. Das heißt, der erste Körper von Gelenk i ist der zweite Körper von Gelenk j und umgekehrt. Dadurch ergibt sich für das i-te Gelenk die folgende relative Geschwindigkeitsänderung:

$$\mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{b}_j) \cdot (-\tilde{\mathbf{x}}_j) - \mathbf{N}_{i,j}(k_{i_1}, \mathbf{b}_i, \mathbf{a}_j) \cdot \tilde{\mathbf{x}}_j = \triangle\tilde{\mathbf{v}}_{i,j}.$$

7. Fall: $k_{i_1} \neq k_{j_1} \ \wedge \ k_{i_2} \neq k_{j_2} \ \wedge \ k_{i_1} \neq k_{j_2} \ \wedge \ k_{i_2} \neq k_{j_1}$

   In diesem Fall haben die beiden Gelenke keinen gemeinsamen Körper und sind nicht direkt abhängig voneinander.

Damit die Impulse $\tilde{\mathbf{x}}_j$ die Geschwindigkeitsänderungen $\triangle\tilde{\mathbf{v}}_i$ bewirken, muss für alle Gelenke $i \in T \cup R$ Folgendes gelten:

$$\sum_{j=1}^{m} \triangle\tilde{\mathbf{v}}_{i,j} = \triangle\tilde{\mathbf{v}}_i$$

Dadurch können die Matrixblöcke $\tilde{\mathbf{M}}_{i,j}$ des linearen Gleichungssystems bestimmt werden:

$$\tilde{\mathbf{M}}_{i,j} = \begin{cases} \mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{a}_j) & \text{falls } k_{i_1} = k_{j_1} \ \wedge \ k_{i_2} \neq k_{j_2} \\ \mathbf{N}_{i,j}(k_{i_2}, \mathbf{b}_i, \mathbf{b}_j) & \text{falls } k_{i_2} = k_{j_2} \ \wedge \ k_{i_1} \neq k_{j_1} \\ \mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{a}_j) + N_{i,j}(k_{i_2}, \mathbf{b}_i, \mathbf{b}_j) & \text{falls } k_{i_1} = k_{j_1} \ \wedge \ k_{i_2} = k_{j_2} \\ -\mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{b}_j) & \text{falls } k_{i_1} = k_{j_2} \ \wedge \ k_{i_2} \neq k_{j_1} \\ -\mathbf{N}_{i,j}(k_{i_2}, \mathbf{b}_i, \mathbf{a}_j) & \text{falls } k_{i_2} = k_{j_1} \ \wedge \ k_{i_1} \neq k_{j_2} \\ -\mathbf{N}_{i,j}(k_{i_1}, \mathbf{a}_i, \mathbf{b}_j) - N_{i,j}(k_{i_2}, \mathbf{b}_i, \mathbf{a}_j) & \text{falls } k_{i_1} = k_{j_2} \ \wedge \ k_{i_2} = k_{j_1} \\ \mathbf{0} & \text{sonst.} \end{cases}$$

Durch Lösen des Gleichungssystems $\tilde{\mathbf{M}} \cdot \tilde{\mathbf{x}} = \triangle\tilde{\mathbf{v}}$ ergeben sich die Impulse, die für die Simulation von Grundgelenken mit dreidimensionalen Zwangsbedingungen benötigt werden.

In Abschnitt 3.2.3 wurde beschrieben, wie die Gleichungen der Korrekturimpulse für einzelne Grundgelenke, die weniger als drei Freiheitsgrade entfernen, mit Hilfe von Projektionsmatrizen aufgestellt werden. Mit den gleichen Projektionsmatrizen werden die Matrixblöcke $\tilde{\mathbf{M}}_{i,j}$ in das jeweils richtige Koordinatensystem projiziert. Der Korrekturimpuls $\mathbf{x}_j \in \mathbb{R}^{f_j}$ des j-ten Grundgelenks, das f_j Freiheitsgrade aus dem System entfernt, bewirkt im i-ten Gelenk die Geschwindigkeitsänderung $\mathbf{P}_i \triangle\tilde{\mathbf{v}}_{i,j}$, wobei $\mathbf{P}_i$ die Projektionsmatrix des i-ten Gelenks ist. Dieser Impuls muss deswegen die folgende Gleichung erfüllen:

$$\mathbf{P}_i\tilde{\mathbf{M}}_{i,j}\mathbf{P}_j^T\,\mathbf{x}_j = \mathbf{P}_i \triangle\tilde{\mathbf{v}}_{i,j}.$$

Dadurch ergibt sich für die Matrixblöcke $\mathbf{M}_{i,j}$ des linearen Gleichungssystems bei allen Grundgelenktypen:

$$\mathbf{M}_{i,j} = \mathbf{P}_i\tilde{\mathbf{M}}_{i,j}\mathbf{P}_j^T.$$

Auf der rechten Seite des Gleichungssystems steht der Vektor $\triangle\mathbf{v}$ mit den Geschwindigkeitsänderungen, die für die Gelenk- bzw. für die Geschwindigkeitskorrektur benötigt werden. Dieser Vektor setzt sich aus den projizierten Geschwindigkeitsdifferenzen der einzelnen Grundgelenke zusammen:

$$\triangle\mathbf{v}_i = \mathbf{P}_i \triangle\tilde{\mathbf{v}}_i.$$

Insgesamt hat dann das lineare Gleichungssystem, mit dem die Korrekturimpulse bestimmt werden, die folgende Form:

$$\begin{pmatrix} \mathbf{P}_1\tilde{\mathbf{M}}_{1,1}\mathbf{P}_1^T & \dots & \mathbf{P}_1\tilde{\mathbf{M}}_{1,m}\mathbf{P}_m^T \\ \vdots & \ddots & \vdots \\ \mathbf{P}_m\tilde{\mathbf{M}}_{m,1}\mathbf{P}_1^T & \dots & \mathbf{P}_m\tilde{\mathbf{M}}_{m,m}\mathbf{P}_m^T \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} \mathbf{P}_1 \triangle\tilde{\mathbf{v}}_1 \\ \vdots \\ \mathbf{P}_m \triangle\tilde{\mathbf{v}}_m \end{pmatrix}.$$

Durch Lösen dieses Gleichungssystems werden die Impulse für alle Gelenke bestimmt. Ein Impuls wird dabei im jeweiligen Koordinatensystem des zugehörigen Gelenks berechnet. Bevor die Impulse angewendet werden, müssen sie daher noch in das Weltkoordinatensystem transformiert werden.

3.3.2.3 Behandlung von geschlossenen kinematischen Ketten

Die Simulation eines Modells, das geschlossene kinematische Ketten enthält, ist mit dem iterativen Verfahren ohne Sonderbehandlung möglich. Beim Verfahren mit linearen Gleichungssystemen kann die Simulation eines solchen Modells Probleme verursachen. Der Grund dafür wird anhand eines Beispiels erläutert. Das Modell in Abbildung 3.25 besteht aus sechs Körpern, die mit sechs Drehgelenken verbunden sind. Einer der Körper ist statisch. Durch die Drehgelenke werden die Körper zu einer geschlossenen kinematischen Kette verbunden. Dieses Modell hat einen Freiheitsgrad [Wit77]. Jeder der fünf dynamischen Körper hat sechs Freiheitsgra-

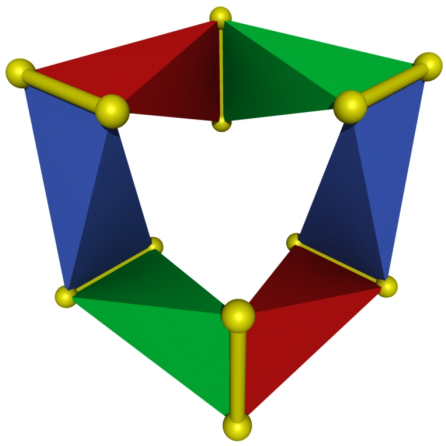


Abbildung 3.25: Geschlossene kinematische Kette mit einem Freiheitsgrad

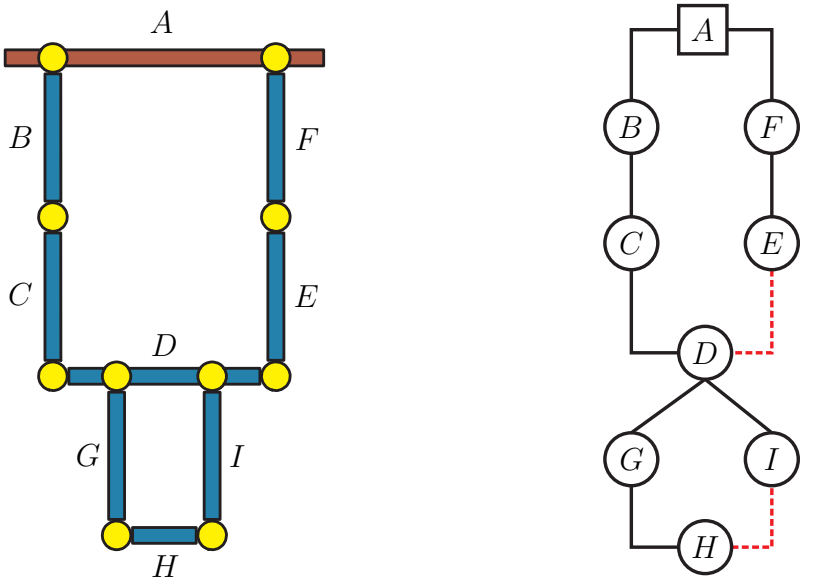
de. Dadurch gibt es im Mehrkörpersystem insgesamt 30 Freiheitsgrade. Durch die Drehgelenke werden also 29 Freiheitsgrade aus dem System entfernt. Wenn das lineare Gleichungssystem für das Modell wie beschrieben aufgestellt wird, hat es allerdings die Dimension $n = 30$, da jedes Drehgelenk, wenn man es einzeln betrachtet, fünf Freiheitsgrade entfernt. Dies bedeutet, dass das Gleichungssystem

überbestimmt ist. Mit einem solchem Gleichungssystem kann nicht in jedem Fall eine korrekte Lösung für die Korrekturimpulse gefunden werden. Dadurch kann die Simulation des Modells instabil werden.

In Modellen mit überbestimmten Gleichungssystemen existieren Gelenke, die die gleichen Freiheitsgrade entfernen. Die Zwangsbedingungen dieser Gelenke sind dann teilweise redundant. Das iterative Verfahren kann die Simulation mit solchen Zwangsbedingungen durchführen. Allerdings müssen Modelle mit geschlossenen kinematischen Ketten beim LGS-Verfahren besonders behandelt werden, damit die Simulation stabil bleibt. Dafür gibt es zwei Möglichkeiten, die im Folgenden beschrieben werden.

Auftrennen der Schleife Die erste Möglichkeit besteht darin, eine Schleife im Modell an einem Gelenk aufzutrennen. Das Gelenk, an dem die Schleife geöffnet wurde, und die restliche Kette werden dann getrennt simuliert.

Bevor eine Schleife behandelt werden kann, muss sie jedoch zuerst im Modell erkannt werden. Dafür wird ein ungerichteter Graph verwendet, der die Verbindungen der Gelenke im Modell widerspiegelt (siehe Abbildung 3.26). Dieser Graph wird im Folgenden *Verbindungsgraph* genannt. Die geschlossenen Ketten in einem Modell müssen mit dem Verbindungsgraph vor Beginn der Simulation einmal bestimmt werden. Jeder Körper wird durch einen Knoten im Graph repräsentiert.



(a) Modell mit zwei geschlossenen kinematischen Ketten

(b) Verbindungsgraph der Gelenke

Abbildung 3.26: Mit dem Verbindungsgraph der Gelenke können geschlossene kinematische Ketten in einem Modell gefunden und aufgetrennt werden.

Zwischen zwei Knoten existiert genau dann eine Kante, wenn die zwei zugehö-

rigen Starrkörper im Modell durch ein Gelenk miteinander verbunden sind. Mit dem Verbindungsgraph können Schleifen und voneinander unabhängige Ketten wie folgt gefunden werden. Zunächst werden alle Knoten in eine Liste eingetragen. Jeder Knoten kennt die Kanten, mit denen er verbunden ist. Ein beliebiger Knoten wird aus der Liste entfernt und markiert. Dieser Knoten gehört zur ersten Kette. Anschließend werden alle Kanten, die mit diesem Knoten verbunden sind, abgelaufen und markiert. Jeder erreichte Knoten, der noch nicht markiert ist, wird zu der Kette hinzugefügt und markiert. Wenn ein erreichter Knoten bereits markiert ist, handelt es sich um eine Schleife im Graph. In diesem Fall wird die Kante, die zu dem markierten Knoten geführt hat, in einer Liste abgespeichert. Alle Kanten in dieser Liste gehören zu Gelenken, an denen später die Schleifen im Modell aufgebrochen werden können. Die unmarkierten Kanten aller Knoten, die zu der Kette hinzugefügt wurden, werden auf diese Weise weiter abgelaufen, bis kein Knoten der Kette noch eine unmarkierte Kante besitzt. Damit ist die erste Kette mit allen Schleifen vollständig bestimmt. Ist danach die Liste der Knoten im Verbindungsgraph nicht leer, dann existieren weitere Ketten. Diese können mit der gleichen Vorgehensweise wie bei der ersten Kette bestimmt und auf Schleifen überprüft werden. Dadurch erhält man alle voneinander unabhängigen Ketten im Mehrkörpersystem und alle Gelenke, an denen geschlossene kinematische Ketten im Modell aufgetrennt werden können. Die Bestimmung der unabhängigen Ketten kann bei der Gelenkkorrektur von Vorteil sein. Da sich die Korrekturimpulse dieser Ketten nicht gegenseitig beeinflussen, können sie mit voneinander unabhängigen Gleichungssystemen bestimmt werden. Wenn die beiden Ketten eine unterschiedliche Anzahl von Iterationen zur Korrektur benötigen, kann durch eine getrennte Berechnung der Korrekturimpulse Rechenzeit eingespart werden.

In Abbildung 3.26(a) ist ein Modell mit zwei Schleifen dargestellt. Der dazugehörige Verbindungsgraph befindet sich in Abbildung 3.26(b). Die zwei Kanten, bei denen Schleifen im Graph festgestellt wurden, sind hervorgehoben. Werden die beiden zugehörigen Gelenke entfernt und damit die geschlossenen kinematischen Ketten des Modells aufgetrennt, entsteht eine Kette ohne Schleifen. Diese kann mit Hilfe des LGS-Verfahrens simuliert werden. Die Impulse für die Gelenk- bzw. Geschwindigkeitskorrektur der beiden Gelenke, die entfernt wurden, werden separat bestimmt. Um die Korrekturen durchzuführen, wird das iterative Verfahren mit dem LGS-Verfahren kombiniert. Bei der Gelenk- und bei der Geschwindigkeitskorrektur werden die Impulse jeweils in einer iterativen Schleife berechnet. Dabei wird in jedem Iterationsschritt ein lineares Gleichungssystem für die Impulse der Kette gelöst, während die Impulse der aus der Kette entfernten Gelenke wie beim iterativen Verfahren einzeln berechnet werden. Durch diese Vorgehensweise lässt sich ausnutzen, dass das iterative Verfahren geschlossene kinematische Ketten ohne Sonderbehandlung simulieren kann. Gleichzeitig werden die Abhängigkeiten der Kette durch die Verwendung eines Gleichungssystems berücksichtigt.

Modifikation der Gelenke Alternativ zum Auftrennen von Schleifen in einem Modell gibt es eine weitere Möglichkeit, geschlossene kinematische Ketten mit dem LGS-Verfahren zu simulieren. Damit das lineare Gleichungssystem zur Bestimmung der Korrekturimpulse nicht überbestimmt ist, müssen die verwendeten Gelenke vor der Simulation angepasst werden. Dabei muss die Summe der Freiheitsgrade, die durch die einzelnen Gelenke entfernt werden, den tatsächlich entfernten Freiheitsgraden des gesamten Mehrkörpersystems entsprechen. Auf diese Weise bekommt das Gleichungssystem die benötigte Dimension. Dies kann am Beispiel des Modells in Abbildung 3.25 veranschaulicht werden. Bei diesem Modell kann eines der Drehgelenke durch ein Kardangelenk ersetzt werden, ohne dass das Mehrkörpersystem danach mehr Freiheitsgrade hat. Die erste Rotationsachse des Kardangelenks muss dabei die gleiche Richtung haben wie die des ersetzten Drehgelenks. Die zweite Achse muss so gewählt werden, dass sich durch die Achse in keiner möglichen Konstellation der Körper des Modells ein neuer Freiheitsgrad ergibt. Dies kann erreicht werden, wenn die zweite Achse senkrecht auf der ersten Achse und auf der Rotationsachse eines benachbarten Drehgelenks in der geschlossenen Kette steht. Durch die Verwendung eines Kardangelenks, das vier Freiheitsgrade entfernt, hat das zugehörige lineare Gleichungssystem zur Bestimmung der Korrekturimpulse die Dimension $n = 29$. Das Gleichungssystem ist daher nicht mehr überbestimmt und das Modell kann stabil mit dem LGS-Verfahren simuliert werden.

Bei beiden Methoden zur Behandlung von Schleifen in einem Mehrkörpersystem müssen die Schleifen vor der Simulation bestimmt werden. Die erste Methode erfordert eine Sonderbehandlung während eines Simulationsschritts für Gelenke, an denen die vorhandenen Schleifen aufgetrennt wurden. Dadurch wird die Simulation des Modells langsamer. Die zweite Methode erlaubt eine Simulation von Modellen mit Schleifen in der gleichen Geschwindigkeit wie bei Modellen ohne Schleifen, da das Ersetzen eines Gelenks vor Beginn der Simulation vorgenommen werden kann und dann keine Sonderbehandlung während eines Zeitschrittes notwendig ist. Allerdings muss bei der zweiten Methode zunächst eine passende Ersetzung der Gelenke gefunden werden, was bisher noch nicht automatisch vorgenommen werden kann.

3.3.3 Vergleich der vorgestellten Verfahren

Das iterative Verfahren ist sehr einfach zu implementieren. Alle Gleichungen, die gelöst werden müssen, haben höchstens die Dimension drei. Da die Gelenke nur einzeln betrachtet werden, sind die Gleichungen nicht komplex und schnell lösbar. Die Impulse können jeweils durch die Invertierung einer regulären, symmetrischen Matrix bestimmt werden, wobei die Inverse nur einmal pro Simulationsschritt berechnet werden muss. Außerdem können Modelle mit geschlossenen kinematischen Ketten mit dem iterativen Verfahren ohne Sonderbehandlung simuliert werden.

Das LGS-Verfahren berücksichtigt sämtliche Abhängigkeiten zwischen den Gelenken. Die Abhängigkeiten werden in einem linearen Gleichungssystem beschrieben, welches gelöst werden muss, um die benötigten Korrekturimpulse zu bestimmen. Beim LGS-Verfahren müssen Modelle mit geschlossenen Ketten allerdings gesondert behandelt werden. Das LGS-Verfahren ist dadurch aufwendiger zu implementieren als das iterative Verfahren. Das lineare Gleichungssystem kann mit Hilfe einer LU-Faktorisierung gelöst werden. Dafür kann zum Beispiel LAPACK [ABB+99] verwendet werden. Da das Gleichungssystem für die meisten Modelle eine dünn besetzte Matrix hat, ist die Wahl eines Gleichungssystemlösers, der diese Eigenschaft ausnutzt, von Vorteil. Aus diesem Grund wurde in dieser Arbeit PARDISO (Parallel Direct Sparse Solver) verwendet. PARDISO wurde von Olaf Schenk und Klaus Gärtner entwickelt [SGFS01, SG02, SG04a, SG04b] und unterstützt das parallele Lösen von dünn besetzten linearen Gleichungssystemen auf mehreren Prozessoren. Da die Matrix für einen Zeitpunkt konstant ist, muss die Faktorisierung, die für die Lösung des Gleichungssystems benötigt wird, nur einmal pro Simulationsschritt berechnet werden. Anschließend kann sie sowohl bei der Geschwindigkeitskorrektur als auch in allen Iterationsschritten der nächsten Gelenkkorrektur verwendet werden. Dadurch können die benötigten Korrekturimpulse mit dem LGS-Verfahren schnell berechnet werden, obwohl in jedem Iterationsschritt ein lineares Gleichungssystem gelöst werden muss. Ein Vorteil des LGS-Verfahrens gegenüber dem iterativen Verfahren ist, dass die Impulse bei der Geschwindigkeitskorrektur in einem Schritt exakt berechnet werden können. Besonders bei komplexen Systemen bringt dies einen erheblichen Geschwindigkeitsvorteil.

Beide Verfahren haben die Eigenschaft, dass sie sogar Systeme, deren Gelenke völlig zerfallen sind, wieder zusammensetzen können. Bei der Gelenkkorrektur wird für ein Gelenk der Fehler bestimmt, den es nach dem nächsten Simulationsschritt haben wird. Dieser Fehler wird anschließend mit Hilfe eines Impulses reduziert. Dabei ist es nebensächlich, ob das Mehrkörpersystem zum aktuellen Zeitpunkt einen korrekten Zustand hat oder nicht. Beim Zusammensetzen zerfallener Systeme muss allerdings berücksichtigt werden, dass die Korrekturimpulse sehr groß werden können, wenn der Fehler groß ist. Dadurch werden die zugehörigen Körper extrem beschleunigt, was dazu führen kann, dass die Simulation durch numerische Fehler instabil wird. Dieses Problem kann durch eine Beschränkung der Impulse bei der Gelenkkorrektur gelöst werden. Das bedeutet, dass für jedes Gelenk eine obere Grenze für den Betrag des angewendeten Korrekturimpulses in einem Simulationsschritt festgelegt ist. Aufgrund der Beschränkung der Impulse erreicht ein völlig zerfallenes System erst nach einigen Simulationsschritten einen zulässigen Zustand. Dagegen hat die Beschränkung bei kleinen Fehlern der Gelenkzustände keine Auswirkung, da in diesem Fall auch die Korrekturimpulse relativ klein sind.

Das Zusammensetzen von Mehrkörpersystemen mit unzulässigen Gelenkzuständen kann z. B. beim Modellieren verwendet werden. Wenn zwei Körper mit einem Kugelgelenk verbunden werden sollen, dann müssen sie nicht unbedingt an der

richtigen Position sein. Es genügt, wenn man in jedem der beiden Körper einen Gelenkpunkt definiert. Die dynamische Simulation sorgt anschließend dafür, dass die beiden Körper in ihren Gelenkpunkten zusammengezogen werden. Alternativ kann diese Eigenschaft der impulsbasierten Simulation ausgenutzt werden, um die Simulation auf Kosten der Genauigkeit zu beschleunigen. Dies ist besonders dann interessant, wenn die Geschwindigkeit der Simulation wesentlich wichtiger ist als die Genauigkeit des Ergebnisses. Da die Berechnung der Korrekturimpulse mit beiden vorgestellten Verfahren iterativ abläuft, besteht auch die Möglichkeit, die Iterationsschleife vorzeitig abzubrechen. Dadurch erhält man sofort nach dem Abbruch ein Ergebnis und kann mit der Simulation fortfahren. Durch den vorzeitigen Abbruch der Iteration können die vorgegebenen Toleranzwerte bei der Gelenk- bzw. der Geschwindigkeitskorrektur nicht erreicht werden. Dadurch entsteht ein System mit unzulässigen Gelenkzuständen. Allerdings sind die auftretenden Fehler i. d. R. relativ klein. Im nächsten Simulationsschritt können diese Fehler dann wieder korrigiert werden. In vielen Anwendungen der virtuellen Realität ist ein plausibles Ergebnis durchaus akzeptabel, solange eine vorgegebene Bildwiederholrate aufrecht erhalten werden kann. Dies wird durch das Abbrechen der Iterationsschleife möglich, was einen entscheidenden Vorteil dieser Methode ausmacht.

3.3.3.1 Messwerte

Beide Verfahren haben ihre Vor- und Nachteile. Diese werden im Folgenden durch genaue Messungen bei verschiedenen Modellen verdeutlicht. Alle Messungen in diesem Abschnitt wurden auf einem PC mit einem 3,4 GHz Intel Pentium 4 Prozessor durchgeführt. Für die numerische Integration der Differentialgleichungen während der Simulation wurde das Runge-Kutta-Verfahren vierter Ordnung (siehe Anhang B.1.3) verwendet.

Kette Die ersten Messungen wurden mit Ketten von Starrkörpern durchgeführt. Die einzelnen Körper der Kette sind durch Kugelgelenke miteinander verbunden. Abbildung 3.27 zeigt eine Kette, die aus fünf Körpern besteht. Jeder Körper im Modell ist 1 m lang, 10 cm breit und 10 cm hoch. Die Masse eines Körpers beträgt 1 kg. Der oberste Körper ist an einem festen Punkt mit Hilfe eines Kugelgelenks aufgehängt. Alle weiteren Körper der Kette hängen jeweils am Ende des vorherigen Körpers. Das erste und das letzte Gelenk in einer Kette sind jeweils von einem anderen Gelenk abhängig. Dagegen hängen die Kugelgelenke in der Mitte der Kette von je zwei anderen Gelenken ab. Die Gelenkstruktur ist also relativ einfach. Es wurde eine Zeitschrittweite von $h = 0,01\,\mathrm{s}$ für die Simulation des Modells verwendet. Die Toleranzwerte wurden auf $\varepsilon_d = 10^{-6}\,\mathrm{m}$ bzw. $\varepsilon_v = 10^{-6}\,\frac{\mathrm{m}}{\mathrm{s}}$ gesetzt.

Die Simulation wurde mit einer Kette, die aus einem Körper bestand, gestartet. Nach jeweils 200 Simulationsschritten bzw. zwei Sekunden Simulationszeit wurden

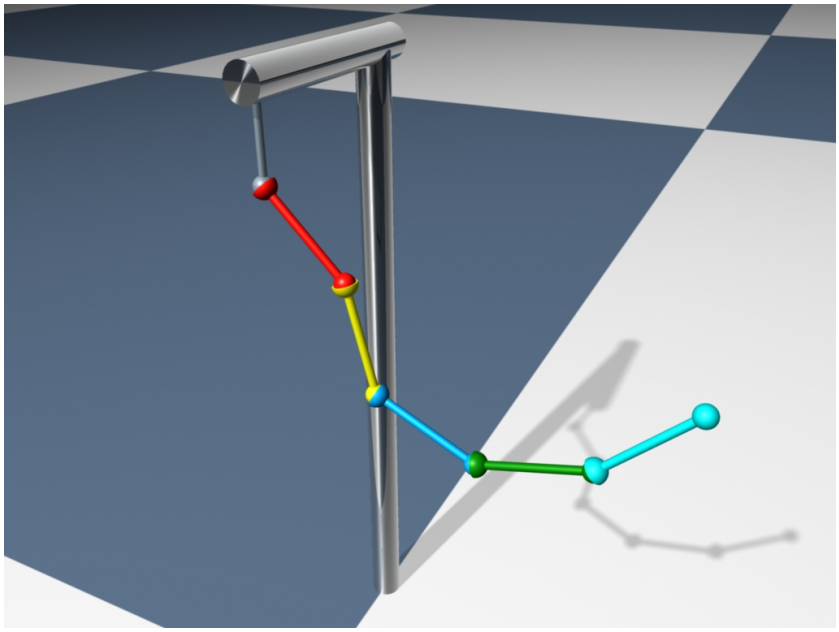


Abbildung 3.27: Kette mit fünf Kugelgelenken

alle Körper der Kette auf ihre Anfangspositionen zurückgesetzt. Dann wurde ein weiterer Körper an die Kette unten angehängt. Die Anfangspositionen der Körper sind so gewählt, dass die Kette exakt nach unten hängt. Damit die Kette in Bewegung kommt, wird am Anfang von jedem Zeitintervall über zwei Sekunden die Geschwindigkeit des Schwerpunktes von jedem Körper auf $2\,\frac{\mathrm{m}}{\mathrm{s}}$ in jeweils unterschiedliche Richtungen gesetzt. Das Ziel der ersten Messung war, das Laufzeitverhalten des iterativen und des LGS-Verfahrens bei einer steigenden Anzahl an Gelenken zu untersuchen. Mit jedem Verfahren wurde das beschriebene Modell über eine Dauer von 100 Sekunden simuliert. Am Ende der Simulation bestand die Kette aus 50 Körpern. Die Ergebnisse dieser Messung sind in Abbildung 3.28 dargestellt. Sie zeigen, dass das iterative Verfahren bis zu einer Kette von acht Körpern schneller ist als das LGS-Verfahren. Für mehr als acht Körper steigt beim iterativen Verfahren die Rechenzeit, die für einen Simulationsschritt benötigt wird, schnell an. Die Rechenzeit des LGS-Verfahrens steigt nur sehr langsam mit der Anzahl der Körper in der Kette an. Die Simulation einer Kette mit 50 Körpern war bei der Messung ungefähr um den Faktor 3 langsamer als die Simulation mit 10 Körpern (siehe Tabelle 3.1). Damit war sogar die Simulation des Modells mit 50 Gelenken fast zweimal schneller als Echtzeit. Dagegen war beim iterativen Verfahren die Simulation mit 10 Körpern um den Faktor 87 schneller im Vergleich zu der Kette mit 50 Körpern. Ab einer Kette von 15 Körpern war das iterative Verfahren nicht mehr echtzeitfähig.

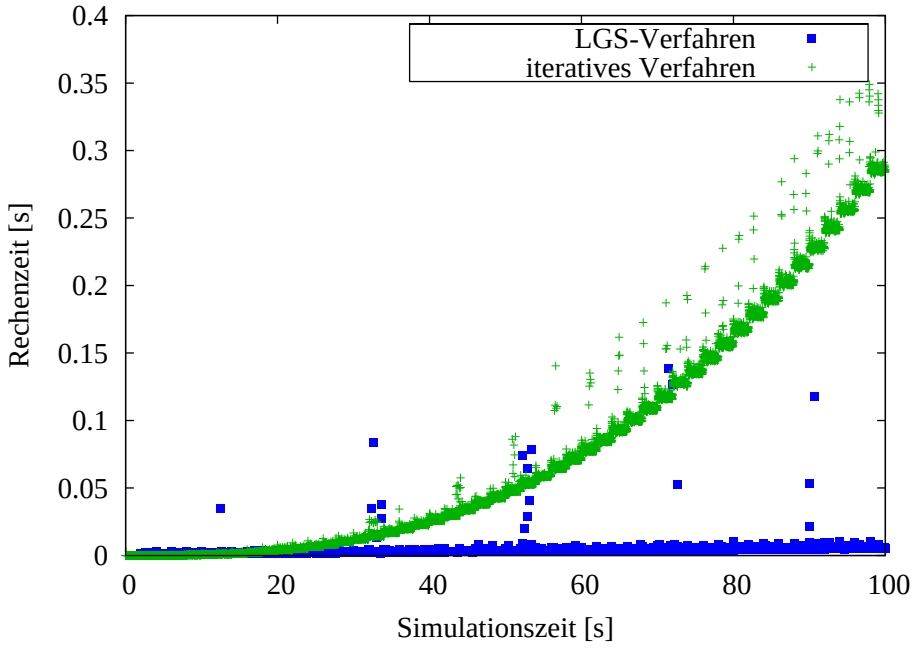


Abbildung 3.28: Rechenzeit pro Simulationsschritt. Nach jeweils zwei Sekunden Simulationszeit wurde die Kette um einen Körper erweitert.

	iteratives Verfahren	LGS-Verfahren
Kette mit 10 Körpern	3,30 ms	2,11 ms
Kette mit 20 Körpern	23,20 ms	2,78 ms
Kette mit 30 Körpern	71,65 ms	3,63 ms
Kette mit 40 Körpern	157,69 ms	4,60 ms
Kette mit 50 Körpern	287,14 ms	6,28 ms

Tabelle 3.1: Durchschnittliche Rechenzeit eines Simulationsschrittes

Baum Für die nächsten Messungen wird ein Baum von Starrkörpern als Modell verwendet. Ein solcher Baum mit sieben Ebenen und 127 Körpern ist in Abbildung 3.29 dargestellt. Die Körper sind durch Kugelgelenke mit drei Freiheitsgraden miteinander verbunden. Bei der Simulation sind die Korrekturimpulse der Gelenke abhängig voneinander, da sie jeweils einen gemeinsamen Körper mit anderen Gelenken haben. Die Gelenke in der Wurzel und in den Blättern des Baums sind jeweils von zwei weiteren Gelenken abhängig, während alle anderen Gelenke jeweils von vier weiteren abhängen. Jeder Körper im Modell ist 4 cm breit und 4 cm hoch. Die Länge eines Körpers wird abhängig von seiner Tiefe im Baum durch die Gleichung

$$l = 1{,}5^{(\text{Maximale Tiefe} - \text{Aktuelle Tiefe})} \cdot 0{,}1\,\text{m}$$

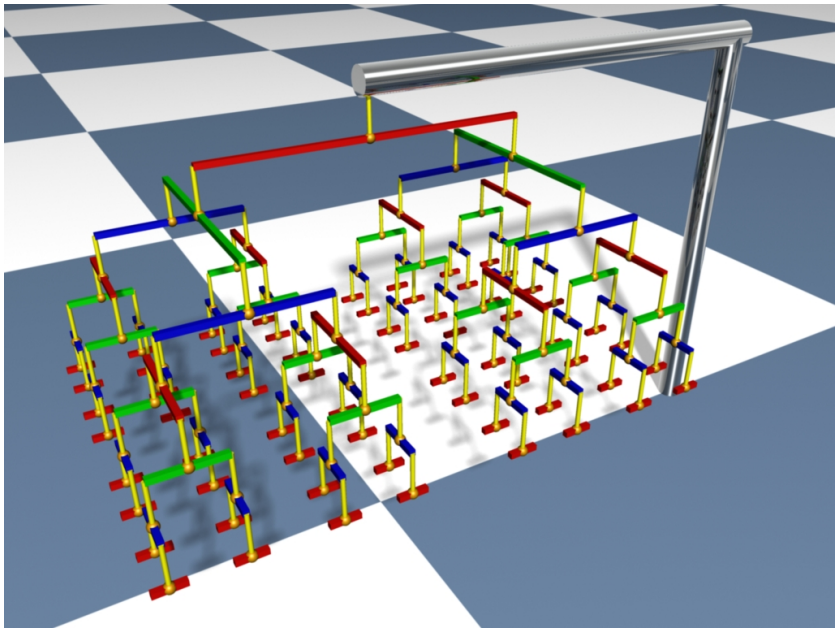


Abbildung 3.29: Ein Baum mit 127 Körpern, die durch 127 Kugelgelenke verbunden sind

bestimmt. Dadurch ergeben sich Längen zwischen 10 cm und 1,14 m für die Körper des Baums aus der Abbildung. Die Starrkörper haben eine gleichmäßige Dichte von $\rho = 600\,\frac{\mathrm{kg}}{\mathrm{m}^3}$, mit der die Massen und die Trägheitstensoren für die Simulation berechnet werden. Bei der Gelenkkorrektur wird der Toleranzwert $\varepsilon_d = 10^{-6}\,\mathrm{m}$ verwendet und bei der Geschwindigkeitskorrektur der Wert $\varepsilon_v = 10^{-6}\,\frac{\mathrm{m}}{\mathrm{s}}$. Die beiden Toleranzwerte werden sehr klein gewählt, um sehr genaue Simulationsergebnisse zu erzielen. Der Toleranzwert für die Geschwindigkeitskorrektur wird nur beim iterativen Verfahren benötigt, da das LGS-Verfahren die genauen Korrekturimpulse in einem Schritt berechnen kann. Bei einer Bildwiederholrate von 30 Bildern pro Sekunde wird eine Bewegung in den Bildern vom menschlichen Auge als flüssige Bewegung wahrgenommen. Die Simulation des Modells wird daher mit einer Zeitschrittweite von $h = \frac{1}{30}\,\mathrm{s}$ durchgeführt. Am Anfang der Simulation wird der oberste Körper des Baums während eines Simulationsschrittes mit Hilfe eines konstanten Drehmoments von $\boldsymbol{\tau}_{ext} = 10\,\mathrm{Nm}$ beschleunigt. Das Drehmoment bewirkt eine Rotation des gesamten Baums.

Die Simulation des Baums wurde mit dem iterativen und mit dem LGS-Verfahren durchgeführt. Dabei wurden die Zeiten gemessen, die das Simulationsverfahren für die Berechnungen während eines Zeitschrittes benötigt hat. Die Ergebnisse dieser Messungen sind in Abbildung 3.30 dargestellt. Es wurden mit beiden Verfahren jeweils 500 Simulationsschritte durchgeführt. Das bedeutet, die Simulation lief 16,67 s. Die mittleren Rechenzeiten der Verfahren sowie die durchschnittlich benötigte Anzahl an Iterationsschritten für die Gelenk- bzw. Geschwindigkeits-

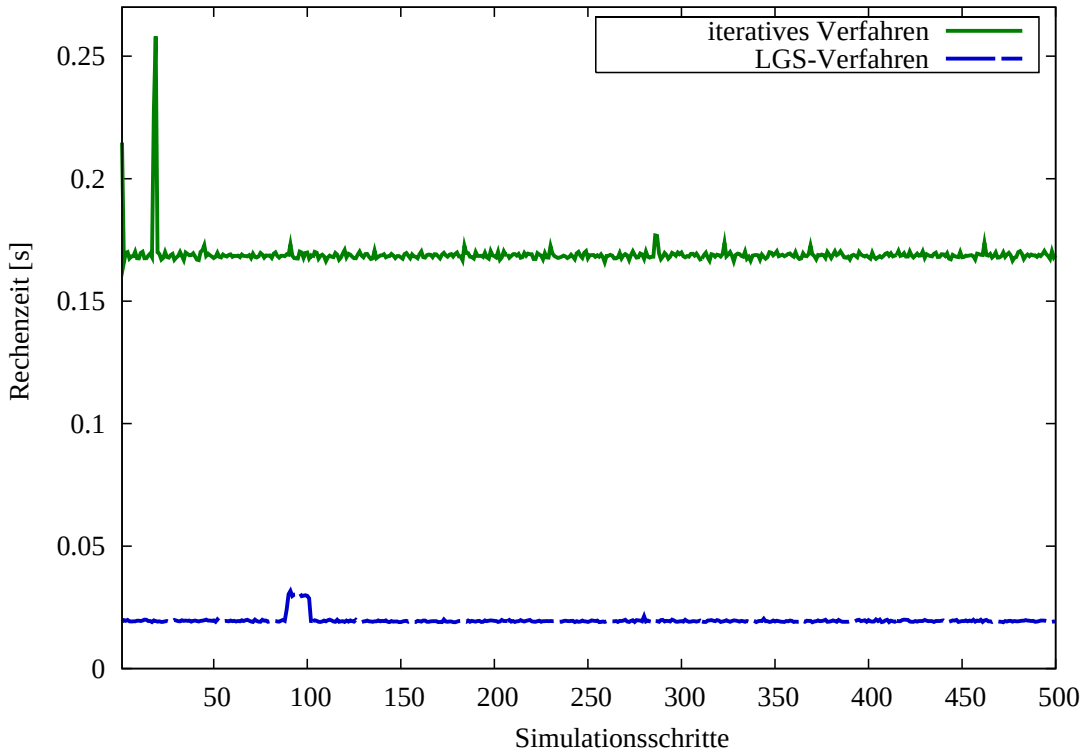


Abbildung 3.30: Vergleich der benötigten Rechenzeit pro Simulationsschritt. Der Baum mit 127 Gelenken wurde mit dem iterativen und dem LGS-Verfahren simuliert. Die Messungen zeigen, dass das LGS-Verfahren deutlich schneller ist.

korrektur sind in Tabelle 3.2 aufgelistet. Diese Werte zeigen, dass die Simulation

	iteratives Verfahren	LGS-Verfahren
Zeit pro Simulationsschritt	169,07 ms	19,66 ms
Iterationen (Gelenkkorr.)	541,65	2
Iterationen (Geschwindigkeitskorr.)	780,89	1

Tabelle 3.2: Durchschnittswerte

mit dem LGS-Verfahren mehr als 1,5-mal schneller war als Echtzeit. Außerdem war das LGS-Verfahren mehr als 8,5-mal schneller gegenüber dem iterativen Verfahren. Dies hängt damit zusammen, dass beim iterativen Verfahren durch die vielen Abhängigkeiten im Mehrkörpersystem und die hohe Genauigkeit, die gefordert war, sehr viele Iterationen bei der Gelenk- und bei der Geschwindigkeitskorrektur benötigt wurden. Dagegen brauchte das LGS-Verfahren im Schnitt nur zwei Iterationen für die Gelenkkorrektur und konnte die Impulse für die Geschwindigkeitskorrektur immer in einem Schritt exakt berechnen.

Es wurden weitere Simulationen mit Baum-Modellen durchgeführt. Dabei wurden vier Bäume mit verschiedenen Größen verwendet, um das Verhalten der beiden Verfahren bei Modellen unterschiedlicher Komplexität zu untersuchen. Außerdem wurde jedes Modell mit drei unterschiedlichen Werten für die Toleranzen ε_d und ε_v simuliert. Die durchschnittlichen Zeiten, die das iterative Verfahren für die Durchführung eines Simulationsschrittes benötigt hat, sind in Tabelle 3.3 aufgelistet. Tabelle 3.4 zeigt die durchschnittlichen Werte der Rechenzeiten des LGS-Verfahrens.

	$\varepsilon_d = 10^{-3}\,\mathrm{m}$, $\varepsilon_v = 10^{-3}\,\mathrm{m/s}$	$\varepsilon_d = 10^{-4}\,\mathrm{m}$, $\varepsilon_v = 10^{-4}\,\mathrm{m/s}$	$\varepsilon_d = 10^{-6}\,\mathrm{m}$, $\varepsilon_v = 10^{-6}\,\mathrm{m/s}$
Baum 31	2,73 ms	7,07 ms	15,87 ms
Baum 63	7,86 ms	21,87 ms	54,25 ms
Baum 127	20,35 ms	63,77 ms	169,07 ms
Baum 255	51,47 ms	176,96 ms	494,15 ms

Tabelle 3.3: Durchschnittliche Dauer eines Simulationsschrittes mit dem iterativen Verfahren

	$\varepsilon_d = 10^{-3}\,\mathrm{m}$,	$\varepsilon_d = 10^{-4}\,\mathrm{m}$,	$\varepsilon_d = 10^{-6}\,\mathrm{m}$,
Baum 31	3,51 ms	3,99 ms	4,53 ms
Baum 63	6,95 ms	7,09 ms	7,89 ms
Baum 127	18,38 ms	19,05 ms	19,66 ms
Baum 255	57,72 ms	57,61 ms	59,48 ms

Tabelle 3.4: Durchschnittliche Dauer eines Simulationsschrittes mit dem LGS-Verfahren

Die Berechnung eines einzelnen Impulses funktioniert sehr schnell. Außerdem konvergiert das iterative Verfahren bei größeren Toleranzwerten nach relativ wenigen Durchläufen. Aus diesen Gründen bringt die Verwendung eines linearen Gleichungssystems bei großen Toleranzen keinen Geschwindigkeitsvorteil. Bei Toleranzwerten von 10^{-3} war das iterative Verfahren bei zwei Modellen sogar schneller als das LGS-Verfahren. Werden noch größere Toleranzwerte verwendet, wird das iterative Verfahren deutlich schneller als das LGS-Verfahren. Wenn allerdings eine höhere Genauigkeit erreicht werden soll und deshalb kleine Toleranzen gewählt werden, dann ist das iterative Verfahren um ein Vielfaches langsamer als das LGS-Verfahren. Anhand der Messwerte erkennt man, dass die Dauer eines Simulationsschrittes beim iterativen Verfahren stark von der geforderten Genauigkeit abhängt. Das LGS-Verfahren ist zwar bei großen Toleranzwerten langsamer als das iterati-

ve Verfahren, dafür erhöht sich der Zeitaufwand nur sehr geringfügig, wenn eine höhere Genauigkeit gefordert wird.

Um dies zu erklären, wurde eine weitere Simulation mit dem Baum-Modell mit 127 Körpern und Gelenken durchgeführt. Dabei wurden die Toleranzwerte $\varepsilon_d = 10^{-6}\,\mathrm{m}$ und $\varepsilon_v = 10^{-6}\,\frac{\mathrm{m}}{\mathrm{s}}$ verwendet. Bei dieser Simulation wurden die Rechenzeiten für die einzelnen Teile eines Simulationsschrittes gemessen. Die durchschnittlichen Werte dieser Messung sind in Tabelle 3.5 aufgelistet. Die Messwerte zeigen, dass das

	Durchschnittliche Rechenzeit
Aufstellen der Matrix und Faktorisierung	15,81 ms
Iterationsschritt bei der Gelenk- bzw. Geschwindigkeitskorrektur	1,01 ms
Zeitschritt von t_0 nach $t_0 + h$ für alle Körper	0,17 ms

Tabelle 3.5: Rechenzeiten in einem Schritt des LGS-Verfahrens für das Baum-Modell mit 127 Körpern und Gelenken

Aufstellen und Faktorisieren der Matrix des linearen Gleichungssystems die meiste Zeit in Anspruch nimmt. Die Berechnung der Positionen und Geschwindigkeiten für den nächsten Simulationsschritt hat dagegen den geringsten Zeitaufwand. Beide Teile müssen nur einmal pro Simulationsschritt durchgeführt werden. Dagegen ist die Anzahl der benötigten Iterationsschritte abhängig von den verwendeten Toleranzwerten. Da selbst bei sehr kleinen Toleranzen von 10^{-6} nur insgesamt drei Iterationsschritte für die Gelenk- und die Geschwindigkeitskorrektur benötigt wurden und ein Iterationsschritt sehr schnell durchgeführt werden kann, ist bei dem LGS-Verfahren die Zeit, die für die Faktorisierung der Matrix benötigt wird, ausschlaggebend.

Beide Verfahren ermöglichen eine schnelle Simulation von Modellen mit einer komplexen Gelenkstruktur. Das iterative Verfahren hat Vorteile gegenüber dem LGS-Verfahren, wenn große Toleranzen verwendet werden. Muss eine bestimmte Geschwindigkeitsanforderung bei der Simulation erfüllt werden, kann dies durch einen vorzeitigen Abbruch der Iterationsschleife geschehen. Da beim LGS-Verfahren der Hauptteil der Rechenzeit beim Faktorisieren der Matrix verbraucht wird und nur sehr wenige Iterationen für die Simulation benötigt werden, ist diese Vorgehensweise nur mit dem iterativen Verfahren sinnvoll. Bei einem Baum-Modell mit 255 Körpern werden mehr als 50 ms für das Faktorisieren der Matrix benötigt. Daher ist eine Simulation dieses Modells mit dem LGS-Verfahren auch bei vorzeitigen Abbruch der Iterationsschleife nicht in Echtzeit möglich. Dagegen hat eine Messung mit dem iterativen Verfahren ergeben, dass ein Simulationsschritt bei diesem Modell durchschnittlich 6,30 ms benötigt, wenn die Gelenk- und die Geschwindigkeitskorrektur jeweils nach zehn Iterationen abgebrochen wird. Auf diese Weise

konnte das Modell mehr als fünfmal schneller als Echtzeit stabil simuliert werden. Wenn allerdings bei der Simulation eines komplexen Modells die Genauigkeit eine entscheidende Rolle spielt, ist die Verwendung des LGS-Verfahrens von Vorteil. Dieses Verfahren arbeitet relativ schnell und wird auch bei höheren Anforderungen an die Genauigkeit nicht signifikant langsamer.

Modelle mit Schleifen Die letzten Messungen wurden mit einem Modell durchgeführt, das drei geschlossene kinematische Ketten enthält (siehe Abbildung 3.31). Jede Schleife besteht aus vier Körpern, die durch Drehgelenke miteinander verbun-

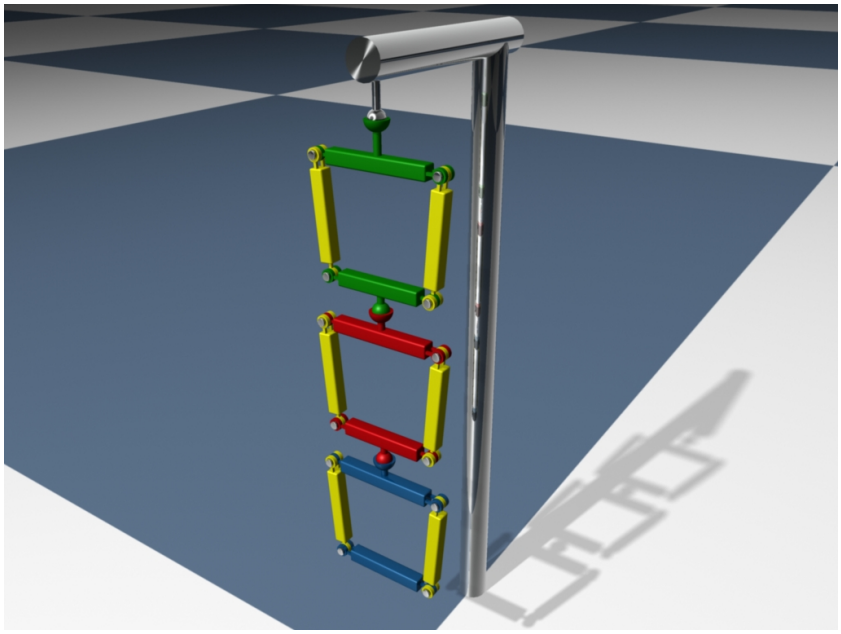


Abbildung 3.31: Modell mit drei Schleifen

den sind. In jeder Schleife ist einer der Körper länger als die anderen drei. Wenn alle die gleiche Länge haben, können die Körper einer Schleife auf einer Geraden liegen. In diesem Fall ist das lineare Gleichungssystem des LGS-Verfahrens nicht eindeutig lösbar.

Das Modell wurde mit dem iterativen Verfahren und mit dem LGS-Verfahren simuliert. Bei der Simulation mit dem LGS-Verfahren wurden zunächst alle Schleifen an jeweils einem Gelenk aufgetrennt. Die Zwangsbedingungen der Gelenke, an denen aufgetrennt wurde, wurden separat vom Rest aufgelöst. In einer weiteren Simulation mit dem LGS-Verfahren wurde ein angepasstes Modell verwendet. Bei diesem Modell wurden die Gelenke so gewählt, dass sie ohne Redundanz die gleichen Freiheitsgrade entfernen. Durch diese Modifikation der Gelenke war eine Simulation ohne Sonderbehandlung möglich (vgl. Abschnitt 3.3.2.3). Alle Simulationen wurden mit einer Zeitschrittweite von $h = \frac{1}{30}\,\mathrm{s}$ durchgeführt. Mit jedem Verfah-

ren wurden drei Simulationen mit verschiedenen Toleranzwerten durchgeführt. Die Ergebnisse sind in Tabelle 3.6 aufgelistet.

	$\varepsilon_d = 10^{-3}\,\mathrm{m}$, $\varepsilon_v = 10^{-3}\,\frac{\mathrm{m}}{\mathrm{s}}$	$\varepsilon_d = 10^{-4}\,\mathrm{m}$, $\varepsilon_v = 10^{-4}\,\frac{\mathrm{m}}{\mathrm{s}}$	$\varepsilon_d = 10^{-6}\,\mathrm{m}$, $\varepsilon_v = 10^{-6}\,\frac{\mathrm{m}}{\mathrm{s}}$
iterativ	32,26 ms	88,86 ms	241,24 ms
LGS – Auftrennen der Schleifen	41,19 ms	64,94 ms	114,70 ms
LGS – Modifikation der Gelenke	3,51 ms	3,89 ms	4,85 ms

Tabelle 3.6: Messung der durchschnittlichen Dauer eines Simulationsschrittes. Das iterative Verfahren kann geschlossene kinematische Ketten ohne Sonderbehandlung simulieren, während beim LGS-Verfahren entweder die Schleifen aufgetrennt oder die Gelenke angepasst werden müssen.

Durch die 12 Drehgelenke ergeben sich 60 eindimensionale Zwangsbedingungen. Aufgrund der Abhängigkeiten in den Schleifen ist eine Auflösung dieser Bedingungen schwierig. Das iterative Verfahren kann das Modell nur mit der geringsten Genauigkeit in Echtzeit simulieren. Der zusätzliche Aufwand durch das Aufstellen und Lösen eines linearen Gleichungssystems beim LGS-Verfahren mit Sonderbehandlung lohnt sich erst ab Toleranzen von 10^{-4}. Bei der höchsten Genauigkeit ist dieses Verfahren doppelt so schnell wie das iterative Verfahren. Die Simulation mit einem angepassten Modell ist mit Abstand am schnellsten. Selbst bei der höchsten Genauigkeitsstufe war die Simulation fast sieben Mal schneller als Echtzeit.

Das iterative Verfahren hat den Vorteil, dass es ohne Sonderbehandlung auskommt. Wenn hohe Genauigkeiten gefordert werden, ist es allerdings das langsamste Verfahren. Das LGS-Verfahren mit Sonderbehandlung ist bei hohen Genauigkeiten zwar schneller, war aber bei den Messungen zu langsam für eine Simulation in Echtzeit. Die Simulation eines angepassten Modells mit dem LGS-Verfahren ist die schnellste Variante. Der einzige Nachteil dabei ist, dass das angepasste Modell bisher noch nicht automatisch erstellt werden kann.

3.4 Verfahren höherer Ordnung

In diesem Abschnitt werden Verfahren höherer Ordnung hergeleitet. Diese Verfahren sind langsamer als die bereits vorgestellten Verfahren. Allerdings kann mit ihnen ein höherer Grad an Genauigkeit erreicht werden [SBP05a].

3.4.1 Verfahren zweiter Ordnung

Betrachtet man einen Massenpunkt $\mathbf{x}(t)$ mit der Masse m, der durch eine kontinuierliche Kraft $\mathbf{F}(t)$ beschleunigt wird, dann ist seine Bewegung durch die Gleichung

$$\ddot{\mathbf{x}}(t) = \frac{1}{m}\mathbf{F}(t) \tag{3.24}$$

bestimmt. Bei dieser Gleichung handelt es sich um Newtons zweites Gesetz. Angenommen die Kraft $\mathbf{F}(t)$ sei bekannt und durch die Potenzreihe

$$\mathbf{F}(t) = \sum_{i=0}^{\infty} \mathbf{a}_i \, t^i$$

bestimmt. Wenn die Position $\mathbf{x}$ und die Geschwindigkeit $\dot{\mathbf{x}}$ des Massenpunktes zu einem Zeitpunkt t_0 bekannt sind, kann durch zweimaliges Integrieren der Gleichung 3.24 die genaue Position des Punktes zum Zeitpunkt t_0+h berechnet werden:

$$\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\, h + \frac{1}{m} \sum_{i=0}^{\infty} \frac{1}{(i+2)(i+1)} \, \mathbf{a}_i \, h^{i+2}.$$

Durch Integration der kontinuierlichen Kraft über dem gleichen Zeitintervall erhält man den Impuls:

$$\mathbf{p} = \int_{t_0}^{t_0+h} \mathbf{F}(t)\, dt = \sum_{i=0}^{\infty} \frac{1}{i+1} \, \mathbf{a}_i \, h^{i+1}.$$

Die Geschwindigkeitsänderung, die durch die Kraft $\mathbf{F}(t)$ während des Zeitintervalls $[t_0, t_0 + h]$ verursacht wird, wird von dem Impuls $\mathbf{p}$ sofort bewirkt. Durch die Anwendung des Impulses in der Mitte des Zeitintervalls kann die Wirkung der kontinuierlichen Kraft approximiert werden. In diesem Fall berechnet sich die Position des Punktes zum Zeitpunkt $t_0 + h$ wie folgt:

$$\begin{aligned} \mathbf{x}_1(t_0 + h) &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\, h + \frac{h}{2\,m}\, \mathbf{p} \\ &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\, h + \frac{1}{m} \sum_{i=0}^{\infty} \frac{1}{2\,i+2} \, \mathbf{a}_i \, h^{i+2}. \end{aligned}$$

Wenn man die genaue Position des Punktes $\mathbf{x}(t_0+h)$ mit der angenäherten Position $\mathbf{x}_1(t_0 + h)$ vergleicht, dann stellt man fest, dass sie bis zu dem Term von h^2 gleich sind. Für die Punkte gilt demnach:

$$\mathbf{x}(t_0 + h) = \mathbf{x}_1(t_0 + h) + O(h^3)$$

und der Fehler, der durch die Approximation verursacht wird, ist von der Ordnung $O(h^3)$.

3.4.2 Verfahren vierter Ordnung

Die Position des Punktes zum Zeitpunkt $t_0 + h$ lässt sich mit Hilfe von Impulsen genauer approximieren, wenn die Position in zwei Zeitschritten berechnet wird. In jedem der Zeitschritte wird ein Impuls in der Mitte des jeweiligen Zeitintervalls auf den Punkt angewendet. Der erste Impuls $\mathbf{p}_1$ wird zum Zeitpunkt $t_0 + h/4$ und der zweite Impuls $\mathbf{p}_2$ zum Zeitpunkt $t_0 + 3h/4$ angewendet. Die beiden Impulse werden durch Integration der Kraft $\mathbf{F}(t)$ über dem jeweiligen Zeitintervall berechnet:

$$\begin{aligned}\mathbf{p}_1 &= \int_{t_0}^{t_0+h/2} \mathbf{F}(t)\,dt = \sum_{i=0}^{\infty} \frac{1}{2^{i+1}\,(i+1)}\,\mathbf{a}_i\,h^{i+1} \\ \mathbf{p}_2 &= \int_{t_0+h/2}^{t_0+h} \mathbf{F}(t)\,dt = \sum_{i=0}^{\infty} \frac{2^{i+1}-1}{2^{i+1}\,(i+1)}\,\mathbf{a}_i\,h^{i+1}.\end{aligned}$$

Die Position des Punktes zum Zeitpunkt $t_0 + h$ ist nach dem Anwenden der Impulse dann

$$\begin{aligned}\mathbf{x}_2(t_0+h) &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\,h + \frac{h}{4\,m}\,(3\,\mathbf{p}_1 + \mathbf{p}_2) \\ &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\,h + \frac{1}{m}\sum_{i=0}^{\infty} \frac{2^i+1}{2^{i+2}\,(i+1)}\,\mathbf{a}_i\,h^{i+2}.\end{aligned}$$

Diese Approximation der Position liegt näher am genauen Wert $\mathbf{x}(t_0 + h)$ als $\mathbf{x}_1(t_0 + h)$. Allerdings ist der Fehlerterm immer noch von der Ordnung $O(h^3)$.

Die Positionen, die mit Hilfe der Impulse bestimmt wurden, weichen jeweils erst ab dem Term von h^3 von der exakten Lösung ab. Durch eine Linearkombination dieser beiden Ergebnisse lässt sich die Fehlerordnung verbessern. Beide Gleichungen für die Positionen werden mit einem Faktor c_i gewichtet. Für die Summe dieser Faktoren muss

$$\sum_{i=1}^{n} c_i = 1 \tag{3.25}$$

gelten, wobei in diesem Fall $n = 2$ ist, da zwei Positionsgleichungen mit Hilfe von Impulsen aufgestellt wurden. Durch diese Bedingung wird gewährleistet, dass die Kombination der Gleichungen weiterhin bis zu dem Term von h^2 mit der exakten Lösung übereinstimmt. Damit durch die Kombination der Gleichungen eine höhere Ordnung erreicht wird, müssen die Faktoren c_i entsprechend gewählt werden. Deswegen wird ein lineares Gleichungssystem für die Faktoren aufgestellt:

$$\begin{pmatrix} 1 & 1 \\ \frac{1}{4} & \frac{3}{16} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{6} \end{pmatrix}.$$

Die erste Zeile dieses Gleichungssystems enthält die Bedingung 3.25. In der zweiten Zeile steht auf der rechten Seite des Gleichungssystems der Faktor des Terms

$\mathbf{a}_1\, h^3$ der exakten Lösung, denn dieser Term soll zum Schluss mit dem entsprechenden Term der kombinierten Gleichung übereinstimmen. In der Matrix stehen die Faktoren dieses Terms aus den Gleichungen für $\mathbf{x}_1(t_0+h)$ und $\mathbf{x}_2(t_0+h)$, die kombiniert werden sollen. Löst man das Gleichungssystem nach den Faktoren auf, dann erhält man als Lösung $c_1 = -\frac{1}{3}$ und $c_2 = \frac{4}{3}$. Wenn diese Faktoren bei der Kombination der Gleichungen verwendet werden, ist bei der resultierenden Gleichung ein Fehler von der Ordnung $O(h^4)$ zu erwarten. Die kombinierte Gleichung hat dann die folgende Form:

$$\begin{aligned}\mathbf{x}_k(t_0+h) &= -\frac{1}{3}\,\mathbf{x}_1(t_0+h) + \frac{4}{3}\,\mathbf{x}_2(t_0+h) \\ &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\,h + \frac{1}{m}\sum_{i=0}^{\infty}\frac{2^{i-1}+1}{(3i+3)\,2^i}\,\mathbf{a}_i\,h^{i+2}.\end{aligned}$$

Vergleicht man die Terme von $\mathbf{x}_k(t_0+h)$ mit der exakten Lösung, so stellt man eine Übereinstimmung bis zu dem Term von h^4 fest. Das bedeutet, für die Position des Punktes gilt

$$\mathbf{x}(t_0+h) = \mathbf{x}_k(t_0+h) + O(h^5)$$

und der Fehler ist damit sogar von der Ordnung $O(h^5)$ und nicht wie erwartet von der Ordnung $O(h^4)$.

3.4.3 Verfahren sechster Ordnung

Die Genauigkeit des Verfahrens kann weiter verbessert werden, wenn man drei Zeitschritte durchführt und die resultierende Position mit den bisherigen Ergebnissen kombiniert. Die Position des Punktes nach den drei Zeitschritten wird bestimmt, indem zunächst die Impulse für die drei Teilintervalle ermittelt werden:

$$\begin{aligned}\mathbf{p}_1 &= \int_{t_0}^{t_0+h/3}\mathbf{F}(t)\,dt = \sum_{i=0}^{\infty}\frac{1}{3^{i+1}\,(i+1)}\,\mathbf{a}_i\,h^{i+1} \\ \mathbf{p}_2 &= \int_{t_0+h/3}^{t_0+2h/3}\mathbf{F}(t)\,dt = \sum_{i=0}^{\infty}\frac{2^{i+1}-1}{3^{i+1}\,(i+1)}\,\mathbf{a}_i\,h^{i+1} \\ \mathbf{p}_3 &= \int_{t_0+2h/3}^{t_0+h}\mathbf{F}(t)\,dt = \sum_{i=0}^{\infty}\frac{3^{i+1}-2^{i+1}}{3^{i+1}\,(i+1)}\,\mathbf{a}_i\,h^{i+1}.\end{aligned}$$

Wenn diese Impulse in der Mitte des jeweiligen Zeitintervalls angewendet werden, dann ergibt sich zum Zeitpunkt t_0+h die folgende Position für den Massenpunkt:

$$\begin{aligned}\mathbf{x}_3(t_0+h) &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\,h + \frac{h}{6\,m}\,(5\,\mathbf{p}_1 + 3\,\mathbf{p}_2 + \mathbf{p}_3) \\ &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\,h + \frac{1}{m}\sum_{i=0}^{\infty}\frac{2^{i+2}+3^{i+1}+2}{2\cdot 3^{i+2}\,(i+1)}\,\mathbf{a}_i\,h^{i+2}.\end{aligned}$$

Der Fehlerterm dieser Position ist von der Ordnung $O(h^3)$.

Durch Kombination der Positionen $\mathbf{x}_1$, $\mathbf{x}_2$ und $\mathbf{x}_3$ zum Zeitpunkt $t_0 + h$ mit geeigneten Faktoren c_i kann die Genauigkeit des Ergebnisses verbessert werden. Die Faktoren c_i müssen die Bedingung 3.25 erfüllen, damit die Kombination bis zu den Termen von h^2 mit der exakten Lösung übereinstimmt. Dadurch ergibt sich die erste Zeile des Gleichungssystems. Die Faktoren vor den Termen $\mathbf{a}_1 h^3$ in den Gleichungen für die Position des Massenpunktes ergeben die zweite Zeile. Bei dem Verfahren vierter Ordnung konnte gezeigt werden, dass durch diese Bedingung für die Faktoren c_i eine Übereinstimmung mit der exakten Lösung bis zu den Termen von h^4 erreicht werden kann. Deshalb wird die letzte Zeile des Gleichungssystems aus den Faktoren der Terme $\mathbf{a}_3 h^5$ zusammengesetzt. Das lineare Gleichungssystem für die Faktoren c_i hat dann die folgende Form:

$$\begin{pmatrix} 1 & 1 & 1 \\ \frac{1}{4} & \frac{3}{16} & \frac{19}{108} \\ \frac{1}{8} & \frac{9}{128} & \frac{115}{1944} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{6} \\ \frac{1}{20} \end{pmatrix}.$$

Wenn dieses Gleichungssystem gelöst wird, ergeben sich die Werte $c_1 = \frac{1}{24}$, $c_2 = -\frac{16}{15}$ und $c_3 = \frac{81}{40}$ für die gesuchten Faktoren. Kombiniert man die Positionen $\mathbf{x}_1$, $\mathbf{x}_2$ und $\mathbf{x}_3$ mit diesen Faktoren, dann ergibt sich die Position

$$\begin{aligned} \mathbf{x}_k(t_0 + h) &= \frac{1}{24}\,\mathbf{x}_1(t_0 + h) - \frac{16}{15}\,\mathbf{x}_2(t_0 + h) + \frac{81}{40}\,\mathbf{x}_3(t_0 + h) \\ &= \mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\,h + \frac{1}{m}\sum_{i=0}^{\infty} \frac{11 \cdot 2^i\,3^i - 2^5\,3^i + 3^3\,2^i + 3^3\,2^{2i+1}}{2^{i+3}\,3^{i+1}\,5\,(i+1)}\,\mathbf{a}_i\,h^{i+2} \end{aligned}$$

für den Massenpunkt. Diese Position weicht erst ab dem Term von h^7 von der exakten Lösung ab:

$$\mathbf{x}(t_0 + h) = \mathbf{x}_k(t_0 + h) + O(h^7).$$

Der Fehler ist demnach von der Ordnung $O(h^7)$.

3.4.4 Praktische Anwendung

Zusätzlich zu den hier vorgestellten Verfahren werden in [SBP05a] noch die Verfahren achter und zehnter Ordnung hergeleitet. Die Verfahren, die bisher beschrieben wurden, wurden für die Position eines Massenpunktes hergeleitet. Bei der dynamischen Simulation werden allerdings Starrkörper verwendet. Aus diesem Grund mussten die Verfahren höherer Ordnung erweitert werden, so dass mit ihnen auch die Rotation eines Starrkörpers simuliert werden konnte. Diese Erweiterung für Starrkörper wird ebenfalls in [SBP05a] vorgestellt.

Bei der Herleitung der Verfahren höherer Ordnung wurde angenommen, dass die kontinuierliche Kraft, die auf einen Körper wirkt, bekannt ist und als Potenzreihe

dargestellt werden kann. Daher konnte ein Impuls durch Integrieren der Kraft bestimmt werden. Die Impulse können im Allgemeinen nicht auf diese Art und Weise berechnet werden, da die kontinuierlichen Kräfte im Normalfall nicht im Voraus bekannt sind. Bei den impulsbasierten Verfahren werden die Impulse mit Hilfe einer Vorschau des Gelenkzustands ermittelt. Die so bestimmten Impulse simulieren die unbekannten kontinuierlichen Kräfte. Aus diesem Grund wurde in [SBP05a] versucht, diese Impulse in den Verfahren höherer Ordnung zu verwenden. Ein Simulationsschritt der impulsbasierten Methode musste wie folgt modifiziert werden, damit diese Methode zur Bestimmung der Impulse mit den Verfahren höherer Ordnung kombiniert werden konnte. Zunächst müssen alle Körper für einen halben Zeitschritt als freie Körper (siehe Abschnitt 3.1) behandelt werden. Dann werden mit dem impulsbasierten Verfahren die Impulse bestimmt und angewendet, die dafür sorgen, dass alle Zwangsbedingungen im System nach dem Zeitschritt erfüllt werden. Nach dieser Gelenkkorrektur wird die zweite Hälfte des Zeitschrittes durchgeführt. Abschließend muss eine Geschwindigkeitskorrektur für die Gelenke durchgeführt werden, so dass das System einen Zustand annimmt, in dem alle Zwangsbedingungen erfüllt sind.

Alle Verfahren höherer Ordnung wurden auf diese Weise mit dem impulsbasierten Verfahren kombiniert. Für diese Verfahren wurden in [SBP05a] und [SB05] verschiedene Genauigkeitsuntersuchungen durchgeführt. Dabei wurden auch Vergleiche mit der Lagrange-Faktoren-Methode und der Methode der reduzierten Koordinaten vorgenommen. Für den Vergleich mit der Lagrange-Faktoren-Methode wurden verschiedene Stabilisierungsverfahren verwendet. Die Untersuchungen ergaben, dass die erwarteten Ordnungen der Verfahren höherer Ordnung in der Praxis nicht erreicht werden konnten. Stattdessen wurde beobachtet, dass alle Verfahren eine Ordnung von $O(h^3)$ hatten. Es konnten trotzdem in jeder Simulation genauere Ergebnisse erreicht werden, denn je höher die theoretisch erwartete Ordnung der Verfahren war, desto kleiner war die Konstante der Fehlerordnung.

3.5 Vergleich der impulsbasierten und der klassischen Verfahren

In diesem Abschnitt werden die beiden vorgestellten, impulsbasierten Verfahren mit den klassischen Ansätzen aus Abschnitt 2.1 verglichen. Der Vergleich wird anhand der Anforderungen durchgeführt, die in Kapitel 1 für ein Verfahren der dynamischen Simulation in Anwendungen der virtuellen Realität aufgestellt wurden.

Allgemeine Eigenschaften Die Zwangsbedingungen der Gelenke werden von den verschiedenen Verfahren unterschiedlich aufgelöst. Die beiden vorgestellten im-

pulsbasierten Verfahren berechnen mit Hilfe einer Vorschau für jedes Gelenk einen Impuls, der die Bedingung innerhalb einer vorgegebenen Toleranz erfüllt. Beide Verfahren können holonome Zwangsbedingungen und Geschwindigkeitsbedingungen behandeln. Das iterative Verfahren beherrscht sogar geschlossene kinematische Ketten ohne Sonderbehandlung. Die Lagrange-Faktoren-Methode verwendet interne Kräfte zur Simulation von Gelenken. Die internen Kräfte werden berechnet, indem die Bedingungen durch Ableitung nach der Zeit für die Beschleunigungen der Körper aufgestellt werden. Anschließend muss ein lineares Gleichungssystem gelöst werden, um die Stärke der Kräfte zu bestimmen. Da die Bedingungen nur für die Beschleunigungen erfüllt werden, können fehlerhafte Geschwindigkeiten oder Positionen nicht korrigiert werden. Aus diesem Grund benötigt die Methode zusätzlich ein Stabilisierungsverfahren. Mit der Lagrange-Faktoren-Methode können Systeme mit holonomen Zwangsbedingungen und Geschwindigkeitsbedingungen simuliert werden. Diese Methode ist den impulsbasierten Verfahren dieser Arbeit am ähnlichsten. Daher wurde sowohl das Standardverfahren als auch das Verfahren von Baraff (siehe Abschnitt 2.1.3) zu Vergleichszwecken implementiert. Außerdem wurde die Stabilisierungsmethode von Baumgarte für beide Verfahren umgesetzt.

Die Penalty-Methode verwendet ebenfalls Kräfte, um die Bedingungen zu erfüllen. Diese Kräfte werden erst dann angewendet, wenn eine Bedingung verletzt wurde. Die Kraft wirkt dann dieser Verletzung entgegen. Mit dieser Methode können holonome Zwangsbedingungen und Geschwindigkeitsbedingungen simuliert werden. Das Problem bei dieser Methode ist, eine geeignete Wahl der konstanten Parameter zu finden, die bei der Bestimmung der Kräfte verwendet werden. Für die Simulation mit reduzierten Koordinaten wird zunächst ein Satz von unabhängigen Koordinaten bestimmt, der den Bewegungszustand des Systems eindeutig definiert. Jeder Freiheitsgrad des Systems wird dabei durch eine Koordinate beschrieben. Die Bewegungsgleichungen werden dann bezüglich der reduzierten Koordinaten aufgestellt. Dadurch sind die Zwangsbedingungen implizit in den Bewegungsgleichungen enthalten. Das ist ein großer Vorteil dieser Methode, denn auf diese Weise werden die Zwangsbedingungen immer erfüllt. Ein Nachteil des Verfahrens ist, dass eine neue Parametrisierung des Systems bestimmt werden muss, wenn sich die Zwangsbedingungen während der Simulation ändern. Die Methode der reduzierten Koordinaten unterstützt ausschließlich holonome Zwangsbedingungen.

Geschwindigkeit Die Penalty-Methode ist aufgrund der einfachen Berechnung der Kräfte sehr effizient. Dies gilt allerdings nur, solange keine hohen Genauigkeiten gefordert werden. Bei genauen Simulationen müssen sehr große Kräfte angewendet und die Zeitschrittweite reduziert werden. Die Methode der reduzierten Koordinaten ist vor allem bei Modellen ohne geschlossene kinematische Ketten schnell. Bei solchen Modellen kann ein Simulationsschritt mit dem Algorithmus von Featherstone [Fea87] in linearer Zeit berechnet werden.

Da die restlichen Verfahren alle während dieser Arbeit implementiert wurden, konnten Geschwindigkeitsmessungen mit diesen Verfahren durchgeführt werden. Für den Vergleich der Verfahren wurde das Baum-Modell aus Abschnitt 3.3.3.1 mit einer Zeitschrittweite von $h = \frac{1}{30}\,\mathrm{s}$ simuliert. Die Zeiten der impulsbasierten Verfahren wurden für verschiedene Toleranzwerte gemessen. Die durchschnittlichen Zeiten aller Verfahren sind in Tabelle 3.7 aufgelistet. Der Anstieg der Rechenzeit

	Baum 31	Baum 63	Baum 127	Baum 255
iterativ ($10^{-3}\,\mathrm{m}$, $10^{-3}\,\mathrm{m/s}$)	2,73 ms	7,86 ms	20,35 ms	51,47 ms
iterativ ($10^{-4}\,\mathrm{m}$, $10^{-4}\,\mathrm{m/s}$)	7,07 ms	21,87 ms	63,77 ms	176,96 ms
iterativ ($10^{-6}\,\mathrm{m}$, $10^{-6}\,\mathrm{m/s}$)	15,87 ms	54,25 ms	169,07 ms	494,15 ms
LGS ($10^{-3}\,\mathrm{m}$)	3,51 ms	6,95 ms	18,38 ms	57,72 ms
LGS ($10^{-4}\,\mathrm{m}$)	3,99 ms	7,09 ms	19,05 ms	57,61 ms
LGS ($10^{-6}\,\mathrm{m}$)	4,53 ms	7,89 ms	19,66 ms	59,48 ms
LFM - Standard	212,12 ms	2,19 s	25,20 s	175,45 s
LFM - Baraff	7,72 ms	16,84 ms	35,29 ms	82,05 ms

Tabelle 3.7: Durchschnittliche Dauer eines Simulationsschrittes

in Abhängigkeit von der Anzahl der Gelenke ist in Abbildung 3.32 dargestellt. Das Standard-Verfahren eignet sich nur für kleine Modelle, da die benötigte Rechenzeit bei vielen Gelenken sehr schnell ansteigt. Dagegen kann das Verfahren von Baraff die beiden kleineren Bäume schneller als Echtzeit simulieren. Damit ist dieses Verfahren schneller als das rein iterative, impulsbasierte Verfahren mit den kleinsten Toleranzwerten. Das iterative Verfahren benötigt sehr viele Iterationen bei komplexen Modellen, wenn eine hohe Genauigkeit gefordert wird. Werden beide Toleranzen auf den Wert 10^{-3} erhöht, dann ist das iterative Verfahren deutlich schneller als die Methode von Baraff. Das LGS-Verfahren ist besser für genaue Simulationen von Modellen mit vielen Abhängigkeiten zwischen den Gelenken geeignet. Daher ist dieses Verfahren bei hoher Genauigkeit wesentlich schneller als das iterative Verfahren. Außerdem steigt die benötigte Rechenzeit des LGS-Verfahrens, wenn die Anzahl der Gelenke erhöht wird, annähernd linear an. Dies liegt daran, dass ein spezieller Gleichungssystemlöser für dünnbesetzte Matrizen verwendet wurde. Beim größten Baum war das LGS-Verfahren um den Faktor 1,4 schneller als die Methode von Baraff. Bei den impulsbasierten Verfahren gibt es im Gegensatz zur Lagrange-Faktoren-Methode keine kontinuierlichen Zwangskräfte in der Bewegungsgleichung. Wenn die externen Kräfte für einen Zeitschritt konstant sind, wovon ausgegangen wurde, dann treten in dieser Gleichung gar keine kontinuierlichen Kräfte auf. Die Geschwindigkeit des Schwerpunkts und seine Position können daher direkt mit den Gleichungen 3.2 und 3.3 berechnet werden. Nur die Rotation und die Winkelgeschwindigkeit eines Körpers müssen mit Hilfe von numerischer

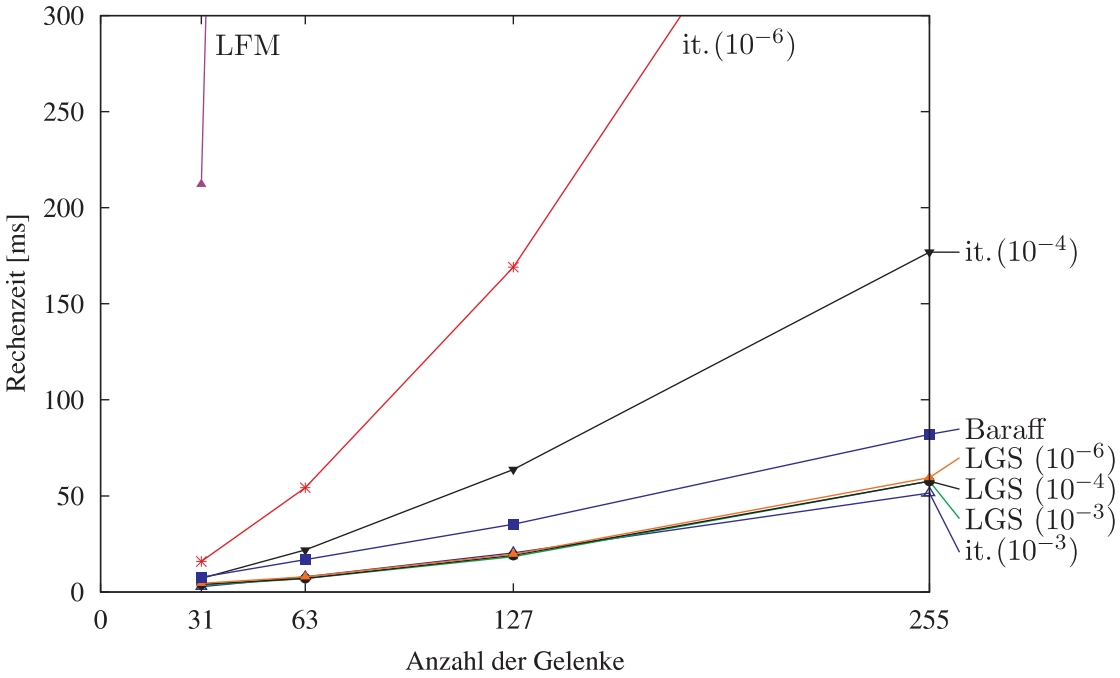


Abbildung 3.32: Vergleich der Rechenzeiten pro Simulationsschritt. Das Baummodell aus Abschnitt 3.3.3.1 wurde in verschiedenen Größen mit dem iterativen Verfahren (it.), dem LGS-Verfahren (LGS), dem Standard-Verfahren mit Lagrange-Faktoren (LFM) und der optimierten Variante von David Baraff simuliert.

Integration bestimmt werden. Die kontinuierlichen Kräfte in der Bewegungsgleichung der Lagrange-Faktoren-Methode müssen für die numerische Integration mit dem Runge-Kutta-Verfahren für mehrere Zeitpunkte berechnet werden. Das Lösen der Bewegungsgleichung kann daher mit den impulsbasierten Verfahren schneller durchgeführt werden als mit der Lagrange-Faktoren-Methode.

Genauigkeit Die Penalty-Methode erfüllt die Zwangsbedingungen eines Modells im Allgemeinen nur näherungsweise und ist daher für eine genaue Simulation nicht geeignet. Die Methode der reduzierten Koordinaten kann dagegen sehr genaue Ergebnisse liefern, da ihre Genauigkeit hauptsächlich vom verwendeten numerischen Integrationsverfahren abhängt. Bei den impulsbasierten Verfahren müssen nur die Parameter für die Rotation numerisch integriert werden. Die anderen Werte werden exakt berechnet, da keine kontinuierlichen Kräfte auftreten. Zusätzlich hängt die Genauigkeit dieser Verfahren von den verwendeten Toleranzwerten bei der Gelenk- bzw. Geschwindigkeitskorrektur ab. Mit den Toleranzen wird festgelegt, wie genau die Zwangsbedingungen erfüllt werden. Die Genauigkeit der Simulation kann mit den vorgestellten Verfahren höherer Ordnung verbessert werden. Bei der Lagrange-Faktoren-Methode ist die Genauigkeit abhängig von dem numerischen

Integrationsverfahren und der verwendeten Stabilisierungsmethode. Im Fall der Baumgarte-Stabilisierung hängt es von den verwendeten Parametern ab, wie genau eine Bedingung erfüllt wird bzw. wie groß die Abweichung ist. Im Gegensatz zu den impulsbasierten Verfahren kann diese Abweichung allerdings nicht durch Toleranzwerte beschränkt werden. Da die Genauigkeit der verschiedenen Verfahren von unterschiedlichen Komponenten abhängt, ist ein direkter Vergleich nur begrenzt möglich. Vergleiche der Verfahren wurden in [SBP05a] und [SB05] durchgeführt.

Unterstützte Gelenktypen Alle Verfahren unterstützen holonome Zwangsbedingungen. Damit können alle Arten von mechanischen Gelenken simuliert werden. Abgesehen von der Methode der reduzierten Koordinaten beherrschen auch alle Verfahren Geschwindigkeitsbedingungen. Bei der Penalty- und der Lagrange-Faktoren-Methode werden die Ableitungen der Zwangsbedingungen nach der Zeit benötigt. Diese müssen berechnet werden, um die Zwangskräfte zu bestimmen. Die Methode der reduzierten Koordinaten integriert die Zwangsbedingungen durch die Parametrisierung in die Bewegungsgleichung. Mit einer Bedingung kann direkt die Größe der Abweichung ermittelt werden, wenn sie nicht erfüllt ist. Daher können die impulsbasierten Verfahren Zwangsbedingungen ohne eine weitere Umformung verwenden. Das iterative Verfahren kann sogar Kollisionen und Kontakte mit Reibung ohne Sonderbehandlung simulieren (siehe Kapitel 5).

Ergebnisse zu jeder Zeit Da die impulsbasierten Verfahren iterativ arbeiten, ist es möglich die Iterationsschleife vorzeitig abzubrechen, um sofort ein Ergebnis zu erhalten. Es handelt sich dabei um sogenannte Anytime-Algorithmen. Diese Eigenschaft ist besonders in Anwendungen nützlich, in denen eine bestimmte Bildwiederholrate eingehalten werden muss. Mit den anderen Verfahren führt ein vorzeitiger Abbruch zu keinem Ergebnis. Beim LGS-Verfahren funktioniert dies auch nur begrenzt, denn die meiste Rechenzeit wird für das Faktorisieren der Matrix gebraucht (vgl. Abschnitt 3.3.3.1). Außerdem werden bei diesem Verfahren nur wenige Iterationen benötigt, so dass ein vorzeitiger Abbruch der Schleife nicht viel Rechenzeit einspart.

Stabilität Die Methode der reduzierten Koordinaten ist sehr stabil, denn jeder Systemzustand in reduzierten Koordinaten erfüllt automatisch alle Zwangsbedingungen. Die Lagrange-Faktoren-Methode hat dagegen das Problem, dass sie einen numerischen Drift nicht korrigieren kann. Bei dieser Methode wird ein zusätzliches Stabilisierungsverfahren benötigt. Die beiden impulsbasierten Verfahren steuern zulässige Gelenkzustände direkt an. Damit können selbst völlig zerfallene Modelle wieder zusammengesetzt werden (vgl. Abschnitt 3.3.3). Diese Eigenschaft ist insbesondere dann wichtig, wenn die Iterationsschleife vorzeitig abgebrochen werden soll, um ein vorläufiges Ergebnis zu erhalten. Der impulsbasierte Ansatz

kann außerdem zur Stabilisierung der Lagrange-Faktoren-Methode verwendet werden [SBP05a]. Bei der Penalty-Methode treten sehr große Kräfte auf, wenn eine hohe Genauigkeit gefordert wird. Eine stabile Simulation ist in diesem Fall nur mit einer kleinen Zeitschrittweite möglich.

Implementierungsaufwand und Erweiterbarkeit Das iterative Verfahren ist sehr einfach zu implementieren, da jedes Gelenk für sich betrachtet wird und nur sehr einfache Gleichungen gelöst werden müssen. Jede Zwangsbedingung kann direkt verwendet werden, daher ist das Verfahren leicht um neue Gelenke erweiterbar. Dies gilt auch für das LGS-Verfahren. Allerdings muss bei diesem Verfahren zusätzlich ein lineares Gleichungssystem aufgestellt und gelöst werden. Die Bestimmung der Positionen und Geschwindigkeiten für den nächsten Zeitschritt ist bei den impulsbasierten Verfahren einfach, da keine kontinuierlichen Kräfte in den Differentialgleichungen auftreten. Die Penalty-Methode benötigt die Ableitungen der Bedingungen nach der Zeit und berechnet dann mit einer einfachen Gleichung die Zwangskräfte. Die Implementierung dieser Methode ist ebenfalls einfach. Die Lagrange-Faktoren-Methode benötigt die Ableitungen der Zwangsbedingungen. Mit diesen Ableitungen muss ein lineares Gleichungssystem aufgestellt und gelöst werden. Zusätzlich benötigt diese Methode noch ein Stabilisierungsverfahren. Aus diesem Grund ist die Lagrange-Faktoren-Methode nicht ganz so einfach zu implementieren wie die impulsbasierten Verfahren. Das Problem bei der Methode der reduzierten Koordinaten ist es, eine geeignete Parametrisierung zu finden. Modelle ohne geschlossene kinematische Ketten können mit dem Verfahren von Featherstone in linearer Zeit simuliert werden. Dieses Verfahren ist allerdings schwer zu implementieren [Mir96b].

Fazit Die Verwendung der Penalty-Methode ist nur dann sinnvoll, wenn keine genauen Simulationsergebnisse benötigt werden. Daher ist diese Methode für Anwendungen der virtuellen Realität nur begrenzt von Interesse.

Dagegen können mit der Methode der reduzierten Koordinaten sehr genaue Ergebnisse erzielt werden. Außerdem ist diese Methode sehr effizient und stabil. Allerdings kann die Bestimmung einer Parametrisierung bei einigen Modellen sehr schwer sein [Bar96] und es können nur holonome Bedingungen und keine Schleifen simuliert werden. Insbesondere in Anwendungen der Robotik ist diese Methode von großer Bedeutung, da bei solchen Anwendungen die Modelle im Allgemeinen im Voraus bekannt und parametrisierbar sind. Die verwendeten Modelle beinhalten meistens nur holonome Bedingungen und keine Schleifen und können daher mit der Methode der reduzierten Koordinaten effizient und genau simuliert werden.

Die Lagrange-Faktoren-Methode kann ebenfalls sehr effizient umgesetzt werden [Bar96]. Außerdem kann diese Methode Geschwindigkeitsbedingungen direkt simulieren. Kollisionen und Kontakte mit Reibung sowie Schleifen im Modell müssen

gesondert behandelt werden. Da die Zwangsbedingungen nur für die Beschleunigungen der Körper aufgelöst werden, wird ein zusätzliches Stabilisierungsverfahren benötigt. Eine Umsetzung der Lagrange-Faktoren-Methode kann modular aufgebaut werden und ist dadurch leicht erweiterbar.

Dies gilt ebenfalls für die impulsbasierten Verfahren. Das iterative Verfahren hat zusätzlich den Vorteil, dass es Schleifen, Kollisionen und Kontakte mit Reibung ohne Sonderbehandlung simulieren kann, da eine Kollision als Gelenk mit Geschwindigkeitsbedingung und ein Kontakt als Gelenk mit Positionsbedingung formuliert werden kann (vgl. Kapitel 5). Allerdings können komplexe Modelle mit diesem Verfahren nur dann effizient simuliert werden, wenn die Anforderungen an die Genauigkeit nicht zu hoch sind. Das LGS-Verfahren ermöglicht eine genaue und effiziente Simulation von komplexen Modellen. Der Nachteil dieses Verfahrens ist, dass es eine Sonderbehandlung für Schleifen, Kollisionen und Kontakte benötigt.

Tabelle 3.8 fasst den Vergleich der Simulationsverfahren zusammen.

	Penalty-Methode	reduzierte Koordinaten	LFM	impulsbasierte Simulation
Geschwindigkeit	+	+	+	+
Genauigkeit	−	+	+	+
Gelenktypen	+	−	+	+
vorläufige Ergebnisse	−	−	−	+
Stabilität	−	+	−	+
Implementierung	+	−	+	+

Tabelle 3.8: Vergleich der Penalty-Methode, der Methode der reduzierten Koordinaten, der Lagrange-Faktoren-Methode (LFM) und der impulsbasierten Simulation

Kapitel 4

Kollisionserkennung

Die dynamische Simulation von Starrkörpern erfordert nicht nur die Simulation von Gelenken, sondern auch die Behandlung von Kollisionen und Kontakten zwischen den Körpern. Damit wird verhindert, dass sich zwei Körper während der Simulation durchdringen. Bevor Kollisionen behandelt werden können (siehe Kapitel 5), müssen sie zunächst einmal erkannt werden. Außerdem muss die Kontaktgeometrie (siehe Abbildung 4.1) im Falle einer Kollision bestimmt werden. Dies sind die beiden Aufgaben der *Kollisionserkennung*. Die Kontaktgeometrie besteht aus al-

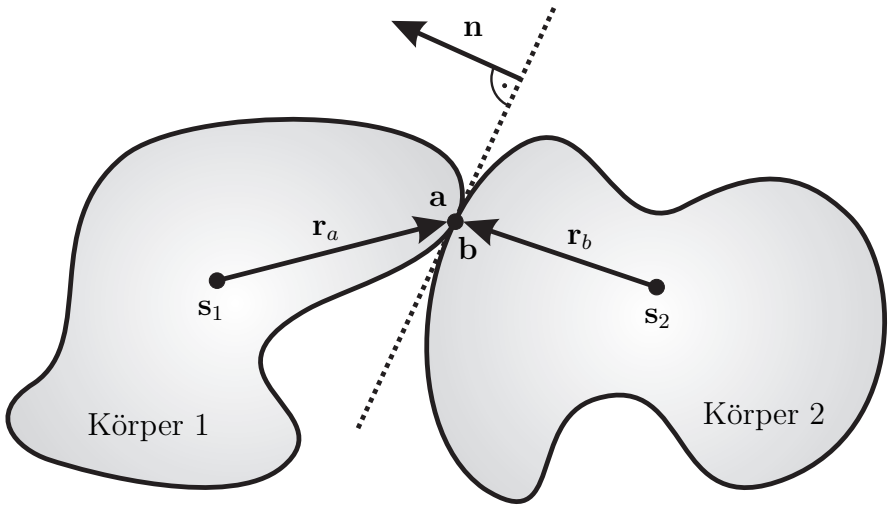


Abbildung 4.1: Kontaktgeometrie zweier kollidierender Körper

len Kontaktpunkten zwischen den beiden Körpern zum Zeitpunkt des Aufpralls. Das ist der Zeitpunkt, zu dem sich die Körper berühren, sich aber nicht gegenseitig durchdringen. Für jeden Kontakt muss außerdem die Normale $\mathbf{n}$ bestimmt werden, die senkrecht zur Kontaktebene steht. Diese sogenannte *Kontaktnormale* gehört ebenfalls zur Kontaktgeometrie. Sie wird benötigt, um eine Kollision in die

richtige Richtung aufzulösen. Im Folgenden wird angenommen, dass die Kontaktnormale immer in Richtung des ersten Körpers einer Kollision zeigt.

Da bereits einige frei verfügbare Bibliotheken mit schnellen Verfahren zur Kollisionserkennung existieren, wird in dieser Arbeit kein neues Verfahren entwickelt. Stattdessen werden vorhandene Verfahren miteinander verglichen, um ein geeignetes für diese Arbeit auszuwählen. Die meisten Verfahren liefern für jedes Paar von kollidierenden Körpern nur maximal ein Paar von Kontaktpunkten und eine Kontaktnormale zurück. Daher wird für diese Verfahren eine Erweiterung benötigt, mit der die restliche Kontaktgeometrie bestimmt werden kann.

Im Folgenden wird der Entwurf eines Kollisionserkennungssystems für diese Arbeit vorgestellt. Im nächsten Abschnitt wird zunächst gezeigt, wie die Kollisionserkennung in einen Simulationsschritt integriert wird. Anschließend werden die Anforderungen definiert, die das System zur Kollisionserkennung erfüllen muss. Ausgehend von diesen Anforderungen wird dann das System in drei Phasen unterteilt. Um das Kollisionserkennungssystem in die Realität umzusetzen, soll ein frei verfügbares Verfahren zur Kollisionserkennung verwendet werden. Daher wird ein allgemeiner Vergleich einiger frei verfügbarer Kollisionserkennungsbibliotheken durchgeführt. Die beiden Verfahren, die sich am besten für die Simulation eignen, werden dann genauer betrachtet. In Abschnitt 4.5 werden die Erweiterungen für die ausgewählten Verfahren vorgestellt, die jeweils benötigt werden, um die Anforderungen an das Kollisionserkennungssystem für die dynamische Simulation zu erfüllen. Am Ende des Kapitels werden dann genaue Messergebnisse zu den beiden erweiterten Verfahren präsentiert.

4.1 Ablauf eines Simulationsschrittes

Für jeden Körper in der dynamischen Simulation, der mit anderen kollidieren kann, ist ein sogenanntes *Kollisionsobjekt* definiert, das den Starrkörper bei der Kollisionserkennung repräsentiert. Dieses Kollisionsobjekt kann aus mehreren voneinander unabhängigen Geometrien bestehen. Dadurch kann zum Beispiel ein nicht konvexer Körper in konvexe Teile zerlegt werden. Die Geometrie des Kollisionsobjektes muss nicht der Geometrie entsprechen, die für die Darstellung verwendet und aus der der Trägheitstensor berechnet wird. Wenn der dargestellte Körper eine komplexe Geometrie hat, wird oftmals eine vereinfachte Form dieser Geometrie für die Kollisionserkennung verwendet. Dies reduziert den Berechnungsaufwand erheblich und erhöht dadurch die Geschwindigkeit der Kontaktbestimmung. Jedes Kollisionsobjekt besitzt eine Transformation, mit der alle seine Geometrien verschoben werden. Diese Transformation muss während der Simulation genau der des zugehörigen Starrkörpers entsprechen.

Ein Zeitschritt, in dem Kollisionen behandelt werden, hat den folgenden Ablauf

(siehe Abbildung 4.2). Am Anfang müssen die Transformationen aller Kollisions-

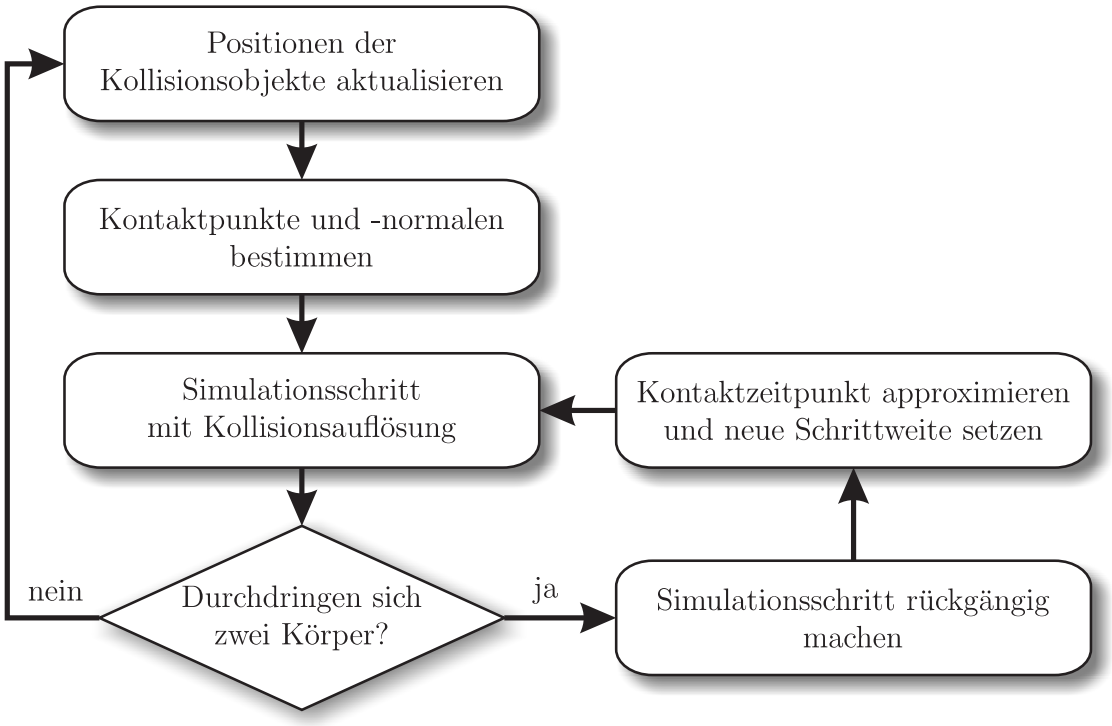


Abbildung 4.2: Ablauf eines Zeitschrittes mit Kollisionserkennung

objekte an die der zugehörigen Starrkörper angepasst werden. Danach bestimmt die Kollisionserkennung alle Kontaktpunkte und -normalen. Diese werden für die Kollisionsauflösung während des folgenden Simulationsschrittes benötigt. Die dynamische Simulation läuft in diskreten Zeitschritten ab und damit auch die Kollisionserkennung. Daher wird eine Kollision zwischen zwei Körpern oft erst erkannt, wenn sich die beiden Körper bereits gegenseitig durchdringen. Dieser Zustand ist bei der dynamischen Simulation allerdings unzulässig. Daher muss nach dem Simulationsschritt mit der Kollisionserkennung überprüft werden, ob sich zwei Körper durchdringen. Dafür müssen zunächst die Transformationen der Kollisionsobjekte aktualisiert werden. Anschließend wird ein Kollisionstest durchgeführt. In diesem Fall muss keine Kontaktgeometrie bestimmt werden. Es muss nur festgestellt werden, ob eine Durchdringung vorliegt oder nicht. Gegebenenfalls wird noch die Eindringtiefe ermittelt. Im Fall einer Durchdringung muss der Zeitpunkt der Kollision ermittelt werden. Um diesen *Kontaktzeitpunkt* zu bestimmen, muss der letzte Simulationsschritt rückgängig gemacht und mit einer neuen Zeitschrittweite erneut durchgeführt werden. Da man den Kontaktzeitpunkt im Allgemeinen nicht exakt berechnen kann, muss er approximiert werden. Der gesuchte Zeitpunkt wird dann in einer Schleife immer weiter angenähert, bis der Abstand der Körper innerhalb einer vorgegebenen Toleranz liegt.

Wenn die Eindringtiefe der Körper nicht bekannt ist, wird ein Intervallhalbierungsverfahren verwendet. Bei einer Durchdringung von Körpern wird das Zeitintervall halbiert und erneut ein Simulationsschritt durchgeführt. Durchdringen sich die Körper nach dem halben Schritt immer noch, dann wird im ersten Teilintervall nach dem Kontaktzeitpunkt weiter gesucht, andernfalls im zweiten Teilintervall. In einem Schritt des Intervallhalbierungsverfahrens genügt es, wenn die Kollisionserkennung nach der ersten festgestellten Durchdringung abbricht. Ist die Eindringtiefe bekannt, dann kann der Kontaktzeitpunkt mit Hilfe von linearer Interpolation approximiert werden. Versuche haben gezeigt, dass die Bestimmung des Kontaktzeitpunktes mit linearer Interpolation wesentlich schneller konvergiert als mit dem Intervallhalbierungsverfahren. Dies liegt daran, dass die Interpolationssuche einen durchschnittlichen Aufwand von $O(\log(\log n))$ und das Intervallhalbierungsverfahren einen Aufwand von $O(\log n)$ hat.

Nachdem der Kontaktzeitpunkt bestimmt ist, kann ein Simulationsschritt bis zu diesem Zeitpunkt durchgeführt werden. Anschließend wird mit dem nächsten Zeitschritt fortgefahren.

4.2 Anforderungen

Die Kollisionserkennung kann bei Modellen mit vielen Objekten zum Flaschenhals der dynamischen Simulation werden. Aus diesem Grund ist Geschwindigkeit eines der wichtigsten Kriterien für ein Verfahren zur Erkennung von Kollisionen. Da diskrete Verfahren schneller arbeiten als kontinuierliche (siehe Abschnitt 2.2.4), wird in dieser Arbeit ein diskretes Verfahren verwendet. Eine weitere Anforderung an ein Kollisionserkennungsverfahren ist, dass es erkennen muss, ob sich zwei Körper gegenseitig durchdringen. Eine Durchdringung stellt einen unzulässigen Zustand bei der dynamischen Simulation dar und muss daher verhindert werden. Im Fall einer Durchdringung wird der Kontaktzeitpunkt bestimmt (siehe Abschnitt 4.1). Dabei ist es von Vorteil, wenn die Kollisionserkennung die Eindringtiefe der Körper berechnen kann. Zum Zeitpunkt des Kontakts muss das Verfahren alle Kontaktpunkte zwischen den Körpern und die Kontaktnormalen zurück liefern, damit die Kollision korrekt aufgelöst werden kann. Wenn diese Kontaktgeometrie auch im Fall einer Durchdringung bestimmt werden kann, hat dies zwei Vorteile. Eine Kollision kann aufgelöst werden, ohne vorher den genauen Kontaktzeitpunkt zu bestimmen. Dadurch wird die Simulation schneller, allerdings nimmt die Genauigkeit der Simulation ab. Der zweite Vorteil ist, dass die Simulation auch dann stabil weiter läuft, wenn es die Kollisionsauflösung in einem Simulationsschritt nicht schaffen sollte, alle kollidierenden Körper auseinander zu bewegen. Die Kollisionserkennung muss robust sein und bei jedem denkbaren Zustand des Modells ein korrektes Ergebnis zurück liefern. Die letzte Anforderung an das Verfahren zur Kollisionserkennung ist, dass es eine möglichst allgemeine Geometrie der Kollisi-

onsobjekte unterstützen sollte. Zumindest muss das Verfahren allgemeine Polyeder handhaben können, da mit ihnen die Geometrie von jedem Körper angenähert werden kann.

4.3 Ablauf der Kollisionserkennung

Ein System für die Kollisionserkennung muss am Anfang eines Zeitschrittes testen, ob eine Kollision vorliegt. Ist dies nicht der Fall, so kann mit der Simulation ohne Kollisionsbehandlung fortgefahren werden. Im Fall einer Kollision muss das System alle Kontaktpunkte und -normalen bestimmen. Ein genauer Kollisionstest ist zeitaufwendig (siehe Abschnitt 2.2.3). Daher ist es sinnvoll, schnelle Techniken (siehe Abschnitt 2.2.2) einzusetzen, mit denen die Anzahl dieser Tests reduziert werden kann.

Das Kollisionserkennungssystem soll in drei Phasen arbeiten. In der ersten Phase findet eine Reduktion der genauen Kollisionstests statt. Diese Phase dient ausschließlich der Beschleunigung des Systems. In der nächsten Phase muss für jedes Paar von Körpern, bei dem in der ersten Phase eine Kollision nicht ausgeschlossen werden konnte, ein genauer Test durchgeführt werden. Bei diesem Test wird der Abstand bzw. die Eindringtiefe der beiden Körper ermittelt. Außerdem werden die nächsten Punkte und eine Kontaktnormale bestimmt. Ausgehend von den nächsten Punkten wird in der dritten Phase die gesamte Kontaktregion ermittelt. Bei der Bestimmung des Kontaktzeitpunktes wird nur getestet, ob sich zwei Körper durchdringen. Die Kontaktgeometrie wird in diesem Fall nicht benötigt und die dritte Phase muss nicht ausgeführt werden. Dadurch beschleunigt sich der Kollisionstest erheblich.

4.4 Vergleich

In diesem Abschnitt werden einige frei verfügbare Kollisionserkennungsbibliotheken miteinander verglichen. Für den Vergleich wurden die Bibliotheken I-Collide, RAPID, IMMPACT, SWIFT++, V-Clip und SOLID ausgewählt. Zunächst werden die Funktionalitäten und die Eigenschaften jeder einzelnen Bibliothek beschrieben. Anschließend wird anhand dieser Eigenschaften ein Vergleich durchgeführt, um ein geeignetes Verfahren für diese Arbeit auszuwählen.

4.4.1 I-Collide

Die Bibliothek I-Collide [CLMP95] berechnet den kürzesten Abstand zwischen zwei konvexen Polyedern mit dem Algorithmus von Ming C. Lin und John F. Can-

ny (siehe Abschnitt 2.2.3). Die Kollisionserkennung von nicht konvexen Körpern wird nicht direkt unterstützt. Nicht konvexe Körper müssen für das Verfahren vom Benutzer in konvexe Teile zerlegt werden. In I-Collide werden für alle Körper achsenorientierte Quader als Hüllkörper verwendet. Die Überschneidungen der Hüllkörper werden mit dem Sweep-and-Prune-Algorithmus (siehe Abschnitt 2.2.2) bestimmt. In der dynamischen Simulation bewegen sich die Körper während eines Simulationsschrittes nicht sehr weit, da die Zeitschrittweite im Allgemeinen relativ klein ist. Dadurch ändert sich auch die Geometrie, die durch die Punkte der Körper definiert ist, während eines Schrittes nur minimal. Diese zeitliche und geometrische Kohärenz wird beim Sweep-and-Prune-Algorithmus und bei dem verwendeten Lin-Canny-Algorithmus ausgenutzt. Die Kollisionserkennung kann dadurch erheblich beschleunigt werden.

Der Lin-Canny-Algorithmus kann den Abstand von zwei Körpern nur dann berechnen, wenn sie sich nicht gegenseitig durchdringen. Die Eindringtiefe kann daher nicht berechnet werden. Dies stellt einen Nachteil des Verfahrens dar, da die Eindringtiefe eine wichtige Größe bei der Bestimmung des Kontaktzeitpunktes ist. Der Fall einer Durchdringung wird von dem Algorithmus nicht behandelt und das Verfahren kann in eine Endlosschleife geraten. Die in I-Collide verwendete Umsetzung des Lin-Canny-Algorithmus arbeitet im Fall einer Durchdringung mit numerischen Toleranzen, um das Problem zu lösen [Mir98]. Dies kann allerdings zu numerischen Problemen führen.

4.4.2 RAPID

Die Bibliothek RAPID [GLM96] arbeitet im Gegensatz zu I-Collide auf einer unstrukturierten Menge von Polygonen (Polygonsuppe). Diese Polygone müssen als eine Menge von Dreiecken übergeben werden. Die Abkürzung RAPID steht für „Robust and Accurate Polygon Interference Detection“. Bei diesem Verfahren wird ein Quader, der am Kollisionsobjekt ausgerichtet ist, als Hüllkörper verwendet. Am Anfang der Kollisionserkennung wird für jedes Objekt eine Hierarchie dieser Hüllkörper bestimmt, die die Geometrie des Objektes annähert. Für einen Kollisionstest durchläuft das Verfahren die Hierarchien zweier Körper, um Überschneidungen zwischen den Hüllkörpern zu finden. Ein schneller Test, der auf dem Separating Axis Theorem (siehe Abschnitt 2.2.2) basiert, wird zum Erkennen von Überschneidungen verwendet. Die Bibliothek RAPID ermöglicht schnelle Kollisionstests bei komplexen Modellen, die aus sehr vielen Polygonen bestehen. Im Gegensatz zu anderen Bibliotheken verfügt RAPID über keine Phase, in der Kollisionstests reduziert werden. Die komplette Kollisionserkennung arbeitet ausschließlich auf den Hierarchien von Hüllkörpern. Bei der Kollisionserkennung mit diesem Verfahren werden die Dreiecke von zwei Objekten auf Schnitt getestet. Daher ist es mit diesem Verfahren nicht möglich, den Abstand zwischen zwei Körpern bzw. die nächsten Elemente der Körper zu bestimmen. Außerdem liefert die Bibliothek RAPID im

Fall einer Kollision keine Kontaktgeometrie zwischen zwei Körpern zurück, sondern nur eine Menge von sich schneidenden Dreiecken. Die Bibliothek PQP [LGLM99] ist eine Weiterentwicklung von RAPID. Die Abkürzung PQP steht für „A Proximity Query Package“. Mit PQP ist es möglich, den Abstand zwischen zwei Körpern zu bestimmen. Allerdings wird auch bei dieser Bibliothek im Fall einer Kollision keine Kontaktgeometrie zurück geliefert.

4.4.3 IMMPACT

Die Kollisionserkennungsbibliothek IMMPACT wurde entwickelt, um mit einem Avatar durch das komplexe Modell eines Kraftwerks zu navigieren [WLML98, WLML99b, WLML99a]. Das Modell des Kraftwerks besteht aus Millionen von Dreiecken und ist statisch. IMMPACT ist die Abkürzung für „Interactive Massive Model Proximity And Collision Tester“. Das Ziel dieser Bibliothek ist eine schnelle Kollisionserkennung für Modelle, die aus mehreren Millionen geometrischen Primitiven bestehen und damit unter Umständen nicht in den Hauptspeicher passen. Dies wird durch Unterteilung des Modells in kleinere Partitionen und lokale Kollisionstests in diesen Partitionen erreicht. Durch diese Vorgehensweise muss nur ein kleiner Teil des Modells für einen Kollisionstest im Speicher gehalten werden. Für ein Modell wird zunächst ein Überlappungsgraph aufgestellt. In diesem Graph existiert für jedes Objekt im Modell ein Knoten. Zwei Knoten werden mit einer Kante verbunden, wenn sich die Hüllkörper der zugehörigen Objekte überlappen. Dieser Graph wird anschließend in verschiedenen Schritten in kleinere Teilgraphen zerlegt. Die Berechnung der Teilgraphen kann sehr aufwendig sein und wird in einem Vorverarbeitungsschritt durchgeführt. Zur Laufzeit werden die Teilgraphen verwendet, um die Kollisionstests lokal durchzuführen. Das Verfahren wird mit Hilfe von Hierarchien von Hüllkörpern beschleunigt.

Die Bibliothek IMMPACT ist geeignet für komplexe Modelle mit sehr großen statischen Objekten. Dagegen bestehen Modelle in der dynamischen Simulation im Allgemeinen aus vielen kleineren, aber dynamischen Objekten. Für solche Modelle kann die Unterteilung der Szene nicht in einem Vorverarbeitungsschritt vorgenommen werden. Daher ist das Verfahren für Modelle der dynamischen Simulation ungeeignet.

4.4.4 SWIFT++

Die Bibliothek SWIFT++ [EL01] ist eine Erweiterung von SWIFT [EL00a]. Die Abkürzung SWIFT steht für „Speedy Walking via Improved Feature Testing“. SWIFT ist eine objektorientierte Weiterentwicklung der Bibliothek I-Collide, die zusätzlich mehr Funktionalität bietet. Die Kollisionserkennung arbeitet ausschließlich mit konvexen Polyedern. Damit auch nicht konvexe Körper bei der Kollisions-

erkennung verwendet werden können, liefert SWIFT++ ein Werkzeug mit, das polygonale Modelle in einem Vorverarbeitungsschritt in konvexe Teile zerlegt. Das Ergebnis dieser Zerlegung wird als ein hierarchisches Modell gespeichert und kann von SWIFT++ als Kollisionsobjekt geladen werden. Das Verfahren zur Kollisionserkennung dieser Bibliothek basiert auf dem Lin-Canny-Algorithmus. Dieser Algorithmus wurde um die sogenannte „Voronoi Marching"-Technik erweitert. Dabei wird zunächst für jedes Objekt in einem Vorverarbeitungsschritt eine Hierarchie von konvexen Polyedern mit unterschiedlichem Detaillierungsgrad bestimmt. Jedes Polyeder in dieser Hierarchie hat die Eigenschaft, dass es das ursprüngliche Polyeder umschließt. Um die nächsten Elemente und damit die Distanz zwischen zwei Körpern zu bestimmen, werden dann zur Laufzeit die Voronoi-Regionen der konvexen Polyeder auf den verschiedenen Ebenen der Hierarchien der Körper abgelaufen. Anschließend können die nächsten Elemente, die nächsten Punkte der Körper, die Distanz und eine Kontaktnormale zurück gegeben werden. Das in dieser Bibliothek verwendete Verfahren kann im Gegensatz zum ursprünglichen Lin-Canny-Algorithmus Durchdringungen von Körpern erkennen. Allerdings kann auch mit SWIFT++ die Eindringtiefe im Fall einer Durchdringung nicht berechnet werden. SWIFT++ verwendet achsenorientierte Quader als Hüllkörper. Überschneidungen dieser Quader werden mit dem Sweep-and-Prune-Algorithmus bestimmt. Damit werden in der ersten Phase die Kollisionstests reduziert. Die Kollisionserkennungsbibliothek ist robust und sehr schnell [EL00b].

4.4.5 V-Clip

V-Clip (oder auch Voronoi-Clip) ist eine Bibliothek, die ebenfalls mit Voronoi-Regionen arbeitet und auf dem Lin-Canny-Algorithmus aufbaut [Mir98]. Das verwendete Verfahren erlaubt eine Kollisionserkennung für konvexe Polyeder. Nicht konvexe Objekte müssen vor der Kollisionserkennung in konvexe Teile zerlegt werden. Gegenüber dem Lin-Canny-Algorithmus wurde in V-Clip speziell die Robustheit des Verfahrens verbessert. V-Clip verwendet keine numerischen Toleranzen in der Implementierung, hat keine Probleme mit Endlosschleifen und kann sogar den Fall einer Durchdringung behandeln. Das Verfahren kann für zwei Körper die nächsten Elemente, die nächsten Punkte, eine Kontaktnormale und die Distanz zwischen den Körpern berechnen. Dies funktioniert auch im Fall einer Durchdringung der Körper. Anstatt der Distanz wird dann die Eindringtiefe zurück gegeben. Mit V-Clip kann, wie mit den anderen Verfahren, nur ein Paar von Kontaktpunkten und eine Kontaktnormale berechnet werden. Allerdings werden die Kontaktpunkte bei V-Clip auch dann zurück gegeben, wenn die beiden kollidierenden Körper sich gegenseitig durchdringen. Dies ist ein großer Vorteil, da mit diesem Verfahren auch bei Durchdringungen eine Auflösung der Kollisionen möglich ist. Um zeitliche und geometrische Kohärenz auszunutzen, werden die nächsten Elemente zweier Körper gespeichert und bei der nächsten Anfrage als Startpunkt des Algorithmus verwen-

det. Dadurch wird das Verfahren sehr schnell, wenn sich die Körper zwischen zwei Anfragen nicht weit bewegen.

Die Bibliothek V-Clip behandelt ausschließlich Kollisionsanfragen für zwei Körper. Es ist nicht möglich, die Kollisionen für das gesamte Modell auf einmal abzufragen. Eine Phase, in der die Anzahl der benötigten Kollisionstests für ein Modell mit mehreren Körpern mit Hilfe von Hüllkörpern oder anderen Verfahren reduziert wird, wurde nicht implementiert.

4.4.6 SOLID

Die Bibliothek SOLID [vdB97, vdB99] basiert auf dem Algorithmus von Elmer G. Gilbert, Daniel W. Johnson und Sathiya S. Keerth (siehe Abschnitt 2.2.3). SOLID ist bei den Kollisionsobjekten nicht auf konvexe Polyeder beschränkt, sondern unterstützt auch Kugeln, Zylinder, Ellipsoide und weitere konvexe Objekte. Stephen Cameron beschreibt in [Cam97a, Cam97b], dass durch eine Abänderung des GJK-Algorithmus die gleiche erwartete Laufzeit erreicht werden kann, die der Lin-Canny-Algorithmus hat. Diese Erweiterung des GJK-Algorithmus wurde in SOLID umgesetzt. Dadurch kann das Kollisionserkennungsverfahren dieser Bibliothek mit der Geschwindigkeit des Lin-Canny-Algorithmus mithalten. In [vdB01] wird eine Methode präsentiert, mit der die Eindringtiefe zweier Körper berechnet werden kann. Diese Methode wurde ebenfalls in SOLID integriert.

SOLID arbeitet in drei Phasen. In der ersten Phase werden achsenorientierte Quader als Hüllkörper verwendet. Mit Hilfe des Sweep-and-Prune-Algorithmus werden diese Hüllkörper auf Durchdringungen überprüft. Dadurch lassen sich die Kollisionstests in den folgenden beiden Phasen erheblich reduzieren. Objekte, die aus mehreren Primitiven zusammengesetzt sind, werden durch eine Hierarchie von Hüllkörpern angenähert. Mit diesen Hierarchien können in der zweiten Phase Teile eines zusammengesetzten Objekts von den weiteren Kollisionstests ausgeschlossen werden. In der dritten Phase werden dann die Primitive, die potentielle Kandidaten für eine Kollision sind, genau untersucht. Im Fall einer Kollision liefert SOLID die nächsten Punkte, eine Kontaktnormale und den Abstand der Körper zurück. Im Fall einer Durchdringung der Körper wird anstatt des Abstands die Eindringtiefe zurück gegeben.

Die Bibliothek SOLID ist vollständig objektorientiert programmiert und bietet Schnittstellen für Callback-Funktionen an, mit denen das Verfahren erweitert werden kann. Von den in dieser Arbeit untersuchten Bibliotheken ist SOLID die einzige, die noch weiterentwickelt wird.

4.4.7 Vergleich der Verfahren

4.4.7.1 Geometrie

Ein Modell in der dynamischen Simulation besteht häufig aus vielen dynamischen Körpern. Die Geometrien dieser Körper können meistens durch Polyeder beschrieben werden und sind oft nicht sehr komplex. Allerdings ist es bei der Erkennung von Kollisionen sinnvoll, runde Körper, wie z. B. Kugeln oder Zylinder, nicht mit einem Polyeder anzunähern, sofern dies möglich ist. Andernfalls können Simulationen, in denen ein runder Körper rollt, nur ungenau durchgeführt werden.

Die Bibliothek IMMPACT ist ungeeignet für die Kollisionserkennung bei einem Modell mit vielen dynamischen Körpern ohne komplexe Geometrie. Daher wird sie im Folgenden nicht mehr weiter betrachtet. Die weiteren Verfahren mit Ausnahme von RAPID arbeiten mit konvexen Polyedern. Nicht konvexe Körper müssen bei diesen Verfahren in konvexe Teile zerlegt werden. SOLID unterstützt zusätzlich Kugeln, Zylinder und weitere konvexe Objekte und ist damit die einzige Bibliothek, die ohne Erweiterung runde Objekte beherrscht. RAPID ist das einzige der Verfahren, das die Kollisionserkennung von Polygonsuppen unterstützt.

4.4.7.2 Geschwindigkeit

Eine wichtige Anforderung an ein Verfahren für die Kollisionserkennung in der dynamischen Simulation ist eine hohe Geschwindigkeit. Um diese Anforderung bei einem Modell mit vielen Körpern zu erfüllen, ist es erforderlich, die Anzahl der zeitaufwendigen, genauen Kollisionstests so gering wie möglich zu halten. Daher werden bei den meisten Verfahren Hüllkörper eingesetzt, um die Anzahl der nötigen Kollisionstests zu reduzieren. RAPID und V-Clip bilden hier Ausnahmen. Bei RAPID wird der gesamte Kollisionstest ausschließlich auf Hierarchien von Hüllkörpern ausgeführt. Dagegen wurde in der Bibliothek V-Clip nur ein schneller Kollisionstest zwischen zwei Körpern implementiert, der mit Voronoi-Regionen arbeitet und auf dem Lin-Canny-Algorithmus basiert. Die Verfahren zur Bestimmung der Kontaktpunkte basieren bei I-Collide und Swift++ ebenfalls auf dem Lin-Canny-Algorithmus. Durch Ausnutzen von Kohärenz kann ein Kollisionstest zwischen zwei Körpern mit dem Lin-Canny-Algorithmus fast in konstanter Laufzeit durchgeführt werden [Mir97]. Das bedeutet, die Laufzeit ist fast unabhängig von der Komplexität der getesteten Körper. SOLID verwendet eine erweiterte Variante des GJK-Algorithmus, die das gleiche Laufzeitverhalten wie der Lin-Canny-Algorithmus aufweist.

4.4.7.3 Robustheit

Ein weiteres wichtiges Kriterium für die Auswahl eines geeigneten Verfahrens ist die Robustheit. Die Bibliothek I-Collide kann im Fall einer Durchdringung zweier Körper in eine Endlosschleife geraten. Daher darf I-Collide nur dann eingesetzt werden, wenn sichergestellt werden kann, dass keine Durchdringung während der Simulation möglich ist. Dies ist allerdings nur machbar, wenn der frühest mögliche Kollisionszeitpunkt von zwei Körpern berechnet wird, wie in der Arbeit von Brian V. Mirtich [MC94, MC95].

SWIFT++ und V-Clip können durch eine Erweiterung des Lin-Canny-Algorithmus Durchdringungen erkennen und ermöglichen damit eine wesentlich robustere Kollisionserkennung ohne Sonderbehandlung. In SOLID wurde der ursprüngliche GJK-Algorithmus erweitert, um Probleme zu lösen, die durch Rundungsfehler bei der Berechnung verursacht werden [vdB99]. Durch diese Erweiterungen ist die Kollisionserkennung mit SOLID ebenfalls sehr robust. Der bei RAPID verwendete Test, der überprüft, ob sich zwei am jeweiligen Kollisionsobjekt ausgerichtete Hüllkörper schneiden, ist numerisch robust. Außerdem wird keine Sonderbehandlung für Spezialfälle benötigt, wenn z. B. zwei Flächen parallel zueinander sind.

4.4.7.4 Kontaktgeometrie

Für eine genaue Kollisionsauflösung wird eine vollständige Kontaktgeometrie benötigt. Diese muss alle Kontaktpunkte und -normalen zum Zeitpunkt der Kollision umfassen. Keine der Bibliotheken im Vergleich bestimmt die vollständige Kontaktgeometrie. RAPID liefert bei einer Kollisionsanfrage für zwei Körper alle Dreiecke zurück, die sich schneiden. Es werden allerdings keine Kontaktpunkte und keine Kontaktnormale ermittelt. Der Abstand bzw. die Eindringtiefe der Körper wird ebenfalls nicht von RAPID bestimmt. Alle anderen Verfahren bestimmen die jeweils nächsten Punkte von zwei Körpern und eine Kontaktnormale, wenn sich die Körper nicht durchdringen. In diesem Fall kann auch mit jedem der Verfahren der Abstand der Körper bestimmt werden. Den Fall einer Durchdringung können die Bibliotheken SWIFT++, V-Clip und SOLID erkennen. SWIFT++ liefert dabei allerdings nur zurück, ob eine Durchdringung stattgefunden hat oder nicht. Die anderen beiden Verfahren können zusätzlich noch eine Eindringtiefe berechnen. Außerdem liefern V-Clip und SOLID im Fall einer Durchdringung Kontaktpunkte und eine Kontaktnormale zurück. Das Erkennen von Durchdringungen ist eine wichtige Eigenschaft für ein Kollisionserkennungsverfahren. Diese Eigenschaft wird benötigt, um den genauen Zeitpunkt einer Kollision ohne eine Sonderbehandlung zu bestimmen.

4.4.7.5 Fazit

Die Bibliothek IMMPACT ist ungeeignet für die dynamische Simulation, da sie für die Erkennung von Kollisionen mit sehr komplexen, statischen Objekten entworfen wurde, die in der dynamischen Simulation im Allgemeinen nicht vorkommen. Von den restlichen Bibliotheken ist RAPID die einzige, die eine Kollisionserkennung für Polygonsuppen erlaubt. Alle anderen Verfahren arbeiten mit konvexen Polyedern. Nicht konvexe Objekte müssen bei diesen Verfahren vor der Kollisionserkennung in konvexe Teile zerlegt werden. SWIFT++ stellt als einzige Bibliothek ein Werkzeug zur Verfügung, mit dem diese Zerlegung in einem Vorverarbeitungsschritt vom Benutzer durchgeführt werden kann.

Die Kollisionserkennungsbibliothek I-Collide verwendet den ursprünglichen Lin-Canny-Algorithmus. Dieser ist im Fall einer Durchdringung von zwei Körpern nicht robust. Damit kann I-Collide nur mit einer Sonderbehandlung eingesetzt werden. Da SWIFT++ und V-Clip auch auf dem Lin-Canny-Algorithmus basieren, aber durch Erweiterungen des Verfahrens wesentlich robuster sind, wird I-Collide als Verfahren für die Kollisionserkennung in dieser Arbeit ausgeschlossen. Die Bibliotheken RAPID und SOLID ermöglichen ebenfalls eine robuste Kollisionserkennung. Damit bleiben vier Verfahren zur Auswahl.

RAPID hat den Nachteil, dass nur erkannt wird, ob sich Dreiecke der Körper schneiden oder nicht. Eine Bestimmung des Abstands oder der Eindringtiefe von zwei Körpern ist mit diesem Verfahren nicht möglich. RAPID gibt im Fall einer Kollision die Menge der sich schneidenden Dreiecke zurück. Die Kontaktpunkte oder eine Kontaktnormale werden von dieser Bibliothek nicht bestimmt. Da diese Daten für die Behandlung von Kollisionen aber dringend benötigt werden, ist RAPID ungeeignet für die dynamische Simulation. SWIFT++, V-Clip und SOLID können die nächsten Punkte, den Abstand und eine Kontaktnormale bestimmen. Mit diesen drei Verfahren ist es ebenfalls möglich, eine Durchdringung zu erkennen. Allerdings kann nur mit V-Clip und SOLID eine Eindringtiefe berechnet werden. Da dieser Wert wichtig bei der Bestimmung des genauen Kollisionszeitpunktes ist, stellt dies einen großen Nachteil von SWIFT++ dar.

Alle Bibliotheken im Vergleich wurden entworfen, um eine schnelle Kollisionserkennung zu ermöglichen. Dies wird durch den Einsatz von Hüllkörpern zur Reduktion der Kollisionstests, durch Ausnutzen der zeitlichen und geometrischen Kohärenz und durch weitere Techniken erreicht. V-Clip hat den Nachteil, dass es keine Reduktion der Kollisionstests mit Hilfe von Hüllkörpern durchführt. In der Literatur gibt es keine umfassenden Geschwindigkeitsvergleiche für alle vorgestellten Verfahren.

Tabelle 4.1 fasst den Vergleich der in Frage kommenden Bibliotheken zusammen. Die beiden Verfahren, die sich am besten für die dynamische Simulation eignen, sind SOLID und V-Clip. Beide Verfahren sind robust, schnell, können zwei Kon-

	I-Collide	RAPID	SWIFT++	V-Clip	SOLID
Geometrie	konvexe Polyeder	Polygon-suppen	konvexe Polyeder	konvexe Polyeder	konvexe Polyeder
Geschwindigkeit	+	+	+	+	+
Robustheit	−	+	+	+	+
vollständige Kontaktgeometrie	−	−	−	−	−
Eindringtiefe	−	−	−	+	+

Tabelle 4.1: Vergleich der Kollisionserkennungsbibliotheken

taktpunkte sowie eine Kontaktnormale bestimmen und berechnen gegebenenfalls die Eindringtiefe. SOLID hat zusätzlich noch die Vorteile, dass es eine Reihe von runden Körpern unterstützt, eine Schnittstelle für Erweiterungen bietet und noch weiterentwickelt wird. SOLID basiert auf dem GJK-Algorithmus und V-Clip auf dem Algorithmus von Lin und Canny. Dies sind zwei der schnellsten bekannten Algorithmen, mit denen ein genauer Kollisionstest durchgeführt werden kann. Im Folgenden werden SOLID und V-Clip so erweitert, dass sie für den Einsatz in der Dynamiksimulation dieser Arbeit geeignet sind. Anschließend wird ein Geschwindigkeitsvergleich der beiden Verfahren durchgeführt.

4.5 Erweiterungen

Im letzten Abschnitt wurden die Bibliotheken V-Clip und SOLID als mögliche Verfahren für die Kollisionserkennung ausgewählt. Damit sie die Anforderungen an die Kollisionserkennung aus Abschnitt 4.2 erfüllen, müssen sie um verschiedene Teile erweitert werden. Alle implementierten Erweiterungen werden in diesem Abschnitt vorgestellt.

Beide Verfahren werden um eine Matrix erweitert, in der gespeichert ist, welche Objekte miteinander auf Kollision getestet werden. Mit Hilfe dieser Matrix ist es möglich, bestimmte Paare von Objekten von der Kollisionserkennung auszuschließen. Dadurch kann die Kollisionserkennung beschleunigt werden. Zum Beispiel macht es keinen Sinn zu überprüfen, ob zwei statische Objekte miteinander kollidieren. In bestimmten Fällen kann es auch erwünscht sein, dass sich zwei Objekte durchdringen dürfen.

4.5.1 V-Clip

4.5.1.1 Reduktion der Kollisionstests

Die Bibliothek V-Clip bietet ausschließlich einen genauen Kollisionstest für zwei Körper an. Über eine erste Phase, in der die Anzahl der Kollisionstests reduziert wird, verfügt sie nicht. Aus diesem Grund wurde das Verfahren um eine solche Phase erweitert. Für die Reduktion der Kollisionstests werden achsenorientierte Quader als Hüllkörper für die Körper verwendet. Diese Quader werden mit Hilfe des Sweep-and-Prune-Algorithmus auf Überschneidungen getestet. Dieser Algorithmus nutzt die zeitliche und die geometrische Kohärenz aus. Dadurch ist sein erwarteter Aufwand linear. Durch die Implementierung der ersten Phase konnte die Kollisionserkennung mit V-Clip erheblich beschleunigt werden.

4.5.1.2 Kugeln als Kollisionsobjekte

V-Clip unterstützt bei der Kollisionserkennung nur Körper, die aus konvexen Polyedern zusammengesetzt werden können. Runde Objekte müssen daher durch Polyeder angenähert werden. Bei der dynamischen Simulation werden oft Kugeln verwendet. Wenn Kugeln durch Polyeder approximiert werden, rollen sie in der Simulation nicht sauber. Daher wurde das Verfahren für diese Arbeit so erweitert, dass auch Kugeln bei der Kollisionserkennung verwendet werden können. Damit wird zumindest die wichtigste Art der runden Körper unterstützt.

Für die Erweiterung von V-Clip um die Unterstützung von Kugeln müssen zusätzlich Kollisionen zwischen zwei Kugeln und Kollisionen zwischen einer Kugel und einem Polyeder erkannt werden. Dafür müssen in beiden Fällen die Distanz, eine Kontaktnormale und zwei Kontaktpunkte bestimmt werden können. Die Di-

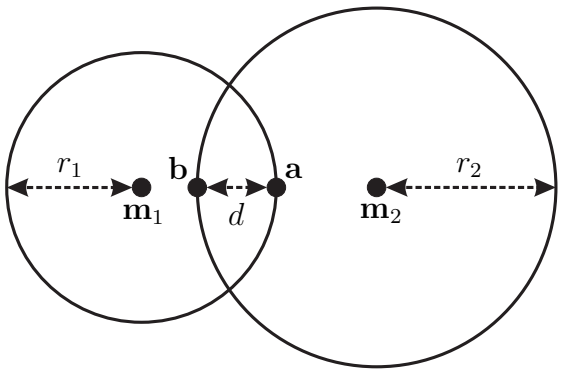


Abbildung 4.3: Distanzbestimmung zwischen zwei Kugeln

stanz d zwischen zwei Kugeln (siehe Abbildung 4.3) kann berechnet werden, indem

die Radien der Kugeln vom Abstand der beiden Mittelpunkte abgezogen werden:

$$d = |\mathbf{m}_1 - \mathbf{m}_2| - (r_1 + r_2).$$

Wenn sich die Kugeln genau in einem Punkt berühren, dann gilt $d = 0$. Bei negativen Werten von d liegt eine Durchdringung der Kugeln vor und d gibt dabei die Eindringtiefe an. Die Kontaktnormale ist durch den normalisierten Vektor von $\mathbf{m}_2$ nach $\mathbf{m}_1$ bestimmt:

$$\mathbf{n} = \frac{\mathbf{m}_1 - \mathbf{m}_2}{|\mathbf{m}_1 - \mathbf{m}_2|}.$$

Die nächsten Punkte **a** und **b** der beiden Kugeln sind die Schnittpunkte der Geraden durch die beiden Mittelpunkte mit den Oberflächen der Kugeln:

$$\begin{aligned} \mathbf{a} &= \mathbf{m}_1 - r_1\,\mathbf{n} \\ \mathbf{b} &= \mathbf{m}_2 + r_2\,\mathbf{n}. \end{aligned}$$

Die Distanz zwischen einer Kugel und einem konvexen Polyeder wird mit Hilfe der Voronoi-Regionen des Polyeders bestimmt (siehe Abbildung 4.4). Zunächst

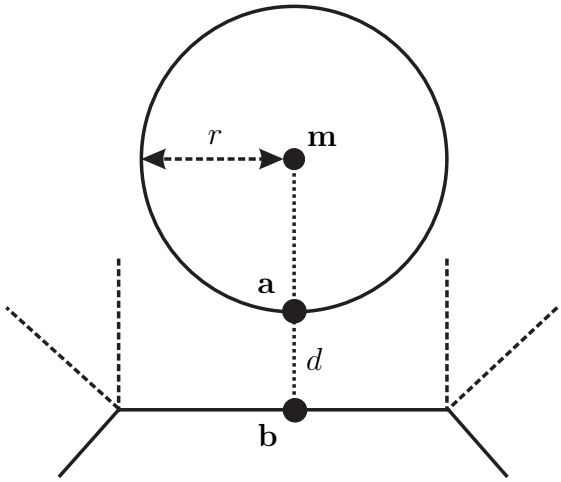


Abbildung 4.4: Distanz zwischen einer Kugel und einem konvexen Polyeder

wird die Voronoi-Region des Polyeders ermittelt, die den Mittelpunkt **m** der Kugel beinhaltet. Das zu dieser Region zugehörige Element des Polyeders ist das nächste Element zum Mittelpunkt. Damit kann der nächste Punkt des Polyeders zum Mittelpunkt bestimmt werden. Da die Bibliothek V-Clip die nächsten Punkte von zwei Körpern mit Voronoi-Regionen bestimmt, kann V-Clip auch dazu verwendet werden, den nächsten Punkt **b** eines Polyeders zu einer Kugel zu ermitteln. Ein Kollisionstest zwischen dem gegebenen Polyeder und einem Körper, der nur aus dem Mittelpunkt **m** der Kugel besteht, liefert das gewünschte Ergebnis. Anschließend kann der nächste Punkt **a** der Kugel ermittelt werden, indem die Gerade zwischen

dem Punkt **b** und dem Mittelpunkt **m** mit der Kugeloberfläche geschnitten wird. Diese Gerade liegt parallel zu der normierten Kontaktnormale **n**, die V-Clip zurück liefert. Daher kann der Schnittpunkt folgendermaßen berechnet werden:

$$\mathbf{a} = \mathbf{m} - r\,\mathbf{n}.$$

Die Bestimmung der nächsten Punkte funktioniert auch im Fall einer Durchdringung der beiden Körper, da V-Clip diesen Fall behandelt. V-Clip gibt bei dem durchgeführten Kollisionstest den Abstand bzw. die Eindringtiefe d_m des Mittelpunktes **m** zurück. Die Distanz der Kugel zu dem Polyeder ist dann $d = d_m - r$.

4.5.1.3 Bestimmung der Kontaktgeometrie

Die letzte Erweiterung für V-Clip ist die Bestimmung der vollständigen Kontaktgeometrie. V-Clip liefert für zwei Körper die nächsten Punkte der Körper und eine Kontaktnormale zurück. Bei einer Kollision zwischen zwei Polyedern kann allerdings eine ganze Kontaktregion existieren (siehe Abbildung 4.5). Um diese

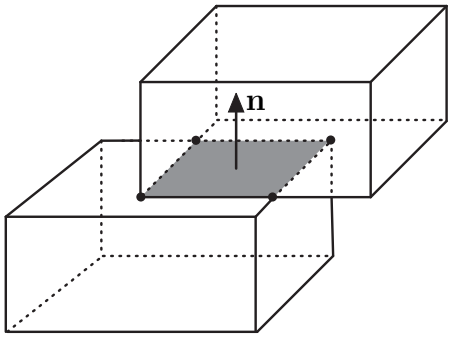


Abbildung 4.5: Kontaktregion bei der Kollision von zwei Körpern

Kontaktregion zu bestimmen, müssen die Punkte beider Körper ermittelt werden, deren Abstand kleiner als ein vorgegebener Toleranzwert ε_c ist. Es genügt dabei, die Eckpunkte der Region zu bestimmen, da nur diese für die Kollisionsauflösung benötigt werden. Bei konvexen Körpern liegen alle Kontaktpunkte in einer gemeinsamen Ebene. Daher haben auch alle Kontakte die gleiche Kontaktnormale. Diese ist nach einem Kollisionstest mit V-Clip bekannt.

In genau zwei Fällen ist es möglich, dass zwei Körper mehr als einen Kontakt haben. Im Fall, dass zwei Flächen der Körper parallel zueinander sind und wenn eine Kante parallel zu einer Fläche des anderen Körpers liegt. In beiden Fällen liegen alle Kontaktpunkte innerhalb einer Fläche des ersten und einer Fläche des zweiten Körpers, da eine Kante den Rand einer Fläche darstellt. Zunächst müssen diese Flächen bestimmt werden. V-Clip gibt bei einem Kollisionstest die nächsten Elemente der Polyeder zurück. Wenn es sich bei einem Element um eine Fläche handelt, dann ist sie die gesuchte Fläche für den zugehörigen Körper. Im Fall einer

Kante oder eines Punktes sind die anliegenden Flächen Kandidaten für die gesuchte Fläche. Wenn es mehrere Kandidaten gibt, dann werden die zwei Flächen ausgewählt, die möglichst parallel zueinander sind. Denn nur bei diesen Flächen sind bei konvexen Polyedern weitere Kontaktpunkte denkbar. Die gesuchten Flächen können mit Hilfe des Skalarprodukts der Flächennormalen ermittelt werden.

Nachdem die gesuchten Flächen bestimmt wurden, müssen sie auf weitere Kontaktpunkte untersucht werden. Weitere Kontakte können auf den Flächen selbst, auf den anliegenden Kanten oder in den Eckpunkten der Fläche liegen. Für je ein Element e_1 des ersten Körpers und ein Element e_2 des zweiten Körpers wird mit Hilfe der bereits bekannten Voronoi-Regionen überprüft, ob ihre nächsten Punkte $\mathbf{a} \in e_1$ und $\mathbf{b} \in e_2$ weitere Kontaktpunkte sind. Dies ist der Fall, wenn der Punkt $\mathbf{a}$ in der Voronoi-Region des Elements e_2 und $\mathbf{b}$ in der Voronoi-Region des Elements e_1 liegt [Mir98]. Außerdem muss der Abstand der beiden Punkte innerhalb der vorgegebenen Toleranz ε_c liegen. Auf diese Weise können alle Eckpunkte der Kontaktregion bestimmt werden.

4.5.2 SOLID

Die Bibliothek SOLID verfügt im Gegensatz zu V-Clip über eine erste Phase, in der die Kollisionstests reduziert werden. Außerdem unterstützt SOLID auch verschiedene runde Körper bei der Kollisionserkennung. Die einzige Anforderung aus Abschnitt 4.2, die SOLID nicht erfüllt, ist die Bestimmung einer vollständigen Kontaktgeometrie.

4.5.2.1 Bestimmung der Kontaktgeometrie

SOLID verwendet keine Voronoi-Regionen bei der Bestimmung der nächsten Punkte. Daher kann das für V-Clip verwendete Verfahren zur Bestimmung der Kontaktregion nicht direkt bei SOLID angewendet werden. Im Fall einer Kollision zweier Körper werden von SOLID die nächsten Punkte und eine Kontaktnormale zurück gegeben. Die nächsten Elemente werden von der Bibliothek nicht ermittelt, können allerdings nach dem Kollisionstest mit Hilfe der nächsten Punkte und der Kontaktnormale bestimmt werden. Der erwartete Aufwand für die Bestimmung der nächsten Elemente ist konstant, wenn die Kohärenz der Szene ausgenutzt wird.

Um das nächste Element eines Körpers zu ermitteln, wird zunächst die Fläche des Körpers bestimmt, die den nächsten Punkt enthält und die möglichst orthogonal zur Kontaktnormale steht. Wenn die Normale dieser Fläche innerhalb eines Toleranzbereichs parallel zur Kontaktnormale ist, dann ist die Fläche das nächste Element. Andernfalls werden alle an die Fläche anliegenden Kanten untersucht, ob sie den nächsten Punkt enthalten. Bei einer Kante, die den Punkt enthält, wird überprüft, ob ihr Richtungsvektor orthogonal zur Kontaktnormale ist. Ist dies der

Fall, so ist die Kante das gesuchte nächste Element des Körpers. Wenn keine Kante die Bedingung für das nächste Element erfüllt, dann muss der nächste Punkt selbst das nächste Element des Körpers sein.

Wenn für zwei kollidierende Körper die nächsten Elemente ermittelt wurden und ein Element ein Punkt ist, dann existieren keine weiteren Kontaktpunkte zwischen den Körpern. Handelt es sich bei den Elementen um zwei Flächen, zwei Kanten oder eine Fläche und eine Kante, dann müssen die Elemente auf weitere Kontaktpunkte untersucht werden.

Im dem Fall, dass die beiden nächsten Elemente Flächen sind, werden ihre konvexen Polygone in die Ebene der ersten Fläche projiziert. In dieser Ebene werden die beiden Polygone miteinander geschnitten. Der Schnitt von zwei konvexen Polygonen kann mit einem Zeitaufwand durchgeführt werden, der linear von der Anzahl der Kanten abhängt. Durch den Schnitt kann die Kontaktregion in der Ebene bestimmt werden. Das daraus resultierende Schnittpolygon wird anschließend zurück in den dreidimensionalen Raum auf die beiden Körper projiziert. Dadurch ergeben sich die zusätzlichen Kontaktpunkte der Körper. Für jedes Paar der neuen Kontaktpunkte muss zum Schluss noch überprüft werden, ob ihr Abstand innerhalb der vorgegebenen Toleranz der Kollisionserkennung liegt. Wenn dies für ein Paar nicht der Fall ist, wird es verworfen.

Handelt es sich bei den beiden nächsten Elementen um Kanten, dann muss zunächst überprüft werden, ob sie parallel zueinander sind. Andernfalls gibt es keine weiteren Kontaktpunkte. Sind die beiden Kanten parallel, dann werden sie auf die Gerade der ersten Kante projiziert. Auf dieser Gerade werden dann die Kanten miteinander geschnitten. Im eindimensionalen Raum der Geraden muss dabei nur die Überlappung der Intervalle, die durch die Endpunkte der Kanten definiert sind, bestimmt werden. Die Endpunkte des Schnittintervalls sind potentielle Kontaktpunkte auf der ersten Kante. Auf der zweiten Kante werden die zugehörigen nächsten Punkte ermittelt, um Kontaktpunktpaare zu erhalten. Am Ende wird überprüft, ob der Abstand von jedem Paar innerhalb des Toleranzbereichs der Kollisionserkennung liegt.

Wenn die beiden nächsten Elemente eine Fläche und eine dazu parallele Kante sind, dann werden die Elemente in die Ebene der Fläche projiziert. In diesem zweidimensionalen Raum wird die Kante mit dem konvexen Polygon der Fläche geschnitten. Daraus ergibt sich eine neue Kante. Die Endpunkte dieser Kante sind potentielle Kontaktpunkte. Diese werden zurück auf die beiden Körper projiziert, um die gesuchten Kontaktpunktpaare zu bestimmen. Der Abstand der Kontaktpunkte muss anschließend noch überprüft werden, um sicherzustellen, dass er im vorgegebenen Toleranzbereich liegt.

4.6 Ergebnisse

In diesem Abschnitt wird die Geschwindigkeit der beiden erweiterten Verfahren untersucht. Die Messungen wurden auf einem PC mit einem 3,4 GHz Intel Pentium 4 Prozessor vorgenommen. Das Modell, das für die Geschwindigkeitsmessungen verwendet wurde, besteht aus 1000 Würfeln, die auf den Boden fallen (siehe Abbildung 4.6). Die Simulation dieses Modells wurde mit den beiden vorgestell-

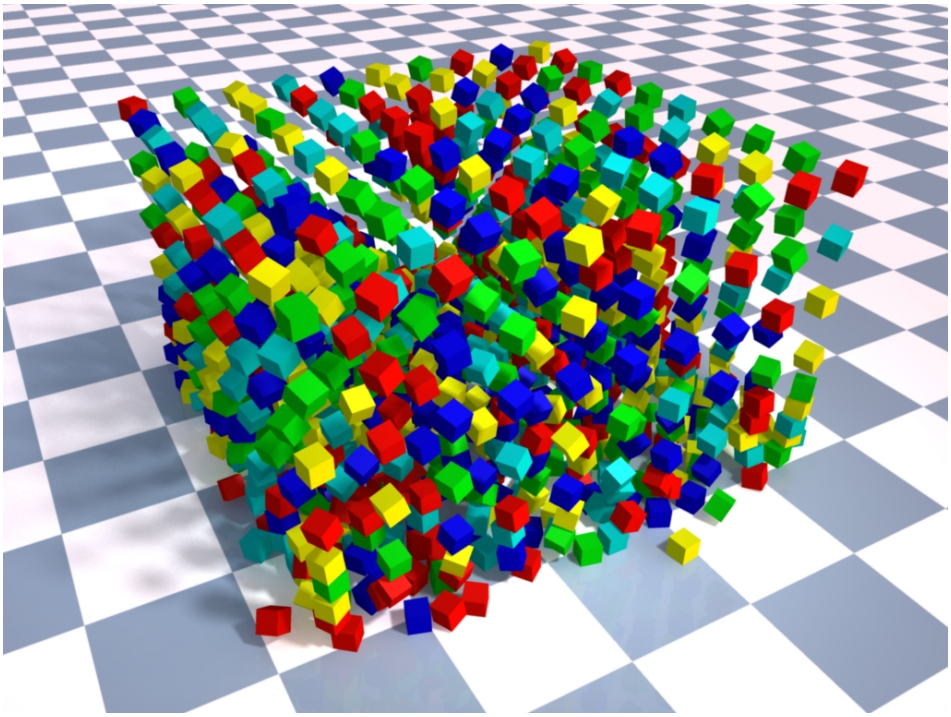


Abbildung 4.6: Kollisionserkennung bei 1000 Würfeln, die auf den Boden fallen

ten Verfahren zur Kollisionserkennung durchgeführt. Für jedes Paar von Körpern wurde mit den ersten beiden Phasen der Kollisionserkennung festgestellt, ob eine Kollision vorliegt. Im Fall einer Kollision wurden in der dritten Phase alle Kontaktpunkte ermittelt. In jedem Simulationsschritt wurde die Anzahl aller Kollisionen bestimmt. Für jedes Paar von kollidierenden Körpern wurde die Anzahl ihrer Kontaktpunkte ermittelt. Außerdem wurde die Zeit gemessen, die für alle drei Phasen benötigt wurde.

Abbildung 4.7 zeigt die Rechenzeiten der beiden Verfahren im Verlauf der Simulation. Am Anfang fallen die Körper und es treten keine Kollisionen auf. Nachdem die ersten Körper den Boden erreicht haben, ergeben sich die ersten Kollisionen. Die Anzahl der Kollisionen steigert sich bis zum Ende der Simulation. Zu diesem

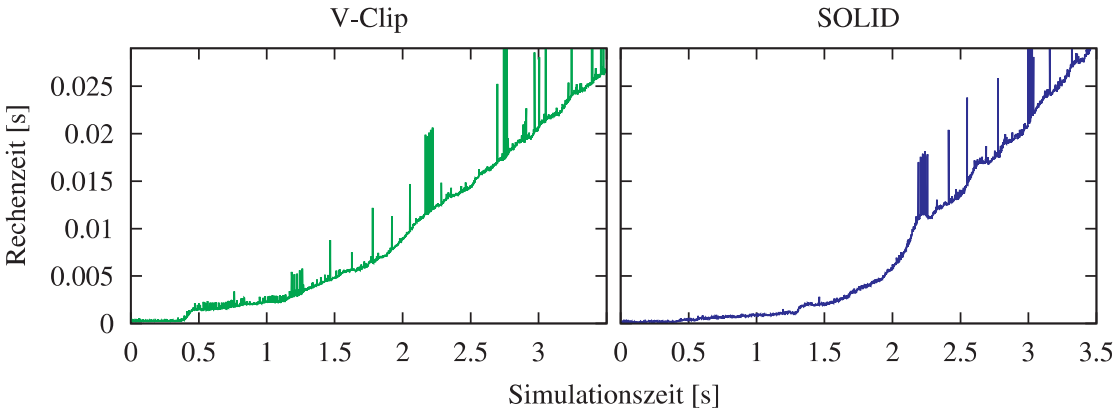


Abbildung 4.7: Rechenzeiten der Kollisionserkennung bei 1000 Würfeln. Die Kurven zeigen den Verlauf der Rechenzeiten während der Simulation mit erweitertem V-Clip bzw. erweitertem SOLID.

Zeitpunkt liegen alle Würfel auf einem Haufen auf dem Boden. Dadurch gibt es sehr viele Kontaktpunkte zwischen den Körpern.

In Abbildung 4.8 ist die Rechenzeit in Abhängigkeit von der Anzahl der Kollisionen bzw. der Kontaktpunkte von dem Verfahren dargestellt, das auf V-Clip basiert. Die Rechenzeit steigt annähernd linear zu der Anzahl der Kollisionen im Modell.

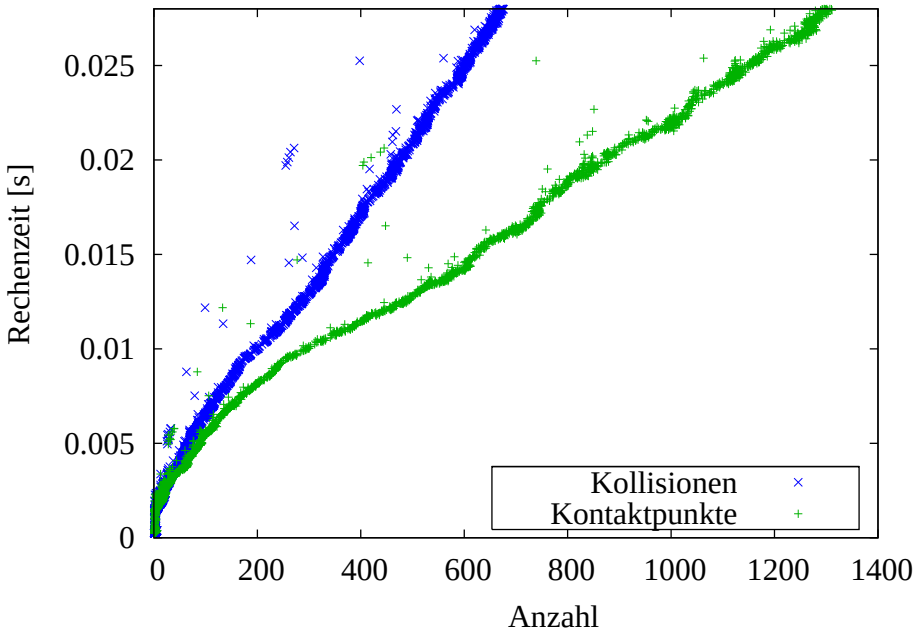


Abbildung 4.8: Kollisionserkennung beim Würfelmodell mit erweitertem V-Clip. Das Diagramm zeigt die benötigte Rechenzeit für alle drei Phasen in Abhängigkeit von der Anzahl der Kollisionen bzw. Kontaktpunkte.

Der Grund dafür ist, dass durch das Ausnutzen zeitlicher und geometrischer Kohärenz, der Erwartungswert der Laufzeit für die beiden ersten Phasen linear ist. Der Aufwand der dritten Phase hängt von der Anzahl der Eckpunkte und Kanten der Polyederflächen ab, die auf weitere Kontaktpunkte überprüft werden müssen. Da diese Anzahl im vorliegenden Modell für jede Fläche gleich ist, ergibt sich insgesamt für die letzte Phase ein Aufwand, der linear zu der Anzahl der Kollisionen ist.

In Abbildung 4.9 sind die Ergebnisse mit dem erweiterten Verfahren, das auf SOLID basiert, dargestellt. Die Ergebnisse zeigen, dass die benötigte Rechenzeit eben-

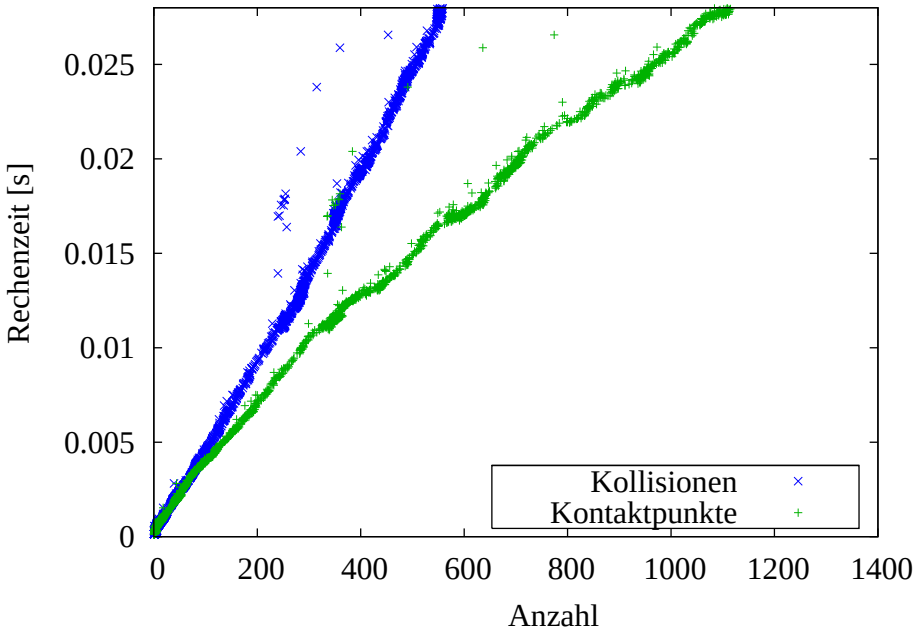


Abbildung 4.9: Kollisionserkennung beim Würfelmodell mit erweitertem SOLID. Das Diagramm zeigt die benötigte Rechenzeit für alle drei Phasen in Abhängigkeit von der Anzahl der Kollisionen bzw. Kontaktpunkte.

falls annähernd linear von der Anzahl der Kollisionen abhängt. Das liegt daran, dass alle drei Phasen dieses Verfahrens einen erwarteten Zeitaufwand haben, der linear ist.

Die beiden vorgestellten Verfahren sind bei der gleichen Anzahl an Kollisionen beinahe gleich schnell. Das auf SOLID basierende Verfahren war bei den Messungen geringfügig schneller, wenn in einem Zeitschritt weniger als 270 Kollisionen aufgetreten sind. Bei mehr Kollisionen war das Verfahren mit V-Clip schneller. Mit diesem Verfahren konnte eine einzige Kollision im Schnitt in 0,047 ms erkannt werden, während mit SOLID 0,049 ms gebraucht wurden. Beide Verfahren haben zur Ermittlung von 500 Kollisionen mit fast 1000 Kontaktpunkten weniger als 25 ms benötigt. Für die große Anzahl an Kollisionen ist dies relativ schnell. Da bei

der Simulation der meisten Modelle weit weniger Kollisionen auftreten, ist diese Geschwindigkeit ausreichend für die Dynamiksimulation in dieser Arbeit.

Kapitel 5

Kollisionsauflösung

Ein wichtiger Teil der dynamischen Simulation ist die Behandlung von Situationen, in denen zwei Starrkörper kollidieren oder permanent Kontakt miteinander haben. Dabei muss verhindert werden, dass sich die Körper gegenseitig durchdringen. In Abschnitt 2.3 wurden verschiedene Verfahren für die Kollisionsauflösung vorgestellt. Die impulsbasierten Verfahren haben den Vorteil, dass sie Kollisionen mit einem geringen Berechnungsaufwand auflösen können, da Impulse einfach zu bestimmen sind. Allerdings lösen sie eine Kollision in nur einem einzigen Kontaktpunkt auf, auch wenn mehrere Kontaktpunkte zwischen den Körpern existieren. Dies führt zu Ungenauigkeiten in der Simulation. Die Kollisionsauflösung mit Zwangsbedingungen verwendet einen anderen Ansatz. Für jeden Kontaktpunkt wird eine Bedingung definiert, die erfüllt sein muss, um eine Durchdringung der Körper in diesem Punkt zu verhindern. Die Bedingungen werden von den meisten Verfahren durch die Bestimmung von Zwangskräften erfüllt. Diese Vorgehensweise ermöglicht sehr genaue Ergebnisse, aber der Aufwand bei der Berechnung der Zwangskräfte ist hoch.

Eine Kollision zwischen zwei Körpern kann nur dann exakt aufgelöst werden, wenn die Auflösung in allen Kontaktpunkten gleichzeitig geschieht. Aus diesem Grund werden in dieser Arbeit für alle Kontaktpunkte Zwangsbedingungen aufgestellt. Es wird allerdings kein Optimierungsproblem gelöst, um die Zwangsbedingungen zu erfüllen, wie es bei vielen Verfahren der Fall ist. Stattdessen werden in einem iterativen Prozess Impulse bestimmt. Diese Impulse sorgen dafür, dass sich die Geschwindigkeiten der kollidierenden Körper ändern und sie dadurch die Bedingungen in den Kontaktpunkten erfüllen. Die Verwendung von Impulsen hat den Vorteil, dass Impulse leicht bestimmt werden können und daher die Berechnungen sehr schnell sind. Das in dieser Arbeit vorgestellte Verfahren kombiniert damit die Vorteile der Kollisionsauflösung mit Zwangsbedingungen und der impulsbasierten Kollisionsauflösung.

Im nächsten Abschnitt werden die Grundlagen vorgestellt, die für die Kollisions-

auflösung benötigt werden. Anschließend wird gezeigt, wie Kollisionen mit Hilfe von Impulsen behandelt werden. Bei der Auflösung einer Kollision wird die Reibung zwischen den Körpern ebenfalls mit Impulsen simuliert. Die Behandlung von bleibenden Kontakten mit Reibung wird in Abschnitt 5.3 näher erläutert. Im darauffolgenden Abschnitt wird gezeigt, wie Zwangsbedingungen von Gelenken bei der Auflösung von Kollisionen und Kontakten berücksichtigt werden. Am Ende des Kapitels werden Ergebnisse präsentiert, die mit dem vorgestellten Verfahren erzielt wurden.

5.1 Grundlagen

Die Kollisionserkennung (siehe Kapitel 4) bestimmt den Zeitpunkt, an dem sich zwei Körper berühren, und alle Kontaktpunkte zwischen den Körpern. Die Kollisionsauflösung hat die folgenden Aufgaben. Zunächst muss für jeden Kontaktpunkt festgestellt werden, ob die beiden zugehörigen Körper in diesem Punkt kollidieren, einen bleibenden Kontakt haben oder sich auseinander bewegen. Eine Kollision liegt vor, wenn sich die beiden Körper in einem Kontaktpunkt aufeinander zu bewegen. In diesem Fall wird abhängig von der Stärke des Aufpralls ein Rückstoß simuliert. Dabei muss die Reibung, die zwischen den Körpern auftritt, berücksichtigt werden. Bewegen sich die Körper in einem Kontaktpunkt auseinander, dann besteht keine Gefahr einer Durchdringung und die Kollisionsauflösung muss nicht eingreifen. Wenn sich zwei Körper in einem Kontaktpunkt weder aufeinander zu noch auseinander bewegen, dann bleiben die beiden Körper in diesem Punkt in Kontakt. Bei einem bleibenden Kontakt muss die Kollisionsauflösung dafür sorgen, dass eine Durchdringung der Körper verhindert wird. Außerdem muss die Reibung zwischen den Körpern simuliert werden.

5.1.1 Fallunterscheidung

Angenommen zwei Körper berühren sich in einem Punkt. Sei $\mathbf{a}$ dieser Kontaktpunkt im ersten Körper und $\mathbf{b}$ der Punkt im zweiten Körper (siehe Abbildung 5.1). Die Geschwindigkeiten dieser Punkte $\mathbf{u}_a$ und $\mathbf{u}_b$ können mit Gleichung 3.8 berechnet werden. Die relative Geschwindigkeit der Kontaktpunkte ist durch die Gleichung

$$\mathbf{u}_{rel} = \mathbf{u}_a - \mathbf{u}_b$$

bestimmt. Die relative Geschwindigkeit in Richtung der Kontaktnormalen

$$u'_{rel,n} = \mathbf{u}_{rel} \cdot \mathbf{n}$$

wird benötigt, um die oben erwähnte Fallunterscheidung durchzuführen. Angenommen die einzige externe Kraft, die auf einen Körper wirkt, sei Gravitation.

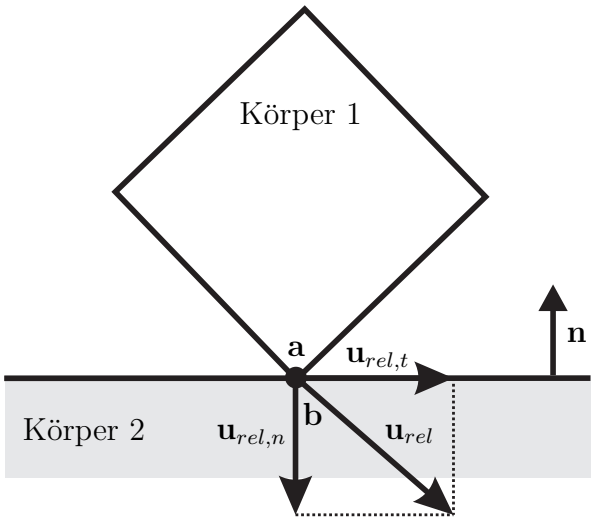


Abbildung 5.1: Relative Punktgeschwindigkeiten bei einer Kollision

Der Körper erreicht aus dem Ruhezustand genau die Geschwindigkeit

$$\varepsilon_{rc} = \sqrt{2\,|\mathbf{g}|\,\varepsilon_c},$$

wenn er die Distanz ε_c des Toleranzbereichs der Kollisionserkennung zurücklegt. Der Vektor $\mathbf{g}$ gibt dabei die Beschleunigung aufgrund der Gravitation an. Die Geschwindigkeit ε_{rc} wird bei der Fallunterscheidung als Toleranz verwendet. Dieser Toleranzwert wurde von Brian V. Mirtich in [Mir96b] vorgeschlagen. Die Bedingung für einen bleibenden Kontakt ist damit

$$-\varepsilon_{rc} < u'_{rel,n} < \varepsilon_{rc}. \tag{5.1}$$

Wenn der Wert $u'_{rel,n}$ kleiner gleich $-\varepsilon_{rc}$ ist, dann handelt es sich um eine Kollision. Andernfalls bewegen sich die Körper auseinander.

5.1.2 Zeitschritt mit Kollisionsauflösung

In einem Zeitschritt muss die Kollisionserkennung als erstes alle Kontaktpunkte und -normalen bestimmen (siehe Abbildung 5.2). Diese werden bei der Auflösung der Kollisionen und bei der Behandlung der bleibenden Kontakte benötigt. Die Kollision von zwei Starrkörpern findet in einem unendlich kleinen Zeitraum statt. Deshalb müssen sich die Geschwindigkeiten der Körper zum Zeitpunkt der Kollision ohne eine Zeitverzögerung ändern, um eine Durchdringung zu verhindern. Dies kann mit einem Impuls bewirkt werden. Da während der Kollision keine Zeit vergeht, haben externe Kräfte und Kräfte, die zwischen den Gelenken wirken, keinen Einfluss auf die Kollision und müssen nicht berücksichtigt werden. Aus diesem

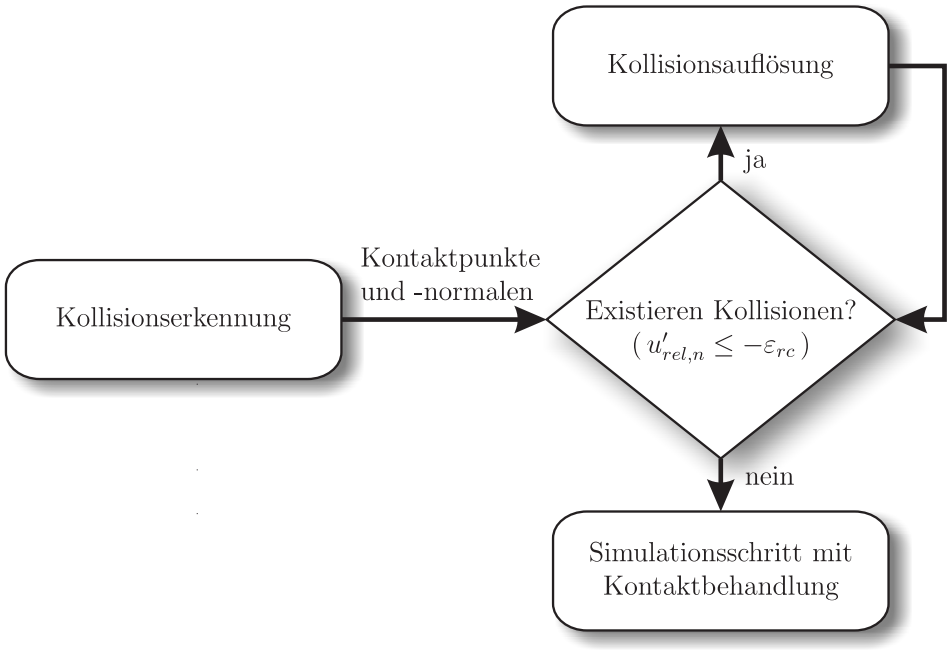


Abbildung 5.2: Ablauf eines Zeitschrittes mit Kollisions- und Kontaktbehandlung

Grund werden Kollisionen vor dem Simulationsschritt, in dem die Gelenkkorrektur stattfindet, aufgelöst.

Bei der Auflösung werden zunächst alle Kontaktpunkte bestimmt, deren relative Geschwindigkeiten die Bedingung für eine Kollision erfüllen. Für diese Kontaktpunkte werden anschließend in einer iterativen Schleife Impulse berechnet, die für jede Kollision einen Rückstoß simulieren. Die angewendeten Impulse können Auswirkungen auf die Geschwindigkeiten von anderen Kontaktpunkten haben. Dadurch können neue Kollisionen entstehen. Es muss erneut überprüft werden, ob Kontakte existieren, die die Bedingung für eine Kollision erfüllen. Die Auflösung der Kollisionen wird daher in einer Schleife durchgeführt. Diese Schleife endet, wenn es keinen Kontaktpunkt in der Szene gibt, der die Bedingung für eine Kollision erfüllt. Nachdem alle Kollisionen aufgelöst sind, erfolgt ein Simulationsschritt mit den geänderten Geschwindigkeiten.

Im Fall eines bleibenden Kontakts bewegen sich die Körper für einen gewissen Zeitraum nicht auseinander. In diesem Zeitraum muss die Kollisionsauflösung eine Durchdringung der Körper verhindern. Die externen und internen Kräfte, die in dem Zeitintervall wirken, beeinflussen die Geschwindigkeiten der Körper in den Kontaktpunkten. Aus diesem Grund werden bleibende Kontakte während des Simulationsschrittes parallel zur Gelenkkorrektur behandelt. Dadurch werden die Wechselwirkungen zwischen den Impulsen der Kontaktbehandlung und den internen und externen Kräften berücksichtigt.

5.1.3 Simulation einer Kollision

Bei der Auflösung einer Kollision zwischen den Körpern k_1 und k_2 muss ein Impuls in Richtung der Kontaktnormalen berechnet werden, der die Körper in den Kontaktpunkten abhängig von einem Elastizitätskoeffizienten e auseinander beschleunigt. Damit wird ein Rückstoß nach der Kollision simuliert. Es gibt verschiedene Ansätze, um diesen Kollisionsimpuls zu bestimmen:

- *Newtons Stoßgesetz* besagt, dass die relative Geschwindigkeit der Kontaktpunkte in Normalenrichtung nach der Kollision $\mathbf{u}^c_{rel,n}$ von der Geschwindigkeit $\mathbf{u}_{rel,n}$ abhängt, mit der die beiden Körper kollidiert sind, und von dem Elastizitätskoeffizienten e der Körper:

$$\mathbf{u}^c_{rel,n} = -e \cdot \mathbf{u}_{rel,n}. \tag{5.2}$$

- Die *Hypothese von Poisson* beschreibt das Verhältnis zwischen dem Impuls $\mathbf{p}_n$, der die relative Geschwindigkeit in den Kontaktpunkten in Normalenrichtung $\mathbf{u}_{rel,n}$ eliminiert, und dem Gesamtimpuls der Kollision in Normalenrichtung $\mathbf{p}^c_n$:

$$\mathbf{p}^c_n = (1 + e) \cdot \mathbf{p}_n. \tag{5.3}$$

- Eine weitere Alternative für die Berechnung des Kollisionsimpulses bietet *Stronges Hypothese*. Eine Kollision wird dabei in eine Kompressions- und eine Rückstoßphase eingeteilt. Die Arbeit W_n, die der Kollisionsimpuls in Normalenrichtung während der Kompressionsphase leistet, wird dann in Relation zu der Arbeit in der Rückstoßphase gesetzt:

$$W^c_n = (1 - e^2) \cdot W_n.$$

Jedem Körper in der Simulation ist ein Elastizitätskoeffizient zugewiesen. Bei der Kollision von zwei Körpern müssen die beiden zugehörigen Koeffizienten kombiniert werden. In [MW88b] und [GBF03] wird das Minimum der beiden Werte verwendet. Dies hat allerdings zur Folge, dass die Auflösung einer Kollision immer nur von dem Koeffizienten eines Körpers abhängt. Wenn zum Beispiel verschiedene Körper auf einen Boden mit wenig Elastizität fallen, dann würden vollkommen elastische und kaum elastische Körper gleich behandelt. Daher wird in dieser Arbeit ein anderer Ansatz verwendet. Die Koeffizienten der beiden Körper werden durch eine Multiplikation miteinander verknüpft. Auf diese Weise werden beide Koeffizienten berücksichtigt.

5.1.4 Reibung

Die Reibung bei einer Kollision oder einem bleibenden Kontakt wird mit Hilfe des Reibungsgesetzes von Coulomb simuliert. Dabei wird zwischen statischer und

dynamischer Reibung[1] unterschieden. Die Art der Reibung ist abhängig von der Geschwindigkeit der Kontaktpunkte in der Tangentialebene, die senkrecht zur Kontaktnormale steht. Angenommen zwei Körper k_1 und k_2 haben Kontakt in einem Punkt und $\mathbf{u}_{rel}$ sei die relative Geschwindigkeit der Kontaktpunkte. Dann ist die relative Geschwindigkeit der Punkte in Normalenrichtung durch $\mathbf{u}_{rel,n} = u'_{rel,n} \cdot \mathbf{n}$ bestimmt. In der Tangentialebene haben die beiden Kontaktpunkte die relative Geschwindigkeit $\mathbf{u}_{rel,t} = \mathbf{u}_{rel} - \mathbf{u}_{rel,n}$ (siehe Abbildung 5.1). Das Reibungsgesetz von Coulomb sagt für die Kräfte $\mathbf{F}_n$ in Normalenrichtung und $\mathbf{F}_t$ in Tangentialrichtung, die der Körper k_2 auf den Körper k_1 ausübt, das Folgende aus:

$$\begin{aligned} |\mathbf{u}_{rel,t}| \neq 0 \quad &\Rightarrow \quad \mathbf{F}_t = -\mu_d \cdot |\mathbf{F}_n| \cdot \hat{\mathbf{u}}_{rel,t} \\ |\mathbf{u}_{rel,t}| = 0 \quad &\Rightarrow \quad |\mathbf{F}_t| \leq \mu_s \cdot |\mathbf{F}_n|. \end{aligned}$$

Dabei ist $\hat{\mathbf{u}}_{rel,t}$ der Einheitsvektor von $\mathbf{u}_{rel,t}$, der die Richtung der Bewegung angibt, und die Werte μ_d und μ_s sind die Koeffizienten für die dynamische und die statische Reibung. Wenn sich die beiden Körper in der Tangentialebene relativ zueinander bewegen, dann tritt dynamische Reibung auf und es wirkt eine Reibungskraft entgegen der Bewegungsrichtung. Die Stärke dieser Reibungskraft hängt ausschließlich von der Kraft in Normalenrichtung und von dem Reibungskoeffizienten μ_d ab. Bei statischer Reibung ist die Geschwindigkeit $\mathbf{u}_{rel,t}$ Null und die Tangentialkraft $\mathbf{F}_t$ liegt in einem sogenannten Reibungskegel. Die statische Reibung wird überwunden, wenn sich die Tangentialkraft aus dem Reibungskegel heraus bewegt. In diesem Fall fangen die Körper an, sich relativ zueinander zu bewegen, und dynamische Reibung löst die statische Reibung ab.

Jeder Körper hat eigene Reibungskoeffizienten für die dynamische und die statische Reibung. Bei der Auflösung einer Kollision oder eines bleibenden Kontaktes müssen die Werte der beiden beteiligten Körper zu einem Wert kombiniert werden. In [MW88b] und [GBF03] wird nur der größere der beiden Werte verwendet. Dies bedeutet aber, dass sich ein Gummireifen auf Eis genauso verhalten würde wie auf einer Straße. In dieser Arbeit wird daher die Summe der beiden Koeffizienten verwendet, so dass beide Werte bei der Simulation berücksichtigt werden.

5.2 Auflösung von Kollisionen

In diesem Abschnitt wird die Auflösung von Kollisionen mit Reibung beschrieben. Die Berechnung der Kollisionsimpulse wird zunächst auf der Grundlage von Newtons Stoßgesetz durchgeführt. Anschließend wird erklärt, wie die Kollisionsauflösung mit der Hypothese von Poisson funktioniert. Durch einen Kontaktgraph können die Abhängigkeiten der Kollisionen beschrieben werden. Die Methode der

[1] Für dynamische und statische Reibung werden auch die Begriffe Gleit- und Haftreibung verwendet.

Schockfortpflanzung verwendet diesen Graph, um die Auflösung der Kollisionen zu beschleunigen. Am Ende des Abschnitts wird die Berechnung von Reibungsimpulsen nach dem Gesetz von Coulomb erläutert.

5.2.1 Auflösung nach Newton

Angenommen zwei Körper kollidieren in einem Punkt und **a** und **b** seien die zugehörigen Kontaktpunkte der beiden Körper. Nach Newtons Stoßgesetz kann die relative Geschwindigkeit der Kontaktpunkte in Normalenrichtung nach der Kollision $\mathbf{u}^c_{rel,n}$ mit Gleichung 5.2 bestimmt werden. Die beiden Kontaktpunkte müssen daher ihre relative Geschwindigkeit um den Wert

$$\triangle \mathbf{u}_{rel,n} = \mathbf{u}^c_{rel,n} - \mathbf{u}_{rel,n} \tag{5.4}$$

ändern, damit die Kollision aufgelöst wird. Diese Geschwindigkeitsänderung kann mit einem Impuls $\mathbf{p}_n$ in Richtung der Kontaktnormalen **n** erreicht werden. Damit die Impulserhaltung des Systems gewährleistet wird, muss der Impuls $\mathbf{p}_n$ auf den ersten Körper und der entgegengesetzte Impuls $-\mathbf{p}_n$ auf den zweiten Körper angewendet werden. Der Impuls in Normalenrichtung, der die relative Geschwindigkeit der Kontaktpunkte **a** und **b** um $\triangle \mathbf{u}_{rel,n}$ verändert, wird durch die Gleichung

$$\mathbf{p}_n = \frac{1}{\mathbf{n}^T \mathbf{K} \mathbf{n}} \triangle \mathbf{u}_{rel,n} \tag{5.5}$$

bestimmt, wobei $\mathbf{K} := \mathbf{K}_{a,a} + \mathbf{K}_{b,b}$ gilt (siehe auch Abschnitt 3.2.2). Der resultierende Impuls $\mathbf{p}_n$ wird positiv auf den Punkt **a** und negativ auf den Punkt **b** angewendet. Dadurch wird die Kollision aufgelöst und die Kontaktpunkte erreichen die relative Geschwindigkeit $\mathbf{u}^c_{rel,n}$ in Normalenrichtung.

Wenn ein Körper mit mehreren anderen Körpern kollidiert oder zwei Körper eine Kollision mit mehreren Kontaktpunkten haben, dann reicht es nicht aus, für jedes Paar von Kontaktpunkten je einen Impuls zu bestimmen. Ein Impuls, der für die Auflösung einer Kollision berechnet und angewendet wird, beeinflusst die Geschwindigkeiten in allen Kontaktpunkten der beiden kollidierenden Körper. Um die Abhängigkeiten zwischen den Kontaktpunkten zu berücksichtigen, wird die Berechnung der Kollisionsimpulse in einer iterativen Schleife durchgeführt, bis alle Kollisionen korrekt aufgelöst sind.

In einer Szene mit mehreren Kollisionen wird zunächst für jedes Paar von Kontaktpunkten der Wert $\mathbf{u}^c_{rel,n}$ mit Newtons Stoßgesetz berechnet. Dann werden die Kollisionsimpulse in einer Schleife bestimmt. In der i-ten Iteration wird für ein Paar von Kontaktpunkten der aktuelle Wert $\mathbf{u}_{rel,n}$ ermittelt. Anschließend wird überprüft, ob dieser Wert den Zielwert $\mathbf{u}^c_{rel,n}$ bereits erreicht hat. Ist dies der Fall, so kann mit den nächsten Kontaktpunkten fortgefahren werden. Andernfalls muss ein Impuls $\mathbf{p}_{n,i}$ berechnet werden, der die relative Geschwindigkeit um die aktuelle

Differenz $\triangle \mathbf{u}_{rel,n}$ verändert. Der gesuchte Impuls $\mathbf{p}_{n,i}$ wird mit der Gleichung 5.5 bestimmt. Es muss beachtet werden, dass der Gesamtimpuls, der auf ein Paar von Kontaktpunkten angewendet wird, die Körper in den Punkten voneinander abstoßen muss. Der Impuls darf die Körper nicht festhalten. Angenommen die relative Geschwindigkeit $\mathbf{u}^c_{rel,n}$ wird für zwei Kontaktpunkte aufgrund von Kollisionsimpulsen in anderen Kontaktpunkten überschritten. Dann muss der Gesamtimpuls für die zwei Kontaktpunkte Null sein, andernfalls werden die beiden Körper in den Kontaktpunkten zusammengehalten. In jeder Iteration muss die Summe der Kollisionsimpulse in Richtung der Kontaktnormalen $\mathbf{n}$ zeigen, um ein Festhalten der Körper zu verhindern. Dies bedeutet, dass in der i-ten Iteration die Bedingung

$$\mathbf{n} \cdot \sum_{j=1}^{i} \mathbf{p}_{n,j} \geq 0 \tag{5.6}$$

für die Kollisionsimpulse erfüllt sein muss. Wenn diese Bedingung in der Iteration $i-1$ für zwei Kontaktpunkte erfüllt ist, aber in der i-ten Iteration nicht mehr, dann gilt

$$-\mathbf{n} \cdot \mathbf{p}_{n,i} > \mathbf{n} \cdot \sum_{j=1}^{i-1} \mathbf{p}_{n,j}.$$

Daher darf maximal der Impuls $-\sum_{j=1}^{i-1} \mathbf{p}_{n,j}$ angewendet werden, um die Bedingung 5.6 zu erfüllen. Daraus resultiert, dass für ein Paar von Kontaktpunkten in der i-ten Iteration der Impuls

$$\mathbf{p}'_{n,i} = \begin{cases} \mathbf{p}_{n,i} & \text{falls } \mathbf{n} \cdot \sum_{j=1}^{i} \mathbf{p}_{n,j} \geq 0 \\ -\sum_{j=1}^{i-1} \mathbf{p}_{n,j} & \text{sonst} \end{cases} \tag{5.7}$$

angewendet werden muss, um die Kollision aufzulösen. Dabei wird $\mathbf{p}_{n,i}$ aus dem aktuellen Wert $\triangle \mathbf{u}_{rel,n}$ der Kontaktpunkte mit der Gleichung 5.5 berechnet. Die Iterationsschleife endet, wenn für jedes Paar von Kontaktpunkten entweder die zugehörige Geschwindigkeit $\mathbf{u}^c_{rel,n}$ innerhalb einer Toleranz $\varepsilon_{v,c}$ erreicht wurde oder der Gesamtimpuls Null ist und $\mathbf{n} \cdot \mathbf{u}_{rel,n} > \mathbf{n} \cdot \mathbf{u}^c_{rel,n}$ gilt.

5.2.2 Auflösung nach Poisson

Die Hypothese von Poisson beschreibt, wie der Gesamtimpuls einer Kollision $\mathbf{p}^c_n$ von dem Impuls $\mathbf{p}_n$ abhängt, der die relative Geschwindigkeit der Kontaktpunkte in Normalenrichtung zu Null macht. Das Verfahren zur Kollisionsauflösung mit Newtons Stoßgesetz (siehe Abschnitt 5.2.1) kann verwendet werden, um den Impuls $\mathbf{p}_n$ zu bestimmen. Der Zielwert der relativen Geschwindigkeit der Kontaktpunkte muss dabei auf Null gesetzt werden anstatt auf $\mathbf{u}^c_{rel,n}$. Deswegen wird bei der Berechnung die Gleichung 5.4 durch

$$\triangle \mathbf{u}_{rel,n} = -\mathbf{u}_{rel,n}$$

ersetzt. Nachdem die iterative Schleife des Verfahrens beendet ist und für jedes Paar von Kontaktpunkten ein Impuls $\mathbf{p}_n$ ermittelt wurde, wird die Gleichung 5.3 von Poissons Hypothese angewendet. Mit dieser Gleichung kann für jeden Kontakt der Gesamtimpuls $\mathbf{p}_n^c$ bestimmt werden. Allerdings wurde während der iterativen Berechnung in jedem Kontakt bereits der Impuls $\mathbf{p}_n$ angewendet. Daher wird nur noch der Impuls $e \cdot \mathbf{p}_n$ benötigt, um einen Rückstoß zu simulieren und die Kollision nach der Hypothese von Poisson aufzulösen.

5.2.3 Kontaktgraph

In einer Simulation mit vielen Kollisionen treten oft Abhängigkeiten zwischen den Impulsen auf, die bei der Auflösung berechnet werden. Die Kollisionen sind direkt voneinander abhängig, wenn sie mindestens einen gemeinsamen Körper haben. Eine indirekte Abhängigkeit ergibt sich bei einer Kette von Körpern, die Kontakt miteinander haben, so dass ein Kollisionsimpuls Auswirkungen auf alle Körper in der Kette hat (siehe Abbildung 5.3). Die Abhängigkeiten der Kollisionen können

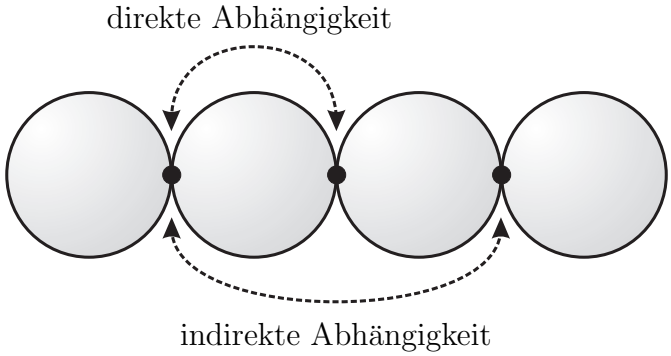


Abbildung 5.3: Abhängigkeiten zwischen einzelnen Kollisionen

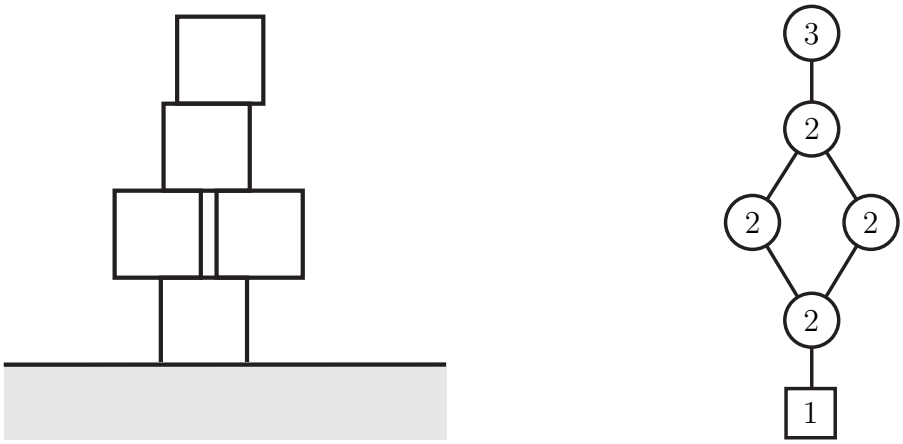
mit Hilfe eines Kontaktgraphen beschrieben werden [Hah88, BW00].

Jeder Körper, der Kontakt mit einem anderen Körper hat, wird im Kontaktgraph durch einen Knoten repräsentiert. Für je zwei Körper, die miteinander Kontakt haben, existiert eine Kante zwischen den zugehörigen Knoten im Graph. Der Wurzelknoten des Kontaktgraphen repräsentiert alle statischen Körper in dem simulierten Modell, die einen Kontakt haben. Wenn keine statischen Körper vorhanden sind, dann wird ein beliebiger Körper als Wurzelknoten verwendet. Um den Graph aufzubauen, werden alle Kanten, die von dem Wurzelknoten ausgehen, abgelaufen und markiert. Die Knoten, die dadurch erreicht werden, werden markiert und in einer Liste gespeichert. Für jeden Knoten in der Liste wird die gleiche Prozedur durchgeführt wie für den Wurzelknoten, wobei nur unmarkierte Kanten abgelaufen werden. Anschließend wird der abgearbeitete Knoten aus der Liste gelöscht. Wenn die Liste leer ist, aber noch Knoten existieren, die nicht markiert wurden,

dann existiert ein weiterer unabhängiger Graph. Ein beliebiger, nicht markierter Knoten dient als Wurzel. Ausgehend von diesem Knoten wird der zweite Graph abgelaufen. Auf diese Weise werden alle Kontakte abgearbeitet und eine Hierarchie aufgestellt.

Die Kollisionsauflösung benötigt im Allgemeinen viele Iterationsschritte, wenn viele Kollisionen voneinander abhängig sind. Durch den Kontaktgraphen ist die Struktur dieser Abhängigkeiten bekannt. Die Reihenfolge der Kollisionsauflösung orientiert sich daher an dem Kontaktgraph. Die Auflösung erfolgt ausgehend vom Wurzelknoten. Es werden immer alle Kollisionen in einer Ebene des Graphen aufgelöst. Anschließend wird mit der nächsten Ebene fortgefahren. Das heißt, die Auflösung beginnt bei den statischen Körpern, da diese nicht beweglich sind, und arbeitet sich dann Ebene für Ebene durch die gestapelten Körper.

Wenn im Kontaktgraph Zyklen existieren, dann sind die zugehörigen Kollisionen abhängig voneinander und müssen gemeinsam aufgelöst werden. Daher werden alle Knoten und Kanten eines Zyklus zusammengefasst und der gleichen Ebene des Graphen zugeordnet. Dies ist wichtig für die Methode der Schockfortpflanzung, die im nächsten Abschnitt vorgestellt wird. Abbildung 5.4 zeigt einen Stapel von mehreren Würfeln und den dazugehörigen Kontaktgraph. Die Nummern in den Knoten des Graphen geben an, welcher Ebene ein Knoten zugeordnet wurde.



(a) Würfelstapel (b) Kontaktgraph

Abbildung 5.4: Das linke Bild zeigt einen Stapel von Würfeln. Der zugehörige Kontaktgraph ist im rechten Bild dargestellt.

5.2.4 Schockfortpflanzung

Die Kollisionsauflösung bei gestapelten Körpern stellt ein besonderes Problem in der Dynamiksimulation dar. Wenn der Kontaktgraph sehr tief ist, dann werden im

Allgemeinen aufgrund der Abhängigkeiten der Impulse sehr viele Iterationen zur Auflösung benötigt. Liegt der Stapel auf einem statischen Körper, dann benötigt die Auflösung noch mehr Iterationen, da alle Körper im Stapel auf den statischen Körper drücken, dieser sich aber nicht entfernen kann. Eran Guendelman stellt in seiner Arbeit [Gue06] eine Methode der Schockfortpflanzung für die Behandlung von Kontakten vor. Mit dieser Methode kann die Anzahl der Iterationen auf Kosten der Genauigkeit reduziert werden. Schockfortpflanzung kann ebenfalls bei der Kollisionsauflösung angewendet werden.

Schockfortpflanzung funktioniert wie folgt. Nach einer vorgegebenen Anzahl an Iterationen wird die Kollisionsauflösung abgebrochen. Anschließend wird der Kontaktgraph ein letztes Mal durchlaufen. Wenn alle Kollisionen auf einer Ebene des Graphen aufgelöst sind, werden die zugehörigen Körper „eingefroren“. Das bedeutet, dass ihre Masse auf unendlich gesetzt wird. Damit werden sie weiterhin bei der Kollisionsauflösung berücksichtigt, aber Impulse haben keine Auswirkung auf ihren Bewegungszustand. Auf diese Weise werden alle Kollisionen der Reihenfolge nach aufgelöst und die Körper eingefroren. Wenn der Graph vollständig abgearbeitet ist, dann werden die Massen der Körper wieder auf ihre ursprünglichen Werte gesetzt und die Kollisionsauflösung ist abgeschlossen. Die Genauigkeit der Kollisionsauflösung ist abhängig von der Anzahl der Iterationen, die vor der Anwendung der Schockfortpflanzung durchgeführt werden.

5.2.5 Reibung

Die Berechnung von statischer und dynamischer Reibung bei der Auflösung einer Kollision basiert auf dem Reibungsgesetz von Coulomb (siehe Abschnitt 5.1.4). Die Reibung zwischen zwei Körpern ist abhängig von dem Impuls in Normalenrichtung, der während der Kollisionsauflösung berechnet wird. Aus diesem Grund werden die Impulse, mit denen Reibung simuliert werden soll, in der gleichen Iterationsschleife bestimmt wie die Kollisionsimpulse. Bei der Auflösung nach der Hypothese von Poisson wird die Berechnung und Anwendung des Rückstoßimpulses als letzte Iteration angesehen, um die Vorgehensweise zu vereinheitlichen.

5.2.5.1 Dynamische Reibung

Dynamische Reibung tritt auf, wenn die relative Tangentialgeschwindigkeit zweier Kontaktpunkte $\mathbf{u}_{rel,t}$ ungleich Null ist oder wenn die Punkte die Bedingung für statische Reibung nicht erfüllen (siehe Abschnitt 5.1.4). Wenn $|\mathbf{u}_{rel,t}| > 0$ gilt, dann muss ein Impuls berechnet werden, der entgegen der Tangentialrichtung $\mathbf{t} = \mathbf{u}_{rel,t}/|\mathbf{u}_{rel,t}|$ wirkt. Die Stärke dieses Reibungsimpulses ist abhängig von dem Koeffizienten für dynamische Reibung μ_d und der Stärke des Impulses in Normalenrichtung. In der i-ten Iteration wird der Impuls zur Simulation von dynamischer

Reibung mit der Gleichung

$$\mathbf{p}_{t,i} = -\mu_d \left|\mathbf{p}'_{n,i}\right| \mathbf{t} \tag{5.8}$$

bestimmt. Es muss allerdings sichergestellt werden, dass nach der Anwendung des Reibungsimpulses $\mathbf{p}_{t,i}$ die relative Geschwindigkeit der Körper in den Kontaktpunkten $\mathbf{u}'_{rel,t}$ die Bedingung

$$0 \leq \mathbf{t} \cdot \mathbf{u}'_{rel,t} \leq \mathbf{t} \cdot \mathbf{u}_{rel,t}$$

erfüllt. Dies bedeutet, der Reibungsimpuls darf die relative Punktgeschwindigkeit nur reduzieren und das höchstens bis zum Stillstand. Der maximal zulässige Reibungsimpuls wird durch die Gleichung

$$\mathbf{p}_{t,max} = -\frac{1}{\mathbf{t}^T \mathbf{K} \mathbf{t}} \mathbf{u}_{rel,t} \tag{5.9}$$

bestimmt. Dieser Impuls eliminiert die relative Geschwindigkeit der Kontaktpunkte in Tangentialrichtung vollständig. Zusammen mit Gleichung 5.8 ergibt sich für den Impuls zur Simulation von dynamischer Reibung:

$$\mathbf{p}'_{t,i} = \begin{cases} \mathbf{p}_{t,i} & \text{falls } \mathbf{t} \cdot \mathbf{p}_{t,max} < \mathbf{t} \cdot \mathbf{p}_{t,i} \\ \mathbf{p}_{t,max} & \text{sonst.} \end{cases}$$

Wenn der Impuls $\mathbf{p}_{t,max}$ angewendet wird, dann wird die relative Tangentialgeschwindigkeit der Punkte Null und in der nächsten Iteration kann es zu statischer Reibung kommen.

5.2.5.2 Statische Reibung

Statische Reibung kann in einer Iteration nur dann auftreten, wenn die relative Geschwindigkeit $\mathbf{u}_{rel,t}$ von zwei Kontaktpunkten Null ist. Außerdem muss der Impuls $\mathbf{p}_t$, der die tangentiale Bewegung der Punkte vollständig abgebremst hat, innerhalb eines Reibungskegels liegen. Dieser Reibungskegel ist bestimmt durch den Koeffizienten der statischen Reibung μ_s und den gesamten Normalenimpuls $\mathbf{p}_n$. Dadurch ergibt sich die Bedingung für die statische Reibung:

$$|\mathbf{p}_t| \leq \mu_s |\mathbf{p}_n|. \tag{5.10}$$

Wenn in der i-ten Iteration die Geschwindigkeit $\mathbf{u}_{rel,t}$ Null ist, dann ergibt die Summe der Reibungsimpulse $\mathbf{p}_t = \sum_{j=1}^{i} \mathbf{p}'_{t,j}$ genau den Impuls, der die tangentiale Bewegung gestoppt hat. Der gesamte Normalenimpuls bis zu dieser Iteration ergibt sich aus der Summe der Kollisionsimpulse $\mathbf{p}_n = \sum_{j=1}^{i} \mathbf{p}'_{n,j}$. Ist die Bedingung 5.10 für die Impulse $\mathbf{p}_n$ und $\mathbf{p}_t$ erfüllt, dann tritt in der i-ten Iteration statische Reibung auf. In den folgenden Iterationen wird für die Kontaktpunkte ein Reibungsimpuls berechnet und angewendet, der ihre relative Geschwindigkeit in Tangentialrichtung

zu Null reduziert. Dies wird solange fortgeführt, bis die Bedingung 5.10 nicht mehr erfüllt ist oder die Iterationsschleife endet. Der Impuls für statische Reibung kann mit Gleichung 5.9 bestimmt werden. Wenn die Bedingung für statische Reibung in einer Iteration nicht mehr erfüllt ist, dann beginnen die beiden Körper in den Kontaktpunkten zu gleiten und dynamische Reibung löst die statische Reibung ab.

Im Fall von statischer Reibung wird die Abbruchbedingung der Iterationsschleife erweitert. Die Schleife kann erst dann abbrechen, wenn für alle Kontakte mit statischer Reibung die Bedingung $\mathbf{u}_{rel,t} = 0$ gilt. Dadurch wird garantiert, dass sich die zugehörigen Kontaktpunkte nach einer Kollision nicht relativ zueinander in der Tangentialebene bewegen.

5.3 Behandlung von bleibenden Kontakten

Nachdem alle Kollisionen mit Impulsen aufgelöst wurden, muss für jedes Paar von Kontaktpunkten überprüft werden, ob es die Bedingung für einen permanenten Kontakt (siehe Abschnitt 5.1.1) erfüllt. Ein solcher Kontakt existiert im Gegensatz zu einer Kollision für einen bestimmten Zeitraum. In diesem Zeitraum wirken auf die Körper eines Kontakts interne und externe Kräfte, welche bei der Auflösung berücksichtigt werden müssen. Aus diesem Grund werden bleibende Kontakte mit Hilfe eines speziellen Gelenks simuliert. Ein solches *Kontaktgelenk* beschreibt die Zwangsbedingung für die Kontaktpunkte in den Körpern. In diesen Punkten muss verhindert werden, dass sich die beiden Körper gegenseitig durchdringen. Zusätzlich wird die Reibung, die zwischen den Körpern auftritt, durch das Gelenk simuliert. Für jeden Kontakt wird ein solches Gelenk temporär in das simulierte Modell eingefügt und bleibt für einen Simulationsschritt ein Teil des Modells. Nach dem Zeitschritt wird das Gelenk wieder aus dem Modell entfernt. Es muss dann erneut geprüft werden, ob der Kontakt weiterhin besteht.

5.3.1 Auflösung der Kontakte

In diesem Abschnitt wird beschrieben, wie die Kontaktbehandlung ohne Reibung durchgeführt wird. Ein Kontaktgelenk definiert für die Kontaktpunkte eine Positionsbedingung. Daher wird für dieses Gelenk, wie bei jedem anderen Gelenk, eine Gelenkkorrektur vor dem Simulationsschritt durchgeführt. Bei der Gelenkkorrektur muss ein Impuls $\mathbf{p}_n$ in Normalenrichtung berechnet und angewendet werden, der dafür sorgt, dass sich die Körper in dem Kontaktpunkt nicht durchdringen.

5.3.1.1 Gelenkkorrektur

Angenommen alle Kollisionen im Modell sind zu dem Zeitpunkt t_0 aufgelöst und es gibt ein Paar von Kontaktpunkten **a** und **b**, das die Bedingung für einen bleibenden Kontakt erfüllt. Dann wird eine Zwangsbedingung aufgestellt, die besagt, dass sich die Körper in den Kontaktpunkten nicht durchdringen dürfen. Wenn ein

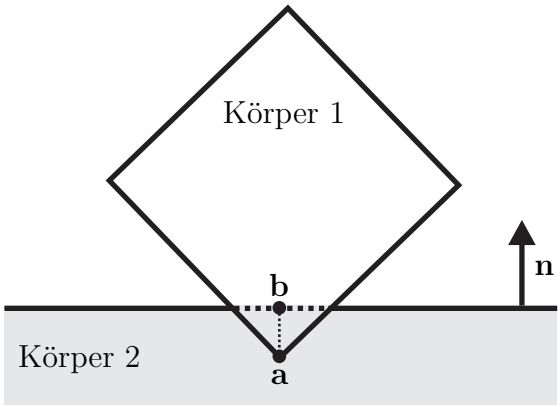


Abbildung 5.5: Durchdringung zweier Körper

Simulationsschritt mit der Zeitschrittweite h durchgeführt wird, ohne diese Bedingung zu beachten, dann ist eine Durchdringung aufgrund der Kräfte, die auf die Körper wirken, möglich (siehe Abbildung 5.5). Um diese Durchdringung zu vermeiden, wird zunächst die Eindringtiefe nach dem Simulationsschritt bestimmt. Dafür müssen die Positionen der Kontaktpunkte nach dem Simulationsschritt ermittelt werden. Die Richtungen, die die Ortsvektoren $\mathbf{r}_{sa}(t_0) = \mathbf{a}(t_0) - \mathbf{s}_1(t_0)$ und $\mathbf{r}_{sb}(t_0) = \mathbf{b}(t_0) - \mathbf{s}_2(t_0)$ von den Schwerpunkten der Körper zu den zugehörigen Kontaktpunkten nach dem Zeitschritt haben, können durch Lösen der Differentialgleichung 3.5 berechnet werden. Die neuen Positionen der beiden Schwerpunkte werden mit Gleichung 3.3 bestimmt. Für die neuen Positionen der Kontaktpunkte und ihren Distanzvektor **d** ergibt sich dann Folgendes:

$$\begin{aligned} \mathbf{a}(t_0 + h) &= \mathbf{r}_{as}(t_0 + h) + \mathbf{s}_1(t_0 + h) \\ \mathbf{b}(t_0 + h) &= \mathbf{r}_{bs}(t_0 + h) + \mathbf{s}_2(t_0 + h) \\ \mathbf{d}(t_0 + h) &= \mathbf{a}(t_0 + h) - \mathbf{b}(t_0 + h). \end{aligned}$$

Die Richtung des Normalenvektors **n** zum Zeitpunkt $t_0 + h$ kann ebenfalls durch numerisches Integrieren der Gleichung 3.5 berechnet werden. Die Distanz bzw. der negative Wert der Eindringtiefe der Körper in Normalenrichtung ist damit:

$$d_n(t_0 + h) = -\mathbf{d}(t_0 + h) \cdot \mathbf{n}(t_0 + h).$$

Die Durchdringung der Körper wird verhindert, indem ein Impuls in Richtung der Kontaktnormalen berechnet und zum Zeitpunkt t_0 angewendet wird. Dieser Impuls

muss bewirken, dass die Eindringtiefe nach dem Zeitschritt Null ist. Er wird, wie bei den Grundgelenken (siehe Abschnitt 3.2.3), in einer iterativen Schleife durch Approximation der benötigten Geschwindigkeitsänderung berechnet. Der Impuls in Normalenrichtung, der den Abstand $d_n(t_0+h)$ reduziert, wird mit der Gleichung

$$\mathbf{p}_n = \frac{1}{\mathbf{n}^T(t_0)\,\mathbf{K}(t_0)\,\mathbf{n}(t_0)} \cdot \frac{1}{h} d_n(t_0+h) \cdot \mathbf{n}(t_0) \tag{5.11}$$

bestimmt. Die Iterationsschleife endet, wenn für die Distanz $|d_n(t_0+h)| \leq \varepsilon_d$ gilt, wobei ε_d der Toleranzwert der Gelenkkorrektur ist.

Wenn ein Körper mehr als einen Kontaktpunkt mit anderen Körpern hat, werden die Abhängigkeiten der Kollisionsimpulse durch die iterative Vorgehensweise aufgelöst. In einer Iteration i wird für ein Paar von Kontaktpunkten ein Impuls $\mathbf{p}_{n,i}$ mit der Gleichung 5.11 berechnet. Bevor dieser Impuls angewendet werden darf, muss sichergestellt sein, dass der Gesamtimpuls $\sum_{j=1}^{i} \mathbf{p}_{n,j}$ die beiden Körper in den Kontaktpunkten abstößt und nicht zusammenhält. Das heißt, er muss die Bedingung 5.6 erfüllen. Dies wird erreicht, indem in der i-ten Iteration der Wert $\mathbf{p}_n$ in die Gleichung 5.7 eingesetzt wird. Der daraus resultierende Impuls $\mathbf{p}'_{n,i}$ muss an Stelle von $\mathbf{p}_n$ angewendet werden. Die iterative Berechnung der Impulse endet, wenn für alle Kontaktpunktpaare entweder die Bedingung $|d_n(t_0+h)| \leq \varepsilon_d$ gilt oder der Abstand in Normalenrichtung positiv und der Gesamtimpuls Null ist. Durch die geänderten Geschwindigkeiten der Körper kann ein Simulationsschritt durchgeführt werden, ohne dass die Körper sich dabei gegenseitig durchdringen.

5.3.2 Schockfortpflanzung

Die Reihenfolge bei der Behandlung der bleibenden Kontakte wird wie bei der Kollisionsauflösung von dem Kontaktgraph (siehe Abschnitt 5.2.3) vorgegeben. Dadurch werden Stapel von Körpern der Reihe nach abgearbeitet. Die Methode der Schockfortpflanzung für Kollisionen wurde in Abschnitt 5.2.4 vorgestellt. Diese Methode kann bei der Behandlung von Kontakten mit Reibung ebenfalls angewendet werden.

Nach einer vorgegebenen Anzahl von Iterationen wird der Kontaktgraph ein letztes Mal durchlaufen. Im letzten Durchlauf werden ausgehend vom Wurzelknoten die Kontakte Ebene für Ebene behandelt. Wenn alle Kontakte einer Ebene aufgelöst wurden, dann werden die zugehörigen Körper eingefroren. Im Gegensatz zu der Methode, die Eran Guendelman verwendet [GBF03, Gue06], dürfen hier die Massen nicht auf unendlich gesetzt werden, um die Körper einzufrieren. Andernfalls wird die Position der Kontaktpunkte nach dem Zeitschritt nicht korrekt bestimmt. Daher werden die eingefrorenen Körper speziell markiert und bei der Impulsberechnung entsprechend behandelt. Nach der Kontaktbehandlung wird die Markierung wieder aufgehoben. Durch die Schockfortpflanzung können insbesondere Modelle mit gestapelten Körpern viel schneller simuliert werden. Ein weiterer

Vorteil ist, dass der Benutzer bestimmen kann, ab welcher Iteration abgebrochen und ein Ergebnis geliefert werden soll.

5.3.3 Reibung

Für die Auflösung von bleibenden Kontakten wird ein spezielles Kontaktgelenk verwendet. Daher muss auch die Reibung, die zwischen den Kontaktpunkten auftritt, mit diesem Gelenk simuliert werden. Im Folgenden wird beschrieben, wie die Reibungsimpulse in der Gelenk- bzw. in der Geschwindigkeitskorrektur bestimmt werden.

5.3.3.1 Gelenkkorrektur

Die Fallunterscheidung zwischen dynamischer und statischer Reibung funktioniert bei der Behandlung von bleibenden Kontakten auf die gleiche Weise, wie bei der Auflösung von Kollisionen (siehe Abschnitt 5.2.5). Dies gilt auch für die Berechnung der Impulse zur Simulation dynamischer Reibung. Bei der Simulation von statischer Reibung kann man sich dagegen den Vorteil zu Nutze machen, dass die Positionen der Kontaktpunkte nach dem Simulationsschritt bekannt sind. Diese Positionen werden in jedem Iterationsschritt der Gelenkkorrektur bei der Auflösung der Kontakte berechnet.

Wenn in einer Iteration der Gelenkkorrektur für ein Paar Kontaktpunkte die relative Tangentialgeschwindigkeit Null und die Bedingung 5.10 erfüllt ist, dann tritt statische Reibung zwischen den Körpern auf. Die beiden Kontaktpunkte haben nach dem Simulationsschritt den Abstandsvektor

$$\mathbf{d}_t(t_0 + h) = -\mathbf{d}(t_0 + h) - d_n(t_0 + h)\mathbf{n}(t_0 + h)$$

in der Tangentialebene. Falls dieser Vektor die Länge Null hat, muss kein Impuls angewendet werden, da sich die Körper während des Zeitschritts nicht in der Tangentialebene bewegen. Andernfalls wird zunächst die Tangente $\mathbf{t}$ durch das Normalisieren des Vektors $\mathbf{d}_t(t_0 + h)$ bestimmt. Danach wird mit der Gleichung

$$\mathbf{p}_t = \frac{1}{\mathbf{t}^T\mathbf{K}(t_0)\mathbf{t}} \cdot \frac{1}{h}\mathbf{d}_t(t_0 + h)$$

ein Reibungsimpuls berechnet, der den Abstand $\mathbf{d}_t(t_0+h)$ reduziert. Die Abbruchbedingung der Gelenkkorrektur wird erweitert, so dass die Iterationsschleife erst endet, wenn für alle Paare von Kontaktpunkten mit statischer Reibung die Bedingung $|\mathbf{d}_t(t_0 + h)| \leq \varepsilon_d$ erfüllt ist. Dadurch kann gewährleistet werden, dass sich zwei Kontaktpunkte mit statischer Reibung nicht in Tangentialrichtung auseinander bewegen.

5.3.3.2 Geschwindigkeitskorrektur

Die Geschwindigkeitskorrektur wird nur im Fall von statischer Reibung benötigt. Durch die Reibungsimpulse bei der Gelenkkorrektur wird erreicht, dass die Kontaktpunkte nach dem Simulationsschritt in Tangentialrichtung keinen Abstand haben. Im Allgemeinen ist die relative Geschwindigkeit der Punkte nach diesem Schritt, bedingt durch die Impulse, ungleich Null. Das heißt, die Kontaktpunkte bewegen sich auseinander. Dies kann verhindert werden, indem in der Geschwindigkeitskorrektur Impulse in Tangentialrichtung berechnet werden, die zur Folge haben, dass $\mathbf{u}_{rel,t}(t_0 + h) = 0$ gilt. Diese Impulse werden mit der Gleichung

$$\mathbf{p}_t = -\frac{1}{\mathbf{t}^T \mathbf{K}(t_0 + h)\mathbf{t}} \cdot \mathbf{u}_{rel,t}(t_0 + h)$$

ermittelt, wobei die Tangente durch $\mathbf{t} = \mathbf{u}_{rel,t}(t_0 + h)/|\mathbf{u}_{rel,t}(t_0 + h)|$ bestimmt ist. Bei Abhängigkeiten zwischen den Kontakten muss die Berechnung der Impulse iterativ fortgesetzt werden, bis die Bedingung $|\mathbf{u}_{rel,t}(t_0+h)| \leq \varepsilon_v$ für alle Kontakte mit statischer Reibung erfüllt ist.

5.4 Kollisionsauflösung in Systemen mit Gelenken

Bisher wurde die Kollisionsauflösung für Modelle ohne Gelenke beschrieben. Wenn in einem Modell Körper durch Gelenke miteinander verbunden sind, müssen die zugehörigen Zwangsbedingungen bei der Auflösung von Kollisionen und bei der Behandlung von Kontakten berücksichtigt werden. Dadurch werden die Abhängigkeiten zwischen den Gelenken und den Kollisionen bzw. Kontakten aufgelöst. In diesem Fall kann die Methode der Schockfortpflanzung für Kollisionen und Kontakte auch in Systemen mit Gelenken angewendet werden.

5.4.1 Auflösung von Kollisionen

Die Auflösung von Kollisionen findet in einem unendlich kleinen Zeitintervall statt. Für jeden Kontaktpunkt werden iterativ Impulse bestimmt, um den Rückstoß der zugehörigen Körper zu simulieren. Dadurch werden die Geschwindigkeiten der kollidierenden Körper verändert. Wenn einer der Körper durch ein Gelenk mit einem weiteren Körper verbunden ist, wird im Allgemeinen die Geschwindigkeitsbedingung dieses Gelenks nach der Anwendung des Impulses nicht mehr erfüllt. Aus diesem Grund wird nach jeder Iteration der Kollisionsauflösung eine Iteration der Geschwindigkeitskorrektur (siehe Abschnitt 3.3) für alle Gelenke durchgeführt. Dies kann mit dem iterativen oder auch mit dem LGS-Verfahren geschehen.

Durch die Geschwindigkeitskorrektur werden die Zwangsbedingungen der Gelenke berücksichtigt. Jeder Impuls, der bei der Kollisionsauflösung angewendet wird, beeinflusst dadurch die Bewegung aller Körper, die mit dem kollidierenden Körper direkt oder indirekt durch Gelenke verbunden sind. Umgekehrt wird auch der simulierte Rückstoß durch die Gelenke beeinflusst.

5.4.2 Behandlung von bleibenden Kontakten

Die Behandlung von bleibenden Kontakten wird mit Hilfe eines speziellen Kontaktgelenks simuliert. Die Zwangsbedingung dieses Gelenks besagt, dass der Abstand der Kontaktpunkte größer oder gleich Null sein muss. Das bedeutet, die Zwangsbedingung wird als Ungleichung formuliert.

Das iterative Verfahren betrachtet bei der Bestimmung der Korrekturimpulse jedes Gelenk für sich. Daher können die Kontaktgelenke ohne Sonderbehandlung in die Gelenk- bzw. die Geschwindigkeitskorrektur integriert werden. Durch die Integration werden die Impulse für die Grundgelenke und die Kontaktgelenke in der gleichen Schleife bestimmt. Auf diese Weise werden die Wechselwirkungen zwischen den Gelenken und den Kontakten berücksichtigt.

Da die Zwangsbedingung eines Kontaktgelenks eine Ungleichung ist, kann sie nicht in das lineare Gleichungssystem beim LGS-Verfahren integriert werden. Bei diesem Verfahren wird ein Gleichungssystem für die Impulse der Grundgelenke während der Gelenkkorrektur in einer iterativen Schleife gelöst. In dieser Schleife können auch die Impulse für die Kontaktgelenke berechnet werden. Die Berechnung dieser Impulse wird mit dem iterativen Verfahren durchgeführt. In einer Iteration der Gelenkkorrektur wird demnach ein lineares Gleichungssystem für die Impulse der Grundgelenke gelöst und anschließend für jedes Kontaktgelenk ein Korrekturimpuls bestimmt. Durch die iterative Vorgehensweise werden die Abhängigkeiten zwischen den Kontakten und den Grundgelenken aufgelöst.

David Baraff beschreibt in [Bar96] die Simulation von Gelenken mit Hilfe von Lagrange-Faktoren. Zwangsbedingungen in Form einer Ungleichung, wie bei den bleibenden Kontakten, oder Bedingungen, die eine Schleife im System definieren, müssen dabei separat behandelt werden. Diese Bedingungen werden auch als sekundäre Bedingungen bezeichnet. In einem Simulationsschritt werden zunächst die Zwangskräfte für die primären Zwangsbedingungen unabhängig von den sekundären berechnet. Diese ergeben zusammen mit den externen Kräften die Kräfte, die auf die sekundären Gelenke wirken. Anschließend wird untersucht, wie sich das System von primären Gelenken verhält, wenn durch ein sekundäres Gelenk eine Kraft auf zwei Körper wirkt. Die Richtung der Kraft muss dabei bekannt sein. Die internen Kräfte der sekundären Gelenke können dann so berechnet werden, dass sie die Reaktion der primären Gelenke berücksichtigen. Zum Schluss werden die primären Zwangsbedingungen aufgelöst, wobei die Zwangskräfte der

sekundären Gelenke als externe Kräfte berücksichtigt werden. Durch diese Vorgehensweise werden die Abhängigkeiten zwischen den primären und den sekundären Gelenken aufgelöst.

Die Grundidee dieser Vorgehensweise kann auf die impulsbasierte Simulation übertragen werden. Die Reaktion der primären Gelenke auf einen Impuls der sekundären wird wie folgt bestimmt. Zunächst werden die relativen Geschwindigkeiten $\mathbf{u}_{rel}$ der Kontaktpunkte bestimmt. Außerdem wird der Zustand des Systems gespeichert. Dann werden für jedes Kontaktgelenk die Impulse $\mathbf{n}$ und $-\mathbf{n}$ auf die beiden Kontaktpunkte angewendet, wobei $\mathbf{n}$ die zugehörige normierte Kontaktnormale ist. Anschließend wird die Gelenkkorrektur der primären Gelenke durchgeführt. Mit den relativen Geschwindigkeiten $\mathbf{u}'_{rel}$ nach dieser Korrektur wird für jedes Kontaktgelenk der Vektor

$$\mathbf{K}_n = \mathbf{K}\,\mathbf{n} + (\mathbf{n} \triangle \mathbf{u}_{rel})\,\mathbf{n}$$

berechnet, wobei der Vektor $\triangle \mathbf{u}_{rel} = \mathbf{u}'_{rel} - \mathbf{u}_{rel}$ die Änderung der relativen Punktgeschwindigkeit angibt. Wenn für jedes Gelenk ein Vektor $\mathbf{K}_n$ bestimmt ist, wird der ursprüngliche Zustand des Systems wieder hergestellt. Setzt man den Vektor $\mathbf{K}_n$ in die Gleichung 5.11 ein, dann erhält man die Gleichung

$$\mathbf{p}_n = \frac{1}{\mathbf{n}^T(t_0)\,\mathbf{K}_n(t_0)} \cdot \frac{1}{h} d_n(t_0 + h) \cdot \mathbf{n}(t_0).$$

Mit dieser Gleichung werden, wie in Abschnitt 5.3.1, Impulse zur Behandlung von Kontakten berechnet, wobei gleichzeitig die Reaktion der primären Gelenke auf diese Impulse berücksichtigt wird. Zum Schluss muss die Gelenkkorrektur der primären Gelenke ein letztes Mal durchgeführt werden. Dann sind alle Zwangsbedingungen aufgelöst und die Positionen und Geschwindigkeiten der Körper für den nächsten Zeitschritt werden berechnet. Auf diese Weise können die Abhängigkeiten der primären und der sekundären Gelenke relativ schnell aufgelöst werden. Allerdings muss die Richtung der Impulse für die sekundären Gelenke im Voraus bekannt sein und dies gilt nur für die Impulse in Richtung der Kontaktnormalen. Das bedeutet, dass Reibung auf diese Weise nicht berücksichtigt werden kann.

5.4.3 Gelenke mit Anschlägen

Die dynamische Simulation von mechanischen Gelenken wurde ausführlich in Kapitel 3 besprochen. Diese Art von Gelenken reduziert die Anzahl der Freiheitsgrade der verbundenen Körper permanent. Wenn ein Freiheitsgrad durch ein Gelenk auf einen bestimmten Bereich begrenzt wird, dann spricht man auch von einem Gelenk mit Anschlag. Zum Beispiel kann ein Drehgelenk auf einen bestimmten Winkelbereich eingeschränkt werden. Dies bedeutet, dass das Gelenk um eine Zwangsbedingung in Form einer Ungleichung erweitert werden muss, um den Anschlag

zu simulieren. Außerdem muss ein Rückstoß simuliert werden, wenn es sich um einen elastischen Anschlag handelt. Anschläge müssen demnach wie Kollisionen gehandhabt werden. Sie können mit den bereits vorgestellten Verfahren simuliert werden, indem zusätzlich einfache Kollisionsobjekte zu den Körpern des Gelenks hinzugefügt werden. Abbildung 5.6 zeigt ein Schienengelenk, bei dem ein Anschlag durch zwei kugelförmige Kollisionsobjekte realisiert wurde. Der Kollisionstest mit

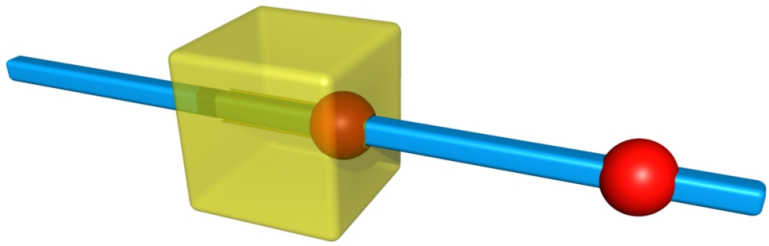


Abbildung 5.6: Schienengelenk dessen Bewegungsfreiheit nach rechts durch einen Anschlag begrenzt ist

Kugeln hat einen geringen Berechnungsaufwand und ist daher schnell durchführbar. Die beiden zusätzlichen Kollisionsobjekte werden nur gegenseitig auf Kontakt getestet, aber nicht mit anderen Objekten. Dies kann mit Hilfe der Matrix für die Kollisionserkennung definiert werden (siehe Abschnitt 4.5). Auf diese Weise lassen sich Gelenke mit Anschlägen mit geringem zusätzlichen Aufwand simulieren.

5.5 Ergebnisse

In diesem Abschnitt werden Ergebnisse präsentiert, die mit der vorgestellten Kollisionsauflösung erzielt wurden. Alle Messungen wurden auf einem PC mit einem 3,4 GHz Intel Pentium 4 Prozessor durchgeführt.

5.5.1 Geschwindigkeit

5.5.1.1 Trichter

Mit dem ersten Modell, das simuliert wurde, sollte die Geschwindigkeit des Verfahrens gemessen werden. Das Modell besteht aus 1000 Würfeln, die durch einen Trichter fallen (siehe Abbildung 5.7). Durch den Trichter wird die Anzahl der Kollisionen in der Simulation erhöht. Die Würfel hatten eine Seitenlänge von einem Meter. Der Koeffizient, der die Elastizität der Körper angibt, hatte einen Wert von $e = 0{,}8$. Für die Simulation der dynamischen und der statischen Reibung wurden die Koeffizienten $\mu_d = \mu_s = 0{,}05$ für die Körper verwendet. Die Kollisionen wurden nach Newtons Stoßgesetz aufgelöst (siehe Abschnitt 5.2.1). Der Toleranzwert

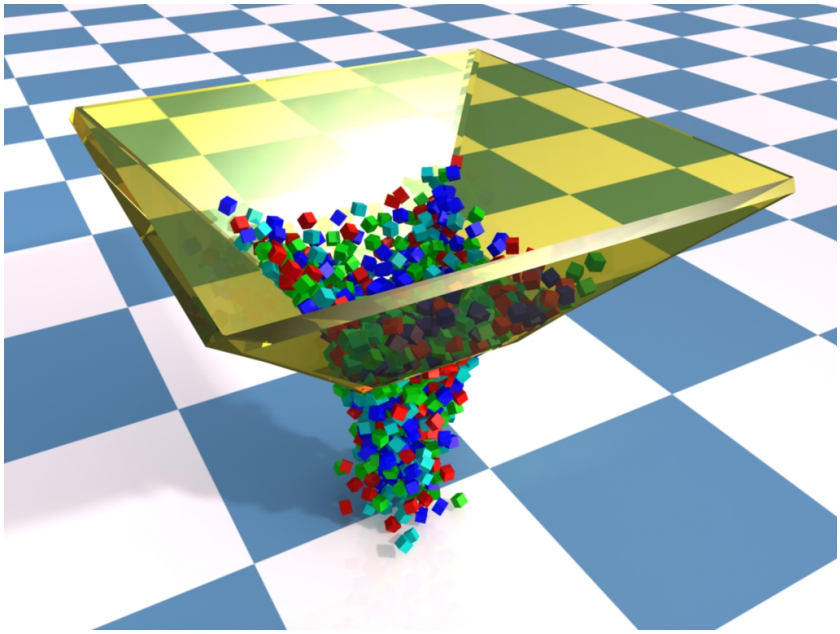


Abbildung 5.7: 1000 Würfeln, die durch einen Trichter fallen

für die Geschwindigkeitsdifferenz war dabei $\varepsilon_{v,c} = 0{,}01\,\frac{\mathrm{m}}{\mathrm{s}}$. Die Behandlung der bleibenden Kontakte wurde mit einem Toleranzwert von $\varepsilon_d = 10^{-4}\,\mathrm{m}$ für die Eindringtiefe durchgeführt. Aufgrund der vielen Kollisionen und Kontakte wurde eine Zeitschrittweite von $h = 0{,}001\,\mathrm{s}$ gewählt. Wenn eine kleine Schrittweite verwendet wird, haben zwei kollidierende Körper bei der Kollisionserkennung eine geringe Eindringtiefe. Diese liegt dann oft im Toleranzbereich. Dadurch muss selten ein Simulationsschritt rückgängig gemacht werden, um den genauen Kollisionszeitpunkt zu bestimmen (vgl. Abschnitt 4.1).

Die Simulation des Modells wurde ohne Schockfortpflanzung durchgeführt, um genaue Ergebnisse zu erzielen. Es wurden sowohl bei der Auflösung von Kollisionen als auch bei der Behandlung von Kontakten die Zeiten gemessen, die in den Simulationsschritten für die Auflösung benötigt wurden. Abbildung 5.8 zeigt die Ergebnisse für die Kollisionsauflösung. Das linke Diagramm zeigt die Rechenzeiten pro Zeitschritt im Verlauf einer Simulation über acht Sekunden. Das rechte Diagramm zeigt eine Kurve für die Anzahl der Kollisionen während des Simulationsverlaufs. An dieser Kurve kann man erkennen, dass zwischen der dritten und der fünften Sekunde die meisten Kollisionen stattgefunden haben. Dies ist der Zeitraum, in dem die Würfel durch den Trichter zusammengedrückt wurden. Die Messung zeigt, dass bei der Simulation in diesem Zeitraum auch am meisten Rechenzeit für die Kollisionsauflösung benötigt wurde. Die Anzahl der Kollisionen und Kontakte ist allerdings nicht allein ausschlaggebend für die benötigte Rechenzeit in einem Simulationsschritt. Eine hohe Anzahl an Abhängigkeiten zwischen den einzelnen Kollisionen und Kontakten kann die Rechenzeit wesentlich erhöhen.

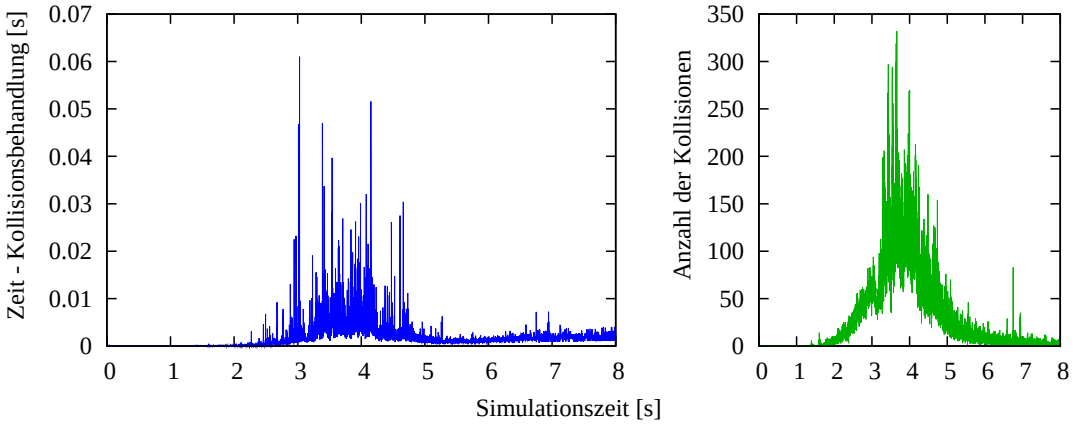


Abbildung 5.8: Messergebnisse der Kollisionsauflösung beim Trichtermodell

Die Ergebnisse der Kontaktbehandlung während der Simulation sind in Abbildung 5.9 dargestellt. Das linke Diagramm zeigt die benötigte Rechenzeit pro Zeit-

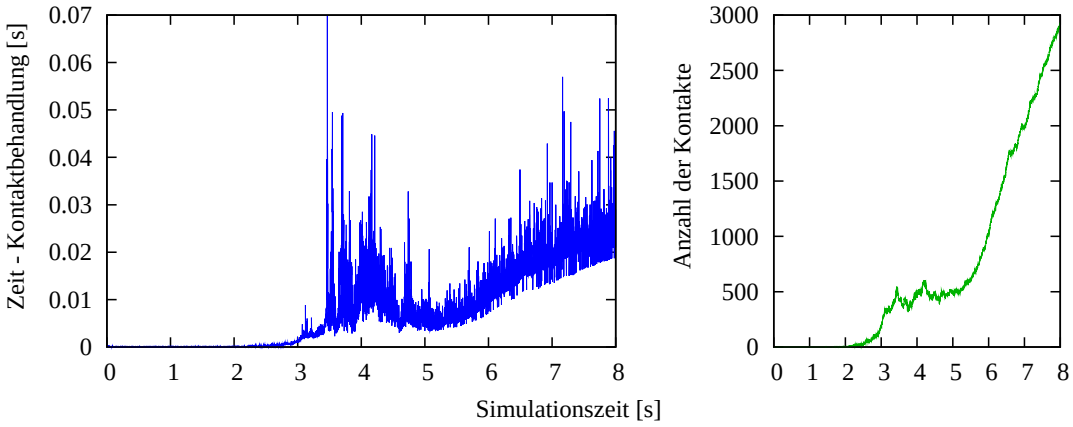


Abbildung 5.9: Messergebnisse der Kontaktbehandlung beim Trichtermodell

schritt und das rechte Diagramm beinhaltet eine Kurve, die die Anzahl der Kontakte während des Simulationsverlaufs beschreibt. Nachdem die Würfel durch den Trichter gefallen sind, kommen sie auf der Ebene unterhalb des Trichters zum Liegen. Daher steigt die Anzahl der bleibenden Kontakte ständig an und erreicht zum Schluss ihren Höhepunkt. Bei den Ergebnissen der Simulation sieht man einen deutlichen Anstieg der Rechenzeiten zwischen der dritten und der fünften Sekunde. Das ist der Zeitraum, in dem die Würfel durch den Trichter fallen. In diesem Zeitraum entstehen sehr komplexe Abhängigkeiten zwischen den bleibenden Kontakten und dementsprechend wird mehr Zeit für die Berechnung der Impulse benötigt.

Die maximalen und die durchschnittlichen Werte der Messung sind in Tabelle 5.1

aufgelistet. Bei der Simulation traten maximal 332 Kollisionen gleichzeitig auf. Die

	Kollisionen	Kontakte
Maximale Anzahl	332	2813
Maximale Rechenzeit	61,02 ms	69,87 ms
Durchschnittliche Anzahl	38	879
Durchschnittliche Rechenzeit	2,02 ms	10,02 ms

Tabelle 5.1: Maximal- und Durchschnittswerte bei der Kollisionsauflösung und Kontaktbehandlung des Trichtermodells

maximale Rechenzeit für die Bestimmung der Impulse bei der Kollisionsauflösung betrug 61 ms. Am Ende der Simulation wurde ein Höchstwert von 2813 bleibenden Kontakten erreicht. Die maximale Zeit für die Kontaktbehandlung wurde allerdings zu einem Zeitpunkt gemessen, zu dem sich noch einige Würfel im Trichter befanden und die ersten Würfel bereits den Boden erreicht hatten. Die Auflösung der Abhängigkeiten zu diesem Zeitpunkt war wesentlich zeitaufwendiger als die Auflösung der vielen Kontakte am Ende der Messung. Durchschnittlich gab es 38 Kollisionen und 879 bleibende Kontakte pro Simulationsschritt. Die Kollisionen konnten im Schnitt in ungefähr 2 ms und die Kontakte in 10 ms aufgelöst werden. Die Simulation des Modells mit den verwendeten Parametern ist nicht echtzeitfähig, aber für die genaue Simulation eines solchen Modells mit Reibung sind die Ergebnisse sehr gut.

5.5.1.2 Mauer

Das Verfahren der Schockfortpflanzung eignet sich besonders gut für Modelle, in denen Körper übereinander gestapelt sind. Aus diesem Grund wurde das Modell einer Mauer simuliert, die aus 252 Steinen besteht (siehe Abbildung 5.10). In diese Mauer schlägt eine Kugel ein, die am Ende eines Pendels hängt. Die Simulation wurde zunächst ohne Schockfortpflanzung durchgeführt. Dabei wurde die Zeit gemessen, die für die gesamte Simulation benötigt wurde. Anschließend wurde in einem zweiten Durchgang Schockfortpflanzung verwendet, um die Anzahl der Iterationsschritte zu verringern und damit die Simulation zu beschleunigen.

Bei der Simulation wurden die Kollisionen nach dem Stoßgesetz von Newton aufgelöst. Jeder Stein der Mauer war 1 m lang, 40 cm breit, 30 cm hoch und hatte eine Masse von 1 kg. Der Radius der Kugel betrug 1 m und ihre Masse war 10 kg. Der Koeffizient der Elastizität war bei allen Körpern $e = 0{,}1$ und die Werte der Reibungskoeffizienten waren $\mu_s = 0{,}2$ und $\mu_d = 0{,}1$. Die Toleranzwerte der Kollisionsauflösung und der Kontaktbehandlung waren $\varepsilon_{v,c} = 0{,}001\,\frac{\mathrm{m}}{\mathrm{s}}$ bzw. $\varepsilon_d = 10^{-5}\,\mathrm{m}$. Die Simulation wurde mit einer kleinen Zeitschrittweite von $h = 0{,}005\,\mathrm{s}$ durchgeführt, um die Bestimmung der genauen Kontaktzeitpunkte zu beschleunigen. Bei

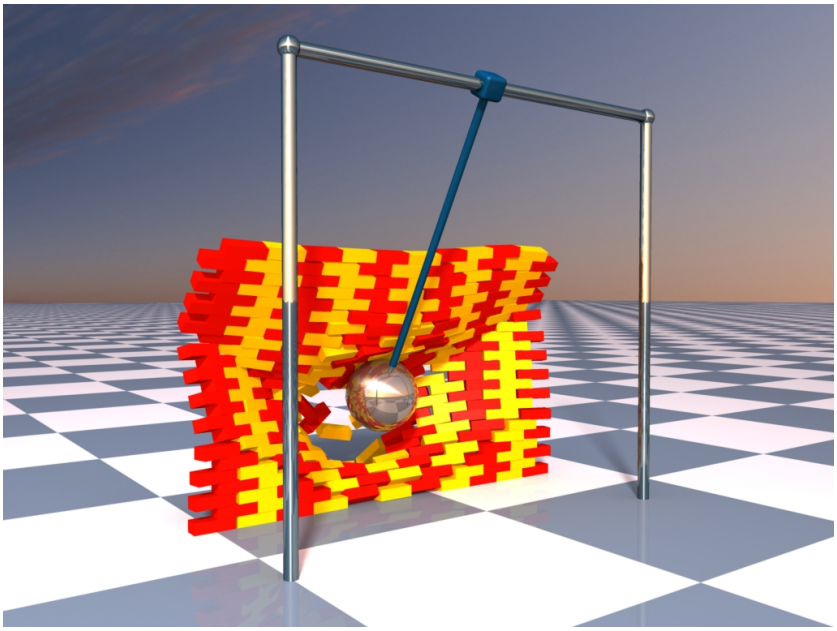


Abbildung 5.10: Kollisionsauflösung mit Schockfortpflanzung

der zweiten Messung wurden die Iterationsschleifen der Kollisionsauflösung und der Kontaktbehandlung jeweils nach maximal zehn Iterationen abgebrochen und das Verfahren der Schockfortpflanzung angewendet. Nach fünf Sekunden Simulationszeit wurde die Rechenzeit gemessen, die für die Simulation des Modells bis dahin benötigt wurde.

Die Simulation eines solchen Modells ist sehr aufwendig. Ein Stein in der Mauer hat theoretisch mit jedem benachbarten Stein vier Kontaktpunkte. In der Praxis kann es durch kleine Ungenauigkeiten dazu kommen, dass ein Stein leicht verdreht auf einem anderen liegt. Dadurch kann sich die Anzahl der Kontaktpunkte sogar noch erhöhen. Bis zum Einschlag der Kugel in die Mauer wurden in jedem Simulationsschritt deutlich mehr als 2000 Kontaktpunkte gezählt. Die Auflösung dieser Kontakte war von daher schwierig, da bis zu 24 Steine übereinander gestapelt waren. Außerdem hatte jeder Impuls, der zur Behandlung eines bleibenden Kontakts angewendet wurde, indirekt eine Auswirkung auf alle anderen Kontaktpunkte in der Mauer.

Die erste Messung ergab eine Rechenzeit von 998 Sekunden, die für die Simulation benötigt wurde. Die Simulation mit Schockfortpflanzung konnte dagegen in nur 91 Sekunden durchgeführt werden. Durch die Einschränkung der Iterationen bei der zweiten Simulation ist die benötigte Rechenzeit wesentlich geringer als bei der ersten Messung. Die Berechnungen wurden mit dem Verfahren der Schockfortpflanzung ungefähr um den Faktor 11 schneller durchgeführt.

5.5.2 Genauigkeit

In diesem Abschnitt soll die Genauigkeit des vorgestellten Verfahrens untersucht werden. Für die Untersuchung der Kontaktbehandlung mit Reibung wird folgendes Modell verwendet. Zehn Würfel mit verschiedenen Reibungskoeffizienten rutschen eine schiefe Ebene herunter (siehe Abbildung 5.11). Die Koeffizienten sind so ge-

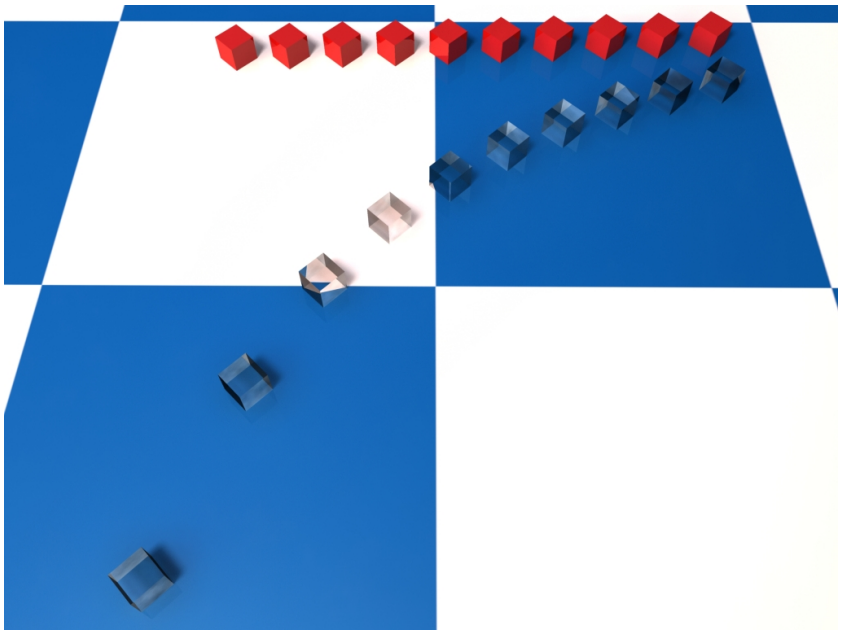


Abbildung 5.11: Zehn Würfel mit unterschiedlichen Reibungskoeffizienten auf einer schiefen Ebene. Die durchsichtigen Würfel zeigen an, an welchen Positionen die Würfel theoretisch zum Stillstand kommen müssen.

wählt, dass bei allen Würfeln nach einer gewissen Zeit der Fall der Haftreibung auftritt. Die einzige externe Kraft, die auf die Körper wirkt, ist Gravitation. Der Bremsweg der Würfel kann analytisch berechnet und mit dem Simulationsergebnis verglichen werden.

Die Berechnung des Bremsweges eines Körpers basiert auf dem Reibungsgesetz von Coulomb. Er ist abhängig vom Reibungskoeffizienten des Körpers und von der Gravitationskraft $\mathbf{F}_g = m \cdot \mathbf{g}$. Für die Berechnung auf der schiefen Ebene mit der Normalen $\mathbf{n}$ wird die Kraft $\mathbf{F}_g$ in eine Normalkomponente $\mathbf{F}_n$ und eine Tangentialkomponente $\mathbf{F}_t$ zerlegt:

$$\begin{aligned} \mathbf{F}_n &= m \cdot (\mathbf{g} \cdot \mathbf{n}) \cdot \mathbf{n} \\ \mathbf{F}_t &= \mathbf{F}_g - \mathbf{F}_n. \end{aligned}$$

Da sich die Körper am Anfang der Simulation auf der Ebene bewegen, tritt dynamische Reibung auf. Die Reibungskraft $\mathbf{F}_r$ ist abhängig von der Kraft in Norma-

lenrichtung und berechnet sich wie folgt:

$$\mathbf{F}_r = -\mu_d \cdot |\mathbf{F}_n| \cdot \hat{\mathbf{v}}_t,$$

wobei $\hat{\mathbf{v}}_t$ der normalisierte Geschwindigkeitsvektor in Tangentialrichtung ist. Insgesamt wirkt die Kraft $\mathbf{F}_{ges} = \mathbf{F}_t + \mathbf{F}_r$ auf den Körper. Da diese Kraft unabhängig von der Zeit ist, kann mit der Gleichung

$$\mathbf{v}(t) = \mathbf{v}(0) + \frac{1}{m}\,\mathbf{F}_{ges}\,t$$

die Geschwindigkeit des Körpers zu einem Zeitpunkt t berechnet werden. Damit kann der Zeitpunkt t_0 ermittelt werden, an dem die Geschwindigkeit $\mathbf{v}(t) \cdot \mathbf{t}$ des Körpers in Richtung der Tangente $\mathbf{t}$ Null wird. Der Bremsweg des Körpers ist dann durch

$$s_B = \left(\mathbf{v}(0) \cdot t_0 + \frac{1}{2\,m}\,\mathbf{F}_{ges}\,t_0^2\right) \cdot \mathbf{t} \tag{5.12}$$

bestimmt.

Die zehn Würfel haben jeweils eine Seitenlänge von 1 m und eine Masse von 1 kg. Die schiefe Ebene wurde um 15 Grad um die z-Achse gedreht. Bei der Simulation wurde eine Zeitschrittweite von $h = 0{,}001\,\mathrm{s}$ verwendet. Der Toleranzwert der Kollisionserkennung war $\varepsilon_c = 0{,}001\,\mathrm{m}$ und die Gelenkkorrektur der Kontaktgelenke wurde mit einem Toleranzwert von $\varepsilon_d = 10^{-5}\,\mathrm{m}$ durchgeführt.

Am Anfang der Simulation sind die Würfel auf der schiefen Ebene in einem Ruhezustand. Zu diesem Zeitpunkt sind die Reibungskoeffizienten aller Körper Null. Während der ersten fünf Sekunden der Simulation rutschen die Körper ungebremst die schiefe Ebene herunter. Nach fünf Sekunden Simulationszeit werden die Reibungskoeffizienten der Körper auf verschiedene Werte von 0,535 bis 0,85 gesetzt. Gleichzeitig wird für jeden Körper, abhängig von seiner Geschwindigkeit und dem Reibungskoeffizienten, der Bremsweg nach Gleichung 5.12 berechnet. Anschließend wird die Simulation fortgesetzt. Nach insgesamt 11 Sekunden Simulationszeit sind alle Körper durch Haftreibung zum Stillstand gekommen. Zu diesem Zeitpunkt wird der Weg gemessen, den die Körper nach der fünften Sekunde zurückgelegt haben. Die Ergebnisse der Simulation, die berechneten Werte und die Abweichungen zwischen den Werten sind in Tabelle 5.2 aufgelistet.

Die maximale Abweichung bei der Messung war 0,0069 m. Diese Abweichung ergab sich über einen Zeitraum von sechs Sekunden Simulationszeit. Bei diesem Körper betrug die durchschnittliche Abweichung in jedem Simulationsschritt $1{,}15 \cdot 10^{-6}\,\mathrm{m}$. Dieser Wert ist fast um den Faktor neun kleiner als die geforderte Genauigkeit von $\varepsilon_d = 10^{-5}\,\mathrm{m}$.

Körper	Reibung	berechneter Bremsweg	simulierter Bremsweg	Abweichung
1	0,535	31,8487 m	31,8508 m	0,0021 m
2	0,57	28,1601 m	28,1535 m	0,0066 m
3	0,605	25,2359 m	25,2308 m	0,0051 m
4	0,64	22,8619 m	22,8550 m	0,0069 m
5	0,675	20,8961 m	20,8919 m	0,0042 m
6	0,71	19,2417 m	19,2420 m	0,0003 m
7	0,745	17,8299 m	17,8295 m	0,0004 m
8	0,78	16,6112 m	16,6136 m	0,0024 m
9	0,815	15,5484 m	15,5541 m	0,0057 m
10	0,85	14,6135 m	14,6168 m	0,0033 m

Tabelle 5.2: Ergebnisse der Bremswegmessung

5.5.3 Gelenke und Reibungseffekte

Das Kugelstoßpendel (siehe Abbildung 5.12) wird als Modell verwendet, um das Zusammenspiel der Kollisionsauflösung und der Simulation von Gelenken zu untersuchen. Dieses Pendel wird oft auch als Newtons Wiege bezeichnet. Das Modell

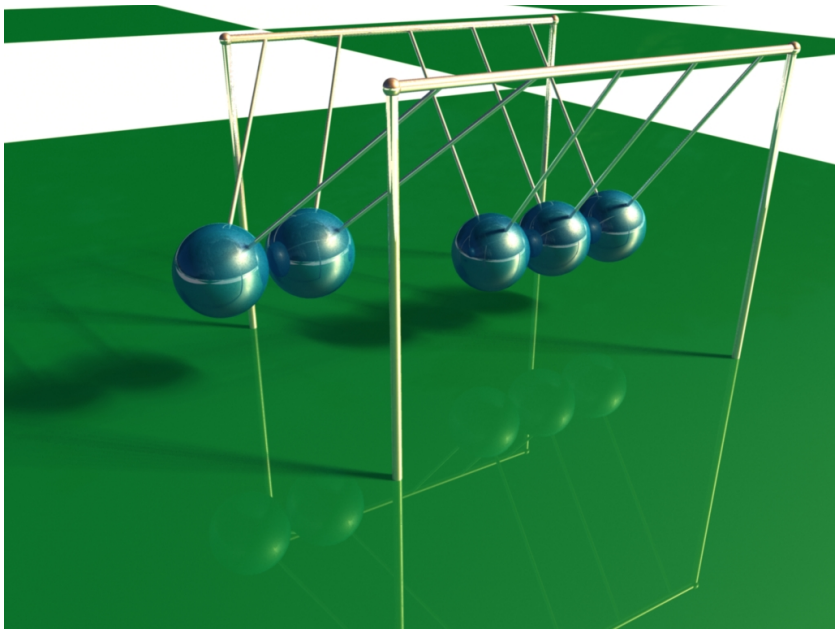


Abbildung 5.12: Kugelstoßpendel

besteht aus fünf aufgehängten Kugeln, die sich gegenseitig berühren. Wird eine

der beiden äußeren Kugeln um einen Winkel α ausgelenkt und losgelassen, dann schwingt sie gegen die restlichen Kugeln. Nach der Kollision wird nur die Kugel am anderen Ende ausgelenkt. Diese erreicht bei einem vollkommen elastischen Stoß eine Auslenkung um den gleichen Winkel α. Wenn mehr als eine Kugel ausgelenkt wird, dann schwingen genauso viele Kugeln nach dem Stoß weiter.

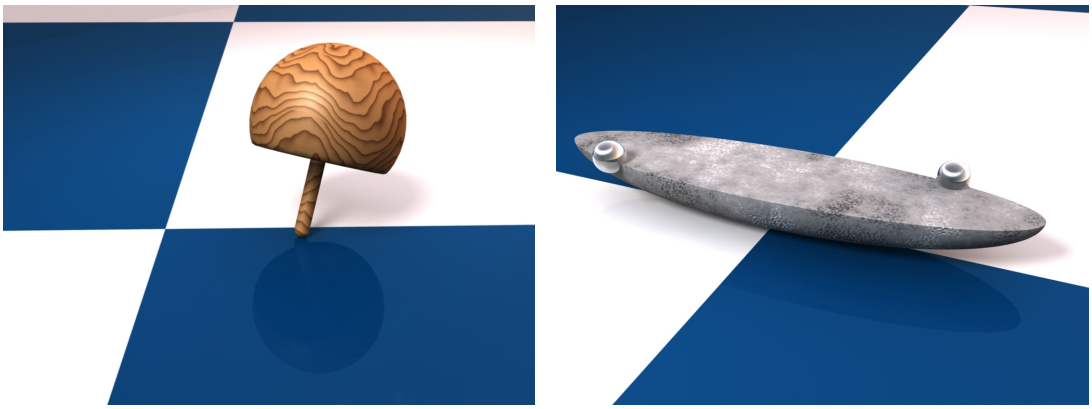
Mit diesem Modell kann die Energie- und Impulserhaltung bei der Kollisionsauflösung untersucht werden. Für die Simulation wurden die vollkommen elastischen Kugeln an Stangen aufgehängt. Die Stangen waren jeweils durch ein Drehgelenk mit dem statischen Gestell verbunden, da bei diesem Modell jede Kugel nur um eine Achse rotieren darf. Die Kugeln hatten einen Radius von 1 m und eine Masse von 1 kg. Die Zeitschrittweite wurde auf 0,001 s gesetzt. Bei der Korrektur der Gelenke und bei der Kollisionsauflösung wurden die Toleranzwerte $\varepsilon_d = 10^{-6}\,\mathrm{m}$ und $\varepsilon_v = \varepsilon_{v,c} = 10^{-6}\,\frac{\mathrm{m}}{\mathrm{s}}$ verwendet. Der Toleranzwert der Kollisionserkennung war $\varepsilon_c = 0{,}001\,\mathrm{m}$. Es wurden verschiedene Simulationen mit dem Kugelstoßpendel durchgeführt. Dabei wurden nacheinander eine bis vier Kugeln um 45 Grad ausgelenkt und losgelassen. Die Gesamtenergie des Systems wurde vor und nach dem Aufprall der Kugeln gemessen, um die Energieerhaltung zu untersuchen. Da der Impuls beim Einschlag vollständig weitergegeben werden müsste, wurde außerdem die Winkelgeschwindigkeit der Kugel, die aufprallt, mit der innersten Kugel, die ausgelenkt wird, verglichen. Die Abweichungen der Gesamtenergie und der Winkelgeschwindigkeit sind in Tabelle 5.3 aufgelistet. Die potentielle Energie ist

ausgelenkte Kugeln	Energieabweichung	Abweichung der Winkelgeschwindigkeit
1	$1{,}106 \cdot 10^{-6}\,\mathrm{Nm}$	$1{,}054 \cdot 10^{-6}\,\frac{\mathrm{rad}}{\mathrm{s}}$
2	$1{,}451 \cdot 10^{-6}\,\mathrm{Nm}$	$2{,}049 \cdot 10^{-6}\,\frac{\mathrm{rad}}{\mathrm{s}}$
3	$1{,}691 \cdot 10^{-6}\,\mathrm{Nm}$	$3{,}273 \cdot 10^{-6}\,\frac{\mathrm{rad}}{\mathrm{s}}$
4	$1{,}975 \cdot 10^{-6}\,\mathrm{Nm}$	$3{,}506 \cdot 10^{-6}\,\frac{\mathrm{rad}}{\mathrm{s}}$

Tabelle 5.3: Messergebnisse

direkt abhängig von den Positionen der Körper. Diese können jeweils nach der Gelenkkorrektur um $10^{-6}\,\mathrm{m}$ von den exakten Positionen abweichen. Die gemessenen Energieabweichungen lagen demnach bei allen Simulationen im erwarteten Bereich. Die Abweichungen der Winkelgeschwindigkeit lassen sich dadurch erklären, dass bei der Geschwindigkeitskorrektur für jede Kugel eine Abweichung von $10^{-6}\,\frac{\mathrm{rad}}{\mathrm{s}}$ erlaubt war.

In Abbildung 5.13 sind die Modelle eines Stehaufkreisels und eines keltischen Wackelsteins abgebildet. In der Realität werden bei diesen beiden Modellen bestimmte Bewegungen durch Reibungseffekte ausgelöst. Die Simulation der Modelle soll zeigen, ob das Verfahren, das in dieser Arbeit vorgestellt wurde, diese Reibungseffekte beherrscht.



(a) Stehaufkreisel (b) Keltischer Wackelstein

Abbildung 5.13: Modelle zur Simulation von Reibungseffekten

Der Stehaufkreisel besteht aus einer abgeschnittenen Kugel und einem Stift (siehe Abbildung 5.13(a)). Die Kugel ist innen hohl. Dadurch liegt der Schwerpunkt des Körpers deutlich unter dem Mittelpunkt der Kugel, wenn der Stift nach oben zeigt. Falls die Kugel Kontakt mit dem Boden hat und der Stehaufkreisel schnell um seine Symmetrieachse rotiert, dann dreht sich der Kreisel nach einer Weile um und rotiert auf dem Stift weiter. Dieses Verhalten ist das Resultat von Reibungseffekten. Eine Erklärung für dieses Verhalten wird zum Beispiel in [Or94] gegeben. Das Modell des Stehaufkreisels wird oft in der Literatur erwähnt und ist ein gutes Beispiel, um die Fähigkeiten der Kontaktauflösung mit Reibung zu testen [SS98a, SS98b].

Das Modell wurde mehrere Male mit verschiedenen Reibungskoeffizienten simuliert. Die Winkelgeschwindigkeit des Kreisels wurde am Anfang der Simulation auf einen relativ hohen Wert gesetzt, um den gewünschten Effekt auszulösen. In jeder Simulation konnte beobachtet werden, dass sich der Kreisel umdreht. Der Zeitpunkt dafür war abhängig von den verwendeten Reibungskoeffizienten. Je höher die Reibung war, umso früher hat sich der Kreisel aufgerichtet.

Der keltische Wackelstein (siehe Abbildung 5.13(b)) ist ein Ellipsoid, das entlang seiner längsten Symmetrieachse in zwei Hälften zerteilt wurde. Durch zwei Kugeln, die diagonal auf der Oberfläche angebracht sind, bekommt der Körper eine asymmetrische Massenverteilung. Die Richtungen der Hauptträgheitsachsen stimmen daher nicht mit den geometrischen Symmetrieachsen überein. Der keltische Wackelstein hat eine Richtung, in der er sich vorzugsweise dreht. Bei einer Rotationsbewegung entgegen dieser Richtung fängt der Körper aufgrund von Reibung an zu wackeln und reduziert seine Winkelgeschwindigkeit. Schließlich dreht der Wackelstein um und dreht in seine Vorzugsrichtung. Die Schwingung des Wackelsteins erreicht ihre maximale Amplitude kurz vor dem Zeitpunkt, an dem der Stein seine Rotationsrichtung umkehrt. Zu dem Zeitpunkt, an dem der Richtungswech-

sel stattfindet, nimmt die Schwingung bereits ab [SSL98]. Die Hintergründe dieses Verhaltens werden in [Bon86] und [GH88] näher erläutert.

Für die Simulation wurde das halbe Ellipsoid durch ein Dreiecksnetz angenähert. Am Anfang der Simulation hatte der Körper eine Winkelgeschwindigkeit entgegen seiner Vorzugsrichtung. Aufgrund der Reibung fing der Stein an, in Richtung der y-Achse zu schwingen. Um diese Schwingung zu messen, wurde die Bewegung

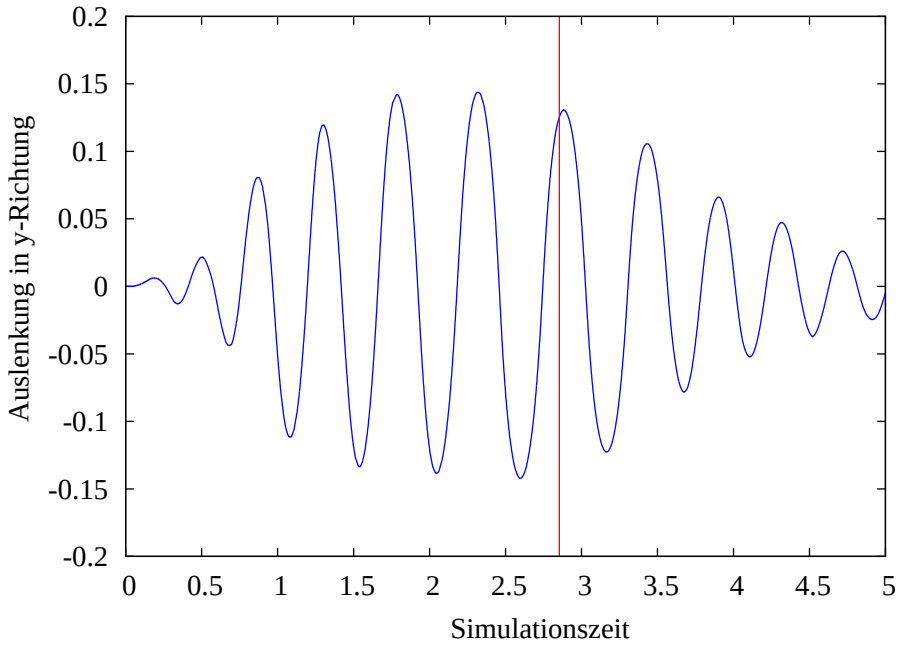


Abbildung 5.14: Schwingung des keltischen Wackelsteins

eines Punktes des Körpers auf seiner längsten Achse verfolgt. Die Auslenkung dieses Punktes auf der y-Achse ist in Abbildung 5.14 dargestellt. Die vertikale Linie in der Abbildung markiert den Zeitpunkt, an dem der Körper seine Drehrichtung gewechselt hat. Dies geschah nach 2,854 s Simulationszeit, kurz nachdem die Schwingung die maximale Amplitude erreicht hatte. Das erwartete Verhalten des keltischen Wackelsteins konnte also mit dem Verfahren zur Kollisionsauflösung simuliert werden.

5.6 Zusammenfassung

Im Folgenden wird das in dieser Arbeit vorgestellte Verfahren zur Kollisionsauflösung mit den bekannten Verfahren aus Abschnitt 2.3 verglichen. Dieser Vergleich wird anhand der Anforderungen für ein Simulationsverfahren in der virtuellen Realität (siehe Abschnitt 1.1) durchgeführt.

Allgemeine Eigenschaften Die impulsbasierten Verfahren berechnen für jede Kollision zwischen zwei Körpern genau einen Impuls zur Auflösung. Kollisionen mit mehreren Kontaktpunkten werden als eine Reihe von zeitlich aufeinander folgenden Kollisionen betrachtet. Bei der Kollisionsauflösung mit Zwangsbedingungen werden dagegen die Bedingungen für alle Kontaktpunkte gleichzeitig behandelt. Die Penalty-Verfahren verwenden Federkräfte, um die Zwangsbedingungen zu erfüllen. Der Betrag einer solchen Kraft ist abhängig von der Eindringtiefe der Körper. Kollisionen werden also erst aufgelöst, nachdem bereits ein unzulässiger Zustand erreicht ist. Die meisten analytischen Verfahren lösen Kollisionen mit Impulsen und bleibende Kontakte mit Kräften auf. Das Problem der Bestimmung der Kontaktkräfte wird dabei oft in ein Optimierungsproblem umgewandelt. Das Verfahren, das in dieser Arbeit vorgestellt wurde, basiert auf der Verwendung von Zwangsbedingungen zur Auflösung der Kollisionen und Kontakte. Diese Zwangsbedingungen werden durch die iterative Berechnung von Impulsen erfüllt.

Geschwindigkeit Penalty-Verfahren sind sehr schnell, da für jede Durchdringung nur eine Federkraft bestimmt werden muss. Diese wirkt während des nächsten Simulationsschritts zusätzlich auf die Körper. Wenn allerdings eine hohe Genauigkeit erreicht werden soll, dann müssen sehr große Kräfte wirken und die Zeitschrittweite muss reduziert werden. Bei genauen Simulationen bremsen diese Verfahren die Simulation aus.

Die impulsbasierten Verfahren arbeiten sehr effizient, da für jede Kollision zwischen zwei Körpern nur ein einziger Impuls berechnet werden muss. Haben zwei Körper in mehreren Punkten Kontakt, werden die Kollisionen in den Punkten zeitlich nacheinander aufgelöst. Zwischen den einzelnen Kollisionen darf nicht viel Zeit vergehen, um genaue Ergebnisse zu erreichen. Daher muss in solchen Fällen die Zeitschrittweite reduziert werden, was die Simulation verlangsamt.

Analytische Verfahren verwenden oft numerische Optimierungsverfahren zur Bestimmung der Kontaktkräfte. Diese Verfahren liefern zwar genauere Ergebnisse als die impulsbasierten und die Penalty-Verfahren, sind aber deutlich langsamer. Das Verfahren in dieser Arbeit löst die Zwangsbedingungen mit Hilfe von Impulsen auf. Diese werden nicht durch ein Optimierungsverfahren, sondern iterativ bestimmt. Da Impulse sehr leicht berechnet werden können, ist das Verfahren deutlich schneller als analytische Verfahren, die mit Optimierungsmethoden arbeiten. Dies kann an einem Beispiel verdeutlicht werden. In [SM04] wird ein ähnliches Modell wie das in Abbildung 5.7 simuliert. Bei dem Modell fallen 1000 Würfel durch einen Trichter in einen Behälter. Die Simulation benötigte 9,5 Stunden, wobei nur physikalisch plausible Ergebnisse erreicht wurden. Außerdem wurden Körper, die sich nur sehr wenig bewegt haben, „eingefroren“. Dadurch konnte viel Rechenzeit auf Kosten der Genauigkeit eingespart werden. Dagegen kann das Verfahren, das in dieser Arbeit vorgestellt wird, innerhalb weniger Minuten genaue Ergebnisse für ein vergleich-

bares Modell liefern. Auf Optimierungen, wie das „Einfrieren“ von Körpern, wird dabei zu Gunsten der Genauigkeit verzichtet.

Genauigkeit Genaue Ergebnisse können mit den Penalty-Verfahren nur dann erreicht werden, wenn sehr steife Federn verwendet werden. Dies führt allerdings zu steifen Differentialgleichungen, welche nur mit kleinen Zeitschrittweiten oder mit speziellen und langsameren Verfahren stabil gelöst werden können [PFTV92].

Die impulsbasierten Verfahren lösen eine Kollision in nur einem Kontaktpunkt auf. Bei mehreren Kontaktpunkten führt dies dazu, dass die Winkelgeschwindigkeit des Körpers nach der Kollision nicht korrekt ist. Da nicht alle Kontaktpunkte gleichzeitig betrachtet werden, vibrieren liegende Körper. Dieses Problem wird zwar in [Mir96b] und [GBF03] auf verschiedene Arten gelöst, aber in beiden Fällen wird die exakte Lösung nur durch eine Vereinfachung angenähert.

Die analytischen Methoden und das in dieser Arbeit vorgestellte Verfahren liefern genaue Ergebnisse, da alle Zwangsbedingungen und alle Abhängigkeiten gleichzeitig aufgelöst werden.

Ergebnisse zu jeder Zeit Ein vorzeitiger Abbruch der Behandlung von Kollisionen und Kontakten ist nur mit den impulsbasierten Verfahren und mit dem Verfahren in dieser Arbeit möglich. Nach dem Abbruch kann z. B. das Verfahren der Schockfortpflanzung (siehe Abschnitt 5.2.4) angewendet werden, um einen zulässigen Zustand des Systems wiederherzustellen. Bei den analytischen und den Penalty-Verfahren wirken die Kontaktkräfte erst im nächsten Simulationsschritt und müssen daher vollständig bestimmt werden, um einen unzulässigen Zustand zu verhindern.

Stabilität In genauen Simulationen treten bei den Penalty-Verfahren sehr große Kräfte auf. Stabile Simulationen sind dann nur mit einer sehr kleinen Zeitschrittweite oder speziellen Integrationsverfahren möglich. Die impulsbasierten Verfahren haben, aufgrund der bei ihnen auftretenden Vibration der Körper, Probleme, gestapelte Körper stabil zu simulieren. Die Methode von Eran Guendelman et al. [GBF03] löst dieses Problem. Allerdings liefert diese Methode keine exakten Ergebnisse. Analytische Verfahren haben bei der Bestimmung der Kontaktkräfte mit Reibung das Problem, dass unter Umständen keine eindeutige oder gar keine Lösung gefunden werden kann [MW88a, Bar93]. Das Verfahren in dieser Arbeit verwendet Impulse, um Kollisionen und Kontakte mit Reibung zu behandeln. Diese können in jeder Situation eindeutig bestimmt werden. Da die Impulse für alle Kontaktpunkte gleichzeitig und unter Berücksichtigung aller Abhängigkeiten berechnet werden, gibt es keine Probleme mit vibrierenden Körpern.

Implementierungsaufwand Das Penalty-Verfahren ist sehr einfach zu implementieren, da pro Kollision nur eine Federkraft berechnet werden muss. Die impulsbasierten Verfahren müssen jeweils einen Impuls für jede Kollision bestimmen. Daher ist der Implementierungsaufwand ebenfalls relativ gering. Das in dieser Arbeit vorgestellte Verfahren verwendet auch Impulse zur Auflösung. Da die Berechnung der Impulse einfach ist, ist die Implementierung dieses Verfahrens nicht schwierig. Analytische Verfahren benötigen oft Optimierungsmethoden, die schwieriger zu implementieren sind. Es existieren allerdings auch analytische Verfahren, die ohne solche Methoden auskommen, wie z. B. das Verfahren von David Baraff [Bar94].

Fazit Die Penalty-Verfahren sind sehr schnell, wenn keine genauen Ergebnisse benötigt werden. Für genaue Simulationen sind diese Verfahren ungeeignet. Die impulsbasierten Verfahren sind ebenfalls schnell, haben allerdings Probleme, wenn zwei Körper in mehreren Punkten Kontakt haben. In diesem Fall liefern diese Verfahren nur plausible Ergebnisse. Das in dieser Arbeit vorgestellte Verfahren berechnet dagegen in einem solchen Fall die Impulse für alle Kontaktpunkte simultan. Daher ist es nur dann langsamer als die impulsbasierten Verfahren, wenn mehrere Kontaktpunkte zwischen zwei Körpern auftreten. Dafür hat es den Vorteil, dass Kollisionen und bleibende Kontakte mit mehreren Kontaktpunkten korrekt aufgelöst werden und dadurch genaue Simulationen möglich sind. Die analytischen Verfahren liefern ebenfalls genaue Ergebnisse. Bei der Verwendung von Optimierungsmethoden sind diese Verfahren allerdings deutlich langsamer.

Das in dieser Arbeit vorgestellte Verfahren kombiniert die impulsbasierte Kollisionsauflösung und die Auflösung mit Zwangsbedingungen. Durch die Zwangsbedingungen wird die hohe Genauigkeit der analytischen Verfahren erreicht, während die Verwendung von Impulsen eine schnelle Simulation ermöglicht.

Tabelle 5.4 fasst den Vergleich der Verfahren zusammen.

	Penalty-Verfahren	impulsbas. Auflösung	Auflösung mit Bedingungen	kombiniertes Verfahren
Geschwindigkeit	+	+	−	+
Genauigkeit	−	−	+	+
vorläufige Ergebnisse	−	+	−	+
Stabilität	−	−	+	+
Implementierung	+	+	−	+

Tabelle 5.4: Vergleich des Penalty-Verfahrens, der impulsbasierten Kollisionsauflösung, der Kollisionsauflösung mit Zwangsbedingungen und des neuen kombinierten Verfahrens, das in dieser Arbeit vorgestellt wurde

Kapitel 6

Die Simulationsumgebung

In diesem Kapitel wird die Simulationsumgebung vorgestellt, die zum Entwickeln und Testen der dynamischen Simulationsverfahren entworfen wurde. Die Simulationsumgebung besteht aus zwei Teilen. Der erste Teil ist eine Erweiterung für die Software Maya[1]. Diese wird zum Modellieren der dreidimensionalen Körper verwendet. Durch die Erweiterung können alle physikalischen Parameter der Körper festgelegt und Gelenke zwischen den Körpern definiert werden. Der zweite Teil der Simulationsumgebung ist der Simulator. In diesem kann der Benutzer physikalische Modelle mit verschiedenen Verfahren simulieren und hat die Möglichkeit zur Interaktion.

6.1 Modellierung

Die Modellierung der dreidimensionalen Körper des physikalischen Modells wird mit Maya durchgeführt (siehe Abbildung 6.1). Da Maya selbst über ein System zur dynamischen Simulation von Starrkörpern verfügt, kann man bereits jedem Körper die notwendigen physikalischen Parameter zuordnen. Allerdings beherrscht die derzeitige Version von Maya nur sehr wenige Gelenktypen. Aus diesem Grund wurde eine Erweiterung von Maya entwickelt, mit der es möglich ist, alle Gelenktypen dieser Arbeit einfach zu definieren. Diese Erweiterung wurde in der Skriptsprache MEL implementiert. Die Abkürzung MEL steht für „Maya Embedded Language“.

Maya bietet die Möglichkeit, für jedes Objekt zusätzliche Attribute zu definieren. Jedem Objekt des dynamischen Modells wird ein Attribut hinzugefügt, das den Typ des Objektes festlegt. Der Typ gibt an, ob es sich um einen Starrkörper, eine Geometrie zur Darstellung, ein Kollisionsobjekt oder ein bestimmtes Gelenk han-

[1]Maya von der Firma Autodesk ist ein Werkzeug zum Modellieren von dreidimensionalen Körpern und Erzeugen von Animationen.

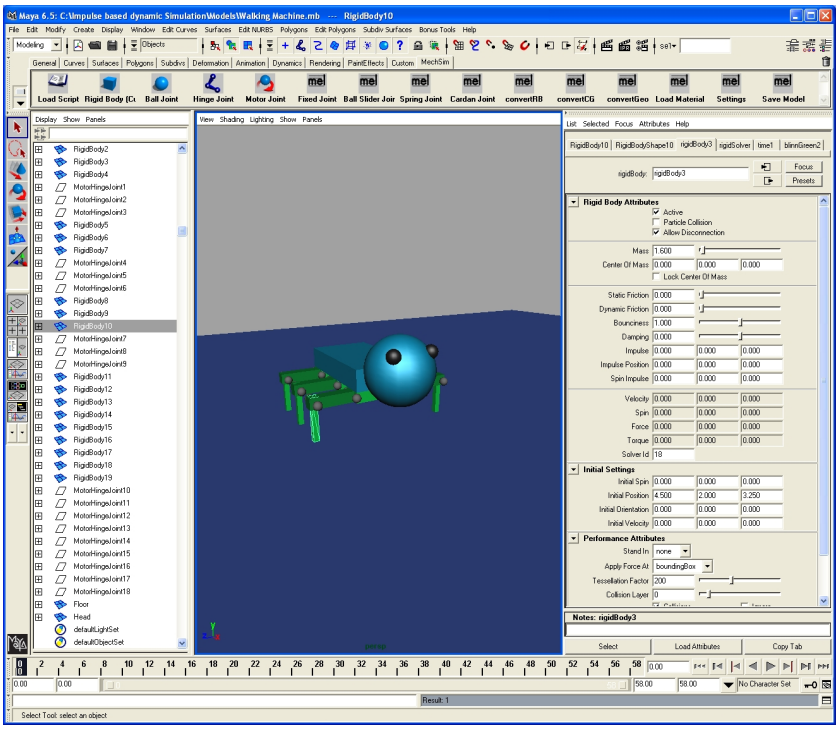


Abbildung 6.1: Entwurf eines physikalischen Modells mit Maya

delt. Die Geometrie eines Starrkörpers wird vom Simulator zur Berechnung des Trägheitstensors und zur Darstellung des Körpers verwendet. Für die Kollisionserkennung kann entweder die Geometrie des Starrkörpers selbst genutzt werden oder es werden beliebig viele Kollisionsobjekte modelliert und dem Starrkörper zugeordnet. In der Simulation repräsentieren diese Kollisionsobjekte den Körper bei der Kollisionserkennung. Ein Starrkörper kann zusätzliche Geometrie haben, die nur zur Darstellung dient. In der Simulation bewegen sich solche geometrischen Objekte mit dem Körper, haben aber keinen Einfluss auf die Simulation. Ein Gelenkobjekt hat mindestens zwei zusätzliche Attribute, durch die definiert ist, welche Körper es miteinander verbindet. Einige Gelenktypen besitzen außerdem Attribute, die bestimmte Eigenschaften des Gelenks festlegen. Zum Beispiel können für einen Servomotor das maximale Drehmoment, Parameter für den zugehörigen PID-Regler usw. definiert werden.

Wenn zwei Körper im Modell durch ein Gelenk miteinander verbunden werden sollen, müssen sie zunächst ausgewählt werden. Anschließend wird ein Gelenktyp gewählt. Das erzeugte Gelenk wird durch ein geometrisches Objekt repräsentiert, das den gewünschten Gelenktyp darstellt. In Abbildung 6.2 sind vier der möglichen Gelenkobjekte dargestellt. Ein Gelenkobjekt wird automatisch mit den nötigen Attributen versehen. Nachdem ein Gelenk erstellt ist, kann der Benutzer das Gelenk in die gewünschte Position bringen und die Werte der Attribute ändern.

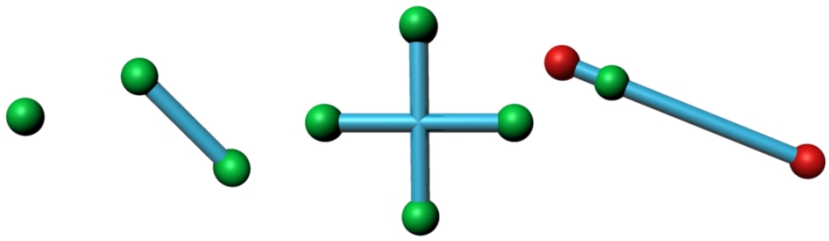


Abbildung 6.2: Kugel-, Dreh-, Kardan- und Schienengelenk in Maya

Das physikalische Modell kann nach der Modellierung in eine XML[2]-Datei exportiert werden. Diese Datei enthält die Geometrie aller Körper, alle Gelenke und die Werte der Attribute.

6.2 Simulator

Der Simulator (siehe Abbildung 6.3), der während dieser Arbeit neu entwickelt wurde, kann die XML-Datei eines physikalischen Modells einlesen und die Simulation des Modells durchführen. In diesem Abschnitt wird zunächst erläutert, wie

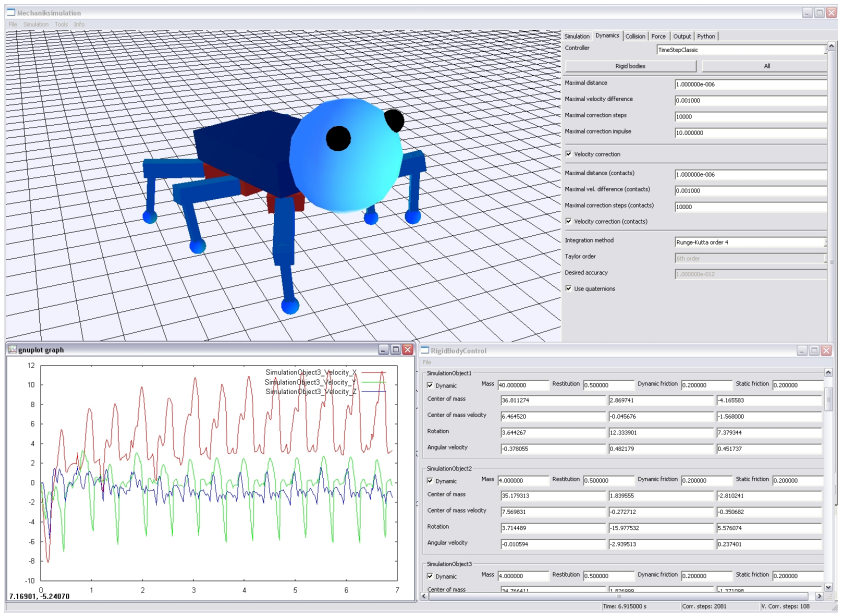


Abbildung 6.3: Simulator

der Simulator aufgebaut ist. Anschließend wird erklärt, welche Möglichkeiten der

[2]Die Abkürzung XML steht für „Extensible Markup Language“. XML ermöglicht das Speichern von Daten in einer Baumstruktur.

Benutzer hat, mit dem Modell während der Simulation zu interagieren. Es gibt zwei Möglichkeiten, die Funktionalität des Simulators zu erweitern. Diese werden im Abschnitt 6.2.3 besprochen.

6.2.1 Aufbau des Simulators

Der Simulator wurde in der Programmiersprache C++ implementiert. Für die graphische Benutzerschnittstelle wurde die Bibliothek „wxWidgets“ verwendet, die für unterschiedliche Plattformen existiert. Dadurch ist der Simulator auf mehreren Plattformen lauffähig und wurde unter Microsoft Windows sowie Linux erfolgreich getestet.

6.2.1.1 Module

Einige Teile des Simulators wurden als austauschbare Module entworfen. Für jedes Modul wurde eine entsprechende Schnittstelle definiert. Module werden eingesetzt für:

- die Simulation der Gelenke,
- die Kollisionserkennung,
- die Kollisionsauflösung,
- die Erzeugung von externen Kräften und
- die Ausgabe der Simulationsdaten.

Für die Simulation von Gelenken wurden in dieser Arbeit verschiedene Verfahren vorgestellt (siehe Abschnitt 3.3). Daher ist es wichtig, dass diese im Simulator problemlos ausgetauscht werden können. Darüber hinaus gibt es im Simulator die Möglichkeit, gleichzeitig mit verschiedenen Verfahren zu simulieren. Eine durch Gelenke miteinander verbundene Kette von Körpern muss dabei einem Verfahren zugeordnet sein. Unabhängige Ketten können mit unterschiedlichen Verfahren simuliert werden. Dadurch können Simulationsverfahren direkt miteinander verglichen werden. Die Verfahren für die Kollisionserkennung und -auflösung sind dagegen nur als Ganzes austauschbar, da die Abhängigkeiten der Körper nicht am Anfang der Simulation vorhersehbar sind.

Die Erzeugung von externen Kräften wurde ebenfalls mit verschiedenen Modulen realisiert. Gravitation wird zum Beispiel durch ein solches Modul simuliert. Es können mehrere unterschiedliche Module gleichzeitig aktiv sein. Jedes dieser Module kennt die Körper, auf die seine Kräfte wirken. Die Module werden vor jedem Simulationsschritt einmal aufgerufen und erzeugen dann die externen Kräfte und

Drehmomente für den nächsten Zeitschritt. Diese können abhängig vom aktuellen Zustand des Modells oder von Benutzereingaben sein. So sind zum Beispiel auch die PID-Regler der Servomotoren (siehe Abschnitt 3.2.5) durch ein solches Modul realisiert.

Schließlich gibt es noch die Module, die für die Ausgabe der Simulationsdaten zuständig sind. Für diese Aufgabe wurden drei verschiedene Module implementiert. Als erstes wurde die dreidimensionale Visualisierung der Simulation mit Hilfe von OpenGL als Modul umgesetzt. Dieses Modul wird in bestimmten Zeitintervallen aufgerufen, um die aktuelle Szene zu zeichnen. Gleichzeitig werden Tastatur- und Mauseingaben verarbeitet, so dass der Benutzer in der Szene navigieren und mit den Objekten interagieren kann (siehe Abschnitt 6.2.2). Die anderen beiden Module werden nach jedem Simulationsschritt aufgerufen. Eines dieser Module kann alle möglichen Simulationsdaten in eine Datei ausgeben und diese dann mit Hilfe des Programms „Gnuplot“ in ein Diagramm einzeichnen. Das letzte Ausgabemodul ermöglicht den Export der Simulationsdaten in Maya. Dabei können die gesamte Geometrie der simulierten Szene, alle Bewegungen der Körper, die Positionen der Gelenkpunkte, die Geschwindigkeitsvektoren der Körper und weitere Daten exportiert werden. Aus diesen Daten wird ein Programm in der Skriptsprache MEL erzeugt, das in Maya verarbeitet werden kann. In Maya lassen sich aus den Simulationsdaten photorealistische Filme erzeugen.

6.2.1.2 Multi-Threading

Die Simulation von physikalischen Modellen ist äußerst rechenintensiv. Bei sehr komplexen Modellen kann die Berechnung eines Simulationsschrittes durchaus einige Sekunden dauern. Damit der Benutzer die graphische Benutzerschnittstelle (GUI) ohne Verzögerung verwenden kann und nicht auf die Simulation warten muss, ist es notwendig, zwei verschiedene Threads zu verwenden. Abbildung 6.4 zeigt die Aufteilung des Simulators in einen Thread für die GUI und einen Thread, in dem die Simulation durchgeführt wird. Um die beiden Threads sauber voneinander zu trennen, wird eine Befehlsliste und ein Datenspeicher verwendet. Wenn der Benutzer die Parameter des simulierten Modells während der Laufzeit ändert, dann wird ein Befehl mit dieser Änderung der Befehlsliste hinzugefügt. Vor jedem Simulationsschritt wird die komplette Liste abgearbeitet und gelöscht. Auf diese Weise wird sichergestellt, dass Parameter nur zwischen den Simulationsschritten verändert werden. Andernfalls könnte die Simulation instabil werden. Nachdem ein Simulationsschritt abgeschlossen ist, werden die aktuellen Daten des Modells, wie z. B. die Positionen und Geschwindigkeiten der Körper, in den Datenspeicher geschrieben. Die Ausgabemodule können dann aus diesem Datenspeicher die benötigten Informationen abfragen. Durch die Verwendung der Befehlsliste und des Datenspeichers wird verhindert, dass ein Thread Daten liest, die gleichzeitig von dem anderen Thread verändert werden.

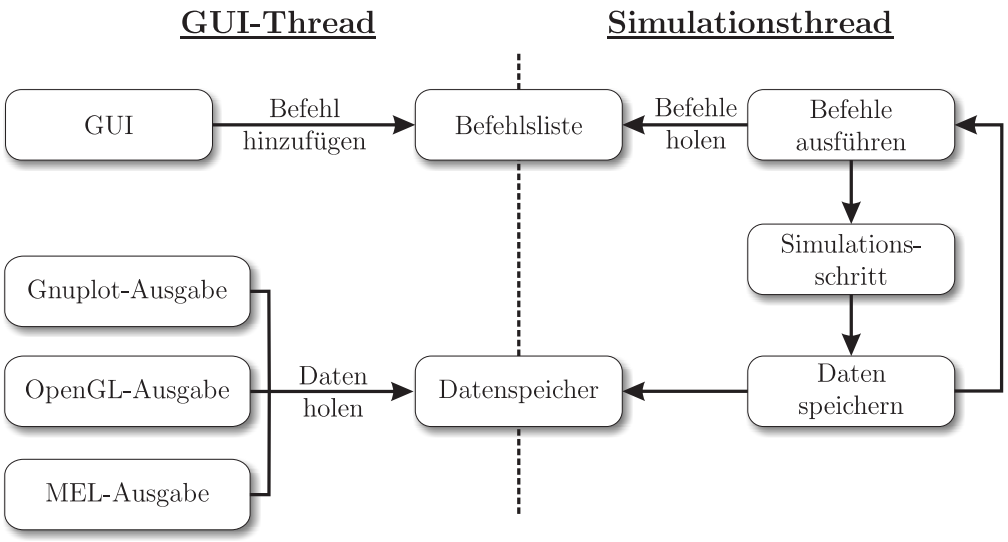


Abbildung 6.4: Aufteilung des Simulators in zwei Threads

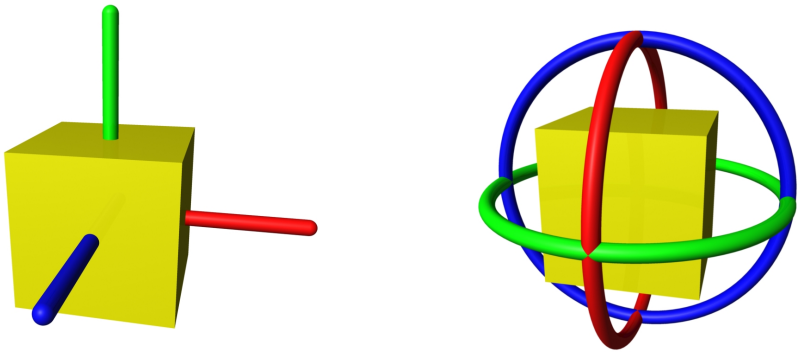
6.2.2 Interaktion

Während der Laufzeit kann der Benutzer in die Simulation eingreifen, indem er Parameter verändert, Impulse auf Körper anwendet oder externe Kräfte auf die Körper wirken lässt. Im Simulator gibt es drei Dialoge, mit denen die Werte aller Parameter der Körper, der Motoren und der Federn gesetzt werden können. Bei den Servomotoren kann der Benutzer zeitliche Abläufe für die Sollwerte definieren. Dadurch lassen sich Bewegungsabläufe simulieren. Abgesehen von den Parametern für die Körper und die Gelenke können alle Einstellungen für die verschiedenen Module über die graphische Benutzerschnittstelle vorgenommen werden. Der Benutzer hat außerdem die Möglichkeit, die Simulation anzuhalten, zu verlangsamen und die Zeitschrittweite anzupassen.

Die Bewegung der Körper in der Simulation lässt sich durch Änderung von Parametern nicht sehr intuitiv beeinflussen. Außerdem kann die Simulation instabil werden, wenn der Benutzer die Position eines Körpers verändert und dadurch Gelenkbedingungen nicht mehr erfüllt werden. Aus diesem Grund wurden für den Simulator zwei weitere Möglichkeiten zur Interaktion mit den Körpern entwickelt. Bei beiden Möglichkeiten kann der Benutzer die Bewegung eines bestimmten Körpers direkt im OpenGL-Fenster durch eine Mauseingabe beeinflussen.

Durch das Anklicken eines Körpers kann ein Punkt auf dem Körper bestimmt werden, auf den ein Impuls angewendet werden soll. Der Benutzer kann durch das Bewegen der Maus die Richtung und die Stärke des Impulses festlegen. Anschließend kann der Körper durch den Impuls weggestoßen oder angezogen werden. Der Impuls ermöglicht dem Benutzer, einen bestimmten Körper gezielt anzustoßen und damit seine Geschwindigkeit zu ändern.

Wenn ein Körper an eine bestimmte Position verschoben oder seine Rotation verändert werden soll, dann müssen alle Zwangsbedingungen des Körpers berücksichtigt werden. Bei Körpern mit Zwangsbedingungen können also nicht einfach die Parameter für den Schwerpunkt bzw. die Drehwinkel verändert werden. Stattdessen hat der Benutzer im Simulator die Möglichkeit, einen Körper auszuwählen und dann seine Position und Rotation direkt in der dreidimensionalen Darstellung zu ändern. Dies geschieht über sogenannte *Manipulatoren* (siehe Abbildung 6.5). Wenn der



(a) Translation (b) Rotation

Abbildung 6.5: Manipulatoren für Translation und Rotation

Benutzer einen Manipulator mit der Maus anwählt, kann er die Translation bzw. die Rotation des Körpers durch das Bewegen der Maus verändern. Die Manipulation des Körpers wird mit Hilfe eines Gelenks mit einer Geschwindigkeitsbedingung (siehe Abschnitt 3.2.6) durchgeführt. Auf diese Weise nimmt der Körper eine bestimmte Position oder Drehung nur dann ein, wenn seine Zwangsbedingungen dies zulassen.

6.2.3 Erweiterbarkeit

Der Simulator bietet zwei Möglichkeiten zur Erweiterung an. Module können beim Programmstart als dynamische Bibliothek eingebunden werden. Außerdem existiert im Simulator die Möglichkeit, zur Laufzeit Programme in der Skriptsprache Python auszuführen.

Module für die Simulation von Gelenken, für die Kollisionserkennung, für die Kollisionsauflösung und für die Erzeugung von externen Kräften können als eine dynamische Bibliothek beim Start des Simulators geladen werden. Dadurch sind diese Module austauschbar und können unabhängig vom Simulator entwickelt werden. Dies bietet die Möglichkeit, verschiedene Verfahren im Simulator zu testen und miteinander zu vergleichen.

Programme, die in der Skriptsprache Python geschrieben sind, können vom Simulator ausgeführt werden. Sie können dabei auf alle Parameter der Körper, der

Gelenke und der verschiedenen Module zugreifen und diese verändern. Ein solches Programm kann zur Laufzeit entweder vom Benutzer manuell aufgerufen werden oder es wird automatisch in Folge eines Ereignisses aufgerufen. So kann z. B. nach jedem Simulationsschritt ein Skript ausgeführt werden. Durch Skriptprogramme kann der Simulator vollständig gesteuert werden. Außerdem ist es durch solche Programme möglich, bei bestimmten Ereignissen automatisch die Parameter des Modells zu ändern.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die dynamische Simulation von Mehrkörpersystemen gewinnt im Bereich der virtuellen Realität immer mehr an Bedeutung. Anwendungen in diesem Bereich haben spezielle Anforderungen an die dynamische Simulation. In dieser Arbeit wurde eine impulsbasierte Simulationsmethode vorgestellt, die im Gegensatz zu anderen Verfahren alle diese Anforderungen erfüllt und damit besonders für den Einsatz in Anwendungen der virtuellen Realität geeignet ist.

Die vorgestellte Simulationsmethode besteht aus:

- der Simulation von Gelenken,
- der Kollisionserkennung und
- der Auflösung von Kollisionen und bleibenden Kontakten mit Reibung.

Durch Gelenke werden Zwangsbedingungen für die Positionen und die Geschwindigkeiten der Körper definiert. Es wurden in dieser Arbeit zwei Verfahren präsentiert, die die Simulation von Mehrkörpersystemen mit solchen Zwangsbedingungen erlauben. Beide Verfahren verwenden ausschließlich Impulse, um einen zulässigen Gelenkzustand herzustellen. Das erste Verfahren bestimmt diese Impulse rein iterativ, während das zweite Verfahren ein lineares Gleichungssystem verwendet, um die Abhängigkeiten der Gelenke zu beschreiben. Das iterative Verfahren ist sehr schnell bei der Simulation von Modellen mit wenig Abhängigkeiten oder wenn bei der Simulation keine hohe Genauigkeit gefordert wird. Außerdem kann es geschlossene kinematische Ketten ohne Sonderbehandlung simulieren. Das zweite Verfahren eignet sich dagegen besonders gut für Modelle mit vielen Abhängigkeiten und bei einer hohen Anforderung an die Genauigkeit. Sechs Grundgelenke wurden vorgestellt,

die mit beiden Verfahren simuliert werden können. Mit diesen Grundgelenken ist es möglich, jede Kombination von translatorischen und rotatorischen Freiheitsgraden aus dem System zu entfernen und damit jedes mechanische Gelenk zu bauen. Außerdem wurden Servomotoren, Federn und Gelenke mit Geschwindigkeitsbedingungen realisiert. Dadurch können unter anderem komplexe Maschinen simuliert werden. Insbesondere bei der Simulation von komplexen Maschinen sind sehr genaue Simulationsergebnisse wünschenswert, da sie Rückschlüsse auf das Verhalten des Modells in der realen Welt zulassen. Aus diesem Grund wurden Verfahren höherer Ordnung hergeleitet und untersucht. Diese Verfahren haben einen höheren Berechnungsaufwand als die beiden ersten Verfahren, liefern dafür aber genauere Simulationsergebnisse.

Für die Simulation von Kollisionen und bleibenden Kontakten mit Reibung wurde ein Verfahren zur Kollisionserkennung benötigt. Zunächst wurden die Anforderungen an ein solches Verfahren aufgestellt. Anschließend wurden verschiedene bekannte Verfahren anhand dieser Anforderungen miteinander verglichen. Die beiden Verfahren, die die Anforderungen am besten erfüllt haben, waren V-Clip und SOLID. Beide wurden so erweitert, dass sie für den Einsatz in der dynamischen Simulation geeignet sind. Die wichtigste Erweiterung war eine Methode zur Bestimmung der Kontaktregionen zwischen kollidierenden Körpern. Die Kontaktregionen werden für eine genaue Auflösung von Kollisionen und bleibenden Kontakten mit Reibung benötigt.

In dieser Arbeit wurde ein neues Verfahren zur Behandlung von Kollisionen und Kontakten mit Reibung vorgestellt. Das Verfahren definiert Zwangsbedingungen für alle Kontaktpunkte, um eine Durchdringung der Körper zu verhindern. Diese Bedingungen werden durch die iterative Berechnung von Impulsen erfüllt. Bei der Auflösung werden Reibungsimpulse nach dem Gesetz von Coulomb bestimmt und angewendet. Dadurch wird dynamische und statische Reibung realisiert. Im Fall einer Kollision wird durch die Impulse zusätzlich ein Rückstoß der Körper simuliert. Wenn zwei Körper in der Simulation miteinander kollidieren, muss sich ihre Geschwindigkeit sofort ändern, um eine Durchdringung zu verhindern. Daher wird für jede Kollision eine Zwangsbedingung für die Geschwindigkeiten der Körper definiert. Diese wird parallel zu den Geschwindigkeitsbedingungen von Gelenken aufgelöst. Dadurch werden die Gelenke bei der Kollisionsauflösung berücksichtigt. Im Fall eines bleibenden Kontakts wird eine Bedingung für die Positionen der Kontaktpunkte definiert. Diese Bedingung sorgt dafür, dass sich die Körper in den Kontaktpunkten während des gesamten Simulationsschritts nicht durchdringen. Die Gelenke werden bei der Behandlung von Kontakten berücksichtigt, indem ihre Positionsbedingungen parallel zu den Bedingungen der Kontakte aufgelöst werden. Für die Auflösung von Kollisionen und die Behandlung von bleibenden Kontakten wurde jeweils eine Methode der Schockfortpflanzung vorgestellt. Mit dieser Methode kann die Simulation auf Kosten der Genauigkeit erheblich beschleunigt werden.

Die in dieser Arbeit vorgestellten Verfahren für die Simulation von Gelenken, Kollisionen und Kontakten ermöglichen sehr genaue Simulationsergebnisse. Mit diesen Verfahren kann die Simulation von Modellen mit holonomen und nichtholonomen Zwangsbedingungen durchgeführt werden. Dadurch kann unter anderem das Verhalten von komplexen Maschinen, wie z. B. Robotern, in der virtuellen Welt untersucht werden. Die Verfahren sind durch den geringen Berechnungsaufwand der Impulse schnell. Daher sind Simulationen von komplexen Modellen in Echtzeit möglich. Außerdem kann der Benutzer in Echtzeit mit den simulierten Modellen interagieren. Dies geschieht zum Beispiel durch die Anwendung von Kräften. Ein großer Vorteil der Simulationsverfahren dieser Arbeit ist, dass sie selbststabilisierend sind. Durch die Vorgehensweise der verwendeten Algorithmen wird ein zulässiger Gelenkzustand immer direkt angesteuert. Ein zusätzliches Stabilisierungsverfahren, wie es zum Beispiel bei der Lagrange-Faktoren-Methode benötigt wird, ist daher überflüssig. Die Simulationsverfahren für Gelenke haben sogar die Eigenschaft, dass sie völlig zerfallene Systeme wieder zusammensetzen können. Durch diese Eigenschaft ist es möglich, die Iterationsschleife zur Bestimmung der Impulse vorzeitig abzubrechen und damit die Simulation auf Kosten der Genauigkeit zu beschleunigen. Bei einem vorzeitigen Abbruch der Schleife können nicht alle Zwangsbedingungen der Gelenke in der vorgegebenen Genauigkeit erfüllt werden. Dadurch werden einige Gelenkzustände unzulässig. Allerdings wird dies durch die stabilisierende Eigenschaft der Verfahren im nächsten Simulationsschritt wieder korrigiert. Die Auflösung von Kollisionen und die Behandlung von Kontakten können durch die vorgestellte Methode der Schockfortpflanzung ebenfalls vorzeitig beendet werden. Dadurch kann die dynamische Simulation zu jeder Zeit ein Ergebnis liefern, falls dies benötigt wird.

Die Implementierung des iterativen Verfahrens für Gelenke, der Kollisionsauflösung und der Kontaktbehandlung ist einfach, da Impulse sehr leicht bestimmt werden können und alle verwendeten Gleichungen maximal dreidimensional sind. Beim LGS-Verfahren muss zusätzlich ein lineares Gleichungssystem aufgestellt und gelöst werden. Für die Lösung eines linearen Gleichungssystems gibt es bereits einige frei verfügbare Bibliotheken mit schnellen Verfahren.

Zusammenfassend lässt sich sagen, dass die vorgestellten Verfahren für die dynamische Simulation die Anforderungen von Anwendungen der virtuellen Realität (siehe Kapitel 1) erfüllen.

7.2 Ausblick

Es wurde bereits gezeigt, dass die Simulationsverfahren dieser Arbeit die Anforderungen für den Einsatz in der virtuellen Realität erfüllen. In diesem Abschnitt wird ein Überblick über die Erweiterungs- und Verbesserungsmöglichkeiten der vorgestellten Verfahren gegeben.

Lineare Laufzeit In dieser Arbeit wurde die Bibliothek PARDISO zum Lösen des linearen Gleichungssystems beim LGS-Verfahren verwendet (siehe Abschnitt 3.3.3). PARDISO arbeitet parallel auf mehreren Prozessoren und wurde speziell für dünnbesetzte Gleichungssysteme optimiert. Daher konnte mit dem LGS-Verfahren ein sehr gutes Laufzeitverhalten erreicht werden. Im Gegensatz zum Algorithmus von Roy Featherstone mit reduzierten Koordinaten [Fea87] und zum Verfahren von David Baraff zur Berechnung von Lagrange-Faktoren [Bar96], kann allerdings mit PARDISO für Modelle mit Baumstruktur keine lineare Laufzeit garantiert werden.

Aktuelle Untersuchungen der Struktur des linearen Gleichungssystems zur Bestimmung der Impulse haben gezeigt, dass für die Matrix $\mathbf{A}$ des Gleichungssystems eine Zerlegung $\mathbf{A} = \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T$ existiert. Dabei ist $\mathbf{M}$ die Massenmatrix, die durch die Gleichung 2.4 bestimmt ist. Die Matrix $\mathbf{B}$ ist eine Blockmatrix, bei der ein Block $\mathbf{B}_{i,j}$ genau dann keine Nullmatrix enthält, wenn das i-te Gelenk den Körper j verbindet. Damit hat die Matrix $\mathbf{B}$ die gleiche Struktur wie die Jacobi-Matrix der Zwangsbedingungen bei der Lagrange-Faktoren-Methode (vgl. Abschnitt 2.1.3). Diese Struktur wird von David Baraff in [Bar96] ausgenutzt, um die Matrix des Gleichungssystems in linearer Zeit und mit linearem Speicheraufwand zu faktorisieren. Dafür wird zunächst eine Matrix $\mathbf{H}$ aufgestellt, die immer dünnbesetzt ist (siehe Abschnitt 2.1.3.2). Diese Matrix kann unter Berücksichtigung der Baumstruktur des Modells so umgeformt werden, dass bei ihrer Faktorisierung kein weiterer von Null verschiedener Block hinzukommt. Anschließend kann die Faktorisierung und die Lösung des zugehörigen Gleichungssystems in linearer Zeit durchgeführt werden. Die Berechnung mit linearem Zeitaufwand wird durch die spezielle Struktur der Matrix $\mathbf{H}$ möglich. Mit Hilfe der Matrizen $\mathbf{B}$ und $\mathbf{M}$ kann für die impulsbasierte Methode eine Matrix mit exakt der gleichen Struktur aufgebaut werden. Diese kann für Modelle mit Baumstruktur ebenfalls in linearer Zeit faktorisiert werden. Auf diese Weise können die Impulse der Gelenk- bzw. der Geschwindigkeitskorrektur in linearer Zeit und mit linearem Speicheraufwand berechnet werden.

Es konnte bereits eine erste Version des impulsbasierten Verfahrens mit linearer Laufzeit implementiert werden. Das Verfahren unterstützt alle Grundgelenke und damit auch alle kombinierten Gelenke. In ersten Messungen mit dem Baummodell (siehe Abschnitt 3.3.3.1) konnte das lineare Laufzeitverhalten bestätigt werden. Der Baum mit 255 Gelenken wurde mit diesem Verfahren sogar schneller als Echtzeit simuliert. Die Simulation des Modells war damit schneller als bei allen bisherigen Messungen in dieser Arbeit. In Zukunft muss das Verfahren mit linearer Laufzeit noch genauer mit verschiedenen Modellen untersucht werden.

Verfahren höherer Ordnung Die Verfahren höherer Ordnung zur Simulation von Gelenken haben nach den theoretischen Untersuchungen eine sehr hohe

Genauigkeitsordnung. Diese konnte in der Praxis allerdings noch nicht erreicht werden [SBP05a]. Weitere Untersuchungen dieser Verfahren könnten zeigen, wie die bisher vorgestellten Simulationsmethoden modifiziert werden müssen, um die erwartete Genauigkeit auch in der Praxis zu erzielen.

Unterstützte Geometrie bei der Kollisionserkennung Die verwendete Kollisionserkennung unterstützt konvexe Dreiecksnetze und einige andere konvexe Körper, wie z. B. Kugeln und Zylinder. Nicht konvexe Objekte können nur verwendet werden, wenn sie in konvexe Teile zerlegt werden. Da Verfahren zur Kollisionserkennung für nicht konvexe Objekte im Allgemeinen langsamer sind als die für konvexe, wäre es zumindest sinnvoll, eine automatische Zerlegung von nicht konvexen Körpern vorzunehmen. Dadurch könnten nicht konvexe Körper durch einen zusätzlichen Vorverarbeitungsschritt unterstützt werden.

Verbesserung der Laufzeit Es existieren verschiedene Möglichkeiten, die Geschwindigkeit der dynamischen Simulation weiter zu steigern. Angenommen ein Körper bleibt auf einer Oberfläche liegen und bewegt sich in den folgenden Simulationsschritten nicht von der Stelle. In diesem Fall werden trotzdem in jedem Simulationsschritt Impulse bestimmt, um zu verhindern, dass der Körper die Oberfläche aufgrund der Gravitation durchdringt. Das bedeutet, dass die Impulse in jedem Schritt genau die Wirkung der Gravitation aufheben. Daher werden in jedem Simulationsschritt, abgesehen von numerischen Ungenauigkeiten, die gleichen Impulse berechnet. Es wäre in einem solchen Fall sinnvoll, die Berechnung der Impulse mit den Gesamtimpulsen des letzten Zeitschritts zu beginnen. Dadurch kann die Kohärenz ausgenutzt werden. Alternativ können Körper, die sich nicht oder nur kaum bewegen, eingefroren werden, wie in [SM04] vorgeschlagen wurde. Ein eingefrorener Körper muss bei der Simulation nicht berücksichtigt werden, bis sich seine Geschwindigkeit z. B. durch eine Kollision ändert. Da bei dieser Vorgehensweise mit Toleranzen gearbeitet werden muss, geht durch das Einfrieren von Körpern allerdings Genauigkeit verloren.

Dynamiksimulation auf einem Graphikprozessor Heutige Graphikkarten verfügen im Allgemeinen über einen Graphikprozessor (GPU), der teilweise programmierbar ist. Programme, die speziell für einen solchen Prozessor implementiert sind, können direkt in der Hardware ausgeführt werden. Das macht diese Programme sehr schnell. Die Implementierung des iterativen Verfahrens für die Simulation von Gelenken und des vorgestellten Verfahrens zur Kollisionsauflösung ist relativ einfach. Die benötigten Gleichungen sind maximal dreidimensional und nicht sehr komplex. Somit ist es theoretisch möglich, diese Verfahren als Programm für den Graphikprozessor umzusetzen. Mit dieser Umsetzung wurde auch bereits begonnen. Zusätzlich wird ein Verfahren zur Kollisionserkennung benötigt, das auf dem

Graphikprozessor funktioniert. Dafür kann z. B. das Verfahren von Alexander Greß et al. verwendet werden [GGK06].

Anhang A

Quaternionen

Quaternionen wurden von Sir William Rowan Hamilton bereits im Jahr 1843 beschrieben [Ham53]. Sie werden oft auch Hamilton-Zahlen genannt. Eine Quaternion ist ein Quadrupel

$$\mathbf{q} = (w, x, y, z) = w + xi + yj + zk$$

mit $w, x, y, z \in \mathbb{R}$ und den imaginären Zahlen i, j und k. Das Symbol für die Menge der Quaternionen ist $\mathbb{H}$. Für die Multiplikation der drei imaginären Zahlen gilt die folgende Verknüpfungstabelle:

$\cdot$	i	j	k
i	-1	k	$-j$
j	$-k$	-1	i
k	j	$-i$	-1

Die Multiplikation von Quaternionen ist nicht kommutativ, wie aus der Tabelle ersichtlich ist. Die Menge $\mathbb{H}$ bildet daher einen Schiefkörper.

Die zu $\mathbf{q}$ konjugierte Quaternion ist durch

$$\mathbf{q}^* \;=\; (w,\; x,\; y,\; z) \;:=\; (w,\; -x,\; -y,\; -z)$$

definiert. Eine Quaternion $\mathbf{q}$, für die $\mathbf{q} \cdot \mathbf{q}^* = \mathbf{q}^* \cdot \mathbf{q} = 1$ gilt, bezeichnet man als Einheitsquaternion. Rotationen lassen sich mit Hilfe von Einheitsquaternionen beschreiben. Sei die normierte Drehachse $\mathbf{x}$ und der Drehwinkel α zu einer Rotation gegeben, dann ergibt sich die zugehörige Einheitsquaternion folgendermaßen:

$$\mathbf{q} = \left(\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}\,\mathbf{x}^T\right).$$

Mit Hilfe dieser Einheitsquaternion kann ein Punkt $\mathbf{a}$ durch die Transformation

$$(0, \mathbf{a}') = \mathbf{q}\,(0, \mathbf{a}^T)\,\mathbf{q}^* \tag{A.1}$$

in der gewünschten Weise gedreht werden.

A.1 Rechenregeln

Die Addition von zwei Quaternionen $\mathbf{q}_1$ und $\mathbf{q}_2$ wird komponentenweise durchgeführt:

$$\begin{aligned} \mathbf{q}_1 + \mathbf{q}_2 &= (w_1, x_1, y_1, z_1) + (w_2, x_2, y_2, z_2) \\ &= (w_1 + w_2,\ x_1 + x_2,\ y_1 + y_2,\ z_1 + z_2). \end{aligned}$$

Da die Addition kommutativ ist, kann auch die Subtraktion komponentenweise vorgenommen werden:

$$\begin{aligned} \mathbf{q}_1 - \mathbf{q}_2 &= (w_1, x_1, y_1, z_1) - (w_2, x_2, y_2, z_2) \\ &= (w_1 - w_2,\ x_1 - x_2,\ y_1 - y_2,\ z_1 - z_2). \end{aligned}$$

Zur Hintereinanderausführung von zwei Rotationen wird das Produkt der beiden zugehörigen Quaternionen berechnet. Dieses ist wie folgt definiert:

$$\begin{aligned} \mathbf{q}_1 \cdot \mathbf{q}_2 &= (w_1, x_1, y_1, z_1) \cdot (w_2, x_2, y_2, z_2) \\ &= (w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2,\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2, \\ &\quad w_1y_2 + y_1w_2 + z_1x_2 - x_1z_2,\ w_1z_2 + z_1w_2 + x_1y_2 - y_1x_2). \end{aligned}$$

Das Quaternionenprodukt $\mathbf{q}_1 \cdot \mathbf{q}_2$ umfasst $4 \cdot 4 = 16$ Multiplikationen und $4 \cdot 3 = 12$ Additionen. Dies ist günstiger als die Berechnung der Rotation mit dem Matrixprodukt zweier 3×3 Rotationsmatrizen, denn das Matrixprodukt kostet $3 \cdot 3 \cdot 3 = 27$ Multiplikationen und $3 \cdot 3 \cdot 2 = 18$ Additionen. Die Rotation eines Punktes mit einer Quaternion (mit Gleichung A.1) ist allerdings wesentlich teurer als die Drehung mit einer Rotationsmatrix. Daher ist es sinnvoll, die Verkettung von Rotationen mit Quaternionen zu berechnen und für die Drehung von Punkten die jeweilige Quaternion in eine Rotationsmatrix zu konvertieren (siehe Abschnitt A.2).

Das multiplikativ inverse Element zu einer Quaternion $\mathbf{q}$ ist

$$\mathbf{q}^{-1} := \frac{\mathbf{q}^*}{|\mathbf{q}|^2} = \frac{\mathbf{q}^*}{\mathbf{q} \cdot \mathbf{q}^*} \qquad (\mathbf{q} \cdot \mathbf{q}^* \in \mathbb{R}).$$

Daraus folgt, dass für eine Einheitsquaternion $\mathbf{q}^* = \mathbf{q}^{-1}$ gilt.

A.2 Konvertierung

Eine Rotationsmatrix $\mathbf{R}$ kann mit Hilfe des folgenden Algorithmus [Sho85] in eine Einheitsquaternion $\mathbf{q}$ umgewandelt werden:

---

matrixToQuaternion()

Eingabe: Rotationsmatrix $\mathbf{R}$
Ausgabe: Einheitsquaternion $\mathbf{q} = (w, x, y, z)$

$Spur := 1 + R_{11} + R_{22} + R_{33}$
if $Spur > 0$
  $s := \sqrt{Spur}$
  $w := 0{,}5\, s$
  $s := 0{,}5/s$
  $x := (R_{32} - R_{23})\, s$
  $y := (R_{13} - R_{31})\, s$
  $z := (R_{21} - R_{12})\, s$
else
  $i := 1$
  if $R_{22} > R_{11}$
    $i := 2$
  if $R_{33} > R_{ii}$
    $i := 3$
  if $i = 1$
    $s := \sqrt{(R_{11} - (R_{22} + R_{33})) + 1}$
    $x := 0{,}5\, s$
    $s := 0{,}5/s$
    $y := (R_{12} + R_{21})\, s$
    $z := (R_{31} + R_{13})\, s$
    $w := (R_{32} - R_{23})\, s$
  else if $i = 2$
    $s := \sqrt{(R_{22} - (R_{33} + R_{11})) + 1}$
    $y := 0{,}5\, s$
    $s := 0{,}5/s$
    $z := (R_{23} + R_{32})\, s$
    $x := (R_{12} + R_{21})\, s$
    $w := (R_{13} - R_{31})\, s$
  else
    $s := \sqrt{(R_{33} - (R_{11} + R_{22})) + 1}$
    $z := 0{,}5\, s$
    $s := 0{,}5/s$
    $z := (R_{31} + R_{13})\, s$
    $x := (R_{23} + R_{32})\, s$
    $w := (R_{21} - R_{12})\, s$

Algorithmus A.1: Konvertierung einer Rotationsmatrix in eine Quaternion

Umgekehrt kann eine Einheitsquaternion $\mathbf{q} = (w, x, y, z)$ in eine Rotationsmatrix konvertiert werden. Die Rotationsmatrix von $\mathbf{q}$ ist gegeben durch:

$$\mathbf{R}_q = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(zx+yw) \\ 2(xy+zw) & 1-2(z^2+x^2) & 2(yz-xw) \\ 2(zx-yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{pmatrix}.$$

Am Anfang des Kapitels wurde bereits gezeigt, wie eine Drehachse mit zugehörigem Drehwinkel in eine Einheitsquaternion konvertiert wird. Mit dem Algorithmus A.2 lässt sich umgekehrt eine Quaternion in eine Drehachse mit Winkel umwandeln.

quaternionToAxisAngle()

Eingabe: Einheitsquaternion $\mathbf{q} = (w, x, y, z)$
Ausgabe: Normierte Drehachse $\mathbf{x}$ und Drehwinkel α

$l := \sqrt{x^2 + y^2 + z^2}$
if $l > 0$
  $\alpha := 2 \arccos(w)$
  $\mathbf{x} := 1/l \cdot (x, y, z)^T$
else
  $\alpha := 0$
  $\mathbf{x} := (1, 0, 0)^T$

Algorithmus A.2: Konvertierung einer Einheitsquaternion in eine normierte Drehachse mit zugehörigem Drehwinkel

Anhang B

Numerische Lösungsverfahren

B.1 Lösung einer gewöhnlichen Differentialgleichung

Eine Differentialgleichung beschreibt die Beziehung zwischen einer unbekannten Funktion und den Ableitungen dieser Funktion. Eine Gleichung der Form

$$y^{(n)}(x) = f(x, y(x), y'(x), \ldots, y^{(n-1)}(x))$$

mit $f : \mathbb{R}^{n+1} \mapsto \mathbb{R}$ heißt gewöhnliche Differentialgleichung n-ter Ordnung in expliziter Form. Gewöhnliche Differentialgleichungen enthalten nur Ableitungen nach einer Variablen. Die Ordnung der Differentialgleichung ergibt sich aus dem höchsten vorkommenden Ableitungsgrad. Wenn die Parameter der Funktion f an einer bestimmten Stelle $x = x_0$ vorgegeben werden, dann spricht man von einem Anfangswertproblem. In dieser Arbeit werden ausschließlich Anfangswertprobleme der Form

$$\dot{\mathbf{x}} = f(t, \mathbf{x}) \tag{B.1}$$

gelöst. Dabei ist f eine bekannte Funktion, t ist der Zeitpunkt, an dem die Funktion ausgewertet wird und $\mathbf{x}$ ist der Zustand des Systems zu diesem Zeitpunkt. Der Term $\dot{\mathbf{x}}$ gibt die Ableitung von $\mathbf{x}$ nach der Zeit an.

B.1.1 Euler-Verfahren

Sei ein Anfangswertproblem der Form B.1 mit dem Anfangswert $\mathbf{x}_0 = \mathbf{x}(t_0)$ gegeben. Durch Integration der Differentialgleichung erhält man

$$\mathbf{x}(t_0 + h) = \mathbf{x}_0 + \int_{t_0}^{t_0+h} f(t, \mathbf{x}) dt$$

für den Zustand des Systems zum Zeitpunkt t_0+h. Da diese Gleichung im Allgemeinen nicht analytisch gelöst werden kann, wird beim Euler-Verfahren die folgende Näherung verwendet, um den Wert $\mathbf{x}(t_0 + h)$ zu approximieren:

$$\mathbf{x}(t_0 + h) = \mathbf{x}_0 + \int_{t_0}^{t_0+h} f(t, \mathbf{x})dt \approx \mathbf{x}_0 + h\, f(t_0, \mathbf{x}_0) = \mathbf{x}_1.$$

Dies führt zu der folgenden Vorschrift zur Berechnung der weiteren Werte:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\, f(t_i, \mathbf{x}_i).$$

In Abbildung B.1 ist die Bestimmung der Lösung des Anfangswertproblems $\dot{y} = f(t, y)$ mit dem Euler-Verfahren veranschaulicht. Das Euler-Verfahren ist sehr

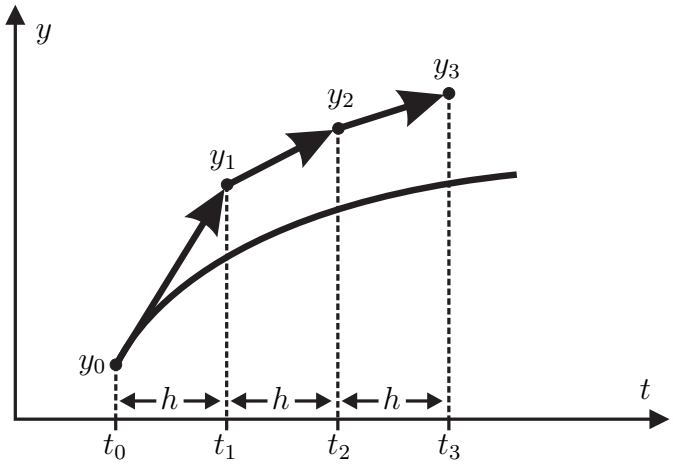


Abbildung B.1: Lösen einer Differentialgleichung mit dem Euler-Verfahren

einfach zu implementieren und berechnet sehr schnell eine Lösung für das gegebene Anfangswertproblem. Allerdings hat es einen lokalen Fehler der Ordnung $O(h^2)$. Daher eignet es sich nicht, wenn genaue Simulationen durchgeführt werden sollen.

B.1.2 Taylor-Reihen

Das Verfahren von Euler ist ein Spezialfall der Entwicklung von Taylor-Reihen. Eine stetige Funktion $f(t)$, die an der Stelle $t = t_0$ beliebig oft differenzierbar ist, kann durch die folgende Taylor-Reihe dargestellt werden:

$$f(t) = f(t_0) + f'(t_0)\, h + f''(t_0)\, \frac{h^2}{2!} + \ldots + f^{(n)}(t_0)\, \frac{h^n}{n!} + \ldots,$$

wobei $h = t - t_0$ die verwendete Schrittweite ist. Damit kann theoretisch der Wert der Funktion f an einer beliebigen Stelle t bestimmt werden, wenn der Funktionswert und alle Ableitungen an einer Stelle t_0 bekannt sind. In der Praxis können nicht unendlich viele Ableitungen berechnet werden. In diesem Fall wird die

Taylor-Reihe nach einem bestimmten Term abgebrochen, wodurch ein Fehler für den Funktionswert $f(t)$ entsteht. Wenn die Reihe nach dem Term mit der n-ten Ableitung der Funktion f abgebrochen wird, dann entsteht ein Fehler der Ordnung $O(h^{(n+1)})$.

Bei dem Verfahren zur dynamischen Simulation, das in dieser Arbeit vorgestellt wird, müssen drei verschiedene Differentialgleichungen gelöst werden. Dies sind die Differentialgleichungen für die Winkelgeschwindigkeit $\boldsymbol{\omega}$, den Ortsvektor $\mathbf{r}$ und die Einheitsquaternion $\mathbf{q}$. Diese Gleichungen können mit Hilfe von Taylor-Reihen gelöst werden.

Die Differentialgleichung für die Winkelgeschwindigkeit wird in lokalen Koordinaten aufgestellt und gelöst, da in diesem Koordinatensystem der Trägheitstensor und seine Inverse konstant sind:

$$\dot{\boldsymbol{\omega}}_l(t) = \mathbf{J}_l^{-1} \left(\boldsymbol{\tau}_{ext,l} - (\boldsymbol{\omega}_l(t) \times (\mathbf{J}_l\, \boldsymbol{\omega}_l(t)))\right).$$

Die höheren Ableitungen von $\boldsymbol{\omega}_l(t)$ können mit der folgenden Gleichung bestimmt werden

$$\boldsymbol{\omega}_l^{(n+1)}(t) = -\mathbf{J}_l^{-1} \sum_{k=0}^{n} \binom{n}{k} \left(\boldsymbol{\omega}_l^{(k)}(t) \times \left(\mathbf{J}_l\, \boldsymbol{\omega}_l^{(n-k)}(t)\right)\right),$$

wobei $n \geq 1$ gelten muss. Wenn die Werte der Ableitungen für einen Zeitpunkt t_0 berechnet werden, dann ist die lokale Winkelgeschwindigkeit zum Zeitpunkt $t_0 + h$ durch die Taylor-Reihe

$$\boldsymbol{\omega}_l(t_0 + h) = \boldsymbol{\omega}_l(t_0) + \dot{\boldsymbol{\omega}}_l(t_0)\, h + \ddot{\boldsymbol{\omega}}_l(t_0)\, \frac{h^2}{2!} + \ldots + \boldsymbol{\omega}_l^{(n)}(t_0)\, \frac{h^n}{n!} + \ldots$$

bestimmt.

Für den Vektor $\mathbf{r}$ vom Schwerpunkt eines Körpers zu einem anderen Punkt in dem Körper gilt die Differentialgleichung

$$\dot{\mathbf{r}}(t) = \boldsymbol{\omega}(t) \times \mathbf{r}(t).$$

Diese beschreibt, wie sich die Richtung des Vektors durch die Rotation des Körpers mit der Zeit ändert. Die Ableitung des Vektors $\mathbf{r}$ ist abhängig von der Winkelgeschwindigkeit des Körpers. Die Ableitungen der Winkelgeschwindigkeit für den Zeitpunkt t_0 werden bei der Berechnung der neuen Winkelgeschwindigkeit in lokalen Koordinaten berechnet. Daher ist es sinnvoll, die Taylor-Reihe für den Vektor $\mathbf{r}$ ebenfalls in lokalen Koordinaten aufzustellen und die bereits berechneten Werte für die Ableitungen wiederzuverwenden. Die höheren Ableitungen für den Vektor in lokalen Koordinaten werden durch folgende Gleichung bestimmt:

$$\mathbf{r}_l^{(n+1)}(t) = \sum_{k=0}^{n} \binom{n}{k} \left(\boldsymbol{\omega}_l^{(k)}(t) \times \mathbf{r}_l^{(n-k)}(t)\right).$$

Der Vektor $\mathbf{r}_l$ zum Zeitpunkt $t_0 + h$ kann mit Hilfe der folgenden Taylor-Reihe berechnet werden:

$$\mathbf{r}_l(t_0 + h) = \mathbf{r}_l(t_0) + \dot{\mathbf{r}}_l(t_0)\, h + \ddot{\mathbf{r}}_l(t_0)\, \frac{h^2}{2!} + \ldots + \mathbf{r}_l^{(n)}(t_0)\, \frac{h^n}{n!} + \ldots .$$

Die Ableitung der Einheitsquaternion $\mathbf{q}$, die die Rotation eines Körpers beschreibt, ist abhängig von der Winkelgeschwindigkeit des Körpers. Die Taylor-Reihe für die Quaternion wird ebenfalls im lokalen Koordinatensystem aufgestellt, da die Ableitungen der Winkelgeschwindigkeit für den Zeitpunkt t_0 in diesem Fall wiederverwendet werden können. Die Differentialgleichung für die Einheitsquaternion in lokalen Koordinaten hat die folgende Form:

$$\dot{\mathbf{q}}_l(t) = \frac{1}{2}\, \boldsymbol{\omega}_l(t) \cdot \mathbf{q}_l(t).$$

Die höheren Ableitungen ergeben sich durch die Anwendung der Produktregel:

$$\mathbf{q}_l^{(n+1)}(t) = \frac{1}{2} \sum_{k=0}^{n} \binom{n}{k} \left(\boldsymbol{\omega}_l^{(k)}(t) \cdot \mathbf{q}_l^{(n-k)}(t) \right).$$

Mit diesen Ableitungen lässt sich die Taylor-Reihe, mit der die Quaternion für den Zeitpunkt $t_0 + h$ berechnet werden kann, aufstellen:

$$\mathbf{q}_l(t_0 + h) = \mathbf{q}_l(t_0) + \dot{\mathbf{q}}_l(t_0)\, h + \ddot{\mathbf{q}}_l(t_0)\, \frac{h^2}{2!} + \ldots + \mathbf{q}_l^{(n)}(t_0)\, \frac{h^n}{n!} + \ldots .$$

Bei der numerischen Integration mit Hilfe von Taylor-Reihen ist die Fehlerordnung abhängig von der Zeitschrittweite h und der Anzahl n der verwendeten Ableitungen. Wenn die Zeitschrittweite konstant ist, kann durch das Anpassen des Parameters n die Fehlerordnung verändert werden. Durch eine Erhöhung von n wird der Fehlerterm kleiner und dadurch das Ergebnis genauer. Allerdings müssen weitere Ableitungsterme berechnet werden.

B.1.3 Runge-Kutta-Verfahren vierter Ordnung

Im Gegensatz zu den Taylor-Reihen benötigt das Runge-Kutta-Verfahren vierter Ordnung für die numerische Integration nur die erste Ableitung einer Funktion. Außerdem hat dieses Verfahren ein besseres Konvergenzverhalten als das Euler-Verfahren.

Sei ein Anfangswertproblem der Form $\dot{\mathbf{x}} = f(t, \mathbf{x})$ und ein Startwert $\mathbf{x}_0$ gegeben. In einem Schritt des Runge-Kutta-Verfahrens vierter Ordnung wird, ausgehend von einem bekannten Wert $\mathbf{x}_i$, mit Hilfe der folgenden Gleichungen der Wert $\mathbf{x}_{i+1}$

berechnet:

$$\begin{aligned}
\mathbf{k}_1 &= h\, f(t_i,\ \mathbf{x}_i) \\
\mathbf{k}_2 &= h\, f(t_i + \frac{1}{2}\, h,\ \mathbf{x}_i + \frac{1}{2}\, \mathbf{k}_1) \\
\mathbf{k}_3 &= h\, f(t_i + \frac{1}{2}\, h,\ \mathbf{x}_i + \frac{1}{2}\, \mathbf{k}_2) \\
\mathbf{k}_4 &= h\, f(t_i + h,\ \mathbf{x}_i + \mathbf{k}_3) \\
\mathbf{x}_{i+1} &= \mathbf{x}_i + \frac{1}{6}\, (\mathbf{k}_1 + 2\, \mathbf{k}_2 + 2\, \mathbf{k}_3 + \mathbf{k}_4)
\end{aligned}$$

Das Verfahren hat einen lokalen Fehler der Ordnung $O(h^5)$. Die einfache Berechnung und der geringe Fehler machen das Runge-Kutta-Verfahren vierter Ordnung zu einem geeigneten Verfahren für die dynamische Simulation.

B.1.4 Runge-Kutta-Verfahren mit adaptiver Schrittweite

Das Ziel bei der Verwendung einer adaptiven Schrittweite in einem Verfahren für die numerische Integration ist es, eine vorgegebene Genauigkeit mit minimalem Rechenaufwand zu erreichen [PFTV92]. Für eine adaptive Schrittweite benötigt man eine Fehlerabschätzung. Ein Zeitschritt wird zunächst mit einer Schrittweite h durchgeführt. Anschließend wird der aufgetretene Fehler abgeschätzt und eine neue Schrittweite h_{neu} bestimmt, bei der die vorgegebene Genauigkeit erreicht wird.

Eine Anpassung der Schrittweite beim Runge-Kutta-Verfahren kann durch die Verwendung der Formeln des *eingebetteten Runge-Kutta-Verfahrens* vorgenommen werden. Diese Formeln wurden ursprünglich von Erwin Fehlberg vorgestellt. Er verwendet ein Runge-Kutta-Verfahren fünfter Ordnung, bei dem die Ableitungsfunktion $f(t, \mathbf{x})$ sechsmal ausgewertet werden muss. Wenn die sechs Funktionswerte mit anderen Parametern kombiniert werden, ergibt sich ein Verfahren vierter Ordnung. Werden mit beiden Verfahren Werte für $\mathbf{x}(t_0 + h)$ bestimmt, dann kann der aufgetretene Fehler durch die Differenz der Werte abgeschätzt werden.

Die Gleichungen des Runge-Kutta-Verfahrens fünfter Ordnung haben die folgende allgemeine Form:

$$\begin{aligned}
\mathbf{k}_1 &= h\, f(t_i,\ \mathbf{x}_i) \\
\mathbf{k}_2 &= h\, f(t_i + a_2 h,\ \mathbf{x}_i + b_{21}\mathbf{k}_1) \\
\mathbf{k}_3 &= h\, f(t_i + a_3 h,\ \mathbf{x}_i + b_{31}\mathbf{k}_1 + b_{32}\mathbf{k}_2) \\
\mathbf{k}_4 &= h\, f(t_i + a_4 h,\ \mathbf{x}_i + b_{41}\mathbf{k}_1 + b_{42}\mathbf{k}_2 + b_{43}\mathbf{k}_3) \\
\mathbf{k}_5 &= h\, f(t_i + a_5 h,\ \mathbf{x}_i + b_{51}\mathbf{k}_1 + b_{52}\mathbf{k}_2 + b_{53}\mathbf{k}_3 + b_{54}\mathbf{k}_4) \\
\mathbf{k}_6 &= h\, f(t_i + a_6 h,\ \mathbf{x}_i + b_{61}\mathbf{k}_1 + b_{62}\mathbf{k}_2 + b_{63}\mathbf{k}_3 + b_{64}\mathbf{k}_4 + b_{65}\mathbf{k}_5)
\end{aligned}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + c_1\,\mathbf{k}_1 + c_2\,\mathbf{k}_2 + c_3\,\mathbf{k}_3 + c_4\,\mathbf{k}_4 + c_5\,\mathbf{k}_5 + c_6\,\mathbf{k}_6 + O(h^6).$$

Die Gleichung des Verfahrens vierter Ordnung wird mit den Parametern $c_1^*, \ldots, c_6^*$ aufgestellt:

$$\mathbf{x}_{i+1}^* = \mathbf{x}_i + c_1^*\,\mathbf{k}_1 + c_2^*\,\mathbf{k}_2 + c_3^*\,\mathbf{k}_3 + c_4^*\,\mathbf{k}_4 + c_5^*\,\mathbf{k}_5 + c_6^*\,\mathbf{k}_6 + O(h^5).$$

Jeff R. Cash and Alan H. Karp verwenden in [CK90] für die Parameter a_i, b_{ij}, c_i und c_i^* die Werte aus Tabelle B.1.

i	a_i	b_{ij}					c_i	c_i^*
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
$j =$		1	2	3	4	5		

Tabelle B.1: Parameter für das eingebettete Runge-Kutta-Verfahren

Der Fehler kann durch den Term

$$\triangle\mathbf{x} = \mathbf{x}_{i+1} - \mathbf{x}_{i+1}^* = \sum_{i=1}^{6}(c_i - c_i^*) \cdot \mathbf{k}_i$$

abgeschätzt werden. Da der Fehler von der Ordnung fünf ist, kann die neue Schrittweite durch

$$h_{neu} = h\left(\frac{\varepsilon}{|\triangle\mathbf{x}|}\right)^{\frac{1}{5}}$$

berechnet werden, wobei der Wert ε die gewünschte Genauigkeit angibt. Wenn der Betrag des Fehlers $|\triangle\mathbf{x}|$ größer ist als der Wert ε, dann wurde die gewünschte Genauigkeit nicht erreicht. Der letzte Schritt muss daher rückgängig gemacht und mit der Schrittweite $h_{neu} < h$ erneut ausgeführt werden. Andernfalls ist die neue Schrittweite h_{neu} größer als h. In diesem Fall wird die aktuelle Schrittweite auf den Wert h_{neu} erhöht und mit dem nächsten Schritt fortgefahren. Anstatt eine gewünschte Genauigkeit für den Betrag des Fehlers anzugeben, kann alternativ ein Vektor mit einer gewünschten Genauigkeit für jede Differentialgleichung verwendet werden. Die neue Schrittweite muss dann anhand der Differentialgleichung bestimmt werden, bei der das Verhältnis zwischen der gewünschten Genauigkeit und dem Fehler am schlechtesten ist.

Anhang C

Matrizen

C.1 Kreuzproduktmatrix

Sind die Vektoren $\mathbf{a}$ und $\mathbf{b}$ linear unabhängig, dann ist das Vektorprodukt $\mathbf{a} \times \mathbf{b}$ definiert als der Vektor mit den folgenden drei Eigenschaften:

- $\mathbf{a} \times \mathbf{b}$ ist senkrecht zu $\mathbf{a}$ und $\mathbf{b}$,
- $\mathbf{a}$, $\mathbf{b}$ und $\mathbf{a} \times \mathbf{b}$ bilden in der gegebenen Reihenfolge ein Rechtssystem und
- für den Winkel φ zwischen $\mathbf{a}$ und $\mathbf{b}$ gilt $|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}|\,|\mathbf{b}| \sin \varphi$.

Das Kreuzprodukt berechnet sich wie folgt:

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}.$$

Das Kreuzprodukt $\mathbf{a} \times \mathbf{b}$ zwischen zwei Vektoren kann als Matrixmultiplikation geschrieben werden, indem für den Vektor $\mathbf{a}$ die Kreuzproduktmatrix $\mathbf{a}^*$ gebildet wird:

$$\mathbf{a} \times \mathbf{b} = \mathbf{a}^* \cdot \mathbf{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \cdot \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$$

Das Kreuzprodukt hat außerdem folgende Eigenschaften:

- $\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$,
- $\mathbf{a} \times \mathbf{b} = 0 \Leftrightarrow \mathbf{a}$ und $\mathbf{b}$ sind linear abhängig,

- $(\lambda \cdot \mathbf{a}) \times \mathbf{b} = \mathbf{a} \times (\lambda \cdot \mathbf{b}) = \lambda \cdot (\mathbf{a} \times \mathbf{b})$,
- $\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$ und
- $(\mathbf{a} + \mathbf{b}) \times \mathbf{c} = \mathbf{a} \times \mathbf{c} + \mathbf{b} \times \mathbf{c}$.

C.2 Berechnung der Rotationsmatrix aus drei Eulerwinkeln

Eine Rotation kann durch die drei Eulerwinkel α, β und γ definiert werden. Die drei Winkel beschreiben dabei jeweils eine Drehung um die x-, y- und z-Achse, die hintereinander ausgeführt werden. Die orthonormale Rotationsmatrix $\mathbf{R}$ kann bestimmt werden, indem zunächst für die Drehung um jede Achse eine Matrix berechnet wird:

$$\begin{aligned}
\mathbf{R}_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \\
\mathbf{R}_y &= \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \\
\mathbf{R}_z &= \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}.
\end{aligned}$$

Anschließend werden diese Matrizen multipliziert, um die gesamte Drehung zu erhalten:

$$\mathbf{R} = \mathbf{R}_x \cdot \mathbf{R}_y \cdot \mathbf{R}_z.$$

Daraus ergibt sich dann die folgende Rotationsmatrix:

$$\mathbf{R} = \begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta \\ \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & \cos\alpha\sin\beta\sin\gamma + \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{pmatrix}.$$

C.3 Orthonormalisieren einer Rotationsmatrix

Eine Rotationsmatrix $\mathbf{R}$, die aufgrund von numerischen Ungenauigkeiten nicht mehr orthonormal ist, kann mit Hilfe des folgenden Algorithmus orthonormalisiert werden:

orthonormalize()

Eingabe: Rotationsmatrix $\mathbf{R} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix}$

// Orthogonalisieren
repeat
$\mathbf{r}_{23} := \mathbf{r}_2 \times \mathbf{r}_3$
$\mathbf{r}_{31} := \mathbf{r}_3 \times \mathbf{r}_1$
$\mathbf{r}_{12} := \mathbf{r}_1 \times \mathbf{r}_2$
$\mathbf{r}_1 := 0{,}5\,(\mathbf{r}_1 + \mathbf{r}_{23})$
$\mathbf{r}_2 := 0{,}5\,(\mathbf{r}_2 + \mathbf{r}_{31})$
$\mathbf{r}_3 := 0{,}5\,(\mathbf{r}_3 + \mathbf{r}_{12})$
until $\mathbf{r}_1$, $\mathbf{r}_2$ und $\mathbf{r}_3$ bilden ein Orthogonalsystem (innerhalb einer kleinen Toleranz)

// Normieren
$\mathbf{r}_1 := \mathbf{r}_1 / |\mathbf{r}_1|$
$\mathbf{r}_2 := \mathbf{r}_2 / |\mathbf{r}_2|$
$\mathbf{r}_3 := \mathbf{r}_3 / |\mathbf{r}_3|$

Algorithmus C.1: Orthonormalisieren einer Rotationsmatrix

C.4 Inverse einer 3x3-Matrix

Gegeben sei eine beliebige nicht singuläre Matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$:

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}.$$

Die Inverse dieser Matrix hat die folgende Form:

$$\mathbf{A}^{-1} = \frac{1}{a(fh-ei) + b(di-fg) + c(eg-dh)} \begin{pmatrix} fh-ei & bi-ch & ce-bf \\ di-fg & cg-ai & af-cd \\ eg-dh & ah-bg & bd-ae \end{pmatrix}$$

Es werden 9 Multiplikationen, eine Division und 5 Additionen für den Term vor der Matrix benötigt und 18 Multiplikationen und 9 Additionen in der Matrix. Wenn man die Terme $(fh-ei)$, $(di-fg)$ und $(eg-dh)$ vorberechnet, dann lassen sich weitere 6 Multiplikationen und 3 Additionen einsparen. Insgesamt benötigt man also 21 Multiplikationen, 11 Additionen und eine Division.

C.5 Inverse einer symmetrischen 3x3-Matrix

Gegeben sei eine beliebige nicht singuläre, symmetrische Matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$:

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

Die Inverse dieser Matrix ist ebenfalls symmetrisch und hat die folgende Form:

$$\mathbf{A}^{-1} = \frac{1}{b\,(fb - 2ce) + c^2 d + a\,(e^2 - df)} \begin{pmatrix} e^2 - df & bf - ce & cd - be \\ bf - ce & c^2 - af & ae - bc \\ cd - be & ae - bc & b^2 - ad \end{pmatrix}$$

Es werden 9 Multiplikationen, eine Division und 4 Additionen für den Term vor der Matrix benötigt und 12 Multiplikationen und 6 Additionen in der Matrix, da sie symmetrisch ist. Wenn man die benötigten Produkte für die Variablenpaare und den Term $e^2 - df$ vorberechnet, lassen sich weitere 5 Multiplikationen und eine Addition einsparen. Insgesamt benötigt man also 16 Multiplikationen, 10 Additionen und eine Division.

Anhang D

Grundlagen der Starrkörpersimulation

D.1 Trägheitstensor

Ein Trägheitstensor $\mathbf{J} \in \mathbb{R}^{3\times3}$ beschreibt die Beziehung zwischen der Winkelgeschwindigkeit $\boldsymbol{\omega}$ eines Körpers und seinem Drehimpuls:

$$\mathbf{l} = \mathbf{J}\,\boldsymbol{\omega}.$$

Damit gibt er die Trägheit des Körpers bezüglich Rotationen an. Sei ein Körper durch n Massenpunkte mit den Massen m_i gegeben und seien weiterhin $\mathbf{r}_i = (x_i \quad y_i \quad z_i)^T$ die Ortsvektoren der Massenpunkte in einem kartesischen Koordinatensystem, das seinen Ursprung im Schwerpunkt des Körpers hat. Dann berechnet sich der Trägheitstensor wie folgt:

$$\mathbf{J} = \sum_{i=1}^{n} m_i \begin{pmatrix} y_i^2 + z_i^2 & -x_i\,y_i & -x_i\,z_i \\ -y_i\,x_i & x_i^2 + z_i^2 & -y_i\,z_i \\ -z_i\,x_i & -z_i\,y_i & x_i^2 + y_i^2 \end{pmatrix}.$$

Der Trägheitstensor ist symmetrisch und für die Komponenten der Matrix gilt:

$$J_{i,j} = J_{j,i} = \sum_{k=1}^{n} m_k \left(\mathbf{r}_k^2\,\delta_{i,j} - r_{k,i}\,r_{k,j}\right)$$

wobei $\delta_{i,j}$ das Kronecker-Symbol ist, das folgendermaßen definiert ist:

$$\delta_{i,j} = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst.} \end{cases}$$

Bei einer kontinuierlichen Massenverteilung im Körper können die Komponenten durch Integration berechnet werden, wenn die Dichte $\rho(\mathbf{r})$ bekannt ist:

$$\mathbf{J}_{i,j} = \int_V \rho(\mathbf{r})(\mathbf{r}^2\,\delta_{i,j} - r_i\,r_j)\,dV.$$

Der Trägheitstensor kann durch eine orthogonale Transformation in Diagonalform gebracht werden, da die Matrix reell und symmetrisch ist. Das bedeutet, dass ein Koordinatensystem gefunden werden kann, in dem nur die Diagonalelemente ungleich Null sind. Diese Diagonalelemente bezeichnet man als Hauptträgheitsmomente und die zugehörigen Koordinatenachsen als Hauptträgheitsachsen.

Bei Körpern mit konstanter Dichte kann die Dichte vor das Integral gezogen werden. Dadurch kann der Trägheitstensor im Hauptachsensystem für einige spezielle Formen direkt berechnet werden:

- Trägheitstensor einer Kugel mit Radius r:

$$\mathbf{J}_{Kugel} = m \cdot \begin{pmatrix} \frac{2}{5}r^2 & 0 & 0 \\ 0 & \frac{2}{5}r^2 & 0 \\ 0 & 0 & \frac{2}{5}r^2 \end{pmatrix}$$

- Trägheitstensor eines Zylinders mit Radius r und Länge l:

$$\mathbf{J}_{Zylinder} = m \cdot \begin{pmatrix} \frac{r^2}{4} + \frac{l^2}{12} & 0 & 0 \\ 0 & \frac{r^2}{4} + \frac{l^2}{12} & 0 \\ 0 & 0 & \frac{r^2}{2} \end{pmatrix}$$

- Trägheitstensor eines Quaders mit Kantenlängen a, b, c:

$$\mathbf{J}_{Quader} = m \cdot \begin{pmatrix} \frac{b^2+c^2}{12} & 0 & 0 \\ 0 & \frac{a^2+c^2}{12} & 0 \\ 0 & 0 & \frac{a^2+b^2}{12} \end{pmatrix}$$

Bei einem Polyeder mit konstanter Dichte kann der Trägheitstensor mit Hilfe des Verfahrens von Brian Mirtich [Mir96a] bestimmt werden.

D.2 Winkelbeschleunigung

Die Winkelbeschleunigung $\dot{\boldsymbol{\omega}}$ wird in der dynamischen Simulation benötigt, um die Winkelgeschwindigkeit $\boldsymbol{\omega}$ zu berechnen. Für den Drehimpuls gilt $\mathbf{l} = \mathbf{J}\,\boldsymbol{\omega}$. Daher

gilt für die Winkelgeschwindigkeit $\boldsymbol{\omega} = \mathbf{J}^{-1}\,\mathbf{l}$. Durch Ableitung erhält man die Winkelbeschleunigung:

$$\dot{\boldsymbol{\omega}} = \dot{\mathbf{J}}^{-1}\,\mathbf{l} + \mathbf{J}^{-1}\,\dot{\mathbf{l}}, \tag{D.1}$$

wobei $\dot{\mathbf{l}} = \boldsymbol{\tau}$ das Drehmoment ist. Da $\mathbf{J}^{-1} = \mathbf{R}^T\,\mathbf{J}_l^{-1}\,\mathbf{R}$ gilt und der Trägheitstensor in lokalen Koordinaten konstant ist, gilt für die Ableitung des inversen Trägheitstensors:

$$\dot{\mathbf{J}}^{-1} = \dot{\mathbf{R}}^T\,\mathbf{J}_l^{-1}\,\mathbf{R} + \mathbf{R}^T\,\mathbf{J}_l^{-1}\,\dot{\mathbf{R}}. \tag{D.2}$$

Für die Ableitung der Rotationsmatrix gilt

$$\begin{aligned}\dot{\mathbf{R}} &= \boldsymbol{\omega}^*\,\mathbf{R}\\ \dot{\mathbf{R}}^T &= \boldsymbol{\omega}^*\,\mathbf{R}^T,\end{aligned}$$

wobei $\boldsymbol{\omega}^*$ die Kreuzproduktmatrix (siehe Anhang C.1) der Winkelgeschwindigkeit ist. Die Kreuzproduktmatrix ist antisymmetrisch, daher gilt ebenfalls

$$\dot{\mathbf{R}} = -\mathbf{R}\,\boldsymbol{\omega}^*.$$

Durch Einsetzen in Gleichung D.2 erhält man

$$\begin{aligned}\dot{\mathbf{J}}^{-1} &= \boldsymbol{\omega}^*\,\mathbf{R}^T\,\mathbf{J}_l^{-1}\,\mathbf{R} - \mathbf{R}^T\,\mathbf{J}_l^{-1}\,\mathbf{R}\,\boldsymbol{\omega}^*\\ &= \boldsymbol{\omega}^*\,\mathbf{J}^{-1} - \mathbf{J}^{-1}\,\boldsymbol{\omega}^*.\end{aligned}$$

Dies kann wiederum in Gleichung D.1 eingesetzt werden

$$\begin{aligned}\dot{\boldsymbol{\omega}} &= (\boldsymbol{\omega}^*\,\mathbf{J}^{-1} - \mathbf{J}^{-1}\,\boldsymbol{\omega}^*)\,\mathbf{l} + \mathbf{J}^{-1}\,\boldsymbol{\tau}\\ &= \boldsymbol{\omega}^*\,\mathbf{J}^{-1}\,\mathbf{l} - \mathbf{J}^{-1}\,\boldsymbol{\omega}^*\,\mathbf{l} + \mathbf{J}^{-1}\,\boldsymbol{\tau}\\ &= \boldsymbol{\omega}^*\,\mathbf{J}^{-1}\,\mathbf{J}\,\boldsymbol{\omega} + \mathbf{J}^{-1}\,(\boldsymbol{\tau} - \boldsymbol{\omega}^*\,\mathbf{l})\\ &= \boldsymbol{\omega}^*\,\boldsymbol{\omega} + \mathbf{J}^{-1}\,(\boldsymbol{\tau} - (\boldsymbol{\omega} \times \mathbf{l}))\\ &= \mathbf{J}^{-1}\,(\boldsymbol{\tau} - (\boldsymbol{\omega} \times (\mathbf{J}\,\boldsymbol{\omega}))),\end{aligned}$$

was zu der Euler-Gleichung führt. Anhand dieser Gleichung lässt sich feststellen, dass die Winkelbeschleunigung $\dot{\boldsymbol{\omega}}$ ungleich Null sein kann, selbst wenn das Drehmoment $\boldsymbol{\tau}$, das auf den Körper wirkt, Null ist. Dies ist der Fall, wenn die Drehachse der Winkelgeschwindigkeit keine Symmetrieachse des Körpers ist.

D.3 Reduktion der Distanz

In Abschnitt 3.2.3 werden die Positionsbedingungen aller Grundgelenke in einem iterativen Prozess mit Hilfe von Impulsen erfüllt. In jeder Iteration wird der Fehler $\mathbf{d}$ nach dem Simulationsschritt bestimmt. Dann wird die zur Korrektur benötigte Geschwindigkeitsänderung der verbundenen Körper durch $1/h \cdot \mathbf{d}$ approximiert.

Schließlich wird ein Impuls, der genau diese Änderung herbeiführt, ermittelt und auf die Körper angewendet. Dadurch ergibt sich ein neuer Fehler $\mathbf{d}'$ nach dem Simulationsschritt.

Wenn der neue Fehler in jeder Iteration kleiner ist als der vorherige, dann konvergiert der Prozess und liefert den Impuls, der dafür sorgt, dass die Positionsbedingung erfüllt wird. Daher wird im Folgenden für ein Kugelgelenk bewiesen, dass immer eine Zeitschrittweite $h > 0$ existiert, für die $|\mathbf{d}'| < |\mathbf{d}|$ gilt. Die Beweise für die anderen Grundgelenke funktionieren analog.

Beweis: Angenommen zwei Körper sind durch ein Kugelgelenk miteinander verbunden. Seien $\mathbf{a}$ und $\mathbf{b}$ die zugehörigen Gelenkpunkte der beiden Körper. Es wird weiterhin angenommen, dass die Gelenkbedingung am Anfang des Simulationsschrittes erfüllt ist. Daher haben $\mathbf{a}$ und $\mathbf{b}$ zum Zeitpunkt t_0 die gleiche Position. Der Abstandsvektor der beiden Punkte nach einem Zeitschritt der Länge h kann mit Hilfe von Taylor-Reihen beschrieben werden:

$$\mathbf{d} = (\mathbf{v}_2 - \mathbf{v}_1)\, h + \triangle \mathbf{F}_{\mathbf{ext}} + \sum_{i=1}^{\infty} (\mathbf{r}_b^{(i)} - \mathbf{r}_a^{(i)})\, \frac{h^i}{i\,!}, \tag{D.3}$$

wobei $\mathbf{r}_a$ und $\mathbf{r}_b$ die Ortsvektoren der Gelenkpunkte bezüglich des jeweiligen lokalen Koordinatensystems sind. Die Differenz der externen Kräfte, die auf beide Körper wirken, ist wie folgt definiert:

$$\triangle \mathbf{F}_{\mathbf{ext}} := \frac{1}{2\, m_2}\, \mathbf{F}_{ext,2}\, h^2 - \frac{1}{2\, m_1}\, \mathbf{F}_{ext,1}\, h^2.$$

Durch die Anwendung des Impulses

$$\mathbf{p} = \frac{1}{h}\, \mathbf{K}^{-1}\, \mathbf{d}$$

soll der Abstand $|\mathbf{d}|$ reduziert werden. Dieser Impuls verändert die Schwerpunktgeschwindigkeiten der Körper:

$$\begin{aligned} \mathbf{v}_1' &= \mathbf{v}_1 + \frac{1}{m_1}\mathbf{p} \\ \mathbf{v}_2' &= \mathbf{v}_2 - \frac{1}{m_2}\mathbf{p}. \end{aligned}$$

Außerdem ändern sich durch den Impuls die Ableitungen der Ortsvektoren nach der Zeit:

$$\begin{aligned} \dot{\mathbf{r}}_a' &= (\boldsymbol{\omega}_1 + \mathbf{J}_1^{-1}(\mathbf{r}_a \times \mathbf{p})) \times \mathbf{r}_a \\ \dot{\mathbf{r}}_b' &= (\boldsymbol{\omega}_2 + \mathbf{J}_2^{-1}(\mathbf{r}_b \times (-\mathbf{p}))) \times \mathbf{r}_b. \end{aligned}$$

Wenn die neuen Geschwindigkeiten in der Gleichung D.3 des Abstandsvektors nach dem Zeitschritt eingesetzt werden, dann ergibt sich

$$\begin{aligned}
\mathbf{d}' &= \left(\mathbf{v}_2 - \frac{1}{m_2}\,\mathbf{p} - \mathbf{v}_1 - \frac{1}{m_1}\,\mathbf{p}\right) h + \triangle\mathbf{F_{ext}} + \sum_{i=1}^{\infty}(\mathbf{r}_b'^{(i)} - \mathbf{r}_a'^{(i)})\,\frac{h^i}{i\,!} \\
&= -\mathbf{K}\,\mathbf{p}\,h + (\mathbf{v}_2 - \mathbf{v}_1)\,h + \triangle\mathbf{F_{ext}} + (\dot{\mathbf{r}}_b - \dot{\mathbf{r}}_a)\,h + \sum_{i=2}^{\infty}(\mathbf{r}_b'^{(i)} - \mathbf{r}_a'^{(i)})\,\frac{h^i}{i\,!} \\
&= -\mathbf{d} + (\mathbf{v}_2 - \mathbf{v}_1)\,h + \triangle\mathbf{F_{ext}} + (\dot{\mathbf{r}}_b - \dot{\mathbf{r}}_a)\,h + \sum_{i=2}^{\infty}(\mathbf{r}_b'^{(i)} - \mathbf{r}_a'^{(i)})\,\frac{h^i}{i\,!} \\
&= \sum_{i=2}^{\infty}(\mathbf{r}_b'^{(i)} - \mathbf{r}_a'^{(i)} - \mathbf{r}_b^{(i)} + \mathbf{r}_a^{(i)})\,\frac{h^i}{i\,!}.
\end{aligned}$$

Der neue Abstand $|\mathbf{d}'|$ ist demnach abhängig von h^2, während der ursprüngliche Wert $|\mathbf{d}|$ von h abhängt. Aus diesem Grund existiert ein $h > 0$, für das $|\mathbf{d}'| < |\mathbf{d}|$ gilt.

□

Anhang E

Notation

In Tabelle E.1 sind die wichtigsten mathematischen Bezeichnungen aufgeführt, die in dieser Arbeit verwendet werden. Matrizen und Vektoren werden im Gegensatz zu skalaren Größen durch fett gedruckte Bezeichner repräsentiert.

Bezeichner	Beschreibung
$\mathbf{E}_n$	n-dimensionale Einheitsmatrix
m	Masse
$\mathbf{s}$	Schwerpunkt
$\mathbf{v}$	Geschwindigkeit des Schwerpunkts
$\mathbf{u}$	Punktgeschwindigkeit
$\mathbf{J}$	Trägheitstensor
$\mathbf{R}$	Rotationsmatrix
$\mathbf{q}$	Quaternion
$\boldsymbol{\omega}$	Winkelgeschwindigkeit
$\mathbf{r}_{ab}$	Ortsvektor von Punkt $\mathbf{a}$ nach Punkt $\mathbf{b}$
$\mathbf{g}$	Gravitation
$\mathbf{F}_{ext}$	Externe Kraft
$\boldsymbol{\tau}_{ext}$	Externes Drehmoment
$\mathbf{p}$	Impuls
$\mathbf{l}$	Drehimpuls
h	Zeitschrittweite

Tabelle E.1: Notation

Stichwortverzeichnis

Literaturverzeichnis

[ABB+99] Edward Anderson, Zhaojun Bai, Christian H. Bischof, Susan Blackford, James W. Demmel, Jack J. Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney und Danny Sorensen: *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 3. Auflage, 1999.

[ACPR95] Uri M. Ascher, Hongsheng Chin, Linda R. Petzold und Sebastian Reich: *Stabilization of Constrained Mechanical Systems with DAEs and Invariant Manifolds*. Journal of Mechanics of Structures and Machines, 23:135–158, 1995.

[APS98] Mihai Anitescu, Florian A. Potra und David E. Stewart: *Time-stepping for three-dimensional rigid body dynamics*. Computer Methods in Applied Mechanics and Engineering, 177(3-4):183–197, 1998.

[Bar89] David Baraff: *Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies*. Computer Graphics, 23(3):223–232, 1989.

[Bar90] David Baraff: *Curved surfaces and coherence for non-penetrating rigid body simulation*. SIGGRAPH Computer Graphics, 24(4):19–28, 1990.

[Bar91] David Baraff: *Coping with friction for non-penetrating rigid body simulation*. In: *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, Seiten 31–41, New York, NY, USA, 1991. ACM Press.

[Bar92] David Baraff: *Dynamic simulation of nonpenetrating rigid bodies*. Doktorarbeit, Department of Computer Science, Cornell University, Ithaca, NY, USA, 1992.

[Bar93] David Baraff: *Issues in computing contact forces for non-penetrating rigid bodies*. Algorithmica, 10(2-4):292–352, 1993.

[Bar94] David Baraff: *Fast contact force computation for nonpenetrating rigid bodies.* In: *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, Seiten 23–34, New York, NY, USA, 1994. ACM Press.

[Bar96] David Baraff: *Linear-time dynamics using Lagrange multipliers.* In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, Seiten 137–146, New York, NY, USA, 1996. ACM Press.

[Bau72] Joachim W. Baumgarte: *Stabilization of constraints and integrals of motion in dynamical systems.* Computer Methods in Applied Mechanics and Engineering, 1:1–16, 1972.

[BB88] Ronen Barzel und Alan H. Barr: *A modeling system based on dynamic constraints.* In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, Seiten 179–188, New York, NY, USA, 1988. ACM Press.

[BBS03] Jan Bender, Matthias Baas und Alfred Schmitt: *Ein neues Verfahren für die mechanische Simulation in VR-Systemen und in der Robotik.* In: *17. Symposium Simulationstechnik, ASIM 2003*, Seiten 111–116, 2003.

[BFS05] Jan Bender, Dieter Finkenzeller und Alfred Schmitt: *An impulse-based dynamic simulation system for VR applications.* In: *Proceedings of Virtual Concept 2005*, Biarritz, France, 2005. Springer.

[Bon86] Hermann Bondi: *The Rigid Body Dynamics of Unidirectional Spin.* Royal Society of London Proceedings Series A, 405:265–274, Juni 1986.

[BW00] George Baciu und Sai K. Wong: *The Impulse Graph: A New Dynamic Structure For Global Collisions.* Computer Graphics Forum, 19(3):229–238, 2000.

[Cam97a] Stephen Cameron: *A comparison of two fast algorithms for computing the distance between convex polyhedra.* IEEE Transactions on Robotics and Automation, 13(6):915–920, 1997.

[Cam97b] Stephen Cameron: *Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra.* In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Seiten 3112–3117, April 1997.

[CD68] Richard W. Cottle und George B. Dantzig: *Complementary Pivot Theory of Mathematical Programming.* Linear Algebra and its Applications, 1:103–125, 1968.

[CK90] Jeff R. Cash und Alan H. Karp: *A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides.* ACM Transactions on Mathematical Software, 16(3):201–222, 1990.

[CLMP95] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha und Madhav K. Ponamgi: *I-COLLIDE: an interactive and exact collision detection system for large-scale environments.* In: *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, Seiten 189–ff., New York, NY, USA, 1995. ACM Press.

[CML+94] Jonathan D. Cohen, Dinesh Manocha, Ming C. Lin, und Madhav K. Ponamgi: *Interactive and Exact Collision Detection for Large-Scaled Environments.* Technischer Bericht TR94-005, Department of Computer Science, University of North Carolina, Chapel Hill, 1994.

[CPS92] Richard W. Cottle, Jong-Shi Pang und Richard E. Stone: *The linear complementarity problem.* Academic Press, 1992.

[CR98] Anindya Chatterjee und Andy Ruina: *A New Algebraic Rigid Body Collision Law Based On Impulse Space Considerations.* Journal of Applied Mechanics, 65(4):939–951, 1998.

[dJB94] Javier Garcia de Jalon und Eduardo Bayo: *Kinematic and Dynamic Simulation of Multibody Systems: the Real Time Challenge.* Springer-Verlag, New York, 1994.

[DLLP03] Laurent Dupont, Daniel Lazard, Sylvain Lazard und Sylvain Petitjean: *Near-optimal parameterization of the intersection of quadrics.* In: *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, Seiten 246–255, New York, NY, USA, 2003. ACM Press.

[EL00a] Stephen A. Ehmann und Ming C. Lin: *Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching.* In: *Proceedings of the International Conference on Intelligent Robots and Systems*, 2000.

[EL00b] Stephen A. Ehmann und Ming C. Lin: *SWIFT: Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi*

Marching. Technischer Bericht, Department of Computer Science, University of North Carolina, Chapel Hill, 2000.

[EL01] Stephan A. Ehmann und Ming C. Lin: *Accurate and Fast Proximity Queries Between Polyhedra Using Convex Surface Decomposition*. In: *Proceedings of Eurographics*, Band 20, Seiten 500–510, 2001.

[Fea87] Roy Featherstone: *Robot Dynamics Algorithm*. Kluwer Academic Publishers, Norwell, MA, USA, 1987. Manufactured By-Kluwer Academic Publishers.

[FO00] Roy Featherstone und David Orin: *Robot Dynamics: Equations and Algorithms*. International Conference on Robotics and Automation, Seiten 826–834, 2000.

[GBF03] Eran Guendelman, Robert Bridson und Ronald Fedkiw: *Nonconvex rigid bodies with stacking*. ACM Transactions on Graphics, 22(3):871–878, Juli 2003.

[GGK06] Alexander Gress, Michael Guthe und Reinhard Klein: *GPU-based Collision Detection for Deformable Parameterized Surfaces*. Computer Graphics Forum, 25(3):497–506, September 2006.

[GH88] A. Garcia und M. Hubbard: *Spin Reversal of the Rattleback: Theory and Experiment*. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 418(1854):165–197, 1988.

[GJK88] Elmer G. Gilbert, Daniel W. Johnson und Sathiya S. Keerth: *A fast procedure for computing the distance between complex objects in three-dimensional space*. IEEE Journal of Robotics and Automation, RA-4(2):193–203, April 1988.

[GLM96] Stefan Gottschalk, Ming C. Lin und Dinesh Manocha: *OBBTree: a hierarchical structure for rapid interference detection*. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, Seiten 171–180, New York, NY, USA, 1996. ACM Press.

[Got00] Stefan Gottschalk: *Collision queries using oriented bounding boxes*. Doktorarbeit, Department of Computer Science, University of North Carolina, Chapel Hill, 2000.

[GPS06] Herbert Goldstein, Charles P. Poole und John L. Safko: *Klassische Mechanik*. Wiley-VCH, 3., vollständig überarbeitete und erweiterte Auflage, 2006.

[GT00] Nicholas I. M. Gould und Philippe L. Toint: *A Quadratic Programming Bibliography*. Technischer Bericht 2000-1, Rutherford Appleton Laboratory, Chilton, England, 2000.

[Gue06] Eran Guendelman: *Physically-based simulation of soldis and solid-fluid coupling*. Doktorarbeit, Department of Computer Science, Stanford University, California, Juni 2006.

[Hah88] James K. Hahn: *Realistic animation of rigid bodies*. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, Seiten 299–308, New York, NY, USA, 1988. ACM Press.

[Ham53] William R. Hamilton: *Lectures on Quaternions*. Royal Irish Academy, 1853.

[HKM96] Martin Held, James T. Klosowski und Joseph S. B. Mitchell: *Real-time collision detection for motion simulation within complex environments*. In: *SIGGRAPH '96: ACM SIGGRAPH 96 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '96*, Seite 151, New York, NY, USA, 1996. ACM Press.

[Hof89] Christoph M. Hoffmann: *Geometric and solid modeling: an introduction*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[Hub95] Philip M. Hubbard: *Collision Detection for Interactive Graphics Applications*. IEEE Transactions on Visualization and Computer Graphics, 1(3):218–230, 1995.

[KEP05] Danny M. Kaufman, Timothy Edmunds und Dinesh K. Pai: *Fast frictional dynamics for rigid bodies*. ACM Trans. Graph., 24(3):946–956, 2005.

[KHM+98] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral und Karel Zikan: *Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs*. IEEE Transactions on Visualization and Computer Graphics, 4(1):21–36, März 1998.

[KKM97] John Keyser, Shankar Krishnan und Dinesh Manocha: *Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic*. In: *SMA '97: Proceedings of the fourth ACM symposium on Solid modeling and applications*, Seiten 42–55, New York, NY, USA, 1997. ACM Press.

[Klo98] James T. Klosowski: *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments*. Doktorarbeit, State University of New York at Stony Brook, Mai 1998.

[KP03] Paul G. Kry und Dinesh K. Pai: *Continuous contact simulation for smooth surfaces*. ACM Transactions on Graphics, 22(1):106–129, 2003.

[KR03] Byungmoon Kim und Jarek Rossignac: *Collision prediction for polyhedra under screw motions*. In: *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, Seiten 4–10, New York, NY, USA, 2003. ACM Press.

[KSK97] Katsuaki Kawachi, Hiromasa Suzuki und Fumihiko Kimura: *Simulation of rigid body motion with impulsive friction force*. In: *IEEE International Symposium on Assembly and Task Planning*, Seiten 182–187, August 1997.

[KSK98] Katsuaki Kawachi, Hiromasa Suzuki und Fumihiko Kimura: *Technical Issues on Simulating Impulse and Friction in Three Dimensional Rigid Body Dynamics*. In: *CA '98: Proceedings of the Computer Animation*, Seite 170, Washington, DC, USA, Juni 1998. IEEE Computer Society.

[LC91] Ming C. Lin und John F. Canny: *A Fast Algorithm for Incremental Distance Calculation*. In: *IEEE International Conference on Robotics and Automation*, Seiten 1008–1014, 1991.

[LG98] Ming C. Lin und Stefan Gottschalk: *Collision Detection between Geometric Models: A Survey*. In: *Proceedings of IMA Conference on Mathematics of Surfaces*, Band 1, Seiten 602–608, 1998.

[LGLM99] Eric Larsen, Stefan Gottschalk, Ming C. Lin und Dinesh Manocha: *Fast Proximity Queries with Swept Sphere Volumes*. Technischer Bericht TR99-018, Department of Computer Science, University of North Carolina, Chapel Hill, 1999.

[Lin93] Ming C. Lin: *Efficient collision detection for animation and robotics*. Doktorarbeit, University of California, Berkeley, 1993. Chair-John F. Canny.

[LM97] Ming C. Lin und Dinesh Manocha: *Efficient contact determination between geometric models*. International Journal of Computational Geometry and Applications, 7(1):123–151, 1997.

[Löt82] Per Lötstedt: *Mechanical Systems of Rigid Bodies Subject to Unilateral Constraints.* SIAM Journal on Applied Mathematics, 42(2):281–296, 1982.

[Löt84] Per Lötstedt: *Numerical simulation of time-dependent contact and friction problems in rigid body mechanics.* SIAM Journal on Scientific Statistical Computing, 5(2):370–393, Juni 1984.

[Lun06] Jan Lunze: *Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen.* Springer, 5., neu bearbeitete und erweiterte Auflage, 2006.

[MC94] Brian V. Mirtich und John F. Canny: *Impulse-based dynamic simulation.* In: *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, Seiten 407–418, Natick, MA, USA, 1994. A. K. Peters, Ltd.

[MC95] Brian V. Mirtich und John F. Canny: *Impulse-based simulation of rigid bodies.* In: *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, Seiten 181–ff., New York, NY, USA, 1995. ACM Press.

[MHHR06] Matthias Müller, Bruno Heidelberger, Marcus Hennix und John Ratcliff: *Position Based Dynamics.* In: *Workshop in Virtual Reality Interactions and Physical Simulations (VRIPHYS 2006)*, Madrid, November 2006.

[Mil96] Victor J. Milenkovic: *Position-based physics: simulating the motion of many highly interacting spheres and polyhedra.* In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, Seiten 129–136, New York, NY, USA, 1996. ACM Press.

[Mir95] Brian V. Mirtich: *Hybrid Simulation: Combining Constraints and Impulses.* Proceedings of the First Workshop on Simulation and Interaction in Virtual Environments, Juli 1995.

[Mir96a] Brian V. Mirtich: *Fast and accurate computation of polyhedral mass properties.* Journal of Graphics Tools: JGT, 1(2):31–50, 1996.

[Mir96b] Brian V. Mirtich: *Impulse-based dynamic simulation of rigid body systems.* Doktorarbeit, University of California, Berkeley, 1996.

[Mir97] Brian V. Mirtich: *Efficient algorithms for two–phase collision detection.* Technical Report TR–97–23, Mitsubishi Electric Research Laboratory, Dezember 1997.

[Mir98] Brian V. Mirtich: *V-Clip: fast and robust polyhedral collision detection.* ACM Transactions on Graphics, 17(3):177–208, 1998.

[MS01] Victor J. Milenkovic und Harald Schmidl: *Optimization-based animation.* In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, Seiten 37–46, New York, NY, USA, 2001. ACM Press.

[MW88a] Matthew T. Mason und Yu Wang: *On the Inconsistency of Rigid-Body Frictional Planar Mechanics.* In: *IEEE International Conference on Robotics and Automation*, Band 1, Seiten 524–528, April 1988.

[MW88b] Matthew Moore und Jane Wilhelms: *Collision detection and response for computer animation.* In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, Seiten 289–298, New York, NY, USA, 1988. ACM Press.

[Or94] A. C. Or: *The dynamics of a Tippe top.* SIAM Journal on Applied Mathematics, 54(3):597–609, 1994.

[PB88] John C. Platt und Alan H. Barr: *Constraints methods for flexible models.* In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, Seiten 279–288, New York, NY, USA, 1988. ACM Press.

[PFTV92] William H. Press, Brian P. Flannery, Saul A. Teukolsky und William T. Vetterling: *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge (UK) and New York, 2. Auflage, 1992.

[PG96] Friedrich Pfeiffer und Christoph Glocker: *Multibody Dynamics with Unilateral Contacts.* Wiley Series in Nonlinear Science. John Wiley and Sons, New York, 1996.

[PT95] Les Piegl und Wayne Tiller: *The NURBS book.* Springer-Verlag, London, UK, 1995.

[Qui94] Sean Quinlan: *Efficient Distance Computation between Non-Convex Objects.* In: *IEEE International Conference on Robotics and Automation*, Seiten 3324–3329. IEEE, 1994.

[RKC00] Stephane Redon, Abderrahmane Kheddar und Sabine Coquillart: *An algebraic solution to the problem of collision detection for rigid polyhedral objects.* In: *Proceedings of the IEEE International Conference on Robotics and Automation, 2000*, Band 4, Seiten 3733–3738, 2000.

[RKC02] Stephane Redon, Abderrahmane Kheddar und Sabine Coquillart: *Fast Continuous Collision Detection between Rigid Bodies*. Computer Graphics Forum, 21(3), 2002.

[RKLM04] Stephane Redon, Young J. Kim, Ming C. Lin und Dinesh Manocha: *Fast Continuous Collision Detection for Articulated Models*. In: *Proceedings of ACM Symposium on Solid Modeling and Applications*, 2004.

[RZ02] Manfred Reuter und Serge Zacher: *Regelungstechnik für Ingenieure*. Vieweg, 10., vollständig neu bearbeitete Auflage, 2002.

[SB05] Alfred Schmitt und Jan Bender: *Impulse-Based Dynamic Simulation of Multibody Systems: Numerical Comparison with Standard Methods*. In: *Proc. Automation of Discrete Production Engineering*, Seiten 324–329, 2005.

[SBP05a] Alfred Schmitt, Jan Bender und Hartmut Prautzsch: *Impulse-Based Dynamic Simulation of Higher Order and Numerical Results*. Internal Report 21, Institut für Betriebs- und Dialogsysteme, 2005.

[SBP05b] Alfred Schmitt, Jan Bender und Hartmut Prautzsch: *On the Convergence and Correctness of Impulse-Based Dynamic Simulation*. Internal Report 17, Institut für Betriebs- und Dialogsysteme, 2005.

[SG02] Olaf Schenk und Klaus Gärtner: *Two-level dynamic scheduling in PARDISO: improved scalability on shared memory multiprocessing systems*. Parallel Computing, 28(2):187–197, 2002.

[SG04a] Olaf Schenk und Klaus Gärtner: *On fast factorization pivoting methods for sparse symmetric indefinite systems*. Technical Report, Department of Computer Science, University of Basel, 2004.

[SG04b] Olaf Schenk und Klaus Gärtner: *Solving unsymmetric sparse systems of linear equations with PARDISO*. Future Generation Computer Systems, 20(3):475–487, 2004.

[SGFS01] Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner und Andreas Stricker: *PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation*. Future Generation Computer Systems, 18(1):69–78, 2001.

[Sho85] Ken Shoemake: *Animating rotation with quaternion curves.* In: *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, Seiten 245–254. ACM Press, 1985.

[SJ91] Ching-Kuang Shene und John K. Johnstone: *On the planar intersection of natural quadrics.* In: *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, Seiten 233–242, New York, NY, USA, 1991. ACM Press.

[SM04] Harald Schmidl und Victor J. Milenkovic: *A Fast Impulsive Contact Suite for Rigid Body Simulation.* IEEE Transactions on Visualization and Computer Graphics, 10(2):189–197, 2004.

[SS98a] Jörg Sauer und Elmar Schömer: *A constraint-based approach to rigid body dynamics for virtual reality applications.* In: *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, Seiten 153–162, New York, NY, USA, 1998. ACM Press.

[SS98b] Jörg Sauer und Elmar Schömer: *Dynamiksimulation starrer Körper für Virtual Reality Anwendungen.* In: *12. Symposium Simulationstechnik, ASIM '98*, Seiten 355–362, 1998.

[SSL98] Jörg Sauer, Elmar Schömer und Christian Lennerz: *Real-Time Rigid Body Simulations Of Some 'Classical Mechanics Toys'.* In: *10th European Simulation Symposium and Exhibition, ESS '98*, Seiten 93–98, 1998.

[ST96] David E. Stewart und Jeff C. Trinkle: *An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction.* International Journal for Numerical Methods in Engineering, 39(15):2673–2691, 1996.

[ST97] David E. Stewart und Jeff C. Trinkle: *Dynamics, Friction, And Complementarity Problems.* In: *Proceedings of the International Conference on Complementarity Problems 1995*, Seiten 425–439, Philadelphia, 1997.

[Ste00] David E. Stewart: *Rigid-Body Dynamics with Friction and Impact.* SIAM Review, 42(1):3–39, 2000.

[vdB97] Gino van den Bergen: *Efficient collision detection of complex deformable models using AABB trees.* Journal of Graphics Tools, 2(4):1–13, 1997.

[vdB99] Gino van den Bergen: *A fast and robust GJK implementation for collision detection of convex objects.* Journal of Graphics Tools, 4(2):7–25, 1999.

[vdB01] Gino van den Bergen: *Proximity Queries and Penetration Depth Computation on 3D Game Objects.* Proceedings of the Game Developers Conference, Seiten 821–837, 2001.

[vdB04] Gino van den Bergen: *Collision detection in interactive 3D environments.* Morgan Kaufmann, 2004.

[Wag01] Friedrich Wagner: *Konzepte und Methoden zu allgemeinen, physikalisch basierten Animationssystemen auf der Grundlage der Lagrange-Faktoren-Methode.* Doktorarbeit, Universität Rostock, 2001.

[WGW90] Andrew Witkin, Michael Gleicher und William Welch: *Interactive dynamics.* In: *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, Seiten 11–21, New York, NY, USA, 1990. ACM Press.

[Wit77] Jens Wittenburg: *Dynamics of systems of rigid bodies.* Teubner, 1. Auflage, 1977.

[WLML98] Andy Wilson, Eric Larsen, Dinesh Manocha und Ming C. Lin: *IMMPACT: a system for interactive proximity queries on massive models.* Technischer Bericht TR98-031, Department of Computer Science, University of North Carolina, Chapel Hill, 1998.

[WLML99a] Andy Wilson, Eric Larsen, Dinesh Manocha und Ming C. Lin: *Graph partitioning and ordering for interactive proximity queries.* In: *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, Seiten 429–430, New York, NY, USA, 1999. ACM Press.

[WLML99b] Andy Wilson, Eric Larsen, Dinesh Manocha und Ming C. Lin: *Partitioning and Handling Massive Models for Interactive Collision Detection.* In: P. Brunet und R. Scopigno (Herausgeber): *Computer Graphics Forum (Eurographics '99)*, Band 18, Seiten 319–330. The Eurographics Association and Blackwell Publishers, 1999.

[WTF06] Rachel L. Weinstein, Joseph Teran und Ron Fedkiw: *Dynamic Simulation of Articulated Rigid Bodies with Contact and Collision.* In: *IEEE Transactions on Visualization and Computer Graphics*, Band 12, Seiten 365–374, 2006.

[WW90] Andrew Witkin und William Welch: *Fast animation and control of nonrigid structures*. In: *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, Seiten 243–252, New York, NY, USA, 1990. ACM Press.

[Zac00] Gabriel Zachmann: *Virtual Reality in Assembly Simulation — Collision Detection, Simulation Algorithms, and Interaction Techniques*. Doktorarbeit, Darmstadt University of Technology, Germany, Department of Computer Science, Mai 2000.